PNX15xx Series Data Book

Volume 1 of 1

Connected Media Processor

Rev. 2 — 1 December 2004

PHILIPS

Table of Contents

Chapter 1: Integrated Circuit Data

1. Introduction . . . 1-1
2. Pin Description . . . 1-1
2.1 Boundary Scan Notice . . . 1-1
2.2 I/O Circuit Summary . . . 1-1
2.3 Signal Pin List . . . 1-3
2.3.1 Power Pin List . . . 1-18
2.3.2 Pin Reference Voltage . . . 1-19
3. Parametric Characteristics . . . 1-19
3.1 Absolute Maximum Ratings . . . 1-19
3.2 Operating Range and Thermal Characteristics . . . 1-20
4. Power Supplies Sequence . . . 1-20
5. Power Supply and Operating Speeds . . . 1-21
6. Power Consumption . . . 1-21
6.1 Leakage current Power Consumption . . . 1-21
6.2 Standby Power Consumption . . . 1-21
6.3 Typical Power Consumption for Typical Applications 1-21
6.4 Expected Maximum Currents . . . 1-22
7. DC/AC I/O Characteristics . . . 1-22
7.1 Input Crystal Specification . . . 1-23
7.2 SSTL_2 type I/O Circuit . . . 1-23
7.3 BPX2T14MCP Type I/O Circuit . . . 1-25
7.4 BPTS1CHP and BPTS1CP Type I/O Circuit . . . 1-26
7.5 BPTS3CHP Type I/O Circuit . . . 1-27
7.6 IPCHP and IPCP Type I/O Circuit . . . 1-28
7.7 BPT3MCHDT5V and BPT3MCHT5V Type I/O Circuit 1-28
7.8 IIC3M4SDAT5V and IIC3M4SCLT5V type I/O circuit 1-29
7.9 PCIT5V type I/O circuit . . . 1-29
8. Timing Specification . . . 1-29
8.1 Reset . . . 1-30
8.2 DDR DRAM Interface . . . 1-30
8.3 PCI Bus Interface . . . 1-31
8.4 QVCP, LCD and FGPO Interfaces . . . 1-33
8.5 VIP and FGPI Interfaces . . . 1-34
8.6 10/100 LAN In MII Mode . . . 1-34
8.7 10/100 LAN In RMII Mode . . . 1-35
8.8 Audio Input Interface . . . 1-36
8.9 Audio Output Interface . . . 1-37
8.10 SPDIF I/O Interface . . . 1-38
8.11 I2C I/O Interface . . . 1-39
8.12 GPIO Interface . . . 1-40
8.13 JTAG Interface . . . 1-41
9. Package Outline . . . 1-42
10. Board Design Guidelines . . . 1-43
10.1 Power Supplies Decoupling . . . 1-43
10.2 Analog Supplies . . . 1-44
10.2.1 The 3.3 V Analog Supply . . . 1-44
10.2.2 The 1.2-1.3-V Analog Supply . . . 1-44
10.3 DDR SDRAM interface . . . 1-45
10.3.1 Do DDR Devices Require Termination? . . . 1-46
10.3.2 What if I really want to use termination for the PNX1500? 1-46
10.4 Package Handling, Soldering and Thermal Properties 1-46
11. Miscellaneous . . . 1-47
12. Soft Errors Due to Radiation . . . 1-47
13. Ordering Information . . . 1-47

Chapter 2: Overview

1. Introduction . . . 2-1
1.1 PNX15xx Series Functional Overview . . . 2-1
1.2 PNX15xx Series Features Summary . . . 2-3
2. PNX15xx Series Functional Block Diagram 2-5
3. System Resources . . . 2-6
3.1 System Reset . . . 2-6
3.2 System Booting . . . 2-6
3.3 Clock System . . . 2-7
3.4 Power Management . . . 2-7
3.5 Semaphores . . . 2-8
3.6 I2C Interface . . . 2-8
4. System Memory . . . 2-9
4.1 MMI - Main Memory Interface . . . 2-9
4.2 Flash . . . 2-9
5. TM3260 VLIW Media Processor Core . . . 2-10
6. MPEG Decoding . . . 2-12
6.1 VLD . . . 2-12
6.2 DVD De-scrambler . . . 2-12
7. Image Processing . . . 2-12
7.1 Pixel Format . . . 2-12
7.2 Video Input Processor . . . 2-14
7.3 Memory Based Scaler . . . 2-14
7.4 2D Drawing and DMA Engine . . . 2-15
7.5 Quality Video Composition Processor . . . 2-15
7.5.1 External Video Improvement Post Processing . 2-17
8. Audio processing and Input/Output . . . 2-17
8.1 Audio Processing . . . 2-17
8.2 Audio Inputs and Outputs . . . 2-17
9. General Purpose Interfaces . . . 2-18
9.1 Video/Data Input Router . . . 2-18
9.2 Video/Data Output Router . . . 2-19
9.3 Fast General Purpose Input . . . 2-20
9.4 Fast General Purpose Output . . . 2-21
10. Peripheral Interface . . . 2-21
10.1 GPIO - General Purpose Software I/O and Flexible Serial Interface 2-21
10.1.1 software I/O . . . 2-22

12NC 9397 750 14321

10.1.2 timestamping . . . 2-22
10.1.3 event sequence monitoring and signal generation 2-22
10.1.4 GPIO pin reset value . . . 2-23
10.2 IR Remote Control Receiver and Blaster . . . 2-23
10.3 PCI-2.2 & XIO-16 Bus Interface Unit . . . 2-23
10.3.1 PCI Capabilities . . . 2-24
10.3.2 Simple Peripheral Capabilities ('XIO-8/16') . . 2-24
10.3.3 IDE Drive Interface . . . 2-26
10.4 10/100 Ethernet MAC . . . 2-26
11. Endian Modes . . . 2-26
12. System Debug . . . 2-27

Chapter 3: System On Chip Resources

1. Introduction . . . 3-1
2. System Memory Map . . . 3-1
2.1 The PCI View . . . 3-2
2.2 The CPU View . . . 3-3
2.3 The DCS View Or The System View . . . 3-4
2.4 The Programmable DCS Apertures . . . 3-5
2.4.1 DCS DRAM Aperture Control MMIO Registers 3-6
2.5 Aperture Boundaries . . . 3-6
3. System Principles . . . 3-7
3.1 Module ID . . . 3-7
3.2 Powerdown bit . . . 3-7
3.3 System Module MMIO registers . . . 3-8
4. System Endian Mode . . . 3-8
4.1 System Endian Mode MMIO registers . . . 3-9
5. System Semaphores . . . 3-9
5.1 Semaphore Specification . . . 3-9
5.2 Construction of a 12-bit ID . . . 3-9
5.3 The Master Semaphore . . . 3-10
5.4 Usage Notes . . . 3-10
5.5 Semaphore MMIO Registers . . . 3-11
6. System Related Information for TM3260 3-12
6.1 Interrupts . . . 3-12
6.2 Timers . . . 3-14
6.3 System Parameters for TM3260 . . . 3-15
6.3.1 TM3260 System Parameters MMIO Registers . 3-16
7. Video Input and Output Routers . . . 3-16
7.1 MMIO Registers for the Input/Output Video/Data Router 3-17
8. Miscellaneous . . . 3-26
8.1 Miscellaneous System MMIO registers . . . 3-27
9. System Registers Map Summary . . . 3-29
10. Simplified Internal Bus Infrastructure . . 3-30
11. MMIO Memory MAP . . . 3-31
12. References . . . 3-32

Chapter 4: Reset

1. Introduction . . . 4-1
2. Functional Description . . . 4-1
2.1 RESET_IN_N or POR_IN_N? . . . 4-3
2.2 The watchdog Timer . . . 4-4
2.2.1 The Non Interrupt Mode . . . 4-4
2.2.2 The Interrupt Mode . . . 4-5
2.3 The Software Reset . . . 4-6
2.4 The External Software Reset . . . 4-6
3. Timing Description . . . 4-7
3.1 The Hardware Timing . . . 4-7
3.2 The Software Timing . . . 4-8
4. Register Definitions . . . 4-9
5. References . . . 4-10

Chapter 5: The Clock Module

1. Introduction . . . 5-1
2. Functional Description . . . 5-1
2.1 The Modules and their Clocks . . . 5-4
2.2 Clock Sources for PNX15xx Series . . . 5-7
2.2.1 PLL Specification . . . 5-8
2.2.2 The Clock Dividers . . . 5-10
2.2.3 The DDS Clocks . . . 5-11
2.2.4 DDS and PLL Assignment Summary . . . 5-11
2.2.5 External Clocks . . . 5-11
2.3 Clock Control Logic . . . 5-13
2.4 Bypass Clock Sources . . . 5-14
2.5 Power-up and Reset sequence . . . 5-15
2.6 Clock Stretching . . . 5-15
2.7 Clock Frequency Determination . . . 5-16
2.8 Power Down . . . 5-17
2.8.1 Wake-Up from Power Down . . . 5-17
2.9 Clock Detection . . . 5-18
2.10 VDO Clocks . . . 5-19
2.11 GPIO Clocks . . . 5-20
2.11.1 Setting GPIO[14:12]/GCLOCK[2:0] as Clock Outputs 5-20
2.11.2 GPIO[6:4]/CLOCK[6:4] as Clock Outputs . . . 5-20
2.12 Clock Block Diagrams . . . 5-20
2.12.1 TM3260, DDR and QVCP clocks . . . 5-21
2.12.2 Clock Dividers . . . 5-23
2.12.3 Internal PNX15xx Series Clock from Dividers 5-24
2.12.4 GPIO Clocks . . . 5-26
2.12.5 External Clocks . . . 5-27
2.12.6 SPDO . . . 5-31
3. Registers Definition . . . 5-31
3.1 Registers Summary . . . 5-31
3.2 Registers Description . . . 5-34

Chapter 6: Boot Module

1. Introduction . . . 6-1
2. Functional Description . . . 6-1
2.1 The Boot Modes . . . 6-2
2.2 Boot Module Operation . . . 6-4
2.2.1 MMIO Bus Interface . . . 6-4
2.2.2 I2C Master . . . 6-4
2.2.3 Boot Control/State Machine . . . 6-5
2.3 The Boot Command Language . . . 6-5
3. PNX15xx Series Boot Scripts Content . . . 6-6
3.1 The Common Behavior . . . 6-6
3.1.1 Binary Sequence for the Common Boot Script 6-9
3.2 The Specifics of the Boot From Flash Memory Devices 6-10
3.2.1 Binary Sequence for the Section of the Flash Boot 6-12
3.3 The Specifics of the Host-Assisted Mode . . . 6-12
4. The Boot From an I2C EEPROM . . . 6-14
4.1 External I2C Boot EEPROM Types . . . 6-14
4.2 The Boot Commands and The Endian Mode. 6-15
4.3 Details on I2C Operation . . . 6-15
5. References . . . 6-16

Chapter 7: PCI-XIO Module

1. Introduction . . . 7-1
2. Functional Description . . . 7-2
2.1 Document title variable Block Level Diagram . 7-3
2.2 Architecture . . . 7-3
3. Operation . . . 7-4
3.1 Overview . . . 7-4
3.1.1 NAND-Flash Interface Operation . . . 7-5
3.1.2 Motorola Style Interface . . . 7-10
3.1.3 NOR Flash Interface . . . 7-11
3.1.4 IDE Description . . . 7-13
3.2 PCI Interrupt Enable Register . . . 7-17
4. Application Notes . . . 7-18
4.1 DTL Interface . . . 7-18
4.2 System Memory Bus Interface, the MTL Bus 7-18
4.3 XIO Interface . . . 7-19
4.3.1 Motorola Interface . . . 7-19
4.3.2 NAND-Flash Interface . . . 7-19
4.3.3 NOR Flash Interface . . . 7-19
4.3.4 IDE Interface . . . 7-20
4.4 PCI Endian Support . . . 7-20
4.5 General Notes . . . 7-20
5. Register Descriptions . . . 7-20
5.1 Register Summary . . . 7-21

Chapter 8: General Purpose Input Output Pins

1. Introduction . . . 8-1
2. Functional Description . . . 8-2
2.1 GPIO: The Basic Pin Behavior . . . 8-2
2.1.1 GPIO Mode settings . . . 8-4
2.1.2 GPIO Data Settings MMIO Registers . . . 8-4
2.1.3 GPIO Pin Status Reading . . . 8-6
2.2 GPIO: The Event Monitoring Mode . . . 8-6
2.2.1 Timestamp Reference clock . . . 8-7
2.2.2 Timestamp format . . . 8-7
2.3 GPIO: The Signal Monitoring & Pattern Generation Modes 8-7
2.3.1 The Signal Monitoring Mode . . . 8-8
2.3.2 The Signal Pattern Generation Mode . . . 8-11
2.4 GPIO Error Behaviour . . . 8-14
2.4.1 GPIO Frequency Restrictions . . . 8-15
2.5 The GPIO Clock Pins . . . 8-17
2.6 GPIO Interrupts . . . 8-17
2.7 Timer Sources . . . 8-18
2.8 Wake-up Interrupt . . . 8-18
2.9 External Watchdog . . . 8-18
3. IR Applications . . . 8-18
3.1 Duty-cycle programming . . . 8-19
3.2 Spike Filtering . . . 8-20
4. MMIO Registers . . . 8-21
4.1 GPIO Mode Control Registers . . . 8-24
4.2 GPIO Data Control . . . 8-26
4.3 Readable Internal PNX15xx Series Signals . . 8-26
4.4 Sampling and Pattern Generation Control Registers for the FIFO Queues 8-27
4.5 Signal and Event Monitoring Control Registers for the Timestamp Units 8-34
4.6 Timestamp Unit Registers . . . 8-34
4.7 GPIO Time Counter . . . 8-34
4.8 GPIO TM3260 Timer Input Select . . . 8-35
4.9 GPIO Interrupt Status . . . 8-35
4.10 Clock Out Select . . . 8-36
4.11 GPIO Interrupt Registers for the FIFO Queues (One for each FIFO Queue) 8-37
4.12 GPIO Module Status Register for all 12 Timestamp Units 8-38
4.13 GPIO POWERDOWN . . . 8-43
4.14 GPIO Module ID . . . 8-43
4.15 GPIO IO_SEL Selection Values . . . 8-43

Chapter 9: DDR Controller

1. Introduction . . . 9-1
2. Functional Description . . . 9-1

2.1 Start and Warm Start 9-2
2.1.1 The Start Mode 9-2
2.1.2 Warm Start 9-2
2.1.3 Observing Start State 9-3
2.2 Arbitration 9-3
2.2.1 The First Level of Arbitration: Between the DMA and the CPU 9-3
2.2.2 Second Level of Arbitration 9-6
2.2.3 Dynamic Ratios 9-6
2.2.4 Pre-Emption 9-8
2.2.5 Back Log Buffer (BLB) 9-9
2.2.6 PMAN (Hub) versus DDR Controller Interaction 9-9
2.3 Addressing 9-10
2.3.1 Memory Region Mapping Scheme 9-10
2.3.2 DDR Memory Rank Locations 9-12
2.4 Clock Programming 9-13
2.5 Power Management 9-13
2.5.1 Halting and Unhalting 9-14
2.5.2 MMIO Directed Halt 9-14
2.5.3 Auto Halt 9-14
2.5.4 Observing Halt Mode 9-15
2.5.5 Sequence of Actions 9-16

3. Application Notes 9-16
3.1 Memory Configurations 9-16
3.2 Error Signaling 9-17
3.3 Latency 9-17
3.4 Data Coherency 9-18
3.5 Programming the Internal Arbiter 9-18
3.6 The DDR Controller and the DDR Memory Devices 9-20

4. Timing Diagrams and Tables 9-20
4.0.1 Tcas Timing Parameter 9-21
4.1 Trrd and Trc Timing Parameters 9-21
4.2 Trfc Timing Parameter 9-21
4.3 Twr Timing Parameter 9-22
4.4 Tras Timing Parameter 9-22
4.5 Trp Timing Parameter 9-22
4.6 Trcd_rd Timing Parameter 9-23
4.7 Trcd_wr Timing Parameter 9-23

5. Register Descriptions 9-23
5.1 Register Summary 9-24
5.2 Register Table 9-25

6. References 9-32

Chapter 10: LCD Controller

1. Introduction 10-1
1.1 LCD Controller Features 10-1

2. Functional Description 10-1
2.1 Overview 10-1
2.2 Power Sequencing 10-2

3. Operation 10-3
3.1 Overview 10-3
3.2 Power Sequencing State Machine 10-3
3.2.1 IDLE state 10-4
3.2.2 DCEN state 10-4
3.2.3 BLEN state 10-5
3.2.4 PEPED state 10-5
3.3 Counter 10-5
3.4 Gating Logic 10-5

4. Register Descriptions 10-6
4.1 LCD MMIO Registers 10-7

Chapter 11: QVCP

1. Introduction 11-1
1.1 Features 11-2

2. Functional Description 11-4
2.1 QVCP Block Diagram 11-4
2.2 Architecture 11-5
2.3 Layer Resources and Functions 11-6
2.3.1 Memory Access Control (DMA CTRL) 11-6
2.3.2 Pixel Formatter Unit (PFU) 11-7
2.3.3 Chroma Key and Undither (CKEY/UDTH) Unit 11-7
2.3.4 Chroma Upsample Filter (CUPS) 11-11
2.3.5 Linear Interpolator (LINT) 11-11
2.3.6 Video/Graphics Contrast Brightness Matrix (VCBM) 11-11
2.3.7 Layer and Fetch Control 11-12
2.4 Pool Resources and Functions 11-13
2.4.1 CLUT (Color Look Up Table) 11-13
2.4.2 DCTI (Digital Chroma/Color Transient Improvement) 11-13
2.4.3 HSRU (Horizontal Sample Rate Upconverter) 11-13
2.4.4 HIST (Histogram Modification) Unit 11-14
2.4.5 LSHR (Luminance/Luma Sharpening) Unit 11-14
2.4.6 Color Features (CFTR) Unit 11-14
2.4.7 PLAN (Semi Planar DMA) Unit 11-15
2.5 Screen Timing Generator 11-15
2.6 Mixer Structure 11-16
2.6.1 Key Generation 11-18
2.6.2 Alpha Blending 11-19
2.7 Output Pipeline Structure 11-19
2.7.1 Supported Output Formats 11-20
2.7.2 Layer Selection 11-20
2.7.3 Chrominance Downsampling (CDNS) 11-20
2.7.4 Gamma Correction and Noise Shaping (GNSH& ONSH) 11-20
2.7.5 Output Interface Modes 11-21
2.7.6 Auxiliary Pins 11-22

3. Programming and Resource Assignment 11-23

12NC 9397 750 14321

3.1 MMIO and Task Based Programming 11-23
3.2 Setup Order for the QVCP 11-24
3.2.1 Shadow Registers 11-25
3.2.2 Fast Access Registers 11-29
3.3 Programming of Layer and Pool Resources 11-30
3.3.1 Resource Assignment and Selection 11-30
3.3.2 Aperture Assignment 11-30
3.3.3 Data Flow Selection 11-32
3.3.4 Pool Resource Assignment Example 11-34
3.4 Programming the STG 11-35
3.4.1 Changing Timing 11-36
3.5 Programming QVCP for Different Output Formats 11-36
4. Application Notes 11-37
4.1 Special Features 11-37
4.1.1 Signature Analysis 11-37
4.2 Programming Help 11-37
4.3 LINT Parameters 11-38
4.4 HSRU Parameters 11-38
4.5 LSHR Parameters 11-39
4.6 DCTI Parameters 11-40
4.7 CFTR Parameters 11-40
4.8 Underflow Behavior 11-40
4.8.1 Layer Underflow 11-41
4.8.2 Underflow Symptom 11-41
4.8.3 Underflow Recovery 11-41
4.8.4 Underflow Trouble-shooting 11-41
4.8.5 Underflow Handling 11-41
4.9 Clock Calculations 11-42
5. Register Descriptions 11-43
5.1 Register Summary 11-43
5.2 Register Tables 11-46

Chapter 12: Video Input Processor

1. Introduction 12-1
1.1 Features 12-1
2. Functional Description 12-2
2.1 VIP Block Level Diagram 12-2
2.2 Chip I/O and Connections 12-3
2.2.1 Data Routing and Video Modes 12-3
2.2.2 Input Timing 12-4
2.3 Test Pattern Generator 12-4
2.4 Input Formats 12-5
2.5 Video Data Path 12-8
2.5.1 Video Data Flow 12-8
2.5.2 Video Data Acquisition 12-8
2.5.3 Internal Timing 12-9
2.5.4 Field Identifier Generation 12-9
2.5.5 Horizontal Video Filters (Sampling, Scaling, Color Space Conversion) 12-12
2.5.6 Video Data Write to Memory 12-13
2.5.7 Auxiliary Data Path 12-15
2.5.8 Interrupt Generation 12-19
3. Register Descriptions 12-19
3.1 Register Summary 12-19
3.2 Register Table 12-21

Chapter 13: FGPO: Fast General Purpose Output

1. Introduction 13-1
1.1 FGPO Overview 13-2
1.2 FGPO to VDO pin mapping 13-3
1.3 DTL MMIO Interface 13-3
1.4 Header Initiator 13-3
1.5 Data Initiator 13-3
1.6 Record Output Mode 13-3
1.7 Message Passing Mode 13-4
2. Functional Description 13-5
2.1 Stopping clk_fgpo for output flow control 13-6
2.2 Reset 13-6
2.3 Base Addresses 13-6
2.4 Sample (data) Size 13-7
2.5 Record or Message Size 13-7
2.6 Records or Messages Per Buffer 13-7
2.7 Stride 13-7
2.8 Interrupt Events 13-7
2.8.1 BUF1DONE and BUF2DONE Interrupts 13-8
2.8.2 THRESH1_REACHED and THRESH2_REACHED Interrupts 13-8
2.8.3 UNDERRUN Interrupt 13-8
2.8.4 MBE Interrupt 13-8
2.9 Record or Message Counters 13-9
2.10 Timestamp 13-10
2.11 Variable Length 13-10
2.12 Output Time Registers 13-10
2.13 Double Buffer Operation 13-10
2.14 Single Buffer Operation 13-11
3. Operation 13-11
3.1 Both Operating Modes 13-11
3.1.1 Setup 13-11
3.1.2 Interrupt Service Routines 13-12
3.1.3 Optimized DMA Transfers 13-12
3.1.4 Terminating DMA Transfers 13-12
3.1.5 Signal Edge Definitions 13-12
3.2 Message Passing Mode 13-13
3.3 PNX1300 Series Message Passing Mode . . 13-13
3.4 Record Output Mode 13-13
3.4.1 Record Synchronization Events 13-14
3.4.2 Buffer Synchronization Events 13-14
4. Register Descriptions 13-15
4.1 Mode Register Setup 13-15
4.2 Status Registers 13-20

Chapter 14: FGPI: Fast General Purpose Interface

1. Introduction . . . 14-1
1.1 FGPI Overview . . . 14-2
1.2 VDI to FGPI pin mapping . . . 14-3
1.3 DTL MMIO Interface . . . 14-3
1.4 Data Packer . . . 14-3
1.4.1 8-Bit Sample Packing Mode . . . 14-4
1.4.2 16-bit Sample Packing Mode . . . 14-4
1.4.3 32-bit Sample Mode . . . 14-4
1.5 Record Capture Mode . . . 14-4
1.6 Message Passing Mode . . . 14-4
2. Functional Description . . . 14-6
2.1 Reset . . . 14-6
2.2 Base Addresses . . . 14-7
2.3 Sample (data) Size . . . 14-7
2.4 Record or Message Size . . . 14-7
2.5 Records or Messages Per Buffer . . . 14-8
2.6 Stride . . . 14-8
2.7 Interrupt Events . . . 14-8
2.7.1 BUF1FULL and BUF2FULL Interrupts . . . 14-8
2.7.2 THRESH1_REACHED and THRESH2_REACHED Interrupts 14-8
2.7.3 OVERRUN Interrupt . . . 14-8
2.7.4 MBE Interrupt . . . 14-9
2.7.5 OVERFLOW Interrupt (Message Passing Mode Only) 14-9
2.8 Record or Message Counters . . . 14-9
2.9 Timestamp . . . 14-9
2.10 Variable Length . . . 14-10
2.11 Double Buffer Operation . . . 14-10
2.12 Single Buffer Operation . . . 14-11
2.13 Buffer Synchronization . . . 14-12
3. Operation . . . 14-12
3.1 Both Operating Modes . . . 14-12
3.1.1 Setup . . . 14-12
3.1.2 Interrupt Service Routines . . . 14-13
3.1.3 Optimized DMA Transfers . . . 14-13
3.1.4 Terminating DMA Transfers . . . 14-13
3.1.5 Signal Edge Definitions . . . 14-13
3.2 Message Passing Mode . . . 14-14
3.2.1 Minimum Message/Record Size . . . 14-14
3.3 PNX1300 Series Message Passing Mode . . . 14-15
3.4 Record Capture Mode . . . 14-15
3.4.1 Record Synchronization . . . 14-15
3.4.2 Buffer Synchronization . . . 14-15
3.4.3 Setup and Operation with Input Router VDI_MODE[7] = 1 14-16
4. Register Descriptions . . . 14-18
4.1 Mode Registers . . . 14-18
4.2 Status Registers . . . 14-21

Chapter 15: Audio Output

1. Introduction . . . 15-1
1.1 Features . . . 15-1
2. Functional Description . . . 15-1
2.1 External Interface . . . 15-2
2.2 Memory Data Formats . . . 15-4
2.2.1 Endian Control . . . 15-4
2.3 Audio Out Data DMA Operation . . . 15-5
2.3.1 TRANS_ENABLE . . . 15-5
2.4 Interrupts . . . 15-6
2.4.1 Interrupt Latency . . . 15-6
2.5 Timestamp Events . . . 15-6
2.6 Serial Data Framing . . . 15-6
2.6.1 Serial Frame Limitations . . . 15-8
2.6.2 WS Characteristics . . . 15-8
2.6.3 I^2S Serial Framing Example . . . 15-8
2.7 Codec Control . . . 15-9
2.8 Data Bus Latency and HBE . . . 15-10
2.9 Error Behavior . . . 15-11
3. Operation . . . 15-12
3.1 Clock Programming . . . 15-12
3.1.1 Sample Clock Generator . . . 15-12
3.1.2 Clock System Operation . . . 15-13
3.2 Reset-Related Issues . . . 15-14
3.3 Register Programming Guidelines . . . 15-14
3.4 Power Management . . . 15-14
4. Register Descriptions . . . 15-15
4.1 Register Summary . . . 15-15
4.2 Register Table . . . 15-15

Chapter 16: Audio Input

1. Introduction . . . 16-1
1.1 Features . . . 16-1
2. Functional Description . . . 16-2
2.1 Chip Level External Interface . . . 16-3
2.2 General Operations . . . 16-4
3. Operation . . . 16-5
3.1 Clock Programming . . . 16-5
3.1.1 Clock System Operation . . . 16-5
3.2 Reset-Related Issues . . . 16-6
3.3 Register Programming Guidelines . . . 16-7
3.4 Serial Data Framing . . . 16-7
3.5 Memory Data Formats . . . 16-10
3.5.1 Endian Control . . . 16-10
3.6 Memory Buffers and Capture . . . 16-11
3.7 Data Bus Latency and HBE . . . 16-11
3.8 Error Behavior . . . 16-12
3.9 Interrupts . . . 16-12
3.10 Timestamp Events . . . 16-13
3.11 Diagnostic Mode . . . 16-13
3.12 Software Reset . . . 16-14
3.13 Raw Mode . . . 16-15

12NC 9397 750 14321

4. Register Descriptions 16-15

4.1 Register Table 16-15

Chapter 17: SPDIF Output

1. Introduction 17-1
1.1 Features 17-1
2. Functional Description 17-2
2.1 Architecture 17-2
2.2 General Operations 17-2
3. Operation 17-2
3.1 Clock Programming 17-2
3.1.1 Sample Rate Programming 17-2
3.2 Register Programming Guidelines 17-3
3.3 Data Formatting 17-4
3.3.1 IEC-60958 Serial Format 17-4
3.3.2 Transparent Mode 17-6
3.4 Errors and Interrupts 17-6
3.4.1 DMA Error Conditions 17-6
3.4.2 HBE and Latency 17-7
3.4.3 Interrupts 17-7
3.4.4 Timestamp Events 17-7
3.5 Endian Mode 17-8
4. Signal Description 17-8
4.1 External Interface 17-8
5. Register Descriptions 17-8
5.1 Register Summary 17-8
5.2 Register Table 17-9

Chapter 18: SPDIF Input

1. Introduction 18-1
1.1 Features 18-1
2. Functional Description 18-1
2.1 SPDIF Input Block Level Diagram 18-1
2.2 Architecture 18-2
2.2.1 Functional Modes 18-2
2.3 General Operations 18-3
2.3.1 Received Serial Format 18-3
2.3.2 Memory Formats 18-3
2.3.3 SPDIF Input Endian Mode 18-4
2.3.4 Bandwidth and Latency Requirements 18-5
3. Operation 18-6
3.1 Clock Programming 18-6
3.1.1 SPDIF Input Clock Domains 18-6
3.1.2 SPDIF Receiver Sample Rate Tolerance and IEC6095818-6
3.1.3 SPDIF Input Receiver Jitter Tolerance 18-6
3.1.4 SPDIF Input and the Oversampling Clock . . . 18-7
3.2 Register Programming Guidelines 18-7
3.2.1 SPDIF Input Register Set 18-7
3.2.2 SPDI_STATUS Register Functions 18-8
3.2.3 LOCK and UNLOCK State Behavior 18-8
3.2.4 UNLOCK Error Behavior and DMA 18-8
3.2.5 SPDI_CTL and Functions 18-9
3.2.6 SPDI_CBITSx and Channel Status Bits 18-10
3.2.7 SPDI_UBITSx and User Bits 18-11
3.2.8 SPDI_BASEx and SPDI_SIZE Registers and Memory Buffers18-12
3.2.9 SPDI_SMPMASK and Sample Size Masking . 18-12
3.2.10 SPDI_BPTR and the Start of an IEC60958 Block 18-12
3.2.11 Interrupts 18-13
3.2.12 Event Timestamping 18-13
4. Signal Descriptions 18-14
4.1 External Interface Pins 18-14
4.1.1 System Interface Requirements 18-14
5. Register Descriptions 18-15
5.1 Register Summary 18-15
5.1.1 SPDIF Input Interrupt Registers 18-15
5.2 Register Table 18-17

Chapter 19: Memory Based Scaler

1. Introduction 19-1
2. Functional Description 19-2
2.1 MBS Block Level Diagram 19-2
2.2 Data Flow 19-3
2.2.1 Horizontal Processing Pipeline 19-3
2.2.2 Vertical Processing Pipeline 19-4
2.3 Data Processing in MBS 19-5
2.4 General Operations 19-6
2.4.1 Task Control 19-6
2.4.2 Video Source Controls 19-7
2.4.3 Horizontal Video Filters 19-8
2.4.4 Vertical Video Filters 19-10
2.4.5 De-Interlacing in MBS 19-10
2.4.6 Color-Key Processing 19-10
2.4.7 Alpha Processing 19-11
2.4.8 Video Data Output 19-11
2.4.9 Address Generation 19-12
2.4.10 Interrupt Generation 19-12
2.4.11 Measurement Functions 19-13
3. Register Descriptions 19-14
3.1 Register Summary 19-14
3.2 Register Table 19-16

Chapter 20: 2D Drawing Engine

1. Introduction . . . 20-1
1.1 Features . . . 20-1
2. Functional Description . . . 20-1
2.1 2D Drawing Engine Block Level Diagram . . . 20-2
2.2 Architecture . . . 20-2
2.2.1 Registers . . . 20-2
2.2.2 Host Interface . . . 20-2
2.2.3 Color Expand . . . 20-2
2.2.4 Rotator . . . 20-3
2.2.5 Source FIFO . . . 20-3
2.2.6 Pattern FIFO . . . 20-3
2.2.7 Destination FIFO . . . 20-3
2.2.8 Write Datapath . . . 20-3
2.2.9 Source State . . . 20-3
2.2.10 Destination State . . . 20-3
2.2.11 Address Stepper . . . 20-3
2.2.12 Bit BLT Engine . . . 20-4
2.2.13 Vector Engine . . . 20-4
2.2.14 Memory Interface . . . 20-4
2.2.15 Byte Masking . . . 20-4
2.3 General Operations . . . 20-4
2.3.1 Raster Operations . . . 20-4
2.3.2 Alpha Blending . . . 20-5
2.3.3 Source Data Location and Type . . . 20-5
2.3.4 Patterns . . . 20-6
2.3.5 Transparency . . . 20-6
2.3.6 Block Writes . . . 20-6
3. Operation . . . 20-6
3.1 Register Programming Guidelines . . . 20-6
3.1.1 Alpha Blending . . . 20-6
3.1.2 Mono Expand . . . 20-9
3.1.3 Mono BLT Register Setup . . . 20-10
3.1.4 Solid Fill Setup . . . 20-11
3.1.5 Color BLT Setup . . . 20-11
3.1.6 PatRam . . . 20-12
4. Register Descriptions . . . 20-13
4.1 Register Summary . . . 20-14
4.2 Register Tables . . . 20-15

Chapter 21: MPEG-1 and MPEG-2 Variable Length Decoder

1. Introduction . . . 21-1
1.1 Features . . . 21-1
2. Functional Description . . . 21-3
2.1 VLD Block Level Diagram . . . 21-3
3. Operation . . . 21-3
3.1 Reset-Related Issues . . . 21-3
3.2 VLD MMIO Registers . . . 21-4
3.2.1 VLD Status (VLD_MC_STATUS) . . . 21-4
3.2.2 VLD Interrupt Enable (VLD_IE) . . . 21-5
3.2.3 VLD Control (VLD_CTL) . . . 21-5
3.2.4 VLD DMA Current Read Address (VLD_INP_ADR) and Read Count (VLD_INP_CNT) 21-6
3.2.5 VLD DMA Macroblock Header Current Write Address (VLD_MBH_ADR) 21-6
3.2.6 VLD DMA Macroblock Header Current Write Count 21-6
3.2.7 VLD DMA Run-Level Current Write Address (VLD_RL_ADR) 21-7
3.2.8 VLD DMA Run-Level Current Write Count . . . 21-7
3.2.9 VLD Command (VLD_COMMAND) . . . 21-7
3.2.10 VLD Shift Register (VLD_SR) . . . 21-9
3.2.11 VLD Quantizer Scale (VLD_QS) . . . 21-9
3.2.12 VLD Picture Info (VLD_PI) . . . 21-9
3.2.13 VLD Bit Count (VLD_BIT_CNT) . . . 21-9
3.3 VLD Operation . . . 21-9
3.3.1 VLD Input . . . 21-10
3.3.2 VLD Output . . . 21-10
3.3.3 Restart the VLD Parsing . . . 21-13
3.4 Error Handling . . . 21-13
3.4.1 Unexpected Start Code . . . 21-14
3.4.2 RL Overflow . . . 21-14
3.4.3 Flush . . . 21-14
4. Application Notes . . . 21-15
4.0.1 PNX1300 Series versus PNX15xx Series VLD 21-15
5. Register Descriptions . . . 21-15
5.1 PNX1300 Series and PNX15xx Series Register Differences 21-15
5.2 VLD Register Summary . . . 21-15
5.3 Register Table . . . 21-16

Chapter 22: Digital Video Disc Descrambler

1. Introduction . . . 22-1
1.1 Functional Description . . . 22-1

Chapter 23: LAN100 — Ethernet Media Access Controller

1. Introduction . . . 23-1
1.1 Features . . . 23-1
2. Functional Description . . . 23-2
2.1 Chip I/O and System Interconnections . . . 23-2
2.2 Functional Block Diagram . . . 23-3
2.3 Description . . . 23-4
3. Register Descriptions . . . 23-5
3.1 Register Summary . . . 23-5
3.2 Register Definitions . . . 23-8

3.3 Pattern Matching Join Register 23-25
4. Descriptor and Status Formats 23-27
4.1 Receive Descriptors and Status 23-27
4.2 Transmit Descriptors and Status 23-30
5. LAN100 Functions 23-33
5.1 MMIO Interface 23-33
5.1.1 Overview 23-33
5.2 Direct Memory Access 23-34
5.2.1 Descriptor FIFOs 23-34
5.2.2 Ownership of Descriptors 23-34
5.2.3 Sequential Order with Wrap-around 23-35
5.2.4 Full and Empty State of FIFOs 23-35
5.2.5 Interrupt Bit 23-36
5.2.6 Packet Fragments 23-36
5.3 Initialization 23-37
5.4 Transmit process 23-38
5.4.1 Overview 23-38
5.4.2 Device Driver Sets Up Descriptors and Data 23-38
5.4.3 Tx(Rt) DMA Manager Reads Tx(Rt) Descriptor Arrays 23-39
5.4.4 Tx(Rt) DMA manager transmits data 23-39
5.4.5 Update ConsumeIndex 23-40
5.4.6 Write Transmission Status 23-40
5.4.7 Transmission Error Handling 23-40
5.4.8 Transmit Triggers Interrupts 23-41
5.4.9 Transmit example 23-42
5.5 Receive process 23-45
5.5.1 Device Driver Sets Up Descriptors 23-46
5.5.2 Rx DMA Manager Reads Rx Descriptor Arrays . . 23-46
5.5.3 Rx DMA Manager Receives Data 23-46
5.5.4 Update ProduceIndex 23-47
5.5.5 Write Reception Status 23-47
5.5.6 Reception Error Handling 23-47
5.5.7 Receive Triggers Interrupts 23-48
5.5.8 Device Driver Processes Receive Data 23-49
5.5.9 Receive example 23-49
5.6 Transmission Retry 23-53
5.7 time-stamps 23-53
5.8 Transmission modes 23-53
5.8.1 Overview 23-53
5.8.2 Real-time/non-real-time transmission mode 23-54
5.8.3 Quality-of-service Transmission Mode 23-57
5.9 Duplex Modes 23-58
5.10 IEEE 802.3/Clause 31 Flow Control 23-59
5.10.1 Overview 23-59
5.10.2 Receive Flow Control 23-59
5.10.3 Transmit Flow Control 23-59
5.11 Half-duplex Mode Back Pressure 23-61
5.12 Receive filtering 23-62
5.12.1 Overview 23-62
5.12.2 Unicast, Broadcast and Multicast 23-64
5.12.3 Perfect Address Match 23-64
5.12.4 Imperfect Hash Filtering 23-64
5.12.5 Pattern Match Filtering and Logic Functions 23-65
5.12.6 Enabling and Disabling Filtering 23-66
5.12.7 Runt Frames 23-66
5.13 Wake-up on LAN 23-66
5.13.1 Overview 23-66
5.13.2 Filtering for WoL 23-67
5.13.3 Magic Packet WoL 23-67
5.14 Enabling and Disabling Receive and Transmit 23-68
5.14.1 Enabling and Disabling Reception 23-68
5.14.2 Enabling and Disabling Transmission 23-69
5.15 Transmission Padding and CRC 23-69
5.16 Huge Frames and Frame Length Checking . 23-70
5.17 Statistics Counters 23-71
5.18 Status Vectors 23-71
5.19 Reset 23-71
5.19.1 Hard Reset 23-71
5.19.2 Soft Reset 23-71
6. System Integration 23-73
6.1 MII Interface I/O 23-73
6.2 Power Management 23-74
6.2.1 Sleep Mode 23-74
6.2.2 Coma Mode 23-74
6.3 Disabling the LAN100 23-75
6.4 Little/big Endian 23-75
6.5 Interrupts 23-75
6.6 Errors and Aborts 23-75
6.7 Cache coherency 23-76

Chapter 24: TM3260 Debug

1. Introduction 24-1
1.1 Features 24-1
2. Functional Description 24-1
2.1 General Operations 24-1
2.1.1 Test Access Port (TAP) 24-1
2.1.2 TAP Controller 24-2
2.1.3 PNX15xx Series JTAG Instruction Set 24-4
3. Operation 24-4
3.1 Register Programming Guidelines 24-4
3.1.1 Handshaking and Communication Protocol . . 24-5
3.2 Debug Settings 24-6
4. Register Descriptions 24-7
4.1 Register Summary 24-9

Chapter 25: I^{2}C Interface

1. Introduction 25-1
1.1 Features 25-2
2. Functional Description 25-2
2.1 General Operations 25-2
2.1.1 IIC Arbitration and Control Logic 25-2
2.1.2 Serial Clock Generator 25-3

2.1.3 Bit Counter . . . 25-3
2.1.4 Control Register . . . 25-3
2.1.5 Status Decoder and Register . . . 25-3
2.1.6 Input Filter . . . 25-3
2.1.7 Address Register and Comparator . . . 25-3
2.1.8 Data Shift Register . . . 25-4
2.1.9 Related Interrupts . . . 25-4
2.1.10 Modes of Operation . . . 25-4
3. Register Descriptions . . . 25-7
3.1 Register Tables . . . 25-8

Chapter 26: Memory Arbiter

1. Introduction . . . 26-1
1.1 Features . . . 26-1
2. Functional Description . . . 26-1
2.1 Arbiter Features . . . 26-2
2.2 ID Mapping . . . 26-2
2.2.1 DCS Gate . . . 26-3
2.3 Arbitration Algorithm . . . 26-3
2.3.1 Arbiter Startup Behavior . . . 26-6
3. Operation . . . 26-6
3.1 Clock Programming . . . 26-6
3.2 Register Programming Guidelines . . . 26-6
4. Register Descriptions . . . 26-7
4.1 Register Table . . . 26-7

Chapter 27: Power Management

1. Power Management Mechanisms . . . 27-1
1.1 Clock Management . . . 27-1
1.1.1 Essential Operating Infrastructure During Powerdown 27-1
1.1.2 Module Powerdown Sequence . . . 27-1
1.1.3 Peripheral Module Wakeup Sequence . . . 27-2
1.1.4 TM3260 Powerdown Modes . . . 27-2
1.1.5 SDRAM Controller . . . 27-3

Chapter 28: Pixel Formats

1. Introduction . . . 28-1
2. Summary of Native Pixel Formats . . . 28-2
3. Native Pixel Format Representation . . . 28-3
3.1 Indexed Formats . . . 28-3
3.2 16-Bit Pixel-Packed Formats . . . 28-4
3.3 32-Bit Pixel-Packed Formats . . . 28-4
3.4 Packed YUV 4:2:2 Formats . . . 28-5
3.5 Planar YUV 4:2:0 and YUV 4:2:2 Formats . . . 28-6
3.5.1 Planar Variants . . . 28-6
3.5.2 Semi-Planar 10-Bit YUV 4:2:2 and 4:2:0 Formats 28-9
3.5.3 Packed 10-bit YUV 4:2:2 format . . . 28-10
4. Universal Converter . . . 28-10
5. Alpha Value and Pixel Transparency . . . 28-11
6. RGB and YUV Values . . . 28-11
7. Image Storage Format . . . 28-11
8. System Endian Mode . . . 28-12

Chapter 29: Endian Mode

1. Introduction . . . 29-1
1.1 Features . . . 29-1
2. Functional Description . . . 29-2
2.1 Endian Mode System Block Diagram . . . 29-2
3. Endian Mode Theory . . . 29-4
3.1 Law 1: The “CPU Rule” . . . 29-4
3.2 Law 2: The “DMA Convention Rule” . . . 29-6
4. PNX15xx Series Endian Mode Architecture Details 29-7
4.1 Global Endian Mode . . . 29-7
4.2 Module Control . . . 29-7
4.3 Module DMA . . . 29-8
4.4 SIMD Programming Issues . . . 29-8
4.5 Optional Endian Mode Override . . . 29-8
5. Example: Audio In—Programmer’s View 29-9
6. Implementation Details . . . 29-10
6.1 PMAN Network Endian Block Diagram . . . 29-10
6.2 DMA Across a DTL Interface . . . 29-11
6.2.1 DTL Data Ordering Rules . . . 29-11
6.2.2 Address Invariant Data Ordering Rules . . . 29-12
6.3 Data Transfers Across the DCS Network . . . 29-12
6.4 DMA Across the MTL Bus . . . 29-13
6.5 DTL-to-MTL Adapters . . . 29-14
6.6 PCI Interface . . . 29-14
7. Detailed Example . . . 29-15
1. Introduction . . . 30-1

Chapter 30: DCS Network

2. Functional Description . . . 30-1
2.1 Error Generation . . . 30-2
2.2 Interrupt Generation . . . 30-2

12NC 9397 750 14321

2.3 Programmable Timeout 30-2
2.3.1 Arbitration 30-2
2.4 Endian Mode 30-3
3. Register Descriptions 30-3
3.1 Register Summary 30-3
3.2 Register Tables 30-4

Chapter 31: TM3260

1. Introduction 31-1
1 Data sheet status 3
2. Definitions 3
3. Disclaimers 3
4. Licenses 3
5. Trademarks 3
6. Contact information 3

Chapter 1: Integrated Circuit Data

Figure 1: Application Diagram of the Crystal Oscillator . . . 1-23
Figure 2: SSTL_2 Test Load Condition . . . 1-24
Figure 3: SSTL_2 Receiver Signal Conditions . . . 1-24
Figure 4: BPX2T14MCP Test Load Condition . . . 1-25
Figure 5: BPTS1CHP and BPTS1CP Test Load Condition . . . 1-26
Figure 6: BPTS3CHP Test Load Condition . . . 1-27
Figure 7: BPT3MCHDT5V and BPT3MCHT5V Test Load Condition . . . 1-28
Figure 8: PCI Tval(min) and Slew Rate Test Load Condition . . . 1-29
Figure 9: Reset Timing . . . 1-30
Figure 10: PCI Output and Input Timing Measurement Conditions . . . 1-32
Figure 11: PCI Tval(max) Rising and Falling Edge . . . 1-32
Figure 12: QVCP and FGPO I/O Timing . . . 1-33
Figure 13: VIP and FGPI I/O Timing . . . 1-34
Figure 14: LAN 10/100 I/O Timing in MII Mode . . . 1-35
Figure 15: LAN 10/100 I/O Timing in RMII Mode . . . 1-36
Figure 16: Audio Input I/O Timing . . . 1-37
Figure 17: Audio Output I/O Timing . . . 1-38
Figure 18: SPDIF I/O Timing . . . 1-38
Figure 19: I2C I/O Timing . . . 1-39
Figure 20: I2C I/O Timing . . . 1-40
Figure 21: Audio Output I/O Timing . . . 1-41
Figure 22: JTAG I/O Timing . . . 1-41
Figure 23: BGA456 Plastic Ball grid Array; 456 Balls; body 27 x 27 x 1.75 mm . . . 1-42
Figure 24: Digital VCCP Power Supply to Analog VCCA/VSSA Power Supply Filter . . . 1-44
Figure 25: Digital VDD Power Supply to Analog VDDA/VSSA_1.2 Power Supply Filter . . . 1-44
Figure 26: Digital VDD Power Supply to Analog VDDA/VSSA_1.2 Power Supply Filter . . . 1-45

Chapter 2: Overview

Figure 1: Block Diagram PNX15xx Series . . . 2-1
Figure 2: PNX15xx Series Functional Block Diagram . . . 2-5

Chapter 3: System On Chip Resources

Figure 1: The Two Operating Modes of PNX15xx Series . . . 3-1
Figure 2: PNX15xx Series System Memory Map . . . 3-4
Figure 3: Simplified Internal Bus Infrastructure . . . 3-30

Chapter 4: Reset

Figure 1: Reset Module Block Diagram . . . 4-3
Figure 2: Watchdog in Non Interrupt Mode . . . 4-5
Figure 3: Watchdog in Interrupt Mode . . . 4-6
Figure 4: POR_IN_N Timing and Reset Sequence . . . 4-7

Chapter 5: The Clock Module

Figure 1: Clock Module Block Diagram . . . 5-3
Figure 2: PLL Block Diagram . . . 5-8
Figure 3: Block Diagram of the Clock Control Logic . . . 5-13
Figure 4: Waveforms of the Blocking Logic . . . 5-14
Figure 5: Clock Stretcher . . . 5-16
Figure 6: Clock Detection Circuit . . . 5-18

12NC 9397 750 14321

Figure 7: TM3260, DDR and QVCP clocks . . . 5-21
Figure 8: QVCP_PROC Clock . . . 5-22
Figure 9: QVCP_PIX Clock . . . 5-22
Figure 10: Clock Dividers . . . 5-23
Figure 11: Internal PNX15xx Series Clock from Dividers . . . 5-24
Figure 12: Internal PNX15xx Series Clock from Dividers: PCI, SPDI, LCD and I2C . . . 5-25
Figure 13: Internal PNX15xx Series Clock from Dividers: LCD Timestamp . . . 5-25
Figure 14: GPIO Clocks . . . 5-26
Figure 15: VDI_CLK1 Block Diagram . . . 5-27
Figure 16: VDI_CLK2 Block Diagram . . . 5-27
Figure 17: VDO_CLK1 Block Diagram . . . 5-28
Figure 18: VDO_CLK2 Block Diagram . . . 5-28
Figure 19: AO Clocks . . . 5-29
Figure 20: AI Clocks . . . 5-29
Figure 21: PHY LAN Clock Block Diagram . . . 5-30
Figure 22: Receive and Transmit LAN Clocks . . . 5-30
Figure 23: SPDO Clock . . . 5-31

Chapter 6: Boot Module

Figure 1: Boot Block Diagram . . . 6-4
Figure 2: System Memory Map and Block Diagram Configuration for PNX15xx Series in Standalone Mode . . . 6-10
Figure 3: System Memory Map and Block Diagram Configuration for PNX15xx Series in Host-assisted Mode . . . 6-13

Chapter 7: PCI-XIO Module

Figure 1: Document title variable Block Diagram . . . 7-3
Figure 2: Read Status . . . 7-7
Figure 3: Read Data . . . 7-8
Figure 4: Write Data . . . 7-9
Figure 5: Block Erase . . . 7-9
Figure 6: Motorola Write . . . 7-10
Figure 7: Motorola Read . . . 7-11
Figure 8: NOR Flash Write . . . 7-12
Figure 9: NOR Flash Read . . . 7-12
Figure 10: IDE Interface . . . 7-13
Figure 11: Isolation Translation Logic . . . 7-14
Figure 12: Register Transfer/PIO Data Transfer on IDE . . . 7-16
Figure 13: Timings on IDE Bus . . . 7-17
Figure 14: IDE Transaction, Flow Controlled by Device IORDY . . . 7-17

Chapter 8: General Purpose Input Output Pins

Figure 1: GPIO Module Block Diagram . . . 8-2
Figure 2: Functional Block Diagram of a GPIO Pin . . . 8-4
Figure 3: 32-bit Timestamp Format . . . 8-7
Figure 4: 1-bit Signal Sampling . . . 8-10
Figure 5: Up to 4-bit Signal Sampling . . . 8-11
Figure 6: 1-bit Pattern Generation . . . 8-13
Figure 7: Up to 4-bit Samples per FIFO in Pattern Generation Mode . . . 8-14
Figure 8: Example of Ir TX Signals with and without Sub-Carrier . . . 8-19
Figure 9: IrDA Control TX with Sub-Carrier Enabled . . . 8-19
Figure 10: Sub-Carrier Multiplexing for TX . . . 8-19
Figure 11: Examples of Duty Cycles for Ir TX Signals . . . 8-20

12NC 9397 750 14321

Chapter 9: DDR Controller

Figure 1: The two MTL Ports of the DDR SDRAM Controller . . . 9-3
Figure 2: Arbitration in the DDR Controller . . . 9-3
Figure 3: CPU account . . . 9-5
Figure 4: Arbitration when DMA has priority . . . 9-6
Figure 5: CPU account using dynamic ratios . . . 9-7
Figure 6: Address Mapping: Interleaved Mode . . . 9-10
Figure 7: DDR SDRAM Controller Start and Halt State Machine . . . 9-15
Figure 8: Examples of Supported Memory Configurations . . . 9-17
Figure 9: Tcas Timing Parameter . . . 9-21
Figure 10: Trrd and Trc Timing Parameters . . . 9-21
Figure 11: Trfc Timing Parameter . . . 9-21
Figure 12: Twr Timing Parameter . . . 9-22
Figure 13: Tras Timing Parameter . . . 9-22
Figure 14: Trp Timing Parameter . . . 9-22
Figure 15: Trcd_rd Timing Parameter . . . 9-23
Figure 16: Trcd_wr Timing Parameter . . . 9-23

Chapter 10: LCD Controller

Figure 1: Block diagram of the LCD Controller . . . 10-2
Figure 2: Generic Power Sequence for TFT LCD Panels . . . 10-2
Figure 3: Power Sequencing State Machine Block Diagram . . . 10-4
Figure 4: Clock Gating Logic . . . 10-5

Chapter 11: QVCP

Figure 1: QVCP Top Level Diagram . . . 11-2
Figure 2: QVCP BLock Diagram . . . 11-4
Figure 3: Undithering and Pedestal Manipulation . . . 11-10
Figure 4: 4:2:2 and 4:4:4 Formats . . . 11-11
Figure 5: Mixer Block Diagram—Pixel Selection . . . 11-18
Figure 6: Mixer Block Diagram—Pixel Processing . . . 11-19
Figure 7: VBI/Programming Data Packet Formats . . . 11-24
Figure 8: Shadow Mechanism . . . 11-27
Figure 9: Shadowing of Registers . . . 11-28
Figure 10: Resource Layer and ID . . . 11-31
Figure 11: Resource Layer and ID . . . 11-32
Figure 12: 2-Layer 1 Resource Elements Scenario . . . 11-32
Figure 13: Pool and Aperture Reassignments . . . 11-34
Figure 14: Video Frame Screen Timing . . . 11-35

Chapter 12: Video Input Processor

Figure 1: Simplified VIP Block Diagram . . . 12-2
Figure 2: VIP Module Interface . . . 12-3
Figure 3: Digital Video Input Port Timing Relationships in HD Mode . . . 12-4
Figure 4: Test Pattern . . . 12-5
Figure 5: D1 Data Stream . . . 12-6
Figure 6: HD Dual Data Stream . . . 12-7
Figure 7: Video Data Flow . . . 12-8
Figure 8: Source and Target Window Parameters . . . 12-9
Figure 9: Acquisition Window Counter Reference . . . 12-9
Figure 10: Field Identifier Timing . . . 12-10

Figure 11: Double Buffer Mode . . . 12-14
Figure 12: Auxiliary Data Flow . . . 12-15
Figure 13: ANC Data Structure . . . 12-16
Figure 14: ANC Masked ID Checking . . . 12-17

Chapter 13: FGPO: Fast General Purpose Output

Figure 1: Top Level Block Diagram . . . 13-2
Figure 2: FGPO Module Block Diagram . . . 13-2
Figure 3: Back-to-back Message Passing Example . . . 13-9
Figure 4: Double Buffer Major States . . . 13-11
Figure 5: Signal Edge Definition . . . 13-12
Figure 6: Back-to-back Message Passing Example . . . 13-13

Chapter 14: FGPI: Fast General Purpose Interface

Figure 1: Top Level Block Diagram . . . 14-2
Figure 2: FGPI Module Block Diagram . . . 14-3
Figure 3: Input data width not equal to sample size setting . . . 14-7
Figure 4: Double Buffer Major States . . . 14-11
Figure 5: Buffer Sync Actions . . . 14-12
Figure 6: Signal Edge Definition . . . 14-13
Figure 7: Back-to-back Message Passing Example . . . 14-14

Chapter 15: Audio Output

Figure 1: Audio Out Block Diagram . . . 15-2
Figure 2: Examples of Audio Out Memory DMA Formats . . . 15-4
Figure 3: Definition of Serial Frame Bit Positions (POLARITY = 1, CLOCK_EDGE = 0) . . . 15-7
Figure 4: Serial Frame (64 Bits) of a 18-Bit Precision I^2S D/A Converter . . . 15-8
Figure 5: Example Codec Frame Layout for a Crystal Semiconductor CS4218 . . . 15-10
Figure 6: Audio Out Clock System and I/O Interface . . . 15-12

Chapter 16: Audio Input

Figure 1: Audio In Block Diagram . . . 16-2
Figure 2: Audio In Clock System and I/O Interface . . . 16-5
Figure 3: Audio In Serial Frame and Bit Position Definition (POLARITY = 1, CLOCK_EDGE = 0, EARLYMODE = 0) 16-8
Figure 4: Audio In Serial Frame and Bit Position Definition (POLARITY = 1, CLOCK_EDGE = 0, EARLYMODE = 1) 16-8
Figure 5: Serial Frame of the SAA7366 18-Bit I2S A/D Converter (Format 2 SWS) . . . 16-9
Figure 6: Audio In Memory DMA Formats . . . 16-10

Chapter 17: SPDIF Output

Figure 1: Serial Format of a IEC-60958 Block . . . 17-4
Figure 2: Bi-Phase Mark Data Transmission . . . 17-5
Figure 3: Suggested External SPDIF Output Interface Circuitry . . . 17-8

Chapter 18: SPDIF Input

Figure 1: SPDIF Input Block Diagram . . . 18-2
Figure 2: Serial Format of an IEC60958 Block . . . 18-3
Figure 3: SPDIF Input: Raw Mode Format . . . 18-4
Figure 4: SPDIF Input Sample Order View of Memory . . . 18-4
Figure 5: Endian Mode Byte Address Memory Format . . . 18-5

Figure 6: SPDIF Input Oversampling Clock Generation 18-7
Figure 7: Lock/Unlock Processing for SPDIF Input 18-9
Figure 8: SPDIF Input Consumer interface 18-14
Figure 9: SPDIF Input MMIO Registers (1 of 2) 18-15
Figure 10: SPDIF Input MMIO Registers (2 of 2) 18-16

Chapter 19: Memory Based Scaler

Figure 1: MBS Block Diagram 19-2
Figure 2: MBS Top Level 19-3
Figure 3: MBS Horizontal Processing Pipeline 19-3
Figure 4: MBS Vertical Processing Pipeline 19-4
Figure 5: Task FIFO and Linked List 19-6
Figure 6: Measurement in the MBS 19-13

Chapter 20: 2D Drawing Engine

Figure 1: 2D Drawing Engine Block Diagram 20-2

Chapter 21: MPEG-1 and MPEG-2 Variable Length Decoder

Figure 1: VLD Block Diagram 21-3
Figure 2: MPEG-2 Macro Block Header Output Format 21-11
Figure 3: MPEG-1 Macro Block Header Output Format 21-12

Chapter 22: Digital Video Disc Descrambler

Chapter 23: LAN100 — Ethernet Media Access Controller

Figure 1: Simplified LAN100 I/O Block Diagram 23-2
Figure 2: LAN100 Functional Block Diagram 23-3
Figure 3: Pattern matching join function 23-26
Figure 4: Receive descriptor memory layout 23-27
Figure 5: Transmit Descriptor Memory Layout 23-30
Figure 6: Transmit example memory and registers 23-42
Figure 7: Transmit example waves 23-45
Figure 8: Receive example memory and registers 23-49
Figure 9: Receive example waves 23-52
Figure 10: Real-time/non-real-time transmit example 23-56
Figure 11: QoS transmission example 23-58
Figure 12: Transmit flow control 23-61
Figure 13: Receive filter block diagram 23-63
Figure 14: Receive Active/Inactive state machine 23-68
Figure 15: Transmit Active/Inactive state machine 23-69

Chapter 24: TM3260 Debug

Figure 1: State Diagram of TAP Controller 24-3
Figure 2: System with JTAG Access 24-6
Figure 3: Additional JTAG Data and Control Registers 24-8

Chapter 25: I^2C Interface

Figure 1: SDA First Transmitted Byte 25-5

Chapter 26: Memory Arbiter

Figure 1: Arbitration Scheme 26-4

Chapter 27: Power Management

Chapter 28: Pixel Formats

Figure 1: Native Pixel Format Unit Layout 28-3
Figure 2: Indexed Formats 28-4
Figure 3: 16-Bit Pixel-Packed Formats 28-4
Figure 4: 32-Bit/Pixel Packed Formats 28-5
Figure 5: UYVY Packed YUV 4:2:2 Format 28-5
Figure 6: YUY2/2vuy Packed YUV 4:2:2 Format 28-6
Figure 7: Spatial Sampling Structure of Packed and Planar YUV 4:2:2 Data 28-6
Figure 8: Spatial Sampling Structure of YUV 4:2:0 Data 28-6
Figure 9: Planar YUV 4:2:0 and 4:2:2 Formats 28-7
Figure 10: Semi-Planar YUV 4:2:0 and YUV 4:2:2 Formats 28-8
Figure 11: Semi-Planar 10-bit YUV 4:2:0 and YUV 4:2:2 Formats 28-9
Figure 12: Packed 10-bit YUV 4:2:2 Format 28-10
Figure 13: Image Storage Format 28-12

Chapter 29: Endian Mode

Figure 1: System Block Diagram: Endian-Related Blocks 29-3
Figure 2: Big-Endian Layout of DMA_Descriptor 29-4
Figure 3: Little-Endian Layout of DMA_Descriptor 29-5
Figure 4: Memory Content Created by the C Program 29-6
Figure 5: Audio In Memory Data Structure (Programmer's View) 29-9
Figure 6: Audio In Control/Status MMIO Registers 29-10
Figure 7: Big-Endian External CPU Drawing Two RGB-565 Pixels 29-16

Chapter 30: DCS Network

Chapter 31: TM3260

Chapter 1: Integrated Circuit Data

Table 1: PNX1500 I/O Types . . . 1-2
Table 2: PNX1500 I/O Modes . . . 1-2
Table 3: PNX1500 Special I/Os . . . 1-3
Table 4: PNX1500 Interface . . . 1-3
Table 5: Power Pin List . . . 1-18
Table 6: Pin Reference Voltage . . . 1-19
Table 7: Absolute Maximum Ratings . . . 1-20
Table 8: Operating Range and Thermal Characteristics . . . 1-20
Table 9: Maximum Operating Speeds Based on the VDD Power Supply for the PNX1501 . . . 1-21
Table 10: Maximum Operating Speeds Based on the VDD Power Supply for the PNX1502 . . . 1-21
Table 11: MPEG-2 Decoding with 720x480P Output on PNX1501 . . . 1-22
Table 12: MPEG-2 Decoding with 720x480P Output on PNX1502 . . . 1-22
Table 13: Estimated PNX1501 Maximum and Peak current . . . 1-22
Table 14: Estimated PNX1502 Maximum and Peak current . . . 1-22
Table 15: Specification of HC-49U 27.00000 MHZ Crystal . . . 1-23
Table 16: SSTL_2 AC/DC Characteristics . . . 1-23
Table 17: BPX2T14MCP Characteristics . . . 1-25
Table 18: BPTS1CHP and BPTS1CP Characteristics . . . 1-26
Table 19: BPTS3CHP Characteristics . . . 1-27
Table 20: IPCHP and IPCP Characteristics . . . 1-28
Table 21: BPT3MCHDT5V and BPT3MCHT5V Characteristics . . . 1-28
Table 22: IIC3M4SDAT5V and IIC3M4SCLT5V Characteristics . . . 1-29
Table 23: PCIT5V Characteristics . . . 1-29
Table 24: Reset Timing . . . 1-30
Table 25: DDR DRAM Interface Timing . . . 1-30
Table 26: PCI Bus Timing . . . 1-31
Table 27: QVCP, LCD and FGPO Timing With Internal Clock Generation . . . 1-33
Table 28: QVCP, LCD and FGPO Timing With External Clock Generation . . . 1-33
Table 29: VIP and FGPI Timing . . . 1-34
Table 30: 10/100 LAN MII Timing . . . 1-34
Table 31: 10/100 LAN RMII Timing . . . 1-35
Table 32: Audio Input Timing . . . 1-36
Table 33: Audio Output Timing . . . 1-37
Table 34: SPDIF I/O Timing . . . 1-38
Table 35: I2C I/O Timing . . . 1-39
Table 36: GPIO Timing . . . 1-40
Table 37: JTAG Timing . . . 1-41
Table 38: DDR Recommended Trance Length . . . 1-45
Table 39: Ordering Information . . . 1-47

Chapter 2: Overview

Table 1: Partitioning of Functions to Resources . . . 2-2
Table 2: PNX15xx Series Boot Options . . . 2-6
Table 3: Footprints for 32-bit and 16-bit DDR Interface . . . 2-9
Table 4: TM3260 Characteristics . . . 2-11
Table 5: Native Pixel Format Summary . . . 2-13
Table 6: Video/Data Input Operating Modes . . . 2-19
Table 7: Video/Data Output Operating Modes . . . 2-20
Table 8: PNX15xx Series PCI capabilities . . . 2-24
Table 9: PCI/XIO-16 Bus Interface Unit Capabilities . . . 2-25

Chapter 3: System On Chip Resources

Table 1: SYSTEM Registers . . . 3-6
Table 2: SYSTEM Registers . . . 3-8
Table 3: SYSTEM Registers . . . 3-9
Table 4: Semaphore MMIO Registers . . . 3-11
Table 5: Interrupt Source Assignments . . . 3-12
Table 6: TM3260 Timer Source Selection . . . 3-14
Table 7: TM3260 System Parameters MMIO Registers . . . 3-16
Table 8: Global Registers . . . 3-18
Table 9: Miscellaneous System MMIO registers . . . 3-27
Table 10: System Registers Map Summary . . . 3-29
Table 11: MMIO Memory MAP . . . 3-31

Chapter 4: Reset

Table 1: RESET Module . . . 4-9

Chapter 5: The Clock Module

Table 1: PNX15xx Series Module and Bus Clocks . . . 5-4
Table 2: Current Adjustment Values Based on N . . . 5-9
Table 3: PLL Setting Examples . . . 5-9
Table 4: PLL Characteristics . . . 5-10
Table 5: Internal Clock Dividers . . . 5-10
Table 6: DDS and PLL Clock Assignment . . . 5-11
Table 7: External Clocks . . . 5-11
Table 8: Bypass Clock Sources . . . 5-14
Table 9: Advantages of Centralized Clock Gating Control . . . 5-17
Table 10: Registers Summar . . . 5-31
Table 11: CLOCK MODULE REGISTERS . . . 5-34

Chapter 6: Boot Module

Table 1: The Boot Modes . . . 6-2
Table 2: The Boot Commands . . . 6-6
Table 3: Default DDR SDRAM Timing Parameters . . . 6-7
Table 4: CAS Latency Related DDR SDRAM Timing Parameters . . . 6-7
Table 5: PCI Setup and PCI Command register Content . . . 6-8
Table 6: Binary Sequence for the Common Boot Script . . . 6-9
Table 7: Flash TIming Parameters Used by the Default Boot Scripts . . . 6-11
Table 8: Binary Sequence for the Section of the Flash Boot . . . 6-12
Table 9: Host Configuration Sequence . . . 6-13
Table 10: Examples of I2C EEPROM Devices . . . 6-14

Chapter 7: PCI-XIO Module

Table 1: Supported PCI Commands . . . 7-3
Table 2: XIO Pin Multiplexing . . . 7-4
Table 3: Recommended Settings for NAND . . . 7-6
Table 4: GPXIO Address Configuration . . . 7-15
Table 5: IDE Timing . . . 7-16
Table 6: PCI-XIO Register Summary . . . 7-21
Table 7: PCI Configuration Register Summary . . . 7-22
Table 8: Registers Description . . . 7-23

Table 9: PCI Configuration Registers . . . 7-43

Chapter 8: General Purpose Input Output Pins

Table 1: GPIO Pin List . . . 8-3
Table 2: GPIO Mode Select . . . 8-4
Table 3: Settings for MASK[xx] and IOD[xx] Bits . . . 8-5
Table 4: GPIO clock sources . . . 8-17
Table 5: Example of IR Characteristics . . . 8-18
Table 6: Register Summary . . . 8-21
Table 7: GPIO Mode Control Registers . . . 8-24
Table 8: GPIO Data Control . . . 8-26
Table 9: Readable Internal PNX1500 Signals . . . 8-26
Table 10: Sampling and Pattern Generation Control Registers for the FIFO Queues . . . 8-27
Table 11: Signal and Event Monitoring Control Registers for the Timestamp Units . . . 8-34
Table 12: Timestamp Unit Registers . . . 8-34
Table 13: GPIO Time Counter . . . 8-34
Table 14: GPIO TM3260 Timer Input Select . . . 8-35
Table 15: GPIO Interrupt Status . . . 8-35
Table 16: Clock Out Select . . . 8-36
Table 17: GPIO Interrupt Registers for the FIFO Queues (One for each FIFO Queue) . . . 8-37
Table 18: GPIO Module Status Register for all 12 Timestamp Units . . . 8-38
Table 19: GPIO POWERDOWN . . . 8-43
Table 20: GPIO Module ID . . . 8-43
Table 21: GPIO IO_SEL Selection Values . . . 8-43

Chapter 9: DDR Controller

Table 1: CPU Preemption Field . . . 9-8
Table 2: 32-Byte Interleaving, 256 Columns . . . 9-11
Table 3: 32-Byte Interleaving, 512 Columns . . . 9-11
Table 4: Mapping scheme: 1024-Byte Interleaving, 256 Columns . . . 9-11
Table 5: 1024-Byte Interleaving, 512 Columns . . . 9-12
Table 6: DDR Timing Parameters . . . 9-20
Table 7: DDR Commands . . . 9-20
Table 8: Register Summary . . . 9-24
Table 9: Register Description . . . 9-25

Chapter 10: LCD Controller

Table 1: LCD Controller Register Summary . . . 10-6
Table 2: LCD CONTROLLER Registers . . . 10-7

Chapter 11: QVCP

Table 1: Summary of Native Pixel Formats . . . 11-7
Table 2: Color Key Combining ROPs . . . 11-8
Table 3: Chroma Key ROP Examples . . . 11-9
Table 4: ROP Table for Invert/Select/Alpha/KeyPass/AlphaPass ROPs . . . 11-17
Table 5: Data Packet Descriptor . . . 11-23
Table 6: Shadow Registers . . . 11-28
Table 7: Fast Access Registers . . . 11-29
Table 8: Resource ID Assignment . . . 11-30
Table 9: Register Space Allocation . . . 11-31
Table 10: Rn Association . . . 11-31

12NC 9397 750 14321

Table 11: Resource-Layer Assignment for Pool Resource 11-32
Table 12: Programming Values for Supported PNX15xx Series Output Formats 11-37
Table 13: LINT programming 11-38
Table 14: HSRU programming 11-38
Table 15: LSHR Programming Parameters 11-39
Table 16: DCTI Programming Parameters 11-40
Table 17: CFTR Programming Parameters 11-40
Table 18: Interface Characteristics for Some Target Resolutions 11-42
Table 19: Register Module Association 11-43
Table 20: QVCP 1 Registers 11-46

Chapter 12: Video Input Processor

Table 1: VIP Submodule Descriptions 12-2
Table 2: Test Pattern Generator Setup 12-5
Table 3: Video Input Formats 12-7
Table 4: Relationship Between Input Formats and Video Data Capture 12-8
Table 5: Field Identifier Generation Modes 12-10
Table 6: Output Pixel Formats 12-13
Table 7: Relationship Between Input Formats and Data Capture 12-15
Table 8: Relationship Between Input Formats and Data Capture 12-19
Table 9: VIP MMIO Register Summary 12-19
Table 10: Video Input Processor (VIP) 1 Registers 12-21

Chapter 13: FGPO: Fast General Purpose Output

Table 1: Module signal pins 13-5
Table 2: Register Summary 13-15
Table 3: Fast general purpose output (FGPO) 13-15
Table 4: Status Registers 13-21

Chapter 14: FGPI: Fast General Purpose Interface

Table 1: Module signal pins 14-6
Table 2: Register Summary 14-18
Table 3: Fast general purpose INput (FGPI) 14-18
Table 4: Status Registers 14-22

Chapter 15: Audio Output

Table 1: Audio Out Unit External Signals 15-3
Table 2: Operating Modes and Memory Formats 15-4
Table 3: Bits Transmitted for Each Memory Data Item 15-7
Table 4: Minimum Serial Frame Length in Bits 15-8
Table 5: Example Setup For 64-Bit I2S Framing 15-9
Table 6: Audio Out Latency Tolerance Examples 15-11
Table 7: Clock System Setting 15-13
Table 8: Register Summary 15-15
Table 9: Audio Output Port Registers 15-15

Chapter 16: Audio Input

Table 1: Audio-In I2S Related Ports 16-3
Table 2: Sample Rate Settings 16-6
Table 3: Bit Positions Assigned for Each Data Item 16-9

Table 4: Example Setup For SAA7366 16-9
Table 5: Operating Modes and Memory Formats 16-10
Table 6: Endian Ordering of Audio Data in Main Memory 16-10
Table 7: Audio In Data Bus Arbiter Latency Requirement Examples — 16-Bit Data Examples 16-12
Table 8: Audio In Data Bus Arbiter Latency Requirement Examples — 32-Bit Data Examples 16-12
Table 9: Raw Mode Format of Input Data and Word Select 16-15
Table 10: Register Summary 16-15
Table 11: Audio (I^2S) Input Ports Registers 16-15

Chapter 17: SPDIF Output

Table 1: SPDIF Out Sample Rates and Jitter 17-2
Table 2: SPDIF Subframe Descriptor Word 17-6
Table 3: SPDO Block Latency Requirements 17-7
Table 4: SPDIF Out External Signals 17-8
Table 5: SPDIF Output Module Register Summary 17-8
Table 6: SPDO Registers 17-9

Chapter 18: SPDIF Input

Table 1: SPDIF Input Oversampling Clock Value Settings 18-6
Table 2: Input Jitter for Different Sample Rates 18-7
Table 3: SPDI_CBITS1 Channel Status Meaning 18-10
Table 4: SPDI_CBITS2 Channel Status Meaning 18-11
Table 5: SPDIF Input Pin Summary 18-14
Table 6: SPDIF Input Registers 18-17

Chapter 19: Memory Based Scaler

Table 1: Pipeline Processing (Horizontal First Mode) 19-5
Table 2: Pipeline Processing (Vertical First Mode) 19-5
Table 3: De-Interlacing Mode Maximum Filter Lengths 19-5
Table 4: Task Descriptor Opcode Table 19-7
Table 5: Input Pixel Formats 19-8
Table 6: Output Pixel Formats 19-11
Table 7: Register Summary 19-14
Table 8: Memory Based Scaler (MBS) Registers 19-16

Chapter 20: 2D Drawing Engine

Table 1: Source and Destination Data 20-5
Table 2: Mono Bitmap & Text Data Parameters 20-10
Table 3: Solid Color Fill Parameters 20-11
Table 4: Color BLT Parameters 20-11
Table 5: 2DE Memory Space Addresses 20-14
Table 6: 2D Command Registers 20-14
Table 7: 2D Real Time Drawing Registers 20-15
Table 8: Registers Description 20-15
Table 9: Destination Address Base 20-16
Table 10: Pixel Size 20-17
Table 11: Pixel Format Bit Assignments 20-17
Table 12: Dithering 20-18
Table 13: Source Linear 20-18
Table 14: Destination Linear 20-18
Table 15: Source Stride 20-19

12NC 9397 750 14321

Table 16: Destination Stride . . . 20-19
Table 17: Color Compare . . . 20-20
Table 18: Mono Host F Color or SurfAlpha . . . 20-21
Table 19: Mono Host B Color or HAlpha Color . . . 20-21
Table 20: Blt Control . . . 20-22
Table 21: Source Address, XY Coordinates . . . 20-23
Table 22: Destination Address, XY Coordinates . . . 20-24
Table 23: BLT Size . . . 20-24
Table 24: Destination Address, XY2 Coordinates . . . 20-25
Table 25: Vector Constant . . . 20-25
Table 26: Vector Count Control . . . 20-26
Table 27: TransMask . . . 20-26
Table 28: MonoPatFColor . . . 20-27
Table 29: MonoPatBColor . . . 20-27
Table 30: EngineStatus . . . 20-28
Table 31: PanicControl . . . 20-29
Table 32: EngineConfig . . . 20-29
Table 33: HostFIFOStatus . . . 20-30
Table 34: POWERDOWN . . . 20-31
Table 35: Module ID . . . 20-31
Table 36: Drawing Engine Data Registers . . . 20-31

Chapter 21: MPEG-1 and MPEG-2 Variable Length Decoder

Table 1: Software Reset Procedure . . . 21-4
Table 2: VLD STATUS . . . 21-5
Table 3: VLD Control . . . 21-6
Table 4: VLD Commands . . . 21-8
Table 5: VLD Command Register . . . 21-9
Table 6: References for the MPEG-2 Macroblock Header Data . . . 21-11
Table 7: References for the MPEG-1 Macroblock Header Data . . . 21-13
Table 8: VLD Error Handling . . . 21-14
Table 9: Register Summary . . . 21-15
Table 10: VLD Registers . . . 21-16

Chapter 22: Digital Video Disc Descrambler

Chapter 23: LAN100 — Ethernet Media Access Controller

Table 1: LAN100 MMIO Register Map . . . 23-6
Table 2: LAN100 Registers . . . 23-9
Table 3: PatternMatchJoin Register Nibble Functions . . . 23-26
Table 4: Receive Descriptor Structure . . . 23-28
Table 5: Receive Descriptor Control Word . . . 23-28
Table 6: Receive Status Structure . . . 23-29
Table 7: Receive Status Information Word . . . 23-29
Table 8: Transmit Descriptor Fields . . . 23-31
Table 9: Transmit Descriptor Control Word . . . 23-32
Table 10: Transmit Status Structure . . . 23-32
Table 11: Transmit Status Information Word . . . 23-33
Table 12: LAN100 Pin Interface to external PHY . . . 23-73

Chapter 24: TM3260 Debug

Table 1: JTAG TM3260 Instruction Encoding . . . 24-4
Table 2: JTAG Instruction Encoding . . . 24-4
Table 3: Transfer of Data In via JTAG . . . 24-6
Table 4: Register Summary . . . 24-9
Table 5: TM_DBG 1 Registers . . . 24-10

Chapter 25: I^2C Interface

Table 1: Register Summary . . . 25-7
Table 2: IIC Registers . . . 25-8
Table 3: IIC Registers . . . 25-10
Table 4: IIC Registers . . . 25-11
Table 5: Status Codes . . . 25-11
Table 6: IIC Registers . . . 25-16
Table 7: IIC Registers . . . 25-17

Chapter 26: Memory Arbiter

Table 1: Peripheral ID and Sub-Arbitration . . . 26-2
Table 2: Register Summary . . . 26-7
Table 3: PMAN (Hub) Arbiter Registers . . . 26-7

Chapter 27: Power Management

Chapter 28: Pixel Formats

Table 1: Native Pixel Format Summary . . . 28-2
Table 2: Alpha Code Value and Pixel Transparency . . . 28-11

Chapter 29: Endian Mode

Table 1: Memory Result of a Store to Address ‘a’ Instruction . . . 29-5
Table 2: Register Result of an (Unsigned) Load Instruction . . . 29-5
Table 3: Register Result of a (Signed) Load Instruction . . . 29-6
Table 4: Precise Mapping Audio In Sample Time and Bits to Memory Bytes . . . 29-9
Table 5: DTL Interface Rules . . . 29-11
Table 6: 32 Bit DTL Interface Byte Address . . . 29-12
Table 7: DTL Interface Rules . . . 29-12
Table 8: DCS Network Data Transfer Rules (32 Bits at-a-time Transfer) . . . 29-12
Table 9: MTL Memory Bus Byte Address . . . 29-13
Table 10: MTL Memory Bus Item DMA Rules . . . 29-13
Table 11: 32 Bit PCI Interface Byte Address . . . 29-15

Chapter 30: DCS Network

Table 1: DCS Controller_TriMedia Configuration Register Summary . . . 30-3
Table 2: DCS Controller_TriMedia Configuration Registers (Rev 0.32) . . . 30-4

Chapter 31: TM3260

Chapter 1: Integrated Circuit Data

PNX15xx Series Data Book – Volume 1 of 1

Rev. 2 — 1 December 2004 Product data sheet

1. Introduction

The PNX1500 Media Processor Series is a complete Audio/Video/Graphics system on a chip that contains a high-performance 32-bit VLIW processor, TriMedia™ TM3260, capable of high quality software video (multi-video standard digital decoder/encoder and image improvement), audio signal processing, as well as general purpose control processing. It can either be used in standalone, or as an accelerator to a general purpose processor. The PNX1500 processes the input signals by utilizing several Audio/Video and co-processor modules before send them to the external peripherals. These modules provide additional video and data processing bandwidth without taking away precious CPU cycles. The combination of the CPU and co-processor modules makes the PNX1500 System On-Chip suitable for most applications, especially those requiring high level of processing power/throughput at a reduced cost.

Refer to Section 13. on page 1-47 for ordering information as well as for the different PNX1500 derivatives available. Throughout this document PNX1500 or PNX15xx Series will be used to refer to any of the derivatives of PNX1500 devices unless otherwise specified.

2. Pin Description

2.1 Boundary Scan Notice

PNX1500 implements full IEEE1149.1 boundary scan. Any pin designated ‘IN’ only (from functionality point of view) can function as an output during boundary scan.

2.2 I/O Circuit Summary

PNX1500 has a total of 275 functional pins, 1 reserved pin, and 180 power pins.

The I/Os are powered by a 2.5 V and 3.3 V power supplies.

PNX1500 supports 5 V input tolerant pins for certain interfaces such as PCI and I^2C.

Refer to Section 2.3.2 on page 1-19 for a summary list of the voltage reference for each pin.

PNX1500 uses different I/Os depending on the type of the interface, e.g. PCI, or electrical characteristics needed for the functionality, e.g. a clock signal requires sharper edges than a regular signal. The following table summarizes the types of I/Os, a.k.a. pads, used in PNX1500.

Table 1: PNX1500 I/O Types

Pad Type	Description
PCIT5V	PCI 2.2 compliant I/O using 3.3- or 5- V PCI signaling conventions.
IIC3M4SDAT5V IIC3M4SCLT5V	Open drain 3.3- or 5- V I^2C I/Os.
BPX2T14MCP	3.3-V low impedance output, with fast rise/fall time, combined with 3.3-V input only. Used for Clock signals requires board level 27-33 Ω series terminator resistor to match 50 Ω PCB trace.
BPTS1CP	3.3-V regular impedance output, with fast rise/fall time, combined with 3.3-V input only.
BPTS1CHP	3.3-V regular impedance output, with fast rise/fall time, combined with 3.3-V input only with hysteresis.
BPTS3CHP	3.3-V regular impedance output, with slow rise/fall time, combined with 3.3-V input only with hysteresis.
BPT3MCHT5V	3.3-V regular impedance output, with slow rise/fall time, combined with 5-V tolerant input with hysteresis.
BPT3MCHDT5V	3.3-V regular impedance output, with slow rise/fall time, combined with 5-V tolerant input with hysteresis and internal pull-down. **Note:** The pull-down is NOT strong enough to actually pull down a 5-V TTL input. Instead the TTL input pin sees a ‘1’.
IPCP	3.3-V input only.
IPCHP	3.3-V input only with hysteresis.
SSTLCLKIO	2.5 V SSTL_2 low impedance, e.g. DDR SDRAM clocks. Requires a board level 10 Ω series terminator resistor to match a 50 Ω PCB trace.
SSTLADDIO	2.5-V SSTL_2 low impedance for output signals, e.g. DDR SDRAM address and control signals. Requires a board level matched 50 Ω PCB trace.
SSTLDATIO	2.5-V SSTL_2 low impedance for DDR SDRAM data signals. Requires a board level matched 50 Ω PCB trace.

The above pad types are used in the modes listed in the following table

Table 2: PNX1500 I/O Modes

Modes	Description
IN	Input only, except during boundary scan or GPIO mode.
OUT	Output only, except when used as a GPIO pin.
OD	Open drain output - active pull low, no active drive high, requires external pull-up.
I/O	Input or Output.
I/OD	Input or open drain output - active pull low, no active drive high, requires external pull-up.
I/O/D	Input or output or open drain output with input - active pull low, no active drive high, requires external pull-up when operated in open drain mode.
O	Output or floating.

Unused pins may remain unconnected, i.e. floating if they contain an internal pull-up or pull-down. More specifically,

- PCI_FRAME_N, PCI_TDRY_N, PCI_IRDY_N, PCI_DEVSEL_N, PCI_STOP_N, PCI_SERR_N, PCI_PERR_N and PCI_INTA_N require an external pull-up. Refer to **Section 4.3.3** of PCI 2.2 specification for more details.

- Any I/O or I/OD signal of the XIO bus must be pulled-up if they are not used.
- GPIO[11:8] must be pulled-up or down.

The following Section 2.3 contains a table that specifies if the pin contains a pull-up, a pull-down or none (column ‘P’).

Remark: The pull-down in the BPT3MCHDT5V pads is NOT strong enough to actually pull down a 5-V TTL input. Instead the TTL input pin sees a ‘1’.

Speciality pads, e.g. power supply, are described in the following table.

Table 3: PNX1500 Special I/Os

Name	Description
APIO1V2	Analog for the 1.2-1.3-V logic.
APIO3V3	Analog for the 3.3-V logic.
APOD	Generic Analog signal.
SSTLREFGEN	Reference voltage for the DDR SDRAM interface.
VDDE3V3	3.3-V I/O power supply for peripherals I/Os. 2.5-V I/O power supply for the memory DDR SDRAM I/Os. These I/Os are 3.3-V capable for Automated Test Equipment (ATE), **not** for functional mode.
VDDI	1.2-1.3-V core power supply.
VSSE	Common ground for I/Os (i.e. the 2.5-V and 3.3-V power supplies).
VSSIS	Common ground for the core (i.e. the 1.2-1.3-V power supply).

2.3 Signal Pin List

The following table details the interface of PNX1500. For pad and I/O types, refer to the tables presented in Section 2.2. The I/O type indicates the functional mode (i.e. a dedicated GPIO pin is always of I/O/D type). The ‘P’ column indicates if the signal is pulled down, ‘D’, or pulled up, ‘P’ or neither ‘-’. Active low signals are suffixed by ‘_N’.

Remark: The pull-down in the BPT3MCHDT5V pads is NOT strong enough to actually pull down a 5-V TTL input. Instead the TTL input pin sees a ‘1’.

Table 4: PNX1500 Interface

Pin Name	BGA Ball	Pad Type	I/O Type	GPIO #	GPIO P	Description
System Clock						
XTAL_IN	D11	APIO1V2	IN	-	-	PNX1500 main input clock. All internal clocks are derived from this 27 MHz input reference clock. The crystal should be placed as close as possible to the package. Refer to Figure 1 and Figure 25 for board level connections. **This input is 1.2V ONLY**.
XTAL_OUT	D9	APIO1V2	OUT	-	-	Crystal oscillator output. Connect external crystal between this pin and XTAL_IN. Refer to Figure 1 and Figure 25 for board level connections.

Table 4: PNX1500 Interface

Pin Name	BGA Ball	Pad Type	I/O Type	GPIO #	P	Description
PCI_SYS_CLK	E25	BPX2T14MCP	OUT	-	U	This clock is intended for use as the PCI clock in simple PNX1500 PCI configurations. It outputs a 33.23 MHz clock. A board level 27-33 Ω series resistor is recommended to reduce ringing.
Miscellaneous System Interface						
POR_IN_N	A11	BPT3MCHT5V	IN	-	U	PNX1500 Power On Reset input. Asserting this input low triggers the hardware reset function of the PNX1500 (**including the JTAG state machine**). This pin can typically be connected to an on-board reset upon voltage drop. It is active low. Upon asserting this reset input, the PNX1500 asserts SYS_RST_OUT_N to reset the attached peripheral chips. This pin can also be tied to the PCI_RST_N signal in PCI bus systems. This pin is 5 V tolerant input.
RESET_IN_N	C7	BPT3MCHT5V	IN	-	U	PNX1500 reset input. Asserting this input low triggers the hardware reset function of the PNX1500 (**This does not reset the JTAG state machine**). Upon asserting this reset input, PNX1500 asserts SYS_RST_OUT_N to reset attached peripheral chips. This pin can also be tied to the PCI_RST_N signal in PCI bus systems. With respect to the POR_IN_N reset pin, this pin can be used has a warm reset. For most applications, both reset pins can be tied together. it is active low. This pin is 5 V tolerant input.
SYS_RST_OUT_N	D10	BPX2T14MCP	OUT	-	U	Active low peripheral reset output. This output is asserted upon any PNX1500 reset (hardware, watchdog timer or software), and de-asserted by PNX1500 system software. It is intended to be used as a reset for external peripherals.
RESERVED	AB23	BPT3MCHDT5V	I/O	-	D	Reserved for future expansion. It has to be left unconnected at the board level for normal operation. It was named RESERVED2.
Main Memory Interface (DDR SDRAM controller) **Refer to Section 10.3 on page 1-45 for board design guidelines**						
MM_CLK	M1	SSTLCLKIO	OUT	-	-	DDR SDRAM Output Clock. Refer to Section 10.3 on page 1-45 for board level connections.
MM_CLK_N	M2	SSTLCLKIO	OUT	-	-	
MM_CS1_N	V4	SSTLADDIO	OUT	-	-	Chip select for DDR SDRAM. It is active low.
MM_CS0_N	L3	SSTLADDIO	OUT	-	-	
MM_RAS_N	L1	SSTLADDIO	OUT	-	-	Row address strobe. It is active low.
MM_CAS_N	M4	SSTLADDIO	OUT	-	-	Column address strobe. It is active low.
MM_WE_N	N3	SSTLADDIO	OUT	-	-	Write enable. It is active low
MM_CKE	J2	SSTLADDIO	OUT	-	-	Clock enable output to DDR SDRAMs.
AVREF	N2	SSTLREFGEN	IN	-	-	Voltage reference.

Table 4: PNX1500 Interface

Pin Name	BGA Ball	Pad Type	I/O Type	GPIO #	P	Description
MM_BA1	P4	SSTLADDIO	OUT	-	-	DDR SDRAM bank address. It supports 4-bank types of SDRAMs.
MM_BA0	R4	SSTLADDIO	OUT	-	-	
MM_ADDR12	K4	SSTLADDIO	OUT	-	-	DDR SDRAM address bus. It is used for row and column addresses.
MM_ADDR11	K3	SSTLADDIO	OUT	-	-	
MM_ADDR10	T4	SSTLADDIO	OUT	-	-	
MM_ADDR09	L4	SSTLADDIO	OUT	-	-	
MM_ADDR08	N4	SSTLADDIO	OUT	-	-	
MM_ADDR07	P1	SSTLADDIO	OUT	-	-	
MM_ADDR06	R1	SSTLADDIO	OUT	-	-	
MM_ADDR05	T1	SSTLADDIO	OUT	-	-	
MM_ADDR04	U3	SSTLADDIO	OUT	-	-	
MM_ADDR03	U4	SSTLADDIO	OUT	-	-	
MM_ADDR02	T3	SSTLADDIO	OUT	-	-	
MM_ADDR01	P3	SSTLADDIO	OUT	-	-	
MM_ADDR00	R2	SSTLADDIO	OUT	-	-	
MM_DQM3	U2	SSTLADDIO	OUT	-	-	Byte write enable signals:
MM_DQM2	V3	SSTLADDIO	OUT	-	-	MM_DQM0 is attached to byte MM_DATA[7:0]
MM_DQM1	J4	SSTLADDIO	OUT	-	-	MM_DQM1 is attached to byte MM_DATA[15:8]
MM_DQM0	K2	SSTLADDIO	OUT	-	-	MM_DQM2 is attached to byte MM_DATA[23:16]
						MM_DQM3 is attached to byte MM_DATA[31:24]
MM_DQS3	V1	SSTLDATIO	I/O	-	-	Byte strobe signals:
MM_DQS2	Y1	SSTLDATIO	I/O	-	-	MM_DQS0 is attached to byte MM_DATA[7:0]
MM_DQS1	G1	SSTLDATIO	I/O	-	-	MM_DQS1 is attached to byte MM_DATA[15:8]
MM_DQS0	J1	SSTLDATIO	I/O	-	-	MM_DQS2 is attached to byte MM_DATA[23:16]
						MM_DQS3 is attached to byte MM_DATA[31:24]

Table 4: PNX1500 Interface

Pin Name	BGA Ball	Pad Type	I/O Type	GPIO #	P	Description
MM_DATA31	AD2	SSTLDATIO	I/O	-	-	DDR SDRAM data I/O bus.
MM_DATA30	AD1	SSTLDATIO	I/O	-	-	
MM_DATA29	AB2	SSTLDATIO	I/O	-	-	
MM_DATA28	AC1	SSTLDATIO	I/O	-	-	
MM_DATA27	AB1	SSTLDATIO	I/O	-	-	
MM_DATA26	AA2	SSTLDATIO	I/O	-	-	
MM_DATA25	AA1	SSTLDATIO	I/O	-	-	
MM_DATA24	W2	SSTLDATIO	I/O	-	-	
MM_DATA23	W4	SSTLDATIO	I/O	-	-	
MM_DATA22	Y3	SSTLDATIO	I/O	-	-	
MM_DATA21	Y4	SSTLDATIO	I/O	-	-	
MM_DATA20	AA3	SSTLDATIO	I/O	-	-	
MM_DATA19	AB3	SSTLDATIO	I/O	-	-	
MM_DATA18	AB4	SSTLDATIO	I/O	-	-	
MM_DATA17	AC3	SSTLDATIO	I/O	-	-	
MM_DATA16	AD3	SSTLDATIO	I/O	-	-	
MM_DATA15	C3	SSTLDATIO	I/O	-	-	
MM_DATA14	D3	SSTLDATIO	I/O	-	-	
MM_DATA13	E4	SSTLDATIO	I/O	-	-	
MM_DATA12	E3	SSTLDATIO	I/O	-	-	
MM_DATA11	F3	SSTLDATIO	I/O	-	-	
MM_DATA10	G4	SSTLDATIO	I/O	-	-	
MM_DATA09	G3	SSTLDATIO	I/O	-	-	
MM_DATA08	H4	SSTLDATIO	I/O	-	-	
MM_DATA07	H2	SSTLDATIO	I/O	-	-	
MM_DATA06	F1	SSTLDATIO	I/O	-	-	
MM_DATA05	F2	SSTLDATIO	I/O	-	-	
MM_DATA04	E1	SSTLDATIO	I/O	-	-	
MM_DATA03	D1	SSTLDATIO	I/O	-	-	
MM_DATA02	E2	SSTLDATIO	I/O	-	-	
MM_DATA01	C1	SSTLDATIO	I/O	-	-	
MM_DATA00	C2	SSTLDATIO	I/O	-	-	
33 MHz, 32-bit PCI 2.2 Bus Interface and XIO 8-bit Interface (Flash, M68K system bus) **(note: buffer design allows drive/receive from either 3.3 or 5 V PCI bus)**						
PCI_CLK	E23	PCIT5V	IN	-	-	All PCI input signals are sampled with respect to the rising edge of this clock. All PCI outputs are generated based on this clock. In small PCI configurations, PCI_SYS_CLK can be used to provide this clock.

Table 4: PNX1500 Interface

Pin Name	BGA Ball	Pad Type	I/O Type	GPIO #	P	Description
PCI_AD31	H24	PCIT5V	I/O	-	-	Multiplexed address and data I/O bus.
PCI_AD30	G26	PCIT5V	I/O	-	-	
PCI_AD29	J23	PCIT5V	I/O	-	-	
PCI_AD28	H25	PCIT5V	I/O	-	-	
PCI_AD27	H26	PCIT5V	I/O	-	-	
PCI_AD26	K23	PCIT5V	I/O	-	-	
PCI_AD25	J25	PCIT5V	I/O	-	-	
PCI_AD24	J26	PCIT5V	I/O	-	-	
PCI_AD23	L23	PCIT5V	I/O	-	-	
PCI_AD22	L24	PCIT5V	I/O	-	-	
PCI_AD21	L25	PCIT5V	I/O	-	-	
PCI_AD20	L26	PCIT5V	I/O	-	-	
PCI_AD19	M24	PCIT5V	I/O	-	-	
PCI_AD18	M23	PCIT5V	I/O	-	-	
PCI_AD17	N23	PCIT5V	I/O	-	-	
PCI_AD16	M25	PCIT5V	I/O	-	-	
PCI_AD15	R26	PCIT5V	I/O	-	-	
PCI_AD14	T26	PCIT5V	I/O	-	-	
PCI_AD13	T25	PCIT5V	I/O	-	-	
PCI_AD12	T24	PCIT5V	I/O	-	-	
PCI_AD11	U26	PCIT5V	I/O	-	-	
PCI_AD10	T23	PCIT5V	I/O	-	-	
PCI_AD09	U24	PCIT5V	I/O	-	-	
PCI_AD08	U23	PCIT5V	I/O	-	-	
PCI_AD07	V26	PCIT5V	I/O	-	-	
PCI_AD06	V23	PCIT5V	I/O	-	-	
PCI_AD05	W26	PCIT5V	I/O	-	-	
PCI_AD04	W25	PCIT5V	I/O	-	-	
PCI_AD03	W24	PCIT5V	I/O	-	-	
PCI_AD02	Y26	PCIT5V	I/O	-	-	
PCI_AD01	W23	PCIT5V	I/O	-	-	
PCI_AD00	Y23	PCIT5V	I/O	-	-	
PCI_C/BE3	K24	PCIT5V	I/O	-	-	Multiplexed bus Commands and Byte Enables.
PCI_C/BE2	M26	PCIT5V	I/O	-	-	
PCI_C/BE1	R23	PCIT5V	I/O	-	-	
PCI_C/BE0	V25	PCIT5V	I/O	-	-	
PCI_PAR	R24	PCIT5V	I/O	-	-	Even Parity across AD[31:0] and C/BE[3:0] lines.
PCI_FRAME_N	N26	PCIT5V	I/O	-	-	Sustained Tri-state. Frame is driven by a master to indicate the beginning and duration of an access.
PCI_IRDY_N	N25	PCIT5V	I/O	-	-	Sustained Tri-state. Initiator Ready indicates that the bus master is ready to complete the current data phase.

Table 4: PNX1500 Interface

Pin Name	BGA Ball	Pad Type	I/O Type	GPIO #	P	Description
PCI_TRDY_N	N24	PCIT5V	I/O	-	-	Sustained Tri-state. Target Ready indicates that the bus target is ready to complete the current data phase.
PCI_STOP_N	P24	PCIT5V	I/O	-	-	Sustained Tri-state. It indicates that the target is requesting that the master stop the current transaction.
PCI_IDSEL	K26	PCIT5V	IN	-	-	Used as Chip Select during configuration read/write cycles.
PCI_DEVSEL_N	P26	PCIT5V	I/O	-	-	Sustained Tri-state. It indicates whether any device on the bus has been selected.
PCI_REQ_N	F23	PCIT5V	I/O	-	-	If the PNX1500 is the arbiter of the PCI bus, this pin acts as a request input for an external device, otherwise it is driven by the PNX1500 as a PCI bus master to request the use of the PCI bus.
PCI_GNT_N	D24	PCIT5V	I/O	-	-	If the PNX1500 is the arbiter of the PCI bus, this pin acts as an output to grant the requester, otherwise it Indicates to the PNX1500 that an access to the bus has been granted.
PCI_REQ_A_N	G23	PCIT5V	IN	-	-	If the PNX1500 is the arbiter of the PCI bus, this pin acts as a request input for an external device. This pin can also be used as an input for an external interrupt line for the TM3260.
PCI_GNT_A_N	D25	PCIT5V	I/O	-	-	If the PNX1500 is the arbiter of the PCI bus, this pin acts as an output to grant the requester. If the internal PCI arbiter is not used, this pin can be used as an input for an external interrupt line for the TM3260.
PCI_REQ_B_N	H23	PCIT5V	IN	-	-	If the PNX1500 is the arbiter of the PCI bus, this pin acts as a request input for an external device. This pin can be used as an input for an external interrupt line for the TM3260. This pins is also used as a DSACK signal when using the M68K system bus on the PCI-XIO interface.
PCI_GNT_B_N	D26	PCIT5V	I/O	-	-	If the PNX1500 is the arbiter of the PCI bus, this pin acts as an output to grant the requester. If the internal PCI arbiter is not used, this pin can be used as an input for an external interrupt line for the TM3260.
PCI_PERR_N	P23	PCIT5V	I/O	-	-	Sustained Tri-state. Parity errors are generated/ received by the PNX1500 through this pin.
PCI_SERR_N	R25	PCIT5V	OD	-	-	System Error. This signal is asserted when operating as a target when it detects an address parity error.

Table 4: PNX1500 Interface

Pin Name	BGA Ball	Pad Type	I/O Type	GPIO #	P	Description
PCI_INTA_N	D23	PCIT5V	I/OD	-	-	It is specifically intended to be used as the $\overline{\text{INTA}}$ pin, so that the software requires less board specific information. It should be configured and used as the PCI interrupt output for the case when an external PCI host exists. Interrupts are asserted by the software running on the TM3260. In standalone systems where the PNX1500 is the PCI host, this pin should be configured as an input allowing external PCI devices to request an interrupt service from the TM3260 CPU.
Additional XIO bus signals to the regular PCI bus signals to implement Flash, IDE drive interface and M68k System Buses.						
XIO_D15	AA25	PCIT5V	I/O	34	-	XIO extended 8-bit data signals for the 16-bit NAND/NOR flash support as well as M68K system buses with a 16-bit wide data path.
XIO_D14	AA26	PCIT5V	I/O	33	-	
XIO_D13	AD25	PCIT5V	I/O	32	-	
XIO_D12	Y24	PCIT5V	I/O	31	-	
XIO_D11	Y25	PCIT5V	I/O	30	-	
XIO_D10	AC19	PCIT5V	I/O	29	-	
XIO_D09	AE26	PCIT5V	I/O	28	-	
XIO_D08	AC22	PCIT5V	I/O	27	-	
XIO_SEL4	AB24	PCIT5V	OUT	-	-	XIO Chip Selects. One is required per component for glue-less connections.
XIO_SEL3	AC23	PCIT5V	OUT	-	-	
XIO_SEL2	AD26	PCIT5V	OUT	-	-	
XIO_SEL1	AB25	PCIT5V	OUT	-	-	
XIO_SEL0	AB26	PCIT5V	OUT	-	-	
XIO_ACK	AC20	PCIT5V	IN	26	-	Flash/EEPROM acknowledge.
XIO_AD	AA24	PCIT5V	OUT	-	-	NOR Flash addressing.
Video/Data In Pin Group **This group provides ITU656 8-, 10- and 20-bit inputs, and up to 8-, 16- and 32-bit data streaming input.** **Refer to Section 7.1 on page 3-17 for a detailed definition of the operating modes of this pin group.**						
VDI_D33	AC5	BPTS3CHP	IN	52	D	Control for the streaming data mode.
VDI_D32	AE2	BPTS3CHP	IN	51	D	

Table 4: PNX1500 Interface

Pin Name	BGA Ball	Pad Type	I/O Type	GPIO #	P	Description
VDI_D31	AC14	BPTS3CHP	IN	-	U	Video or Streaming Parallel Data and control Inputs.
VDI_D30	AF12	BPTS3CHP	IN	-	U	
VDI_D29	AE12	BPTS3CHP	IN	-	U	
VDI_D28	AF11	BPTS3CHP	IN	-	U	
VDI_D27	AC13	BPTS3CHP	IN	-	U	
VDI_D26	AD11	BPTS3CHP	IN	-	U	
VDI_D25	AF10	BPTS3CHP	IN	-	U	
VDI_D24	AE10	BPTS3CHP	IN	-	U	
VDI_D23	AF9	BPTS3CHP	IN	-	U	
VDI_D22	AC12	BPTS3CHP	IN	-	U	
VDI_D21	AD10	BPTS3CHP	IN	-	U	
VDI_D20	AE9	BPTS3CHP	IN	-	U	
VDI_D19	AF8	BPTS3CHP	IN	-	U	
VDI_D18	AD9	BPTS3CHP	IN	-	U	
VDI_D17	AC11	BPTS3CHP	IN	-	U	
VDI_D16	AC10	BPTS3CHP	IN	-	U	
VDI_D15	AE7	BPTS3CHP	IN	-	U	
VDI_D14	AC9	BPTS3CHP	IN	-	U	
VDI_D13	AF6	BPTS3CHP	IN	-	U	
VDI_D12	AD8	BPTS3CHP	IN	-	U	
VDI_D11	AE8	BPTS3CHP	IN	-	U	
VDI_D10	AC8	BPTS3CHP	IN	-	U	
VDI_D09	AE5	BPTS3CHP	IN	-	U	
VDI_D08	AF5	BPTS3CHP	IN	-	U	
VDI_D07	AC7	BPTS3CHP	IN	-	U	
VDI_D06	AD7	BPTS3CHP	IN	-	U	
VDI_D05	AD6	BPTS3CHP	IN	-	U	
VDI_D04	AD5	BPTS3CHP	IN	-	U	
VDI_D03	AF4	BPTS3CHP	IN	-	U	
VDI_D02	AE3	BPTS3CHP	IN	-	U	
VDI_D01	AF3	BPTS3CHP	IN	-	U	
VDI_D00	AE4	BPTS3CHP	IN	-	U	
VDI_CLK1	AF7	BPX2T14MCP	I/O	-	U	A positive edge on this internally or externally generated clock samples video data. When generated internally, the clock can be software adjusted with sub one Hertz accuracy to allow generation of a precisely timed sequence of samples locked to an arbitrary reference, such as a broadcast transport stream source. A board level 27-33 Ω series resistor is recommended to reduce ringing.
VDI_V1	AF13	BPTS3CHP	IN	58	D	Data Valid clock qualifier associated with VDI_CLK1.

Table 4: PNX1500 Interface

Pin Name	BGA Ball	Pad Type	I/O Type	GPIO #	P	Description
VDI_CLK2	AC6	BPX2T14MCP	I/O	-	U	A positive edge on this internally or externally generated clock samples streaming data. When generated internally, the clock can be software adjusted with sub one Hertz accuracy to allow generation of a precisely timed sequence of samples locked to an arbitrary reference, such as a broadcast transport stream source. A board level 27-33 Ω series resistor is recommended to reduce ringing.
VDI_V2	AE1	BPTS3CHP	IN	59	D	Data Valid clock qualifier associated with VDI_CLK2.
Video/Data Out Pin Group						
The video mode provides ITU656 8-, 10- and 16-bit outputs, or digital 24-/30-bit HD YUV outputs, or digital 24-/30-bit RGB/VGA outputs. The data streaming mode provides 8-, 16-bit or 32-bit data streaming output. Refer to Section 7.1 on page 3-17 for a detailed definition of the operating modes of this pin group.						
VDO_D34	B2	BPTS1CHP	OUT	-	U	FGPO data bit 7 for extended mode. It was named RESERVED1.
VDO_D33	A19	BPTS1CHP	OUT	54	D	Control for Streaming Parallel Data Outputs.
VDO_D32	B18	BPTS1CHP	OUT	53	D	FGPO data bits [4:3] for extended mode.

Table 4: PNX1500 Interface

Pin Name	BGA Ball	Pad Type	I/O Type	GPIO #	P	Description
VDO_D31	C26	BPTS1CHP	OUT	-	U	Video and/or Streaming Parallel Data Outputs. VDO_D29 can be used as an input when QVCP is used in VSYNC slave mode.
VDO_D30	E26	BPTS1CHP	OUT	-	U	
VDO_D29	D20	BPTS1CHP	I/O	-	U	
VDO_D28	F24	BPTS1CHP	OUT	-	U	
VDO_D27	F25	BPTS1CHP	OUT	-	U	
VDO_D26	F26	BPTS1CHP	OUT	-	U	
VDO_D25	G24	BPTS1CHP	OUT	-	U	
VDO_D24	G25	BPTS1CHP	OUT	-	U	
VDO_D23	D19	BPTS1CHP	OUT	-	U	
VDO_D22	C25	BPTS1CHP	OUT	-	U	
VDO_D21	B26	BPTS1CHP	OUT	-	U	
VDO_D20	D22	BPTS1CHP	OUT	-	U	
VDO_D19	D21	BPTS1CHP	OUT	-	U	
VDO_D18	C23	BPTS1CHP	OUT	-	U	
VDO_D17	A26	BPTS1CHP	OUT	-	U	
VDO_D16	A25	BPTS1CHP	OUT	-	U	
VDO_D15	B24	BPTS1CHP	OUT	-	U	
VDO_D14	A24	BPTS1CHP	OUT	-	U	
VDO_D13	D17	BPTS1CHP	OUT	-	U	
VDO_D12	C22	BPTS1CHP	OUT	-	U	
VDO_D11	B23	BPTS1CHP	OUT	-	U	
VDO_D10	C21	BPTS1CHP	OUT	-	U	
VDO_D09	A23	BPTS1CHP	OUT	-	U	
VDO_D08	C20	BPTS1CHP	OUT	-	U	
VDO_D07	B22	BPTS1CHP	OUT	-	U	
VDO_D06	B21	BPTS1CHP	OUT	-	U	
VDO_D05	A22	BPTS1CHP	OUT	-	U	
VDO_D04	D16	BPTS1CHP	OUT	-	U	
VDO_D03	C19	BPTS1CHP	OUT	-	U	
VDO_D02	B20	BPTS1CHP	OUT	-	U	
VDO_D01	A21	BPTS1CHP	OUT	-	U	
VDO_D00	A20	BPTS1CHP	OUT	-	U	
VDO_CLK1	D18	BPX2T14MCP	I/O	-	U	A positive or negative edge on this internally or externally generated clock causes transitions of the video samples. When generated internally the clock can be software adjusted with sub one Hertz accuracy, to allow generation of a precisely timed sequence of samples locked to an arbitrary reference, such as a broadcast transport stream source. A board level 27-33 Ω series resistor is recommended to reduce ringing.

Table 4: PNX1500 Interface

Pin Name	BGA Ball	Pad Type	I/O Type	GPIO #	P	Description
VDO_CLK2	B19	BPX2T14MCP	I/O	-	U	A positive edge on this internally or externally generated clock causes transitions of the streaming data samples. When generated internally, the clock can be software adjusted with sub one Hertz accuracy to allow generation of a precisely timed sequence of samples locked to an arbitrary reference, such as a broadcast transport stream source. A board level 27-33 Ω series resistor is recommended to reduce ringing.
VDO_AUX	E24	BPTS1CHP	OUT	55	D	VDO_AUX can be programmed to output, a CBLANK signal, a Field indicator or a video/ graphics detector.
FGPO_REC_SYNC	C17	BPTS1CHP	I/O	60	D	Synchronization signal for Streaming Parallel Data Outputs. The FGPO data bit 5 is intended for the extended mode.
FGPO_BUF_SYNC	A18	BPTS1CHP	I/O	-	D	Synchronization signal for Streaming Parallel Data Outputs. The FGPO data bit 6 is intended for the extended mode.
Octal Audio In (audio in always acts as receiver, but can be set as master or slave for A/D timing)						
AI_OSCLK	AF23	BPX2T14MCP	OUT	-	U	Over-Sampling Clock. This output can be programmed to emit any frequency up to 50 MHz with a sub one Hertz resolution. It is intended to be used as the 256 f_s or 384 f_s over sampling clock by the external A/D subsystem. A board level 27-33 Ω series resistor is recommended to reduce ringing.
AI_SCK	AD20	BPX2T14MCP	I/O	-	U	AI can operate in either master or slave mode. • When Audio-In is programmed as the serial-interface timing slave (power-up default), AI_SCK is an input. AI_SCK receives the serial bit clock from the external A/D subsystem. This clock is treated as fully asynchronous to the PNX1500 main clock. • When Audio In is programmed as the serial-interface timing master, AI_SCK is an output. AI_SCK drives the serial clock for the external A/ D subsystem. The frequency is a programmable integral divide of the AI_OSCLK frequency. AI_SCK is limited to 25 MHz. The sample rate of valid samples embedded is variable. If used as a output, a board level 27-33 Ω series resistor is recommended to reduce ringing.

Table 4: PNX1500 Interface

Pin Name	BGA Ball	Pad Type	I/O Type	GPIO #	P	Description
AI_WS	AD21	BPTS3CHP	I/O	16	U	AI can operate in either master or slave mode. • When Audio In is programmed as the serial-interface timing slave (power-up default), AI_WS acts as an input. AI_WS is sampled on the same edge as selected for AI_SD[3:0]. • When Audio In is programmed as the serial-interface timing master, AI_WS acts as an output. It is asserted on the opposite edge of the AI_SD[3:0] sampling edge. AI_WS is the word-select or frame-synchronization signal from/to the external A/D subsystem.
AI_SD3	AD22	BPT3MCHDT5V	IN	20	D	Serial Data from external A/D subsystem. Data on this pin are sampled on positive or negative edge of AI_SCK as determined by the CLOCK_EDGE bit in the AI_SERIAL register. These pins are 5 V tolerant input.
AI_SD2	AC17	BPT3MCHDT5V	IN	19	D	
AI_SD1	AF24	BPT3MCHDT5V	IN	18	D	
AI_SD0	AE23	BPT3MCHDT5V	IN	17	D	
Octal Audio Out (audio out always acts as sender, but can be set as master or slave for D/A timing)						
AO_OSCLK	AD19	BPX2T14MCP	OUT	-	U	Over Sampling Clock. This output can be programmed to emit any frequency up to 50 MHz, with a sub one Hertz resolution. It is intended to be used as the 256 or 384 f_s over sampling clock by the external D/A conversion subsystem. A board level 27-33 Ω series resistor is recommended to reduce ringing.
AO_SCK	AE18	BPX2T14MCP	I/O	-	U	AO can operate in either master or slave mode. • When Audio Out is programmed to act as the serial interface timing slave (power up default), AO_SCK acts as input. It receives the Serial Clock from the external audio D/A subsystem. The clock is treated as fully asynchronous to the PNX1500 main clock. • When Audio Out is programmed to act as serial interface timing master, AO_SCK acts as output. It drives the Serial Clock for the external audio D/A subsystem. The clock frequency is a programmable integral divide of the AO_OSCLK frequency. AO_SCK is limited to 25 MHz. The sample rate of the valid samples is variable. If used as an output, a board level 27-33 Ω series resistor is recommended to reduce ringing.

Table 4: PNX1500 Interface

Pin Name	BGA Ball	Pad Type	I/O Type	GPIO #	P	Description
AO_WS	AE20	BPTS3CHP	I/O	21	U	AO can operate in either master or slave mode. • When Audio-Out is programmed as the serial-interface timing slave (power-up default), AO_WS acts as an input. AO_WS is sampled on the opposite AO_SCK edge at which AO_SD[3:0] are asserted. • When Audio Out is programmed as serial-interface timing master, AO_WS acts as an output. AO_WS is asserted on the same AO_SCK edge as AO_SD[3:0]. AO_WS is the word-select or frame-synchronization signal from/to the external D/A subsystem. Each audio channel receives 1 sample for every WS period.
AO_SD3	AF21	BPTS3CHP	OUT	25	U	Serial Data to external audio D/A subsystem for first 2 of 8 channels. The timing of the transitions on these outputs is determined by the CLOCK_EDGE bit in the AO_SERIAL register, and can be on a positive or negative AO_SCK edge.
AO_SD2	AF20	BPTS3CHP	OUT	24	U	
AO_SD1	AE19	BPTS3CHP	OUT	23	U	
AO_SD0	AF19	BPTS3CHP	OUT	22	U	
SPDIF interface						
SPDI	A6	BPT3MCHDT5V	IN	56	D	Input for SPDIF (Sony/Philips Digital Audio Interface, a.k.a. Dolby Digital™), a self clocking audio data stream as per IEC958 with 1937 extensions. This pin is 5 V tolerant input.
SPDO	AF22	BPX2T14MCP	OUT	57	U	Output for SPDIF. Note that this low-impedance driver requires a 27-33 Ω resistor close to the PNX1500 to match the board line impedance. This resistor becomes a part of the voltage divider necessary to drive the IEC958 isolation transformer.
10/100 LAN interface (MII)						
LAN_CLK	AF18	BPTS1CP	OUT	-	U	Clock to feed the external PHY, usually 50 MHz.
LAN_TX_CLK/ LAN_REF_CLK	AF14	BPTS3CP	IN	-	U	MII Transmit clock or RMII reference clock. Both LAN_TX_CLK and LAN_RX_CLK have to be connected to the RMII reference clock in RMII mode.
LAN_TX_EN	AD13	BPTS3CHP	OUT	35	D	MII or RMII Transmit Enable
LAN_TXD3	AF15	BPTS3CHP	OUT	39	D	MII Transmit Data
LAN_TXD2	AD14	BPTS3CHP	OUT	38	D	MII Transmit Data
LAN_TXD1	AC15	BPTS3CHP	OUT	37	D	MII or RMII Transmit Data
LAN_TXD0	AE14	BPTS3CHP	OUT	36	D	MII or RMII Transmit Data
LAN_TX_ER	AE13	BPTS3CHP	OUT	40	D	MII Transmit Error
LAN_CRS/ LAN_CRS_DV	AC24	BPT3MCHDT5V	IN	41	D	MII Carrier Sense or RMII Carrier Sene and Receive Data Valid. This pin is 5 V tolerant input.
LAN_COL	AA23	BPT3MCHDT5V	IN	42	D	Collision Detect. This pin is 5 V tolerant input.

Table 4: PNX1500 Interface

Pin Name	BGA Ball	Pad Type	I/O Type	GPIO #	P	Description
LAN_RX_CLK/ LAN_REF_CLK	AF16	BPTS3CP	IN	-	U	MII Receive Clock. Both LAN_TX_CLK and LAN_RX_CLK have to be connected to the RMII reference clock in RMII mode.
LAN_RXD3	AD17	BPTS3CHP	IN	46	U	MII Receive Data
LAN_RXD2	AD16	BPTS3CHP	IN	45	U	MII Receive Data
LAN_RXD1	AF17	BPTS3CHP	IN	44	U	MII or RMII Receive Data
LAN_RXD0	AE16	BPTS3CHP	IN	43	U	MII or RMII Receive Data
LAN_RX_DV	AE15	BPTS3CHP	IN	47	U	MII Receive Data Valid.
LAN_RX_ER	AD15	BPTS3CHP	IN	48	D	MII or RMII Receive Error.
LAN_MDIO	AC26	BPTS3CHP	I/O	49	U	MII Management data I/O.
LAN_MDC	AC25	BPTS3CHP	OUT	50	U	MII Management Data clock.
I^2C Interface						
IIC_SDA	C8	IIC3M4SDAT5V	I/OD	-	-	I^2C serial data. This pin is 5 V tolerant input.
IIC_SCL	D8	IIC3M4SCLT5V	I/OD	-	-	I^2C clock. This pin is 5 V tolerant input.
GPIO - Multi-function flexible software I/O and universal serial interface						
Each GPIO pin can be individually set/read by software, or connected to a DMA engine that makes it function as a serial pattern generator or serial observer, so that the software can implement complex bit serial I/O protocols. Typically, it is used for the IR receiver, IR blaster, switches, lights and serial communications protocols. In addition, any pin with an entry in the GPIO column of this pin list can be (individually) set to act as a GPIO pin instead of for its primary function. After power-on reset, every GPIO is set to the input mode to avoid any potential electrical conflict on the board.						
GPIO15/WAKEUP	AC21	BPT3MCHDT5V	I/O/D	15	D	Used as a GPIO pin. This pin can also be used as the wake-up event once the PNX1500 has been sent into deep power down mode. This pin is 5 V tolerant input.
GPIO14/GCLOCK02	AE22	BPTS1CHP	I/O/D	14	U	Used as GPIO pins. These pins can also be used to output internally generated clocks for external components present on the board (Section 2.11.1 on page 5-20). GPIO12/GCLOCK00 requires a board level 27-33 Ω series resistor to reduce ringing.
GPIO13/GCLOCK01	AE21	BPTS1CHP	I/O/D	13	U	
GPIO12/GCLOCK00	AC16	BPX2T14MCP	I/O/D	12	U	
GPIO11/ BOOT_MODE07	AC18 -	BPT3MCHT5V -	I/O/D -	11 -	- -	After the power up and boot sequence, these pins are used as GPIO[11:8] pins. These GPIO pins must be strapped with resistors to VDD or VSS to determine the PNX1500 boot mode upon reset.
GPIO10/ BOOT_MODE06	AD23 -	BPT3MCHT5V -	I/O/D -	10 -	- -	
GPIO09/ BOOT_MODE05	AF26 -	BPT3MCHT5V -	I/O/D -	9 -	- -	GPIO[11:10] pins can also be used as input external interrupt lines for the TM3260. The software can assert at regular intervals the WDOG_OUT output pin to prevent an external watchdog device to reset the entire system. Other GPIO pins can be used for that feature. These pins are 5 V tolerant input.
GPIO08/ BOOT_MODE04/ WDOG_OUT	AF25	BPT3MCHT5V -	I/O/D -	8 - -	- - -	
GPIO7	AE24	BPT3MCHDT5V	I/O/D	7	D	Used as a GPIO pin. This pin is 5 V tolerant input.

Table 4: PNX1500 Interface

Pin Name	BGA Ball	Pad Type	I/O Type	GPIO #	P	Description
GPIO06/CLOCK06	B9	BPTS1CHP	I/O/D	6	U	Used as GPIO pins. These pins can also be used to output internally generated clocks for the external components present on the board. These GPIO pins can also be used as clocks for sampling or pattern generation in the GPIO module (Section 2.11.2 on page 5-20). GPIO05/ GCLOCK05 requires a board level 27-33 Ω series resistor to reduce ringing.
GPIO05/CLOCK05	A8	BPX2T14MCP	I/O/D	5	U	
GPIO04/CLOCK04	A7	BPTS1CHP	I/O/D	4	U	
GPIO03/CLOCK03/ BOOT_MODE03	A4	BPTS1CHP	I/O/D	3	D	After the power up and boot sequence, this pin functions as a GPIO[3] pin. This pin can also be used as a clock for sampling or pattern generation in the GPIO module. This GPIO pin may be strapped with a resistor to VDD or VSS to determine the PNX1500 boot mode upon reset.
GPIO02/CLOCK02/ BOOT_MODE02	A3 -	BPTS1CHP -	I/O/D -	2 -	U -	After the power up and boot sequence, these pins are configured as GPIO[2:0] pins. These pins can also be used as clocks for sampling or pattern generation in the GPIO module. These GPIO pins may be strapped with resistors to VDD or VSS to determine the PNX1500 boot mode upon reset.
GPIO01/CLOCK01/ BOOT_MODE01	B3 -	BPTS1CHP -	I/O/D -	1 -	U -	
GPIO00/CLOCK00/ BOOT_MODE00	B4 -	BPTS1CHP -	I/O/D -	0 -	U -	
JTAG Interface (debug access port and 1149.1 boundary scan port)						
JTAG_TDI	A1	IPCHP	IN	-	U	JTAG Test Data Input.
JTAG_TDO	D6	BPTS3CHP	O	-	-	JTAG Test Data Output. This pin can either be an output, or float. It is never an input.
JTAG_TCK	B1	IPCP	IN	-	U	JTAG Test Clock Input.
JTAG_TMS	D5	IPCHP	IN	-	U	JTAG Test Mode Select Input.
Power Supplies and Ground **Refer to Section 10. on page 1-43 for board level connection and decoupling associated with these pins.**						
VDDA	A10	APOD	PWR	-	-	Analog, quiescent VDD, 1.2-1.3 V. Refer to Figure 25 for board level connections.
VSSA_1.2	C11	APOD	GND	-	-	Analog, quiescent ground for the 1.2 V analog supply. Refer to Figure 25 for board level connections.
VCCA[]	-	APOD	PWR	-	-	Analog, quiescent VCCP, 3.3 V. Refer to Figure 24 for board level connections.
VSSA[]	-	APOD	GND	-	-	Analog, quiescent ground. Refer to Figure 24 for board level connections.
VCCP[]	-	VDDE3V3	PWR	-	-	3.3 V I/O power supply for peripherals I/Os.
VCCM[]	-	VDDE3V3	PWR	-	-	2.5 V power supply for the memory I/Os (3.3 V capable of ATE, **not** for functional operation).
VDD[]	-	VDDI	PWR	-	-	1.2-1.3 V core power supply.
VSS[]	-	VSSIS	GND	-	-	Ground for the core.
VSS[]	-	VSSE	GND	-	-	Ground for the memory I/Os.
VSS[]	-	VSSE	GND	-	-	Ground for the peripherals I/Os.

2.3.1 Power Pin List

Table 5: Power Pin List

Digital Ground			3.3-V	2.5-V	1.2-1.3-V	Analog 3.3-V		Analog 1.2-1.3-V	
VSS			VCCP	VCCM	VDD	VSSA	VCCA	VSSA_1.2	VDDA
T11	N11	V5	AB7	Y5	E10	B15	D15	C11	A10
T12	N12	U5	AB8	T5	E15	B13	C13		
T13	N13	T2	AB13	R5	E16	B16	A16		
T14	N14	M3	AB14	U1	E9	A14	D14		
T15	N15	H3	P22	R3	AB10	A5	D7		
T16	N16	J5	N22	N1	AB15	B8	B5		
R11	M11	F4	E13	M5	AB16				
R12	M12	F5	E14	L5	AB9				
R13	M13	V22	E7	J3	T22				
R14	M14	U22	E8	G2	R22				
R15	M15	M22	AF2	H5	P5				
R16	M16	L22	E19	G5	N5				
P11	L11	AA5	E20	W5	K22				
P12	L12	AA4	C12	D2	J22				
P13	L13	AF1	C18	D4	A15				
P14	L14	F22	C24	AC4	B14				
P15	L15	E11	B6	AC2	D13				
P16	L16	E12	A2	Y2	C10				
E5	W1	E17	AE6	V2	C6				
E6	W3	E18	AD12	L2	C9				
C4	U25	E21	AD18	K1	D12				
B11	P2	E22	AD24		A12				
B17	P25	AB5	AB19		A17				
B25	K25	AB6	AB20						
AE11	K5	AA22	Y22						
AE17	H1	AB11	W22						
AE25	AB18	AB12	V24						
AD4	AB21	AB17	J24						
C15	AB22	C14	H22						
A13	C16	B12	G22						
A9	C5	B7							
		B10							

Remark: The digital ground for the signals and clocks comes from the same digital ground plane.

Remark: The digital 1.2 V power supply for the signals and clocks comes from the same digital power plane.

2.3.2 Pin Reference Voltage

Table 6: Pin Reference Voltage

3.3 V Input and/or Output 5.0 V Input Tolerant		3.3 V Input and/or Output			2.5 V Only		Special
POR_IN_N	PCI_AD31	PCI_SYS_CLK	AI_OSCLK	VDI_D33	MM_CLK	MM_DATA31	XTAL_IN
RESET_IN_N	PCI_AD30	SYS_RST_OUT_N	AI_SCK	VDI_D32	MM_CLK_N	MM_DATA30	XTAL_OUT
PCI_CLK	PCI_AD29	VDO_CLK1	AI_WS	VDI_D31	MM_CKE1	MM_DATA29	
PCI_C/BE03	PCI_AD28	VDO_CLK2	AO_OSCLK	VDI_D30	MM_CKE2	MM_DATA28	
PCI_C/BE2	PCI_AD27	VDO_D33	AO_SCK	VDI_D29	MM_DQS3	MM_DATA27	
PCI_C/BE1	PCI_AD26	VDO_D32	AO_WS	VDI_D28	MM_DQS2	MM_DATA26	
PCI_C/BE0	PCI_AD25	VDO_D31	AO_SD3	VDI_D27	MM_DQS1	MM_DATA25	
PCI_PAR	PCI_AD24	VDO_D30	AO_SD2	VDI_D26	MM_DQS0	MM_DATA24	
PCI_FRAME_N	PCI_AD23	VDO_D29	AO_SD1	VDI_D25	MM_ADDR12	MM_DATA23	
PCI_IRDY_N	PCI_AD22	VDO_D28	AO_SD0	VDI_D24	MM_ADDR11	MM_DATA22	
PCI_TRDY_N	PCI_AD21	VDO_D27	SPDO	VDI_D23	MM_ADDR10	MM_DATA21	
PCI_STOP_N	PCI_AD20	VDO_D26	LAN_CLK	VDI_D22	MM_ADDR09	MM_DATA20	
PCI_IDSEL	PCI_AD19	VDO_D25	LAN_TX_CLK	VDI_D21	MM_ADDR08	MM_DATA19	
PCI_DEVSEL_N	PCI_AD18	VDO_D24	LAN_TX_EN	VDI_D20	MM_ADDR07	MM_DATA18	
PCI_REQ_N	PCI_AD17	VDO_D23	LAN_TDX03	VDI_D19	MM_ADDR06	MM_DATA17	
PCI_GNT_N	PCI_AD16	VDO_D22	LAN_TDX02	VDI_D18	MM_ADDR05	MM_DATA16	
PCI_REQ_A_N	PCI_AD15	VDO_D21	LAN_TDX01	VDI_D17	MM_ADDR04	MM_DATA15	
PCI_GNT_A_N	PCI_AD14	VDO_D20	LAN_TDX00	VDI_D16	MM_ADDR03	MM_DATA14	
PCI_REQ_B_N	PCI_AD13	VDO_D19	LAN_TX_ER	VDI_D15	MM_ADDR02	MM_DATA13	
PCI_GNT_B_N	PCI_AD12	VDO_D18	LAN_RX_CLK	VDI_D14	MM_ADDR01	MM_DATA12	
PCI_PERR_N	PCI_AD11	VDO_D17	LAN_RXD3	VDI_D13	MM_ADDR00	MM_DATA11	
PCI_SERR_N	PCI_AD10	VDO_D16	LAN_RXD2	VDI_D12	MM_BA1	MM_DATA10	
PCI_INTA_N	PCI_AD09	VDO_D15	LAN_RXD1	VDI_D11	MM_BA0	MM_DATA09	
XIO_ACK	PCI_AD08	VDO_D14	LAN_RXD0	VDI_D10	MM_CS1_N	MM_DATA08	
XIO_D15	PCI_AD07	VDO_D13	LAN_MDIO	VDI_D09	MM_CS0_N	MM_DATA07	
XIO_D14	PCI_AD06	VDO_D12	LAN_MDC	VDI_D08	MM_RAS_N	MM_DATA06	
XIO_D13	PCI_AD05	VDO_D11	LAN_RX_DV	VDI_D07	MM_CAS_N	MM_DATA05	
XIO_D12	PCI_AD04	VDO_D10	LAN_RX_ER	VDI_D06	MM_WE_N	MM_DATA04	
XIO_D11	PCI_AD03	VDO_D09	GPIO14	VDI_D05	MM_DQM3	MM_DATA03	
XIO_D10	PCI_AD02	VDO_D08	GPIO13	VDI_D04	MM_DQM2	MM_DATA02	
XIO_D09	PCI_AD01	VDO_D07	GPIO12	VDI_D03	MM_DQM1	MM_DATA01	
XIO_D08	PCI_AD00	VDO_D06	GPIO06	VDI_D02	MM_DQM0	MM_DATA00	
XIO_SEL4	GPIO15	VDO_D05	GPIO05	VDI_D01			
XIO_SEL3	GPIO11	VDO_D04	GPIO04	VDI_D00			
XIO_SEL2	GPIO10	VDO_D03	GPIO03	VDI_V1			
XIO_SEL1	GPIO09	VDO_D02	GPIO02	VDI_V2			
XIO_SEL0	GPIO08	VDO_D01	GPIO01				
XIO_AD	GPIO07	VDO_D00	GPIO00				
LAN_CRS	SPDI	VDO_AUX	JTAG_TDI				
LAN_COL	AI_SD3	FGPO_REC_SYNC	JTAG_TCK				
IIC_SDA	AI_SD2	FGPO_BUF_SYNC	JTAG_TMS				
IIC_SCL	AI_SD1	VDO_D34	JTAG_TDO				
RESERVED	AI_SD0		VDI_CLK1				
			VDI_CLK2				

3. Parametric Characteristics

3.1 Absolute Maximum Ratings

Permanent damage may occur if absolute maximum ratings are exceeded. Prolonged

operation above the operation range described in Section 3.2 but below the maximum ratings may significantly reduce the reliability of the PNX1500.

Table 7: Absolute Maximum Ratings

Symbol	Description	Minimum	Maximum	Units	Note
V_{CCP}	3.3 V I/O supply voltage	-0.5	4.6	V	
V_{CCM}	2.5 V DDR I/O supply voltage	-0.5	3.6	V	
V_{DD}	1.2-1.3-V Core supply voltage	-0.5	1.7	V	
V_{ICCP}	Input voltage for 5 V tolerant input pins (i.e. pins supplied by V_{CCP})	-0.5	6.0	V	
T_{stg}	Storage temperature range	-65	150	˚C	
T_{case}	Operating case temperature range	0	120	˚C	
HBM_{ESD}	Human Body Model Electrostatic handling for all pins	-	2000	V	[1]
MM_{ESD}	Machine Model Electrostatic handling for all pins	-	150	V	[2]

[1] CLASS 2, JEDEC Standard, June 2000

[2] CLASS A, JEDEC Standard, October 1997

3.2 Operating Range and Thermal Characteristics

Functional operation, long-term reliability and AC/DC characteristics are guaranteed for the operating conditions described in the following table.

Table 8: Operating Range and Thermal Characteristics

Symbol	Description	Minimum	Typical	Maximum	Units
V_{CCP}	Global I/O supply voltage	3.13	3.30	3.47	V
V_{CCM}	DDR I/O supply voltage. Up to DDR333 SDRAMs require 2.5V. At DDR400 most SDRAMs require 2.6 V	2.37	2.50 – 2.60	2.73	V
V_{DD}	Core supply voltage	1.14 – 1.23	1.2 – 1.3	1.26 – 1.37	V
V_{REF}	Input reference level voltage for the DDR I/Os. V_{CCM}/2 +/- 100 mV	1.08 – 1.13	1.25 – 1.3	1.35 – 1.47	V
T_{case}	Operating case temperature range	0	-	85	˚C
Ψ_{jt}	Junction to case thermal resistance	-	6.1	-	˚C/W

4. Power Supplies Sequence

No special power sequence is required to operate the PNX15xx Series. However, in order to guarantee that MM_CKE remains low at power up, the PNX1500 is required to have the 1.2V to come-up before the 2.5V. This is a JEDEC DDR specification requirement.

Remark: DDR SDRAM devices power supply sequence must also be met. Refer to the DDR SDRAM vendor specification.

5. Power Supply and Operating Speeds

Table 9: Maximum Operating Speeds Based on the VDD Power Supply for the PNX1501

Core Supply (V)	TM3260 (MHz)	MMI (MHz)	MMIO (MHz)	2DDE MBS VLD (MHz)	QVCP (out, proc) (MHz)	VIP (MHz)	FGPO FGPI (MHz)	DVDD (MHz)	PCI-XIO (MHz)	LAN (MHz)	AO AI (MHz)	SPDO (MHz)	GPIO (MHz)
1.2	266	200	157	123	81, 96	81	100	78	33	30	25	40	108

Table 10: Maximum Operating Speeds Based on the VDD Power Supply for the PNX1502

Core Supply (V)	TM3260 (MHz)	MMI (MHz)	MMIO (MHz)	2DDE MBS VLD (MHz)	QVCP (out, proc) (MHz)	VIP (MHz)	FGPO FGPI (MHz)	DVDD (MHz)	PCI-XIO (MHz)	LAN (MHz)	AO AI (MHz)	SPDO (MHz)	GPIO (MHz)
1.3	300	200	157	123	81, 96	81	100	78	33	30	25	40	108

6. Power Consumption

The power consumption of the PNX15xx Series is dependent on the activity of the TM3260, the number of modules operating, the frequencies at which the system is running, the core voltage, as well as the loads at board level on each pin.

6.1 Leakage current Power Consumption

Leakage current is a new variable of the advanced CMOS processes. The resultant power dissipation is at most 220 mW. It includes the 3 different power supplies. The main part of it is coming from the core power supply, i.e. 1.2 V or 1.3 V with 200 mW out of the maximum 220 mW. This leakage current is variable from silicon to silicon. This means that two PNX1502 can have a significant different leakage current, e.g. as low as 50 mW and as high as 220 mW. The resultant thermal effect must be taken into consideration while designing a board with PNX1500/01/02.

6.2 Standby Power Consumption

During the standby (sleep) mode, all the clocks of the PNX1500 system are turned off. A small amount of logic stays alive in order to wake-up the system.

The standby mode is obtain by specifically turning off the different clocks, i.e. it is not just a simple bit to flip into a register. Once all the clocks have been shutdown the power dissipation is at most 300 mW.

6.3 Typical Power Consumption for Typical Applications

Three main techniques can be applied to reduce the ‘Out of the Box’ power consumption of the PNX1500 system:

- Turn off the unused modules. After reset, the modules are clocked with a 27 MHz clock (input crystal clock, XTAL_IN). Turning off the clocks of the unused modules significantly reduces the power consumption.
- Run the PNX1500 system with the adjusted clock speeds for each active module.

Powerdown the TM3260 every time the OS (Operating System) reaches the idle task.

Table 11 and Table 12 present a typical example (not optimized for power consumption savings).

Table 11: MPEG-2 Decoding with 720x480P Output on PNX1501

PNX1501	1.2V	2.6V	3.3V	Total
mA	990	104	53	n/a
W	1.188	0.270	0.175	1.634

Table 12: MPEG-2 Decoding with 720x480P Output on PNX1502

PNX1502	1.3V	2.6V	3.3V	Total
mA	1002	104	53	n/a
W	1.302	0.270	0.175	1.747

6.4 Expected Maximum Currents

Table 13 and Table 14 presents estimated maximum currents, i.e. all modules operating at full speed which is not what a real application will do. Board design, i.e. decoupling and regulators, should plan for peak current. Peak currents are possible for few cycles it is not sustained current consumption. These peaks will be averaged out by the decoupling capacitors, but regulators should also not be under-dimensioned.

Table 13: Estimated PNX1501 Maximum and Peak current

PNX1501	1.2V	2.6V	3.3V
Maximum, mA	1400	300	200
Peak, mA	2000	500	300

Table 14: Estimated PNX1502 Maximum and Peak current

PNX1502	1.3V	2.6V	3.3V
Maximum, mA	1400	300	200
Peak, mA	2000	500	300

7. DC/AC I/O Characteristics

The characteristics listed in the following tables apply to the worst case operating condition defined in Section 3.2 on page 1-20. All voltages are referenced to VSS (0 V digital ground). The following I/O characteristics includes the effect of process variation.

7.1 Input Crystal Specification

Table 15: Specification of HC-49U 27.00000 MHZ Crystal

Frequency	27.00000 MHZ fundamental
Temperature range	0˚C to 85˚C
Capacitive load (CL)	18pF-20pF
Frequency accuracy (all included: temperature, aging, frequency at 0 to 85˚C)	+/-30 ppm
Series resonance resistor	40 Ω max.
Shunt capacitance	7pF max.
Drive level	1mW max.
Motion capacitance	20fF maximum. (as low as possible).

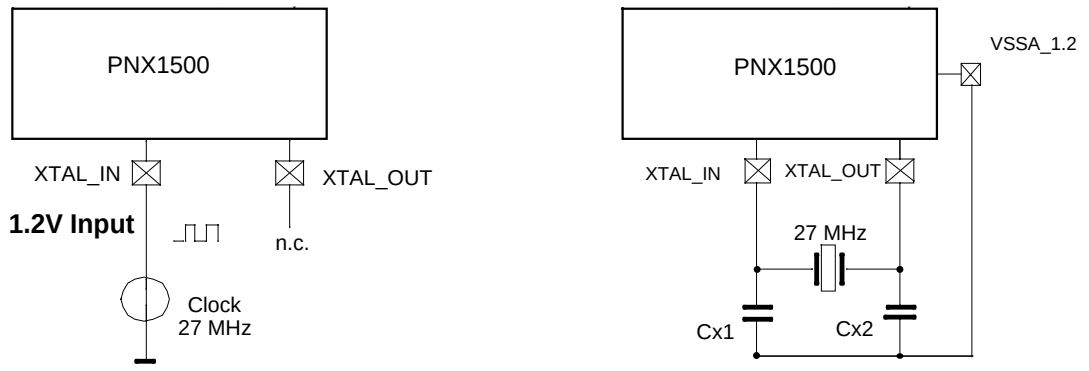


Figure 1: Application Diagram of the Crystal Oscillator

7.2 SSTL_2 type I/O Circuit

Table 16: SSTL_2 AC/DC Characteristics

Symbol	Parameter	Condition/Notes	Min	Typ	Max	Unit	Notes
V_{OH}	Output High Voltage		$0.9V_{CCM}$			V	
V_{OL}	Output Low Voltage				$0.1V_{CCM}$	V	
V_{IH}	DC Input High Voltage	This is the overshoot/ undershoot protection specification of the pad			V_{CCM} + 0.3	V	
V_{IL}	DC Input Low Voltage		-0.3			V	
V_{IH-DC}	DC Input High Voltage	Logic Threshold			V_{REF} + 0.18	V	
V_{IL-DC}	DC Input Low Voltage	Logic Threshold	V_{REF} - 0.18			V	
V_{IH-AC}	AC Input High Voltage	Used for timing specification. See Figure 3.	V_{REF} + 0.35			V	
V_{IL-AC}	AC Input Low Voltage	Used for timing specification. See Figure 3.			V_{REF} - 0.35	V	

Table 16: SSTL_2 AC/DC Characteristics

Symbol	Parameter	Condition/Notes	Min	Typ	Max	Unit	Notes
R_{SSTL}	Series Output Resistance	High/Low level output state	30	40	50	Ω	1
T_{SLEW}	Slew rate, $(V_{IH-AC} - V_{IL-AC})/dt$	Refer to Figure 2 and Figure 3.	0.3	0.4	0.5	V/ns	
C_{IN}	Input pin capacitance				5	pF	

[16-1] Notes:

[16-2] 1. Measured into 50 Ω load terminated to $V_{CCM}/2$.

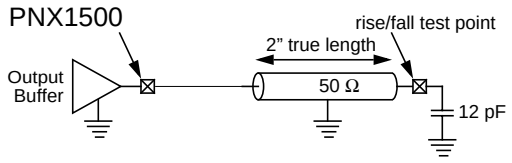


Figure 2: SSTL_2 Test Load Condition

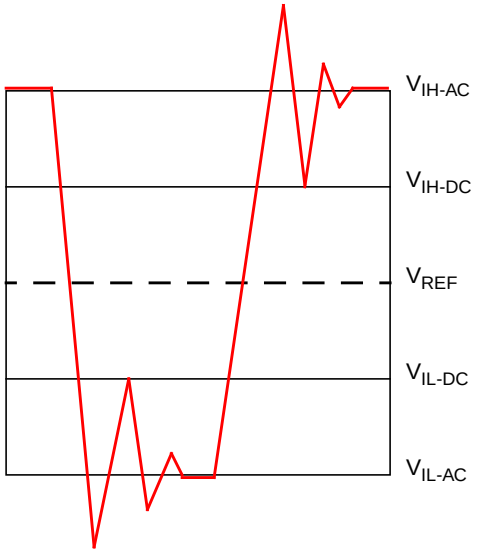


Figure 3: SSTL_2 Receiver Signal Conditions

7.3 BPX2T14MCP Type I/O Circuit

Table 17: BPX2T14MCP Characteristics

Symbol	Parameter	Condition/Notes	Min	Typ	Max	Unit
V_{OH}	Output High Voltage		$0.9V_{CCP}$			V
V_{OL}	Output Low Voltage				$0.1V_{CCP}$	V
V_{IHT}	DC Input High Voltage	Logic Threshold			2.0	V
V_{ILT}	DC Input Low Voltage	Logic Threshold	0.8			V
V_{IH}	DC Input High Voltage	This is the overshoot/ undershoot protection specification of the pad			V_{CCP} + 0.3	V
V_{IL}	DC Input Low Voltage		-0.3			V
Z_O	Output AC Impedance	High/Low level output state		22		Ω
Pull	Pull-up/down Resistor	If applicable	38	66	165	KΩ
C_{IN}	Input pin capacitance				6	pF

BPX2T14MCP I/Os require a board level 27-33 Ω series resistor to reduce ringing.

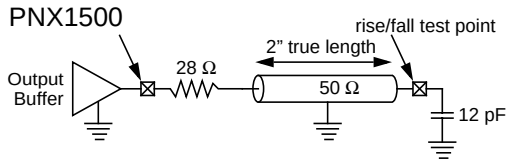


Figure 4: BPX2T14MCP Test Load Condition

7.4 BPTS1CHP and BPTS1CP Type I/O Circuit

Table 18: BPTS1CHP and BPTS1CP Characteristics

Symbol	Parameter	Condition/Notes	Min	Typ	Max	Unit
V_{OH}	Output High Voltage		$0.9V_{CCP}$			V
V_{OL}	Output Low Voltage				$0.1V_{CCP}$	V
V_{IHT}	DC Input High Voltage	Logic Threshold			2.0	V
V_{ILT}	DC Input Low Voltage	Logic Threshold	0.8			V
V_{IH}	DC Input High Voltage	This is the overshoot/ undershoot protection specification of the pad			V_{CCP} + 0.3	V
V_{IL}	DC Input Low Voltage		-0.3			V
Z_O	Output AC Impedance	High/Low level output state		38		Ω
T_{RF}	Output Rise/Fall Time	Test Load in Figure 5	1.2	1.6	2.0	ns
Pull	Pull-up/down Resistor	If applicable	38	66	165	KΩ
C_{IN}	Input pin capacitance				6	pF

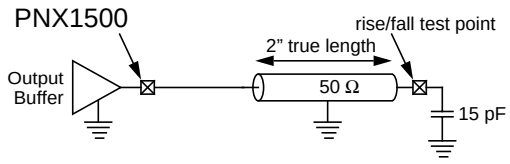


Figure 5: BPTS1CHP and BPTS1CP Test Load Condition

7.5 BPTS3CHP Type I/O Circuit

Table 19: BPTS3CHP Characteristics

Symbol	Parameter	Condition/Notes	Min	Typ	Max	Unit
V_{OH}	Output High Voltage		$0.9V_{CCP}$			V
V_{OL}	Output Low Voltage				$0.1V_{CCP}$	V
V_{IHT}	DC Input High Voltage	Logic Threshold			2.0	V
V_{ILT}	DC Input Low Voltage	Logic Threshold	0.8			V
V_{IH}	DC Input High Voltage	This is the overshoot/ undershoot protection specification of the pad			V_{CCP} + 0.3	V
V_{IL}	DC Input Low Voltage		-0.3			V
Z_O	Output AC Impedance	High/Low level output state		45		Ω
T_{RF}	Output Rise/Fall Time	Test Load in Figure 6	3.0	4.0	5.0	ns
Pull	Pull-up/down Resistor	If applicable	38	66	165	KΩ
C_{IN}	Input pin capacitance				6	pF

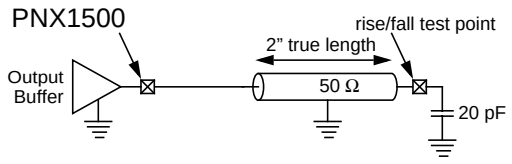


Figure 6: BPTS3CHP Test Load Condition

7.6 IPCHP and IPCP Type I/O Circuit

Table 20: IPCHP and IPCP Characteristics

Symbol	Parameter	Condition/Notes	Min	Typ	Max	Unit
V_{IHT}	DC Input High Voltage	Logic Threshold			2.0	V
V_{ILT}	DC Input Low Voltage	Logic Threshold	0.8			V
V_{IH}	DC Input High Voltage	This is the overshoot/ undershoot protection specification of the pad			5.3	V
V_{IL}	DC Input Low Voltage		-0.3			V
Pull	Pull-up/down Resistor	If applicable	38	66	165	KΩ
C_{IN}	Input pin capacitance				5	pF

7.7 BPT3MCHDT5V and BPT3MCHT5V Type I/O Circuit

Table 21: BPT3MCHDT5V and BPT3MCHT5V Characteristics

Symbol	Parameter	Condition/Notes	Min	Typ	Max	Unit
V_{OH}	Output High Voltage		$0.9V_{CCP}$			V
V_{OL}	Output Low Voltage				$0.1V_{CCP}$	V
V_{IHT}	DC Input High Voltage	Logic Threshold			2.0	V
V_{ILT}	DC Input Low Voltage	Logic Threshold	0.8			V
V_{IH}	DC Input High Voltage	This is the overshoot/ undershoot protection specification of the pad			5.5	V
V_{IL}	DC Input Low Voltage		-0.3			V
Z_O	Output AC Impedance	High/Low level output state		60		Ω
T_{RF}	Output Rise/Fall Time	Test Load in Figure 7	3.0	4.0	5.0	ns
Pull	Pull-up/down Resistor	If applicable	38	66	165	KΩ
C_{IN}	Input pin capacitance				6	pF

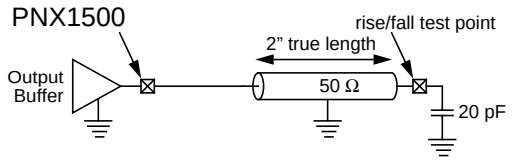


Figure 7: BPT3MCHDT5V and BPT3MCHT5V Test Load Condition

7.8 IIC3M4SDAT5V and IIC3M4SCLT5V type I/O circuit

Table 22: IIC3M4SDAT5V and IIC3M4SCLT5V Characteristics

Symbol	Parameter	Condition/Notes	Min	Typ	Max	Unit
V_{IH}	Input High Voltage		2.3		5.5	V
V_{IL}	Input Low Voltage		-0.5		1.0	V
V_{HYS}	Input Schmitt trigger Hysteresis		0.25			V
V_{OL}	Output Low Voltage				0.6	V
T_F	Output Fall Time	10 - 400 pF	1.5		250	ns
C_{IN}	Input pin capacitance				6	pF

7.9 PCIT5V type I/O circuit

Table 23: PCIT5V Characteristics

Symbol	Parameter	Condition/Notes	Min	Typ	Max	Unit
V_{IH-5V}	Input High Voltage		2.0		5.75	V
V_{IL-5V}	Input Low Voltage		-0.5		0.8	V
V_{IH-3V}	Input High Voltage		1.5		5.75	V
V_{IL}	Input Low Voltage		-0.5		1.08	V
V_{OH}	Output High Voltage		2.7			V
V_{OL}	Output Low Voltage				0.55	V
T_{RF}	Output Fall Time	Between 0.2 V_{CCP} and 0.6 V_{CCP}	1.3			ns
C_{IN}	Input pin capacitance			6	8	pF

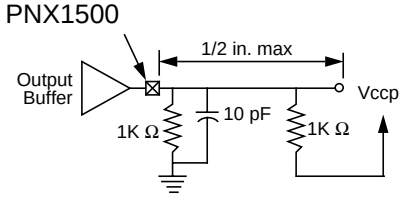


Figure 8: PCI T_{val}(min) and Slew Rate Test Load Condition

8. Timing Specification

The characteristics listed in the following tables apply to the worst case operating condition defined in Section 3.2 on page 1-20. The following I/O characteristics includes the effect of process variation.

8.1 Reset

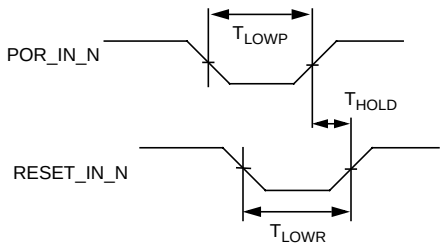


Figure 9: Reset Timing

Table 24: Reset Timing

Symbol	Parameter	Min	Max	Units	Notes
T_{LOWP}	Reset active time after power and clock stable	100		µs	1
T_{HOLD}	Reset active after POR_IN_N is pulled high	0		ns	2
T_{LOWR}	Reset active time after power and clock stable	100		µs	1

[24-1] Notes:

[24-2] 1. Can be asserted and de-asserted asynchronously with respect to CLK.

[24-3] 2. If POR_IN_N and RESET_IN_N are asserted low then RESET_IN_N must stay low for at least as long as POR_IN_N is asserted low.

8.2 DDR DRAM Interface

PNX1500 supports DDR200, DDR266, DDR400{A,B,C} DDR devices as defined in the JEDEC STANDARD JESD79C, March 2003. Refer to Section 10.3 DDR SDRAM interface for more details.

Table 25: DDR DRAM Interface Timing

Symbol	Parameter	Min	Max	Units	Notes
F_{ddr}	MM_CLK and MM_CLK_N frequency	83	200	MHz	i.e. up to DDR400
T_{ddr}	MM_CLK and MM_CLK_N period	5	12	ns	
T_{cs}	MM_CLK and MM_CLK_N skew		0.01	ns	
$T_{pd\text{-}cmd}$	Propagation delay for command signals	1.4	3.6	ns	1, 2, 3, 5
$T_{s\text{-}dq}$	Setup time for MM_DQ and MM_DQM (when writing to DDR SDRAM)	- 0.12		T_{ddr}	4, 5
$T_{oh\text{-}dq}$	Output hold time for MM_DQ and MM_DQM (when writing to DDR SDRAM)	0.12		T_{ddr}	4, 5

Table 25: DDR DRAM Interface Timing

Symbol	Parameter	Min	Max	Units	Notes
$T_{iskew-dqs}$	Maximum input skew supported (when reading from DDR SDRAM)	0.2	1.8	ns	2, 5
T_{is-dq}	Input setup time for MM_DQ (when reading from DDR SDRAM)	- 0.6		ns	4, 5
T_{ih-dq}	Input hold time for MM_DQ (when reading from DDR SDRAM)	1.5		ns	4, 5

[25-1] Notes:

[25-2] 1. Command signals include MM_CKE_N[1:0], MM_CS[1:0]_N, MM_RAS_N, MM_CAS_N, MM_WE_N, MM_BA[1:0] and MM_A[13:0] signals.

[25-3] 2. Times are measured w.r.t. the positive edge of MM_CLK and the crossing point of MM_CLK and MM_CLK_N.

[25-4] 3. Refer to Figure 2 on page 1-24 for load conditions.

[25-5] 4. Times are measured w.r.t. the corresponding edge of MM_DQS[3:0], i.e. MM_DQS[0] if the DDR device is organized in x32, or respectively MM_DQ[31:24], MM_DQ[23:16], MM_DQ[15:8] and MM_DQ[7:0] (when applicable) if the DDR devices organized in x8 or x16 are used.

[25-6] 5. These timings allow a 250 ps maximum board level skew for MM_CK. MM_CK_N, MM_DQS[3:0] and MM_DQ[31:0] for a 200 MHz operating frequency (i.e. DDR400).

8.3 PCI Bus Interface

Table 26: PCI Bus Timing

Symbol	Parameter	Min	Max	Units	Notes
T_{clock}	Minimum High and Low times	11		ns	1
$T_{val-PCI\ (Bus)}$	Clk to signal valid delay, bus signals	2	11	ns	1,2,3
$T_{val-PCI\ (ptp)}$	Clk to signal valid delay, point-to-point signals	2	12	ns	1,2,3
T_{on-PCI}	Float to active delay	2		ns	1
$T_{Off-PCI}$	Active to float delay		28	ns	1,7
T_{su-PCI}	Input setup time to CLK - bus signals	7		ns	3,4
$T_{su-PCI\ (ptp)}$	Input setup time to CLK - point-to-point signals	12		ns	3,4
T_{h-PCI}	Input hold time from CLK			ns	4
$T_{rst-off-PCI}$	Reset active to output float delay		40	ns	5,6

[26-1] Notes:

[26-2] 1. See the timing measurement conditions in Figure 10.

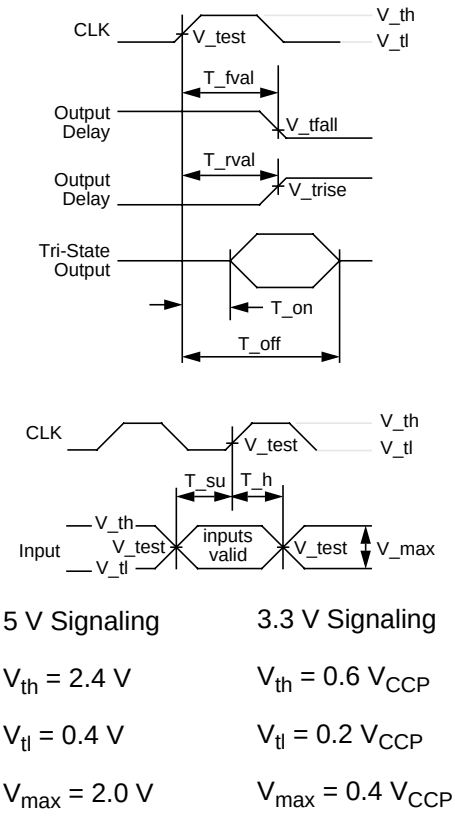
[26-3] 2. Minimum times are measured at the package pin with the load circuit shown in Figure 8. Maximum times are measured with the load circuits shown in Figure 11.

[26-4] 3. **PCI_REQ_N** and **PCI_GNT_N** are point-to-point signals and have different input setup times. All other signals are bused.

[26-5] 4. See the timing measurement conditions in Figure 10.

[26-6] 5. All output drivers are floated when **$\overline{\text{PCI\_RST}}$** (may be connected to RESET_IN_N and/or POR_IN_N) is active.

[26-7] 6. For the purpose of Active/Float timing measurements, the Hi-Z or ‘off’ state is defined to be when the total current delivered through the component pin is less than or equal to the leakage current specification.



5 V Signaling	3.3 V Signaling
V_{th} = 2.4 V	V_{th} = 0.6 V_{CCP}
V_{tl} = 0.4 V	V_{tl} = 0.2 V_{CCP}
V_{max} = 2.0 V	V_{max} = 0.4 V_{CCP}

Figure 10: PCI Output and Input Timing Measurement Conditions

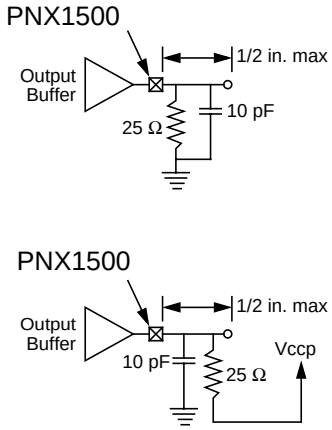


Figure 11: PCI T_{val}(max) Rising and Falling Edge

8.4 QVCP, LCD and FGPO Interfaces

Table 27: QVCP, LCD and FGPO Timing With Internal Clock Generation

Symbol	Parameter	Min	Max	Units	Notes
F_{QVCP}	VDO_CLK1 frequency		65-81	MHz	5
F_{FGPO}	VDO_CLK2 frequency		100	MHz	
T_{CLK-DV}	Clock to VDO_D[33:0] and VDO_AUX	1.5	6	ns	1, 2, 3
T_{SU-CLK}	Input setup time	3		ns	1, 2, 3, 4
T_{H-CLK}	Input hold time	2		ns	1, 2, 3, 4

[27-1] See timing measurement conditions Figure 12.

[27-2] Timing applies when the data is output on a positive or a negative edge in double edge clock mode, see Table 1 on page 3-6.

[27-3] If the VDO_CLK[1,2] is inverted internally then the timing applies to the negative edge.

[27-4] Timing applies for VDO_D[29], FGPO_REC_SYNC and FGPO_BUF_SYNC. VDO_D[29] and FGPO_BUF_SYNC. This inputs are assumed asynchronous.

[27-5] In double edge clock mode, the maximum VDO_CLK1 frequency is 65 MHz. In single edge clock mode, positive or negative edge, the maximum VDO_CLK1 is 81 MHz.

Table 28: QVCP, LCD and FGPO Timing With External Clock Generation

Symbol	Parameter	Min	Max	Units	Notes
F_{QVCP}	VDO_CLK1 frequency		81	MHz	5
F_{FGPO}	VDO_CLK2 frequency		81	MHz	5
T_{CLK-DV}	Clock to VDO_D[33:0] and VDO_AUX	3	11	ns	1, 2, 3
T_{SU-CLK}	Input setup time	4		ns	1, 2, 3, 4
T_{H-CLK}	Input hold time	4		ns	1, 2, 3, 4

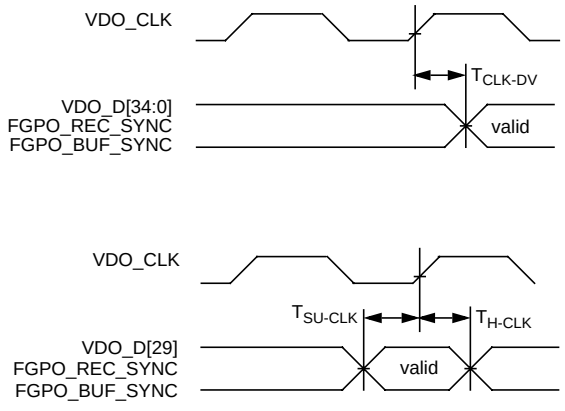


Figure 12: QVCP and FGPO I/O Timing

[28-1] See timing measurement conditions Figure 12.

[28-2] Timing applies when the data is output on a positive or a negative edge in double edge clock mode, see Table 1 on page 3-6.

[28-3] 3. If the VDO_CLK[1,2] is inverted internally then the timing applies to the negative edge.

[28-4] Timing applies for VDO_D[29], FGPO_REC_SYNC and FGPO_BUF_SYNC. VDO_D[29] and FGPO_BUF_SYNC. These inputs are assumed asynchronous.

[28-5] Maximum frequency may get reduced by the wide spread of propagation delay depending on the application needs, i.e. input setup/hold time requirements of the receiving device.

8.5 VIP and FGPI Interfaces

Table 29: VIP and FGPI Timing

Symbol	Parameter	Min	Max	Units	Notes
F_{VIP}	VDI_CLK1 frequency		81	MHz	
F_{FGPI}	VDI_CLK2 frequency		100	MHz	
T_{SU-CLK}	Input setup time	3		ns	1, 2, 3
T_{H-CLK}	Input hold time	2		ns	1, 2, 3

[29-1] Notes:

[29-2] 1. Timing applies whether the clock is external or internal.

[29-3] 2. Timing applies whether the data is output on a positive or a negative edge.

[29-4] 3. See timing measurement conditions Figure 13.

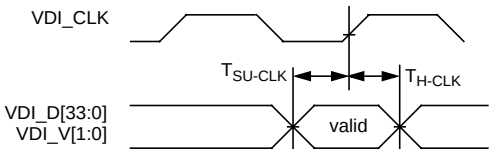


Figure 13: VIP and FGPI I/O Timing

8.6 10/100 LAN In MII Mode

Table 30: 10/100 LAN MII Timing

Symbol	Parameter	Min	Max	Units	Notes
F_{LAN_CLK}	LAN_CLK frequency		60	MHz	
F_{CLK}	LAN_TX_CLK and LAN_RX_CLK frequency		25	MHz	
T_{CLK-DV}	Clock to LAN Outputs	6	17	ns	1, 2, 3
T_{SU-CLK}	Input setup time	5		ns	1, 2, 3
T_{H-CLK}	Input hold time	3		ns	1, 2, 3

[30-1] Notes:

[30-2] 1. Timing applies whether the clock is external or internal.

[30-3] 2. Timing applies whether the data is output on a positive or a negative edge.

[30-4] 3. See timing measurement conditions Figure 14.

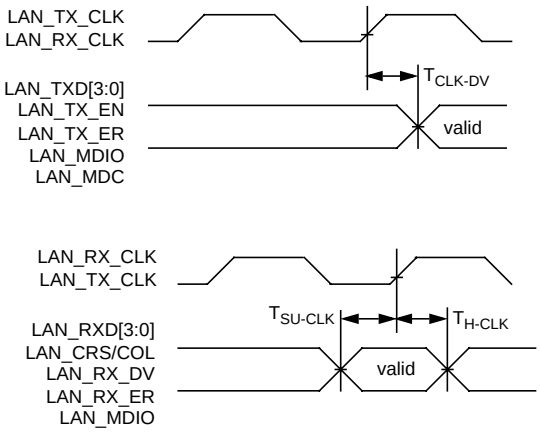


Figure 14: LAN 10/100 I/O Timing in MII Mode

8.7 10/100 LAN In RMII Mode

Table 31: 10/100 LAN RMII Timing

Symbol	Parameter	Min	Max	Units	Notes
F_{LAN_CLK}	LAN_CLK frequency		60	MHz	
F_{CLK}	LAN_TX_CLK and LAN_RX_CLK frequency		50	MHz	
T_{CLK-DV}	Clock to LAN Outputs	5	13	ns	1, 2, 3
T_{SU-CLK}	Input setup time	5		ns	1, 2, 3
T_{H-CLK}	Input hold time	2		ns	1, 2, 3

[31-1] Notes:

[31-2] 1. Timing applies whether the clock is external or internal.

[31-3] 2. Timing applies whether the data is output on a positive or a negative edge.

[31-4] 3. See timing measurement conditions Figure 14.

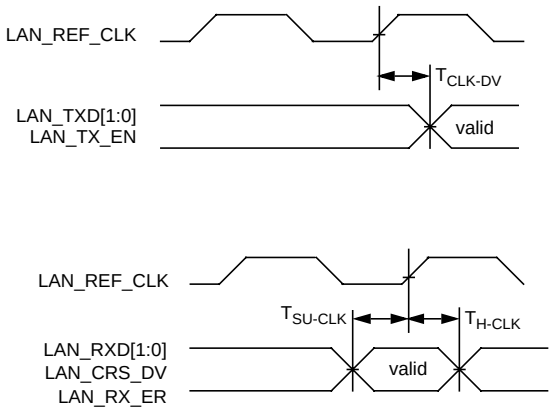


Figure 15: LAN 10/100 I/O Timing in RMII Mode

8.8 Audio Input Interface

Table 32: Audio Input Timing

Symbol	Parameter	Min	Max	Units	Notes
F_{OSCLK}	Audio Input oversampling frequency		50	MHz	
F_{AI_CLK}	Audio Input frequency		25	MHz	
T_{CLK-DV}	Clock to AI_WS	4	10	ns	3
T_{SU-CLK}	Input setup time	4		ns	1, 2, 3
T_{H-CLK}	Input hold time	0		ns	1, 2, 3

[32-1] Notes:

[32-2] 1. Timing applies whether the clock is external or internal.

[32-3] 2. Timing applies whether the data is output on a positive or a negative edge.

3. See timing measurement conditions Figure 16.

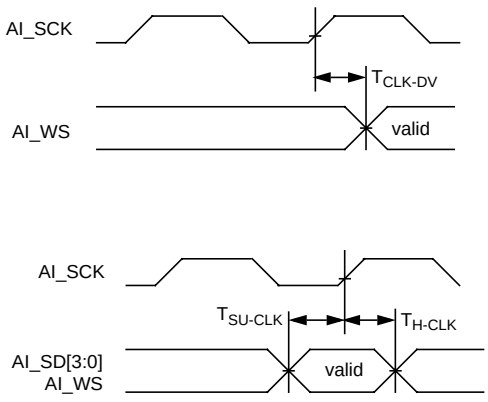


Figure 16: Audio Input I/O Timing

8.9 Audio Output Interface

Table 33: Audio Output Timing

Symbol	Parameter	Min	Max	Units	Notes
F_{OSCLK}	Audio Output oversampling frequency		50	MHz	
F_{AO_CLK}	Audio Output frequency		25	MHz	
T_{CLK-DV}	Clock to AO_WS and AO_SD[3:0]	4	10	ns	3
T_{SU-CLK}	Input setup time	4		ns	1, 2, 3
T_{H-CLK}	Input hold time	0		ns	1, 2, 3

[33-1] Notes:

[33-2] 1. Timing applies whether the clock is external or internal.

[33-3] 2. Timing applies whether the data is output on a positive or a negative edge.

3. See timing measurement conditions Figure 17.

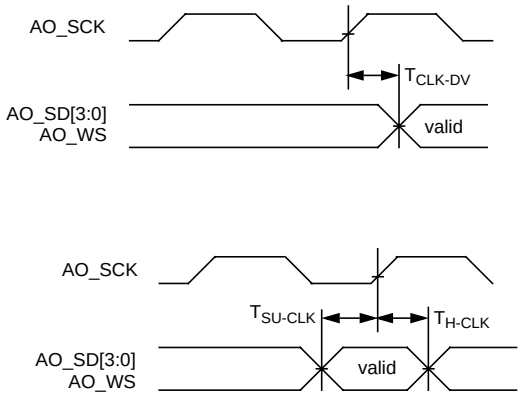


Figure 17: Audio Output I/O Timing

8.10 SPDIF I/O Interface

Table 34: SPDIF I/O Timing

Symbol	Parameter	Min	Max	Units	Notes
T_{HIGH}	Data/Clock Output High Time	12.5		ns	Figure 18
T_{LOW}	Data/Clock Output Low Time	12.5		ns	Figure 18
T_{IHIGH}	Data/Clock Input High Time	8.5		µs	Figure 18
T_{ILOW}	Data/Clock Input Low Time	8.5		µs	Figure 18

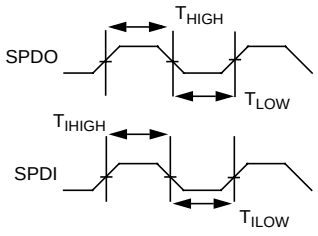


Figure 18: SPDIF I/O Timing

8.11 I²C I/O Interface

Table 35: I²C I/O Timing

Symbol	Parameter	Min	Max	Units	Notes
f_{SCL}	SCL clock frequency		400	kHz	Figure 19
T_{BUF}	Bus free time	1		µs	Figure 19
T_{SU-STA}	Start condition set up time	1		µs	Figure 20
T_{H-STA}	Start condition hold time	1		µs	Figure 20
T_{LOW}	IIC_SCL LOW time	1		µs	Figure 19
T_{HIGH}	IIC_SCL HIGH time	1		µs	Figure 19
T_F	IIC_SCL and IIC_SDA fall time (Cb = 10-400 pF, from V_{IH-IIC} to V_{IL-IIC})	20+0.1Cb	250	ns	Figure 19
T_{SU-SDA}	Data setup time	100		ns	Figure 20
T_{H-SDA}	Data hold time	0		ns	Figure 20
T_{DV-SDA}	IIC_SCL LOW to data out valid		0.5	µs	Figure 20
T_{DV-STO}	IIC_SCL HIGH to data out	1		ns	Figure 20

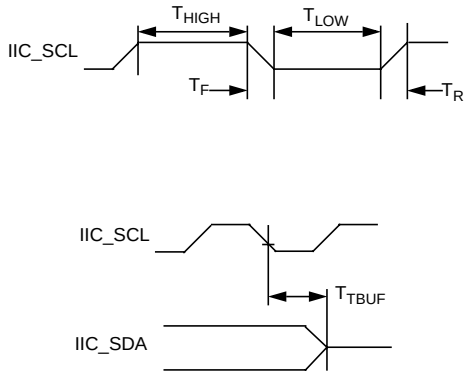


Figure 19: I²C I/O Timing

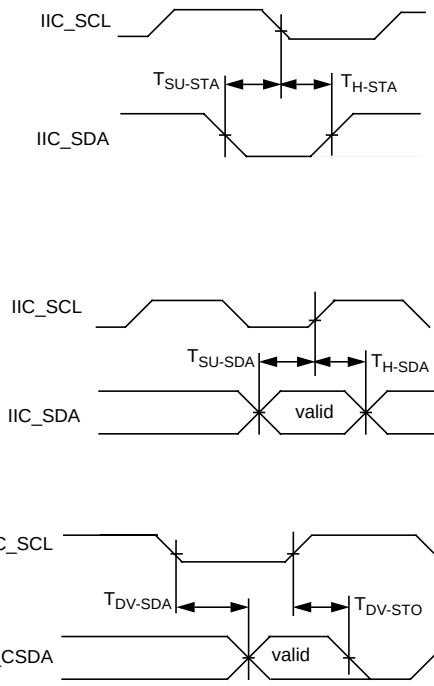


Figure 20: I^2C I/O Timing

8.12 GPIO Interface

Table 36: GPIO Timing

Symbol	Parameter	Min	Max	Units	Notes
F_{CLOCK}	GPIO sampling/pattern generation CLOCK frequency		108	MHz	1
$T_{CLK-DV1}$	GPIO[6:0] CLOCK to DATA valid for GPIO[15:0] pins	3.5	15	ns	1
$T_{CLK-DV2}$	GPIO[6:0] CLOCK to DATA valid for GPIO[60:16] pins	3	17.5	ns	1
T_{SU-CLK}	Input setup time	6.5		ns	2
T_{H-CLK}	Input hold time	1.5		ns	2
T_{VALID}	Valid time required for sampling mode, i.e. $F_{CLOCK}/8$	75		ns	

[36-1] Notes:

[36-2] 1. The GPIO module can operate up to 108 MHz, however the maximum operating frequency may be limited due to the wide variation of $T_{CLK-DV[1,2]}$. For example if a 4 ns valid window is required for data out then the maximum recommended operating frequency is 50 MHz for $T_{CLK-DV1}$ and 65 MHz for $T_{CLK-DV2}$.

[36-3] 2. Timing applies whether the data is output on a positive or negative edge. GPIO[6:0] can be selected as clocks. Data can be any of the GPIO[60:0] as defined in Section 2.3 on page 1-3.

3. See timing measurement conditions Figure 21.

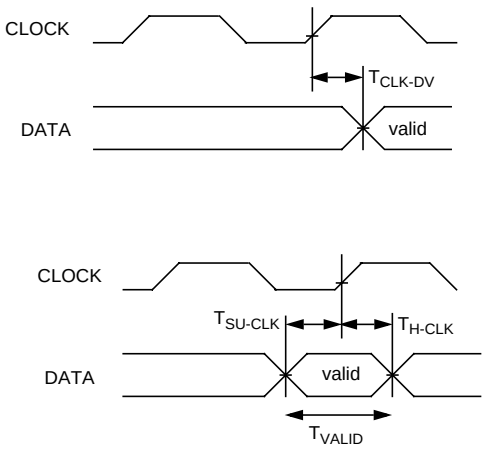


Figure 21: Audio Output I/O Timing

8.13 JTAG Interface

Table 37: JTAG Timing

Symbol	Parameter	Min	Max	Units	Notes
F_{BSCAN}	Boundary scan frequency		15	MHz	
F_{JTAG}	JTAG frequency		20	MHz	
T_{CLK-DV}	Falling edge of the JTAG_TCK to JTAG_TDO	0	8	ns	Figure 22
T_{SU-CLK}	Input setup time	8		ns	Figure 22
T_{H-CLK}	Input hold time	3		ns	Figure 22

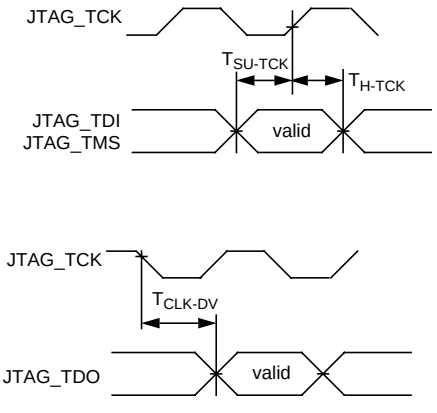
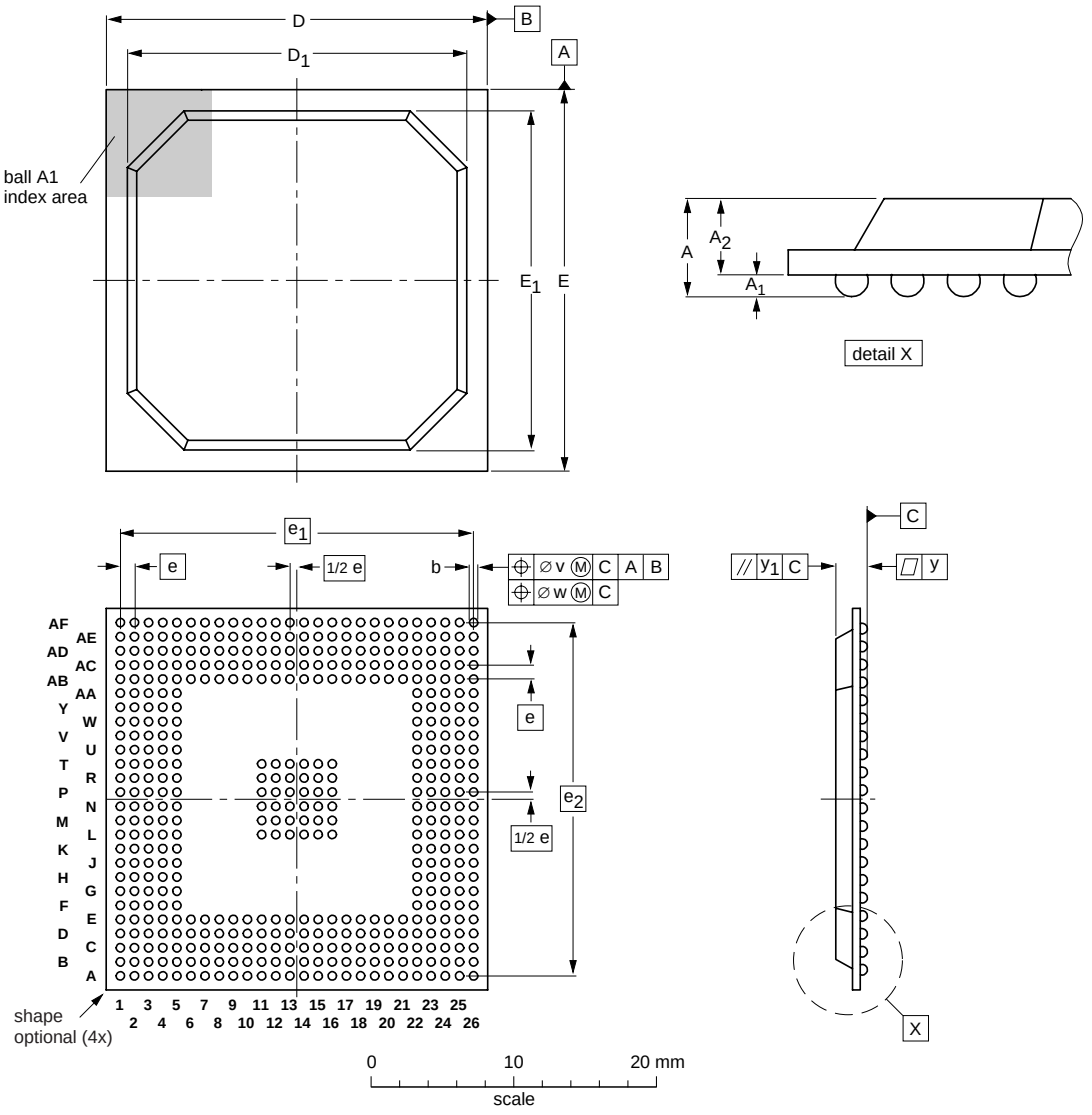


Figure 22: JTAG I/O Timing

9. Package Outline

Latest information may be found at http://www.semiconductors.philips.com/package/SOT795-1.html#footprint

BGA456: plastic ball grid array package; 456 balls; body 27 x 27 x 1.75 mm **SOT795-1**



DIMENSIONS (mm are the original dimensions)

UNIT	A max.	A_1	A_2	b	D	D_1	E	E_1	e	e_1	e_2	v	w	y	y_1
mm	2.45	0.6 0.4	1.85 1.60	0.7 0.5	27.2 26.8	24.75 23.75	27.2 26.8	24.75 23.75	1	25	25	0.3	0.15	0.2	0.35

OUTLINE VERSION	REFERENCES				EUROPEAN PROJECTION	ISSUE DATE
	IEC	JEDEC	JEITA			
SOT795-1	144E	MS-034	- - -			02-11-18

Figure 23: BGA456 Plastic Ball grid Array; 456 Balls; body 27 x 27 x 1.75 mm

10. Board Design Guidelines

The following sections discuss the fundamentals of board design for the PNX1500 system. The intent is to give general guidelines on the subject, not the complete in depth coverage.

A minimum of four layers board is recommended.

10.1 Power Supplies Decoupling

Power supply regulators require large smoothing capacitors to deliver the current until the regulator can follow the load conditions. These smoothing capacitors are typically large electrolytic capacitors with considerable parasitic inductance, typically in the order of 10 nH. This high inductance does not allow for rapid supply of varying currents required in high speed processors as the PNX1500. The following recommendations assume a load transient of up to 1 A within 2 ns which is considered conservative for the PNX1500. However, this does guarantee adequate decoupling.

In “high frequency” applications, each power plane VCCP, VCCM and VDD should be decoupled with at least 10 capacitor of 0.1 μF. Capacitors should be chosen such that the total series inductance is approximately within the order of 0.2 nH (i.e. 2 nH per capacitor). The parasitic series resistance per capacitor should be in the order of 0.1 Ω. Ceramic capacitors may be used. These surface mount capacitors should be placed as closely as possible to the power pins of the PNX1500.

For “medium frequency”, each power plane VCCP, VCCM and VDD should be decoupled with at least 10 capacitors of 47 μF. Capacitors should be chosen such that the total series inductance is approximately with the order of 0.5 nH. The parasitic series resistance per capacitor should be in the order of 0.1 Ω. Aluminum or wet “wound foil” tantalum capacitors should not be used. Instead, dry tantalum capacitors or equivalent total series resistor and inductance capacitors like the new ceramic or polymer tantalum can be used. Despite the larger footprint these surface mount “medium frequency” decoupling capacitors should still be placed as closely as physically possible to the PNX1500 power pins.

Last step before the power regulator itself is the bulk decoupling. The bulk decoupling can be achieved with five 100 μF or 220 μF capacitors. These capacitors usually have an inductance of 10 nH and internal equivalent series resistance (ESR) of 0.1 Ω. The amount and size are dependant on how fast the regulator operates.

The VIA connection to the power planes should be as wide as the capacitor soldering lead which is different from a VIA of a regular signal. The routing and VIA inductance and resistance must be included when computing the total series inductance and resistance.

Other devices like the memories also require local decoupling capacitors. Three 0.1 μF capacitors combined with one 22 μF or 47 μF are recommended for each memory device.

Additional global decoupling can also be distributed across the board.

10.2 Analog Supplies

10.2.1 The 3.3 V Analog Supply

The entire analog ground/supply is kept free-floating on the PCB.

Quiet VCCA for the PLL subsystem should be supplied from VCCP through a 18 Ω series resistor. It should be bypassed for AC to VSSA, using a dual capacitor bypass (hi and low frequency AC bypass).

Quiet VSSA for the PLL subsystem: the bypass should only be connected to the PNX1500 VSSA[] pins and not to the global VSS (i.e. ground) network. No external coil or other connection to board ground is needed, such connection would create a ground loop.

Figure 24 illustrates the analog filtering for the 3.3 V Analog Supply. One 47µF and

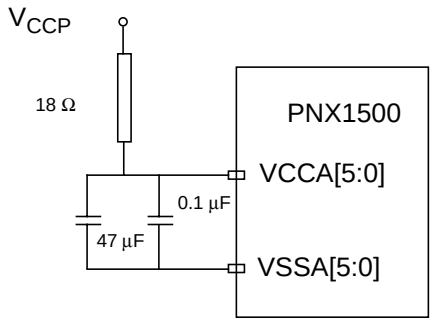


Figure 24: Digital VCCP Power Supply to Analog VCCA/VSSA Power Supply Filter

one 10 µF is sufficient for the six VCCA[] pins.

10.2.2 The 1.2-1.3-V Analog Supply

The entire analog ground/supply is kept free-floating on the PCB.

All the key components (the analog bypass capacitor and crystal capacitors) are on the PCB connected to the free-floating analog VSSA_1.2 net (Figure 25, Figure 26).

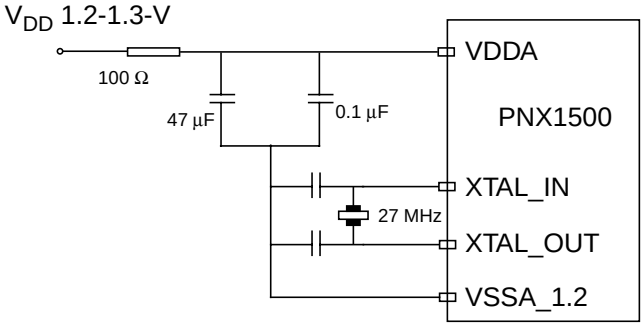


Figure 25: Digital VDD Power Supply to Analog VDDA/VSSA_1.2 Power Supply Filter

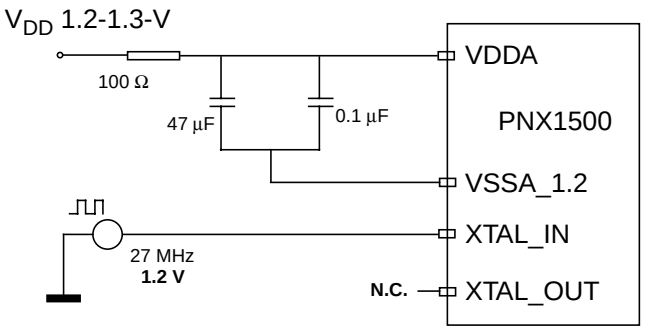


Figure 26: Digital VDD Power Supply to Analog VDDA/VSSA_1.2 Power Supply Filter

10.3 DDR SDRAM interface

Designing a proper DDR SDRAM interface with the PNX1500 system that guarantees correct signal integrity and timing margins (even at 200 MHz, i.e. DDR400) can be achieved by implementing the following board level design rules:

- 50 Ω trace impedance. The width of the PCB trace as well as the dielectric layer must be adjusted to meet the 50 Ω impedance traces. The PNX1500 SSTL_2 drivers must be fine tuned to limit undershoot and/or overshoot over traces with 50 Ω impedance. This should guarantee high quality signal integrity.
- 'T' shape connection when a signal must be connected to two (or more) memory devices. The bar of the 'T' should have impedance higher than 50 Ω in order to compensate for the trace split. 70 Ω is recommended but not required if the bar of the 'T' is less than half of the 'leg' of the 'T'.
- Recommended Trace lengths for operating frequency of up to DDR400 are shown in Table 38.

Table 38: DDR Recommended Trance Length

Signal	Maximum (cm)	Minimum (cm)
MM_CK, MM_CK#	5	5
MM_AD[12:0], MM_BA[1:0] MM_RAS/CAS/WE/CKE MM_CS[1:0]	7	2
MM_DQS[3:0]	3	3
MM_DATA[31:0] MM_DQM[3:0]	3	1

DDR devices that are DDR400{A,B,C} JEDEC compliant, revision JESD79C, have tDQSS defined as 0.72*tCK (min) and 1.25*tCK (max). Faster DDR devices have a more stringent requirement of 0.8*tCK and 1.2*tCK or even 0.85*tCK and 1.15*tCK. The PNX1500 can support these fast DDR devices as long as Table 38 is strictly followed. In case of using DDR400 only DDR devices, MM_CK/MM_CK# may have a minimum value of 4 cm, the remaining signals should still follow as close as possible the Table 38.

The ball assignment implies that the two outside rows of balls are routed on a different board layer than the next two rows of balls. This is recommended to reduce the skew. The DQS lines are the exception since they are located on the outside row for better package signal integrity.

A 10-22 Ω series resistor is recommended on the two clock lines. They need to be placed as close as possible to the PNX1500 clock output pins. In addition a 100 Ω shunting both memory clocks, i.e. MM_CLK and MM_CLK#, will reduce the swing of the signals and improve signal integrity.

No other termination is required at board level to achieve maximum speed if these rules are strictly followed.

Above DDR333, i.e. MM_CLK of 166 MHz, the 183 or 200 MHz operating speeds (i.e. DDR400) are only available for a maximum of 2 loads.

10.3.1 Do DDR Devices Require Termination?

Most DDR devices are meant to drive very long and highly loaded track lines. Their drivers are usually very strong and could use a 22 Ω series resistors on the data/dqm and dqs lines on the DDR device's end.

10.3.2 What if I really want to use termination for the PNX1500?

It is possible to parallel terminate each line to a termination voltage with a 50 Ω resistor for the 200 MHz operation (i.e. DDR400). The resistor should be placed as close as possible to the intersection of the leg of 'T' and the bar of the 'T' (this applies when the signal has two or more loads). For single loaded tracks and bi-directional signals, the parallel termination resistor should be placed about 50% of the way to the DDR SDRAM device. For unidirectional signals and single loaded tracks, the termination should be placed after the pin of DDR SDRAM device. In this case, the VTT supply must be carefully designed with very wide tracks since the current through that power supply is very high due to the termination and its active current consumption over 80+ pins.

MM_CKE must not be parallel terminated since it requires a 0V level at initialization time.

Similarly for signal integrity purpose, it is possible to only series terminate the address, the command lines, and the data lines (at the PNX1500 side). There is no need for series termination if the parallel termination was chosen.

10.4 Package Handling, Soldering and Thermal Properties

Up to date package information can be found at

http://www.philipslogic.com/packaging/handbook

11. Miscellaneous

In order to limit clock jitter on the TM3260 and DDR clocks, it is recommended to shutdown the clocks of the unused modules, typically by programming these modules to enter the powerdown mode and switch the others to their functional clocks (i.e. switch the module's clocks to a frequency higher than the default 27 MHz crystal clock when possible).

12. Soft Errors Due to Radiation

Soft errors can be caused by radiation, electromagnetic interference, or electrical noise. This section reports the soft error rate (SER) caused by the radiation component.

There are three primary radiation sources namely alpha particles, high-energy cosmic rays, and neutron-induced boron fission. Alpha particles originate from radioactive impurities in chip and package materials. Cosmic rays indirectly generate charges by colliding with nuclei within the chip. The boron fission occurs when a low-energy (thermal) neutron hits a 10B nucleus, which then breaks up into an alpha and lithium recoil. The SER generated by these radiation sources is of 9900 Failure-In-Time (FIT) which is equivalent to one failure every 10 years.

In the PNX1500, the SER is statistically improved since some of the memory elements (that are affected by the radiation) may contain pixel data rather than control data which further extends the SER.

13. Ordering Information

Table 39: Ordering Information

Part Name	12 NC	Speed	Core Voltage	Package	Version	Leadfree	End of Life
PNX1500E	12NC 9352 729 05557	240 MHz	1.2-V	BGA456	SOT795	NO	30 June 2005
PNX1501E	12NC 9352 747 28557	266 MHz	1.2-V	BGA456	SOT795	NO	30 June 2005
PNX1502E	12NC 9352 747 44557	300 MHz	1.3-V	BGA456	SOT795	NO	30 June 2005
PNX1500E/G	12NC 9352 777 46557	240 MHz	1.2-V	BGA456	SOT795	YES	
PNX1501E/G	12NC 9352 777 47557	266 MHz	1.2-V	BGA456	SOT795	YES	
PNX1502E/G	12NC 9352 777 48557	300 MHz	1.3-V	BGA456	SOT795	YES	

Chapter 2: Overview

PNX15xx Series Data Book – Volume 1 of 1

Rev. 2 — 1 December 2004 Product data sheet

1. Introduction

The PNX15xx Series Media Processor is a complete Audio/Video/Graphics system on a chip that contains a high-performance 32-bit VLIW processor, TriMedia™ TM3260, capable of software video and audio signal processing, as well as general purpose control processing. It is capable of running a pSOS operating system with real-time signal processing tasks in a single programming and task scheduling environment. An abundance of interfaces make the PNX15xx Series suitable for networked audio/visual products. The processor is assisted by several image and video processing accelerators that support image scaling and compositing. Figure 1 pictures a high level functional block diagram.

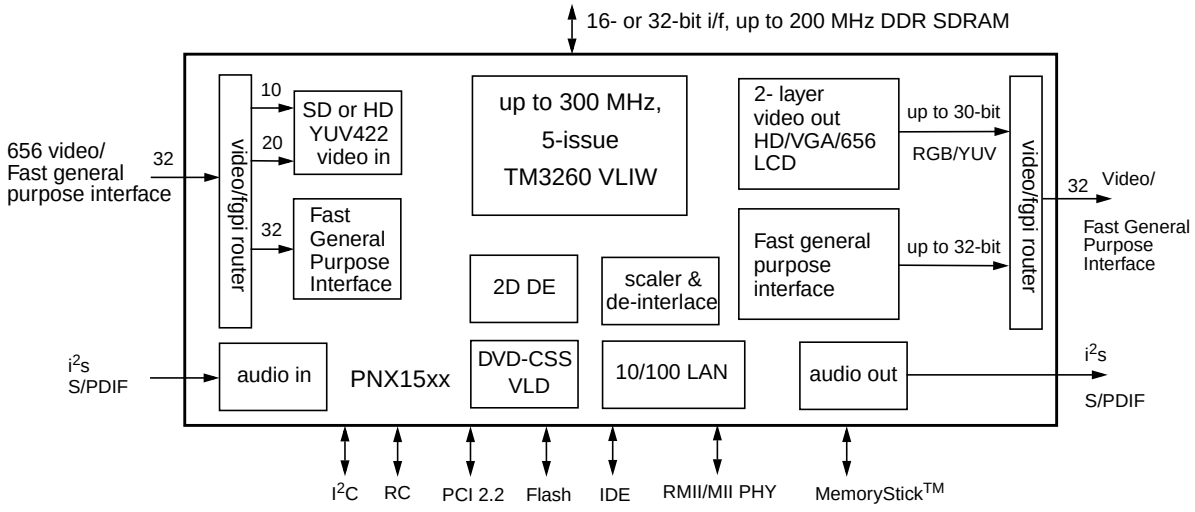


Figure 1: Block Diagram PNX15xx Series

1.1 PNX15xx Series Functional Overview

The functionality achieved within the PNX15xx Series can be divided into three major categories: *decode*, *processing,* and *display.*

Decode functions take input data streams and convert those streams into memory based structures that the PNX15xx Series may further process. Decode functions may be simple, as in the case of storing 656 input video into memory, or substantially more complex, as in the case of MPEG-2.

Processing functions are those that modify an existing data structure and prepare that structure for display functions.

Display functions take the processed data structures from memory and generate the appropriate output stream. As in the case of the decode functions, display functions can be relatively simple, such as an I^2S audio output, or very complex, as in the case of multi-surface composited display.

All decoded data structures are stored in memory, even when further processing is not required. This mechanism implies that there is no direct path between input and output data streams. The memory serves as the buffer to de-couple input and output data streams. Based on the mode of operation, there may be multiple data structures in memory for a given input stream. The PNX15xx Series uses the TM3260 CPU and a timestamping mechanism to determine when a specific memory data structure is to be displayed.

The PNX15xx Series implements the required decode, processing, and display functions with a combination of fixed function hardware and TM3260 CPU software modules. The PNX15xx Series provides a good balance between those functions that are implemented in fixed hardware and those that are programmed to run on the TM3260 CPU. The following tables illustrate how the major tasks are implemented under each of the three main functional areas, and how they map to hardware resources or software.

Table 1: Partitioning of Functions to Resources

function	Resource	description
video decoding/acquisition		
digital video acquisition	VIP	includes optional horizontal down scaling or color space conversion, and conversion to a variety of memory pixel formats
MPEG-1/2/4 video decoding	software	
DVD authentication & de-scrambling	DVD-CSS	authentication & de-scrambling in hardware
audio decoding and improvement processing		
audio decoding AC3, AAC, MPEG L1, L2, MP3, others	software	decoders for almost any audio format available
audio processing	software	improvement processing and mixing
graphics		
2D graphics rendering and DMA	2D DE	
non-motion compensated de-interlacing	MBS	median, 2-field majority select, 3-field majority select with or without EDDI post pass for edge improvement
motion compensated de-interlacing	MBS + software	software provides the MBS with a motion compensated field, to which the MBS applies the chosen de-interlacing algorithm
motion estimation	software	pixel accurate and quarter pixel accurate versions available
temporal up conversion	software	creates images temporally between two originals using motion vectors
luminance histogram measurement	MBS	luminance histogram collection during a de-interlace or scaling pass

Table 1: Partitioning of Functions to Resources

function	Resource	description
image scaling	VIP, MBS, QVCP	VIP can perform horizontal down-scaling during acquisition MBS can perform up-and down scaling horizontal and vertical in a single pass, optionally combined with de-interlacing and format conversions QVCP can perform panoramic horizontal scaling during output
video format conversions, including color space conversion	VIP, MBS, QVCP	MBS can convert any pixel format to any other format VIP can generate multiple video formats, QVCP can read multiple video formats
histogram correction, black stretch, luminance sharpening (LTI, CDS, HDP), color features (green enhancement, skin tone correction, blue stretch, dynamic color transient improvement)	QVCP	performed during output to display
display processing		
surface composition with alpha blending, chroma (range) keying	QVCP	
video and graphics scaling	QVCP	hi-quality panoramic horizontal scaler for video, linear interpolator for graphics
gamma correction contrast, brightness, saturation control	QVCP	
discretionary processing		
MPEG-4 video encoding	software	
MPEG-4 Simple or Advanced Simple Profile decoding	software	
MPEG-2 video encoding	software	1/2 D1 and other versions available
transrating/transcoding	software, VLD	the VLD hardware can be used to parse a MPEG-2 video stream. Software composes a new MPEG-2 stream including the video stream with reduced bitrate.
video conferencing		a large variety of applications is available

1.2 PNX15xx Series Features Summary

- 32-bit, up to 300 MHz 5-issue VLIW CPU with 128 32-bit registers and an extensive set of video and audio media instructions.
- Allows V^2F power management to control frequency and power consumption based on application requirement.
- High quality hardware image scaler and advanced de-interlacer, augmented with media processing software to do motion compensated de-interlacing.
- 2D Drawing Engine capable of 3 operand BitBlt (all 256 raster operations), line drawing, and host font expansion.
- 10-bit YUV Video capture supporting horizontal downscaling scaling up to 40.5 Mpix/s.

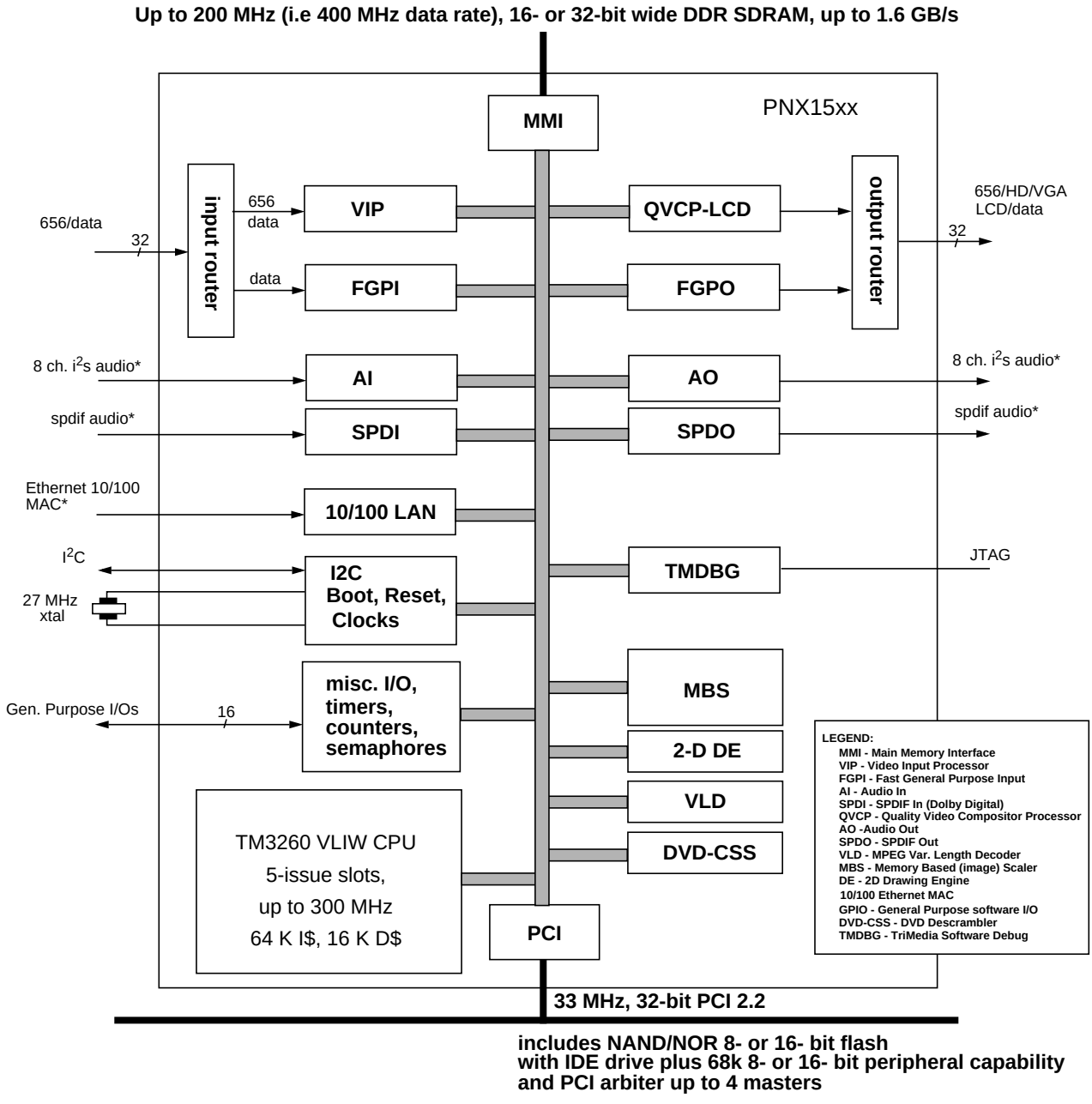
12NC 9397 750 14321

- 2-layer compositing video output, with integrated scaling and video improvement processing, supporting W-XGA TFTs, 1280 x 768 60 Hz, HD video, up to 1920 x 1080 60 I, or up to 81 Mpix/s.
- Data Streaming and Message Passing ports with up to 400 MB/s bandwidth capability.
- Variable Length Decoder assist engine.
- Integrated DVD descrambler for DVD playback functionality.
- Octal digital audio in plus S/PDIF (Dolby Digital^{TM}) input.
- Octal channel digital audio output plus S/PDIF (Dolby Digital^{TM}) output.
- Integrated controller for unified DDR SDRAM memory system of 16 - 256 MB using 32-bit wide data at up to 400 MHz data rate, i.e. up to 1.6 GB/s. Configurable to a 16-bit wide DDR SDRAM interface.
- 32-bit, 33 MHz PCI 2.2 interface with integrated PCI bus arbiter up to 4 masters.
- Glueless NOR and NAND 8- or 16-bit Flash interface.
- 4 timers/counters, capable of counting internal and external events.
- 16 dedicated General Purpose I/O pins, suitable as software I/O pins, external interrupt pins, universal Remote Control Blaster, clock source/gate for system event timers/counters and emulating high-speed serial protocols.
- Additional multiplexed General Purpose I/O pins.
- On-chip MPEG-1 and MPEG-2 VLD to facilitate transrating, transcoding and software SD MPEG decoding.
- Integrated low-speed, DVD drive capable, IDE controller (shares PCI pins, requires 2 external buffers to isolate, up to ATAPI/PIO-4).
- All video/audio timing derived from a single low-cost external crystal (no VCXO's required).
- 10/100 RMII and MII IEEE 802.3 PHY interface.

12NC 9397 750 14321

2. PNX15xx Series Functional Block Diagram

Figure 1 gives a quick overview of the inside of the PNX15xx Series system. Each component is further explained in this chapter and later more detailed with a dedicated chapter.



NOTE: I/Os marked with * can also function as General Purpose serial I/O pins instead of in primary function mode

Figure 2: PNX15xx Series Functional Block Diagram

3. System Resources

3.1 System Reset

The PNX15xx Series includes a system reset module. This reset module provides a synchronous reset to internal PNX15xx Series logic and a reset output pin for initialization of external system components. A system reset can be initiated in response to a board level reset input pin, a software configuration write or as a result of a programmable watchdog timer time-out. This watchdog timer is a fail-safe recovery mechanism which may be enabled by software. When enabled, a periodic interrupt is sent to the TM3260 CPU. If the CPU does not respond to the interrupt within a programmable time-out period, then the system is assumed to be hung and the system reset is asserted.

Boot also resets board level peripherals by asserting the SYS_RST_OUT_N pin.

3.2 System Booting

The PNX15xx Series boot method is controlled by the BOOT_MODE[7:0] pins' resistive straps. The Table 2 shows the main boot modes available. More details can be found in Chapter 6 Boot Module. At the time of the RESET_IN input deassertion, the code on these pins is sampled. The pins operate as GPIO pins after boot.

Table 2: PNX15xx Series Boot Options

BOOT_MODE	Description
000	Set up system, and start the TM3260 CPU from a 8-bit NOR Flash or ROM attached to PCI/XIO
100	Set up system, and start the TM3260 CPU from a 16-bit NOR Flash or ROM attached to PCI/XIO
001	Set up system, and start the TM3260 CPU from a 8-bit NAND Flash attached to PCI/XIO
101	Set up system, and start the TM3260 CPU from a 16-bit NAND Flash attached to PCI/XIO
x10	Boots in host assisted mode with a default SubSystem ID of 0x1234 and a default System Vendor ID of 0x5678. This boot mode can be used for standalone system but should not be used for a PC PCI plug-in card since such a board requires a personal System Vendor and SubSystem ID. Instead the I^2C boot EEPROM should be used.
s11	Boots from a I^2C EEPROM attached to the I^2C bus. EEPROMs of 2 KB - 64 KB size are supported. The entire system can be initialized in a custom fashion by the boot command structure. The I^2C EEPROM holds write commands and writes data to internal MMIO registers and to the main memory. BOOT_MODE[2] defines the speed of the I^2C bus, i.e. 100 or 400 KHz.
other	Reserved

The PNX15xx Series on-chip TM3260 CPU is capable of direct standard Flash execution to allow for booting. Note: Direct execution from NAND Flash, a.k.a. disk Flash is not supported. Direct execution from flash, however, has very limited performance. Hence, the TM3260 typically copies a Flash file to high-performance system DRAM, and executes it in DRAM. That Flash file contains the self-decompressing initial system software application. This multi-stage boot process that starts a compressed code module minimizes system memory cost.

The scripted boot, in combination with an appropriately programmed I^2C EEPROM, allows the PNX15xx Series to boot in many ways.

A stand-alone PNX15xx Series system is able to reliably update its own Flash boot image, whether the Flash is standard or nand Flash. In most systems this is done by extra Flash storage capacity that is used by the Flash update software to guarantee atomicity of a boot image update under power failure. The update either succeeds or the old boot image is retained. In some systems, however, it may be cost attractive to use a medium size boot I^2C EEPROM instead. This boot EEPROM would hold the code to recover a corrupted Flash from some system resource such as a network or disk drive.

In the presence of an external host processor boot is very different. PNX15xx Series must execute an I^2C EEPROM boot script that loads a small amount of board level personality data. Once this data is obtained, PNX15xx Series is ready to follow the standardized PCI enumeration and configuration protocol executed by the host. In external host configurations a single small I^2C EEPROM is required, and no Flash memory is needed. The host is responsible for configuring a list of PNX15xx Series internal registers, loading an application software image into PNX15xx Series DRAM and starting the TM3260.

3.3 Clock System

PNX15xx Series provides a low cost, highly programmable clock system. All the clocks used within PNX15xx Series system can be generated internally with a mixed combination of PLLs, Direct Digital Synthesizers (DDSs) or simple clock dividers depending on the clock module requirements. All the clocks are derived from a low cost 27 MHz crystal clock. This input clock is multiplied internally by 64 to generate a 1.728 GHz clock from which each PNX15xx Series module receives a derived clock. This internal high speed clock allows minimal jitter on the generated clocks.

3.4 Power Management

The PNX15xx Series system, with its programmable clocks, can be set to operate in 3 different power modes.

- Normal mode in which each module runs at the required speed and the CPU runs at its maximum speed.
- Saving mode in which each module runs at required speed and the CPU runs at the speed that the application needs. For example MP3 audio decoding will require less than 30 MHz, while a simple profile MPEG-4 video decoding will require less than 100 MHz.
- Sleep mode in which all the clocks of the system are turned off. A small amount of logic stays alive in order to wake up the system. Before going into sleep mode, the CPU can decide that some generated clocks, like the PCI clock may remain active. In that case the clocks are gated for each module belonging to PNX15xx Series. Also the PCI outgoing clock may be reduced to XTAL_IN (27 MHz recommended) divided by 16. The system will not respond to incoming PCI transactions or generate outgoing PCI transactions, but other PCI components may remain operational.The system can wake up upon one of these three events:

- an external wake-up event on pin GPIO[15]. When entering in sleep mode, the GPIO[15] pin state (i.e. value of the pin) is sampled and registered. The CPU is woken up if the pin GPIO[15] changes state (from low to high) after the system has gone into sleep mode. The GPIO[15] pin is observable by software.
- an expired internal counter. Before entering in sleep mode, this special counter is set up to count XTAL_IN clock ticks. Once the count is satisfied, the CPU is woken up. The counter has 32 bits.
- an incoming event is detected by the GPIO module (could be a Remote Control ‘power on’ command). Before going into sleep mode, the CPU sets the GPIO event queues to monitor a selected group of GPIO pins. Once the queues are full or have monitored an event, the CPU is woken up (via an interrupt). This is a more sophisticated wake-up event than the wake-up upon transition on GPIO[15] pin event, since several events are sampled and therefore keep the GPIO alive.

After wake-up from sleep mode, the TM3260 CPU can examine the tentative wake-up attempt, and if the wake-up is genuine, bring the system back to full operational mode.

In addition, the clocks to individual unused modules can be turned off altogether and the idle() task of the operating system can be used to activate a voluntary powerdown mechanism in the CPU. These modes are not managed by a hardware power mode controller, but by software using the standard provisions of the CPU and the clock system.

3.5 Semaphores

The semaphore module implements 16 semaphores for mutual-exclusion in a multi-processor environment. Each processor in the system (at board level) can request a particular semaphore. All 16 semaphores are accessed through the same bus which guarantees atomic accesses.

There is no built-in mapping of semaphores to sharable hardware system resources. Such mapping is done by software convention.

Each semaphore behaves as follows:

```
if (current_content == 0)    new_content = write_value;
else if (write_value == 0)   new_content = 0;
```

Only the lower 12 bits of the semaphore are writable. These lower 12 bits are used by software to write a unique ID decided by software convention. The upper 20 bits always return 0 when read.

3.6 I^2C Interface

The I^2C interface on the PNX15xx Series provides I^2C master and slave capability. The I^2C interface supports two operating modes, the standard mode, which runs at 100 KHz, and the fast mode, which runs at 400 kHz.

The I^2C interface may be used to connect an optional boot EEPROM and/or other peripherals like video/audio ADC/DACs at board level.

4. System Memory

4.1 MMI - Main Memory Interface

PNX15xx Series has an unified memory system for the PNX15xx Series CPU and all of its modules. This memory is also visible from any PCI master as PCI attached memory.

The 32-bit DDR SDRAM interface can operate up to 200 MHz. This is equivalent to a 64-bit SDR SDRAM interface running at 200 MHz, resulting in theoretical available bandwidth of up to 1.6 GB/s.

This interface can support memory footprints from 8 up to 256 MB. The supported memory configurations are displayed in Table 3.

Table 3: Footprints for 32-bit and 16-bit DDR Interface

Total DRAM size	Devices for 32-bit I/F	Devices for 16-bit I/F
8 MB	1 device of 2M x 32 (64 Mbits)	1 device of 4M x 16 (64 Mbits)
16 MB	2 devices of 4M x 16 (64 Mbits) 1 device of 4M x 32 (128 Mbits)	1 device of 8M x 16 (128 Mbits)
32 MB	2 devices of 8M x 16 (128 Mbits) 1 device of 8M x 32 (256 Mbits)	1 device of 16M x 16 (256 Mbits)
64 MB	2 devices of 16M x 16 (256 Mbits) 1 device of 16M x 32 (512 Mbits)	1 device of 32M x 16 (512 Mbits)
128 MB	2 devices of 32M x 16 (512 Mbits)	n/a
256 MB	4 devices of 64M x 8 (512 Mbits) 1 rank 4 devices of 32M x 16 (512 Mbits) 2 ranks	n/a

The memory interface also performs the arbitration of the internal memory bus, guaranteeing adequate bandwidth and latency to the TM3260 CPU, DMA devices and other internal resources that require memory access. A programmable list-based memory arbitration scheme is used to customize the memory bandwidth usage of various hardware modules for a given application. The CPU in the system is given the ability to intersect long DMA transfers, up to a programmable number of times per interval. This allows optimal CPU performance at high DDR DMA utilization rate, and guarantees the real-time needs of audio/video DMA modules.

The memory controller supports most, if not all, DDR SDRAM devices thanks to programmable memory timing parameters. For example CAS latency, T_{RC}, T_{RAS}, T_{RP} and many others can be programmed after the default boot initialization.

4.2 Flash

NAND and NOR type flash memory connects to the PNX15xx Series by sharing some PCI bus pins. The XIO bus created by this pin-sharing supports 8- and 16-bit data peripherals, and uses a few side-band control signals. Refer to Section 10.3.2 on page 2-24 for more details.

PNX15xx Series provides 5 chip selects, one of which is intended for a Flash device. Address range, and wait states for a Flash device are programmable. The TM3260 can execute or read from direct addressable Flash types. Execution from Flash is low performance, and only recommended for boot usage. After boot, it is recommended that code files be transferred from Flash to DRAM where they can be executed more efficiently. Flash cannot be the target of a module DMA write, because write operations require a software flash programming protocol.

Execution and direct addressed read operations only apply to addressable Flash types, such as traditional Flash, and not to the “file system like” NAND Flash type.

Peak page mode read performance is 66 MB/s for 16-bit devices and 33 MB/s for 8-bit devices such as the configurable x8/x16 Intel® StrataFlash® (28FxxxJ3A, 32Mbits, 64Mbits, 128Mbits) and ST MLC-NOR flash (M58LW064A, 64Mbits). Cross-page random read accesses each take 4 to 5 PCI clock cycles depending on the access-time of the device.

Flash is mostly used during system boot or low bandwidth system operation to provide a small, non-volatile file system.

5. TM3260 VLIW Media Processor Core

The TM3260 CPU is a version of the TriMedia 32-bit VLIW media processor. This Very Long Instruction Word (VLIW) processor operates at up to 300 MHz with 5 instructions per clock cycle, and provides an extensive set of multimedia instructions. It implements the TriMedia PNX1300 Series instruction set, and has a superset of the PNX1300 Series functional units as well as a superset of the multimedia instruction set for better fit with MPEG-4 advanced profile decoding. It is backwards compatible with PNX1300 Series CPU, but has a larger Instruction cache (also referred as I$ or Icache) for improved performance. In addition, re-compilation of source code results in higher media performance due to the additional functional units.

The TM3260 supports 32-bit integer and IEEE compatible 32-bit floating point data formats. It also provides a Single Instruction Multiple Data (SIMD) style operation set for operating on dual 16-bit or quad 8-bit packed data.

- At 266 MHz it has a peak floating point compute capacity of 1.0 Goperations/s, and has 1.3 Gmultiply-add/s capability on 16-bit data. Its dual access 16 KB 8-way set-associative data cache provides a CPU local data bandwidth of 2.0 GB/s. Its 64 KB 8-way set-associative instruction cache provides 224 bits of instructions every clock cycle (7.1 GB/s), for an instruction rate of 8.8 Gop/s.
- At 300 MHz it has a peak floating point compute capacity of 1.4 Goperations/s, and has 1.5 Gmultiply-add/s capability on 16-bit data. Its dual access 16 KB 8-way set-associative data cache provides a CPU local data bandwidth of 2.3 GB/s. Its 64 KB 8-way set-associative instruction cache provides 224 bits of instructions every clock cycle (8.0 GB/s), for an instruction rate of 9.9 GB/s.

The TM3260 has sufficient compute performance to deal with a variety of future operating modes. By itself, the processor can decode any known compressed video stream and associated audio at full frame rate, such as decoding a DV camcorder

image stream, MPEG-2 or MPEG-4 decode. The processor is also capable of doing all audio and video compression, decompression and processing necessary for bi-directional video conferencing.

The TM3260 is responsible for all media processing and real-time processing functions within the PNX15xx Series. It runs a small real-time operating system, pSOS, which allows it to respond efficiently and predictably to real-time events.

The TM3260 is capable of operating in little or big-endian mode. The mode is chosen shortly after CPU startup by setting the endian bit in the Program Control Status Word (PCSW).

Debug of software running on TM3260 is performed using an interactive source debugger with a PC JTAG plug-in board. The PC talks to the TM3260 through the PNX15xx Series JTAG pins. The TMDBG module provides an improved version of the PNX1300 Series JTAG debug port. The PNX15xx Series is in standalone mode.

TM3260 media processor features are presented bellow.

Table 4: TM3260 Characteristics

TM3260 VLIW CPU Features	
ISA	PNX1300 Series, with 32-bit RISC style load/store/compute instruction set and an extensive set of 8-, 16-bit SIMD multimedia instructions
Instructions	5 RISC or SIMD instructions every clock cycle
Data types	boolean, 8-, 16- and 32-bit signed and unsigned integer, 32-bit IEEE floats
Functional units	5 CONST, 5 Integer ALU's, **5** multi-bit SHIFTERs, **3** DSPALU's, 2 DSPMUL, 2 IFMUL, 2 FALU, 1 FCOMP, 1 FTOUGH (divide, sqrt) 3 BRANCH, 2 LD/ST
Caches	64 KB 8-way set associative ICache 16 KB 8-way set associative dual-ported Dcache
Cache policies	critical word first refill, write-back, write-allocate, automatic heuristic hardware prefetch
Line size	64 bytes (both ICache and DCache)
MMU	none, virtual = physical, full 4 GB space supported
Protection	Base, limit style protection, where CPU can be set to only use part of system DRAM, and hardware ensures no references take place outside this range
Multipliers	up to 2 32x32-bit integer multiplies per clock up to 2 32-bit IEEE floating point multiplies per clock up to 4 16x16-bit multiply-adds per clock up to 8 8x8-bit multiplies per clock
Debug	JTAG based software debugger, including hardware breakpoints for instruction and data addresses
Register file	128 entry 32-bit register file
Interrupts	64 auto-vectoring interrupts, with 8 programmable priority levels
Timers	Four 32-bit timers/counters are provided. A wide selection of sources allows them to be used for performance analysis, real-time interrupt generation and/or system event counting

Table 4: TM3260 Characteristics

TM3260 VLIW CPU Features	
System Interface	The TM3260 runs asynchronously with respect to system DRAM, and can operate at a frequency lower than system DRAM to save power, or higher than system DRAM to gain performance
Software Development Environment	The TM3260 is supported by the advanced C/C++ compiler tools available for the PNX1300 Series family
Application Software Architecture	Applications use the TSSA, Trimedia Streaming Software Architecture, allowing modular development of audio, video processing functions

6. MPEG Decoding

The TM3260 processes the audio, video and the stream de-multiplexing via software. The Variable Length decoding as well as the authentication and the de-scrambling are supported by two coprocessors.

6.1 VLD

The PNX15xx Series VLD is an MPEG-1 and MPEG-2 parser that writes to memory a separate data structure for macro block header and coefficient information. It is capable of sustaining an ATSC (High Definition) bitrate. It off-loads the CPU in applications involving MPEG-2 decoding or transcoding. Low to medium bitrate VLD decoding, as well as VLC encoding may be done by the TM3260 CPU. MPEG-2 HD decoding by the CPU is not supported due to CPU and system limitations.

6.2 DVD De-scrambler

The DVD-CSS module is provided to allow integrated DVD playback capability.

It provides authentication and de-scrambling for DVDs. A DVD drive can be attached to the integrated medium-bandwidth IDE controller, and provides its data either across the IDE interface or across a multi bit serial interface to the GPIO pins. The resulting system memory scrambled program stream is de-scrambled by invoking a memory to memory operation on the DVD-CSS module. The ‘cleartext’ program stream is then de-multiplexed by software on the TM3260.

More detailed Information available on (legal) request

7. Image Processing

7.1 Pixel Format

The on-chip hardware image processing modules all use the same ‘native’ pixel formats, as shown in Table 5. This ensures that image data produced by one module can be read by another module.

- A limited number of *native pixel formats* are supported by all image subsystems, as appropriate.

- The Memory Based Scaler supports *conversion from arbitrary pixel formats* to any native format during the anti-flicker filtering operation. This operation is usually required on graphics images anyway, hence no extra passes are introduced.
- Hardware subsystems support all native pixel formats in both *little-endian* and *big-endian* system operation.
- Software always sees the *same component layout* for a native pixel format unit, whether it is running in little-endian or big-endian mode. i.e. for a given native format, R, G, B (or Y,U,V) and alpha are always in the same place.
- Software (on the TM3260 CPU) can be written endian-mode independent, even when doing SIMD style vectorized computations

Remark: The native formats of PNX15xx Series include the most common indexed, packed RGB, packed YUV and planar YUV formats used by Microsoft DirectX and Apple Quicktime, with 100% bit layout compatibility in little and big-endian modes of operation, respectively.

Remark: TM3260 software image processing stages and encoders/decoders typically use semi-planar or planar 4:2:0 or 4:2:2 formats as input and output.

Table 5: Native Pixel Format Summary

Name	Note	VIP out	MBS in	MBS out	2D engine (2)	QVCP-LCD in
1 bpp indexed	CLUT entry = 24-bit color + 8-bit alpha		x			x
2 bpp indexed			x			x
4 bpp indexed			x			x
8 bpp indexed			x		x	x
RGBa 4444	16-bit unit, containing one pixel with alpha	(1)	x	x	x	x
RGBa 4534		(1)	x	x	x	x
RGB 565	16-bit unit, containing one pixel, no alpha	(1)	x	x	x	x
RGBa 8888	32-bit unit, containing one pixel with alpha	(1)	x	x	x	x
packed YUVa 4:4:4	32-bit unit containing one pixel with alpha	x	x	x	x	x
packed YUV 4:2:2 (UYVY)	16-bit unit, two successive units contain two horizontally adjacent pixels, no alpha	x	x	x		x
packed YUV 4:2:2 (YUY2, 2vuy)		x	x	x		x
planar YUV 4:2:2	three arrays, one for each component	x	x	x		
semi-planar YUV 4:2:2	two arrays, one with all Ys, one with U and Vs	x	x	x		x
planar YUV 4:2:0	three arrays, one for each component		x	x		
semi-planar YUV 4:2:0	two arrays, one with all Ys, one with U and Vs		x	x		x

1. VIP output of RGB is mutually exclusive with horizontal scaling
2. Shown are the 2D engine frame buffer formats where drawing, RasterOps and alpha-blending of surfaces can be accelerated. Additionally, the 2D Drawing Engine host port supports 1 bpp monochrome font/pattern data, and 4 and 8-bit alpha only data for host-initiated anti-aliased drawing.

7.2 Video Input Processor

The Video Input Processor (VIP) handles incoming digital video and processes it for use by other components of the PNX15xx Series. VIP provides 10-bit accurate processing. The VIP provides the following functions:

- Receives 10-bit YUV4:2:2 digital video data from the video port. The data is dithered down to in-memory 8-bit data format. The YUV4:2:2 data stream typically comes from devices such as the SAA 711x, which digitize PAL or NTSC analog video.
- Stores video data inside the video acquisition window in system memory in any of the native pixel formats indicated in Table 5, and performs error feedback rounding to convert the10-bit input to the selected format.
- Provides an internal Test Pattern Generator with NTSC, PAL, and variable format support.
- Acquires VBI data using a separate acquisition window from the video acquisition window.
- Performs horizontal scaling, cropping and pixel packing on video data from a continuous video data stream or from a single field or frame.
- ANC header decoding or window mode for VBI data extraction.
- Horizontal up scaling up to 2x.
- Interrupt generation for VBI or video written to memory.
- SD pixel frequency up to 81 MHz input clock (SD using up to 10-bit YUV CCIR-656).
- HD pixel frequency up to 81 MHz input clock (HD using 20-bit Y,UV input mode).
- color space conversion (mutually exclusive with scaling).
- raw data capture up to 81 MHz in either 8- or 10-bit, packed mode with double buffering.

VIP shares its allocated pins with the FGPI module through an input router. Section 9. shows the different operating modes of VIP and FGPI modules.

7.3 Memory Based Scaler

The PNX15xx Series contains a Memory Based Scaler that performs operations on images in main memory. The scaler hardware can either be controlled task by task by the TM3260, or it can be given a list of scaling tasks. The performance of the scaler on large images is typically limited either by the 120 Mpixel/s internal processing rate or by the allocated main memory bandwidth.

The PNX15xx Series MBS can perform:

- de-interlacing using either a median, 2-field majority select, or 3-field majority select algorithm with an edge detect/correct post-pass (these three provide increasing quality, at the expense of increased bandwidth requirements)

- edge detect/correct on an input frame that has been software de-interlaced (this provides future capabilities in case we develop a better core de-interlacer than 3-field majority select)
- horizontal & vertical scaling (on the input image, or on the result of edge detect/correct stage)
- linear and non-linear aspect ratio conversion
- anti flicker filtering
- conversions from any input pixel format to any non-indexed pixel format, including conversions between 4:2:0, 4:2:2 and 4:4:4, indexed to true color conversion, color expansion / compression, de-planarization/planarization (to convert between planar and packed pixel formats, programmable color space conversion)
- luminance histogram collection, during a scaling or de-interlacing pass
- note that not all combinations of format conversion with scaling are supported

The video processing functions are based on 4- & 6-tap polyphase filters with up to 64 phases. Three 6-tap filter units are used for horizontal scaling/filtering while three 4-tap filter units are assigned to vertical scaling/filtering. For some video formats (e.g. YUV 4:2:x) the three 4-tap filters can be combined to work as two 6-tap filters.

7.4 2D Drawing and DMA Engine

A 2D rendering and DMA engine (‘2D DE’) is included to perform high speed 2D graphics operations. Solid fills, three operand BitBlt, lines, and monochrome data expansion are available. Supported drawing formats include 8-, 16-, and 32-bit/pixel. Monochrome data can be color expanded to any supported pixel format. Anti-aliased lines and fonts are supported via a 16 level alpha blend BitBlt. A full 256 level alpha BitBlt is available to blend source and destination images together. Drawing is supported to any naturally aligned memory location and at any naturally aligned image stride, i.e. 16- and 32-bit pixels should be allocated at byte addresses that are a multiple of 2 and 4 respectively.

7.5 Quality Video Composition Processor

The PNX15xx Series Quality Video Composition Processor (QVCP) provides a high resolution graphics controller with graphics and video processing. The QVCP in combination with other modules such as the 2D Drawing Engine and the MBS (Memory Base Scaler) provides a new generation of graphics and video capability far exceeding the PNX1300 Series family.

QVCP allows composition of 2 layers, and can output in 656/HD/VGA or LCD format in up to 10-bit per component and up to 81 Mpixel/s.

QVCP contains a series of layers and mixers. The QVCP creates a series of display data layers (pixel streams) and mixes them logically from back to front to create the composited output picture. In order to achieve high quality video and graphics, the QVCP performs the following tasks:

- Fetching of the image surfaces from memory

- Per component table lookup, allowing de-indexing or gamma equalization
- Video Quality Enhancement (Luminance Transient Improvement, Color Dependent Sharpening, Horizontal Dynamic Peaking, Histogram Modification, Digital Color Transient Improvement, Black Stretch, Skin Tone Correction, Blue Stretch and Green Enhancement)
- Video and Graphics horizontal up scaling
- Color space unification of all the display surfaces
- Contrast and Brightness Control
- Positioning of the various surfaces
- Merging of the image surfaces (alpha blending and pixel selection based on chroma range keying)
- Screen timing generation adopted to the connected display requirements (SD-TV standards, HD-TV standards, progressive, interlaced formats, LCD panel control)

QVCP supports the semi-planar YUV formats for one layer. Both layers support only indexed, RGB and packed YUV formats. QVCP does not support planar video formats. See Table 5 for more details.

The mixer stage combines images from back to front, also allowing mixing in of a fixed backdrop color. The mixing operation can be controlled by chroma range keying. Mixing modes include per-pixel alpha blending, and color inverting. Mixing operations can be programmed by a set of raster operations (ROP). Mixing is performed either entirely in the RGB domain or the YUV domain, depending on the output mode of operation of the QVCP. After mixing, post-processing optionally down samples 4:4:4 to 4:2:2 in the Chroma Down Sampler (CDS). Then, VBI insertion may be performed (656 mode only), and the output is formatted to one of the forms as described below:

- 24- or 30-bit full parallel RGB or YUV
- 16- or 20-bit Y and U/V multiplexed data
- 8- or 10-bit 656 (full D1, 4:2:2 YUV with embedded sync codes)
- 8- or 10-bit 4:4:4 format in 656-style with RGB or YUV

In each of the output modes, optional H-sync, V-sync and blanking or odd/even outputs are available.

The QVCP can be slaved to an external timing source that provides a pixel clock and a frame sync, i.e. VSYNC. The horizontal sync reference is taken from the frame sync. Synchronizing to a traditional field-based Vertical Sync is not supported. The clock direction is programmed in the clock module while the VSYNC direction, pin VDO_D[29] is programmed in the QVCP module.

PNX15xx Series contains a TFT LCD controller. It has integrated control of the synchronization signals but also all the LCD specific commands like power management. De-Interlacing of video material is provided in the MBS module. Dithering is handled by the QVCP-GNSH block.

The QVCP has separate synthesizers for pixel-clock generation. Software may use these synthesizers to achieve perfect lock to the transmission source of the digital video that is being displayed by the QVCP.

QVCP shares its allocated pins with the FGPO module through an output router. Refer to Section 9. for the different operating modes of QVCP and FGPO and pin allocation.

7.5.1 External Video Improvement Post Processing

The PNX15xx Series has a 'VDO_AUX' output pin that can be set to signal whether a pixel is a graphics or video pixel. This can be used to suppress post-processing on graphics elements for an attached proprietary video improvement post processor.

Motion vectors computed by TM3260 software can be sent to a video improvement post-processor over the PCI interface.

The function of VDO_AUX is programmed using the QVCP capability to combine alpha or chroma-keying information during blending. For example, chroma keys in a graphics plane could be used to drive VDO_AUX. For another example, a threshold value for an alpha value of a graphics plane could be used to indicate whether a pixel is more than 80% video.

8. Audio processing and Input/Output

8.1 Audio Processing

All audio processing in PNX15xx Series is performed in software on the TM3260. This includes decoding of audio from compressed formats, sample rate conversion, mixing and special effects processing. There is sufficient performance, if required, to transcode received audio to multi-channel compressed audio sent over S/PDIF to an attached receiver. (should this say "from an attached receiver" ?)

8.2 Audio Inputs and Outputs

The PNX15xx Series has several Audio Input/Output facilities:

- PNX15xx Series Audio In can capture up to 8 stereo audio inputs with up to 32-bit/sample precision at sample rates up to 96 kHz. Both Audio In and Audio Out support most A/D converter serial protocols, including I^2S. Sample rate is internally or externally generated. The internal generator is programmable with sub one Hertz sampling rate accuracy. Audio In also includes a raw mode which allows the capture of any quantity of bits out of the programmable frame (up to 512 bits per frame). The word strobe (AI_WS pin) is also captured and stored into memory.
- PNX15xx Series Audio Out can generate up to 8 channels of audio, and directly drives up to 4 external stereo I^2S or similar D/A converters. It supports up to 32-bit/sample precision at sample rates up to 96 kHz. The sample rate can be internally or externally generated. The internal generator is programmable with sub one Hertz sampling rate accuracy. Audio out does not include a raw mode as the Audio In module does.
- PNX15xx Series supports a SPDIF (Sony Philips Digital Interface) output with IEC-1937 capabilities. Transmitted data is generated by TM3260 software. This output port can carry either stereo PCM samples from an internal audio mix, or

one of the originally received compressed audio programs (5.1 channel AC-3, multi-channel MPEG audio, multi-channel AAC). Sample rate of transmitted audio is set by software, allowing perfect synchronization to any time reference in the system.

- PNX15xx Series supports a SPDIF input to connect to external sources, such as a DVD player. The incoming data is timestamped and written to unified system memory. Data interpretation and sample rate recovery is achieved by software on the TM3260. The audio data received can be in a variety of formats, such as stereo PCM data, 5.1 channel AC-3 data per IEC-1937 or other. Software decoded audio can be used for mixing with other audio for output along one of the audio outputs. The sample rate is determined by the S/PDIF source, and cannot be software controlled.

9. General Purpose Interfaces

VIP and QVCP share a set of pins with two general purpose interface modules, FGPI and FGPO (respectively). The input and output data routers allocate a different amount of pins between these four modules. The allocation depends on the operating mode of each module. The following sections describe the different modes of the input and output routers.

9.1 Video/Data Input Router

These inputs can provide combinations of the following functions:

- capture of video streams into DRAM, while performing horizontal scaling and conversion to one of the standard pixel formats, simultaneously with data stream capture
- low-latency reception of messages from another PNX15xx Series
- capture of unstructured, infinite parallel data streams into DRAM
- capture of 1 or 2-dimensional parallel data streams in DRAM
- for message passing and data modes, operating speeds of up to 100 MHz, with 8-, 16- or 32-bit parallel data are supported, providing an aggregate input bandwidth of up to 400 MB/s

The VDI pins consist of 38 pins, split into 32 data pins, 2 clock pins and 2 valid signals that indicate whether data is valid on the respective clocks.

12NC 9397 750 14321

The operating modes of the video/data input router are set by the VDI_MODE MMIO register. A subset of the operating modes are presented in Table 6, which combines 656 digital video source with streaming data inputs. A complete behavior of the output router is available in Section 7. on page 3-16. Section 7.2 summarizes the VIP features, while Section 9.3 presents some of the FGPI capabilities.

Table 6: Video/Data Input Operating Modes

mode	VIP function	FGPI function
VDI_MODE[1:0] = 0x0 (Default after reset)	8- or 10-bit ITU 656 with additional H&V synchronization signals or 8- or 10-bit raw data	up to 22-bit data capture. FGPI is usually set in 16- or 32-bit mode storing into main memory respectively 16- or 32-bit words
VDI_MODE[1:0] = 0x1	20-bit ITU 656 as for HD video with additional H&V synchronization signals	up to 12-bit data capture. FGPI can be set in 8-, 16-, or 32-bit mode storing into main memory respectively 8-, 16-, or 32-bit words
VDI_MODE[1:0] = 0x2	8-bit ITU 656 or 8-bit raw data	up to 24-bit data capture FGPI is usually set in 32-bit mode storing 32-bit words into main memory.
VDI_MODE[1:0] = 0x3	n/a	32-bit data capture. FGPI is usually set to 32-bit mode.

In addition to controlling the operating mode of the VDI pins, VDI_MODE[7] bit controls the activation of a pre-processing module for the 8-bit data that is routed to the FGPI module. When VDI_MODE[7] = ‘1’ then the input router scans the lower VDI_D[7:0] inputs for SAV/EAV codes as defined in the video CCIR 656 standard and uses the ‘start’ and ‘stop’ signals that are routed to the FGPI module as a line and field detector. FGPI can then be programmed to store in DRAM each field or line at a specific location which eases the software processing of the data. This processing stage allows to use of FGPI as a second Video Input as long as ‘on the fly’ pixel processing is not required.

A subset of the VDI pins can individually be set to operate as GPIO pins in case they are not used for their primary video/streaming data function.

9.2 Video/Data Output Router

The output router can provide certain combinations of the following functions:

- Refresh a TFT LCD display up to W-XGA (1280*768) at 60 Hz with RGB 18/24-bit per pixel.
- Refresh progressive or interlaced standard definition video screens using ITU 656 with YUV4:2:2 or 4:4:4 data, with each screen receiving pixels resulting from the composition and processing of two display surfaces stored in DRAM.
- Refresh of a single high-definition[1] or VGA resolution screen.

1. PNX15xx Series *does not* have the bandwidth and processing power to do a full HDTV decode/process and HD display, but it can refresh a HD screen and present graphics and video windows on such a screen.

- Broadcast of messages to 1 or more receiving PNX15xx Series's.
- Message or unstructured data transmission is in 8-, 16- or 32-bit parallel format, with data rates up to 100 MHz, providing an aggregate data rate of up to 400 MB/s.

The VDO pins consist of 39 pins, split into 32 data pins, 2 clock pins and 4 control signals.

The operating modes of the video/data output router are set by the VDO_MODE MMIO register. A subset of the operating modes is presented in Table 7. A complete behavior description of the output router is available in Section 7. on page 3-16. Section 7.5 provides a description of the Video generation capabilities of the QVCP module, while Section 9.4 briefly describes the data streaming/generation features of the FGPO module.

Table 7: Video/Data Output Operating Modes

mode	QVCP function	FGPO function
VDO_MODE[2:0] = 0x0 (Reset)	TFT LCD controller with 24- or 18-bit digital RGB output and associated control signals.	3- or 8-bit data streaming. FGPO is usually set in 8-bit mode.
VDO_MODE[2:0] = 0x1	Digital ITU 656 YUV 8-/10-bit and Hsync, Vsync and Cblank signals.	19-bit data streaming. FGPO is usually set in 16- or 32-bit mode, but only the 19 lower bits are output per 16- or 32-bit words.
VDO_MODE[2:0] = 0x2	Digital 16-bit YUV and Hsync, Vsync and Cblank signals.	13-bit data streaming. FGPO can be set in 8-, 16- or 32-bit mode, but only the 13 lower bits are output per 8-, 16- or 32-bit words.
VDO_MODE[2:0] = 0x3	Digital 20-bit YUV and Hsync, Vsync and Cblank signals.	9-bit data streaming. FGPO is usually set in 8- or 16-bit mode, but only the 9 lower bits are output per 8- or 16-bit words.
VDO_MODE[2:0] = 0x4	Digital 24-bit YUV or RGB and Hsync, Vsync and Cblank signals.	5-bit data streaming. FGPO is usually set in 8-bit mode, but only the 5 lower bits are output per 8-bit words.
VDO_MODE[2:0] = 0x5	Digital 30-bit YUV or RGB and Hsync, Vsync and Cblank signals.	n/a
VDO_MODE[2:0] = 0x6	Digital ITU 656 YUV 8-bit	24-bit data streaming. FGPO is actually set in 32-bit mode but only the 24 lower bits are output per 32-bit words.
VDO_MODE[2:0] = 0x7	n/a	32-bit data streaming.

A subset of the VDO pins can individually be set to operate as GPIO pins in case they are not used for their primary video/streaming data function.

9.3 Fast General Purpose Input

The Fast General Purpose Input (FGPI) captures data in a variety of modes:

- raw mode 8 or 16-bit parallel data. The data is continuously captured as soon as enabled, and is written to memory using double buffering to prevent loss of data

- 8-, 16- or 32-bit message passing between PNX15xx Series's. Messages of up to 16 MB in length are received and written to memory. Upon completion, an interrupt is generated, and the FGPI switches to the next software input buffer.
- 8-, 16- or 32-bit structured data capture. Data is captured in records, using the REC_START signals to designate when records are started. The BUF_START signal can, optionally, be used to force a software buffer switch. This mode can be used to capture 2-dimensionally structured data, such as raw video samples.
- In combination with VDI_MODE[7] bit, see Section 9.1, FGPI can be used as a basic Video In module by storing in memory at specific locations the different lines and fields of the in-coming video data. Note that the YUV data is stored consecutively in memory and not stored in three different planes.

9.4 Fast General Purpose Output

The Fast General Purpose Output (FGPO) provides data generation capabilities that match the FGPI:

- generation of a structured data stream, indicating record and buffer start over two control wires. Generated data can be 8-, 16- or 32-bit wide, with data rates up to 100 MHz at respectively 100, 200 and 400 MB/s.
- message passing (8-, 16- or 32-bit wide)
- External synchronization available

10. Peripheral Interface

10.1 GPIO - General Purpose Software I/O and Flexible Serial Interface

PNX15xx Series has 16 dedicated GPIO pins. In addition, 45 other pins that have a high likelihood of not being used in certain applications are designated as optional GPIO pins that can either operate in regular mode or in GPIO mode. As an example, some of the data pins of the LAN module are available as fully functional GPIO in case the system based on PNX15xx Series is not connected to a LAN network module. The complete list is available in the pin list where a dedicated column defines the GPIO pin number, see Section 2.3 on page 1-3.

The GPIO module is connected to many pins. Hence it is the ideal place to provide useful central system functions. It performs the following major functions, each detailed below:

- software I/O - set a pin or pin group, enable a pin (group), inspect pin values
- precise timestamping of internal and external events (up to 12 signals simultaneously)
- signal event sequence monitoring or signal generation (up to 4 signals simultaneously)

10.1.1 software I/O

Each GPIO pin is a tri-state pin that can be individually enabled, disabled, written or read. Pins are grouped in groups of 16 and signals within a group can be simultaneously enabled and changed or observed. Changes can use a mask to allow certain pins to remain unchanged.

Note that this capability is useful for low/medium speed software implemented protocols, as well as for observing switches, driving LEDs etc. It is highly recommended to first use the powerful GPIO pins as protocol emulators, and not just for static switches/LEDs (for which a solution such as a PCF8574 I^2C parallel I/O is fine).

10.1.2 timestamping

The GPIO module contains 12 timestamp units, each of which can be designated to monitor an external GPIO pin or internal system event. For a monitored event, a timestamp unit can be set to trigger on a rising edge, falling edge or either edge. When a trigger occurs, a precise occurrence time (31-bit timestamp value, 75 ns resolution) is put in a register, and an interrupt is generated.

This capability is particularly valuable for precise monitoring of key audio/video events and controlling the internal software phase-locked loops that lock to broadcast time references. It can also be used for medium speed signal analysis.

10.1.3 event sequence monitoring and signal generation

GPIO contains 4 queue units, each capable of monitoring or generating high-speed signals on up to 4 GPIO pins.

This capability creates a universal protocol emulator, capable of emulating many medium speed (0 - 20 Mbit/s) protocols using software on the TM3260 media processor. Complex protocols, such as the MemoryStick™ protocol with 20 Mbits/s peak rate and 800 KB/s sustained file transfer rate have been successfully implemented on the PNX8525 GPIO module. The PNX15xx Series GPIO is similar to the PNX8525 GPIO module.

High speed signal analysis uses one of two modes:

- event queue hardware samples 1, 2 or 4 GPIO inputs using one out of a variety of clocks in the system, including clock inputs or clocks generated from other GPIO pins. Samples are packed in a word and stored in a list in system memory for software analysis.
- event queue hardware builds an in-system memory list of timestamped GPIO pin change events, individual per monitored GPIO pin. Edge events are timestamped with 75 ns resolution.

Signal generation uses the same 2 features, but in reverse, i.e. a sampled signal is transmitted, or an in-memory timestamped list of change events is output over a pin.

The event sequence monitoring mechanism can be used for many functions, and is particularly useful for interpreting Remote Control commands, as described in Section 10.2. Signal generation is useful for RC Blaster applications.

The GPIO module has a total of 4 complex signal analysis/signal synthesis resources capable of sampling or timestamped list generation/creation.

10.1.4 GPIO pin reset value

Dedicated GPIO pins come in two types:

- 50% of the pins will have a ‘low’ reset value
- 50% have a ‘high’ reset value

This allows use of GPIO for a variety of functions.

10.2 IR Remote Control Receiver and Blaster

PNX15xx Series uses the GPIO pin event sequence timestamping mechanism and software to interpret remote control commands. The event sequence timestamping can resolve events on signal edges with 75 ns accuracy. A sequence of events followed by a period of inactivity causes generation of an interrupt. Software then interprets the ‘character’ by looking at the event list consisting of (time, direction) encoded in memory.

This allows interpretation of a wide variety of Remote Control protocols. The Philips RC-5, RC-6 and RC-MM remote control protocols are all decoded with this mechanism, provided that the RF demodulation is performed externally. Most other Consumer Electronic vendor remote control protocols can be supported by appropriate software.

Similarly, the event generation mechanism can be used to implement IR blaster capability. In this case, the modulator is included - the software generated pulses can be superimposed on an internally generated carrier.

There are some speed considerations with this mechanism. Each character communicated generates at least one interrupt, and possibly more if the number of edge events exceeds the FIFO size. Hence, this mechanism is suitable only for protocols that use frequencies up to a few 10’s of kHz, with low character repetition rates, and not for high speed protocols.

10.3 PCI-2.2 & XIO-16 Bus Interface Unit

PNX15xx Series contains an expansion bus interface unit ‘PCI/XIO-16’ that allows easy connection of a variety of board level memory components and peripherals. The bus interface is a single set of pins that allows simultaneous connection of 32-bit PCI master/slave devices as well as separated address/data style 8- and 16-bit micro processor slave peripherals and standard (NOR) or disk-type (NAND) Flash memory.

The bus interface unit contains a built-in single-channel DMA unit that can move blocks of data to or from an external peripheral (PCI bus master or slave) to or from PNX15xx Series DRAM. The DMA unit can access PCI as well as 8- and 16-bit wide XIO devices. The DMA unit packs XIO device data to/from 32-bit words, so that no CPU involvement is required to pre/post process data.

10.3.1 PCI Capabilities

PNX15xx Series complies with Revision 2.2 of the PCI bus specification, and operates as a 32-bit PCI master/target up to 33 MHz.

PNX15xx Series as PCI master allows TM3260 to generate all single cycle PCI transaction types, including memory cycles, I/O cycles, configuration cycles and interrupt acknowledge cycles. As PCI target, PNX15xx Series responds to memory transactions and configuration type cycles, but not to I/O cycles.

PNX15xx Series can act as PCI bus arbiter for up to 3 external masters, i.e. total of 4 masters with PNX15xx Series, without external logic.

PCI clock is an input to PNX15xx Series, but if desired the general purpose PNX15xx Series PCI_SYS_CLK clock output can be used as the PCI 33 MHz clock for the entire system.

Table 8 summarizes the PCI features supported by the PNX15xx Series.

Table 8: PNX15xx Series PCI capabilities

As PCI Target it responds to	As PCI master it initiates
	IO Read
	IO Write
Memory Read	Memory Read
Memory Write	Memory Write
Configuration Read	Configuration Read
Configuration Write	Configuration Write
Memory Read Multiple	Memory Read Multiple
Memory Read Line	Memory Read Line
Memory Write and Invalidate	Memory Write and Invalidate
	Interrupt Acknowledge

10.3.2 Simple Peripheral Capabilities (‘XIO-8/16’)

The 16-bit micro-processor peripheral interface is a master-only interface, and provides non-multiplexed address and data lines. A total of 26 address bits are provided, as well as a bi-directional, 16-bit data bus. Five device profiles are provided, each generating a chip-select for external devices. Up to 64 MB of address space is allowed per device profile. The interface control signals are compatible with a Motorola 68360 bus interface, and support both fixed wait-state or dynamic completion acknowledgment.

A total of 5 pre-decoded Chip Select pins are available to accommodate typical outside slave configurations with minimal or no external glue logic. Each chip select pin has an associated programmable address range within the XIO address space. Each chip select pin can also choose to obey external DTACK completion signalling, or be set to have a pre-programmed number of wait cycles.

The peripheral interface derives 24 of the 26 address wires and 8 out of the 16 data wires from the PCI AD[31:0] pins. The remaining pins are XIO specific and non PCI shared. An ‘XIO’ access looks like a valid PCI transaction to PCI master/targets on the same wires. Unused XIO pins are available as GPIO pins.

The table below summarizes extension capabilities of the bus interface unit.

Table 9: PCI/XIO-16 Bus Interface Unit Capabilities

External Device	Device Type	Capabilities
external PCI master	32-bit, up to 33 MHz PCI masters	Arbitration built-in for up to 3 external PCI masters. Additional external masters can be supported with external arbitration. External PCI bus masters can perform high bandwidth, low latency DMA into and out of PNX15xx Series DRAM. Large block transfer capable devices can sustain up to 100 MB/s into DRAM.
external PCI slave	32-bit, up to 33 MHz PCI targets	Glueless connection supported for multiple devices subject only to capacitive loading constraints. The TM3260 can perform low-latency 8/16/32-bit writes and reads to/from PCI targets. Access by TM3260 can be enabled or disabled.
external 8-bit slave	8- and 16-bit wide, de-muxed address / data devices on ‘XIO bus’	Up to 5 devices supported gluelessly, or unlimited number subject to capacitive loading rules with external address decode logic. The TM3260 can perform 8-. 16- or 32-bit reads and writes to these ‘XIO’ devices, which are automatically mapped to 8- or 16 bit wide transfers by the bus interface unit.
standard (NOR) Flash	8- and 16-bit wide	PNX15xx Series provides 5 chip selects, one of which is available for a Flash device. Address range, and wait states for a Flash device are programmable. The TM3260 can execute or read from Flash. Execution is low performance, and only recommended for boot usage. The TM3260 can re-program Flash using special software. Flash cannot be the target of a module DMA write - writes require a software flash programming protocol. Peak page mode read performance is at 66 MB/s for 16-bit devices and 33 MB/s for 8-bit devices such as Intel StrataFlash (28FxxxJ3A, 32 Mbits, 64 Mbits, 128 Mbits) and ST MLC-NOR flash (M58LW064A, 64 Mbits). Cross-page random read accesses each take 4 to 5 PCI clock cycles at 33 MHz depending on the access-time of the device. Flash is mostly active during system booting, or with low bandwidth during system operation in order to implement a small non-volatile file system.
NAND Flash	8- and 16-bit wide	Direct execution, direct PI bus read or direct PI bus write from this Flash type are not supported. Explicit programmed I/O through special NAND Flash PCI/XIO-8/16 control/status registers is used to implement a file system on this disk-like Flash type. Using the NAND-Flash XIO provisions, a peak bandwidth of 13 MB/s, and a sustained bandwidth of 11 MB/s can be obtained from a AM30LV0064D 8Mx8 UltraNAND or equivalent Flash device. Maximum throughput for serial burst accesses is 33 MB/s for 16-bit devices such as a Samsung K9F5616U0B (16 Mbits x 16).
CIMaX device	8-bit data, 26-bit address	The external logic for conditional access consists of a CIMaX device, with 2 PCMCIA slot devices and glue logic (373, 245). This entire subsystem behaves as an 8-bit wide slave with an up to 26-bit address space. This subsystem interfaces gluelessly to the XIO bus, except for the possible logic needed to combine the DTACK signalling of multiple devices. There is medium bandwidth of communication between CIMaX and PNX15xx Series, which is expected to not be an issue w.r.t. PCI performance.
1394 link core	8-bit data and 9-bit address (Philips PDI1394LXX)	The Philips PDI1394LXX family connects gluelessly to XIO in 8-bit data mode using 8-bit data and 9-bit address with dedicated read and write strobes, optional wait signal and a separate chip select. For systems which require high asynchronous performance a 1394 link device with direct PCI connection can be used.

Table 9: PCI/XIO-16 Bus Interface Unit Capabilities

External Device	Device Type	Capabilities
DOCSIS devices		Future DOCSIS devices are expected to be PCI bus mastering devices. They connect gluelessly.
external SRAM, ROM, EEPROM	8- and 16-bit wide	Counts as generic XIO slave device.
external DRAM	not supported	not supported on PCI/XIO.
external Motorola style masters	not supported	PNX15xx Series PCI/XIO does NOT support external Motorola style masters. PNX15xx Series assumes that it is always the master over the XIO bus.
external 8/16-bit XIO DMA devices	not supported	not supported. Use one of the streaming DV inputs or outputs instead.

10.3.3 IDE Drive Interface

The PNX15xx Series contains an IDE controller that uses some of the PCI pins and a few sideband signals. Two external TTL devices are all that is required to interface to an actual IDE cable/drive. The IDE controller capabilities are:

- controls attached disks in PIO mode, for a peak data rate of 16.6 MB/s (PIO4)
- supports sustained bandwidth of up to 10 MB/s
- sends DMA blocks of disk data to and from system DRAM
- all IDE registers are accessible to TM3260 software

10.4 10/100 Ethernet MAC

The PNX15xx Series integrates a 10/100 Ethernet MAC sub-layer of the IEEE 802.3 standard enabling an external PHY to be attached through a Reduced Media Independent Interface (RMII) or a standard Media Independent Interface (MII). It implements dual transmit descriptor buffers, support for both real-time and non-real-time traffic and support for quality of service (QoS) using low-priority and a high-priority transmit queues. Among other features the 10/100 Ethernet MAC module includes:

- Wake-on-LAN power management support. This allows system wake-up using receive filters or a magic packet detection filter.
- Receive filtering with perfect address matching, a hash table imperfect filter and 4 pattern match filters.
- Memory traffic optimization via buffering and prefetching

The MAC address is programmable into an MMIO register. The MAC address could be located in an externally attached EEPROM.

11. Endian Modes

PNX15xx Series fully supports little- and big-endian software stacks.

PNX15xx Series always starts in a fixed endian mode which is determined by the boot script. There is a system provision for TM3260 software to reset and restart the TM3260 in the opposite endian mode such that a field software Flash upgrade can release a ‘endian mode opposite boot’ software upgrade.

PNX15xx Series on-chip modules and co-processors observe the system global endian mode flag. The TM3260 endian mode can be set by the TM3260 program module itself, and should always be set identical to system endian mode.

When selecting PCI peripherals for a dual-endian mode product, care must be taken to ensure that they can operate without ‘CPU fixup’ in either endian mode. Typically, PowerPC compatible PCI devices support both endian-modes in the exact same way as the PNX15xx Series.

12. System Debug

PNX15xx Series uses the JTAG port for both the purpose of boundary scan, as well as to implement a remote debug capability for software running on the PNX15xx Series CPU. By connecting a PC (running the Trimedia SDE Debugger) through JTAG to a PNX15xx Series, full start-stop/breakpoint/download type interactive debugging is possible.

Chapter 3: System On Chip Resources

PNX15xx Series Data Book – Volume 1 of 1

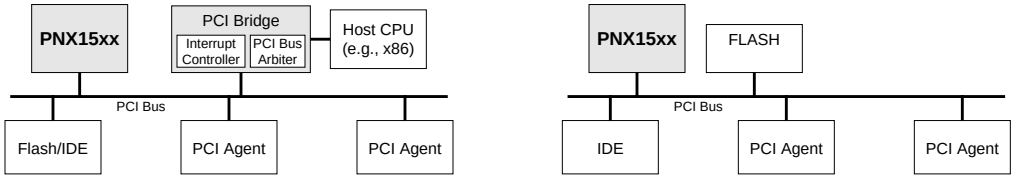
Rev. 2 — 1 December 2004 Product data sheet

1. Introduction

This chapter presents information on the PNX15xx Series System On-Chip (SOC) and its MMIO registers. Further details on each module composing PNX15xx Series are available on dedicated chapters though this databook. Reading this chapter is recommended before jumping to the individual module documentation.

2. System Memory Map

PNX15xx Series is designed to work in two different environments: standalone and host mode (Figure 1). In standalone mode PNX15xx Series retrieves its program (i.e. the software application that runs on the TM3260 CPU) from an EEPROM or a Flash memory device. In this mode the PNX15xx Series acts as the master. In host mode PNX15xx Series program is downloaded into the PNX15xx Series main memory before the TM3260 CPU is released from reset. In this mode the PNX15xx Series acts as a slave. This mode is typically used for a PCI plug-in card or a standalone system where a control processor is the master. In both modes the PCI bus is the main bus used to attach other components of the board system. In order to successfully get all these components working together, it is important to understand PNX15xx Series system memory map and its bus structure.



a) PNX15xx Series in host mode

b) PNX15xx Series in standalone mode as the host (i.e.

Figure 1: The Two Operating Modes of PNX15xx Series

Following the PCI memory addressing principles, PNX15xx Series system provides several apertures in its 32-bit address space to communicate to the other devices through the PCI bus. At system level, there are three different views of these apertures. The view from the TM3260 CPU, the view from the internal bus, called DCS, and the view from the PCI module. The DCS view is introduced to present the overall view of the system memory map.

Before going into the details of the three different views the following generic rules should be noted:

- The three views must be consistent. For example, it is not allowed to have a different DRAM aperture location for the TM3260 CPU and the PCI module.
- The apertures are “naturally aligned”. For example a 32-Megabyte aperture has a starting address that is a multiple of 32 Megabytes.
- Each aperture can be located anywhere in the 32-bit addressing space.
- All the modules in the PNX15xx Series SOC sees the same memory map, i.e. an address represents an unique location for all the modules.

These apertures need to be programmed at boot time or by the host before the system can be operational. The internal boot scripts have pre-defined values for these apertures (refer to Chapter 6 Boot Module).

2.1 The PCI View

The PCI module provides three different apertures to the external PCI bus masters:

- the MMIO aperture, used to access all the internal PNX15xx Series registers. See Section 11. on page 3-31 for offset allocation per module.
- the DRAM aperture, used to access to the main memory of PNX15xx Series.
- the XIO aperture, used by TM3260 to access low speed slave devices like Flash memories or IDE disk drives.

Any supported request on the PCI bus that falls outside of these three apertures is discarded by the PCI module and therefore does not interfere with the PNX15xx Series system.

In addition PCI transactions to the XIO aperture from external PCI agents are discarded.

Figure 2 presents the memory map seen by the PCI module and the remaining of the PNX15xx Series system. The apertures can be placed in any order with respect to each other.

The aperture locations is programmed by the host CPU.

The aperture sizes can be programmed at boot time via some GPIO/BOOT_MODE[] pins as defined in Chapter 6 Boot Module or they can be programmed by the host CPU using PCI configuration cycles.

- The MMIO aperture is starting at the address contained in the BASE_14 PCI configuration space register.
- The DRAM aperture is starting at the address contained in the BASE_10 PCI configuration space register.
- The XIO aperture is starting at the address contained in the BASE_18 PCI configuration space register.

Remark: Partial 32-bit load or stores from a PCI master to an MMIO register is not supported. Therefore byte of 16-bit half-word accesses are not supported.

2.2 The CPU View

The TM3260 CPU supports three different apertures:

- the MMIO aperture, used to access all the internal PNX15xx Series registers. See Section 11. on page 3-31 for offset allocation per module.

Remark: To ensure backward compatibility with future devices, writes to any undefined or reserved MMIO bit should be ‘0’, and reads should be ignored. This rules applies to ALL the modules of PNX15xx Series.

- the DRAM aperture, used to access the main memory of PNX15xx Series which contains the instruction and the data for TM3260 and data used by other PCI masters.
- the APERT1 aperture, used by TM3260 to access low speed slave devices like Flash memories or IDE disk drives that are located in the XIO aperture or any other PCI slave.

TM3260 CPU accesses the three apertures using regular load/store operations. Some internal logic in the data cache unit surveys the load/store addresses and routes the request to the appropriate internal PNX15xx Series registers (this includes the registers belonging to TM3260) if the address falls into the MMIO aperture. If the load/store address falls into the DRAM aperture the load/store request is routed to the data cache and eventually the main memory. Finally if the load/store address falls into the APERT1 aperture, the request is send to the PCI bus (if it maps to an XIO device or a PCI internal aperture, see the following Section 2.3).

Figure 2 presents the memory map seen by the TM3260 and the remaining of the PNX15xx Series system. The apertures can be placed in any order with respect to each other.

PNX15xx Series allows a host CPU to prevent TM3260 to change its own aperture registers. This can be obtained by flipping TM32_CONTROL.TM32_APERT_MODIFIABLE to ‘1’ (Section 2.4.1). The aperture locations are defined as follows:

- The MMIO aperture is starting at the address contained in the BASE_14 MMIO register. The register is located and owned by the PCI module. It is equivalent to the BASE_14 PCI Configuration space register. This is different with respect to PNX1300 Series or PNX1300 Series where an MMIO_BASE MMIO register was available.
- The DRAM aperture is starting at the address contained in the TM32_DRAM_LO MMIO register and finishes at TM32_DRAM_HI - 1.

Remark: If the value 0x0000,0000 is stored into TM32_DRAM_HI, this value is understood as 0x1,0000,0000.

- The APERT1 aperture is starting at the address contained in the TM32_APERT1_LO MMIO register and finishes at TM32_APERT1_HI - 1.

Remark: If the value 0x0000,0000 is stored into TM32_APERT1_HI, this value is understood as 0x1,0000,0000.

2.3 The DCS View Or The System View

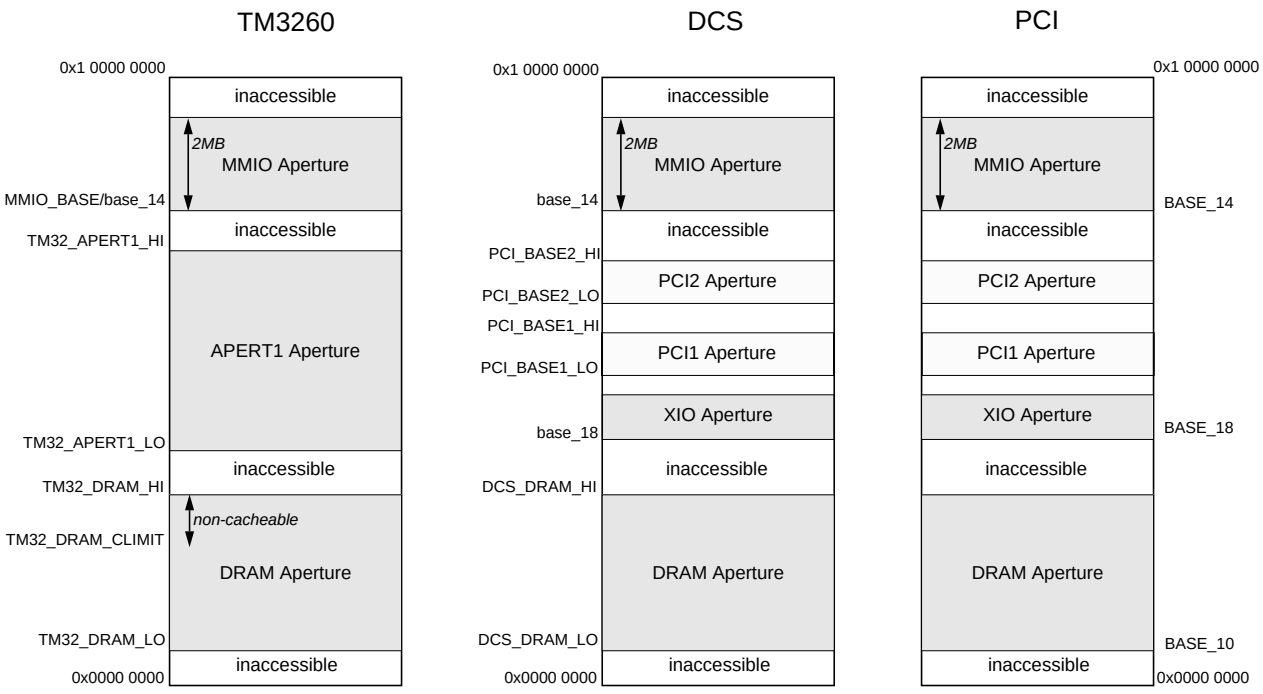


Figure 2: PNX15xx Series System Memory Map

The DCS bus can be seen as the link between the PCI side and the CPU side:

- Requests from the PCI bus or the TM3260 targeting the MMIO aperture converge to the DCS bus through the MMIO apertures and then are dispatched to the corresponding MMIO registers.
- Requests from the TM3260 to the APERT1 aperture are transferred to the DCS bus and then dispatched to the PCI module if the address of the request matches one of the three apertures, PCI2, PCI1 or XIO. These apertures are used to map loads and stores from the CPU to any slave connected to the PCI bus. The definition of the MMIO registers containing the address ranges for the two internal PCI apertures can be found in Chapter 7 PCI-XIO Module.

Remark: Requests from the TM3260 to APERT1 may fall in an non accessible address region in the DCS bus, like between the PCI1 and PCI2 apertures. It is legal to do so. The request is discarded by the DCS bus controller and a random value is returned upon reads.

Remark: TM3260 compiler uses speculative loads (i.e. the result of the load may not be used by the CPU) to improve performance. These speculative loads often contain addresses coming from the TM3260 internal register file that are not initialized properly since the return value of the load is not to be used (unless the execution of

the program is in a phase where it is planned to be used). This creates random addresses that can target the APERT1 aperture. Therefore the load may generate a transaction on the PCI bus that may have some side effects. Furthermore the performance are deteriorated by a long CPU stall cycle that is dependent on the completion of PCI bus transaction (the CPU does not continue unless the read has completed). To avoid these long CPU stall cycles it is recommended to disable the APERT1 when not used. This is achieved by setting the right mode into the TM3260 DC_LOCK_CTL MMIO register or by setting TM32_APERT1_LO and TM32_APERT1_HI to the same value.

- Requests from the PCI bus or the TM3260 targeting the DRAM aperture do not go through the DCS bus. Instead the requests are routed directly to the MMI module. The DRAM aperture defined in the DCS bus is exclusively defined for the boot module. When the boot module is programmed to boot PNX15xx Series from an EEPROM, the boot module fetches write commands from the EEPROM. Each write command is sent to the DCS bus. If the write address falls between the aperture defined by DCS_DRAM_LO and DCS_DRAM_HI, Section 2.4.1, then the write data is transferred to the MMI module. This gate allows transfer to the main memory, a binary program, (that is stored into the EEPROM) for the TM3260. The bus connecting the module to the MMI is referenced as the MTL bus (see Section 10. on page 3-30 Figure 3).

2.4 The Programmable DCS Apertures

The address range defined by the content of DCS_DRAM_LO or DCS_DRAM_HI must not overlap the address ranges of the other apertures on the DCS bus. This can happen temporarily when changing either the DCS_DRAM_LO or the DCS_DRAM_HI. Therefore any change of the DCS_DRAM_LO or DCS_DRAM_HI registers must be done by first disabling the DCS DRAM aperture. This is achieved by starting to change DCS_DRAM_LO or DCS_DRAM_HI such that DCS_DRAM_LO is greater than DCS_DRAM_HI.

Similar constraints apply respectively to PCI_BASE1_LO and PCI_BASE1_HI, and PCI_BASE2_LO and PCI_BASE2_HI.

2.4.1 DCS DRAM Aperture Control MMIO Registers

Table 1: SYSTEM Registers

Bit	Symbol	Access	Value	Description
DCS DRAM Aperture Control Registers				
Offset 0x06 3200	***DCS_DRAM_LO***			
31:16	DCS_DRAM_LO	R/W	0x0000	DCS_DRAM_LO indicates the lowest DCS bus address mapped to DRAM. Its granularity is of 64 Kilobytes. The reset value is 0. Writes to this register are controlled by the DCS_DRAM_WE bit in the APERTURE_WE MMIO register.
15:0	Unused	-	-	To ensure software backward compatibility, writes to unused or reserved bits should be zero and reads must be ignored.
Offset 0x06 3204	***DCS_DRAM_HI***			
31:16	DCS_DRAM_HI	R/W	0x0000	DCS_DRAM_HI indicates the highest DCS bus address mapped to DRAM. Its granularity is of 64 Kilobytes. The reset value of 0 disables memory accesses from the DCS bus. Writes to this register are controlled by the DCS_DRAM_WE bit in the APERTURE_WE MMIO register.
15:0	Unused	-	-	To ensure software backward compatibility, writes to unused or reserved bits should be zero and reads must be ignored.
Offset 0x06 3208	***APERTURE_WE***			
31:1	Unused	-	-	To ensure software backward compatibility, writes to unused or reserved bits should be zero and reads must be ignored.
0	DCS_DRAM_WE	R/W	0x0	• '0': Writing to DCS_DRAM_LO or DCS_DRAM_HI is disabled. • '1': Writing to DCS_DRAM_LO or DCS_DRAM_HI is enabled. • When writing to either DCS_DRAM_LO or DCS_DRAM_HI occurs, this bit is automatically cleared. • By default it is not authorized to write to the DCS_DRAM_LO and DCS_DRAM_HI registers. • The address range defined by the content of DCS_DRAM_LO or DCS_DRAM_HI must not overlap the address ranges of the other apertures on the DCS bus. This can happen temporarily when changing either the DCS_DRAM_LO or the DCS_DRAM_HI. Therefore any change of the DCS_DRAM_LO or DCS_DRAM_HI registers must be done by first disabling the DCS DRAM aperture. This is achieved by starting to change DCS_DRAM_LO or DCS_DRAM_HI such that DCS_DRAM_LO is greater than DCS_DRAM_HI.

2.5 Aperture Boundaries

The MMIO aperture is always 2 Megabytes.

The DRAM aperture size range is from 1 to 256 Megabytes. Defined at boot time, it may be changed later on by the TM3260 CPU.

The XIO aperture size range is from 1 to 128 Megabytes.

Other than the PCI module, only the TM3260 CPU can emit requests to the PCI bus, i.e. none of the other PNX15xx Series modules can do so.

Only the TM3260 CPU and external PCI master can request MMIO reads or writes.

The XIO aperture can only be accessed by the TM3260 CPU.

3. System Principles

The system resources module is like any other module composing the PNX15xx Series system. Like the other modules it has a Module ID MMIO register as well as powerdown MMIO register.

3.1 Module ID

The module ID MMIO register is used to differentiate between the different modules of the system and different revisions of the same module. For all the modules the MMIO content is composed of:

- An unique 16-bit Module ID. This ID is only changed if the functionality of the Module changes significantly. Module IDs 0 and 1 are reserved.
- An 8-bit revision ID composed of a 4-bit MAJOR_REV ID and a 4-bit MINOR_REV ID. MAJOR_REV ID is changed upon changing functionality of the module, while the MINOR_REV ID is changed in case of bug fixing or non functional fixes like yield improvements.
- An 8-bit value to code the range of recognized MMIO addresses by the module. This aperture size allows the module to claim one offset region of the MMIO Aperture. The offset region or local aperture is defined by the following formula, (N + 1) * 4 Kilobytes, where N is the 8-bit code stored in the module ID register.

This is a read only register. See Section 3.3 for details on the system module ID.

3.2 Powerdown bit

Major powerdown saving is achieved by turning off the clock that feeds the module. The safe procedure to turn off the clock of a module is to write a ‘1’ to the powerdown bit located in each module of the system before turning off its clock (whenever it is possible). Similarly when powering the module back up, the clock should be turned on before the powerdown bit is flipped back to ‘0’. When the powerdown bit is activated the module will no longer respond to MMIO read or writes other than transactions targeting the powerdown bit.

Most of the PNX15xx Series modules need two different clocks to operate. The streaming clock, e.g. the video pixel clock for QVCP, and the MMIO or DCS clock. Only the streaming clock should be turned off. Therefore, locally some modules may do extra clock gating on the DCS clock when the powerdown bit is turned on.

For the system module there is no streaming clock to turn off. Details on the MMIO register layout is available in the next Section 3.3.

3.3 System Module MMIO registers

Table 2: SYSTEM REGISTERS

Bit	Symbol	Access	Value	Description
System Module Registers				
Offset 0x06 3FF4	***GLB_REG_POWER_DOWN***			
31	POWER_DOWN	R/W	0x0000	Power down register for the module 0: Normal operation of the module. This is the reset value. 1: Module is powered down and the module clock can be removed. At power down, module responds to all reads with 0xDEADABBA (except for reads of powerdown register). Writes are answered with DCS ERR signal (except for writes to power down register).
30:0	Unused	-	-	To ensure software backward compatibility, writes to unused or reserved bits should be zero and reads must be ignored.
Offset 0x06 3FFC	***GLB_REG_MOD _ID***			
31:16	MODULE_ID	R	0x0126	Unique 16-bit code. Module ID 0 and -1 are reserved for future use. 0x0126 indicates a the system register module.
15:12	MAJOR_REV	R	0x8	Changed upon functional revision, like new feature added to previous revision
11:8	MINOR_REV	R	0x1	Changed upon bug fix or non functional changes like yield improvement.
7:0	APERTURE	R	0x0	Encoded as: Aperture size = 4K*(bit_value+1). The bit value is reset to 0 meaning a 4K aperture for the Global register 1 module according to the formula above.

4. System Endian Mode

PNX15xx Series supports both big-endian and little-endian modes, allowing it to run either little-endian or big-endian software, as required by a particular application or system.

The operating endian mode of the PNX15xx Series system is defined in one unique location and it is observed by all the modules in the system.

Section 4.1 presents the layout of the system endian mode MMIO register.

4.1 System Endian Mode MMIO registers

Table 3: SYSTEM REGISTERS

Bit	Symbol	Access	Value	Description
System Endian Mode Registers				
Offset 0x06 3014	***SYS_ENDIANMODE***			
31:1	Unused	-	-	To ensure software backward compatibility, writes to unused or reserved bits should be zero and reads must be ignored.
0	BIG_ENDIAN	R/W	0	System endian mode. ‘0’: little endian. ‘1’: big endian.

5. System Semaphores

PNX15xx Series has 16 simple Multi-Processor (MP) semaphore-assist devices. They are built out of 32-bit registers, accessible through MMIO by either the local TM3260 CPU or by any other CPU located on the PCI bus through the aperture made available on the PCI module.

The semaphores operation is as follows: each master in the system constructs a personal nonzero 12-bit ID (Section 5.2). To obtain a semaphore, a master is required to do the following actions:

- write the unique ID to one of the 16 semaphores using a 32-bit store. This uses a 32-bit write with the ID in the 12 LSBs
- read back the ID. This uses a 32-bit load that returns 0x00000nnn. Then

```
if (0x00000nnn == ID) {
   "perform the short critical section action for which the semaphore was
requested";
    "then write 0x00000000 back to the selected semaphore to release it for the
other tasks"
} else {"try again later, or loop back to write"}
```

5.1 Semaphore Specification

Each of the 16 semaphores behavior is defined by the following pseudo-code:

```
if (cur_content == 0) {
   new_content = write_value;
} else {if (write_value == 0) new_content = 0;}
/* ELSE NO ACTION! */
```

Layout and offset address of the 16 semaphores is available in Section 5.5.

5.2 Construction of a 12-bit ID

A system based on PNX15xx Series can construct a personal, non-zero 12-bit ID in a variety of ways:

- PCI configspace PERSONALITY entry. Each PNX15xx Series receives a 16-bit PERSONALITY value from the EEPROM during boot. This PERSONALITY register is located at offset 0x40 in configuration space. In a MP system, some of the bits of PERSONALITY can be individualized for each CPU involved, giving it a unique 2-, 3- or 4-bit ID, as needed given the maximum number of CPUs in the design.
- In the case of a host-assisted PNX15xx Series boot, the PCI BIOS assigns a unique MMIO_BASE and DRAM_BASE to every PNX15xx Series. In particular, the 11 MSBs of each MMIO_BASE are unique, since each MMIO aperture is 2 Megabytes in size. These bits can be used as a personality ID. Set bit 11 (MSB) to '1' to guarantee a non-zero ID value.

5.3 The Master Semaphore

Each PNX15xx Series in the system adds a block of 16 semaphores to the mix. The intended use is to treat one of these block of 16 semaphores as THE master semaphore block in the system. To determine which semaphore block is master each TM3260 can use PCI configuration space accesses to determine which other PNX15xx Seriess are present in the board system. Then, the PNX15xx Series with the lowest PERSONALITY number, or the lowest MMIO_BASE is chosen as the PNX15xx Series containing the master semaphores.

5.4 Usage Notes

To avoid contention between the different tasks trying to access the different critical resources of the system or the application, PNX15xx Series offers 16 different semaphore devices. This allows to use them not only for inter-processor semaphores but also for processes running on a single PNX15xx Series. However these process synchronizations within the same processor can use regular memory to memory transactions to implement primitive synchronization.

As described here, obtaining a semaphore does not guarantee starvation-free access to critical resources. Claiming of one of the semaphores is purely stochastic. This works fine as long as a particular semaphore is not overloaded. Despite a large amount of available semaphores, utmost care should be taken in semaphore access frequency and duration of the basic critical sections to keep the load conditions reasonable.

5.5 Semaphore MMIO Registers

Table 4: Semaphore MMIO Registers

Bits	Symbol	Access	Value	Description
Semaphore Registers				
Offset 0x06 3800	***SEMAPHORE0***			
31:12	Unused	-	-	To ensure software backward compatibility, writes to unused or reserved bits should be zero and reads must be ignored.
11:0	SEMAPHORE0	R/W	0	Read action does not change this field. Writing to this field is accepted only when • its current content is zero, upon which the semaphore is locked. • the data to be written is zero, upon which the semaphore is unlocked.
Offset 0x06 3804	***SEMAPHORE1***			
31:0	SEMAPHORE1	R/W	0	Same as semaphore0 register.
Offset 0x06 3808	***SEMAPHORE2***			
31:0	SEMAPHORE2	R/W	0	Same as semaphore0 register.
Offset 0x06 380C	***SEMAPHORE3***			
31:0	SEMAPHORE3	R/W	0	Same as semaphore0 register.
Offset 0x06 3810	***SEMAPHORE4***			
31:0	SEMAPHORE4	R/W	0	Same as semaphore0 register.
Offset 0x06 3814	***SEMAPHORE5***			
31:0	SEMAPHORE5	R/W	0	Same as semaphore0 register.
Offset 0x06 3818	***SEMAPHORE6***			
31:0	SEMAPHORE6	R/W	0	Same as semaphore0 register.
Offset 0x06 381C	***SEMAPHORE7***			
31:0	SEMAPHORE7	R/W	0	Same as semaphore0 register.
Offset 0x06 3820	***SEMAPHORE8***			
31:0	SEMAPHORE8	R/W	0	Same as semaphore0 register.
Offset 0x06 3824	***SEMAPHORE9***			
31:0	SEMAPHORE9	R/W	0	Same as semaphore0 register.
Offset 0x06 3828	***SEMAPHORE10***			
31:0	SEMAPHORE10	R/W	0	Same as semaphore0 register.
Offset 0x06 382C	***SEMAPHORE11***			
31:0	SEMAPHORE11	R/W	0	Same as semaphore0 register.
Offset 0x06 3830	***SEMAPHORE12***			
31:0	SEMAPHORE12	R/W	0	Same as semaphore0 register.
Offset 0x06 3834	***SEMAPHORE13***			
31:0	SEMAPHORE13	R/W	0	Same as semaphore0 register.
Offset 0x06 3838	***SEMAPHORE14***			

Table 4: Semaphore MMIO Registers …Continued

Bits	Symbol	Access	Value	Description
31:0	SEMAPHORE14	R/W	0	Same as semaphore0 register.
Offset 0x06 383C	*SEMAPHORE15*			
31:0	SEMAPHORE15	R/W	0	Same as semaphore0 register.

6. System Related Information for TM3260

This section contains information on how the internal TM3260 resources like its interrupt lines or timers have been assigned or used in the PNX15xx Series system. More specific details on how to program or on the exact behavior of these resources is found in [1].

6.1 Interrupts

A fundamental aspect of PNX15xx Series system is to provide hardware modules (or hardware accelerators) that relieve the TM3260 CPU for other video/audio processing. These modules are mainly internal bus DMA masters. Thus once programmed by the TM3260 they only require limited CPU processing power. For example the video module only requires the TM3260 to update the pointers to the next frame 60 times per seconds. An interrupt line is used to signal TM3260 of that need.

The TM3260 Vectored Interrupt Controller (VIC) provides 64 inputs for interrupt request lines. The interrupt controller prioritizes and maps the multiple requests from the several PNX15xx Series modules onto successive interrupt requests to the TM3260 execution unit.

Table 5 shows the assignment of modules to interrupt source numbers, as well as the recommended operating mode (edge or level triggered). Note that there are a total of 7 possible external pins to assert interrupt requests. Only PCI_INTA_N is a dedicated pin for external interrupts. The other pins may be used for other functionality. The first 5 interrupt sources, i.e. source 0 through 4, are asserted by active low signal conventions, i.e. a zero level or a negative edge asserts a request. The remaining two external interrupt lines, i.e. source 26 and 27, like all the other regular interrupt lines, operate with active high signalling conventions.

Table 5: Interrupt Source Assignments

SOURCE NAME	SOURCE NUMBER	INTERRUPT OPERATING MODE	SOURCE DESCRIPTION
PCI_INTA_N	0	level	External PCI $\overline{\text{INTA}}$ interrupt used by the host CPU. Active LOW
PCI_GNT_A_N	1	level	Direct external interrupt input line, active LOW
PCI_GNT_B_N	2	level	Direct external interrupt input line, active LOW
PCI_REQ_A_N	3	level	Direct external interrupt input line, active LOW
PCI_REQ_B_N	4	level	Direct external interrupt input line, active LOW
TIMER1	5	edge	General purpose internal TM3260 timer.
TIMER2	6	edge	General purpose internal TM3260 timer.

Table 5: Interrupt Source Assignments

SOURCE NAME	SOURCE NUMBER	INTERRUPT OPERATING MODE	SOURCE DESCRIPTION
TIMER3	7	edge	General purpose internal TM3260 timer.
SYSTIMER	8	edge	General purpose internal TM3260 timer.
VIP	9	level	Video Input Processor
QVCP	10	level	Quality Video Composition Processor
AI	11	level	Audio Input
AO	12	level	Audio Output
SPDI	13	level	S/PDIF Input
SPDO	14	level	S/PDIF Output
ETHERNET	15	level	Ethernet MAC 10/100
I2C	16	level	I^2C interface
TMDBG	17	level	JTAG interface
FGPI	18	level	Fast Generic Parallel Input interface
FGPO	19	level	Fast Generic Parallel Output interface
Reserved	20...21	n/a	Reserved for future devices
MBS	22	level	Memory Base Scaler
DE	23	level	2D Drawing Engine
VLD	24	level	Variable Length Decoder
DVD-CSS	25	level	DVD Descrambler
GPIO[10]	26	level	Direct external interrupt input line, active HIGH
GPIO[11]	27	level	Direct external interrupt input line, active HIGH
HOSTCOM	28	edge	(software) Host Communication
APPLICATION	29	edge	(software) Application
DEBUGGER	30	edge	(software) Debugger
RTOS	31	edge	(software) Real Time Operating System
GPIO_INT0	32	level	General Purpose I/O interrupt line 0, FIFO 0
GPIO_INT1	33	level	General Purpose I/O interrupt line 1, FIFO 1
GPIO_INT2	34	level	General Purpose I/O interrupt line 2, FIFO 2
GPIO_INT3	35	level	General Purpose I/O interrupt line 3, FIFO 3
GPIO_INT4	36	level	General Purpose I/O interrupt line 4, TSU Units
PCI	37	level	Peripheral Component Interconnect error monitoring
PCI_GPPM	38	level	PCI single data phase transfer completed
PCI_GPXIO	39	level	PCI XIO transaction completed
PCI_DMA	40	level	PCI DMA transaction completed
CLOCK	41	level	Clock generation
WATCHDOG	42	level	On-chip Watchdog timer
Reserved	43...59	n/a	Reserved for future devices

12NC 9397 750 14321

Table 5: Interrupt Source Assignments

SOURCE NAME	SOURCE NUMBER	INTERRUPT OPERATING MODE	SOURCE DESCRIPTION
DCS	60	level	Internal DCS bus
MMI	61	level	Main Memory Interface, i.e. the DRAM controller
Reserved	62...63	n/a	Reserved for future devices

6.2 Timers

The TM3260 CPU contains four programmable timer/counters, all with the same function. The first three (TIMER1, TIMER2, TIMER3) are intended for general use. The fourth timer/counter (SYSTIMER) is reserved for use by the system software and should not be used by applications.

Each timer/counter can be set to count one of the event types specified in Table 6. Note that source 3 to 6 are special TM3260 events used for program debug support as well as cache performance monitoring. Full description can be found in [1]. For all the other source signals, like the VDO_CLK1 pin, positive-going edges on the signal are counted. Each timer increments its value until the programmed count is reached. On the clock cycle when the timer reaches its programmed count value, an interrupt is generated.

The timer interrupt source mode should be set as edge-sensitive as presented in Table 5. No software interrupt acknowledge to the timer device is necessary.

Table 6: TM3260 Timer Source Selection

SOURCE NAME	SOURCE NUMBER	SOURCE DESCRIPTION
TM3260 CLOCK	0	The CPU clock
PRESCALE	1	Pre-scaled CPU clock
Reserved	2	Reserved for future devices
DATABREAK	3	Data breakpoints
INSTBREAK	4	Instruction breakpoints
CACHE1	5	Cache event 1
CACHE2	6	Cache event 2
VDI_CLK1	7	VIP clock pin
VDI_CLK2	8	FGPI clock pin
VDO_CLK1	9	QVCP clock pin
VDO_CLK2	10	FGPO clock pin
AI_WS	11	AI Word Strobe pin
AO_WS	12	AO Word Strobe pin
GPIO_TIMER0	13	GPIO pin selection 0
GPIO_TIMER1	14	GPIO pin selection 1
REFERENCE_CLOCK	15	The 27 MHz input crystal clock

6.3 System Parameters for TM3260

Few more control parameters are available to tune the use of TM3260 and PNX15xx Series. The MMIO register layout and offsets are described in Section 6.3.1.

- The CPU apertures (DRAM and APERT1 described in Section 2.2) can be modified by the TM3260 itself, if the TM32_APERT_MODIFIABLE bit is set to ‘1’. In host mode the host CPU can decide to prevent TM3260 to go out of its allowed apertures by flipping to ‘0’ the bit TM32_APERT_MODIFIABLE.
- The TM32_LS_DBLLINE and TM32_IFU_DBLLINE parameters influence the overall performance of the TM3260. These parameters are related to the cache line sizes and the optimal memory burst than can be obtained with PNX15xx Series MMI. The default values favor the main memory bandwidth usage and improve, in most cases, the TM3260 processing power. However some applications may require a shorter memory burst to reduce the bandwidth usage or to avoid some pathological cache trashing cases. TM32_LS_DBLLINE and TM32_IFU_DBLLINE can then be flipped to ‘0’. There is no available formula to know if a particular application benefits from one setting or the other. Experimentation on the final application is recommended to determine the optimal settings.
- It is possible for a host CPU to shutdown entirely the high speed clock of the TM3260. The safe procedure consists in first requesting the TM3260 to prepare itself for major powerdown mode. The host CPU needs first to alert the software running on the TM3260 that a powerdown sequence is coming. The TM3260 software acknowledges that it is ready. Then the host CPU toggles the TM32_PWRDWN_REQ bit to inform the TM3260 module that a full powerdown mode is requested. The TM3260 hardware state machine replies by asserting the TM32_PWRDWN_ACK bit. From this point TM3260 will not answer to any request and its high speed CPU clock can be turned off by the CPU host. The wake-up sequence starts by turning back on the high speed CPU clock and then flip to ‘0’ the TM32_PWRDWN_REQ bit.

Remark: It is not recommended to have the TM3260 to flip itself to ‘1’ the TM32_PWRDWN_REQ bit.

6.3.1 TM3260 System Parameters MMIO Registers

Table 7: TM3260 System Parameters MMIO Registers

Bit	Symbol	Access	Value	Description
System Module Registers				
Offset 0x06 3700	***TM32_CONTROL***			
31:4	Unused	-	-	To ensure software backward compatibility, writes to unused or reserved bits should be zero and reads must be ignored.
3	TM32_APERT_MODIFIABLE	R/W	0x1	TM3260 Aperture Modifiable. This bit is usually written once at boot time. The value of this bit can only be altered once. • 0: Disables writes by the TM3260 to the MMIO registers TM32_DRAM_HI, TM32_DRAM_LO, TM32_APERT_HI and TM32_APERT_LO. • 1: Enables writes by the TM3260 to the MMIO registers TM32_DRAM_HI, TM32_DRAM_LO, TM32_APERT_HI and TM32_APERT_LO.
2	TM32_LS_DBLLINE	R/W	0x1	TM3260 Load/Store Unit (i.e. Data Cache) Double Line Fill enable • 0: Do not enable Double Line fills for the Load/Store Unit • 1: Enable Double Line fills for the Load/Store Unit
1	TM32_IFU_DBLLINE	R/W	0x1	TM3260 Instruction Fetch Unit (i.e. Instruction Cache) Double Line Fill enable • 0: Do not enable Double Line fills for the Instruction Fetch Unit • 1: Enable Double Line fills for the Instruction Fetch Unit
0	TM32_PWRDWN_REQ	R/W	0x0	TM3260 full powerdown request Upon writes: • 1->0: Request a TM3260 Power Up • 0->1: Request a TM3260 Power Down Upon reads • Undefined
Offset 0x06 3704	***TM32_STATUS***			
31:1	Unused	-	-	To ensure software backward compatibility, writes to unused or reserved bits should be zero and reads must be ignored.
0	TM32_PWRDWN_ACK	R	0x0	0: TM3260 is in full power mode. 1: TM3260 is in full powerdown mode.

7. Video Input and Output Routers

PNX15xx Series provides two groups of high speed pins to stream data or video in and out. The input group of pins is prefixed by VDI, Video Data Input. The output group is prefixed by VDO, Video Data Output. Each group is shared between two modules. On the input side, VIP and FGPI get their pin allocation through the input router. On the output side QVCP and FGPO get their pin assignment through the output router. The input router is controlled by VDI_MODE. The output router is controlled by the VDO_MODE.

Section 7.1 details the VDI and VDO pin assignment based on the content of the VDI_MODE and VDO_MODE MMIO registers. Section 9.1 and Section 9.2 on page 2-19 give an overview of the different modes.

7.1 MMIO Registers for the Input/Output Video/Data Router

In the following tables

- The **X** associated with a bit value means ‘do not care’.
- (clk_vip FF) means the data is registered by the clock assigned to VIP before presenting the signals to the VIP module.
- (clk_fgpi FF) means the data is registered by the clock assigned to FGPI before presenting the signals to the FGPI module.

Table 8: Global Registers

Bit	Symbol	Access	Value	Description
Input and Output Control Registers				
Offset 0x06 3000 VDI_MODE				
31:8	Unused	-	-	To ensure software backward compatibility, writes to unused or reserved bits should be zero and reads must be ignored.
7	VDI_MODE_7	R/W	0	This bit should be set to '1' only when FGPI is set to work in 8-bit mode. This bit controls dedicated hardware located in the input router that allows to use the FGPI module as a second module to capture a 656 video source. However in this mode there is no on-the-fly video image processing possible and the YUV data is linearly stored in memory (VIP uses YUV planes). The dedicated hardware allows to generate fgpi_start and fgpi_stop signals that directs FGPI to store each field of the in-coming 656 video stream into a separate buffer. The description bellow explains the behavior of the state machine for that dedicated pattern matching hardware. 0: Disable pattern matching state machine for FGPI start/stop signals. 1: Enable pattern matching state machine for FGPI start/stop signals. When first enabled, the pattern matching state machine is in its "INIT" state and begins comparing fgpi_data[7:0] for the pattern 0xFF, 0x00, 0x00, and 0xEC on each fgpi clock. Once this pattern is detected, it enters the "MAIN" state. Below are listed the patterns for fgpi_start and fgpi_stop signal assertion/de-assertion when in the MAIN state. The fgpi_start signal asserts for one fgpi clock when the fourth byte of the pattern is matched. The fgpi_start signal de-asserts on the next fgpi clock and remains de-asserted until one of the patterns is detected. The fgpi_stop asserts when the assertion pattern is detected and remains asserted until the de-assertion pattern is detected. The pattern matching state machine returns to the "INIT" state when VDI_MODE[7] = 0 or the FGPI block is reset with a Hardware or Software reset. fgpi_start = 1 when fgpi_data[7:0] =0xFF, 0x00, 0x00, 0x9D or 0xFF, 0x00, 0x00, 0xDA or 0xFF, 0x00, 0x00, 0xF1 or 0xFF, 0x00, 0x00, 0xB6 else fgpi_start = 0. fgpi_stop = 1 when fgpi_data[7:0] =0xFF, 0x00, 0x00, 0xF1 fgpi_stop = 0 when fgpi_data[7:0] =0xFF, 0x00, 0x00, 0xB6
6:5	Unused	-	-	To ensure software backward compatibility, writes to unused or reserved bits should be zero and reads must be ignored.

Table 8: Global Registers *…Continued*

Bit	Symbol	Access	Value	Description
4:3 2 1:0	VDI_MODE[4:3] VDI_MODE[2] is unused VDI_MODE[1:0]	R/W - R/W	0 - 0	VDI-to-VIP mapping **XX000**: 8- or 10-bit ITU 656, 8- or 10-bit raw data VDI_V1-> (clk_vip FF)-> vip_dv_valid VDI_D[29:20] -> (clk_vip FF)-> vip_dv_data[9:0] “0”-> vip_dv_d_data[9:0] Reserved-> (clk_vip FF)-> vip_vrefhd Reserved-> (clk_vip FF)-> vip_hrefhd “0”-> vip_frefhd In 8-bit ITU 656 mode the YUV[7:0] maps to vip_dv_data[9:2], therefore it maps to VDI_D[29:22]. Similarly in 8-bit raw data mode VDI_D[29:22] contains the 8-bit data. Note: H/V sync can only be used when VIP is operated in 8-bit VMI mode. In that mode the H/V syncs must be connected to VDI_D[20] and VDI_D[21] respectively. **XX001**: 20-bit ITU 656 like for HD VDI_V1-> (clk_vip FF)-> vip_dv_valid VDI_D[19:10] -> (clk_vip FF)-> vip_dv_data[9:0] VDI_D[29:20] -> (clk_vip FF)-> vip_dv_d_data[9:0] VDI_D[30]-> (clk_vip FF)-> vip_vrefhd VDI_D[31]-> (clk_vip FF)-> vip_hrefhd VDI_D[9]-> (clk_vip FF)-> vip_frefhd HD can be 10- or 8-bit YUV data. In 8-bit mode VDI_D[19:12] contains the UV data. VDI_D[29:22] is expecting the 8-bit Y data. In 10-bit mode VDI_D[19:10] contains the UV bus. VDI_D[29:20] is expecting the 10-bit Y data. **XX010**: 8-bit ITU 656 or 8-bit raw data VDI_V1-> (clk_vip FF)-> vip_dv_valid VDI_D[31:24] -> (clk_vip FF)-> vip_dv_data[9:2] “0”-> vip_dv_data[1:0] “0”-> vip_dv_d_data[9:0] “0”-> vip_vrefhd “0”-> vip_hrefhd “0”-> vip_frefhd **XX011**: N/A VDI_V1-> (clk_vip FF)-> vip_dv_valid “0”-> vip_dv_data[9:0] “0”-> vip_dv_d_data[9:0] “0”-> vip_vrefhd “0”-> vip_hrefhd “0”-> vip_frefhd

Table 8: Global Registers *...Continued*

Bit	Symbol	Access	Value	Description
4:3 2 1:0	VDI_MODE[4:3] VDI_MODE[2] is unused VDI_MODE[1:0]	R/W - R/W	0 - 0	VDI-to-FGPI mapping up to 22-bit data capture **XX000**: VDI_V2 -> (clk_fgpi FF) -> fgpi_d_valid VDI_D[15:0] -> (clk_fgpi FF) -> fgpi_data[15:0] VDI_D[32] -> (clk_fgpi FF) -> fgpi_start (*) VDI_D[33] -> (clk_fgpi FF) -> fgpi_stop (*) **00000**: “0”-> fgpi_data[31:20] VDI_D[19:16] -> (clk_fgpi FF) -> fpgi_data[19:16] **01000**: “1”-> fgpi_data[31:20] VDI_D[19:16] -> (clk_fgpi FF) -> fpgi_data[19:16] **10000**: VDI_D[19] -> (clk_fgpi FF)-> fgpi_data[31:20] VDI_D[19:16] -> (clk_fgpi FF)-> fpgi_data[19:16] **11000**: VDI_D[15] -> (clk_fgpi FF) -> fpgi_data[31:16] (*) For VDI_MODE[7] = 0. When VDI_MODE[7] = 1, fgpi_start and fgpi_stop are controlled by a simple pattern matching state machine.
4:3 2 1:0	VDI_MODE[4:3] VDI_MODE[2] is unused VDI_MODE[1:0]	R/W - R/W	0 - 0	VDI-to-FGPI mapping (continued) up to 12-bit data capture **XX001**: VDI_V2 -> (clk_fgpi FF) -> fgpi_d_valid VDI_D[7:0] -> (clk_fgpi FF) -> fgpi_data[7:0] VDI_D[32] -> (clk_fgpi FF) -> fgpi_start (*) VDI_D[33] -> (clk_fgpi FF) -> fgpi_stop (*) **00001**: “0”-> fgpi_data[31:9] VDI_D[8] -> (clk_fgpi FF) -> fpgi_data[8] **01001**: “1”-> fgpi_data[31:9] VDI_D[8] -> (clk_fgpi FF) -> fpgi_data[8] **10001**: VDI_D[8] -> (clk_fgpi FF) -> fgpi_data[31:9] VDI_D[8] -> (clk_fgpi FF) -> fpgi_data[8] **11001**: VDI_D[7] -> (clk_fgpi FF) -> fpgi_data[31:8] (*) For VDI_MODE[7] = 0. When VDI_MODE[7] = 1, fgpi_start and fgpi_stop are controlled by a simple pattern matching state machine.

Table 8: Global Registers *...Continued*

Bit	Symbol	Access	Value	Description
				VDI-to-FGPI mapping (continued) up to 24-bit data capture **XX010**: VDI_V2 -> (clk_fgpi FF) -> fgpi_d_valid VDI_D[23:0] -> (clk_fgpi FF) -> fgpi_data[23:0] VDI_D[32] -> (clk_fgpi FF) -> fgpi_start (*) VDI_D[33] -> (clk_fgpi FF) -> fgpi_stop (*) **00010**: “0” -> fgpi_data[31:24] **01010**: “1” -> fgpi_data[31:24] **10010**: VDI_D[23] -> (clk_fgpi FF) -> fgpi_data[31:24] 11010: “0” -> fgpi_data[31:24] (*) For VDI_MODE[7] = 0. When VDI_MODE[7] = 1, fgpi_start and fgpi_stop are controlled by a simple pattern matching state machine.
				VDI-to-FGPI mapping (continued) up to 32-bit data capture XX011: VDI_V2 -> (clk_fgpi FF) -> fgpi_d_valid VDI_D[31:0] -> (clk_fgpi FF) -> fgpi_data[31:0] VDI_D[32] -> (clk_fgpi FF) -> fgpi_start (*) VDI_D[33] -> (clk_fgpi FF) -> fgpi_stop (*) (*) For VDI_MODE[7] = 0. When VDI_MODE[7] = 1, fgpi_start and fgpi_stop are controlled by a simple pattern matching state machine.
Offset 0x06 3004	***VDO_MODE***			
31:8	Unused	-	-	To ensure software backward compatibility, writes to unused or reserved bits should be zero and reads must be ignored.
7	VDO_MODE	R/W	0	If set to ‘1’ and VDO_MODE[2:0] set to 0, then in addition to the QVCP to the TFT interface mapping the FGPO 8-bit LSBs map as follows: VDO_D34 -> (clk_fgpo FF) -> fgpo_data[7] FGPO_BUF_SYNC -> (clk_fgpo FF) -> fgpo_data[6] FGPO_REC_SYNC -> (clk_fgpo FF) -> fgpo_data[5] VDO_D33 -> (clk_fgpo FF) -> fgpo_data[4] VDO_D32 -> (clk_fgpo FF) -> fgpo_data[3] VDO_D[2:0] -> (clk_fgpo FF) -> fgpo_data[2:0] This mode allows to have, for example, a ITU-656 video stream coming out of FGPO while the QVCP drives a 24-bit TFT LCD panel.

Table 8: Global Registers *...Continued*

Bit	Symbol	Access	Value	Description
6	VDO_MODE	R/W	0	'0': No action '1': When VDO_MODE[2:0] = 100, i.e. digital 24-bit YUV or RGB video: QVCP_DATA[15:12,9:2] -> VDO_D[16:5] when VDO_CLK1=1 QVCP_DATA[29:22,19:16] -> VDO_D[16:5] when VDO_CLK1=0 i.e. G[3:0], B[7:0] -> VDO_D[16:5] when VDO_CLK1=1 i.e. R[7:0], G[7:4] -> VDO_D[16:5] when VDO_CLK1=0 i.e. U[3:0], V[7:0] -> VDO_D[16:5] when VDO_CLK1=1 i.e. Y[7:0], U[7:4] -> VDO_D[16:5] when VDO_CLK1=0 All the other VDO pins are mapped as described below for VDO_MODE[2:0] = 100. This mode is typically used to interface with Video Encoders like the Philips SAA7104 that require the video data to be presented on both edges of the pixel clock. This mode allows to transfer the 24-bit data over a 12-bit interface, VDO_D[16:5]. **Note:** The YUV mode does not match the SAA7104 expected inputs. Use the RGB mode instead. **Note:** This mode requires a 50/50 duty cycle clock. This can be achieved by programming the QVCP PLL at twice the speed and divide it by 2 by setting the P divider to 1, or use a times 4 or 8 as described in Section PLL Settings page 5-9.
5	VDO_MODE	R/W	0	'0': No action '1': When VDO_MODE[2:0] = 010, i.e. digital 16-bit YUV video: QVCP_DATA[19:12] -> VDO_D[20:13] when VDO_CLK1=1 QVCP_DATA[9:2] -> VDO_D[20:13] when VDO_CLK1=0 i.e. UV[7:0] -> VDO_D[20:13] when VDO_CLK1=1 i.e. Y[7:0] -> VDO_D[20:13] when VDO_CLK1=0 All the other VDO pins are mapped as described below for VDO_MODE[2:0] = 010. This mode is typically used to interface with Video Encoders like the Philips SAA7104 that require the video data to be presented on both edges of the pixel clock. This mode allows to transfer the 16-bit data over an 8-bit interface, VDO_D[20:13]. **Note:** This mode requires a 50/50 duty cycle clock. This can be achieved by programming the QVCP PLL at twice the speed and divide it by 2 by setting the P divider to 1, or use a times 4 or 8 as described in Section PLL Settings page 5-9.
4:3	Unused	-	-	To ensure software backward compatibility, writes to unused or reserved bits should be zero and reads must be ignored.

Table 8: Global Registers *...Continued*

Bit	Symbol	Access	Value	Description
2:0	VDO_MODE	R/W	0	TFT/QVCP mapping to VDO interface **000*: TFT LCD controller with 24- or 18-bit digital RGB/YUV video** TFT_DATA[23:0] -> VDO_D[28:5] TFT_VSYNC -> VDO_D[29] TFT_HSYNC -> VDO_D[30] TFT_DE -> VDO_D[31] TFT_VDDON -> VDO_D[4] TFT_BKLTON -> VDO_D[3] TFT_CLK -> VDO_CLK1 In 18-bit mode VDO_D[28:23] -> R[5:0] or Y[5:0] VDO_D[20:15] -> G[5:0] or U[5:0] VDO_D[12:7] -> B[5:0] or V[5:0] In 24-bit mode VDO_D[28:21] -> R[7:0] or Y[5:0] VDO_D[20:13] -> G[7:0] or U[5:0] VDO_D[12:5] -> B[7:0] or V[5:0]
				001*: Digital ITU 656 YUV 8-/10-bit QVCP_DATA[9:0] -> VDO_D[28:19] QVCP_VSYNC -> VDO_D[29] QVCP_HSYNC -> VDO_D[30] QVCP_AUX1 -> VDO_D[31] QVCP_CLK -> VDO_CLK1 In 8-bit mode YUV[7:0] is mapped to VDO_D[28:21]. QVCP_AUX1 can be programmed to output, a CBLANK signal, a Field indicator or a video/graphics detector.
				010*: Digital 16-bit YUV video QVCP_DATA[19:12,9:2] -> VDO_D[28:13] QVCP_VSYNC -> VDO_D[29] QVCP_HSYNC -> VDO_D[30] QVCP_AUX1 -> VDO_D[31] QVCP_CLK -> VDO_CLK1 Y[7:0] is mapped to VDO_D[20:13]. UV[7:0] is mapped to VDO_D[28:21]. QVCP_AUX1 can be programmed to output, a CBLANK signal, a Field indicator or a video/graphics detector.
				011*: Digital 20-bit YUV video QVCP_DATA[19:10,9:0] -> VDO_D[28:9] QVCP_VSYNC -> VDO_D[29] QVCP_HSYNC -> VDO_D[30] QVCP_AUX1 -> VDO_D[31] QVCP_CLK -> VDO_CLK1 Y[9:0] is mapped to VDO_D[18:9]. UV[9:0] is mapped to VDO_D[28:19]. QVCP_AUX1 can be programmed to output, a CBLANK signal, a Field indicator or a video/graphics detector.

Table 8: Global Registers *…Continued*

Bit	Symbol	Access	Value	Description
				100*: Digital 24-bit YUV or RGB video QVCP_DATA[29:22,19:12,9:2] -> VDO_D[28:5] QVCP_VSYNC -> VDO_D[29] QVCP_HSYNC -> VDO_D[30] QVCP_AUX1 -> VDO_D[31] QVCP_CLK -> VDO_CLK1 In 24-bit mode VDO_D[28:21] -> R[7:0] or Y[7:0] VDO_D[20:13] -> G[7:0] or U[7:0] VDO_D[12:5] -> B[7:0] or V[7:0] In 18-bit mode VDO_D[28:23] -> R[5:0] or Y[5:0] VDO_D[20:15] -> G[5:0] or U[5:0] VDO_D[12:7] -> B[5:0] or V[5:0] QVCP_AUX1 can be programmed to output, a CBLANK signal, a Field indicator or a video/graphics detector.
				101*: Digital 30-bit YUV or RGB video QVCP_DATA[29:0] -> VDO_D[32,28:0] QVCP_VSYNC -> VDO_D[29] QVCP_HSYNC -> VDO_D[30] QVCP_AUX1 -> VDO_D[31] QVCP_CLK -> VDO_CLK1 In 30-bit mode VDO_D[32,28:20] -> R[9:0] or Y[9:0] VDO_D[19:10] -> G[9:0] or U[9:0] VDO_D[9:0] -> B[9:0] or V[9:0] In 24-bit mode VDO_D[32,28:22] -> R[7:0] or Y[7:0] VDO_D[19:12] -> G[7:0] or U[7:0] VDO_D[9:2] -> B[7:0] or V[7:0] In 18-bit mode VDO_D[32,28:24] -> R[5:0] or Y[5:0] VDO_D[19:14] -> G[5:0] or U[5:0] VDO_D[9:4] -> B[5:0] or V[5:0] QVCP_AUX1 can be programmed to output, a CBLANK signal, a Field indicator or a video/graphics detector.
				110*: Digital ITU 656 YUV 8-bit QVCP_DATA[9:2] -> VDO_D[31:24] QVCP_CLK -> VDO_CLK1
				111*: No TFT/QVCP-to-VDO mapping.

Table 8: Global Registers *...Continued*

Bit	Symbol	Access	Value	Description
2:0	VDO_MODE	R/W	0	FGPO mapping to VDO interface **000*** and VDO_MODE[7] = '1': FGPO_DATA[2:0] -> VDO_D[2:0] FGPO_DATA[3] -> VDO_D[32] FGPO_DATA[4] -> VDO_D[33] FGPO_DATA[5] -> FGPO_REC_SYNC FGPO_DATA[6] -> FGPO_BUF_SYNC FGPO_DATA[7] -> VDO_D[34] FGPO_CLK -> VDO_CLK2 **000*** and VDO_MODE[7] = '0': FGPO_DATA[2:0] -> VDO_D[2:0] FGPO_START/REC_START -> VDO_D[32] FGPO_STOP/BUF_START -> VDO_D[33] FGPO_CLK -> VDO_CLK2
				001*: FGPO_DATA[18:0] -> VDO_D[18:0] FGPO_START/REC_START -> VDO_D[32] FGPO_STOP/BUF_START -> VDO_D[33] FGPO_CLK -> VDO_CLK2
				010*: FGPO_DATA[12:0] -> VDO_D[12:0] FGPO_START/REC_START -> VDO_D[32] FGPO_STOP/BUF_START -> VDO_D[33] FGPO_CLK -> VDO_CLK2
				011*: FGPO_DATA[8:0] -> VDO_D[8:0] FGPO_START/REC_START -> VDO_D[32] FGPO_STOP/BUF_START -> VDO_D[33] FGPO_CLK -> VDO_CLK2
				100*: FGPO_DATA[4:0] -> VDO_D[4:0] FGPO_START/REC_START -> VDO_D[32] FGPO_STOP/BUF_START -> VDO_D[33] FGPO_CLK -> VDO_CLK2
				101*: No FGPO-to-VDO mapping.
				110*: FGPO_DATA[23:0] -> VDO_D[23:0] FGPO_START/REC_START -> VDO_D[32] FGPO_STOP/BUF_START -> VDO_D[33] FGPO_CLK -> VDO_CLK2
				111*: FGPO_DATA[31:0] -> VDO_D[31:0] FGPO_START/REC_START -> VDO_D[32] FGPO_STOP/BUF_START -> VDO_D[33] FGPO_CLK -> VDO_CLK2

[8-1] Note: *When the LCD IF is enabled, VDO_MODE[2:0] is forced to "000".

8. Miscellaneous

Several other system MMIO registers are described in the following paragraphs and detailed in the next Section 8.1:

- By default PCI_INTA_N is an input/output pin used in open drain mode for the PCI bus. When a host CPU wants to assert an interrupt to the TM3260 it asserts the PCI_INTA_N low. Similarly if TM3260 wants to notify a host CPU of an interrupt it can assert low the PCI_INTA_N pin by programming the PCI_INTA MMIO register.
- The 8 SCRATCH MMIO registers are mainly used for debug purpose. Since they are not reset by the external POR_IN_N or RESET_IN_N signals they can be used for post-mortem system crash to retain some critical or debug values.
- Event timestamping for the SPDI interface comes with a diversity of requirements. To keep PNX15xx Series as a programmable system, a system multiplexer is implemented to select which event or signal to timestamp. The multiplexer is controlled by the SPDI_MUX_SEL MMIO register. The different selectable signals coming from the SPDI module are displayed in Section 8.1.
- The SPARE_CTRL MMIO register is reserved for future usage.

8.1 Miscellaneous System MMIO registers

Table 9: Miscellaneous System MMIO registers

Bit	Symbol	Access	Value	Description
System Registers				
Offset 0x06 3050	***PCI_INTA***			
31:2	Unused	-	-	To ensure software backward compatibility, writes to unused or reserved bits should be zero and reads must be ignored.
1	PCI_INTA	W	0x1	Writes PCI_INTA_N pin value if PCI_INTA_OE is enabled 0: PCI_INTA_N is 0 (asserted) 1: PCI_INTA_N is 1 (de-asserted) To read the PCI_INTA_N pin value use IPENDING MMIO register.
0	PCI_INTA_OE	R/W	0x0	Enable of PCI_INTA_N output 0: Disable PCI_INTA_N output 1: Enable PCI_INTA_N output Note: In order to operate the PCI_INTA_N pin as an open drain pin as required by the PCI specification, the software must enable the output only when driving a '0', i.e. asserting an interrupt. Note: In order to avoid a race condition between the data and the enable or glitches on the PCI_INTA_N pin, the enable should only be changed once the data is stable.
Offset 0x06 3500	***SCRATCH0***			
31:0	SCRATCH0	R/W	-	32-bit writable and readable register. Not cleared at reset for debug purposes.
Offset 0x06 3504	***SCRATCH1***			
31:0	SCRATCH1	R/W	-	32-bit writable and readable register. Not cleared at reset for debug purposes.
Offset 0x06 3508	***SCRATCH2***			
31:0	SCRATCH2	R/W	-	32-bit writable and readable register. Not cleared at reset for debug purposes.
Offset 0x06 350C	***SCRATCH3***			
31:0	SCRATCH3	R/W	-	32-bit writable and readable register. Not cleared at reset for debug purposes.
Offset 0x06 3510	***SCRATCH4***			
31:0	SCRATCH4	R/W	-	32-bit writable and readable register. Not cleared at reset for debug purposes.
Offset 0x06 3514	***SCRATCH5***			
31:0	SCRATCH5	R/W	-	32-bit writable and readable register. Not cleared at reset for debug purposes.
Offset 0x06 3518	***SCRATCH6***			
31:0	SCRATCH6	R/W	-	32-bit writable and readable register. Not cleared at reset for debug purposes.
Offset 0x06 351C	***SCRATCH7***			
31:0	SCRATCH7	R/W	-	32-bit writable and readable register. Not cleared at reset for debug purposes.

Table 9: Miscellaneous System MMIO registers *…Continued*

Bit	Symbol	Access	Value	Description
Offset 0x06 3600	***SPDI_MUX_SEL***			
31:4	Unused	-	-	To ensure software backward compatibility, writes to unused or reserved bits should be zero and reads must be ignored.
3:0	SPDI_MUX_SEL	R/W	0x0	SPDIF IN timestamping, The specific events that may be timestamped are 0000: WS - Word strobe 0001: SWS - Last sub-frame 0010: SPDI_STATUS[0] - Buffer 1 full. 0011: SPDI_STATUS[1] - Buffer 2 full. 0100: SPDI_STATUS[2] - Buffer 1 active. 0101: SPDI_STATUS[3] - Bandwidth Error. 0110: SPDI_STATUS[4] - Parity Error. 0111: SPDI_STATUS[5] - Validity Error. 1000: SPDI_STATUS[6] - User/Channel bits available. 1001: SPDI_STATUS[7] - unlock active. 1010-1111: WS - Word strobe
Offset 0x06 360C	***SPARE_CTRL***			
31:8	Unused	-	-	To ensure software backward compatibility, writes to unused or reserved bits should be zero and reads must be ignored.
7:0	SPARE_CTRL	R/W	-	Spare control register.

9. System Registers Map Summary

Table 10: System Registers Map Summary

Offset	Name	Description
0x06_3000	VDI _MODE	Video/Data input router control register.
0x06_3004	VDO_MODE	Video/Data output router control register.
0x06_3014	SYS_ENDIANESS	System Endian Mode register.
0x06_3050	PCI_INTA	PCI_INTA_N pin control register.
0x06_3200	DCS_DRAM_LO	16-bit DCS-to-MTL memory range low register.
0x06_3204	DCS_DRAM_HI	16-bit DCS-to-MTL memory range high register.
0x06_3208	APERTURE_WE	Write enable register for DCS_DRAM_HI and DCS_DRAM_LO registers.
0x06_3500	SCRATCH0	32-bit writable and readable register.
0x06_3504	SCRATCH1	32-bit writable and readable register.
0x06_3508	SCRATCH2	32-bit writable and readable register.
0x06_350C	SCRATCH3	32-bit writable and readable register.
0x06_3510	SCRATCH4	32-bit writable and readable register.
0x06_3514	SCRATCH5	32-bit writable and readable register.
0x06_3518	SCRATCH6	32-bit writable and readable register.
0x06_351C	SCRATCH7	32-bit writable and readable register.
0x06_3600	SPDI_MUX_SEL	SPDIF IN timestamping multiplexer select register.
0x06_360C	SPARE_CTRL	Spare control register.
0x06_3700	TM32_CONTROL	TM3260 control register.
0x06_3704	TM32_STATUS	TM3260 status register.
0x06_3800	SEMAPHORE0	12-bit semaphore register.
0x06_3804	SEMAPHORE1	12-bit semaphore register.
0x06_3808	SEMAPHORE2	12-bit semaphore register.
0x06_380C	SEMAPHORE3	12-bit semaphore register.
0x06_3810	SEMAPHORE4	12-bit semaphore register.
0x06_3814	SEMAPHORE5	12-bit semaphore register.
0x06_3818	SEMAPHORE6	12-bit semaphore register.
0x06_381C	SEMAPHORE7	12-bit semaphore register.
0x06_3820	SEMAPHORE8	12-bit semaphore register.
0x06_3824	SEMAPHORE9	12-bit semaphore register.
0x06_3828	SEMAPHORE10	12-bit semaphore register.
0x06_382C	SEMAPHORE11	12-bit semaphore register.
0x06_3830	SEMAPHORE12	12-bit semaphore register.
0x06_3834	SEMAPHORE13	12-bit semaphore register.
0x06_3838	SEMAPHORE14	12-bit semaphore register.

Table 10: System Registers Map Summary *...Continued*

Offset	Name	Description
0x06_383C	SEMAPHORE15	12-bit semaphore register.
0x06_3FF4	GLB_REG_PWR_DWN	Power Down Bit for the Global Registers
0x06_3FFC	GLB_REG_MOD _ID	Module Identification and revision information

10. Simplified Internal Bus Infrastructure

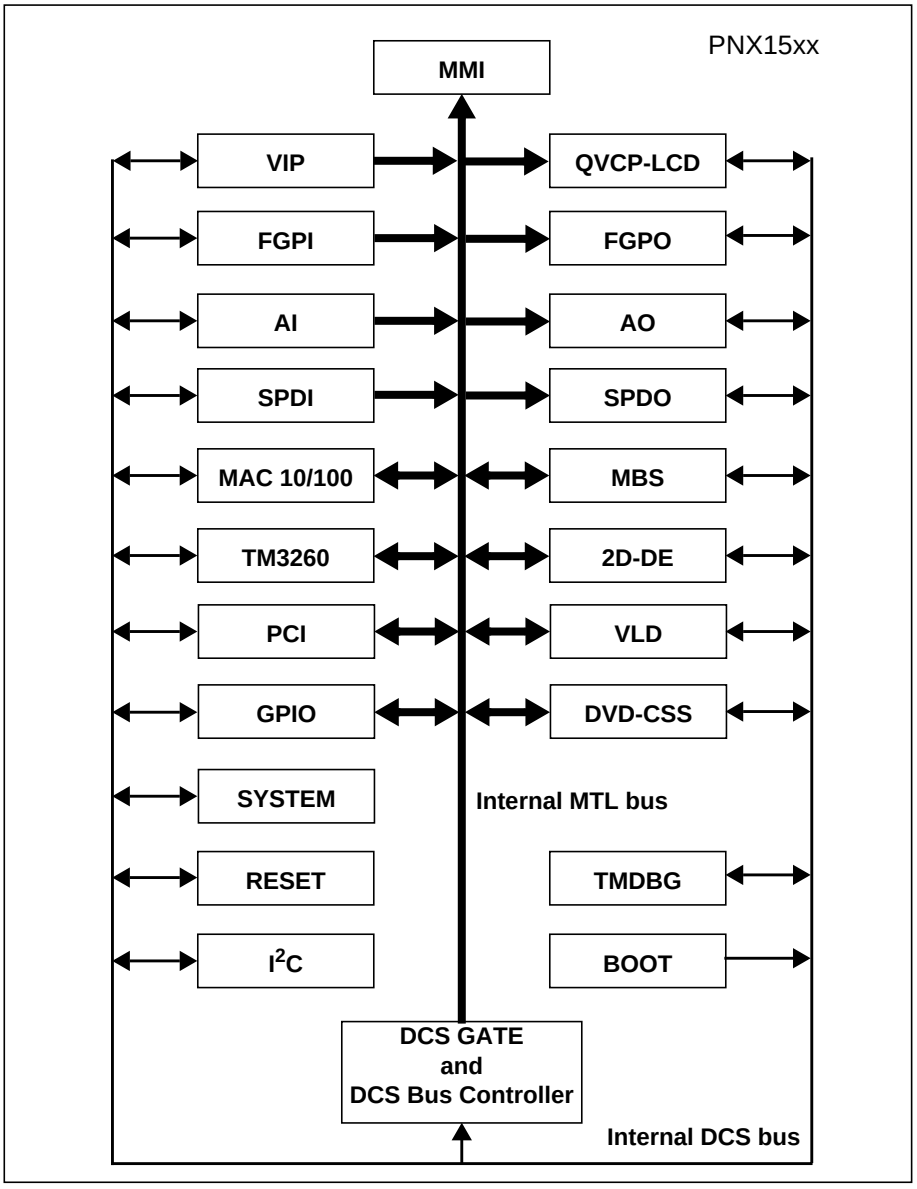


Figure 3: Simplified Internal Bus Infrastructure

More details on the DCS bus in Chapter 30 DCS Network.

11. MMIO Memory MAP

Each module has an address range in the MMIO aperture from which its registers can be accessed. This address range is defined by its starting address, a.k.a. its offset, and the aperture size defined in the MODULE_ID MMIO register. The following table gives the offset position for each module of the PNX15xx Series system. Each module specification contains the internal registers location within its aperture. Therefore the physical address of each MMIO register in the system is defined by the equation:

- MMIO_BASE + Module Offset + Register Offset.

Table 11: MMIO Memory MAP

address offset from MMIO_BASE (PCI base 14)	Module Name	Module ID	Major Module Revision	Minor Module Revision	MMIO size	Summary
0x04,0000	PCI/XIO	0xA051	0x0	0x1	0x00	PCI and XIO (Flash, 68k, IDE) status/control
0x04,5000	IIC	0x0105	0x0	0x3	0x00	I^2C for boot & devices up to 400 kHz
0x04,7000	CLOCK	0xA063	0x0	0x0	0x00	PNX15xx Series Modules Clock Control & Status
0x04,F000	2D DE	0x0117	0x2	0x0	0x10	2D Drawing Engine, includes RAM area
0x06,0000	RESET	0xA064	0x0	0x1	0x00	Endian Mode control, system & peripheral reset control/status, watchdog
0x06,1000	TMDBG	0x0127	0x0	0x0	0x00	TM software debug through JTAG
0x06,3000	GLOBAL	0x0126	0x8	0x1	0x00	Global MMIO registers controlling miscellaneous settings, input & output router settings.
0x06,4000	ARBITER	0x1010	0x0	0x0	0x00	Arbiter
0x06,5000	DDR Ctrl	0x2031	0x1	0x1	0x00	Main Memory Interface
0x07,0000	FGPI	0x014B	0x0	0x1	0x00	Fast Generic Parallel Input
0x07,1000	FGPO	0x014C	0x0	0x2	0x00	Fast Generic Parallel Output
0x07,2000	LAN100	0x3902	0x1	0x1	0x00	10/100 LAN Controller
0x07,3000	LCD Ctrl	0xA050	0x0	0x0	0x00	LCD Controller
0x07,5000	VLD	0x014D	0x0	0x0	0x00	Variable Length Decoder
0x10,0000	TM3260	0x2B80	0x4	0x0	0x01	TM3260 CPU control/status registers
0x10,3000	DCS Bus Ctrl	0xA049	0x0	0x0	0x00	MMIO bus Controller

Table 11: MMIO Memory MAP

address offset from MMIO_BASE (PCI base 14)	Module Name	Module ID	Major Module Revision	Minor Module Revision	MMIO size	Summary
0x10,4000	GPIO	0xA065	0x0	0x1	0x00	GPIO General Purpose Software Serial I/O pins
0x10,6000	VIP	0x011A	0x3	0x0	0x00	Video Input
0x10,9000	SPDIF OUT	0x0121	0x0	0x1	0x00	Sony Philips Digital Interface for serial audio
0x10,A000	SPDIF IN	0x0110	0x0	0x1	0x00	Sony Philips Digital Interface for serial audio
0x10,C000	MBS	0x0119	0x2	0x8	0x00	Memory Based Scaler
0x10,E000	QVCP	0xA052	0x0	0x1	0x00	Quality Video Composition Processor (2 layers)
0x11,0000	AO	0x0120	0x0	0x2	0x00	Audio Output (8 channels)
0x11,1000	AI	0x010D	0x1	0x1	0x00	Audio Input (8 channels)
0x1F,0000	TM3260	n/a	n/a	n/a	0x0F	TM3260 cache tags

12. References

[1] “The TM3260 Architecture Databook”, Oct. 13 2003, Philips.

Chapter 4: Reset

PNX15xx Series Data Book – Volume 1 of 1

Rev. 2 — 1 December 2004 Product data sheet

1. Introduction

The Reset module initiates life for the PNX15xx Series system since it generates all reset signals required for a correct initialization of the entire system (may include board devices).

- It sends reset signals to all DCS bus modules and the TM3260 CPU.
- It sends a reset signal on the SYS_RST_OUT_N pin that can be used by external board devices. This signal is then de-asserted by software.

These resets signals are triggered by hardware (one type) or by software (three types):

- Hardware external reset input to the PNX15xx Series through the pins, POR_IN_N or RESET_IN_N.
- Software assert and release of the SYS_RST_OUT_N reset pin through a write to an MMIO register write.
- Software programmable watchdog timer which asserts the same reset signals as the hardware reset induced by the assertion of RESET_IN_N pin when a time-out is reached.
- Software PNX15xx Series system reset which asserts the same reset signals as the hardware reset induced by the assertion of RESET_IN_N pin.

RST_CAUSE MMIO register holds the cause of the previous reset which allows the software to know what happened before.

2. Functional Description

The Reset module generates three different reset signals to fully initialize a PNX15xx Series system:

- jtag_rst_n. This signal is used internally to reset the JTAG state machine. The signal is only asserted if the POR_IN_N pin is asserted. Therefore the only mean to reset the JTAG state machine of PNX15xx Series is by asserting the POR_IN_N pin.Figure 1

Remark: The JTAG state machine can also be reset through the JTAG pins.

- peri_rst_n. This signal is used internally to reset all the PNX15xx Series modules including the TM3260 CPU. This signal is asserted when one of the following conditions occurs:
  - the POR_IN_N pin is asserted.
  - the RESET_IN_N pin is asserted.
  - the watchdog timer reaches a time-out, Section 2.2.
  - a software reset is asserted, Section 2.3.

Remark: This signal does not reset the JTAG state machine, i.e. it does not assert jtag_rst_n.

- sys_rst_out_n. This signal is sent to the SYS_RST_OUT_N pin and provides a software and hardware solution to reset external devices present on a PNX15xx Series system board. This signal is asserted when one of the following conditions occurs:
  - the POR_IN_N pin is asserted.
  - the RESET_IN_N pin is asserted.
  - the watchdog timer reaches a time-out, Section 2.2.
  - a software reset is asserted, Section 2.3.
  - a software external reset is requested, Section 2.4

In the following the PNX15xx Series system reset refers to the assertion of peri_rst_n and sys_rst_out_n signals.

Figure 1 shows an overview of the Reset module connections to the remaining of the PNX15xx Series system.

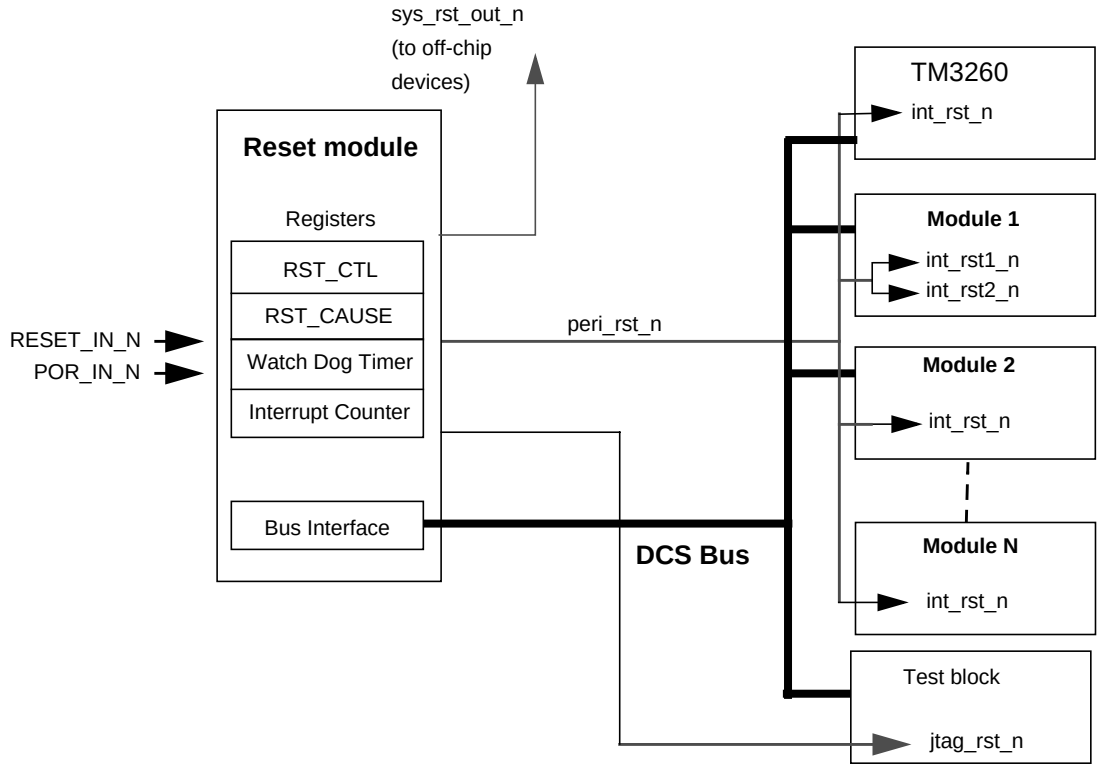


Figure 1: Reset Module Block Diagram

2.1 RESET_IN_N or POR_IN_N?

POR_IN_N is meant to be used at power up of the system. By asserting this pin low as soon as the power sequencing starts ensures limited (if not none) contentions inside the PNX15xx Series system as well as the PNX15xx Series pin level. Furthermore by resetting the JTAG state machine the POR_IN_N signal ensures the PNX15xx Series pins start with the correct mode. This is the cold reset and must always be connected.

RESET_IN_N is complementary to the POR_IN_N signal and could be referenced as the warm reset. A typical application where the feature can be used is a system board where the JTAG boundary scan is to be used to reset PNX15xx Series without executing a full power down and up sequence. In this case the PNX15xx Series JTAG state machine should not be reset. Since all PNX15xx Series pins can become outputs in boundary scan mode it is possible to assert a 0 on the RESET_IN_N pin while the PNX15xx Series system is still under the control of the internal JTAG state machine. This pin may not be connected at board level.

2.2 The watchdog Timer

The internal PNX15xx Series watchdog timer has two operating modes. Both modes result in the assertion of the internal reset signals, peri_rst_n and sys_rst_out_n signals based upon a time-out condition. The modes are referenced as the non interrupt mode and the interrupt mode.

2.2.1 The Non Interrupt Mode

In this mode, the watchdog timer operates as a simple counter. The counter operates with the DCS clock also called MMIO clock (clk_dtl_mmio).

By default, i.e. after a PNX15xx Series system reset, this watchdog counter is not active. The activation is done by writing a value different than 0x0 to the WATCHDOG_COUNT MMIO register. Upon that write, an internal counter of the watchdog timer is reset to 0x0 and starts to count. If the internal counter reaches the WATCHDOG_COUNT value then peri_rst_n and sys_rst_out_n internal reset signals are asserted and the PNX15xx Series system is reset. The reset follows then the regular software reset timing, Section 3.2. If the CPU writes a 0x0 value to the WATCHDOG_COUNT MMIO register before the internal counter reaches the previous WATCHDOG_COUNT value then the internal reset signals are not generated and the internal counter stops counting. Similarly if the CPU writes a value different than 0x0 then the internal counter is reset to 0x0 and starts to count to the new WATCHDOG_COUNT value.

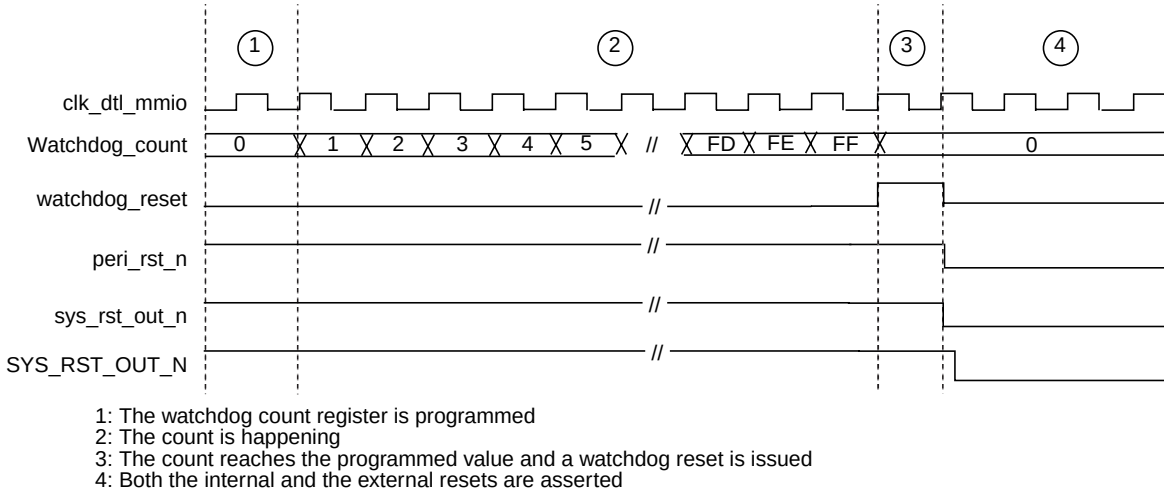
This mode requires the CPU to come back in time to reset the internal counter on a regular basis. TM3260 software may use some of its internal hardware timers [1] to reset on time on the internal counter. The interrupt handler needs to first write a 0x0 value to the WATCHDOG_COUNT register then write a new count value.

The layout of the WATCHDOG_COUNT MMIO register is presented in Section 4..

The following summarizes the sequence of operations

1. Start the internal counter by writing a nonzero value to the WATCHDOG_COUNT MMIO register.
2. A write with 0x0 value to the WATCHDOG_COUNT MMIO register will stop the count. For continuous watchdog timer operation it is not required to write 0x0 first but instead start back directly from step 1).
3. If step 2 does not occur before the count reaches the WATCHDOG_COUNT value the PNX15xx Series system reset is asserted.

The following Figure 2 pictures the events.



1: The watchdog count register is programmed
2: The count is happening
3: The count reaches the programmed value and a watchdog reset is issued
4: Both the internal and the external resets are asserted

Figure 2: Watchdog in Non Interrupt Mode

2.2.2 The Interrupt Mode

In this mode, the watchdog timer generates first an interrupt to the TM3260 before a PNX15xx Series system reset is generated (when a time-out occurs because the TM3260 does not answer in time to the interrupt). The sequence of operations is similar to the non interrupt mode.

First TM3260 CPU writes a value different than 0x0 to the WATCHDOG_COUNT MMIO register. This starts an internal counter from the value 0x0. When the internal counter reaches the WATCHDOG_COUNT value an interrupt, SOURCE 42 (see Section 6.2 on page 3-14) is asserted. From here a second internal counter is started. If this second counter reaches the value previously stored into the INTERRUPT_COUNT MMIO register then a PNX15xx Series system reset is asserted. The reset follows then the regular software reset timing, Section 3.2. If the TM3260 CPU clears the pending interrupt by writing to the INTERRUPT_CLEAR MMIO register, then the PNX15xx Series system reset is not generated.

The following summarizes the sequence of operations

1. Enable the watchdog interrupt. This includes proper set-up of TM3260 internal interrupt controller[1] as well as an enable of the INTERRUPT_ENABLE MMIO register.
2. Initialize the INTERRUPT_COUNT MMIO register with the maximum interrupt latency authorized before a PNX15xx Series reset is asserted.
3. Start the first counter by writing a nonzero value to the WATCHDOG_COUNT MMIO register.
4. A write with 0x0 value to the WATCHDOG_COUNT MMIO register will stop the count. However this is not intended to be used as such.

Remark: A write of any nonzero value other than the current value will reset the count. However this is not intended to be used as such.

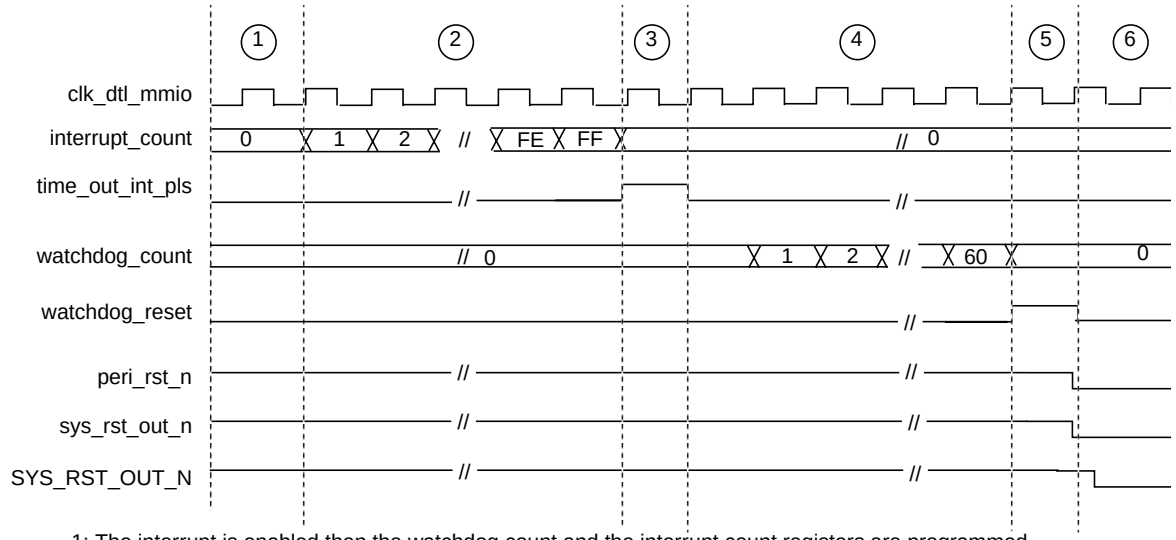
5. If step 4 does not occur before the count reaches the WATCHDOG_COUNT value an interrupt is issued to the TM3260 CPU and the second internal counter (the interrupt counter) starts. The internal watchdog counter is reset and waits the interrupt to be cleared.
6. A write with 0x1 to INTERRUPT_CLEAR stops the interrupt counter and restarts the watchdog counter. Therefore for continuous watchdog timer operation start back at step 5).

Here once the interrupt is asserted then the first counter is reset to zero

7. The interrupt counter reaches the INTERRUPT_COUNT value, the PNX15xx Series system reset is asserted.

The counters operate with the DCS clock also called MMIO clock (clk_dtl_mmio).

The following Figure 3 pictures the events.



1: The interrupt is enabled then the watchdog count and the interrupt count registers are programmed.
2: The interrupt count is happening.
3: The interrupt count reaches the programmed value and a time out interrupt pulse is issued to the CPU.
4: The watchdog counter begins.
5: The interrupt has not been cleared. A watchdog reset is issued.
6: The internal and external resets are asserted.

Figure 3: Watchdog in Interrupt Mode

2.3 The Software Reset

The software reset is started by writing a 0x1 to RST.CTL.DO_SW_RST bit.

The reset follows then the regular software reset timing, Section 3.2.

2.4 The External Software Reset

The signal sys_rst_out_n signal can be asserted by writing a 0x1 to the RST_CTL.ASSERT_SYS_RST_OUT bit.

The signal sys_rst_out_n signal can be de-asserted by writing a 0x1 to the RST_CTL.REL_SYS_RST_OUT bit.

Remark: Upon any of the described ways to reset the PNX15xx Series system the sys_rst_out_n remains asserted until a write with 0x1 occurs to the RST_CTL.REL_SYS_RST_OUT bit.

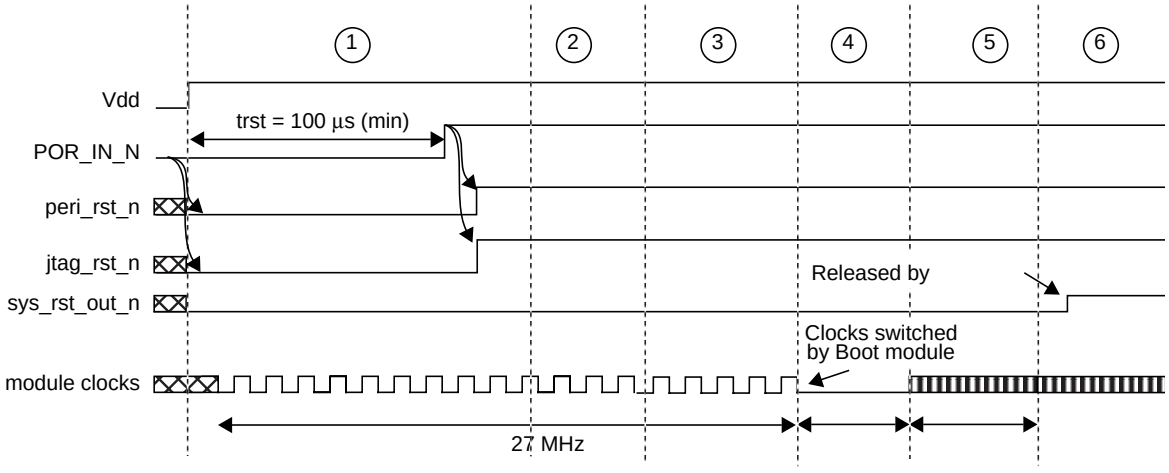
3. Timing Description

3.1 The Hardware Timing

The assertion of POR_IN_N or RESET_IN_N signals causes the assertion of peri_rst_n, sys_rst_out_n and jtag_rst_n (only when POR_IN_N is asserted). See Figure 4. When the Clock module receives the peri_rst_n signal, it ensures that all the PNX15xx Series modules receive the 27 MHz crystal oscillator input. The 27 MHz clock remains active for all the modules until the registers in the Clock module are programmed to switch from 27 MHz to their functional module clocks (either by the boot scripts or by the TM3260). The use of this generic 27 MHz clock allow all the modules to be reset synchronously.

After de-asserting the RESET_IN_N pin, the peri_rst_n is also de-asserted and all modules release their internal resets synchronously. The PLLs come up to their default values while POR_IN_N or RESET_IN_N are asserted. The Clock module will safely (i.e. glitch free) switch clocks from the 27 MHz clock to the separate module functional clocks.

Figure 4 details the hardware reset. Only POR_IN_N is shown. The reset sequence is exactly the same when RESET_IN_N is asserted except that in that case the jtag_rst_n signal is not asserted.



1. POR_IN_N is asserted for 100 µs (min) after power stable. peri_rst_n and jtag_rst_n follows the assertion and the

   release of POR_IN_N. The Clock module kicks off 27 MHz clock to all modules.

2. All module resets sync to 27 MHz and all modules are reset at the same time. The Boot script can now kick

Figure 4: POR_IN_N Timing and Reset Sequence

3.2 The Software Timing

Whenever a watchdog timer time-out occurs or when a software reset is requested by writing to the RST_CTL.DO_SW_RST bit the PNX15xx Series system is reset. Both are referred as software reset. As seen in the previous Section 3.1 it is required to hold the POR_IN_N or the RESET_IN_N signal for at least 100 µs. Therefore the software reset mechanism implements an internal counter that allows to assert the peri_rst_n signal for 100 µs. Similarly to the hardware reset the sys_rst_out_n is also asserted until the TM3260 CPU releases it. The internal counter uses the initial 27 MHz to estimate 100 µs.

4. Register Definitions

Table 1: RESET Module

Bit	Symbol	Access	Value	Description
Reset Module				
Offset 0x06,0000 RST_CTL				
31:3	Unused	W	-	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
2	DO_SW_RST	W	0	0 = No action 1 = Do Software Reset.
1	REL_SYS_RST_OUT	W	0	0 = No action 1 = Release System Reset of External Peripherals.
0	ASSERT_SYS_RST_OUT	W	0	0 = No action 1 = Do System Reset of External Peripherals.
Offset 0x06,0004 RST_CAUSE				
Remark: RST_CTL is set on every time an hardware or software reset occurs.				
31:2	Unused		-	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
1:0	RST_CAUSE	R	N/A	Reset Cause register: 00 = Cause is External System Reset, RESET_IN_N. 01 = Cause is Software System Reset. 10 = Cause is External System Reset, POR_IN_N 11 = Cause is watchdog time-out. Note if multiple resets occur then only the one that is highest in the above order will be registered. As an example RESET_IN_N (00) and POR_IN_N (10) are both asserted. A read would return “10”
Offset 0x06,0008 WATCHDOG_COUNT				
31:0	WATCHDOG_COUNT	R/W	0	Value to count to in order to either assert an interrupt (interrupt mode) or a reset (non interrupt mode)
Offset 0x06,000C INTERRUPT_COUNT				
31:0	INTERRUPT_COUNT	R/W	0	Value to count to after the interrupt is asserted before asserting the system reset
Offset 0x06,0FE0 INTERRUPT STATUS				
31:1	Unused	R/W	-	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
0	WATCHDOG_INTERRUPT	R	0	1: watchdog interrupt is asserted
Offset 0x06,0FE4 INTERRUPT_ENABLE				
31:1	Unused	R/W	-	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
0	WATCHDOG_INTERRUPT_ENABLE	R/W	0	1: interrupt enabled 0: interrupt NOT enabled
Offset 0x06,0FE8 INTERRUPT_CLEAR				
31:1	Unused	R/W	-	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.

Table 1: RESET Module *…Continued*

Bit	Symbol	Access	Value	Description
0	WATCHDOG_INTERRUPT_CLEAR	R/W	0	1: clear interrupt
Offset 0x06,0FEC INTERRUPT_SET				
31:1	Unused	R/W	-	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
0	WATCHDOG_INTERRUPT_SET	R/W	0	1: set interrupt
Offset 0x06,0FFC MODULE_ID				
31:16	MODULE_ID	R	0xA064	Reset module ID
15:12	MAJOR_REV	R	0x0	Changed upon functional revision, like new feature added to previous revision
11:8	MINOR_REV	R	0x1	Changed upon bug fix or non functional changes like yield improvement.
7:0	APERTURE	R	0x0	Encoded as: Aperture size = 4K*(bit_value+1). The bit value is reset to 0 meaning a 4K aperture for the Global register 1 module according to the formula above.

5. References

[1] “The TM3260 Architecture Databook”, Aug. 1st 2003, Philips.

Chapter 5: The Clock Module

PNX15xx Series Data Book – Volume 1 of 1

Rev. 2 — 1 December 2004 Product data sheet

1. Introduction

The Clock module is the heart of the PNX15xx Series system. Its role is to provide and control all the clocks of the system. The main characteristics of the Clock module is to be low cost. It generates all the PNX15xx Series system clocks from one unique source, a 27 MHz input crystal. The clock module features can be regrouped as follows:

- Use of Phase Locked Loop (PLL) circuits, Direct Digital Synthesizers (DDS) or simple clock dividers to meet the frequency and jitter requirements of all PNX15xx Series modules.
- All the clocks are software programmable and support powerdown features.
- Clock switching or clock frequency changes occur glitch free thank to dedicated hardware.

2. Functional Description

The Clock Module has three main internal interfaces:

- an interface to a Custom Analog Block (CAB). The CAB module includes 2 PLLs, several high speed clock dividers and 9 DDS blocks.
- an interface to a dedicated low jitter PLL used for the DDR memory controller.
- an MMIO interface to allow the programming of all configuration registers.

A 27 MHz crystal clock provides the source clock for all PLLs in the CAB block and for the low jitter PLL. The PLLs are programmable from the Clock module registers to generate a range of possible frequencies. The DDS blocks are required to make slight adjustments to each video and audio clock to track transmission sources. Software controls this tracking by programming the relevant DDS block to adjust the clock. These adjustments are made in steps of 0.4 Hz. The DDS clocks are derived from the internal 1.728 GHz PLL (64 times the 27 MHz input crystal). The DDS jitter is less than 0.58 ns. The video clock requirements may require a shorter term jitter so an additional PLL is provided to smooth out the DDS jitter. This combines the two video clock requirements, low jitter and high precision adjustment of the clock frequency to meet color burst requirements but also track the audio signals.

The Clock Module consists of an MMIO-interface with programmable Clock and PLL control registers, and a series of control logic for every clock generated. The clock control logic will consist of:

- programmable dividers, controlled by configuration registers
- clock blocking circuitry to allow for safe, glitch-free switching of clocks. Clocks are typically switched when:
  - PLLs or dividers are reprogrammed
  - clocks are switched on/off for powerdown reasons
  - following reset and boot-up of the chip when all clocks are switched from 27 MHz to their programmed functional frequencies
  - Design for Debug (DfD) features e.g. clock stretching, Section 2.6.

Figure 1 shows a block diagram of the Clock Module. Additional Design For Test (DFT) have been added into the drawing and can be disregarded for functional behavior. The signals in red are for ATE purpose and are disabled in normal functional operating mode.

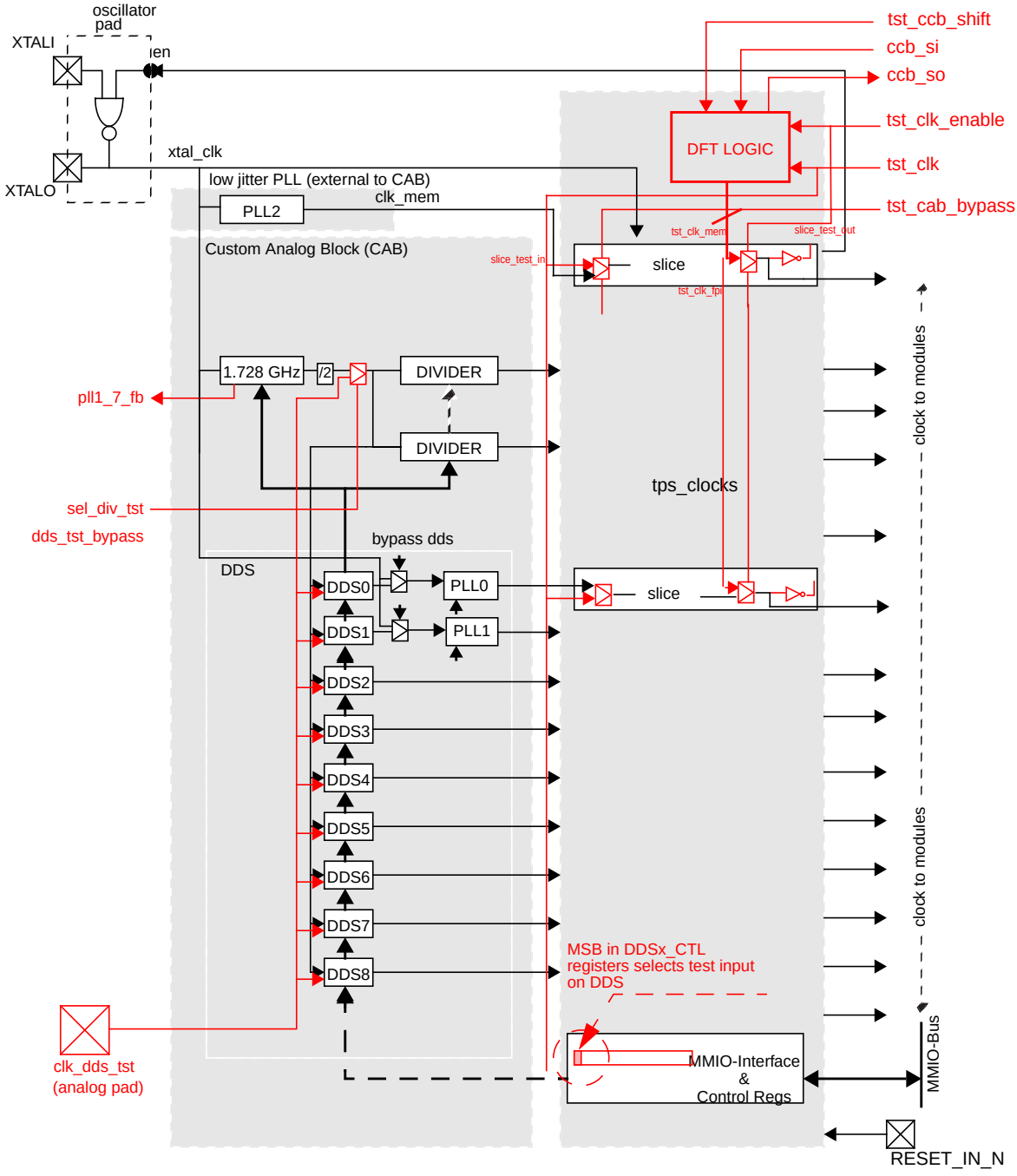


Figure 1: Clock Module Block Diagram

Remark: Not all the clocks to the modules are generated in the Clock Module, there will be other clocks which will come into PNX15xx Series from external sources. Some of these clocks will be fed through the Clock Module so that they may undergo the same controls required during reset, powerdown, DFT and DfD.

2.1 The Modules and their Clocks

Table 1 presents a summary of all the clocks used in the PNX15xx Series system. The table is organized with the module name, the corresponding internal clock signal name, a brief description, the operating frequency range or the available clock speeds, the MMIO registers that control the clock selection and the “standard” clock used. The “standard” clock used is the recommended clock use when all the clock generation capabilities are used. This is based on common board systems, however it is possible to use other clock sources. See Section 3. on page 5-31 for MMIO registers layout. Table 1 can be used as a quick reference to see the PNX15xx Series clocking capabilities.

Table 1: PNX15xx Series Module and Bus Clocks

Bus or Module	Signal Name	Description	Frequencies	MMIO Clock Module Control Register(s)	Standard Clock Source
DDR SDRAM	clk_mem	MM_CLK	up to 200 MHz	PLL2_CTL CLK_MEM_CTL	PLL2
TM3260 CPU	clk_tm	The TM3260 clock	up to 300 MHz depending on speed grade	PLL0_CTL DDS0_CTL CLK_TM_CTL	PLL0, fed by the input 27 MHz crystal)
MMIO	clk_dtl_mmio	MMIO clock or DCS clock	• 157 MHz • 144 MHz • 133 MHz • 123 MHz • 115 MHz • 108 MHz • 102 MHz • 54 MHz	CLK_DTL_MMIO_CTL	1.728 GHz DIVIDERS
2DDE	clk_2ddE	2D drawing engine clock	• 144 MHz • 123 MHz • 108 MHz • 96 MHz • 86 MHz • 78 MHz • 72 MHz • 66 MHz	CLK_2DDE_CTL	1.728 GHz DIVIDERS
PCI	clk_pci	PCI_SYS_CLK	33.23 MHz	CLK_PCI_CTL The PCI module gets its primary clock directly from the PCI_CLK pin.	N/A

Table 1: PNX15xx Series Module and Bus Clocks

Bus or Module	Signal Name	Description	Frequencies	MMIO Clock Module Control Register(s)	Standard Clock Source
MBS	clk_mbs	MBS clock	• 144 MHz • 123 MHz • 108 MHz • 96 MHz • 86 MHz • 78 MHz • 72 MHz • 66 MHz	CLK_MBS_CTL	1.728 GHz DIVIDERS
TMDBG GPIO	clk_tstamp	Timestamp clock	108 MHz	CLK_TSTAMP_CTL	1.728 GHz DIVIDERS
10/100 Ethernet MAC	clk_lan	Ethernet PHY Clock	up to 50 MHz	CLK_LAN_CTL and • PLL1_CTL and DDS1_CTL • or DDS4_CTL • or DDS7_CTL	DDS7
	clk_lan_tx	Ethernet Transmit Clock	up to 27 MHz	CLK_LAN_TX_CTL	EXTERNAL
	clk_lan_rx	Ethernet Receiver Clock	up to 27 MHz	CLK_LAN_RX_CTL	EXTERNAL
IIC	clk_iic	I^2C module clock	24 MHz	CLK_IIC1_CTL	1.728 GHz DIVIDERS
	scl1_out	IIC_SCL pin	24 MHz/**n**	**n** is controlled by the I^2C module to generate an up to 400 KHz clock.	INTERNAL
DVDD	clk_dvdd	DVDD block	• 144 MHz • 123 MHz • 108 MHz • 96 MHz • 86 MHz • 78 MHz • 72 MHz • 54 MHz	CLK_DVDD_CTL	1.728 GHz DIVIDERS

Table 1: PNX15xx Series Module and Bus Clocks

Bus or Module	Signal Name	Description	Frequencies	MMIO Clock Module Control Register(s)	Standard Clock Source
QVCP	clk_qvcp_out	VDO_CLK1 External pixel clock	Up to 81 MHz Typical values: • 27 MHz • 54 MHz • 65 MHz	PLL1_CTL DDS1_CTL CLK_QVCP_CTL	Smoothing DDS1/PLL1 combination
	clk_qvcp_pix	internal pixel clock	Up to 50 MHz	CLK_QVCP_PIX_CTL	INTERNAL
	clk_qvcp_proc	processing layer clock	• 144 MHz • 133 MHz • 108 MHz • 96 MHz • 86 MHz • 78 MHz • 58 MHz • 39 MHz • 33 MHz • 17 MHz	CLK_QVCP_PROC_CTL Maximum speed supported is 96 MHz. Other higher speeds are reserved for future use.	1.728 GHz DIVIDERS
	clk_lcd_tstamp	LCD timestamp	27 MHz	N/A	
VIP	clk_vip	VDI_CLK1 External pixel clock	up to 81 MHz	DDS7_CTL CLK_VIP_CTL	EXTERNAL
VLD	clk_vld	MPEG-2 Variable Length Decoder	• 144 MHz • 133 MHz • 108 MHz • 96 MHz • 86 MHz • 78 MHz • 72 MHz • 66 MHz	CLK_VLD_CTL	1.728 GHz DIVIDERS
AI	ai_osclk	AO_OSCLK External Oversampling clock	up to 50 MHz	DDS4_CTL AI_OSCLK_CTL	DDS4
	ai_sck		up to 25 MHz	AI_SCK_CTL	EXTERNAL or INTERNAL
AO	ao_osclk	AI_OSCLK External Oversampling clock	up to 50 MHz	• PLL1_CTL and DDS1_CTL • or DDS3_CTL AO_OSCLK_CTL	DDS3
	ao_sck		up to 25 MHz	AO_SCK_CTL	EXTERNAL or INTERNAL

Table 1: PNX15xx Series Module and Bus Clocks

Bus or Module	Signal Name	Description	Frequencies	MMIO Clock Module Control Register(s)	Standard Clock Source
GPIO	clk_gpio_4q	GPIO FIFO clock	up to 108 MHz	DDS8_CTL	DDS8
	clk_gpio_5q	GPIO FIFO clock	up to 108 MHz	DDS7_CTL	
	clk_gpio_6q_12	GPIO FIFO clock/ external clock	up to 108 MHz	DDS6_CTL	DDS6
	clk_gpio_13	external clock	up to 108 MHz	DDS5_CTL	
	clk_gpio_14	external clock	up to 108 MHz	DDS2_CTL	-
SPDIO	clk_spdo	SPDO module clock	up to 40 MHz	DDS5_CTL CLK_SPDO_CTL	DDS5
	clk_spdi	SPDI module clock	• 72 MHz • 144 MHz	CLK_SPDI_CTL	1.728 GHz DIVIDERS
FGPI	clk_fgpi		up to 100 MHz	• DDS3_CTL • or DDS8_CTL CLK_FGPI_CTL	DDS8
FGPO	clk_fgpo		up to 100 MHz	• PLL1_CTL and DDS1_CTL • or DDS2_CTL CLK_FGPO_CTL	DDS2

2.2 Clock Sources for PNX15xx Series

All clocks in the PNX15xx Series clock system are generated from 5 possible sources:

- 2 identical PLLs within the CAB block
- 1 separate PLL for the memory system called PLL2
- high frequency dividers from the 1.728 GHz PLL in the CAB
- the DDS blocks within the CAB
- external clock inputs, or derived from input data streams

2.2.1 PLL Specification

A PLL consists of a Voltage Controlled Oscillator (VCO) and a Post Divide (PD) circuit, as presented in Figure 2.

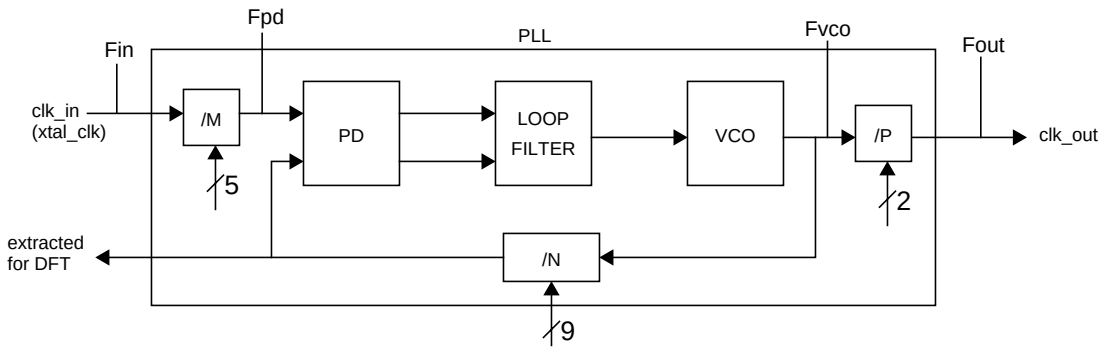


Figure 2: PLL Block Diagram

The frequency from the VCO, F_{VCO} can be determined as follows:

$$F_{VCO} = 27MHz \times \frac{N}{M} \quad (1)$$

F_{VCO} can be post divided by 1, 2, 4 and 8 according to the following equation:

$$F_{out} = \frac{F_{vco}}{2^P} \quad (2)$$

The bit width of N, M, P is 9, 5 and 2 bits respectively. The N, M and P bits are programmable register bits in the Clock module control registers, PLL0_CTL and PLL1_CTL. PLL2_CTL does not allow to control the P parameter since it is fixed to '1', i.e. divides F_{VCO} by 2, to ensure a 50% duty cycle clock on the DDR SDRAM interface.

Remark: Using a value of 0 for either M or N could lead to undesirable behavior. For that reason, setting either M or N to 0 will result in a value of 1 being used for both M and N. Assuming the P value is set to 0, this will result in a PLL output frequency of 27 MHz.

PLL Limitations

The following equations must be met

$$2MHz \le F_{in} \le 150MHz \quad (3)$$

$$100MHz \le F_{vco} \le 600MHz \quad (4)$$

$$2MHz \le F_{pd} \le 27MHz \quad (5)$$

General Recommendations

- Keep M with low values

- Run the VCO as high as possible, therefore for low output frequencies chose high P values
- Ensure $30 \leq N \leq 180$ and track N with the following current adjustment values:

Table 2: Current Adjustment Values Based on N

30-37	38-46	47-54	55-63	64-72	73-82	83-89	90-97	98-107	108-116	117-125	126-133	134-142	143-151	152-160	161-180
0xF	0xE	0xD	0xC	0xB	0xA	0x9	0x8	0x7	0x6	0x5	0x4	0x3	0x2	0x1	0x0

PLL Settings

An easy way to determine the N over M ratio is to meet the PLL limitations seen above and solve the following equation:

$$\frac{N}{M} = \frac{F_{vco}}{F_{in}} \quad (6)$$

PLL Setting Examples

Table 1 presents some other typical examples to set the PLL N, M and P parameters. PLL2 (for the DDR) has the P parameter wired to '1'.

Table 3: PLL Setting Examples

F_{out}	F_{vco}	F_{in}	M	N	P	ADJ	Destination Examples
27 MHz	216 MHz	DDS1 27 MHz	4	0x20	3	0xF	QVCP from DDS1 50% duty cycle recommended
54 MHz	432 MHz	DDS1 27 MHz	3	0x30	3	0xD	QVCP from DDS1 50% duty cycle recommended
65 MHz	520 MHz	DDS1 27.012987 MHz	4	0x4D	3	0xA	QVCP from DDS1 50% duty cycle recommended
81 MHz	324 MHz	DDS1 27 MHz	3	0x24	2	0xF	QVCP from DDS1 50% duty cycle recommended
133.07 MHz	266.14 MHz	27 MHz CRYSTAL	7	0x45	1	0xB	DDR266, i.e. < 133.333333 MHz MM_CK
166.5 MHz	333 MHz	27 MHz CRYSTAL	3	0x25	1	0xF	DDR333, i.e. < 166.666666 MHz MM_CK
199.8 MHz	399.6 MHz	27 MHz CRYSTAL	5	0x4A	1	0xA	DDR400, i.e. < 200 MHz MM_CK
266.63 MHz	533.25 MHz	27 MHz CRYSTAL	4	0x4F	1	0xA	266 MHz TM3260
300.38 MHz	600.75 MHz	27 MHz CRYSTAL	4	0x59	1	0x9	300 MHz TM3260

PLL Characteristics

Table 4: PLL Characteristics

PLL Data	
Input clock frequency	$2MHz \leq F_{in} \leq 150MHz$
VCO input frequency	$3MHz \leq F_{pd} \leq 27MHz$
VCO output frequency	$100MHz \leq F_{vco} \leq 600MHz$
Output frequency	$50MHz \leq F_{out} \leq 600MHz$
Jitter (high frequency)	< 150 ps
Lock time	< 100 µs
Duty Cycle	50-50 (with P=1, 2, 3) [37,63]-[63,37] (with P=0)

2.2.2 The Clock Dividers

The clock dividers allow to generate internally low jitter fixed clocks derived from the 1.728 GHz PLL. Resulting jitter is higher than the PLL jitter but remains less than 200 ps. Table 5 shows the 22 available internal clocks.

Table 5: Internal Clock Dividers

Clock Name	Clock Source	Divider Value	Exact Frequency
clk_192	1.728 GHz	9	192 MHz
clk_173	1.728 GHz	10	172.8 MHz
clk_157	1.728 GHz	11	157.0909 MHz
clk_144	1.728 GHz	12	144 MHz
clk_133	1.728 GHz	13	132.9231 MHz
clk_123	1.728 GHz	14	123.4286 MHz
clk_115	1.728 GHz	15	115.2 MHz
clk_108	1.728 GHz	16	108 MHz
clk_102	1.728 GHz	17	101.6471 MHz
clk_96	clk_192	2	96 MHz
clk_86	clk_173	2	86.4 MHz
clk_78	clk_157	2	78.54545 MHz
clk_72	clk_144	2	72 MHz
clk_66	clk_133	2	66.46155 MHz
clk_62	clk_123	2	61.7143 MHz
clk_58	clk_115	2	57.6 MHz
clk_54	clk_108	2	54 MHz
clk_48	clk_192	4	48 MHz
clk_39	clk_157	4	39.272725 MHz
clk_33	clk_133	4	33.230775 MHz
clk_24	clk_192	8	24 MHz
clk_17	clk_133	8	16.6153875 MHz
clk_13_5	clk_108	8	13.5 MHz

2.2.3 The DDS Clocks

The DDS clocks are recommended for clocks that need to track dynamically another frequency by very small steps. The following equations characterize the PNX15xx Series DDS blocks:

$F_{DDS} = \frac{1.728GHz \times N}{2^{32}}$, where N is a 31-bit value stored in the DDS[8:0]_CTL MMIO registers (7)

$$jitter = \frac{1}{1.728GHz} = 0.579ns \quad (8)$$

$$step = \frac{1.728GHz}{2^{32}} = 0.4Hz \quad (9)$$

2.2.4 DDS and PLL Assignment Summary

The Figure 6 summarizes the assignment of the different DDSes of the PNX15xx Series system.

Table 6: DDS and PLL Clock Assignment

Source	Destinations			
PLL0	clk_tm			
PLL1	clk_fgpo	clk_lan	clk_qvcp_out	ao_osclk
PLL2	clk_mem			
DDS0/PLL0	clk_tm			
DDS1/PLL1	clk_fgpo	clk_lan	clk_qvcp_out	ao_osclk
DDS2	clk_fgpo	clk_gpio_14		
DDS3	ao_osclk			
DDS4	ai_osclk	clk_lan		
DDS5	clk_spdo	clk_gpio_13		
DDS6	clk_gpio_q6_12			
DDS7	clk_vip	clk_gpio_q5	clk_lan	
DDS8	clk_gpio_q4	clk_fgpi		

2.2.5 External Clocks

Table 7 lists all the possible external clocks to PNX15xx Series. The definition of an external clock is any in-coming clock that feeds a PNX15xx Series module or any internal PNX15xx Series clock that can drive a PNX15xx Series I/O pin.

Table 7: External Clocks

Signal Name	Frequency	IN/OUT	PIN I/O Name	Description
xtal_clk	27 MHz	CRYSTAL IN	XTAL_IN	27 MHz clock input from oscillator pad
clk_pci	33.23 MHz	OUT	PCI_SYS_CLK	Clock to off-chip PCI devices; note this signal may be routed back into the PCI_CLK input pad.
clk_pci_i	up to 33.33 MHz	IN	PCI_CLK	External PCI module clock

Table 7: External Clocks

Signal Name	Frequency	IN/OUT	PIN I/O Name	Description
mm_clk_out, clk_mem	up to 200 MHz	OUT	MM_CLK MM_CLK#	DDR SDRAM clock output
clk_vip	up to 81 MHz	IN/OUT	VDI_CLK1	VIP clock
clk_fgpi	up to 100 MHz	IN/OUT	VDI_CLK2	FGPI clock
clk_qvcp	up to 81 MHz	IN/OUT	VDO_CLK1	QVCP clock
clk_fgpo	up to 100 MHz	IN/OUT	VDO_CLK2	FGPO clock
ai_osclk	up to 50 MHz	OUT	AI_OSCLK	Audio Input oversampling clock
ai_sck	up to 25 MHz	IN/OUT	AI_SCK	Audio Input input/output bit clock
ao_osclk	up to 50 MHz	OUT	AO_OSCLK	Audio Output oversampling clock
ao_sck	up to 25 MHz	IN/OUT	AO_SCK	Audio Output input/output bit clock
clk_lan	up to 50 MHz	OUT	LAN_CLK	To 10/100 MAC PHY clock
clk_lan_tx	up to 27 MHz	IN	LAN_TX_CLK	From 10/100 MAC PHY transmit clock
clk_lan_rx	up to 27 MHz	IN	LAN_RX_CLK	From 10/100 MAC PHY receive clock
clk_gpio_4q	up to 108 MHz	IN/OUT	GPIO04	GPIO sampling/pattern generation clock
clk_gpio_5q	up to 108 MHz	IN/OUT	GPIO05	GPIO sampling/pattern generation clock
clk_gpio_6q_12	up to 108 MHz	IN/OUT OUT	GPIO06 GPIO12	GPIO sampling/pattern generation clock GPIO board level clock
clk_gpio_13	up to 108 MHz	OUT	GPIO13	GPIO board level clock
clk_gpio_14	up to 108 MHz	OUT	GPIO14	GPIO board level clock

Remark: Refer to Chapter _ to see series resistors board requirements.

2.3 Clock Control Logic

All the generated PNX15xx Series clocks follow the generic block diagram presented in Figure 3. The signals in red are for ATE purpose and are disabled in normal

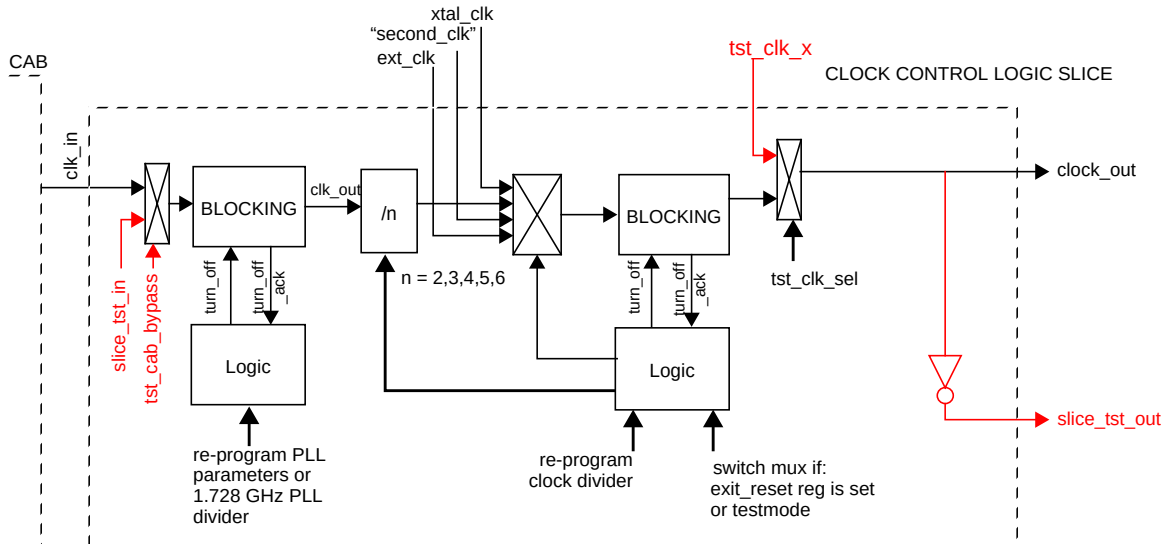


Figure 3: Block Diagram of the Clock Control Logic

functional operating mode.

The clock module allows several clock sources per clock signal. The different clock sources are selected with a multiplexer. In order to guaranty a glitch free dynamic clock switch a blocking block is added after the clock multiplexer.

The same blocking mechanism is necessary when the PLL control register is re-programmed since the PLL clock needs first to be stable, i.e. locks, before it can be used by any module. So the PLL clock is first blocked by the blocking circuit before the new PLL parameters are passed to the PLL. The Blocking circuit will block the clock output when the turn_off signal is set by the blocking logic. The clock is blocked after a falling edge to ensure the clock is held low. Once the blocking circuit has blocked the clock, the turn_off_ack signal is set to high, and it is then safe to pass the new parameters to the PLL.

The blocking will be released after a safe interval of 300 µs. The 300 µs is counted using the 27 MHz xtal_clk. Figure 4 illustrates the sequence of events. The second blocking lasts for less than 10 xtal_clk cycles since it assumes the clocks are stable.

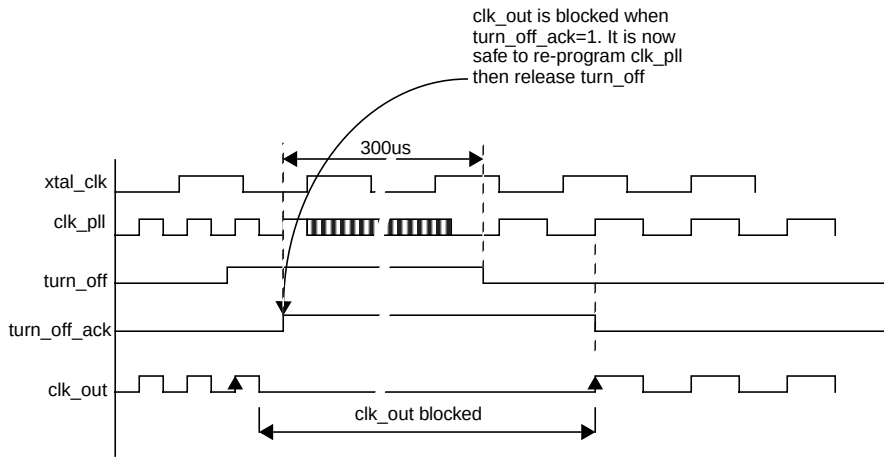


Figure 4: Waveforms of the Blocking Logic

Remark: That 2 blocking circuits are used so that xtal_clk may continue being output uninterrupted while the PLL is being re-programmed

Clocks are also switched if:

- the system has come out of reset and boot-up sequence
- a clock needs to be stretched or stopped for DfD (Section 2.6)

2.4 Bypass Clock Sources

In the event of any issue with the clock sources from the CAB, it is possible to switch these clocks to off-chip sources. These external clock sources will be routed through the GPIO pins as summarized in Table 8. This mode is not meant to be a functional operating mode but just a help for bringup systems based on PNX15xx Series.

Table 8: Bypass Clock Sources

Clocks from Clock Module	Bypass Control Register	GPIO pin Assignment
clk_tm	CLK_TM_CTL	AI_WS
clk_mem	CLK_MEM_CTL	GPIO[7]
clk_2dde	CLK_2DDE_CTL	AI_SD[1]
clk_pci	CLK_PCI_CTL	AI_SD[2]
clk_mbs	CLK_MBS_CTL	AI_SD[3]
clk_tstamp	CLK_TSTAMP_CTL	AO_WS
clk_lan	CLK_LAN_CTL	AO_SD[0]
clk_iic	CLK_IIC_CTL	AO_SD[1]
clk_dvdd	CLK_DVDD_CTL	AO_SD[2]
clk_dtl_mmio	CLK_DTL_MMIO_CTL	AO_SD[3]

Table 8: Bypass Clock Sources

Clocks from Clock Module	Bypass Control Register	GPIO pin Assignment
clk_qvcp	CLK_QVCP_OUT_CTL	XIO_ACK
clk_qvcp_pix	CLK_QVCP_PIX_CTL	XIO_D[8]
clk_qvcp_proc	CLK_QVCP_PROC_CTL	XIO_D[9]
clk_lcd_tstamp	CLK_LCD_TSTAMP_CTL	XIO_D[10]
clk_vip	CLK_VIP_CTL	XIO_D[11]
clk_vld	CLK_VLD_CTL	XIO_D[12]
ai_osclk	AI_OSCLK_CTL	XIO_D[13]
ao_osclk	AO_OSCLK_CTL	XIO_D[14]
clk_spdo	CLK_SPDO_CTL	XIO_D[15]
clk_spdi	CLK_SPDI_CTL	LAN_TXD[0]
clk_gpio_q4	CLK_GPIO_Q4_CTL	LAN_TXD[1]
clk_gpio_q5	CLK_GPIO_Q5_CTL	LAN_TXD[2]
clk_gpio_q6_12	CLK_GPIO_Q6_12_CTL	LAN_TXD[3]
clk_gpio_13	CLK_GPIO_13_CTL	LAN_RXD[0]
clk_gpio_14	CLK_GPIO_14_CTL	LAN_RXD[1]
clk_fgpo	CLK_FGPO_CTL	LAN_RXD[2]
clk_fgpi	CLK_FGPI_CTL	LAN_RXD[3]

2.5 Power-up and Reset sequence

On power-up, the Clock module outputs the default 27 MHz clocks to all the PNX15xx Series modules. Once the Reset module has released the internal module resets, the boot-up sequence executed by the Boot module starts off the 27 MHz clock. At some point in the boot up sequence, the Boot module switches TM3260 and the DDR clocks to the associated PLLs, PLL0 and PLL2. The Clock module keeps feeding the other PNX15xx Series modules with the initial 27 MHz clock until the software decides otherwise.

2.6 Clock Stretching

The TM3260 clock, clk_tm, can be paused or stretched for one clock pulse. A counter counts to a pre-programmed value. When this value is reached the clock gating circuit will turn off the TM3260 clock for one clock period. Then the TM3260 clock is turned back on.

The procedure to operate the clock stretching circuit is to program the CLK_STRETCHER_CTL MMIO register to the value desired between clock stretches. For example a value of 3 turns off the clock every 3 clocks as pictured in Figure 5.

A Write to the CLK_STRETCHER_CTL register acts as the enable for the feature.

A write with a 0 value stops the clock stretching circuit.

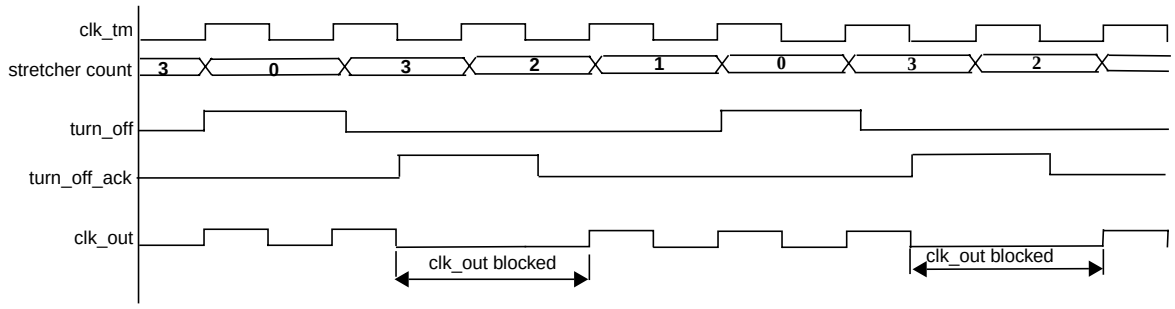


Figure 5: Clock Stretcher

2.7 Clock Frequency Determination

This feature allows the measuring of the internal PLL's and DDS's. This is used for basic test mode only. The enable bits of the CLK_FREQ_CTL choose which of the 12 clocks to test (PLL0 - PLL2, DDS2 - DDS8). The count bits choose a count that is based on the Xtal clock. While the counting proceeds another counter counts the number of clocks of the chosen clock. When the xtal count ends the done bit in the CLK_FREQ_CTL will be set. At this point the CLK_COUNT_RESULTS register can be read. Knowing the pre-programmed value of xtal clocks and the number of clocks of the chosen clock then the frequency of the chosen clock can be determined.

Example:

Program the CLK_FREQ_CTL register for a count of 0x7F. The CLK_COUNT_RESULTS register is read after seeing the DONE bit in the CLK_FREQ_CTL set. The value is 0x24B.

if xtal is 27 MHz (37ns) then the total period of count time is $0x7F \times 37 = 4699ns$

So 0x24B clocks were counted in 4699 ns. Therefore the period of the measured clock is:

$$\frac{4699}{0x24B} = 8.01ns$$

which is approximately 125 MHz. A simpler formula is:

$$Frequency = \frac{CountResult}{XtalPeriod \times Programmed}$$

where:

CountResult is the value read from the CLK_COUNT_RESULTS register, Programmed is the value programmed in the CLK_FREQ_CTL register and XtalPeriod is the period in ns of the input crystal clock.

2.8 Power Down

All clocks generated in the clock module may be disabled by programming the relevant clock enable bit of each clock control register. It is possible to gate module clocks in individual modules rather than in the Clock Module. The advantages of centralizing the clock gating are summarized in Table 9.

Table 9: Advantages of Centralized Clock Gating Control

	Clock Gating in Module	Clock Gating in Clock Module	Comments
Logic & s/w point of view	+	-	More logical for s/w to write to Module reg's to switch off module_clks
History (existing modules)	-	+	Existing Modules and IP modules are usually not delivered with clock gating implemented
Risk	-	+	Clock control is safer being centralized, rather than scattered in every module
Switching of PLLs/debug mode	-	+	Clocks are already blocked in the clock module during re-programming of PLLs and dividers or during debug mode.

To power down all the clocks including the MMIO clock software running on TM3260 must follow this simple procedure.

1. Power down all the clocks with the exception of the TM3260 CPU clock, clk_tm, and the MMIO clock. Accomplish this by writing a zero to bit 0 of each of the clock control registers. Before doing so, proper care has to be taken to ensure that the relevant modules have been disabled.
2. Write to the CLK_TM_CTL MMIO register with a value of 0x00000008. This will first turn off the TM3260 clock and later the MMIO clock.

Remark: The MMIO clock needs to be turned off last but the command needs to come from the TM3260 so they both need to be turned off together.

More details on the PNX15xx Series powerdown can be found in the Chapter 27 Power Management.

2.8.1 Wake-Up from Power Down

There are three ways to wake up the PNX15xx Series when the MMIO clock is turned off

1) Wake-up Timer

2) GPIO Interrupt

3) External wake-up signal on GPIO[15]

The wake-up timer is in the clock block and is controlled by the CLK_WAKEUP_CTL. The wake-up timer is enabled when any value except 0 is written to it. After a value is written to this register the timer starts counting Xtal clocks (27 MHz) until the value programmed in the register is reached. Once the value is reached both the MMIO and the TM3260 clocks are re-activated to 27 MHz.

The GPIO interrupt comes from the GPIO block and is the “OR” of all the FIFO and timestamp registers. This way a GPIO pin can be monitored and when an event occurs the interrupt to the processor awakes the system. Bit ‘0’ of the CLK_WAKEUP_CTL enables the GPIO interrupt.

The external signal is the dedicated GPIO pin 15. This signal must be active for at least one xtal_clk clock period. It is expected that this signal will stay active until the CPU responds which will be several xtal clock periods. Bit ‘1’ of the CLK_WAKEUP_CTL enables the external interrupt. GPIO[15] must be low when entering in power down mode since the wake-up procedure is started when the GPIO[15] pin is set to high for at least one xtal_clk clock cycle.

2.9 Clock Detection

Clock detection is required in the case of an external clock being removed or disconnected e.g if the video cable to the set top box is suddenly removed and an external video clock thereby stopped. this type of event is detected by the Clock module. Also the Clock module can detect when the cable is re-connected and a clock is present again.

These events are flagged by an interrupt which is routed to the TM3260.

The clock detection will be done on the following clocks inputs to PNX15xx Series:

- VDI_CLK1 (clk_vip)
- AI_SCK
- AO_SCK
- VDI_CLK2 (clk_fgpi)
- VDO_CLK2 (clk_fgpo)

Clock detection is done based on a 5-bit counter running at the crystal clock frequency. The implementation detects clocks between 1 MHz and 200 MHz. It will take up to 2 µs from when the clock is removed until the interrupt condition is generated. A block diagram of the clock detection circuit is shown in Figure 6.

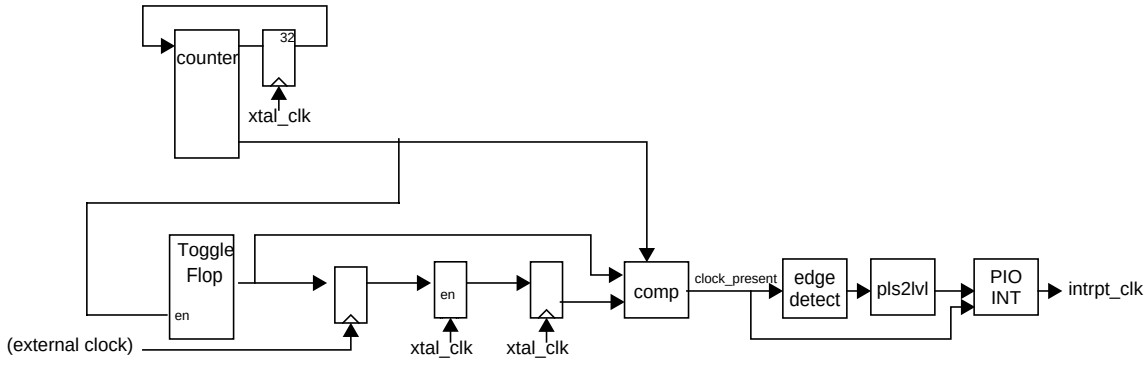


Figure 6: Clock Detection Circuit

An interrupt is generated whenever the signal 'clock present' changes status. Therefore an interrupt is generated if a clock changes from 'present' to 'non-present' OR from 'non-present to 'present'. The interrupt registers are implemented using the standard peripheral interrupt module and can thus be enabled/cleared/set by software.

In the PNX15xx Series all of the above clocks can also be generated internally. In this case the clock detection circuit can still be enabled. If the internal source is changed then the clock detection circuit will detect the period of time that there is not a clock. At this time the logic updates the interrupt status register and asserts an interrupt if the interrupt is enabled. The interrupts are by default disabled and should remain that way as long as the clock is generated internally. If in the course of time the output clock is changed to an input the interrupt status register needs to be cleared before the interrupts are enabled.

2.10 VDO Clocks

The two VDO out clocks, VDO_CLK1 and VDO_CLK2, have several operating modes. A brief explanation of these modes is included in this section. Each clock has three possible modes, input, separate output, and feedback mode. In input mode an external clock is driving these clocks (hence driving QVCP/LCD and FGPO). In separate output mode the clock module drives both the clocks going to the IP (QVCP/LCD and FGPO) and to its related output clock VDO_CLK1 and VDO_CLK2. In this case the source of the clock is the same, but the paths are totally separate. The third mode is feedback mode. In feedback mode the clock module drives the output clock, VDO_CLK1 and VDO_CLK2. This clock is then feedback through the pad to the clock module. Then it goes on to the IP (QVCP/LCD and FGPO). Diagrams of these clocks can be found in Figure 17 on page 5-28 and Figure 18 on page 5-28.

To select between output and input mode a bit is provided in each of the configuration registers for qvcp and fgpo. Writing to the qvcp_output_enable bit will change the direction of the qvcp clock. Writing to the fgpo_output_enable bit will change the direction of the fgpo clock. The output mode (separate or feedback) for the qvcp is selected by the qvcp_output_select bit. The fgpo_output_select bit selects the mode (separate or feedback) for the fgpo clock.

Both VDO clocks can also be programmed to have an inverted clock. There are two possible ways to invert the clock. If the invert clock bit is set then the inverted clock goes to the IP and the non inverted clock goes to the clock outputs. The qvcp clock is inverted by setting the invert_qvcp_clock bit in the qvcp configuration register. The fgpo clock is inverted by setting the invert_fgpo_clock bit in the fgpo configuration register. Also in output mode the qvcp source clock can be inverted by setting the sel_clk_qvcp bit to ‘10’. The fgpo source clock can also be inverted by setting the sel_clk_fgpo bit to ‘10’. By doing this the clock is inverted to both the internal and external version of the clock. In input mode the clock coming into the chip is inverted before being sent to the IP. In qvcp this is done by again writing to the invert_qvcp_clock bit. In fgpo the invert_fgpo_clock bit can also be set to invert the clock to the IP. In input mode the sel_clk_qvcp does not get used.

For both clocks they come out of reset in a quasi-input/output mode. The pad is set to be an input and the IP is being driven by the crystal clock (XTAL_IN) and not the input clock (if any). This is to allow the IP to reset if there isn’t an input clock as well as

protecting an input clock from contention by having the pad set to an input (in the case of an input clock). In both cases a write to each control register is necessary to properly put the clock into an input or output configuration (otherwise the logic will remain in the quasi-input/output mode).

As indicated above VDO_CLK1 can either be QVCP or LCD. After reset the clocks are in the above mentioned quasi-input/output mode. If it is to be LCD then the qvcp_out control register must be programmed to “separate” output mode. If the LCD only bit (bit 31 in the LCD_SETUP MMIO register) is set then the output select bit in the qvcp_out control register cannot be written to a ‘1’ (feedback mode). The LCD mode register can only be written to once and then only to disable LCD mode. If this is done then the output select bit can be programmed to any value.

2.11 GPIO Clocks

The following sections present the sequence of actions required to enable clocks on the GPIO[12:14,6:4’ pins.

2.11.1 Setting GPIO[14:12]/GCLOCK[2:0] as Clock Outputs

- Set gpio pin to gpio mode 2 using GPIO_MODE_0_15 (Table 7 on page 8-24)
- Set gpio pin to output a 0 using GPIO_MASK_IOD_0_15 (Table 8 on page 8-26)
- Set dds frequency using DDSx_CTL (Table 11 on page 5-34)
- Enable dds output to clk_gpio_y using CLK_GPIO_y_CTL (Table 11 on page 5-34)
- Enable clk_gpio_y to pin using DDS_OUT_SEL (Table 16 on page 8-36)

2.11.2 GPIO[6:4]/CLOCK[6:4] as Clock Outputs

- Set gpio pin to gpio mode 2 using GPIO_MODE_0_15 (Table 7 on page 8-24)
- Set gpio pin to output a 0 using GPIO_MASK_IOD_0_15 (Table 8 on page 8-26)
- Set dds frequency using DDSx_CTL (Table 11 on page 5-34)
- Enable dds output to clk_gpio_y using CLK_GPIO_y_CTL (Table 11 on page 5-34)GPIO_EV_x.
- Set GPIO_EV_x.EN_DDS_SOURCE = 1 and GPIO_EV_x.CLOCK_SEL = 4 for GPIO[4], 5 for GPIO[5] and 6 for GPIO[6] (Table 10 on page 8-27)

2.12 Clock Block Diagrams

The following sections present the block diagrams of the different clocks generated by the Clock module.

2.12.1 TM3260, DDR and QVCP clocks

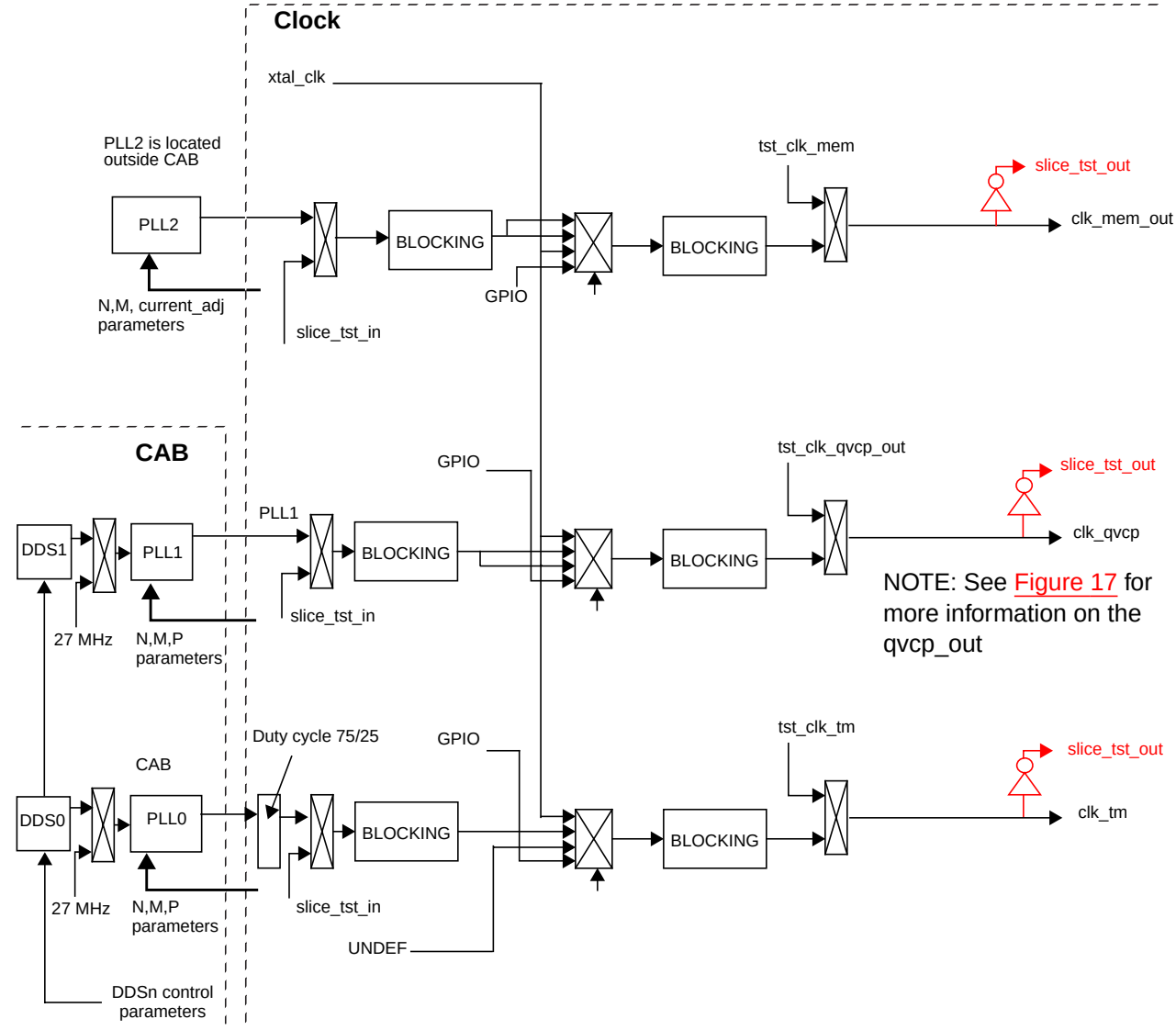


Figure 7: TM3260, DDR and QVCP clocks

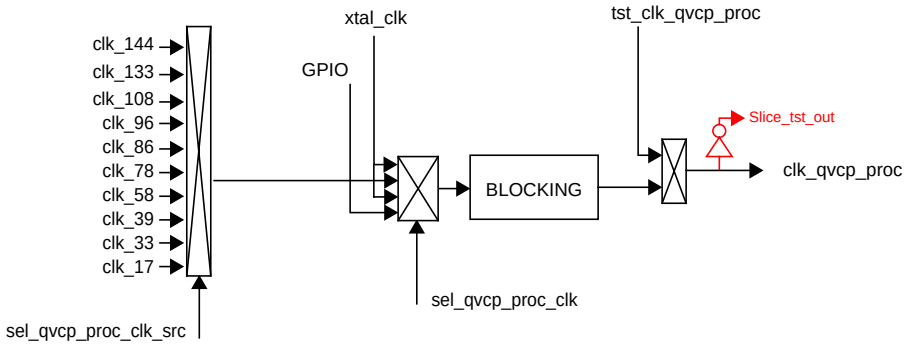


Figure 8: QVCP_PROC Clock

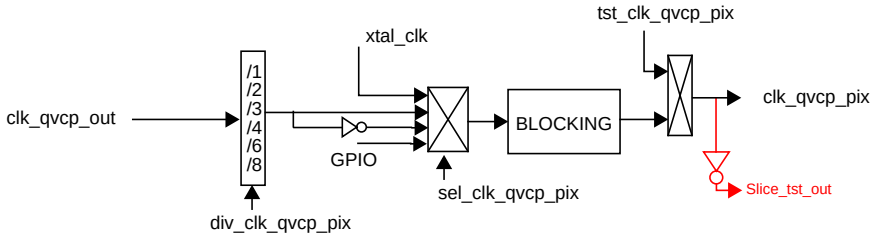


Figure 9: QVCP_PIX Clock

clk_qvcp_out generation is presented in Figure 17. It is important to notice that the clock used for clk_qvcp_out can be the inverted version of the clock present in the VDO_CLK1 pin. This allows the QVCP block to output data on the falling edge instead of the default positive edge. This feature may also be used to translate the AC timing characteristics that are computed with respect to the VDO_CLK1 positive edge.

2.12.2 Clock Dividers

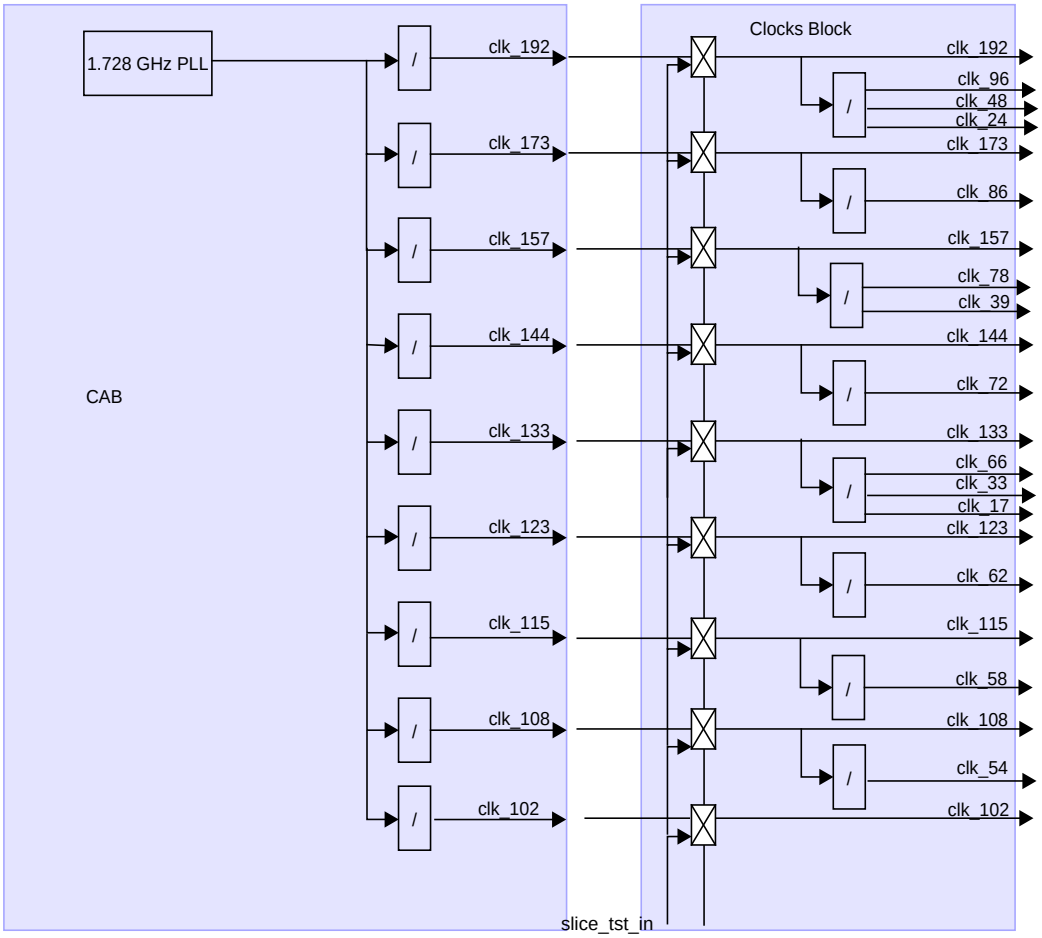


Figure 10: Clock Dividers

2.12.3 Internal PNX15xx Series Clock from Dividers

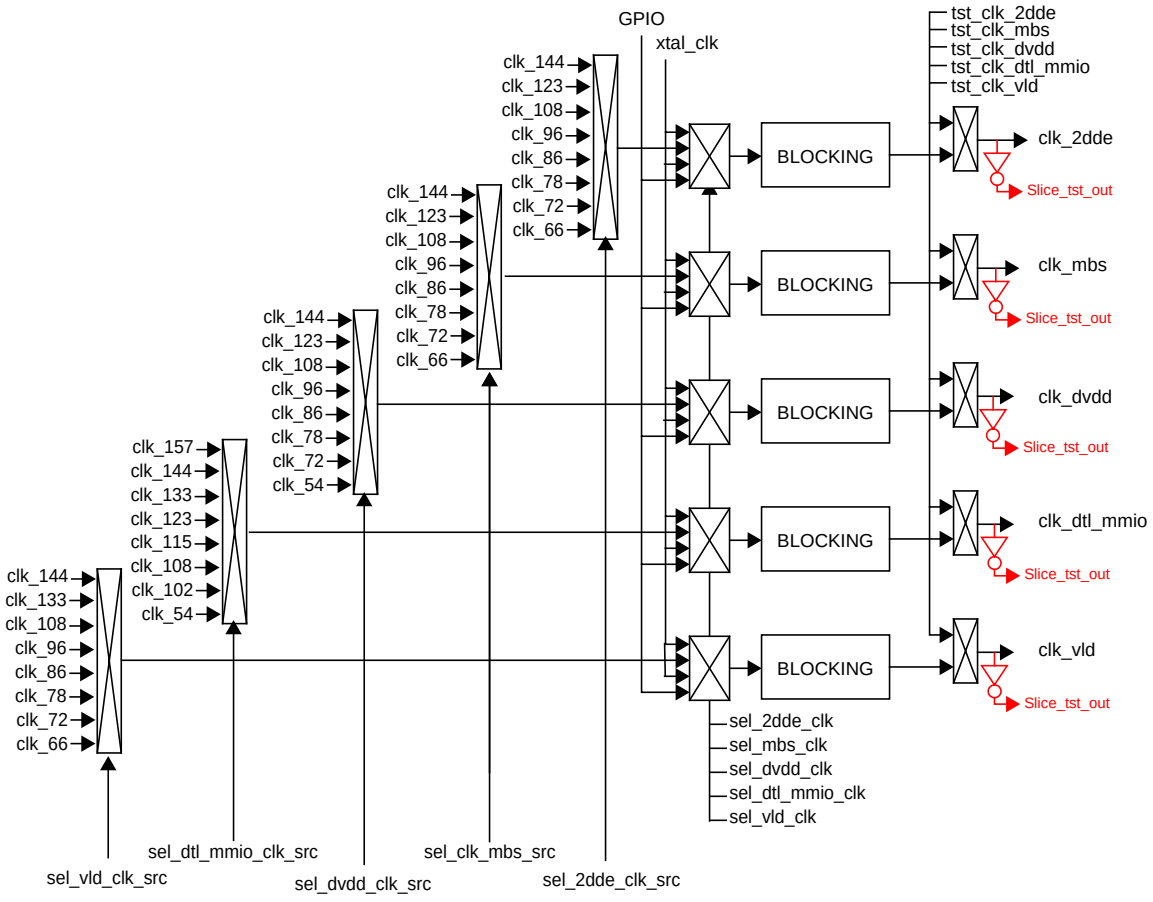


Figure 11: Internal PNX15xx Series Clock from Dividers

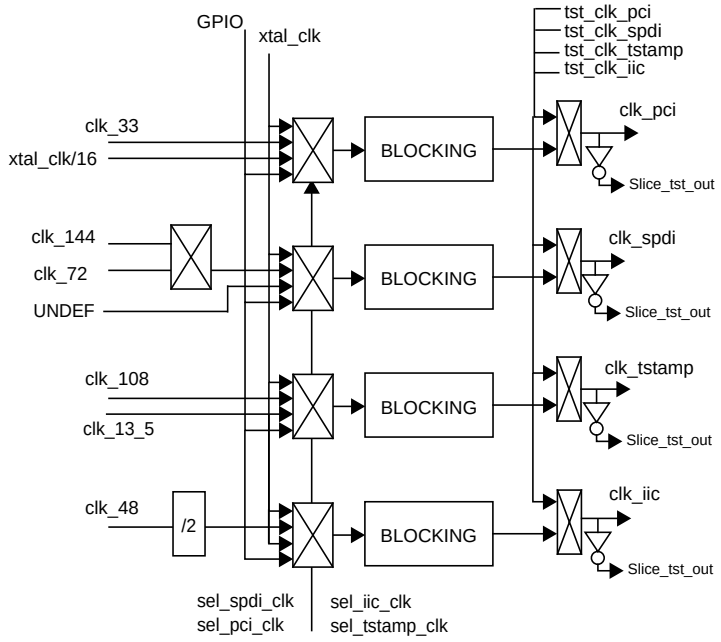


Figure 12: Internal PNX15xx Series Clock from Dividers: PCI, SPDI, LCD and I²C

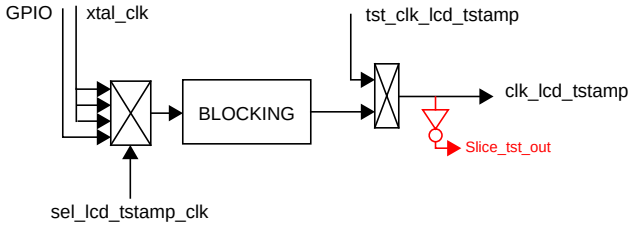


Figure 13: Internal PNX15xx Series Clock from Dividers: LCD Timestamp

2.12.4 GPIO Clocks

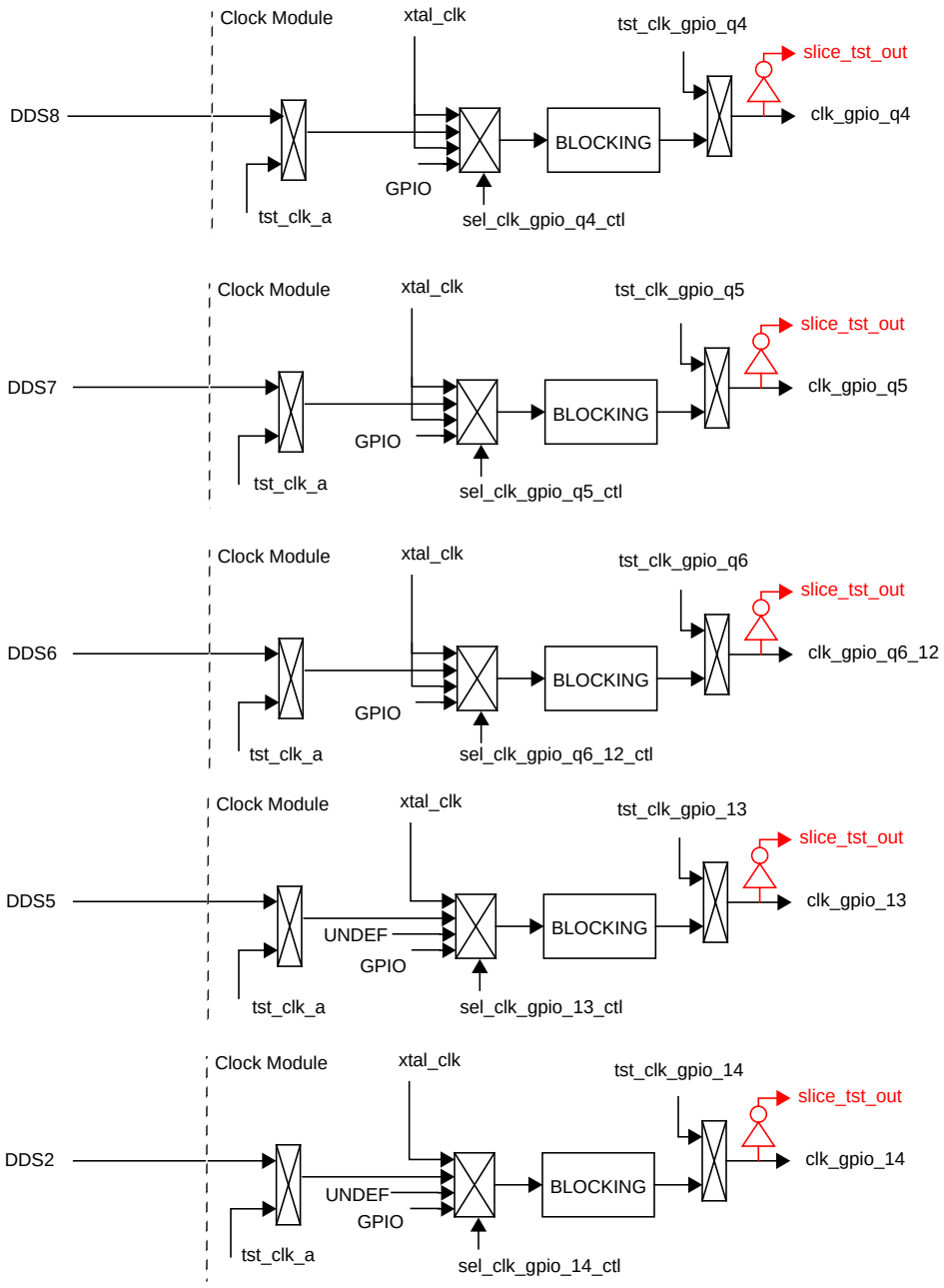


Figure 14: GPIO Clocks

2.12.5 External Clocks

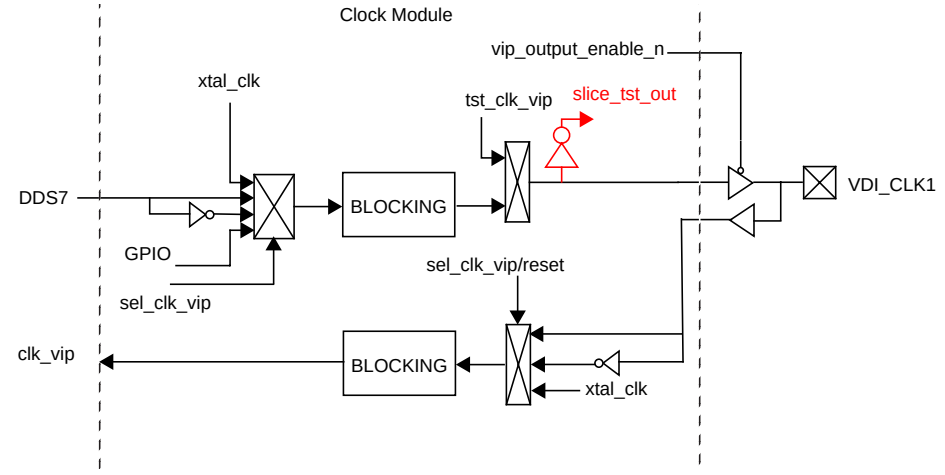


Figure 15: VDI_CLK1 Block Diagram

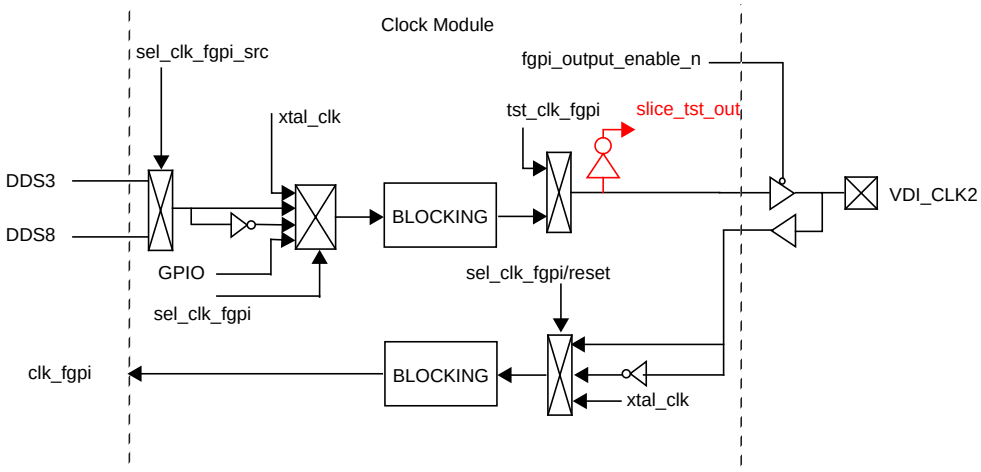


Figure 16: VDI_CLK2 Block Diagram

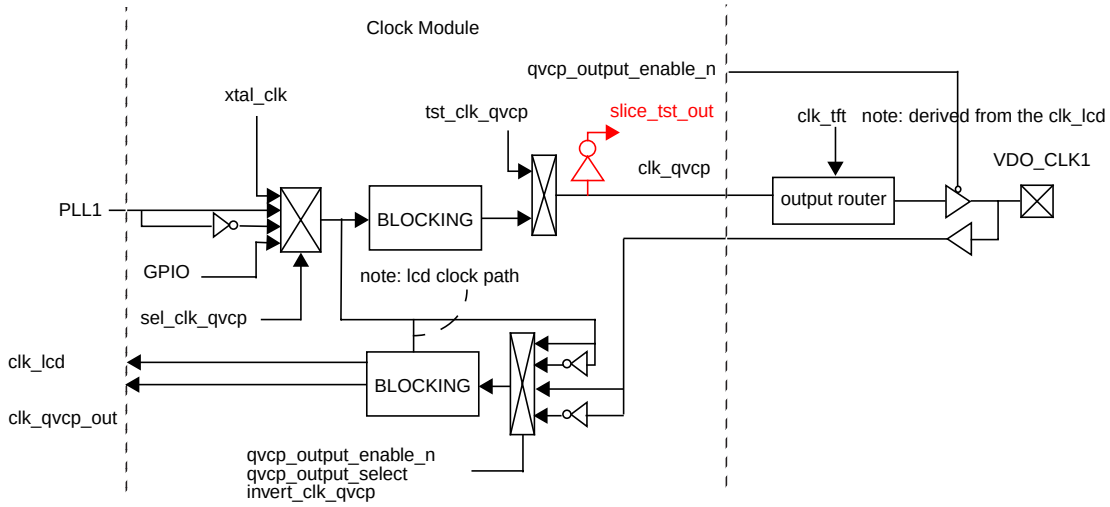


Figure 17: VDO_CLK1 Block Diagram

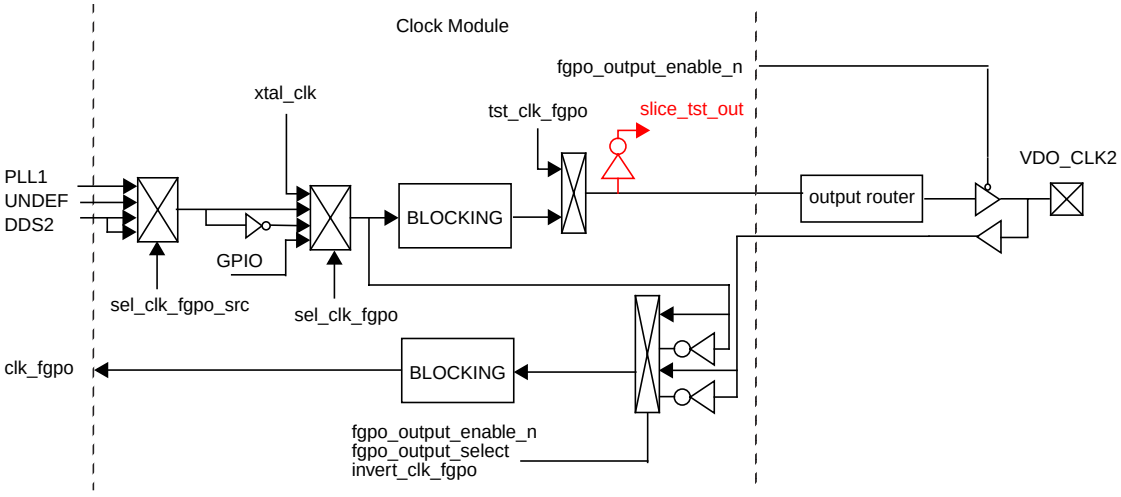


Figure 18: VDO_CLK2 Block Diagram

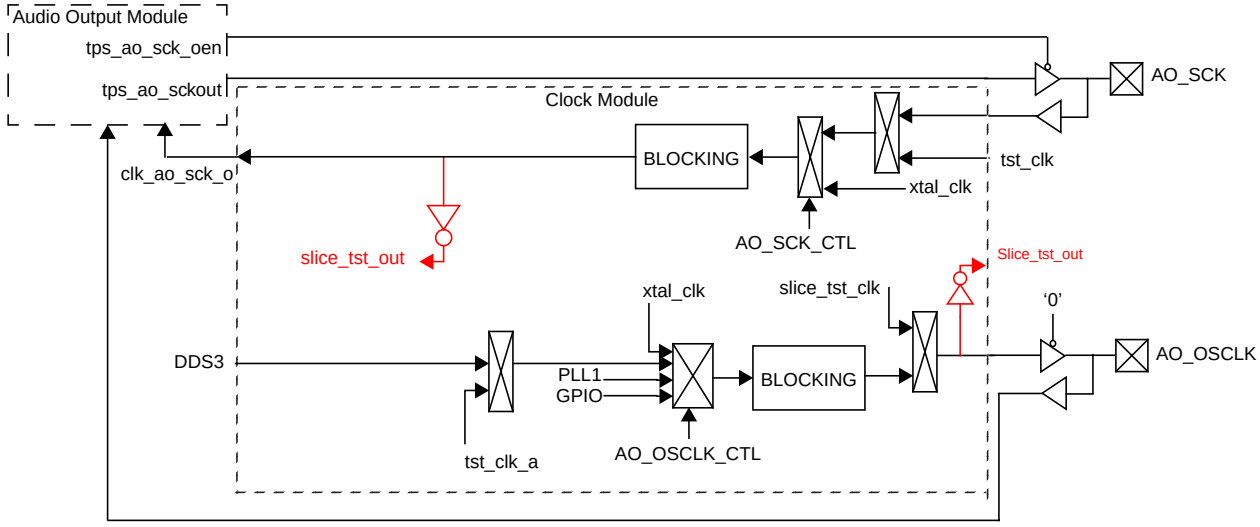


Figure 19: AO Clocks

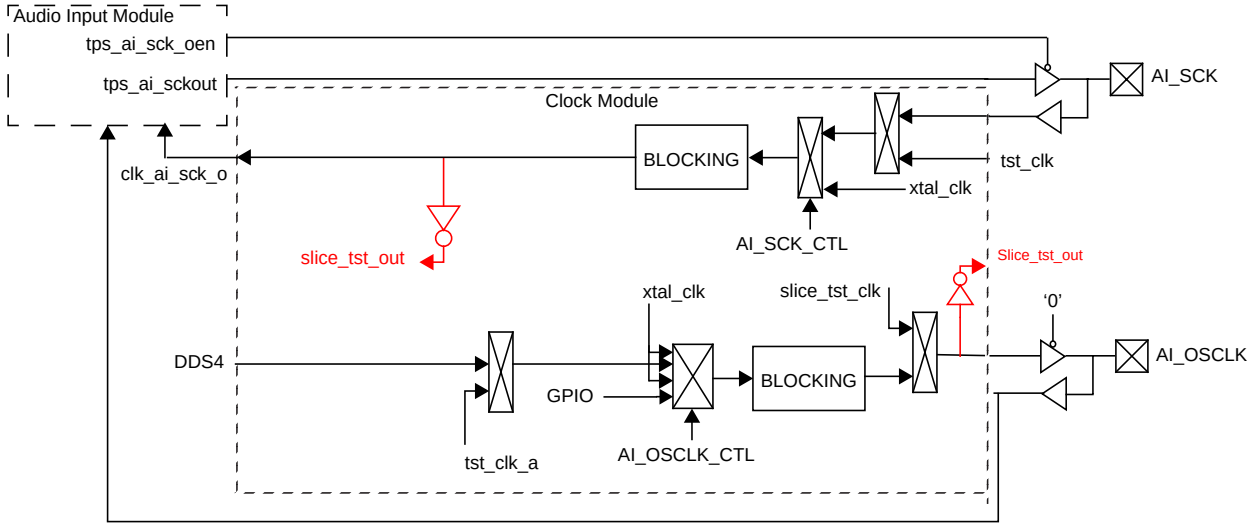


Figure 20: AI Clocks

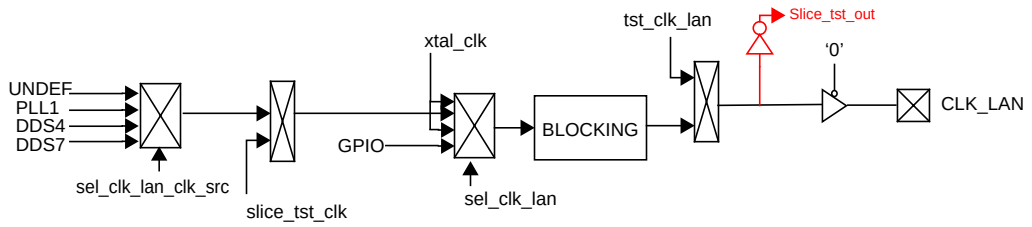


Figure 21: PHY LAN Clock Block Diagram

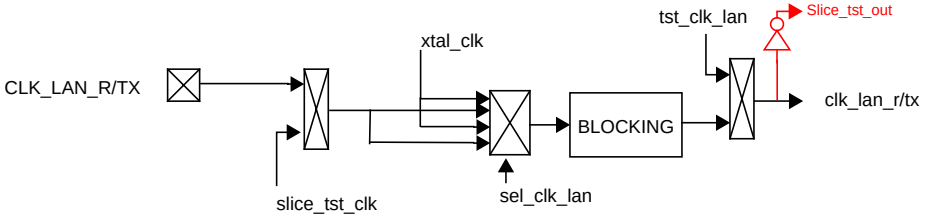


Figure 22: Receive and Transmit LAN Clocks

2.12.6 SPDO

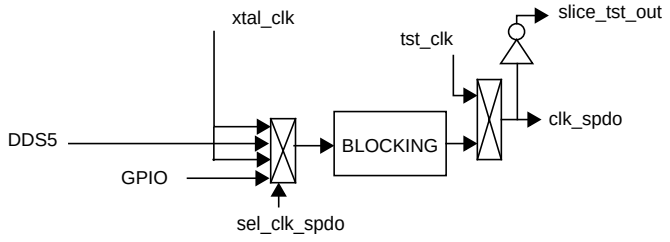


Figure 23: SPDO Clock

3. Registers Definition

3.1 Registers Summary

Table 10: Registers Summar

Offset	Name	Description
0x04,7000	PLL0_CTL	PLL0 Control Register
0x04,7004	PLL1_CTL	PLL1 Control Register
0x04,7008	PLL2_CTL	PLL2 Control Register
0x04,700C	PLL1_7_CTL	PLL 1.728 GHz Control Register
0x04,7010	DDS0_CTL	DDS0: frequency control
0x04,7014	DDS1_CTL	DDS1: frequency control
0x04,718	DDS2_CTL	DDS2: frequency control
0x04,701C	DDS3_CTL	DDS3: frequency control
0x04,7020	DDS4_CTL	DDS4: frequency control
0x04,7024	DDS5_CTL	DDS5: frequency control
0x04,7028	DDS6_CTL	DDS6: frequency control
0x04,702C	DDS7_CTL	DDS7: frequency control
0x04,7030	DDS8_CTL	DDS8: frequency control
0x04,7034	CAB_DIV_PD	CAB Clocks divider powerdown signals
0x04,7038- 0x04,70FC	RESERVED	RESERVED
0x04,7100	CLK_TM_CTL	TM3260 clock control
0x04,7104	CLK_MEM_CTL	DDR Memory clock control
0x04,7108	CLK_2D2_CTL	2D Drawing engine clock control
0x04,710C	CLK_PCI_CTL	PCI Clock control
0x04,7110	CLK_MBS_CTL	MBS Clock control
0x04,7114	CLK_TSTAMP_CTL	Time Stamp Clock control
0x04,7118	CLK_LAN_CTL	Ethernet Clock control
0x04,711C	CLK_LAN_RX_CTL	Ethernet RX Clock control
0x04,7120	CLK_LAN_TX_CTL	Ethernet TX Clock control

Table 10: Registers Summar

Offset	Name	Description
0x04,7124	CLK_IIC_CTL	I^2C clock control
0x04,7128	CLK_DVDD_CTL	DVDD clock control
0x04,712C	CLK_MMIO_CTL	MMIO Clock control, a.k.a. DCS clock
0x04,7130- 0x04,71FC	RESERVED	RESERVED
0x04,7200	CLK_QVCP_OUT_CTL	QVCP clock output control
0x04,7204	CLK_QVCP_PIX_CTL	QVCP PIX clock control
0x04,7208	CLK_QVCP_PROC_CTL	QVCP PROC Clock control
0x04,720C	CLK_LCD_TSTAMP_CTL	LCD Timestamp clock control
0x04,7210	CLK_VIP_CTL	Video Input Processor clock control
0x04,7214	CLK_VLD_CTL	VLD clock control
0x04,7218- 0x04,72FC	RESERVED	RESERVED
0x04,7300	AI_OSCLK_CTL	Audio in over sampling clock control
0x04,7304	AI_SCK_CTL	Audio In sampling Clock control
0x04,7308	AO_OSCLK_CTL	Audio out over sampling clock control
0x04,730c	AO_SCK_CTL	Audio out sampling clock control
0x04,7310	CLK_SPDO_CTL	SPDO clock control
0x04,7314	CLK_SPDI_CTL	SPDI clock control
0x04,7318- 0x04,73FC	RESERVED	RESERVED
0x04,7400	GPIO_CLK_Q4_CTL	GPIO clock to FIFO and pin 4 control
0x04,7404	GPIO_CLK_Q5_CTL	GPIO clock to FIFO and pin 5 control
0x04,7408	GPIO_CLK_Q6_12_CTL	GPIO clock to FIFO and pin 6/12 control
0x04,740C	GPIO_CLK_13_CTL	GPIO clock to pin 13
0x04,7410	GPIO_CLK_14_CTL	GPIO clock to pin 14
0x04,7414	CLK_FGPO_CTL	FGPO clock control
0x04,7418	CLK_FGPI_CTL	FGPI clock control
0x04,741C- 0x04,74FC	RESERVED	RESERVED
0x04,7500	CLK_STRETCHER_CTL	Clock stretcher count register
0x04,7504	CLK_WAKEUP_CTL	Wake-up count register
0x04,7508	CLK_FREQ_CTL	PLL/DDS frequency count register
0x04,750C	CLK_RESULT_CTL	PLL/DDS frequency count result register
0x04,7510	ALIGNER_ADJUST	RESERVED
0x04,7514- 0x04,7FDC	RESERVED	RESERVED
0x04,7FE0	INTERRUPT_STATUS	Status of Clock Detection interrupts
0x04,7FE4	INTERRUPT_ENABLE	Enable Clock Detection interrupts
0x04,7FE8	INTERRUPT_CLEAR	Clear Clock Detection interrupts

12NC 9397 750 14321

Table 10: Registers Summar

Offset	Name	Description
0x04,7FEC	INTERRUPT_SET	Set Clock Detection interrupts
0x04,7FF0- 0x04,7FF8	RESERVED	RESERVED
0x04,7FFC	MODULE_ID	Module Identification and revision information

3.2 Registers Description

Table 11: CLOCK MODULE REGISTERS

Bit	Symbol	Access	Value	Description
PLL Registers				
Offset 0x04,7000	***PLL0_CTL***			
Reset values set for expected frequencies for faster boot-up, shorter boot code.				
31:30	Reserved	R/W	-	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
29	Turn Off Acknowledge	R	-	Indicates that during a frequency change that the clock has been driven low.
28	PLL Lock	R	-	A '1' indicates that the PLL is locked
27:24	pll0_adj	R/W	0	Current adjustment. Section 2.2.1 on page 5-8.
23:21	Reserved	R/W	-	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
20:12	pll0_n	R/W	0x4A	9-bit N parameter to PLL0
11:10	Reserved	R/W	-	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
9:4	pll0_m	R/W	0x5	6-bit M parameter to PLL0. Section 2.2.1 on page 5-8.
3:2	pll0_p	R/W	0	2-bit P parameter to PLL0. Section 2.2.1 on page 5-8.
1	pll0_pd	R/W	0	1: powerdown PLL0
0	pll0_bp	R/W	1	0: Do not bypass the DDS 1: Bypass the DDS and use the xtal (27 MHz). **Normal Operating mode.**
Offset 0x04,7004	***PLL1_CTL***			
Reset values set for expected frequencies for faster boot-up, shorter boot code.				
31:30	Reserved	R/W	-	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
29	Turn Off Acknowledge	R	-	Indicates that during a frequency change that the clock has been driven low.
28	PLL Lock	R	-	A '1' indicates that the PLL is locked
27:24	pll1_adj	R/W	4	Current adjustment. Section 2.2.1 on page 5-8.
23:21	Reserved	R/W	-	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
20:12	pll1_n	R/W	0x22	9-bit N parameter to PLL1. Section 2.2.1 on page 5-8.
11:10	Reserved	R/W	-	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
9:4	pll1_m	R/W	6	6-bit M parameter to PLL1. Section 2.2.1 on page 5-8.
3:2	pll1_p	R/W	2	2-bit P parameter to PLL1. Section 2.2.1 on page 5-8.
1	pll1_pd	R/W	0	1: powerdown PLL1
0	pll1_bp	R/W	1	0: Do not bypass the DDS. 1: Bypass the DDS and use the xtal (27 MHz)

Table 11: CLOCK MODULE REGISTERS *…Continued*

Bit	Symbol	Access	Value	Description
Offset 0x04,7008	***PLL2_CTL***			
Reset values set for expected frequencies for faster boot-up, shorter boot code.				
31:30	Reserved	R/W	-	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
29	Turn Off Acknowledge	R	-	Indicates that during a frequency change that the clock has been driven low.
28	PLL Lock	R	-	A one indicates that the PLL is locked
27:24	pll2_adj	R/W	0	Current adjustment. Section 2.2.1 on page 5-8.
23:21	Reserved	R/W	-	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
20:12	pll2_n	R/W	0x2E	9-bit N parameter to PLL2. Section 2.2.1 on page 5-8.
11:10	Reserved	R/W	-	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
9:4	pll2_m	R/W	0x5	6-bit M parameter to PLL2. Section 2.2.1 on page 5-8.
3:2	Reserved	R/W	-	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
1	pll2_pd	R/W	0	1: powerdown PLL2
0	Reserved	R/W	-	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
Offset 0x04,700C	***PLL1_7GHZ_CTL***			
31:3	Reserved	R/W	-	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
2	pll1_7ghz_pd	R/W	0	1: powerdown PLL1_7GHZ
1:0	Reserved	R/W	-	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
DDS Registers				
Offset 0x04,7010	***DDS0_CTL***			
31	Enable	R/W	0	1: Enables the DDS. The input of the DDS is then the 1.7GHz clock. 0: Test mode. Do not use.
30:0	dds0_ctl[30 :0]	R/W	0x07684 bd0	31-bit DDS0 control (default = 50 MHz)
Offset 0x04,7014	***DDS1_CTL***			
31	Enable	R/W	0	1: Enables the DDS. The input of the DDS is then the 1.7GHz clock. 0: Test mode. Do not use.
30:0	dds1_ctl[30:0]	R/W	0x04000 000	31-bit DDS1 control (default = 27 MHz)
Offset 0x04,7018	***DDS2_CTL***			
31	Enable	R/W	0	1: Enables the DDS. The input of the DDS is then the 1.7GHz clock. 0: Test mode. Do not use.
30:0	dds2_ctl[30:0]	R/W	0x04000 000	31-bit DDS2 control (default = 27 MHz)

Table 11: CLOCK MODULE REGISTERS *…Continued*

Bit	Symbol	Access	Value	Description
Offset 0x04,701C	***DDS3_CTL***			
31	Enable	R/W	0	1: Enables the DDS. The input of the DDS is then the 1.7GHz clock. 0: Test mode. Do not use.
30:0	dds3_ctl[30:0]	R/W	0x02F684C0	31-bit DDS3 control (default = 20 MHz)
Offset 0x04,7020	***DD4_CTL***			
31	Enable	R/W	0	1: Enables the DDS. The input of the DDS is then the 1.7GHz clock. 0: Test mode. Do not use.
30:0	dds4_ctl[30:0]	R/W	0x02F684C0	31-bit DDS4 control (default = 20 MHz)
Offset 0x04,7024	***DDS5_CTL***			
31	Enable	R/W	0	1: Enables the DDS. The input of the DDS is then the 1.7GHz clock. 0: Test mode. Do not use.
30:0	dds5_ctl[30:0]	R/W	0x00E90452	31-bit DDS5 control (default = 128*48kHz = 6.14 MHz)
Offset 0x04,7028	***DDS6_CTL***			
31	Enable	R/W	0	1: Enables the DDS. The input of the DDS is then the 1.7GHz clock. 0: Test mode. Do not use.
30:0	dds6_ctl[30:0]	R/W	0x04000000	31-bit DDS6 control (default = 27 MHz)
Offset 0x04,702C	***DDS7_CTL***			
31	Enable	R/W	0	1: Enables the DDS. The input of the DDS is then the 1.7GHz clock. 0: Test mode. Do not use.
30:0	dds7_ctl[30:0]	R/W	0x04000000	31-bit DDS7 control (default = 27 MHz)
Offset 0x04,7030	***DDS8_CTL***			
31	Enable	R/W	0	1: Enables the DDS. The input of the DDS is then the 1.7GHz clock. 0: Test mode. Do not use.
30:0	dds8_ctl[30:0]	R/W	0x04000000	31-bit DDS8 control (default = 27 MHz)
Divider Registers: For register 34h power down appropriate clocks before setting these bits				
Offset 0x04,7034	***CAB_DIVIDER_CTL***			
31:8	Reserved	R/W		To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
8	pd_192	R/W	0	Power down 192 MHz divider in the CAB block.
7	pd_173	R/W	0	Power down 173 MHz divider in the CAB block.
6	pd_157	R/W	0	Power down 157 MHz divider in the CAB block.
5	pd_144	R/W	0	Power down 144 MHz divider in the CAB block.
4	pd_133	R/W	0	Power down 133 MHz divider in the CAB block.
3	pd_123	R/W	0	Power down 123 MHz divider in the CAB block.
2	pd_115	R/W	0	Power down 115 MHz divider in the CAB block.

Table 11: CLOCK MODULE REGISTERS *...Continued*

Bit	Symbol	Access	Value	Description
1	pd_108	R/W	0	Power down 108 MHz divider in the CAB block.
0	pd_102	R/W	0	Power down 102 MHz divider in the CAB block.
Offset 0x04,7038-0x04,70FCReserved				
Module Clocks				
Offset 0x04,7100	***CLK_TM_CTL***			
31:6	Reserved	R/W	-	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
5	turn_off_ack	R	0	0 - Indicates if the enabled clock is running 1 - Indicates that the clock is being blocked during a frequency change to avoid glitches
4	tm_stretch_n	R/W	0	0 - turns on the 75/25 duty cycle adjust circuit 1 - turns off the 75/25 duty cycle adjust circuit **MUST BE SET TO '1' for normal operation.**
3	sel_pwrdwn_clk_mmio	W	0	This bit allows the TM3260 to turn off the MMIO clock simultaneously with the TM3260 clock. This mechanism allows to go into deep sleep mode and allows to keep the capability to wake-up from this deep sleep mode (Section 2.8.1 on page 5-17). When deep sleep mode is requested by TM3260, it must turn off its own clock, clk_tm, by setting en_clk_tm to '0' and sel_pwrdwn_clk_mmio to '1'. Writing to a '0' to en_clk_tm without setting sel_pwrdwn_clk_mmio to '1' shuts down TM3260 clock forever (unless a host writes back a '1' to 'en_clk_tm' or a system reset occurs). Therefore, the ONLY use of sel_pwrdwn_clk_mmio is to set it to '1' at the same time en_clk_tm is set to '0'. The TM3260 must run a waiting loop of 10 27 MHz cycles after the write to CLK_TM_CTL is done since the clk_tm is not immediately turned off. Upon wake-up, en_clk_tm and sel_pwrdwn_clk_mmio get their initial reset value and TM3260 resumes from where it stopped. Maximum power saving is achieved by turning off the PLL0 and therefore switch to the 27 MHz xtal_clk clock before requesting a deep sleep mode. Similarly the other clocks of the system must be turned off separately if maximum power saving needs to be achieved. This may include the DDR clock. Upon wake-up, if a PLL has been turned off, a minimum of 100 µs is required to lock it.
2:1	sel_clk_tm	R/W	0	00: clk_tm = 27 MHz xtal_clk 01: clk_tm = tm_stretch_n (output of the duty cycle stretcher) 10: clk_tm = UNDEF 11: clk_tm = AI_WS
0	en_clk_tm	R/W	1	1: enable clk_tm
Offset 0x04,7104	***CLK_MEM_CTL***			
31:4	Reserved	R/W	-	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.

Table 11: CLOCK MODULE REGISTERS *…Continued*

Bit	Symbol	Access	Value	Description
3	turn_off_ack	R	0	0 - Indicates if the enabled clock is running 1 - Indicates that the clock is being blocked during a frequency change to avoid glitches
2:1	sel_clk_mem	R/W	00	00: clk_mem = PLL2 01: clk_mem = PLL2 10: clk_mem = 27 MHz xtal_clk 11: clk_mem = GPIO[7]
0	en_clk_mem	R/W	1	1: enable clk_mem
Offset 0x04,7108	***CLK_2DDE_CTL***			
31:7	Reserved	R/W	-	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
6	turn_off_ack	R	0	0 - Indicates if the enabled clock is running 1 - Indicates that the clock is being blocked during a frequency change to avoid glitches
5:3	sel_clk_2dde_src	R/W	111	000: clk_2dde_src = clk_144 001: clk_2dde_src = clk_123 010: clk_2dde_src = clk_108 011: clk_2dde_src = clk_96 100: clk_2dde_src = clk_86 101: clk_2dde_src = clk_78 110: clk_2dde_src = clk_72 111: clk_2dde_src = clk_66
2:1	sel_clk_2dde	R/W	00	00: clk_2dde = 27 MHz xtal_clk 01: clk_2dde = clk_2d2_src 10: clk_2dde = 27 MHz xtal_clk 11: clk_2dde = AI_SD[1]
0	en_clk_2dde	R/W	1	1: enable clk_2dde
Offset 0x04,710C	***CLK_PCI_CTL***			
31:4	Reserved	R/W	-	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
3	turn_off_ack	R	0	0 - Indicates if the enabled clock is running 1 - Indicates that the clock is being blocked during a frequency change to avoid glitches
2:1	sel_clk_pci	R/W	01	00: clk_pci = 27 MHz xtal_clk 01: clk_pci = clk_33 10: clk_pci = xtal_clk/16 = 1.68 MHz 11: clk_pci = AI_SD[2]
0	en_clk_pci	R/W	1	1: enable clk_pci
Offset 0x04,7110	***CLK_MBS_CTL***			
31:7	Reserved	R/W	-	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.

Table 11: CLOCK MODULE REGISTERS *...Continued*

Bit	Symbol	Access	Value	Description
6	turn_off_ack	R	0	0 - Indicates if the enabled clock is running 1 - Indicates that the clock is being blocked during a frequency change to avoid glitches
5:3	sel_clk_mbs_src	R/W	111	000: clk_mbs_src = clk_144 001: clk_mbs_src = clk_123 010: clk_mbs_src = clk_108 011: clk_mbs_src = clk_96 100: clk_mbs_src = clk_86 101: clk_mbs_src = clk_78 110: clk_mbs_src = clk_72 111: clk_mbs_src = clk_66
2:1	sel_clk_mbs	R/W	00	00: clk_mbs = 27 MHz xtal_clk 01: clk_mbs = clk_mbs_src 10: clk_mbs = 27 MHz xtal_clk 11: clk_mbs = AI_SD[3]
0	en_clk_mbs	R/W	1	1: enable clk_mbs
Offset 0x04,7114	***CLK_TSTAMP_CTL***			
31:4	Reserved	R/W	-	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
3	turn_off_ack	R	0	0 - Indicates if the enabled clock is running 1 - Indicates that the clock is being blocked during a frequency change to avoid glitches
2:1	sel_clk_tstamp	R/W	00	00: clk_tstamp = 27 MHz xtal_clk 01: clk_tstamp = Source clock (clk_108) 10: clk_tstamp = Second Clock (clk_13_5) 11: clk_tstamp = AO_WS
0	en_clk_tstamp	R/W	1	1: enable clk_tstamp
Offset 0x04,7118	***CLK_LAN_CTL***			
31:6	Reserved	R/W	-	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
5	turn_off_ack	R	0	0 - Indicates if the enabled clock is running 1 - Indicates that the clock is being blocked during a frequency change to avoid glitches
4:3	sel_lan_clk_src	R/W	00	00: clk_lan_src = UNDEF 01: clk_lan_src = PLL1 10: clk_lan_src = DDS4 11: clk_lan_src = DDS7

Table 11: CLOCK MODULE REGISTERS *…Continued*

Bit	Symbol	Access	Value	Description
2:1	sel_clk_lan	R/W	00	00: clk_lan = 27 MHz xtal_clk 01: clk_lan = clk_lan_src 10: clk_lan = 27 MHz xtal_clk 11: clk_lan = AO_SD[0]
0	en_clk_lan	R/W	1	1: enable clk_lan
Offset 0x04,711C	***CLK_LAN_RX_CTL***			
31:4	Reserved	R/W	-	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
3	turn_off_ack	R	0	0 - Indicates if the enabled clock is running 1 - Indicates that the clock is being blocked during a frequency change to avoid glitches
2:1	sel_clk_lan_rx	R/W	00	00: clk_lan_rx = 27 MHz xtal_clk 01: clk_lan_rx = CLK_LAN_RX pin 10: clk_lan_rx = 27 MHz xtal_clk 11: clk_lan_rx = CLK_LAN_RX pin
0	en_clk_lan_rx	R/W	1	1: enable clk_lan_rx
Offset 0x04,7120	***CLK_LAN_TX_CTL***			
31:4	Reserved	R/W	-	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
3	turn_off_ack	R	0	0 - Indicates if the enabled clock is running 1 - Indicates that the clock is being blocked during a frequency change to avoid glitches
2:1	sel_clk_lan_tx	R/W	00	00: clk_lan_tx = 27 MHz xtal_clk 01: clk_lan_tx = CLK_LAN_TX pin 10: clk_lan_tx = 27 MHz xtal_clk) 11: clk_lan_tx = CLK_LAN_TX pin
0	en_clk_lan_tx	R/W	1	1: enable clk_lan_tx
Offset 0x04,7124	***CLK_IIC_CTL***			
31:4	Reserved	R/W	-	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
3	turn_off_ack	R	0	0 - Indicates if the enabled clock is running 1 - Indicates that the clock is being blocked during a frequency change to avoid glitches
2:1	sel_clk_iic	R/W	00	00: clk_iic_tx = 27 MHz xtal_clk 01: clk_iic_tx = clk_24 10: clk_iic_tx = 27 MHz xtal_clk 11: clk_iic_tx = AO_SD[1]
0	en_clk_iic	R/W	1	1: enable clk_iic
Offset 0x04,7128	***CLK_DVDD_CTL***			
31:7	Reserved	R/W	-	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.

Table 11: CLOCK MODULE REGISTERS *...Continued*

Bit	Symbol	Access	Value	Description
6	turn_off_ack	R	0	0 - Indicates if the enabled clock is running 1 - Indicates that the clock is being blocked during a frequency change to avoid glitches
5:3	sel_clk_dvdd_src	R/W	111	000: clk_dvdd_src = clk_144 001: clk_dvdd_src = clk_123 010: clk_dvdd_src = clk_108 011: clk_dvdd_src = clk_96 100: clk_dvdd_src = clk_86 101: clk_dvdd_src = clk_78 110: clk_dvdd_src = clk_72 111: clk_dvdd_src = clk_54
2:1	sel_clk_dvdd	R/W	00	00: clk_dvdd = 27 MHz xtal_clk 01: clk_dvdd = clk_dvdd_src 10: clk_dvdd = 27 MHz xtal_clk 11: clk_dvdd = AO_SD[2]
0	en_clk_dvdd	R/W	1	1: enable clk_dvdd
Offset 0x04,712C	***CLK_DTL_MMIO_CTL***			
31:7	Reserved	R/W	-	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
6	turn_off_ack	R	0	0 - Indicates if the enabled clock is running 1 - Indicates that the clock is being blocked during a frequency change to avoid glitches
5:3	sel_clk_dtl_mmio_src	R/W	000	000: clk_dtl_mmio_src = clk_102 001: clk_dtl_mmio_src = clk_108 010: clk_dtl_mmio_src = clk_115 011: clk_dtl_mmio_src = clk_123 100: clk_dtl_mmio_src = clk_133 101: clk_dtl_mmio_src = clk_144 110: clk_dtl_mmio_src = clk_157 111: clk_dtl_mmio_src = clk_54
2:1	sel_clk_dtl_mmio	R/W	00	00: clk_dtl_mmio = 27 MHz xtal_clk 01: clk_dtl_mmio = clk_dtl_mmio_src 10: clk_dtl_mmio = 27 MHz xtal_clk 11: clk_dtl_mmio = AO_SD[3]
0	en_dtl_mmio	R/W	1	1: enable clk_dtl_mmio
Offset 0x04,7200	***CLK_QVCP_OUT_CTL***			
31:7	Reserved	R/W	-	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
6	turn_off_ack	R	0	0 - Indicates if the enabled clock is running 1 - Indicates that the clock is being blocked during a frequency change to avoid glitches

Table 11: CLOCK MODULE REGISTERS *…Continued*

Bit	Symbol	Access	Value	Description
5	Invert_qvcp_clock	R/W	0	Invert QVCP clock 0 : do not invert the clock 1: invert the clock only to the qvcp block and not to the pad.
4	qvcp_output_select	R/W	0	QVCP output select 0: Seperate output mode, The clock to the qvcp and to the pad share the same source, but have seperate paths. This mode is also the LCD only mode (see QVCP/LCD description). If the LCD only bit is set then this bit cannot be set to a '1' (feedback mode). 1: Feedback output mode, The clock is driven to the pad then is feedback to the clock block. It then goes through gating logic to the qvcp block.
3	qvcp_output_enable_n	R/W	1	QVCP output enable 0: output, the clock is generated internally 1: input, the clock is provided by an external source. Note: during and after reset the xtal clock is forced onto the qvcp clock. In order to actually allow the input clock to go to the qvcp this register must be written to. This also implies that writing qvcp_output_enable_n = 1 overrides a sel_clk_qvcp = 0.
2:1	sel_clk_qvcp	R/W	00	The following 4 settings are valid when qvcp_output_enable_n = 0. 00: clk_qvcp = 27 MHz xtal_clk (see qvcp_output_enable_n). 01: clk_qvcp = PLL1 10: clk_qvcp = $\overline{\text{PLL1}}$ 11: clk_qvcp = XIO_ACK The following 3 settings are valid when qvcp_output_enable_n = 1 (The input mode). 01: clk_qvcp_out = VDO_CLK1
0	en_clk_qvcp	R/W	1	1: enable clk_qvcp
Offset 0x04,7204	***CLK_QVCP_PIX_CTL***			
31:7	Reserved	R/W	-	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
6	turn_off_ack	R	0	0 - Indicates if the enabled clock is running 1 - Indicates that the clock is being blocked during a frequency change to avoid glitches
5:3	div_clk_qvcp_pix	R/W	001	000: clk_qvcp_pix_src = qvcp_clk_out clock divided by 1 001: clk_qvcp_pix_src = qvcp_clk_out clock divided by 2 010: clk_qvcp_pix_src = qvcp_clk_out clock divided by 3 011: clk_qvcp_pix_src = qvcp_clk_out clock divided by 4 100: clk_qvcp_pix_src = qvcp_clk_out clock divided by 6 101: clk_qvcp_pix_src = qvcp_clk_out clock divided by 8 (refer to Figure 17 for the qvcp_clk_out)
2:1	sel_clk_qvcp_pix	R/W	00	00: clk_qvcp_pix = 27 MHz xtal_clk 01: clk_qvcp_pix = clk_qvcp_pix_src 10: clk_qvcp_pix = $\overline{\text{clk\_qvcp\_pix\_src}}$ 11: clk_qvcp_pix = XIO_D[8]

Table 11: CLOCK MODULE REGISTERS *...Continued*

Bit	Symbol	Access	Value	Description
0	en_clk_qvcp_pix	R/W	1	1: enable clk_qvcp_pix
Offset 0x04,7208	***CLK_QVCP_PROC_CTL***			
31:8	Reserved	R/W	-	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
7	turn_off_ack	R	0	0 - Indicates if the enabled clock is running 1 - Indicates that the clock is being blocked during a frequency change to avoid glitches
6:3	sel_clk_qvcp_proc_src	R/W	0111	0000: clk_qvcp_proc_src = clk_144 0001: clk_qvcp_proc_src = clk_133 0010: clk_qvcp_proc_src = clk_108 0011: clk_qvcp_proc_src = clk_96 0100: clk_qvcp_proc_src = clk_86 0101: clk_qvcp_proc_src = clk_78 0110: clk_qvcp_proc_src = clk_58 0111: clk_qvcp_proc_src = clk_39 1000: clk_qvcp_proc_src = clk_33 1001: clk_qvcp_proc_src = clk_17 Maximum speed supported is 96 MHz. Other higher speeds are reserved for future use.
2:1	sel_clk_dtl_mmio	R/W	00	00: clk_qvcp_proc = 27 MHz xtal_clk 01: clk_qvcp_proc = clk_qvcp_proc_src 10: clk_qvcp_proc = 27 MHz xtal_clk 11: clk_qvcp_proc = XIO_D[9]
0	en_clk_proc	R/W	1	1: enable clk_qvcp_proc
Offset 0x04,720C	***CLK_LCD_TIMESTAMP_CTL***			
31:4	Reserved	R/W	-	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
3	turn_off_ack	R	0	0 - Indicates if the enabled clock is running 1 - Indicates that the clock is being blocked during a frequency change to avoid glitches
2:1	sel_clk_lcd_timestamp	R/W	00	00: clk_lcd_timestamp = 27 MHz xtal_clk 01: clk_lcd_timestamp = 27 MHz xtal_clk 10: clk_lcd_timestamp = 27 MHz xtal_clk 11: clk_lcd_timestamp = XIO_D[10]
0	en_clk_lcd_timestamp	R/W	1	1: enable clk_lcd_timestamp
Offset 0x04,7210	***CLK_VIP_CTL***			
31:5	Reserved	R/W	-	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
4	turn_off_ack	R	0	0 - Indicates if the enabled clock is running 1 - Indicates that the clock is being blocked during a frequency change to avoid glitches

Table 11: CLOCK MODULE REGISTERS *…Continued*

Bit	Symbol	Access	Value	Description
3	vip_output_enable_n	R/W	1	VIP output enable 0: output, the clock is generated internally 1: input, the clock is provided by an external source unless sel_clk_vip is 00 then it is still the xtal clock.
2:1	sel_clk_vip	R/W	00	00: clk_vip = 27 MHz xtal_clk (overrides vip_output_enable_n). The following is only valid when vip_output_enable_n is 0. 01: clk_vip = DDS7 10: clk_vip = $\overline{\text{DDS7}}$ 11: clk_vip = XIO_D[11]
0	en_clk_vip	R/W	1	1: enable clk_vip
Offset 0x04,7214	***CLK_VLD_CTL***			
31:7	Reserved	R/W	-	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
6	turn_off_ack	R	0	0 - Indicates if the enabled clock is running 1 - Indicates that the clock is being blocked during a frequency change to avoid glitches
5:3	sel_clk_vld_src	R/W	000	000: clk_vld_src = clk_144 001: clk_vld_src = clk_123 010: clk_vld_src = clk_108 011: clk_vld_src = clk_96 100: clk_vld_src = clk_86 101: clk_vld_src = clk_78 110: clk_vld_src = clk_72 111: clk_vld_src = clk_66
2:1	sel_clk_vld	R/W	00	00: clk_vld = 27 MHz xtal_clk 01: clk_vld = clk_vld_src 10: clk_vld = 27 MHz xtal_clk 11: clk_vld = XIO_D[12]
0	en_clk_vld	R/W	1	1: enable clk_vld
Offset 0x04,7300	***AI_OSCLK_CTL***			
31:4	Reserved	R/W	-	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
3	turn_off_ack	R	0	0 - Indicates if the enabled clock is running 1 - Indicates that the clock is being blocked during a frequency change to avoid glitches
2:1	sel_ai_osclk	R/W	00	00: ai_osclk = 27 MHz xtal_clk 01: ai_osclk = DDS4 10: ai_osclk = 27 MHz xtal_clk 11: ai_osclk = XIO_D[13]
0	en_ai_osclk	R/W	1	1: enable clk_ai_osclk
Offset 0x04,7304	***CLK_AI_SCK_CTL***			

Table 11: CLOCK MODULE REGISTERS *…Continued*

Bit	Symbol	Access	Value	Description
31:3	Reserved	R/W	-	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
2	turn_off_ack	R	0	0 - Indicates if the enabled clock is running 1 - Indicates that the clock is being blocked during a frequency change to avoid glitches
1	sel_clk_ai_sck	R/W	0	0: clk_ai_sck = 27 MHz xtal_clk 1: clk_ai_sck = AI_SCK pin
0	en_clk_ai_sck	R/W	1	1: enable clk_ai_sck
Offset 0x04,7308	***CLK_AO_OSCLK***			
31:4	Reserved	R/W	-	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
3	turn_off_ack	R	0	0 - Indicates if the enabled clock is running 1 - Indicates that the clock is being blocked during a frequency change to avoid glitches
2:1	sel_ao_osclk	R/W	00	00: ao_osclk = 27 MHz xtal_clk 01: ao_osclk = DDS3 10: ao_osclk = PLL1 11: ao_osclk = XIO_D[14]
0	en_ao_osclk	R/W	1	1: enable clk_ao_osclk
Offset 0x04,730C	***CLK_AO_SCK_CTL***			
31:3	Reserved	R/W	-	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
2	turn_off_ack	R	0	0 - Indicates if the enabled clock is running 1 - Indicates that the clock is being blocked during a frequency change to avoid glitches
1	sel_clk_ao_sck	R/W	0	0: clk_ao_sck = 27 MHz xtal_clk 1: clk_ao_sck = AO_SCK pin
0	en_clk_ao_sck	R/W	1	1: enable clk_ao_sck
Offset 0x04,7310	***CLK_SPDO_CTL***			
31:4	Reserved	R/W	-	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
3	turn_off_ack	R	0	0 - Indicates if the enabled clock is running 1 - Indicates that the clock is being blocked during a frequency change to avoid glitches
2:1	sel_clk_spdo	R/W	00	00: clk_spdo = 27 MHz xtal_clk 01: clk_spdo = DDS5 10: clk_spdo = 27 MHz xtal_clk 11: clk_spdo = XIO_D[15]
0	en_clk_spdo	R/W	1	1: enable clk_ao_osclk
Offset 0x04,7314	***CLK_SPDI_CTL***			
31:5	Reserved	R/W	-	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.

Table 11: CLOCK MODULE REGISTERS *…Continued*

Bit	Symbol	Access	Value	Description
4	turn_off_ack	R	0	0 - Indicates if the enabled clock is running 1 - Indicates that the clock is being blocked during a frequency change to avoid glitches
3	sel_spdi_clk_src	R/W	0	0: clk_spdi_src = clk_144 1: clk_spdi_src = clk_72
2:1	sel_spdi_clk	R/W	00	00: clk_spdi = 27 MHz xtal_clk 01: clk_spdi = clk_spdi_src 10: clk_spdi = UNDEF 11: clk_spdi = LAN_TXD[0]
0	en_clk_spdi	R/W	1	1: enable clk_spdi
Offset 0x04,7318-0x04,73FCReserved				
General Purpose				
Offset 0x04,7400	***CLK_GPIO_Q4_CTL***			
31:4	Reserved	R/W	-	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
3	turn_off_ack	R	0	0 - Indicates if the enabled clock is running 1 - Indicates that the clock is being blocked during a frequency change to avoid glitches
2:1	sel_clk_gpio_q4_ctl	R/W	00	00: clk_gpio_q4_ctl = 27 MHz xtal_clk 01: clk_gpio_q4_ctl = DDS8 10: clk_gpio_q4_ctl = 27 MHz xtal_clk 11: clk_gpio_q4_ctl = LAN_TXD[1]
0	en_clk_gpio_q4_ctl	R/W	1	1: enable clk_gpio_q4_ctl
Offset 0x04,7404	***CLK_GPIO_Q5_CTL***			
31:4	Reserved	R/W	-	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
3	turn_off_ack	R	0	0 - Indicates if the enabled clock is running 1 - Indicates that the clock is being blocked during a frequency change to avoid glitches
2:1	sel_clk_gpio_q5_ctl	R/W	00	00: clk_gpio_q5_ctl = 27 MHz xtal_clk 01: clk_gpio_q5_ctl = DDS7 10: clk_gpio_q5_ctl = 27 MHz xtal_clk 11: clk_gpio_q5_ctl = LAN_TXD[2]
0	en_clk_gpio_q5_ctl	R/W	1	1: enable clk_gpio_q5_ctl
Offset 0x04,7408	***CLK_GPIO_Q6_12_CTL***			
31:4	Reserved	R/W	-	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
3	turn_off_ack	R	0	0 - Indicates if the enabled clock is running 1 - Indicates that the clock is being blocked during a frequency change to avoid glitches

Table 11: CLOCK MODULE REGISTERS *...Continued*

Bit	Symbol	Access	Value	Description
2:1	sel_clk_gpio_q6_12_ctl	R/W	00	00: clk_gpio_q6_12_ctl = 27 MHz xtal_clk 01: clk_gpio_q6_12_ctl = DDS6 10: clk_gpio_q6_12_ctl = 27 MHz xtal_clk 11: clk_gpio_q6_12_ctl = LAN_TXD[3]
0	en_clk_gpio_q6_12_ctl	R/W	1	1: enable clk_gpio_q6_12_ctl
Offset 0x04,740C	***CLK_GPIO_13_CTL***			
31:4	Reserved	R/W	-	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
3	turn_off_ack	R	0	0 - Indicates if the enabled clock is running 1 - Indicates that the clock is being blocked during a frequency change to avoid glitches
2:1	sel_clk_gpio_13_ctl	R/W	00	00: clk_gpio_13_ctl = 27 MHz xtal_clk 01: clk_gpio_13_ctl = DDS5 10: clk_gpio_13_ctl = UNDEF 11: clk_gpio_13_ctl = LAN_RXD[0]
0	en_clk_gpio_13_ctl	R/W	1	1: enable clk_gpio_13_ctl
Offset 0x04,7410	***CLK_GPIO_14_CTL***			
31:4	Reserved	R/W	-	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
3	turn_off_ack	R	0	0 - Indicates if the enabled clock is running 1 - Indicates that the clock is being blocked during a frequency change to avoid glitches
2:1	sel_clk_gpio_14_ctl	R/W	00	00: clk_gpio_14_ctl = 27 MHz xtal_clk 01: clk_gpio_14_ctl = DDS2 10: clk_gpio_14_ctl = UNDEF 11: clk_gpio_14_ctl = LAN_RXD[1]
0	en_clk_gpio_14_ctl	R/W	1	1: enable clk_gpio_14_ctl
Offset 0x04,7414	***CLK_FGPO_CTL***			
31:9	Reserved	R/W	-	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
8	turn_off_ack	R	0	0 - Indicates if the enabled clock is running 1 - Indicates that the clock is being blocked during a frequency change to avoid glitches
7	Invert_fgpo_clock	R/W	0	Invert FGPO clock 0 : do not invert the clock 1: invert the clock only to the fgpo block and not to the pad.
6	fgpo_output_select	R/W	0	FGPO output select 0: Seperate output mode, The clock to the fgpo and to the pad share the same source, but have seperate paths. 1: Feedback output mode, The clock is driven to the pad then is feedback to the clock block. It then goes through gating logic to the fgpo block.

Table 11: CLOCK MODULE REGISTERS *...Continued*

Bit	Symbol	Access	Value	Description
5	fgpo_output_enable_n	R/W	1	FGPO output enable 0: output, the clock is generated internally 1: input, the clock is provided by an external source. Note: during and after reset the xtal clock is forced onto the fgpo clock. In order to actually allow the input clock to go to the fgpo this register must be written to. This also implies that writing fgpo_output_enable_n = 1 overrides a sel_fgpo_clk = 0.
4:3	sel_clk_fgpo_src	R/W	00	00: clk_fgpo_src = PLL1 01: clk_fgpo_src = UNDEF 10: clk_fgpo_src = DDS2 11: clk_fgpo_src = clk_tm (It is not meant to be used in normal operating mode. The observation is after the output of the duty cycle stretcher, therefore it is the clock that feeds the TM3260).
2:1	sel_clk_fgpo	R/W	00	The following 4 settings are valid when fgpo_output_enable_n = 0 (either of the two output modes). 00: clk_fgpo = 27 MHz xtal_clk 01: clk_fgpo = clk_fgpo_src 10: clk_fgpo = $\overline{\text{clk\_fgpo\_src}}$ 11: clk_fgpo = LAN_RXD[2] The following 3 settings are valid when fgpo_output_enable_n = 1 (the input mode). 01: clk_fgpo = VDO_CLK2
0	en_clk_fgpo	R/W	1	1: enable clk_fgpo
Offset 0x04,7418 CLK_FGPI_CTL				
31:6	Reserved	R/W	-	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
5	turn_off_ack	R	0	0 - Indicates if the enabled clock is running 1 - Indicates that the clock is being blocked during a frequency change to avoid glitches
4	fgpi_output_enable_n	R/W	1	Fgpi output enable 0: output, the clock is generated internally 1: input, the clock is provided by an external source unless sel_fgpi_clk is 00 then it is xtal clock.
3	sel_clk_fgpi_src	R/W	0	0: clk_fgpi_src = DDS3 1: clk_fgpi_src = DDS8
2:1	sel_clk_fgpi	R/W	00	00: clk_fgpi = 27 MHz xtal_clk (voids fgpi_output_enable_n) Only used when fgpi_output_enable_n = 0 : 01: clk_fgpi = clk_fgpi_src 10: clk_fgpi = $\overline{\text{clk\_fgpi\_src}}$ 11: clk_fgpi = LAN_RXD[3]
0	en_clk_fgpi	R/W	1	1: enable clk_fgpi
Offset 0x04,741C-0x04,74FCReserved				
Debug Registers				
Offset 0x04,7500 CLK_STRETCHER_CTL				

Table 11: CLOCK MODULE REGISTERS *…Continued*

Bit	Symbol	Access	Value	Description
31:0	count_stretcher_bits	R/W	0	The count between clock stretches
Offset 0x04,7504	***CLK_WAKEUP_CTL***			
31:2	count_wakeup_bits	R/W	0	The count to use to automatically wake-up the MMIO and processor clocks. The register is a 32-bit register with the two LSB bit hard-coded to zero. If the CLK_WAKEUP_CTL register is written with a value of 0x0000_0008. Then the wake-up count will be set to a count value of 8. This means that the lowest count value is 4 (0x0000_0004 written to the CLK_WAKEUP_CTL register)
1	external_wakeup_enable	R/W	0	Enables the use of pin GPIO[15] as a wake-up event.
0	gpio_interrupt_enable	R/W	0	Enables the use of the GPIO interrupt as an wake-up event.
Offset 0x04,7508	***CLK_FREQ_CTL***			
31:5	freq_ctr_bits	R/W	0	The total time to count clock edges
4	freq_ctr_done	R	1	Signifies that the count is done
3:0	en_ctr_enable	R/W	1	selects which clock to count 0000: Disabled 0001: PLL0 0010: PLL1 0011: PLL2 0100: UNDEF 0101: UNDEF 0110: DDS2 0111: DDS3 1000: DDS4 1001: DDS5 1010: DDS6 1011: DDS7 1100: DDS8
Offset 0x04,750C	***CLK_COUNT_RESULTS***			
31:0	freq_ctr_results	R	-	The result of the count of the clock frequency counting.
Offset 0x04,7510	***ALIGNER_ADJUST (RESERVED DO NOT MODIFY)***			
31:26	Reserved	R	-	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
25:24	Aligner_adjust_vdo_clk2	W1	11	Adjust the aligner for fgpo out (VDO_CLK2) Note: this clock can not have latency added to it.
23:22	Aligner_adjust_area3	W1	10	Adjust the aligner for the clock going to area 3
21:20	Aligner_adjust_fgpo	R/W1	10	Adjust the aligner for fgpo out internal clock
19:18	Aligner_adjust_qvcp	R/W1	10	Adjust the aligner for qvcp out internal clock
17:16	Aligner_adjust_qvcp_pix	R/W1	10	Adjust the aligner for qvcp pix clock
15:14	Aligner_adjust_qvcp_out	R/W1	10	Adjust the aligner for qvcp out (VDO_CLK1)
13:12	Aligner_adjust_area7	R/W1	10	Adjust the aligner for the clock going to area 7
11:10	Aligner_adjust_area6	R/W1	10	Adjust the aligner for the clock going to area 6

Table 11: CLOCK MODULE REGISTERS *…Continued*

Bit	Symbol	Access	Value	Description
9:8	Aligner_adjust_area2	R/W1	10	Adjust the aligner for the clock going to area 2
7:6	Aligner_adjust_area1	R/W1	10	Adjust the aligner for the clock going to area 1
5:4	Aligner_adjust_l_area0	R/W1	10	Adjust the aligner for the late clock going to area 0
3:2	Aligner_adjust_e_area0	R/W1	10	Adjust the aligner for the early clock going to area 0
1:0	Aligner_adjust	R/W1	10	Adjust the aligner for the 3ns aligner The below values apply to each of the above except the 25:24 bits 11 : adds to the clock latency 01, 10 : medium clock latency (default) 00 : decreases the clock latency
Offset 0x04,7514-FDC	***RESERVED***			
Interrupt Registers				
Offset 0x04,7FE0	***INTERRUPT STATUS***			
31	VDO_CLK2_present	R	0	1: Clock present 0: Clock NOT present
30	VDI_CLK2_present	R	0	1: Clock present 0: Clock NOT present
29	ao_sckin_present	R	0	1: Clock present 0: Clock NOT present
28	ai_sckin_present	R	0	1: Clock present 0: Clock NOT present
27	VDI_CLK1_present	R	0	1: Clock present 0: Clock NOT present
20:5	Reserved	R/W	-	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
4	VDO_CLK2 (clk_fgpo)	R	0	1: Clock interrupt
3	VDI_CLK2 (clk_fgpi)	R	0	1: Clock interrupt
2	AO_SCK	R	0	1: Clock interrupt
1	AI_SCK	R	0	1: Clock interrupt
0	VDI_CLK1 (clk_vip)	R	0	1: Clock interrupt
Offset 0x04,7FE4	***INTERRUPT ENABLE***			
31:5	Reserved	R/W	-	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
4	VDO_CLK2 (clk_fgpo)	R/W	0	1: Interrupt enabled 0: Interrupt NOT enabled
3	VDI_CLK2 (clk_fgpi)	R/W	0	1: Interrupt enabled 0: Interrupt NOT enabled
2	AO_SCK	R/W	0	1: Interrupt enabled 0: Interrupt NOT enabled
1	AI_SCK	R/W	0	1: Interrupt enabled 0: Interrupt NOT enabled

Table 11: CLOCK MODULE REGISTERS *…Continued*

Bit	Symbol	Access	Value	Description
0	VDI_CLK1 (clk_vip)	R/W	0	1: Interrupt enabled 0: Interrupt NOT enabled
Offset 0x04,7FE8	***INTERRUPT CLEAR***			
31:5	Reserved	W	-	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
4	VDO_CLK2 (clk_fgpo)	W	0	1: clear interrupt
3	VDI_CLK2 (clk_fgpi)	W	0	1: clear interrupt
2	AO_SCK	W	0	1: clear interrupt
1	AI_SCK	W	0	1: clear interrupt
0	VDI_CLK1 (clk_vip)	W	0	1: clear interrupt
Offset 0x04,7FEC	***SET INTERRUPT***			
31:15	Reserved	W	-	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
4	VDO_CLK2 (clk_fgpo)	W	0	1: set interrupt
3	VDI_CLK2 (clk_fgpi)	W	0	1: set interrupt
2	AO_SCK	W	0	1: set interrupt
1	AI_SCK	W	0	1: set interrupt
0	VDI_CLK1 (clk_vip)	W	0	1: set interrupt
Offset 0x04,7FF0-FF8	***RESERVED***			
Offset 0x04,7FFC	***MODULE_ID***			
31:16	MODULE_ID	R	0xA063	Module ID
15:8	MODULE_ID	R	0	MAJOR_REV ID
7:0	MODULE_ID	R	0	MINOR_REV ID

Chapter 6: Boot Module

PNX15xx Series Data Book – Volume 1 of 1

Rev. 2 — 1 December 2004 Product data sheet

1. Introduction

Before a PNX15xx Series system can begin operating, there are a couple of system related MMIO registers, Chapter 3 System On Chip Resources, but also the main PNX15xx Series interfaces like the main memory interface (MMI) or the PCI that require to be configured. Since the TM3260 cannot begin operating before these registers and circuits are initialized, the TM3260 cannot be relied on to initialize these resources. Consequently, PNX15xx Series needs an independent bootstrap facility for low-level initialization: the Boot module.

The boot module provides boot scripts to reduce the board system cost. The scripts allow support for both host-assisted mode, and standalone boot mode. The supported four boot modes are:

- Boot from an external EEPROM attached on the I^2C bus interface. The EEPROM contains a list of MMIO registers and memory locations to be written.
- Boot for host-assisted systems. The host CPU kicks off TM3260. Therefore the host is in charge of downloading the TM3260 binary program and in charge of setting properly the remaining of the system.
- Boot from NAND or NOR Flash memory devices. The first 8 Kilobytes of the Flash memory contains the bootstrapping program for TM3260. The Flash memory can be 8- or 16-bit wide.

The different modes are determined by the GPIO[11:8,3:0]/BOOT_MODE[7:0] pins.

Once the boot procedure is complete, the boot module goes to sleep until another system reset event occurs, see Chapter 4 Reset.

2. Functional Description

The PNX15xx Series boot sequence begins with the assertion of the system reset. The system reset is seen by the boot module through the internal signal peri_rst_n. After this internal reset is de-asserted only the system boot block and the PCI interfaces are allowed to operate. In particular, the TM3260 remains in the reset state until it is explicitly released from reset during or after the boot procedure. In the standalone boot mode the system boot module is responsible for releasing the TM3260 from the reset state. In host-assisted boot the boot logic releases the PNX15xx Series system from reset such that the PNX15xx Series software driver (which runs on the host processor) can finish the configuration of the PNX15xx Series system, download the TM3260 binary code and then release the TM3260 from its reset state.

2.1 The Boot Modes

The boot modes are defined by the state of the BOOT_MODE[7:0] pins at reset time. Therefore adequate pull-ups and pull-downs should be placed on the system board in order to select the correct mode. Once the internal signal peri_rst_n is released, the BOOT_MODE[7:0] pins are sampled. The sampling mechanism delays the peri_rst_n signal by two clk_27 (the initial 27 MHz clock) periods and delays the sampling of the BOOT_MODE[7:0] pins by five clk_27 periods. This ensures the correct values of the BOOT_MODE pins are latched properly, since after the reset goes away, the values on the GPIO pins may become indeterministic.

The different boot modes based on the state of the BOOT_MODE[7:0] pins is described in the following Table 1.

Table 1: The Boot Modes

BOOT_MODE Bits	GPIO pins	Default	Function	Description
7	11	-	EN_PCI_ARB	1 - Enables the internal PCI system arbiter 0 - Disables the internal PCI system arbiter
6:4	10:8	-	MEM_SIZE	Informs the boot scripts of the total memory size available on the system board. This information is crucial to properly set-up the PCI configuration management in host-assisted mode. The pin code is as follows: 000 - 2MB 001 - 4MB 010 - 8MB 011 - 16MB 100 - 32MB 101 - 64MB 110 - 128MB 111 - 256MB

Table 1: The Boot Modes *...Continued*

BOOT_MODE Bits	GPIO pins	Default	Function	Description
3	3	0x0	CAS_LATENCY	DDR SDRAM devices support different types of CAS latencies. However they do not support all the combinations. PNX15xx Series offers the possibility to program the MMI (and therefore the DDR SDRAM devices) with the appropriate CAS latency at boot time. This is crucial for standalone boot from Flash memories devices since 8 Kilobytes of data is stored into the main memory during the execution of the boot scripts. 0 - 2.5 clock periods 1 - 3 clocks periods
2	2	0x1	ROM_WIDTH/ IIC_FASTMODE	This pin has a dual functional mode: If BOOT_MODE[1:0] = “00”, “01”, or “10” (Boot from Flash memory) **0** - 8-bit data wide ROM **1** - 16-bit data wide ROM If BOOT_MODE[1:0] = “11” (Boot from I^2C EEPROM) **0** - 100 KHz **1** - 400 KHz
1:0	1:0	0x3	BOOT_MODE	The main boot mode is determined as follows: **00** - Set up the system and start the TM3260 CPU from a 8- or 16-bit NOR Flash memory or ROM attached to the PCI-XIO bus. **01** - Set up the system and start the TM3260 CPU from a 8- or 16-bit NAND Flash memory or ROM attached to the PCI-XIO bus. **10** - Set up the PNX15xx Series system in host-assisted mode and allow the host CPU to finish the configuration of the PNX15xx Series system and start the TM3260 CPU. **11** - Boot from an I^2C EEPROM attached to the I^2C interface. EEPROMs of 2 to 64 Kilobytes are supported. The entire system can be initialized in a custom fashion by the boot commands contained in the EEPROM. This mode can be used for standalone or host-assisted boot mode when the other internal boot scripts are not meeting the specific requirements of the application. In this mode the boot script is in the EEPROM. Refer to Section 2.3 for further details on the EEPROM content.

The default state of the BOOT_MODE[3:0] pins may be determined by the internal pull-ups and pull-downs presents in the I/Os of PNX15xx Series. However the BOOT_MODE[7:4] pins must be pulled up or down at board level to ensure a proper boot function.

2.2 Boot Module Operation

The following presents a high level block diagram of the boot module.

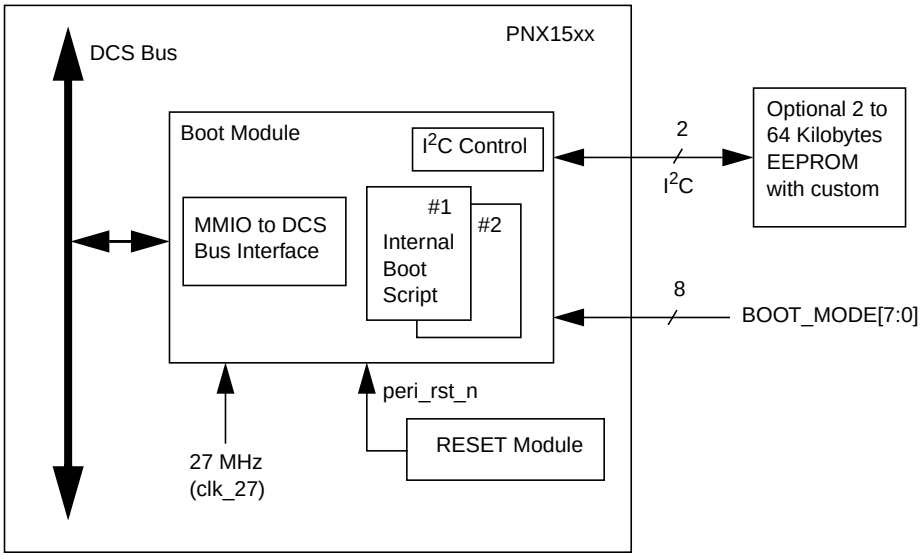


Figure 1: Boot Block Diagram

The four main components of the boot module are:

1. The MMIO to the DCS bus interface.
2. The I^2C Master Interface & Control.
3. The Boot Control & State Machine.
4. The Internal Scripts (detailed in Section 3.)

2.2.1 MMIO Bus Interface

The MMIO bus sub-module contains only the master interface. Therefore despite the general rule there is no MODULE_ID for the boot module and the master interface module can only perform writes. It does not perform reads from other modules sitting on the DCS bus. As a master, this module writes full 32-bit words to the DCS bus. These write requests are then routed to the appropriate MMIO register or to the MMI.

2.2.2 I^2C Master

Depending on the state of the BOOTMODE[1:0] pins, the I^2C master interface gets activated after the reset is released. If the BOOTMODE[1:0] is equal to 0x3 then the boot module takes over the control of the I^2C interface. The data received from the external EEPROM is decoded by the boot state machine. The MMIO bus sub-module is activated to write data on the DCS bus per the command encoding described in Section 2.3. The I^2C master does not arbitrate for the I^2C bus since it is expected that there will be no other bus masters during the boot process. However, the I^2C master does allow clock stretching by the slave (here the EEPROM). The clock stretching is not expected from the EEPROM but the feature is there in order to meet the I^2C

specification. Depending on the state of the BOOT_MODE[2] pin the operating speed of the I^2C interface is 400 KHz, if set to ‘1’, or 100 KHz, if set to ‘0’. The I^2C master derives the 100 KHz or 400 KHz clock from 27 MHz input clock.

The I^2C interface can handle EEPROMs with both 1-byte and 2-byte addressing formats.

2.2.3 Boot Control/State Machine

The boot control/state machine is in fact a mini processor. It fetches commands from the boot scripts or the content of the EEPROM, decodes the command and processes the address or the data depending on the command as documented Section 2.3. When an end of boot command is reached, the I^2C interface is released and can later on be used by the I^2C module.

2.3 The Boot Command Language

The boot script consists of a sequence of 32-bit commands. These commands constitute the language understood by the boot module. The valid commands are:

- A write to a given 32-bit value at a given address (useful for writing device control registers).
- A write to an arbitrary length list of 32-bit values starting at a given address (useful for filling memory with a processor binary image).
- A delay by a given number of 27 MHz clock ticks (useful to wait for completion of an action, such as a PLL frequency change, or a device DMA operation).
- Terminate Boot.

The following Table 2 documents the coding of the four commands.

Table 2: The Boot Commands

Command	Encoding (32 bits each)	Description
write(address,value)	Address: aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa**00** Value: vvvvvvvvvvvvvvvvvvvvvvvvvvvv	Write a single 32-bit value, 'V' at address 'A' (32-bit word aligned) MMIO bus or memory address. 'A' is the 30-bit value composed by the <a...a> bits concatenated with two 0s (makes it a 32-bit word address). 'V' is the 32-bit value composed by the <v...v> bits.
writelist(a,lenght,valarray)	Address: aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa**01** length<n>: nnnnnnnnnnnnnnnnnnnnnnnnnnnn value<1>: vvvvvvvvvvvvvvvvvvvvvvvvvvvv value<2>: vvvvvvvvvvvvvvvvvvvvvvvvvvvv . . value<n>: vvvvvvvvvvvvvvvvvvvvvvvvvvvv	Write an arbitrary length list of 32-bit values, value<1>,..., value<n>, starting at address 'A' (32-bit aligned) MMIO bus or memory address. value<1> is written at address 'A' + 0, value<n> is written at the 32-bit word address 'A+n-1'. 'A' is the 30-bit value composed by the <a...a> bits concatenated with two 0s (makes it a 32-bit word address).
delay(ncycles)	nnnnnnnnnnnnnnnnnnnnnnnnnnnn**0010**	Delay by N 27 MHz cycle periods. Where N is the 28-bit value composed by the <n...n> bits.
Terminate Boot	nnnnnnnnnnnnnnnnnnnnnnnnnnnn**0110**	End boot process. The Boot module releases I2C bus and becomes non-active until a hardware reset or software reset occurs.
	xxxxxxxxxxxxxxxxxxxxxxxxxxxx1x1x	Reserved for future use.

3. PNX15xx Series Boot Scripts Content

Unlike PNX1300 Series systems [1], PNX15xx Series uses internal boot scripts to provide some cost savings at board level by allowing the building of products without the need of an external EEPROM. The following sections describes the content of these scripts. If the content of these internal boot scripts is not suitable for the PNX15xx Series based product, then an EEPROM should be used to override the internal boot scripts, see Section 4..

3.1 The Common Behavior

The three scripts have a common section which is the initial configuration sequence of the PNX15xx Series system. The differences between the boot scripts is detailed in the next sections. The common behavior is described bellow in the order in which it happens:

1. Enable the clocks for the TM3260, 100 MHz, and the MMI, 125 MHz, modules.
2. Enable the clocks for the MMIO bus (a.k.a. the DCS Bus), 102 MHz, and the PCI_SYS_CLK, 33.23 MHz, clocks.

3. Configure the MMI with default DDR SDRAM timing parameter settings that support as many DRAM vendors as possible. It is recommended to verify these default parameters comply with the DDR SDRAM devices used to build the PNX15xx Series system board. Not all the MMI parameters are initialized in the boot scripts some are the reset defaults of the MMI module. The Table 3 summarizes the values of DDR SDRAM timing parameters once the configurations of the MMI is completed by the boot. It is then the TM3260 or the host CPU that is in charge to fine tune these parameters by re-programming the MMI module according to the DDR SDRAM devices used on the PNX15xx Series system board. Furthermore ROW_WIDTH and COLUMN_WIDTH have been set

Table 3: Default DDR SDRAM Timing Parameters

Parameter	Value (Clocks)
tRCD read	4
tRCD write	4
tRRD	4
tMRD	4
tWTR	1
tWR	3
tRP	4
tRAS	9
tRFC	15
tRC	13

to 11 and 8, respectively, which allows the use of any kind of DDR SDRAM densities and configurations during the boot process (i.e. in standalone only 8 Kilobytes of data is written to memory). Finally, some parameters are dependant on the CAS latency of the devices. After review of different DDR SDRAM device datasheets, it is found that devices organized in x32 support, at least, CAS latencies of 3.0. Similarly the devices organized in x16 support at least a CAS latencies of 2.5. In addition to the CAS latencies the x32 and x16 devices require some different settings for the auto-precharge bit. Therefore PNX15xx Series BOOT_MODE[3] pin is also used to determine if a x32 or x16 devices are used in the board system. This assumption is not bullet proof but works for most of the DDR SDRAM vendors.The boot scripts assume a x32 device when CAS latency is 3.0 and a x16 device when the latency is 2.5. Table 4 shows the MMI parameters affected by the BOOT_MODE[3] pin, a.k.a. CAS_LATENCY.

Table 4: CAS Latency Related DDR SDRAM Timing Parameters

Parameter	CAS_LATENCY = 3.0	CAS_LATENCY = 2.5
PRECHARGE_BIT	8	10
tCAS	6	5

4. In all cases the PCI modules PCI Setup and PCI Command registers are programmed as shown in Table 5.

Table 5: PCI Setup and PCI Command register Content

Parameter	Bit Field Values	Comments
EN_PCI2MMI	1	PCI Master can write to PNX15xx Series DDR SDRAM.
EN_XIO	1	XIO operations are enabled
BASE18_PREFETCHABLE	0	
BASE18_SIZE	5	64 Megabytes
EN_BASE18	1	XIO Aperture is enabled
BASE14_PREFETCHABLE	0	
BASE14_SIZE	000	2 Megabytes
EN_BASE14	1	MMIO Aperture is enabled
BASE10_PREFETCHABLE	0	
BASE10_SIZE	BOOT_MODE[6:4] pins	Configured at pin level, see Table 1.
EN_CONFIG_MANAG	1	PCI configuration is authorized
EN_PCI_ARB	BOOT_MODE[7] pin	Configured at pin level, see Table 1.
BUS_MASTER	1	PCI Command register
MEMORY_SPACE	1	PCI Command register

5. The next step is to configure the PCI-XIO module to fetch data from the Flash memory devices connected to the PCI-XIO bus if PNX15xx Series is configured in standalone mode, see next step in Section 3.2. The other option is to configure the PNX15xx Series in host-assisted mode, see Section 3.3.
6. The final step is obviously to send a terminate boot command.

The next Section 3.1.1 contains the content in hexadecimal of the common boot scripts.

3.1.1 Binary Sequence for the Common Boot Script

Table 6: Binary Sequence for the Common Boot Script

CAS_LATENCY = 3.0	CAS_LATENCY = 2.5	Comments
0x1be4_7101		
0x0000_0002		
0x0000_0003		TM3260 Clock
0x0000_0003		MMI Clock
0x1be4_712c		
0x0000_0003		MMIO Clock
0x1be4_710c		
0x0000_0003		PCI_SYS_CLK Clock
0x1be6_5004		
0x0000_0003		MMI DEF_BANK_SWITCH
0x1be6_50C0		
0x0000_000B		MMI ROW_WIDTH
0x1be6_50C4		
0x0000_0008		MMI COLUMN_WIDTH
0x1be6_5088		
0x0000_0008	0x0000_000a	MMI PRECHARGE_BIT: BOOT_MODE[3] • 0x8 for x32 devices, and CAS latency 3.0 • 0xa for x16 devices, and CAS latency 2.5
0x1be6_5080		
0x0000_0133	0x0000_0163	MMI MR
0x1be6_5128		
0x0000_0006	0x0000_0005	MMI TCAS
0x1be6_512c		
0x0000_0760		MMI RF_PERIOD
0x1be6_5000		
0x0000_0001	0x0000_000d	MMI CTL
0x1be6_5100		
0x0004_0004		MMI TRCD
0x1be6_511c		
0x0000_0004		MMI TRRD
0x1be6_5124		
0x0000_0004		MMI TMRD
0x1be4_0010 0x01D60F03		The value of PCI_SETUP depends on the BOOT_MODE[7:4] pins. In this example: • 128 MB DRAM aperture • Internal PCI arbiter enabled
0x1be4_0044		
0x0000_0006		PCI Command Register
<Standalone or Host-assisted Specifics>		Section 3.2 or Section 3.3
0x0000_0006		Terminate boot

3.2 The Specifics of the Boot From Flash Memory Devices

In standalone mode PNX15xx Series fetches its TM3260 binary program and data from a Flash memory device. The PCI module has internal DMA logic that allows, with few MMIO register writes, to autonomously fetch data from Flash memory devices (connected to the PCI-XIO bus) to the main memory. The typical simplified board system is sketched in Figure 2. The aperture settings are also presented in the same Figure 2. The size of the DRAM is programmable with bootstrap options (resistor pull-up or -down) on the BOOT_MODE[6:4] pins. The Flash memory device used for boot must be connected and therefore selectable by the XIO_SEL0 pin.

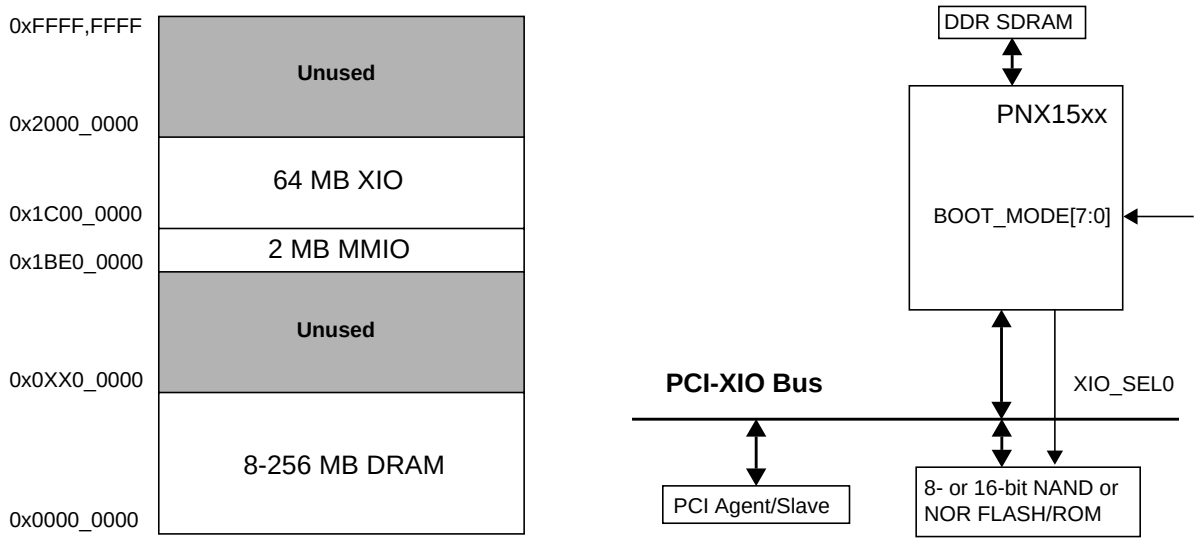


Figure 2: System Memory Map and Block Diagram Configuration for PNX15xx Series in Standalone Mode

In order to be able to access the content of the Flash memory devices the following actions are taken in the boot scripts:

1. The XIO_ACK and the XIO_D[15:8] are removed from their reset state that sets them as GPIO pins. This is done whether the connected Flash memory device is 8- or 16-bit wide.
2. The PCI module DMA is configured depending on the BOOT_MODE[2:0] pin value. Based on the values of these pins the PCI module knows the width and type of the Flash memory device. Table 7 describes the different settings used to access the two types of Flash memory devices. The XIO Sel0 Profile MMIO

register and the DMA Controls MMIO register are the two MMIO registers modified by the boot script. The remaining MMIO registers use the reset state of the PCI module.

Table 7: Flash TIming Parameters Used by the Default Boot Scripts

	NOR Flash		NAND Flash	
Parameter	**Bit Field Value**	**Comment**	**Bit Field Value**	**Comment**
MISC_CTRL	0		0	
SEL0_16BIT	BOOT_MODE[2] pin		BOOT_MODE[2] pin	
SEL0_USE_ACK	0	Fixed wait states	1	Wait for ack
SEL0_WE_HI	0	N/A	0xA	10 PCI clock cycles of HIGH and LOW time for REN
SEL0_WE_LO	0	N/A	0xA	10 PCI clock cycles of HIGH and LOW time for REN
SEL0_WAIT	6	6 PCI clock cycles for the Output Enable signal	2	2 PCI clock cycles for the address to data phase delay
SEL0_OFFSET	0	No Offset	0	No Offset
SEL0_TYPE	1	NOR Flash	2	NAND Flash
SEL0_SIZ	0	8 Megabytes	0	8 Megabytes
EN_SEL0	1	Enabled	1	Enabled
SINGLE_DATA_PHASE	0		0	
SND2XIO	1	Target XIO	1	Target XIO
FIX_ADDR	0	Linear address	0	Linear address
MAX_BURST_SIZE	4	128 data phases	4	128 data phases
INIT_DMA	1	Start to fetch data	1	Start to fetch data
CMD_TYPE	6	XIO read command	6	XIO read command

3. The boot module executes an idle loop to wait for the completion of the fetched data from the Flash memory device to the main memory.
4. The last step before completing the terminate boot command is to set up the TM3260 DRAM aperture registers and kick off the TM3260 CPU.

The next Section 3.3.1.1 contains the content, in hexadecimal, of the Flash boot scripts.

3.2.1 Binary Sequence for the Section of the Flash Boot

Table 8: Binary Sequence for the Section of the Flash Boot

NOR Flash		NAND Flash		Comments
8-bit Mode	**16-bit Mode**	**8-bit Mode**	**16-bit Mode**	
0x1bf0_4005 0x0000_0002 0x5550_0000 0x0000_0155				Enables the functional mode of XIO_ACK and XIO_D[15:8] pins instead of the GPIO mode (default after RESET).
0x1be4_0814 0x0000_0C09	0x0080_0C09	0x006A_8411	0x00EA_8411	XIO Profile 0
0x1be4_080c 0x0000_0296				Start the 8-Kilobyte data fetch from the Flash memory device.
0x000f_0002	0x0007_8002	0x000a_8002	0x0006_1282	Waits for the completion of the 8-Kilobyte fetch.
0x1bf0_0038 TM32_DRAM_HI 0x1bf0_0048 0x0010_0000 0x1bf0_0030 0x8000_0000				TM32_DRAM_HI depends upon the state of the BOOT_MODE[6:4] pins. TM3260 starts executing code from TM32_DRAM_START set to 1 Megabyte. Start TM3260.

3.3 The Specifics of the Host-Assisted Mode

In host-assisted boot mode the PNX15xx Series is in a configuration where an external CPU, such as an external MIPS™, x86, PowerPC™, or SH-5™ is the host. In that case, the PNX15xx Series behaves as a plug-in PCI device. Most of the responsibility of booting is taken care of by the host PCI BIOS and by the PNX15xx Series driver. However, there is still a requirement for a boot script in order to initialize the hardware and get it ready to act as a recognizable PCI device. In addition to the common boot script sequence, the only extra step required is to set a:

- PCI Subsystem Vendor ID. This is a 16-bit value that identifies the board vendor. Philips has the code 0x1131. Manufacturers of PCI plug-in cards for the open market must obtain and use their own ID (from the PCI Special Interest Group)[2].
- Subsystem ID. This is a 16-bit value established by the board vendor to identify a particular board. This is allocated by the vendors PCI Special Interest Group representative. This value is used to identify a suitable driver for PC plug-and-play.

Since these IDs are vendor specific any PCI plug-in board based on PNX15xx Series requires an external EEPROM. This EEPROM has the responsibility to bring the system into a good initial state (can be similar to the data presented in Section 3.1.1) and to personalize the Subsystem ID and Subsystem Vendor ID.

The PNX15xx Series Boot system also provides a host-assisted boot script for standalone board system (i.e. not a PCI plug-in card) that use PNX15xx Series in host-assisted mode. Since the PCI bus of this standalone board system is not visible by the rest of the world it is possible to assign a default value and let the host driver recognize a PNX15xx Series system with the following IDs:

Table 9: Host Configuration Sequence

Boot Script Content	Comments
0x1be4_0010	PCI Setup Register
(0x7583<<10) | (dram_size<<7) | en_pci_arb	Depends on GPIO[11:8]
0x1be4_006c	PCI Subsystem ID is 0x0009
0x0009_1131	PCI Subsystem Vendor ID is 0x1131

Finally on a PCI bus the sizes for all the apertures must be given an unique physical addresses at PCI BIOS device enumeration time (DRAM, MMIO and XIO for the PNX15xx Series). This is the work of the host PCI BIOS driver.

Remark: The aperture sizes are written at boot time into the PCI module MMIO registers. The host PCI BIOS software retrieves the values by a write, followed by a read to the PCI Configuration space base address registers. It then assigns a suitable value to each base address. Refer to [2], Section 6.2.5.1, “Address Maps” for more details.

A typical simplified board system is sketched in Figure 3. The aperture allocation seen in the Figure 3 is an example of how the host BIOS can set the location of the apertures.

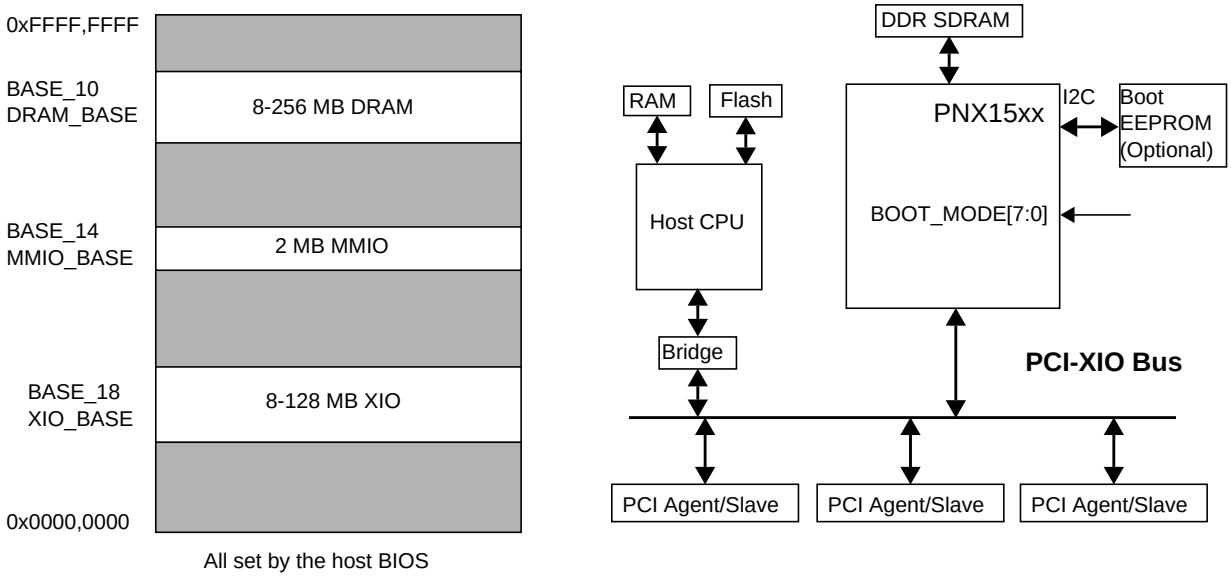


Figure 3: System Memory Map and Block Diagram Configuration for PNX15xx Series in Host-assisted Mode

4. The Boot From an I²C EEPROM

If none of the built-in scripts is suitable e.g., due to a different type of NAND-Flash or a different memory organization or anything not matching the internal boot scripts, an external serial boot EEPROM is required. Depending on the application characteristics, this can be a small (1 Kilobyte or less) EEPROM that contains a small boot script and starts the PNX15xx Series system in host-assisted mode or boot from Flash memory or ROM devices. Alternately, a large serial EEPROM can be used to contain a complete disk file system or an upload capability from another device than Flash/ROM.

For a 2-Kilobyte or smaller EEPROM, the script must start at byte address 1 (not 0). For a 4-Kilobyte or larger EEPROM, the boot script must start at byte address 0. More details in Section 4.3. Each set of four successive bytes is assembled into a 32-bit word value (the byte read first ends up as the least significant byte, LBS). The 32-bit words are then interpreted as commands, as documented earlier in Section 2.3.

Remark: It has been seen that depending on the Write Protect pin status, some EEPROMs do not behave the same way on a write of 0 bytes (Section 4.3). The internal counter gets or does not get incremented which makes this rule of where the first byte is located at address 0 or 1 different. Refer to EEPROM datasheet or try both options.

4.1 External I²C Boot EEPROM Types

The PNX15xx Series Boot module supports all I²C EEPROMs (sometimes called 2-wire EEPROMs) that use a 1-byte or 2-byte address protocol and respond to an I²C device code 1010 (binary). Subtle differences exist between devices For example:

- It is recommended to avoid devices that have partial array write protection, since such devices could be overwritten by accidental or intentional I²C writes, causing boot failure during the next reset.
- Some devices may have additional functionality that is useful, like a watchdog timer or a power voltage drop sensor.
- Availability from different vendors may vary.
- Programming protocols may vary.

Table 10 lists a variety of devices. This list is by no means exhaustive, nor has operation for all these devices been verified.

Table 10: Examples of I²C EEPROM Devices

Size	Device	Write Protection Coverage	Address Protocol	Comment
256 bytes	ATMEL 24C02	Full Array	1 byte	
512 bytes	ATMEL 24C04	Full Array	1 byte	
1 kilobytes	ATMEL 24C08	Full Array	1 byte	Tested
2 kilobytes	PHILIPS PCF85116-3	Full Array	1 byte	
2 kilobytes	SUMMIT SMS8198	Full Array	1 byte	Includes power-on reset for board system reset generation

Table 10: Examples of I²C EEPROM Devices

Size	Device	Write Protection Coverage	Address Protocol	Comment
16 kilobytes	ATMEL 24C128	Full Array	2 bytes	
32 kilobytes	ATMEL 24C256	Full Array	2 bytes	Tested
64 kilobytes	ATMEL 24C512	Full Array	2 bytes	

4.2 The Boot Commands and The Endian Mode

When writing to an MMIO register address, there is no endian mode issue. The msbit of the word ‘V’ (Table 2) end up as the msbit of the MMIO register. When writing to an SDRAM address there is an endian mode issue. Depending on the current endian mode (Section 4. on page 3-8), 32-bit words get written to memory through the DCS DRAM gate (Section 2.3 on page 3-4) in one of these two ways:

- In little-endian mode, the MSB of ‘V’ (or the last read EEPROM byte of the word), end up in memory byte address ‘A+3’ and LSB (or first read EEPROM byte), end up at the byte address ‘a’.
- In big-endian mode, the MSB of ‘V’ (or last read EEPROM byte), end up at the byte address ‘A’ and the LSB (or first read EEPROM byte), end up at the byte address ‘A+3’.

4.3 Details on I²C Operation

To retrieve the boot script, the Boot module performs the following I²C transactions:

- START, 1010000, wait-for-ack, 00000000, wait-for-ack, 00000000, wait-for-ack, STOP
- START, 1010001, wait-for-ack, <accept data byte, send ACK or STOP if done>.

The interpretation of this sequence by 2048 bytes or smaller EEPROMs is:

- Write a byte value 0 to address 0 (setting the next address-pointer to byte address 1).
- Read, starting from address 1.

Hence, for a 2048-byte or smaller EEPROM, the boot image must start at byte 1.

The interpretation of this sequence by 4096 bytes or larger EEPROMs is:

- Write a 0 byte-long sequence to address 0 (setting next address pointer to byte address 0).
- Read, starting from address 0.

Hence, for a 4096-byte or larger EEPROM, the boot image must start at byte 0.

5. References

[1] “PNX1300 Series Media Processor”, Feb. 15th 2002, Philips Semiconductors, Inc.

[2] “PCI Local Bus Specification, Rev 2.2”, Dec. 18th, 1998, PCI Special Interest Group.

Chapter 7: PCI-XIO Module

PNX15xx Series Data Book – Volume 1 of 1

Rev. 2 — 1 December 2004 Product data sheet

1. Introduction

PNX15xx Series includes a PCI interface for easy integration into personal computer applications (where the PCI-bus is the standard for high-speed peripherals). In embedded applications the PCI bus can interface to peripheral devices that implement functions not provided by the on-chip modules or to connected several CPUs together.

The main function of the PCI interface is to connect the PNX15xx Series on-chip MTL bus (and therefore its main memory) and its internal registers to the rest of the world. A bus cycle on PCI that targets an address mapped into PNX15xx Series memory space will cause the PCI interface to create a MTL bus cycle targeted at DRAM. From PNX15xx Series, only the TM3260 CPU can cause the PCI interface to create PCI bus cycles; the other on-chip modules cannot see external hardware through the PCI interface. From PCI, DRAM and most of the registers in MMIO space can be accessed by external PCI initiators.

The PCI interface implements DMA (also called block or burst transfers) and non-DMA transfers. DMA transfers are interruptible on 64-byte boundaries. The PCI interface can service outbound (PNX15xx Series → PCI) and inbound (PCI → PNX15xx Series) data flows simultaneously.

The following classes of operations invoked by PNX15xx Series cause the PCI interface to act as a PCI initiator:

- Transparent, single-word (or smaller) transactions caused by TM3260 loads and stores to one of the two available the PCI address aperture, PCI1 and PCI2.
- Explicitly programmed single-word I/O or configuration read or write transactions
- Explicitly programmed multi-word DMA transactions.

The PNX15xx Series PCI interface responds as a target to external initiators for a limited set of PCI transaction types:

- Configuration read/write.
- Memory read/write, read line, and read multiple to the PNX15xx Series DRAM or MMIO apertures.

PNX15xx Series ignores PCI transactions other than the above.

The PCI-XIO module also includes an XIO interface. The XIO interface “steals PCI cycle” to run XIO transfers before giving control back to PCI. The XIO interface supports IDE, NAND and NOR type Flash and Motorola devices, in 8- or 16-bit datapath.

2. Functional Description

The Document title variable module supports 33 MHz PCI spec version 2.2. It can operate as a configuration manager or it can also act as a target to external configuration cycles when an external processor and north bridge are used in the system.

Features:

- Three base addresses, i.e. apertures, are supported.
- Option to enable internal PCI system arbiter which can support up to three external PCI masters.
- As a PCI master, it can generate all non-reserved types of single transaction PCI cycles: IO, memory, interrupt acknowledge and configuration cycle.
- Linear burst mode is supported on memory transactions. Other burst mode transfers are terminated after a single data transfer.
- A DMA engine provides high speed transfer to and from SDRAM and an external PCI device. The DMA can also be used to transfer data to and from XIO devices.
- The PCI clock and $\overline{\text{PCI\_RST}}$ are generated externally and input to this module.
- In PCI terminology it is a single function device.

The following general PCI features are not implemented in the Document title variable module:

- As a PCI target, the device only responds to memory and configuration cycles.
- Subtractive decoding is not supported.
- There is no hard-coded legacy decoding of address space (such as VGA IO and memory).
- Burst to configuration space is not supported.

2.1 Document title variable Block Level Diagram

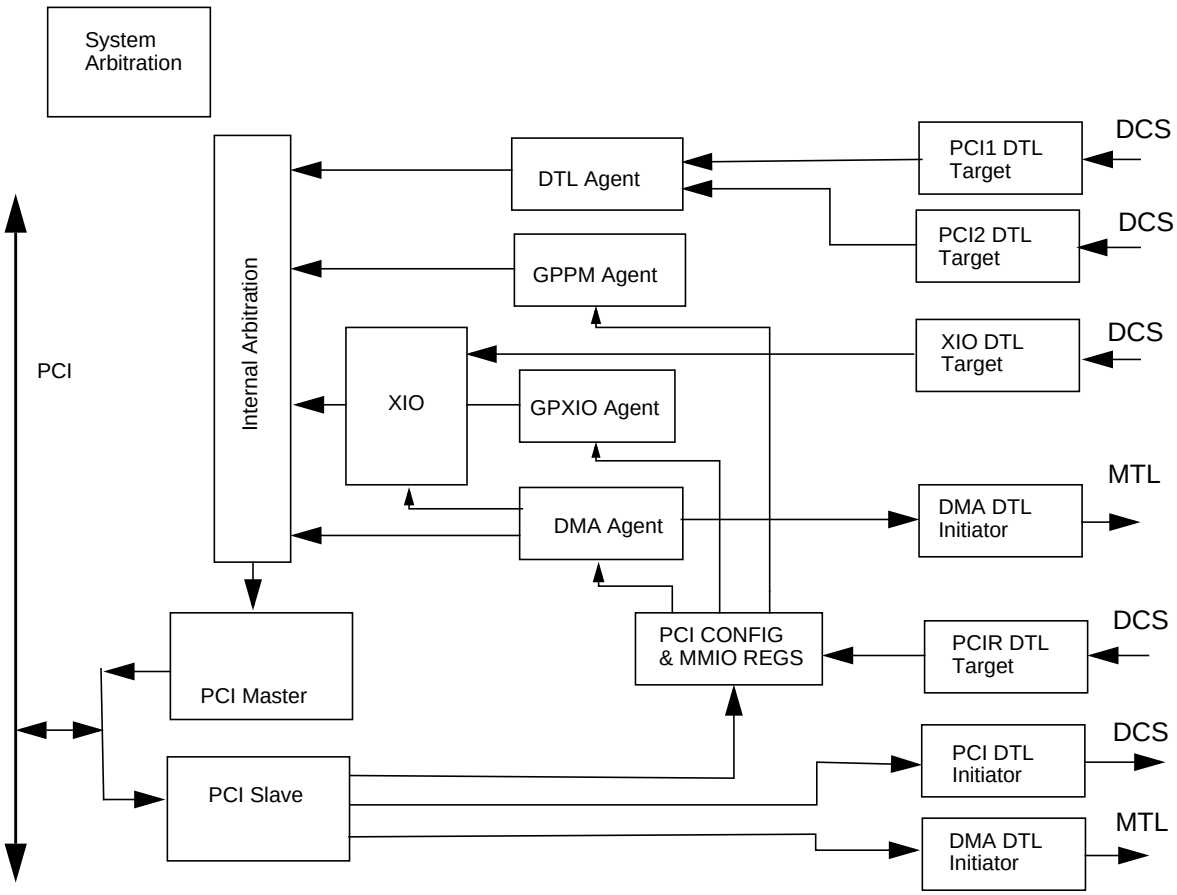


Figure 1: Document title variable Block Diagram

2.2 Architecture

Supported commands on the Document title variable are shown in the following table:

Table 1: Supported PCI Commands

Command: PCI Target Responses	Command: PCI Master Generates
Memory Read	IO Read
Memory Write	IO Write
Configuration Read	Memory Read
Configuration Write	Memory Write
Memory Read Multiple	Configuration Read
Memory Read Line	Configuration Write [1]
Memory Write and Invalidate	Memory Read Multiple
	Memory Read Line
	Memory Write and Invalidate
	Interrupt Acknowledge

[1] Configuration write can be initiated only when configuration management is enabled.

3. Operation

3.1 Overview

The Document title variable module supports the 33 MHz PCI spec version 2.2.

Access from external masters may be restricted to word-only if desired. When this feature is enabled, attempted access with less than a word will result in the PCI slave terminating the transaction with a target abort or ignoring the write and returning 0 on read. This behavior is determined by a configuration switch.

When the PCI device can not return data on reads within 16 PCI clocks, the transactions will terminate in a retry. The read will be completed internally and the PCI will hold the data exclusively for the initiating agent for the duration of the “read_lifetime” timer. No other read will be accepted while the timer is active. After the timer expires, any read request will be accepted. The saved data will be tossed if a different master requests a read.

Any XIO device(s) can be accessed any time after the PCI configuration space has been initialized. To be PCI compliant, it will monitor the internal address rather than the IO pads. At this time, an XIO cycle is run on the PCI pins. PCI pins AD[23:0] present the address to the device while AD[31:24] contain the data. The PCI CBE pins are used for XIO control. There are five dedicated pins to be used as chip selects to the device(s).

The following table shows how the XIO supports various 8- and 16-bit XIO devices.

Table 2: XIO Pin Multiplexing

Signals	I/O	68360 16-Bit	68360 8-Bit	NOR flash 16-Bit	NOR Flash 8-Bit	NAND-flash 16-Bit	NAND-Flash 8-Bit	IDE
PCI Signals								
DEVSEL#	I/O	NA	NA	NA	NA	NA	NA	NA
FRAME#	I/O	NA	NA	NA	NA	NA	NA	NA
IRDY#	I/O	NA	NA	NA	NA	NA	NA	NA
TRDY#	I/O	NA	NA	NA	NA	NA	NA	NA
STOP#	I/O	NA	NA	NA	NA	NA	NA	NA
IDSEL	I	NA	NA	NA	NA	NA	NA	NA
PAR	I/O	NA	NA	NA	NA	NA	NA	NA
PERR#	I/O	NA	NA	NA	NA	NA	NA	NA
SERR#	I/O	NA	NA	NA	NA	NA	NA	NA
REQ_A#	I	NA	NA	NA	NA	NA	NA	NA
REQ_B#	I	DSACK	DSACK	NA	NA	NA	NA	NA
REQ#	I/O	NA	NA	NA	NA	NA	NA	NA
GNT_A#	O	NA	NA	NA	NA	NA	NA	NA

Table 2: XIO Pin Multiplexing *…Continued*

Signals	I/O	68360 16-Bit	68360 8-Bit	NOR flash 16-Bit	NOR Flash 8-Bit	NAND-flash 16-Bit	NAND-Flash 8-Bit	IDE
GNT_B#	O	NA	NA	NA	NA	NA	NA	NA
GNT#	I/O	NA	NA	NA	NA	NA	NA	NA
AD[31:24]	I/O	D[7:0]	D[7:0]	D[7:0]	D[7:0]	AD[7:0]	AD[7:0]	D[7:0]
AD[23:16]	I/O	A[23:16]	A[23:16]	A[23:16]	A[23:16]	D[15:8]	NA	D[15:8]
AD[15]	I/O	A[15]	A[15]	A[15]	A[15]	NA	NA	CS1
AD[14]	I/O	A[14]	A[14]	A[14]	A[14]	NA	NA	CS0
AD[13:11]	I/O	A[13:11]	A[13:11]	A[13:11]	A[13:11]	NA	NA	A[2:0]
AD[10]	I/O	A[10]	A[10]	A[10]	A[10]	NA	NA	DIOW
AD[9]	I/O	A[9]	A[9]	A[9]	A[9]	NA	NA	DIOR
AD[8]	I/O	A[8]	A[8]	A[8]	A[8]	NA	NA	DATA_DIR
AD[7:2]	I/O	A[7:2]	A[7:2]	A[7:2]	A[7:2]	NA	NA	NA
AD[1]	I/O	A[1]	A[1]	A[1]	A[1]	ALE	ALE	NA
AD[0]	I/O	A[0]	A[0]	A[0]	A[0]	CLE	CLE	IORDY
CBE3#	I/O	A[24]	A[24]	A[24]	A[24]	NA	NA	NA
CBE2#	I/O	AS	AS	OEN	OEN	REN	REN	NA
CBE1#	I/O	R/WN	R/WN	WN	WN	WEN	WEN	NA
CBE0#	I/O	DS	DS	NA	NA	NA	NA	NA
XIO Signals								
XIO_A[25]	O	A[25]	A[25]	A[25]	A[25]	NA	NA	NA
XIO_ACK	I	NA	NA	R/BN	R/BN	R/BN	R/BN	NA
XIO_SEL0,1, 2,3,4	O	CS	CS	CE	CE	CE	CE	CE
XIO_DAT[15:8]	I/O	D[15:8]	NA	D[15:8]	NA	NA	NA	NA
GPIO Signals								
INTREQ is an input for IDE style XIO. It may be connected to any available GPIO. This signal is then routed to the TM3260 VIC interrupt controller. Or any of the direct interrupt lines can be used. See interrupt assignments in Section 6.1 on page 3-12.								

3.1.1 NAND-Flash Interface Operation

Interfacing to a NAND-Flash involves both hardware setup and software support. The hardware support is designed to be very flexible in supporting the standard devices plus extensions that may be provided by some flash vendors. Table 3 shows recommended settings for the hardware configured for various NAND-Flash operations.

A NAND transaction may consist of 0, 1, or 2 command phases and 0, 1, 2, 3 or 4 address phases, and n data phases. The sequence is as follows: first command, low address (address bits [7:0]), middle address (address bits [16:8]), high address

(address bits [24:17]), data, second command. For transactions with fewer than three address phases, low address is first dropped, then middle address. Any transaction that includes an address phase must include at least one command phase.

Table 3: Recommended Settings for NAND

Description	Cmd No.	Addr No.	Include Data	Monitor ACK	Cmd A	Cmd B	Notes
Read Data	1	3[1]	Y	Y	00h or 01h	NA	Recommended to use DMA. This may be set to more than one segment if restricting max_burst_size to 128.
Read ID	1	1	Y	N	90h	NA	Recommended to use direct (or indirect) access.
Read Status	1	0	Y	N	70h	NA	May read up to four bytes of status with direct access.
Write Data	2	3[1]	Y	Y	80h	10h	Recommended to use DMA.
Block Erase	2	2[1]	N	Y	60h	d0h	Recommended to use direct (or indirect) access.
Reset	1	0	N	N	ffh	NA	Recommended to use direct (or indirect) access.

[1] 64-MB devices will require more address phases than shown..

With a direct access to the NAND, n is limited to 4 bytes. Using the DMA, n is limited to the segment length, 512 or 528 bytes with spare area. This is to allow time for the busy signal to become stable at segment boundaries. The DMA may be programmed to read much larger areas if the NAND does not assert its busy state or is allowed to pause at segment boundaries. Programmers should consult the vendor's data sheet for the appropriate NAND-Flash selection.

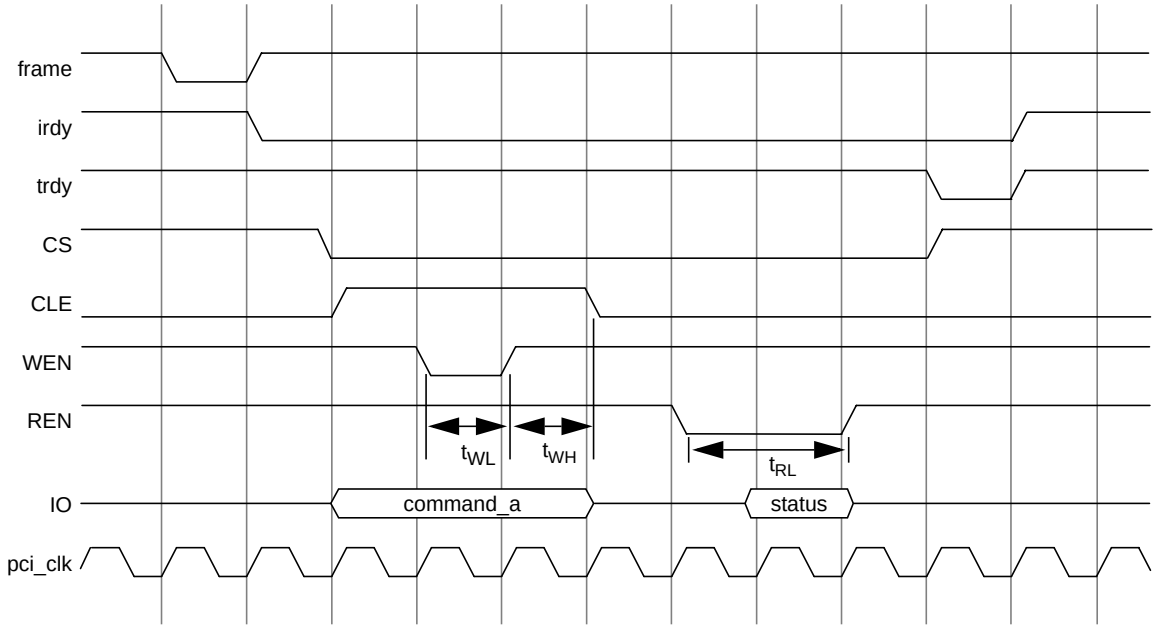
The WEN and REN timing information will also be found in the data sheets. The Document title variable module supports read profiles with low time from 1 to 4 PCI clock periods. Write profiles of 1 to 4 PCI clock periods is supported for command and address writes. Data writes must use a high time of at least 2 PCI clock periods. If data is not part of the transaction, the second command will follow the last address phase.

The ACK signal is monitored, when enabled, only at predetermined parts of the transaction. During read operations, it will monitor the ACK after the last address phase, before the read begins. The fixed delay must be programmed to a value sufficient to allow the ACK to become valid before sampling it. This should include time to double synchronize the ACK to the PCI clock. The ACK is also sampled before starting a NAND transaction (but after the PCI wrapper has started). This applies to all types of transactions. Even a status read will stall until the device is ready if monitor ACK is enabled when starting the NAND transaction.

The read data operation may be done by blending DMA and direct access to minimize the time the PCI bus is blocked from other types of transactions. To do this, set the profile to issue 1 command, 3 address phase, and no data. Also load the appropriate command into the Command A register. Next do a write to the starting address of interest. Change the profile to 0 command, 0 address, include data. The DMA should be programmed to transfer the selected amount of data to SDRAM. If the DMA is started before the device is ready, it will stall until the device is ready.

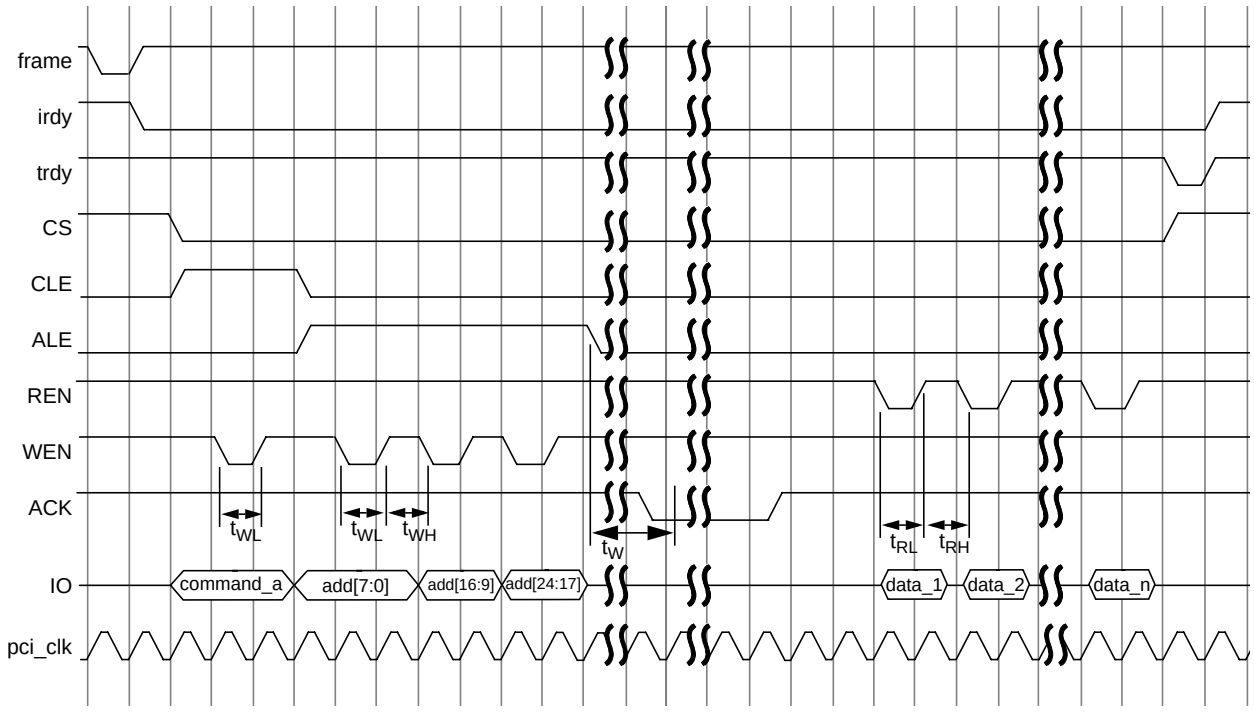
The ACK may be monitored to determine when the device is ready prior to initiating the DMA. Once the device is ready, no further monitoring of the ACK takes place. If the amount of data to be transferred exceeds one segment, the max burst size should be set to 128 to allow for pause in the transfer that allows the ACK to be monitored between segments. Note that this approach will not pause at the correct location if the spare area is being accessed.

Examples of block erase, data read and write and status read are shown in the following timing diagrams.



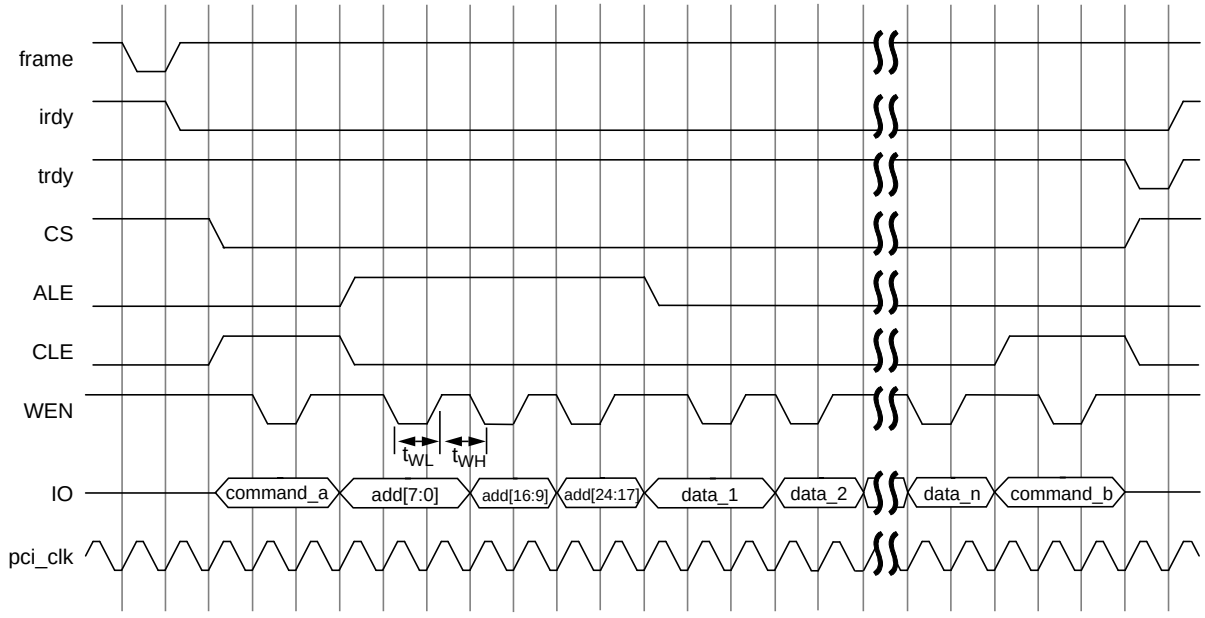
t_{WH}: (WEN_hi + 1) * pci_clk period Shown with WEN_hi = 0
t_{WL}: (WEN_lo + 1) * pci_clk periodShown with WEN_lo = 0
t_{RL}: (REN_lo + 1) * pci_clk periodShown with REN_lo = 1

Figure 2: Read Status



t_{WH}: (WEN_hi + 1) * pci_clk period Shown with WEN_hi = 0
t_{WL}: (WEN_lo + 1) * pci_clk periodShown with WEN_lo = 0
t_{RH}: (REN_hi + 1) * pci_clk periodShown with REN_hi = 0
t_{RL}: (REN_lo + 1) * pci_clk periodShown with REN_lo = 0
t_{W}: (DLY + 1) * pci_clk periodWait time until ACK valid

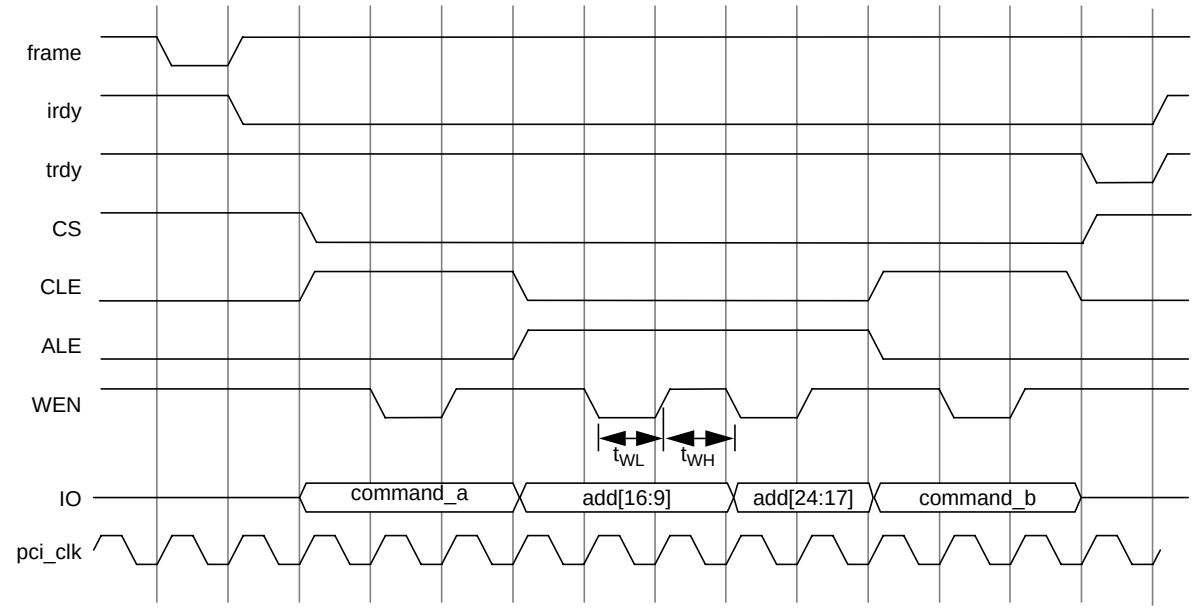
Figure 3: Read Data



t_{WH}: (WEN_hi + 1) * pci_clk period Shown with WEN_hi = 0
t_{WL}: (WEN_lo + 1) * pci_clk periodShown with WEN_lo = 0

Figure 4: Write Data

Refer to Table 2 for signal descriptions.



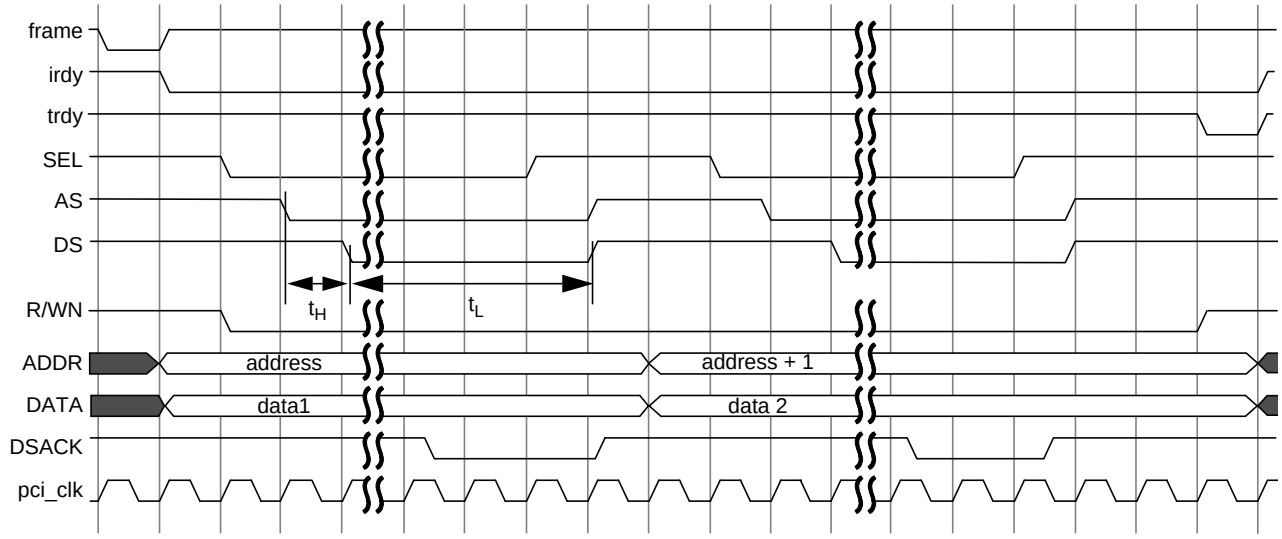
t_{WH}: (WEN_hi + 1) * pci_clk period Shown with WEN_hi = 0
t_{WL}: (WEN_lo + 1) * pci_clk periodShown with WEN_lo = 0

Figure 5: Block Erase

3.1.2 Motorola Style Interface

The Motorola style interface supports 8-bit or 16-bit devices. The following timing diagrams illustrate a 2-byte write and 2-byte read operation. The time between the falling edge of AS and DS is controlled by the DS time high field in the profile register. The time low is determined by the DS time low field of the profile register or by the external device if “wait for ACK” is enabled.

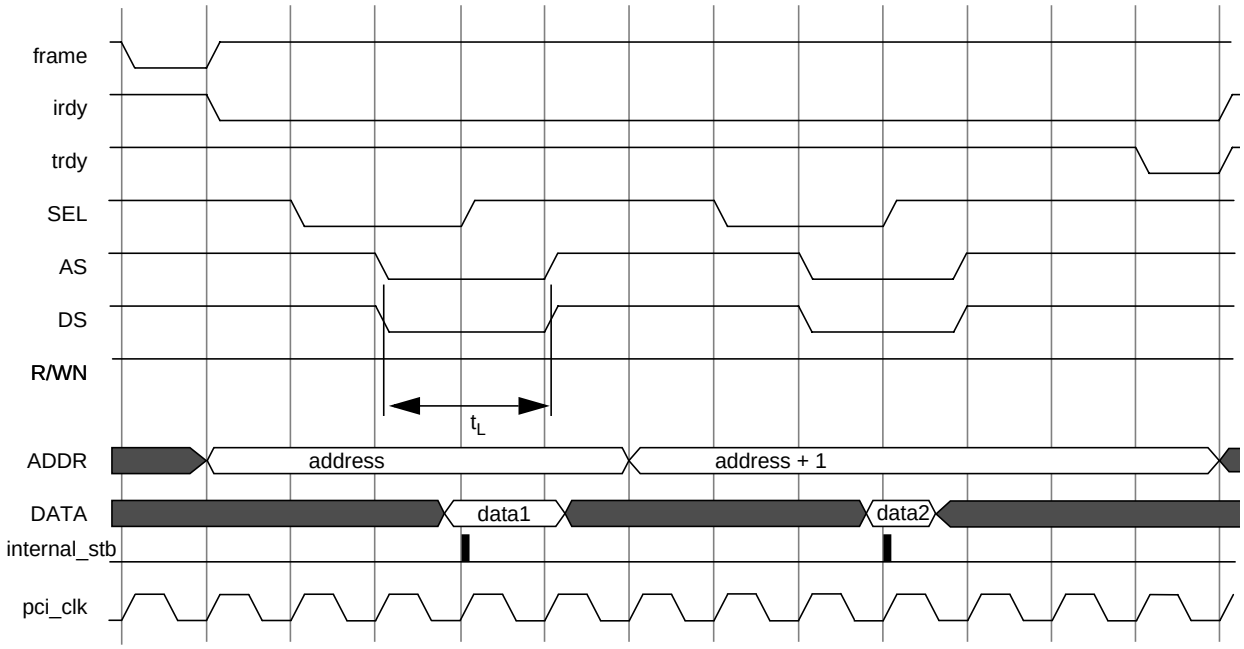
The t_H (write time high) and t_L (wait low) timing should be programmed to match the device according to the vendor’s specification. The resolution is a multiple of the PCI clock period. Refer to the descriptions for the XIO Select Profile registers.



t_H: DS_high * pci_clk period Shown with DS_high = 1
t_L: DS_low * pci_clk periodShown with DSACK monitoring

Figure 6: Motorola Write

Refer to Table 2 for signal descriptions.



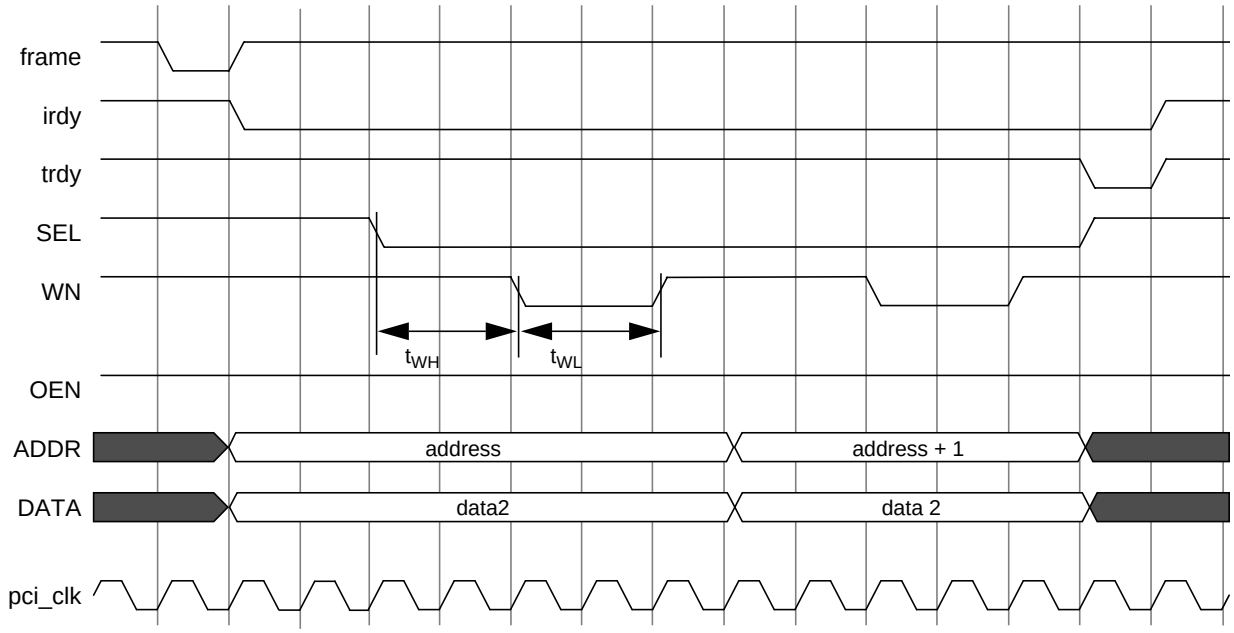
t_L: (DS_low +1) * pci_clk periodShown with DS_lo = 1

Figure 7: Motorola Read

Refer to Table 2 for signal descriptions.

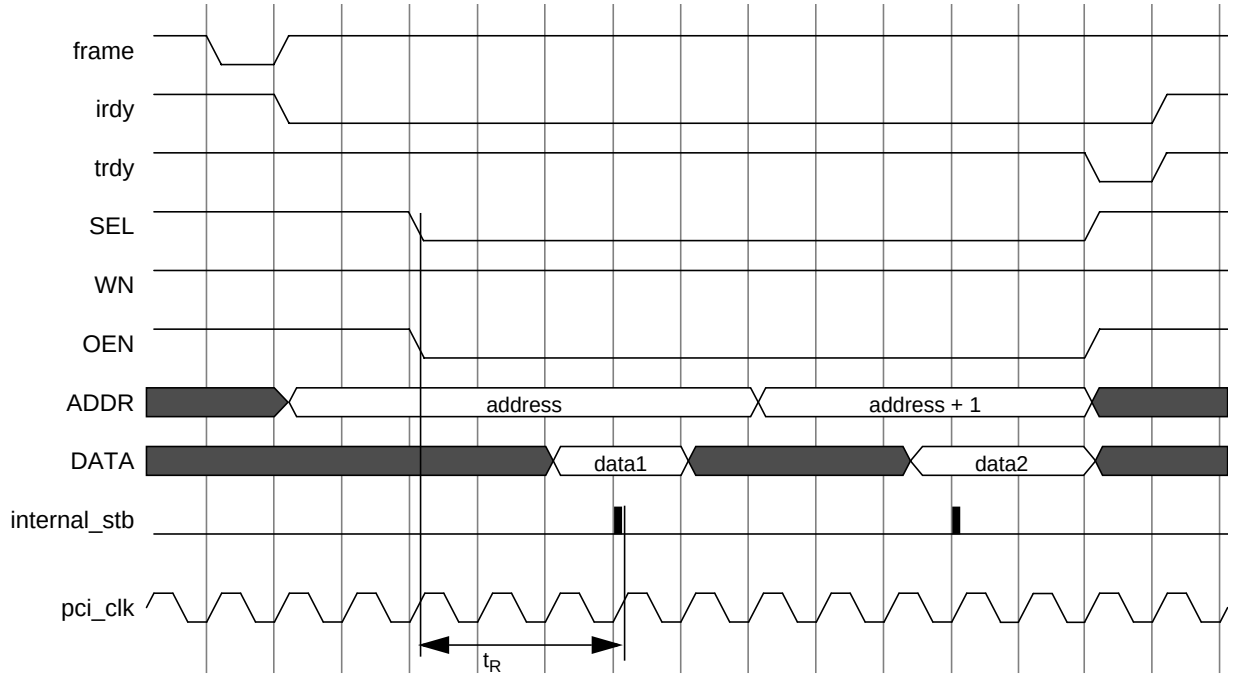
3.1.3 NOR Flash Interface

The NOR flash interface supports 8-bit or 16-bit devices. The following timing diagrams illustrate write and read timings for a typical NOR device. The busy signal is not shown; it should be inactive during these cycles. Typically, the busy signal will be monitored before starting a transaction to the NOR flash. The t_{WH} (write time high) and t_{WL} (write time low) timing should be programmed to match the device based on the flash vendor's specification. Refer to the descriptions for the XIO Select Profile registers. The resolution is a multiple of the PCI clock period. The t_R (read time, or "wait for data") is also programmed in the profile bits[13:9].



t_{WH}: (WN_high + 1) * pci_clk period Shown with WN_high = 1
t_{WL}: (WN_low + 1) * pci_clk periodShown with WN_low = 1

Figure 8: NOR Flash Write



t_R: OEN_lo * pci_clk period Shown with OEN_lo = 3

Figure 9: NOR Flash Read

3.1.4 IDE Description

The IDE (ATA) interface supports PIO mode transfer with a theoretical maximum transfer rate of 16.6 MB/s (PIO-4 mode). The XIO module DMA is used for data transfer to and from the disk.

All IDE disk registers (eight command and one control) are accessible via PI. All IDE disk registers are indirectly accessed via GPXIO registers. Figure 10 shows a block diagram of the IDE interface.

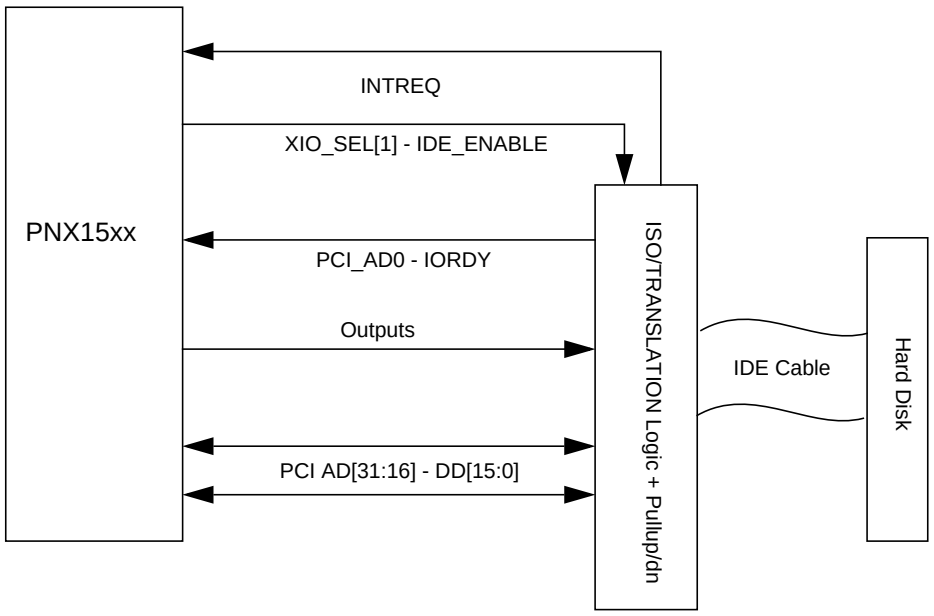


Figure 10: IDE Interface

The IDE port is multiplexed with PCI, FLASH and Motorola interface pins. There are two dedicated pins, IDE_ENABLE (XIO_SEL[1]) and INTREQ. The IDE Disk interrupt (INTREQ) is connected to a GPIO signal, which is routed to the VIC through GPIO or any direct interrupt line.

The PNX15xx Series SYS_RSTN_OUT can be connected with the IDE interface reset. All outputs are driven on PCI_CLK. All inputs are registered on PCI_CLK. The Low and High periods of DIOR/DIOW are programmable (using sel profile register).

All physical signals need to be isolated from PCI on the board as shown in Figure 11

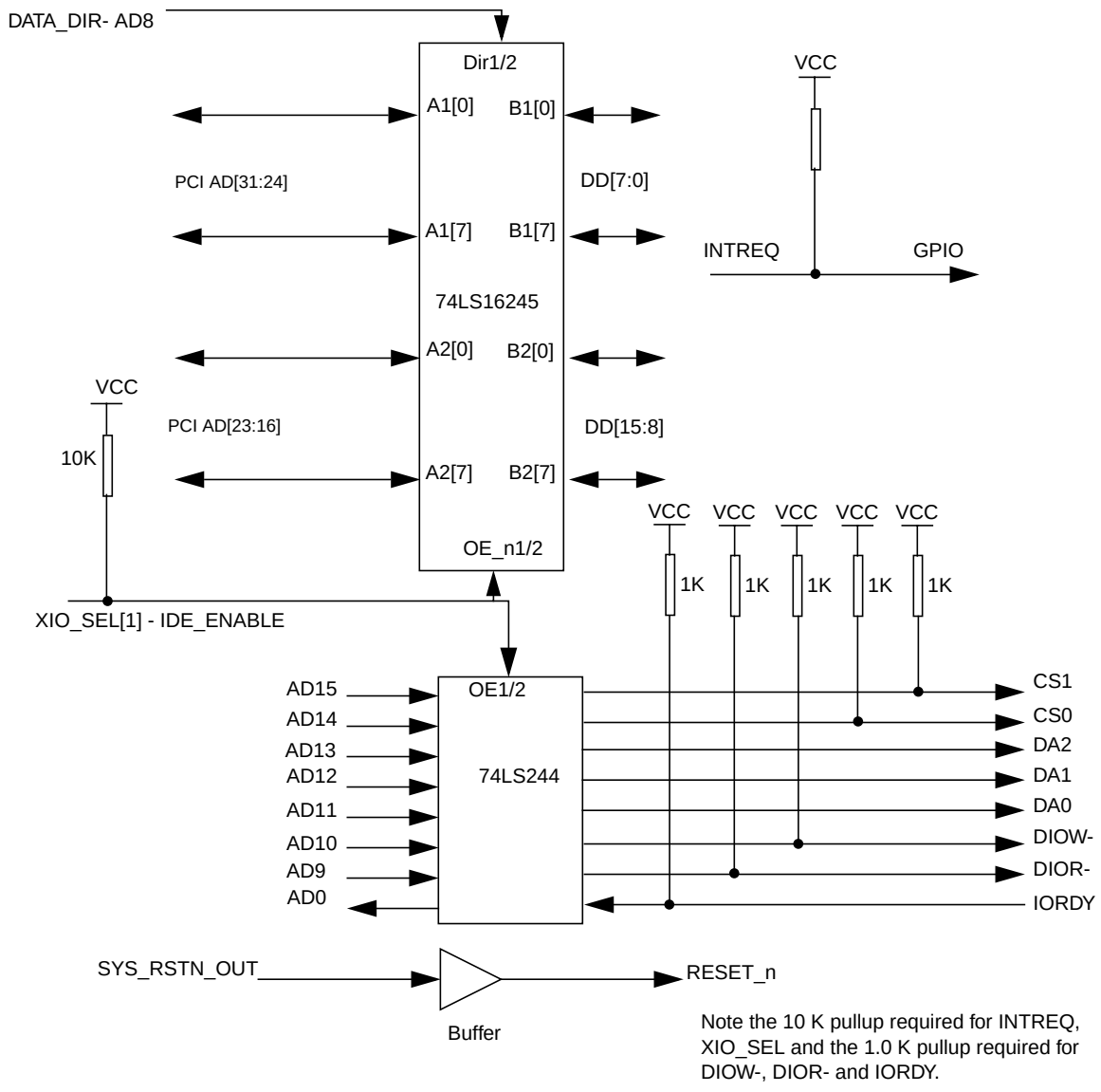


Figure 11: Isolation Translation Logic

Data Transfer Operation

In PIO mode, data transfer to/from disk is done using read/write operations of the command and control block registers. PI/PO protocol is explained in the ATA-2 specification.

All command block registers can be programmed using direct or indirect access in the XIO block. All disk registers are programmed. When the disk is ready to transfer data, DMA is enabled.

Registers

All IDE device registers are defined in the ATA-2 Specification. These registers can be accessed directly from PI or indirectly via GPXIO registers. The lower five bits of the GPXIO address register need to be configured as follows:

Table 4: GPXIO Address Configuration

Address to be Written	Register Name	Address on IDE				
		CS1	CS0	DA2	DA1	DA0
5'b40	Data register	1	0	0	0	0
5'b44	ERR/Feature	1	0	0	0	1
5'b48	Sector count	1	0	0	1	0
5'b4C	Sector number	1	0	0	1	1
5'b50	Cylinder Low	1	0	1	0	0
5'b54	Cylinder High	1	0	1	0	1
5'b58	Device/Head	1	0	1	1	0
5'b5C	Status/Command	1	0	1	1	1
5'b38	Alternate status/Device control	0	1	1	1	0

Programming IDE Registers

IDE is a submodule of Document title variable. It shares PCI pins with other XIO blocks. Three XIO SEL pins can be configured for use by any XIO device. Each SEL pin is associated with the profile register in the PCI block. The profile register determines the mode of the SEL pin, pulse width for control signals and memory apertures for each mode.

Before accessing any IDE register, the appropriate profile register needs to be programmed. For example, if XIO_SEL[1] has been used for IDE, the sel1_profile register needs to be programmed and IDE needs to be enabled.

- At power on, the IDE disk will respond in PIO-0 mode only.
- Program the appropriate register in PIO-0 mode to set PIO-4 mode.
- Using sel1_profile register, set lo and high period of DIOR/DIOW pulses for PIO-4 mode.
- High period in selx_profile register is used for the setup time of DA/CS lines with DIOR/DIOW.
- Low period in selx_profile register is used for the lo period of the DIOR/DIOW pulse.
- Hold of DA/CS with respect to DIOR/DIOW is always for one PCI clock.
- Recommended values for sel_we_hi and sel_we_lo for PIO-0 mode are 7 and 13, respectively (assuming a 33 MHz PCI clock).
- Recommended values for sel_we_hi and sel_we_lo for PIO-4 mode are 1 and 3 respectively.

- During DMA transactions the high period is used for the setup of the first transaction only.

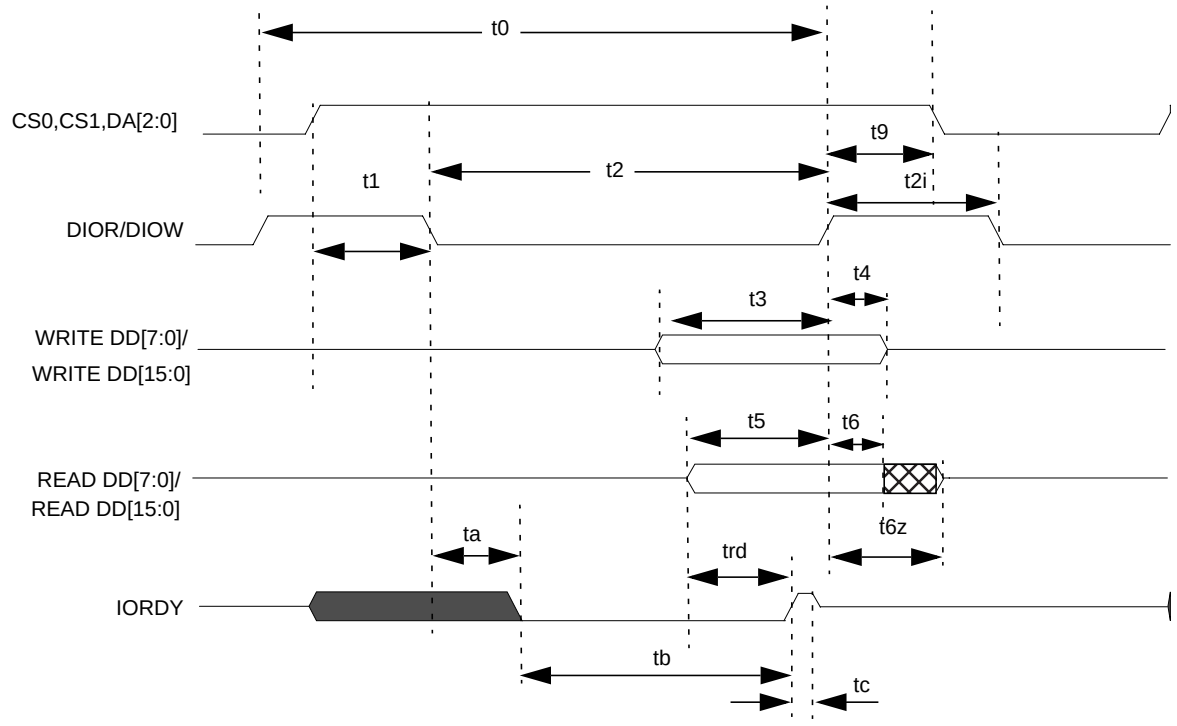


Figure 12: Register Transfer/PIO Data Transfer on IDE

Table 5: IDE Timing

PIO Timings (ATA-2 Spec)		Mode 0	Mode 4 (ns)
t0	Cycle time (min)	600	120
t1	ADD valid to DIOR/DIOW setup (min)	70	25
t2	DIOR/DIOW pulse width (min)	165	70
t2i	DIOR/DIOW Recovery time (min)	-	25
t3	DIOW data setup (min)	60	20
t4	DIOW data hold (min)	30	10
t5	DIOR data setup (min)	50	20
t6	DIOR data hold (min)	5	5
t6z	DIOR data tristate (max)	30	30
t9	DIOR/DIOW to add, cs hold (min)	20	10
trd	Read data valid to IORDY active (min)	0	0
ta	IORDY setup time (max)	35	35
tb	IORDY pulse width (max)	1250	1250
tc	IORDY assertion to release (max)	-	5

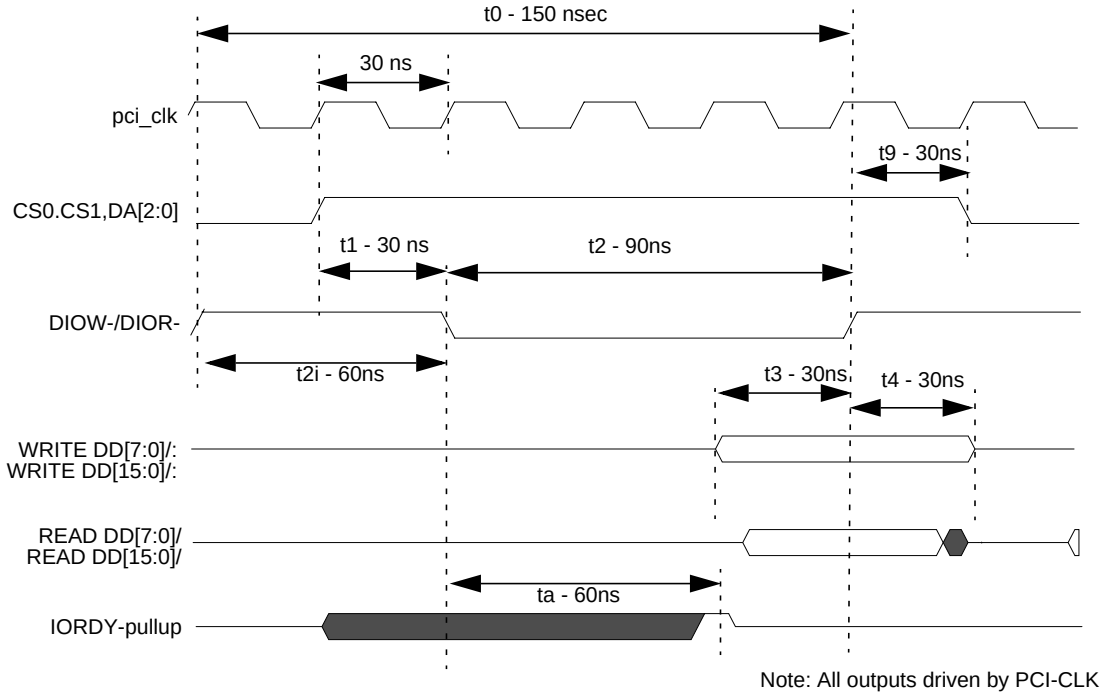


Figure 13: Timings on IDE Bus

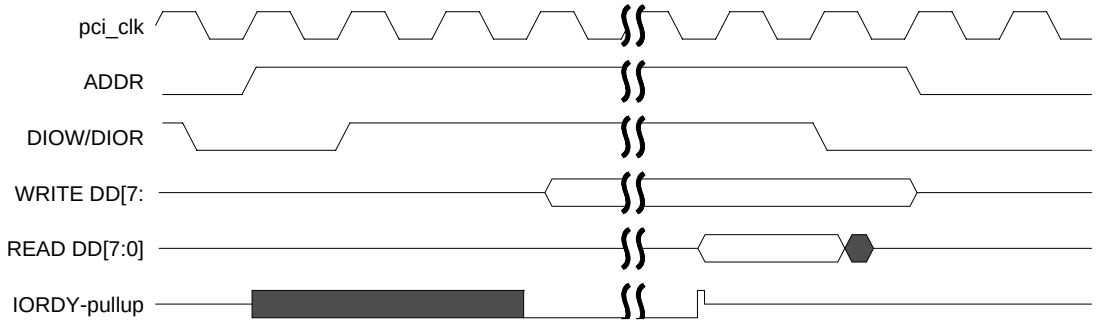


Figure 14: IDE Transaction, Flow Controlled by Device IORDY

3.2 PCI Interrupt Enable Register

The $\overline{\text{PCI\_INTA}}$ function is not implemented within the PCI module.

4. Application Notes

4.1 DTL Interface

The DTL side of the PCI module, Figure 1, consists of a single initiator and 4 targets. It supports both big and little-endian systems.

Features:

- Dedicated port for MMIO register access
- Dedicated port for direct access to XIO devices
- Dedicated port for PCI memory space
- Second PCI port which may be configured to access PCI memory or IO space
- Each port may be configured for posted or non-posted writes.
- Bursting to internal MMIO register space is not supported.
- The 2 PCI targets support “retry” on PCI for reads and non-posted single writes.

4.2 System Memory Bus Interface, the MTL Bus

To optimize PCI-to-system memory throughput in the PNX15xx Series system, a direct path is provided between PCI and the system memory bus using the MTL interface.

Features:

- For PCI burst reads, speculative read of user-selectable number of words is done from the memory.
- Two read and two write channels
- Continuous PCI write/read bursts can be sustained (contingent on availability of data on the DVP memory bus).

The memory interface has two registers that allow the interface to be tuned for optimum performance. A slave tuning register allows the user to select how much data will be prefetched from memory during reads. For mem_read commands, anywhere from 2 to 32 32-bit words may be selected. For mem_read_line commands, one cache line will be prefetched. And for mem_read_multiple, anywhere from 8 to 1024 32-bit words may be prefetched. A threshold is used to determine when additional data should be requested. This must be set to a value smaller then the smallest of the 3 prefetch sizes of the various read memory command types. Note that the cache line size must be set to a non-zero value before using cache line read commands.

The DMA read channel also has a prefetch size and threshold register. Improper settings of these registers combined with improper command type can result in an external master being starved for data. An example of this is when 2 masters are both attempting to do reads from the PCI.

The first is doing large burst with the memory-read command and the other single or burst reads. Since the memory read command is intended for relatively short bursts, only a small amount of data is prefetched. When it is nearly all consumed, additional data will be prefetched. While the data is being prefetched, the rule the additional data phases must complete within 8 clocks may come into play. This results in a disconnect on the first master. When the second master gets the GNT and attempts a read, it will be RETRIED since the internal state machine is busy with the prefetch of data requested by the first master. Now the first returns for a continuation of its read. When data runs low again, additional data will be prefetched, during which another disconnect occurs. This cycle may repeat until the first master has completed its entire burst.

4.3 XIO Interface

The XIO interface uses a part of the PCI interface and some additional signals to interface with external Flash (NAND and NOR types), NOR-ROM, IDE and Motorola devices. This function “steals” a PCI cycle and runs an XIO transfer using part of the PCI bus before giving control back to PCI. The XIO port may be accessed at any time after the configuration registers have been initialized. Up to five profiles may be enabled at one time. Each one requires a chip select. When 64 MB addressing is required, an extra pin (XIO_A[25]) is required with NOR flash and Motorola style devices. Flash profiles have a dedicated ACK pin to allow PCI transactions to continue while the device is busy.

4.3.1 Motorola Interface

In this XIO mode, any 8-bit or 16 bit Motorola 68360 type external slave can be addressed. For details about connecting a Motorola device to a PCI interface, please refer to Table 2. Even though the Motorola interface is an asynchronous interface, internal timings are generated in multiples of PCI clock. For programming to do Motorola cycles, please refer to XIO Sel_X Profile registers. For writes, data-strobe (DS) assertion time is made programmable by using sel0_we_hi field. There is an option to use the acknowledge from the device DSACK, or to have a fixed wait time before probing for read-data and removing DS for write-data.

4.3.2 NAND-Flash Interface

A flexible interface is provided to interface to a NAND-Flash. There are two registers that define the type of cycle that will be performed. The read and write strobes can be programmed independently with a high timer from one to four PCI clocks. A cycle may contain 0, 1, or 2 commands and 0, 1, 2, 3 or 4 address phases with or without data. Refer to Section 3.1.1 NAND-Flash Interface Operation for information on how to use this interface.

4.3.3 NOR Flash Interface

In this XIO mode, any 8-bit or 16-bit NOR flash can be addressed. Up to 64 MB may be addressed. The DS timing is programmable as is the WN timing. The user has the option of monitoring the R/BN signal from the flash or using a fixed response for the DS low timing.

4.3.4 IDE Interface

In this XIO mode, an IDE disk drive can be addressed. Only PIO mode is supported. The internal DMA engine can be programmed to perform data transfer to and from the IDE once the disk drive's registers have been programmed. The DIOR and DIOW strobe high and low times are programmable. Refer to Section 3.1.4 IDE Description for more details.

The IDE interface is internally grouped with 16bit XIO devices. This restricts the software in direct and indirect IDE register access to using 16 or 32 bit opcodes for writes and reads. These are mapped to a single write or read on accessing the IDE drive.

4.4 PCI Endian Support

The PCI module supports both big-endian and little-endian systems. The global system endian mode signal is used to determine which endian mode is in use.

4.5 General Notes

The cache line size register (PCI configuration register C) should be initialized to a non-zero value larger than the “slv_threshold” (Slave DTL tuning register) if using cache line read commands in the system. See note on recommended slv_threshold setting in the register description.

5. Register Descriptions

The following section describes the registers in the Document title variable block. The PCI configuration registers and the memory mapped IO registers are included.

5.1 Register Summary

Table 6: PCI-XIO Register Summary

Bit	Symbol	Description
0x0000—0x000C	Reserved	
0x04 0010	pci_setup	PCI Setup register
0x04 0014	pci_control	PCI Control register
0x04 0018	pci_base1_lo	Internal view of external PCI bottom address, 1st aperture
0x04 001C	pci_base1_hi	Internal view of external PCI top address, 1st aperture
0x04 0020	pci_base2_lo	Internal view of external PCI bottom address, 2nd aperture
0x04 0024	pci_base2_hi	Internal view of external PCI top address, 2nd aperture
0x04 0028	read_lifetime	Length of time data is held exclusively for requesting agent.
0x04 002C	gppm_addr	General purpose PCI Master Cycle address register
0x04 0030	gppm_wdata	General purpose PCI Master Cycle write data register
0x04 0034	gppm_rdata	General purpose PCI Master Cycle read data register
0x04 0038	gppm_ctrl	General purpose PCI Master Cycle control register
0x04 003C	unlock_register	Unlock pci_setup, class code, subsystem_ids
0x04 0040	device/vendorid	Image of device id and vendor id (config reg 00)
0x04 0044	config_cmd_stat	Image of configuration command and status register (config reg 04)
0x04 0048	class code/rev id	Image of class code and revision id (config reg 08)
0x04 004C	latency timer	Image of latency timer, cache line size (config reg 0C)
0x04 0050	base10	Image of configuration base address10 (config reg 10)
0x04 0054	base14	Image of configuration base address14 (config reg 14)
0x04 0058	base18	Image of configuration base address18 (config reg 18)
0x04 005C—0068	Reserved	
0x04 006C	subsystem ids	Subsystem id, subsystem vendor id (config reg 2C)
0x04 0070	Reserved	
0x04 0074	cap_pointer	Image of capabilities pointer (config reg 34)
0x04 0078	Reserved	
0x04 007C	config_misc	Image of interrupt line, and interrupt line registers (config reg 3C)
0x04 0080	pmc	Power management capabilities (config reg 40)
0x04 0084	pwr_state	Power Management control (config reg 44)
0x04 0088	pci_io	PCI IO properties
0x04 008C	slv_tuning	Slave DTL tuning
0x04 0090	dma_tuning	DMA DTL tuning
0x04 0094—07FC	Reserved	
0x04 0800	dma_eaddr	PCI address for DMA transaction
0x04 0804	dma_iaddr	Internal address for DMA transaction
0x04 0808	dma_length	DMA length in words
0x04 080C	dma_ctrl	DMA control

Table 6: PCI-XIO Register Summary *...Continued*

Bit	Symbol	Description
0x04 0810	xio_ctrl	XIO misc control
0x04 0814	xio_sel0_prof	XIO sel0 profile
0x04 0818	xio_sel1_prof	XIO sel1 profile
0x04 081C	xio_sel2_prof	XIO sel2 profile
0x04 0820	gpxio_addr	Indirect general purpose XIO address
0x04 0824	gpxio_wdata	Indirect general purpose XIO write data
0x04 0828	gpxio_rdata	Indirect general purpose XIO read data
0x04 082C	gpxio_ctrl	Indirect general purpose XIO control
0x04 0830	nand_ctrls	NAND-Flash profile controls
0x04 0834	xio_sel3_prof	XIO sel3 profile
0x04 0838	xio_sel4_prof	XIO sel4 profile
0x04 083C—0FAC	Reserved	
0x04 0FB0	gpxio_status	GPXIO Interrupt Status
0x04 0FB4	gpxio_int_mask	GPXIO Interrupt Enable
0x04 0FB8	gpxio_int_clr	GPXIO Interrupt Clear
0x04 0FBC	gpxio_int_set	GPXIO Interrupt Set
0x04 0FC0	gppm_status	GPPM Interrupt Status
0x04 0FC4	gppm_int_mask	GPPM Interrupt Enable
0x04 0FC8	gppm_int_clr	GPPM Interrupt Clear
0x04 0FCC	gppm_int_set	GPPM Interrupt Set
0x04 0FD0	dma_status	DMA Interrupt Status
0x04 0FD4	dma_int_mask	DMA Interrupt Enable
0x04 0FD8	dma_int_clr	DMA Interrupt Clear
0x04 0FDC	dma_int_set	DMA Interrupt Set
0x04 0FE0	pci_status	PCI Interrupt Status
0x04 0FE4	pci_int_mask	PCI Interrupt Enable
0x04 0FE8	pci_int_clr	PCI Interrupt Clear
0x04 0FEC	pci_int_set	PCI Interrupt Set
0x04 0FF0—0FF8	Reserved	
0x04 0FFC	module_id	Module ID

Remark: The PCI Configuration registers have no base address in the PNX15xx Series.

Table 7: PCI Configuration Register Summary

Bit	Symbol	Description
0x0000	Device / Vendor ID	Device ID and Vendor ID
0x0004	Command / Status	Command and Status register
0x0008	Class Code/Rev ID	Class code to be specified appropriate for the application. This will be implemented as a parameter. The Rev ID will initially be 0.

Table 7: PCI Configuration Register Summary *...Continued*

Bit	Symbol	Description
0x000C	Latency Timer/ Cache Line size	Latency Timer, Cache Line Size.
0x0010	Base Address 10	Base Address, memory
0x0014	Base Address 14	Base Address, memory — MMIO
0x0018	Base Address 18	Base Address, memory — XIO
0x001—0028	Reserved	
0x002C	Subsystem ID	Subsystem ID and Subsystem Vendor ID
0x0030	Reserved	
0x0034	Capability Pointer	Capabilities Pointer Register
0x0038	Reserved	
0x003C	INTR	Interrupt Line, Interrupt Pin, Min_Gnt, MAX_Lat
0x0040	pmc	Power management Capability
0x0044	pwr_state	Power Management control

The following table is a summary of all the registers in this module.

Table 8: Registers Description

Bit	Symbol	Access	Value	Description
PCI Control Registers				
This register must be initialized before any PCI cycles will be entertained. The boot loader is expected to load the values at boot time. Write once by boot loader, otherwise read only. Because this register is “written once” the bit fields are designated “R/W1.” An unlock is available to update this register if necessary. A write of “CA” to bits [7:0] of the unlock_setup register will allow one additional write to the setup register before locking again				
Offset 0x04 0010	***PCI Setup***			
31	Reserved	R	0	
30	dis_reqgnt	R/W1	0	Disable use of REQ/GNT when using internal arbiter. These pins may be released for other uses when using an internal arbiter and no external PCI masters are used in the system.
29	dis_reqgnt_a	R/W1	0	Disable use of REQ_A/GNT_A when using internal arbiter. These pins are not used when using an external harborer.
28	dis_reqgnt_b	R/W1	0	Disable use of REQ_B/GNT_B when using internal arbiter. These pins are not used when using an external arbiter.
27	d2_support	R/W1	1	Support for D2 power state
26	d1_support	R/W1	1	Support for D1 power state
25	Reserved	R/W1	0	
24	en_ta	R/W1	0	Terminate restricted access attempt with target abort (otherwise, ignore writes, return 0 on read).
23	en_pci2mmi	R/W1	1	Enable memory hwy interface.
22	en_xio	R/W1	1	Enable XIO functionality.
21	base18_prefetchable	R/W1	0	PCI base address 18 is a prefetchable memory aperture.

Table 8: Registers Description

Bit	Symbol	Access	Value	Description
20:18	base18_siz	R/W1	011	The size of aperture located by PCI cfg base18 is: 011 = 16 MB 100 = 32 MB 101 = 64 MB 110 = 128 MB This aperture is used as the XIO aperture in the PNX15xx Series. Note: If expanding to 128 MB, the default setting of base18 address will overlap with the default base14 address. To avoid an address conflict, the base18 address or the base14 address should be relocated before setting the base18_siz.
17	en_base18	R/W1	1	Enable 3rd aperture, PCI base address 18. The PNX15xx Series will always use this aperture.
16	base14_prefetchable	R/W1	0	PCI Base address 14 is a non-prefetchable memory aperture.
15	Reserved	R	0	
14:12	base14_siz	R/W1	000	The size of aperture located by PCI cfg base 14 is 000 = 2 MB. This aperture is used as the MMIO aperture in the PNX15xx Series.
11	en_base14	R/W1	1	Enable 2nd aperture, PCI base address 14. The PNX15xx Series will always use this aperture.
10	base10_prefetchable	R/W1	1	PCI Base address 10 is a prefetchable memory aperture.
9:7	base10_siz	R/W1	100	The size of aperture located by PCI cfg base 10 is: 011 = 16 MB 100 = 32 MB 101 = 64 MB 110 = 128 MB This aperture is used as the DRAM aperture in the PNX15xx Series.
6:2	Reserved			
1	en_config_manag	R/W1	1	Enable configuration management.
0	en_pci_arb	R/W1	0	Enable internal PCI system arbitration.
Offset 0x04 0014	***PCI Control***			
31:17	Reserved	R	0	
16	dis_swapper2targ	R/W	0	0 = Enable byte swapping in big endian mode from DCS to PCI path. 1 = Disable byte swapping in big endian mode from DCS to PCI path.
15	dis_swapper2intreg	R/W	0	0 = Enable byte swapping in big endian mode from PCI to PCI mmio registers. 1 = Disable byte swapping in big endian mode from PCI to PCI mmio registers.
14	dis_swapper2dtlinit	R/W	0	0 = Enable byte swapping in big endian mode from PCI to DCS. 1 = Disable byte swapping in big endian mode from PCI to DCS.
13	regs_wr_post_en	R/W	0	Enable write posting to internal PCI registers.
12	xio_wr_post_en	R/W	0	Enable write posting to XIO address range.
11	pci2_wr_post_en	R/W	0	Enable write posting to pci_base2 address range.

Table 8: Registers Description

Bit	Symbol	Access	Value	Description
10	pci1_wr_post_en	R/W	0	Enable write posting to pci_base1 address range.
9	en_serr_seen	R/W	0	Enable monitoring of the SERR pin.
8:7	Reserved	R	0	
6	en_base10_spec_rd	R/W	1	Read ahead to optimize PCI read latency to base 10.
5	en_base14_spec_rd	R/W	0	Read ahead to optimize PCI read latency to base 14.
4	en_base18_spec_rd	R/W	0	Read ahead to optimize PCI read latency to base 18.
3	disable_subword2_10	R/W	0	Disable subword access to/from Base10 aperture.
2	disable_subword2_14	R/W	1	Disable subword access to/from Base14 aperture.
1	disable_subword2_18	R/W	1	Disable subword access to/from Base18 aperture.
0	en_retry_timer	R/W	1	Enables timer for 16 tic rule enforcer. This bit does not affect access to the XIO aperture.
Offset 0x04 0018	***PCI_Base1_lo***			
31:21	pci_base1_lo	R/W	0	For internal address decoding: low bar of first aperture for external PCI access. This register affects the decode and routing of the bus controllers. It should not be relied on as stable for 10 clocks after writing. It is recommended that the PCI_Base1_lo be initialized before the PCI_Base1_hi to avoid a potentially large segment of address space being temporarily allocated to PCI space.
20:0	Reserved	R	0	
Offset 0x04 001C	***PCI_Base1_hi***			
31:21	pci_base1_hi	R/W	0	For internal address decoding: high bar of first aperture for external PCI access (up to but not including). This register affects the decode and routing of the bus controllers. It should not be relied on as stable for 10 clocks after writing. It is recommended the PCI_Base1_lo be initialized before the PCI_Base1_hi to avoid a potentially large segment of address space being temporarily allocated to PCI space.
20:0	Reserved	R	0	
Offset 0x04 0020	***PCI_Base2_lo***			
31:21	pci_base2_lo	R/W	0	For internal address decoding: low bar of second aperture for external PCI access. This register affects the decode and routing of the bus controllers. It should not be relied on as stable for 10 clocks after writing. It is recommended the PCI_Base2_lo be initialized before the PCI_Base2_hi to avoid a potentially large segment of address space being temporarily allocated to PCI space. The PCI_Base2 aperture may be declared as a internal view of PCI IO space or as PCI memory space. See pci_io register for more information.
20:0	Reserved	R	0	

Table 8: Registers Description

Bit	Symbol	Access	Value	Description
Offset 0x04 0024	***PCI_Base2_hi***			
31:21	pci_base2_hi	R/W	0	For internal address decoding: high bar of second aperture for external PCI access (up to but not including). This register affects the decode and routing of the bus controllers. It should not be relied on as stable for 10 clocks after writing. It is recommended the PCI_Base2_lo be initialized before the PCI_Base2_hi to avoid a potentially large segment of address space being temporarily allocated to PCI space. The PCI_Base2 aperture may be declared as a internal view of PCI IO space or as PCI memory space. See pci_io register for more information.
20:0	Reserved	R	0	
Offset 0x04 0028	***Read Data Lifetime Timer***			
31:16	Unused		-	
15:0	read_lifetime	R/W	8000	This register is the amount of time (in PCI clocks) the PCI will hold a piece of data exclusively for an external PCI master. The timer is initiated when the PCI can not complete the requested read in 16 clock cycles and issues a retry.
Offset 0x04 002C	***General Purpose PCI Master (GPPM) Address***			
31:0	gppm_addr	R/W	0	This register will be written with the address for the single data phase cycle to be issued on the PCI bus. It will accept only 32-bit writes. When issuing type 0 configuration transactions, the device number (bits [15:11]) is expanded to bits [31:11] on the PCI bus as defined in the PCI 2.2 spec.
Offset 0x04 0030	***General Purpose PCI Master (GPPM) Write Data***			
31:0	gppm_wdata	R/W	0	This register will be written with the data for the single data phase cycle to be issued on the PCI bus. This register will accept any size write.
Offset 0x04 0034	***General Purpose PCI Master (GPPM) Read Data***			
31:0	gppm_rdata	R	0	This register will hold data from the selected target after completion of the read.
Offset 0x04 0038	***General Purpose PCI Master (GPPM) Control***			
31:11	Reserved	R	0	
10	gppm_done	R	0	1 = cycle has completed. This bit can also be viewed in the gppm_status register. Write to register 0x40FC8 to clear.
9	init_pci_cycle	R/W	0	1 = initiate a PCI single data phase transaction on the PCI bus with address “gppm_addr” and data “gppm_data.”
8	Reserved	R	0	
7:4	gppm_cmd	R/W	0	Command to be used with PCI cycle. Acceptable commands to use in the command field include IO read, IO write, memory Read, memory Write, configuration read and interrupt acknowledge. If configuration management is enabled, configuration write may be used.
3:0	gppm_ben	R/W	0	Byte enables to be used with PCI cycle
Offset 0x04 003C	***Unlock Register***			
31:16	Reserved	R	0	

Table 8: Registers Description

Bit	Symbol	Access	Value	Description
15:8	unlock_ssid	W	0	Writing a “0xCA” to this field will unlock the “subsystem_id” and “subsystem_vendor” registers. A writer to the subsystem_id/ subsystemvendor” register will lock the register again.
7:0	unlock_setup	W	0	Writing a “0xCA” to this field will unlock the “classcode”, “max_latency”, “min_gnt” and “pci_setup” registers. A write to the “pci_setup” register to lock registers again.
Offset 0x04 0040	***Image of Device ID and Vendor ID***			
31:16	device_id	R	0x5405	PCI configuration device ID
15:0	vendor_id	R	0x1131	PCI configuration vendor ID
Offset 0x04 0044	***Image of Command/Status***			
31:16	status	R	0x0290	PCI configuration status register
15:0	command	R/W*	0x0000	PCI configuration command register. *This register is read/write if configuration management is enabled (pci_setup[1]). If not enabled, it is read only. Refer to configuration register 4 for details on which bits are implemented and controllable.
Offset 0x04 0048	***Image of Class Code/Revision ID***			
31:8	class code	R/W*	048000	PCI configuration class code. *Write-once/Read-only
7:0	revision id	R	1	PCI configuration revision ID
Offset 0x04 004C	***Image of Latency Timer/Cache Line Size***			
31:24	BIST	R	0	PCI configuration BIST
23:16	Header Type	R	0	PCI configuration Header Type
15:8	latency timer	R/W*	0	PCI configuration latency timer. *This register is read/write if configuration management is enabled (pci_setup[1]). If not enabled, it is read only.
7:0	cache line size	R/W*	0	PCI configuration cache line size. *This register is read/write if configuration management is enabled (pci_setup[1]). If not enabled, it is read only.
Offset 0x04 0050	***Base Address 10 Image***			
31:21	Base Address 10	R/W*	0	PCI configuration Base address for DRAM. This register affects the decode and routing of the bus controllers. It should not be relied on as stable for 10 clocks after writing. *This register is read/write if configuration management is enabled (pci_setup[1]). If not enabled, it is read only.
20:4	Reserved	R	0	
3	Prefetchable	R	cfg*	*Value is determined at boot time by pci_setup register.
2:0	Type	R	0	Indicates type 0 memory space (locatable anywhere in 32-bit address space).
Offset 0x04 0054	***Base Address 14 Image***			
31:4	Base Address 14	R/W*	1BE00000	PCI configuration Base address for MMIO. This register affects the decode and routing of the bus controllers. It should not be relied on as stable for 10 clocks after writing. *This register is read/write if configuration management is enabled (pci_setup[1]). If not enabled, it is read only.

Table 8: Registers Description

Bit	Symbol	Access	Value	Description
3	Prefetchable	R	cfg*	*Value is determined at boot time by pci_setup register.
2:0	Type	R	0	Indicates type 0 memory space (locatable anywhere in 32-bit address space).
Offset 0x04 0058	***Base Address 18 Image***			
31:4	Base Address 18	R/W*	1C00000	PCI configuration Base address for XIO. This register affects the decode and routing of the bus controllers. It should not be relied on as stable for 10 clocks after writing. *This register is read/write if configuration management is enabled (pci_setup[1]). If not enabled, it is read only.
3	Prefetchable	R	cfg*	*Value is determined at boot time by pci_setup register.
2:0	Type	R	0	Indicates PCI “type 0” memory space (locatable anywhere in 32-bit address space).
Offset 0x04 006C	***Subsystem ID/Subsystem Vendor ID Write Port***			
This register must be initialized before any PCI cycles will be entertained. The boot loader is expected to load the values at boot time. This register is a Write-once/Read-only register (R/W1).				
31:16	subsystem ID	R/W1	0	This is the write port for the Subsystem ID (PCI config 2C).
15:0	subsystem vendor ID	R/W1	0	This is the write port for the Subsystem Vendor ID (PCI config 2C).
Offset 0x04 0074	***Image of Configuration Reg 34***			
31:8	Reserved	R	0	
7:0	CAP_PTR	R	40	Capabilities Pointer
Offset 0x04 007C	***Image of Configuration Reg 3C***			
31:24	max_lat	R/W1	0x18	Max Latency
23:16	min_gnt	R/W1	0x09	Minimum Grant
15:8	interrupt pin	R	0x01	Interrupt pin information
7:0	Interrupt Line	R/W*	0x00	This register conveys interrupt line routing information. *This register is read/write if configuration management is enabled (pci_setup[1]). If not enabled, it is read only.
Offset 0x04 0080	***Image of Configuration Reg 40***			
31:27	Reserved	R	00000	
26	d2_support	R	cfg*	1 = Device supports D2 power management state. *Value is determined by pci_setup register.
25	d1_support	R	cfg*	1 = Device supports D1 power management state. *Value is determined by pci_setup register.
24:19	Reserved	R	0	
18:16	version	R	010	Indicates compliance with version 1.1 of PM.
15:8	Next Item Pointer	R	00	There are no other extended capabilities.
7:0	Cap_ID	R	01	Indicates this is power management data structure.
Offset 0x04 0084	***Image of Configuration Reg 44***			
31:1	Reserved	R	0	
1:0	pwr_state	R/W*	0	Power State *This register is read/write if configuration management is enabled (pci_setup[1]). If not enabled, it is read only.

Table 8: Registers Description

Bit	Symbol	Access	Value	Description
Offset 0x04 0088 PCI_IO				
31:24	upper_io3_addr	R/W	0	Bits [31:24] of IO address during PCI IO transactions.
23:16	upper_io2_addr	R/W	0	Bits [23:16] of IO address during PCI IO transactions.
15:3	Reserved	R	0	
2	use_io3_addr	R/W	0	Use “upper_io3_addr” as the upper address for PCI IO transactions.
1	use_io2_addr	R/W	0	Use “upper_io3_addr” and “upper_io2_addr” as the upper address for PCI IO transactions.
0	use_pcibase2_as_io	R/W	0	1: PCI_Base2 will forward PCI2 DTL transactions to PCI bus as IO transactions. The address will unchanged or modified with an alternate upper addresses selected above. 0: PCI_Base2 will forward PCI2 DTL transactions to PCI bus as memory transactions with unchanged address.
Offset 0x04 008C Slave DTL tuning				
31:24	Reserved	R	0	
20:16	slv_memrd_fetch	R/W	111	PCI slave DTL read block size for memory read command. Default value is 8 32-bit words. Maximum is 64 32-bit words.
11:8	slv_threshold	R/W	10	Threshold (amount of data not consumed from previous read request) for when PCI slave DTL requests more read data when responding to memory read command. This must be set to a value less than the smallest of slv_memrd_fetch, Cache Line Size or read_block_siz. Default is 3 32-bit words. Maximum value is 32 32-bit words.
7:3	Reserved	R	0	
2:0	slv_mrmul_fetch	R/W	001	Encoded PCI slave DTL read block size for memory read multiple command siz : read_block_siz 000: 8 bytes 001: 16 bytes 010: 32 bytes 011: 64 bytes 100: 128 bytes 101: 256 bytes 110: 512 bytes 111: 1024 bytes
Offset 0x04 0090 DMA DTL tuning				
31:16	Reserved	R	0	
15:8	dma_threshold	R/W	0x1B	Threshold for when DMA DTL requests more read data when initial fetch is less than total dma length.
7:3	Reserved	R	0	

Table 8: Registers Description

Bit	Symbol	Access	Value	Description
2:0	dma_fetch	R/W	010	Encoded DMA DTL read block size siz read_block_siz 000: 8 bytes 001: 16 bytes 010: 32 bytes 011: 64 bytes 100: 128 bytes 101: 256 bytes 110: 512 bytes 111: 1024 bytes
Offset 0x04 0094—07FC	***Reserved***			
Offset 0x04 0800	***DMA PCI Address***			
This register will accept only word writes.				
31:0	dma_eaddr	R/W	1C00_0000	This is the external starting address for the DMA engine. It is used for DMA transfers over PCI and XIO. Bit 0 and 1 are not used because all DMA transfers are word aligned.
Offset 0x04 0804	***DMA Internal Address***			
This register will accept only word writes.				
31:0	dma_iaddr	R/W	0010_0000	This is the internal read source/ write destination address in SDRAM.
Offset 0x04 0808	***DMA Transfer Size***			
This register will accept any size writes.				
31:16	Reserved	R/W	0	
15:0	dma_length	R/W	800	This is the length of the DMA transfer (number of 4-byte words).
Offset 0x04 080C	***DMA Controls***			
This register will accept any size writes.				
31:11	Reserved	R	0	
10	single_data_phase	R/W	0	1 = Limit DMA to single data phase transactions. This overrides “max_burst_size.” 0 = Use max_burst_size to determine burst size.
9	snd2xio	R/W	0	0 = DMA will target PCI. 1 = DMA will target XIO.
8	fix_addr	R/W	0	0 = DMA will use linear address. 1 = DMA will use a fixed address.

Table 8: Registers Description

Bit	Symbol	Access	Value	Description
7:5	max_burst_size	R/W	0	PCI transaction will be split into multiple transactions. Max size: 000 = 8 data phase 001 = 16 data phase 010 = 32 data phase 011 = 64 data phase 100 = 128 data phase 101 = 256 data phase 110 = 512 data phase 111 = No restriction in transfer length
4	init_dma	R/W	0	Initiate DMA transaction. This bit is cleared by the DMA engine when it begins its operation.
3:0	cmd_type	R/W	0	Command to be used for DMA. This field is restricted to memory type or IO type commands as defined in the PCI 2.2 spec.
Offset 0x04 0810	***XIO Control Register***			
31:2	Reserved	R	0	
1	xio_ack	R		Live XIO_ACK status bit.
0	Reserved	R	0	
Offset 0x04 0814	***XIO Sel0 Profile***			
This register sets up the profile of the XIO select 0 line. All times are in reference to PCI clocks.				
31:25	Reserved	R	0	
24	misc_ctrl	R/W	0	68360: 1 synchronous DSACK; 0 asynchronous DSACK. NOR: Not used NAND: Not used IDE: Not used
23	en_16bit_xio	R/W	0	0 = 8 bit XIO device 1 = 16 bit XIO device
22	sel0_use_ack	R/W	0	0 = Fixed wait state 1 = Wait for ACK Not used for IDE.
21:18	sel0_we_hi	R/W	0	68360: DS time high. NOR: WN time high NAND: REN profile, [19:18] low time; [21:20] high time IDE: DIOR and DIOW high time
17:14	sel0_we_lo	R/W	0	68360: Not used. NOR: WN time low NAND: WEN profile, [15:14] low time; [17:16] high time IDE: DIOR and DIOW low time
13:9	sel0_wait	R/W	0	68360: DS time low if using fixed timing. NOR: OEN time low if not using ACK. NAND: Delay between address and data phase if not using ACK, delay until monitoring ACK. IDE: Not used.
8:5	sel0_offset	R/W	0	Starting address offset from start address of XIO aperture, in 8M increments. This field must be naturally aligned with the size of the profile.

Table 8: Registers Description

Bit	Symbol	Access	Value	Description
4:3	sel0_type	R/W	0	Device type selected: 00 = 68360 type device 01 = NOR Flash 10 = NAND-Flash 11 = IDE
2:1	sel0_siz	R/W	0	Amount of address space allocated to Sel0: 00 = 8M 01 = 16M 10 = 32M 11 = 64M
0	en_sel0	R/W	0	1 = Enable sel0 profile.
Offset 0x04 0818	***XIO Sel1 Profile***			
This register sets up the profile of the XIO select 1 line. All times are in reference to PCI clocks.				
31:25	Reserved	R	0	
24	misc_ctrl	R/W	0	68360: 1 synchronous DSACK; 0 asynchronous DSACK. NOR: Not used NAND: Not used IDE: Not used
23	en_16bit_xio	R/W	0	0 = 8 bit XIO device 1 = 16 bit XIO device
22	sel1_use_ack	R/W	0	1 = Wait for ACK 0 = fixed wait state. Not used for IDE.
21:18	sel1_we_hi	R/W	0	63860: time high. NOR: WN time high NAND: REN profile, [19:18] low time; [21:20] high time IDE: DIOR and DIOW high time
17:14	sel1_we_lo	R/W	0	63860: Not used. NOR: WN time low NAND: WEN profile, [15:14] low time; [17:16] high time IDE: DIOR and DIOW low time
13:9	sel1_wait	R/W	0	63860: DS time low if using fixed timing. NOR: OEN time low if not using ACK. NAND: Delay between address and data phase if not using ACK, delay until monitoring ACK. IDE: Not used.
8:5	sel1_offset	R/W	0	Address offset form start address of XIO aperture, in 8M increments. This field must be naturally aligned with the size of the profile.
4:3	sel1_type	R/W	0	Sel1 is configured as: 00 = 68360 type device 01 = NOR Flash 10 = NAND-Flash 11 = IDE

Table 8: Registers Description

Bit	Symbol	Access	Value	Description
2:1	sel1_siz	R/W	0	Amount of address space allocated to Sel1: 00 = 8M 01 = 16M 10 = 32M 11 = 64M
0	en_sel1	R/W	0	Enable sel1 profile.
Offset 0x04 081C	***XIO Sel2 Profile***			
This register sets up the profile of the XIO select 2 line. All times are in reference to PCI clocks.				
31:25	Reserved	R	0	
24	misc_ctrl	R/W	0	68360: 1 synchronous DSACK; 0 asynchronous DSACK. NOR: Not used NAND: Not used IDE: Not used
23	en_16bit_xio	R/W	0	0 = 8 bit XIO device 1 = 16 bit XIO device
22	sel2_use_ack	R/W	0	0 = Fixed wait state. 1 = Wait for ACK Not used for IDE
21:18	sel2_we_hi	R/W	0	68360: DS time high. NOR: WN time high NAND: REN profile, [19:18] low time; [21:20] high time IDE: DIOR and DIOW high time
17:14	sel2_we_lo	R/W	0	63860: Not used. NOR: WN time low NAND: WEN profile, [15:14] low time; [17:16] high time IDE: DIOR and DIOW low time
13:9	sel2_wait	R/W	0	68360: DS time low if using fixed timing. NOR: OEN time low if not using ACK. NAND: Delay between address and data phase if not using ACK, delay until monitoring ACK. IDE: Not used.
8:5	sel2_offset	R/W	0	Address offset form start address of XIO aperture, in 8M increments. This field must be naturally aligned with the size of the profile.
4:3	sel2_type	R/W		Sel2 is configured as: 00 = 68360 type device 01 = NOR Flash 10 = NAND-Flash 11 = IDE
2:1	sel2_siz	R/W	0	Amount of address space allocated to Sel2: 00 = 8M 01 = 16M 10 = 32M 11 = 64M
0	en_sel2	R/W	0	Enable sel2 profile.

Table 8: Registers Description

Bit	Symbol	Access	Value	Description
Offset 0x04 0820	***GPXIO_address***			
31:0	gpxio_addr	R/W	0	General Purpose XIO cycle address. This register sets the address for an indirect read or write to/from XIO address space. Only 4 byte writes are allowed in this register. The values programmed for bits 0 and 1 are not used by the XIO module. Refer to gpxio_ben.
Offset 0x04 0824	***GPXIO_write_data***			
31:0	gpxio_wdata	R/W	0	General Purpose XIO cycle data. This register is programmed with data for a write cycle.
Offset 0x04 0828	***GPXIO_read_data***			
31:0	gpxio_rdata	R	0	General Purpose XIO cycle data. This register contains the data of a read cycle after completion.
Offset 0x04 082C	***GPXIO_ctrl***			
This register controls the type of access to XIO and provides status.				
31:10	Reserved	R	0	
9	gpxio_cyc_pending	R	0	1 = GPXIO transaction on XIO is pending. 0 = GPXIO has completed or not yet started.
8	gpxio_done	R	0	General Purpose XIO cycle complete. This bit is cleared by writing 1 to bit 6 or 7. It will also be cleared by writing to the GPXIO interrupt clear register.
7	clr_gpxio_done	W	0	1 = Clear “gpxio_done.”
6	gpxio_init	R/W	0	1 = Initiate a transaction on XIO. The type of transaction will match the profile of the selected aperture. This bit gets cleared if the cycle has been initiated. This bit clears bit 8 if set.
5	Reserved	R	0	
4	gpxio_rd	R/W	0	1 = Read command on XIO 0 = Write command on XIO
3:0	gpxio_ben	R/W	0	Active low byte enables to be used on the indirect XIO cycle. These are used to determine how many bytes to access and the lower two address bits for use in “gpxio_addr”.
Offset 0x04 0830	***NAND-Flash controls***			
31:22	Reserved			
21:16	nand_ctrls	R/W	17	This field controls the type of NAND-Flash access cycle. The bits are defined as follows: [21]: 1= 64-MB device support; 0 = 32 MB and smaller device support [20]: 1 = Include data in access cycle; 0 access does not include data phase(s) [19:18] = No. of commands to be used in NAND-Flash access [17:16] = No. of address phases to be used in NAND-Flash access. For 64-MB devices, 11 provide four address phases and 10 provide three address phases.
15:8	command_b	R/W	0	This is the second command for NAND-Flash when two commands are required to complete a cycle.
7:0	command_a	R/W	0	This is the command type to be used with NAND-Flash cycles when one or more commands are required to complete a cycle.

Table 8: Registers Description

Bit	Symbol	Access	Value	Description
Offset 0x04 0834	***XIO Sel3 Profile***			
This register sets up the profile of the XIO select 3 line. All times are in reference to PCI clocks.				
31:25	Reserved	R	0	
24	misc_ctrl	R/W	0	68360: 1 synchronous DSACK; 0 asynchronous DSACK. NOR: Not used NAND: Not used IDE: Not used
23	en_16bit_xio	R/W	0	0 = 8 bit XIO device 1 = 16 bit XIO device
22	sel3_use_ack	R/W	0	0 = Fixed wait state 1 = Wait for ACK Not used for IDE.
21:18	sel3_we_hi	R/W	0	68360: DS time high. NOR: WN time high NAND: REN profile, [19:18] low time; [21:20] high time IDE: DIOR and DIOW high time
17:14	sel3_we_lo	R/W	0	68360: Not used. NOR: WN time low NAND: WEN profile, [15:14] low time; [17:16] high time IDE: DIOR and DIOW low time
13:9	sel3_wait	R/W	0	68360: DS time low if using fixed timing. NOR: OEN time low if not using ACK. NAND: Delay between address and data phase if not using ACK, delay until monitoring ACK. IDE: Not used.
8:5	sel3_offset	R/W	0	Starting address offset from start address of XIO aperture, in 8M increments. This field must be naturally aligned with the size of the profile.
4:3	sel3_type	R/W	0	Device type selected: 00 = 68360 type device 01 = NOR Flash 10 = NAND Flash 11 = IDE
2:1	sel3_siz	R/W	0	Amount of address space allocated to Sel3: 00 = 8M 01 = 16M 10 = 32M 11 = 64M
0	en_sel3	R/W	0	1 = Enable sel3 profile
Offset 0x04 0838	***XIO Sel4 Profile***			
This register sets up the profile of the XIO select 4line. All times are in reference to PCI clocks.				
31:25	Reserved	R	0	
24	misc_ctrl	R/W	0	68360: 1 synchronous DSACK; 0 asynchronous DSACK. NOR: Not used NAND: Not used IDE: Not used

Table 8: Registers Description

Bit	Symbol	Access	Value	Description
23	en_16bit_xio	R/W	0	0 = 8 bit XIO device 1 = 16 bit XIO device
22	sel4_use_ack	R/W	0	0 = Fixed wait state 1 = Wait for ACK Not used for IDE.
21:18	sel4_we_hi	R/W	0	68360: DS time high. NOR: WN time high NAND: REN profile, [19:18] low time; [21:20] high time IDE: DIOR and DIOW high time
17:14	sel4_we_lo	R/W	0	68360: Not used. NOR: WN time low NAND: WEN profile, [15:14] low time; [17:16] high time IDE: DIOR and DIOW low time
13:9	sel4_wait	R/W	0	68360: DS time low if using fixed timing. NOR: OEN time low if not using ACK. NAND: Delay between address and data phase if not using ACK, delay until monitoring ACK. IDE: Not used.
8:5	sel4_offset	R/W	0	Starting address offset from start address of XIO aperture, in 8M increments. This field must be naturally aligned with the size of the profile.
4:3	sel4_type	R/W	0	Device type selected: 00 = 68360 type device 01 = NOR Flash 10 = NAND Flash 11 = IDE
2:1	sel4_siz	R/W	0	Amount of address space allocated to Sel4: 00 = 8M 01 = 16M 10 = 32M 11 = 64M
0	en_sel4	R/W	0	1 = Enable sel4 profile
Offset 0x04 0FB0	***GPXIO Interrupt Status***			
31:15	Reserved	R	0	
14	gpxio_xio_ack_done	R	0	Rising edge of xio_ack has been observed
13	gpxio_done	R	0	GPXIO transaction completed
12:10	Reserved	R	0	
9	gpxio_err	R	0	Non-supported GPXIO command attempted or not enabled
8:3	Reserved	R	0	
2	gpxio_r_mabort	R	0	GPXIO Received Master Abort
1:0	Reserved	R	0	
Offset 0x04 0FB4	***GPXIO Interrupt Enable***			
31:15	Reserved	R	0	

Table 8: Registers Description

Bit	Symbol	Access	Value	Description
14	en_int_gpxio_xio_ack_done	R	0	Enable Interrupt on rising edge of xio_ack has been observed
13	en_int_gpxio_done	R	0	Enable Interrupt on GPXIO transaction completed
12:10	Reserved	R	0	
9	en_int_gpxio_err	R	0	Enable Interrupt on non-supported GPXIO command attempted or not enabled
8:3	Reserved	R	0	
2	en_int_gpxio_r_mabort	R	0	Enable Interrupt on GPXIO Received Master Abort
1:0	Reserved	R	0	
Offset 0x04 0FB8	***GPXIO Interrupt Clear***			
31:15	Reserved	R	0	
14	clr_gpxio_xio_ack_done	R	0	Clear rising edge of xio_ack has been observed
13	clr_gpxio_done	R	0	Clear GPXIO transaction completed
12:10	Reserved	R	0	
9	clr_gpxio_err	R	0	Clear non-supported GPXIO command attempted or not enabled
8:3	Reserved	R	0	
2	clr_gpxio_r_mabort	R	0	Clear GPXIO Received Master Abort
1:0	Reserved	R	0	
Offset 0x04 0FBC	***GPXIO Interrupt Set***			
31:15	Reserved	R	0	
14	set_gpxio_xio_ack_done	R	0	Set rising edge of xio_ack has been observed
13	set_gpxio_done	R	0	Set GPXIO transaction completed
12:10	Reserved	R	0	
9	set_gpxio_err	R	0	Set non-supported GPXIO command attempted or not enabled
8:3	Reserved	R	0	
2	set_gpxio_r_mabort	R	0	Set GPXIO Received Master Abort
1:0	Reserved	R	0	
Offset 0x04 0FC0	***GPPM Interrupt Status***			
31:11	Reserved	R	0	
10	gppm_done	R	0	GPPM transaction completed
9	gppm_err	R	0	Non-supported GPPM command attempted or not enabled
8:6	Reserved	R	0	
5	gppm_mstr_parity_err	R	0	GPPM master set or observed parity error (PERR)
4	gppm_err_parity	R	0	GPPM Detected parity error (PERR)
3	Reserved	R	0	
2	gppm_r_mabort	R	0	GPPM Received Master Abort
1	gppm_r_tabor	R	0	GPPM Received Target Abort
0	Reserved	R	0	

Table 8: Registers Description

Bit	Symbol	Access	Value	Description
Offset 0x04 0FC4	***GPPM Interrupt Enable***			
31:11	Reserved	R	0	
10	en_int_gppm_done	R	0	GPPM transaction completed
9	en_int_gppm_err	R	0	Non-supported GPPM command attempted or not enabled
8:6	Reserved	R	0	
5	en_int_gppm_mstr_parity err	R	0	GPPM master set or observed parity error (PERR)
4	en_int_gppm_err_parity	R	0	GPPM Detected parity error (PERR)
3	Reserved	R	0	
2	en_int_gppm_r_mabort	R	0	GPPM Received Master Abort
1	en_int_gppm_r_tabor	R	0	GPPM Received Target Abort
0	Reserved	R	0	
Offset 0x04 0FC8	***GPPM Interrupt Clear***			
31:11	Reserved	R	0	
10	clr_gppm_done	R	0	Clear GPPM transaction completed
9	clr_gppm_err	R	0	Clear non-supported GPPM command attempted or not enabled
8:6	Reserved	R	0	
5	clr_gppm_mstr_parity err	R	0	Clear GPPM master set or observed parity error (PERR)
4	clr_gppm_err_parity	R	0	Clear GPPM Detected parity error (PERR)
3	Reserved	R	0	
2	clr_gppm_r_mabort	R	0	Clear GPPM Received Master Abort
1	clr_gppm_r_tabor	R	0	Clear GPPM Received Target Abort
0	Reserved	R	0	
Offset 0x04 0FCC	***GPPM Interrupt Set***			
31:11	Reserved	R	0	
10	set_gppm_done	R	0	Set GPPM transaction completed
9	set_gppm_err	R	0	Set non-supported GPPM command attempted or not enabled
8:6	Reserved	R	0	
5	set_gppm_mstr_parity_err	R	0	Set GPPM master set or observed parity error (PERR)
4	set_gppm_err_parity	R	0	Set GPPM Detected parity error (PERR)
3	Reserved	R	0	
2	set_gppm_r_mabort	R	0	Set GPPM Received Master Abort
1	set_gppm_r_tabor	R	0	Set GPPM Received Target Abort
0	Reserved	R	0	
Offset 0x04 0FD0	***DMA Interrupt Status***			
31:15	Reserved	R	0	
14	dma_xio_ack_done	R	0	Rising edge of xio_ack has been observed

Table 8: Registers Description

Bit	Symbol	Access	Value	Description
13	Reserved	R	0	
12	dma_done	R	0	DMA transaction completed
11:10	Reserved	R	0	
9	dma_err	R	0	Non-supported DMA command attempted or not enabled
8:6	Reserved	R	0	
5	dma_mstr_parity_err	R	0	DMA master set or observed parity error (PERR)
4	dma_err_parity	R	0	DMA Detected parity error (PERR)
3	Reserved	R	0	
2	dma_r_mabort	R	0	DMA Received Master Abort
1	dma_r_tabor	R	0	DMA Received Target Abort
0	Reserved	R	0	
Offset 0x04 0FD4	***DMA Interrupt Enable***			
31:15	Reserved	R	0	
14	en_int_dma_xio_ack_done	R	0	Rising edge of xio_ack has been observed
13	Reserved	R	0	
12	en_int_dma_done	R	0	DMA transaction completed
11:10	Reserved	R	0	
9	en_int_dma_err	R	0	Non-supported DMA command attempted or not enabled
8:6	Reserved	R	0	
5	en_int_dma_mstr_parity_err	R	0	DMA master set or observed parity error (PERR)
4	en_int_dma_err_parity	R	0	DMA Detected parity error (PERR)
3	Reserved	R	0	
2	en_int_dma_r_mabort	R	0	DMA Received Master Abort
1	en_int_dma_r_tabor	R	0	DMA Received Target Abort
0	Reserved	R	0	
Offset 0x04 0FD8	***DMA Interrupt Clear***			
31:15	Reserved	R	0	
14	clr_dma_xio_ack_done	R	0	Rising edge of xio_ack has been observed
13	Reserved	R	0	
12	clr_dma_done	R	0	Clear DMA transaction completed
11:10	Reserved	R	0	
9	clr_dma_err	R	0	Clear non-supported DMA command attempted or not enabled
8:6	Reserved	R	0	
5	clr_dma_mstr_parity_err	R	0	Clear DMA master set or observed parity error (PERR)
4	clr_dma_err_parity	R	0	Clear DMA Detected parity error (PERR)
3	Reserved	R	0	

Table 8: Registers Description

Bit	Symbol	Access	Value	Description
2	clr_dma_r_mabort	R	0	Clear DMA Received Master Abort
1	clr_dma_r_tabor	R	0	Clear DMA Received Target Abort
0	Reserved	R	0	
Offset 0x04 0FDC DMA Interrupt Set				
31:15	Reserved	R	0	
14	set_dma_xio_ack_done	R	0	Set Rising edge of xio_ack has been observed
13	Reserved	R	0	
12	set_dma_done	R	0	Set DMA transaction completed
11:10	Reserved	R	0	
9	set_dma_err	R	0	Set Non-Supported DMA command attempted or not enabled
8:6	Reserved	R	0	
5	set_dma_mstr_parity_err	R	0	Set DMA master set or observed parity error (PERR)
4	set_dma_err_parity	R	0	Set DMA Detected parity error (PERR)
3	Reserved	R	0	
2	set_dma_r_mabort	R	0	Set DMA Received Master Abort
1	set_dma_r_tabor	R	0	Set DMA Received Target Abort
0	Reserved	R	0	
Offset 0x04 0FE0 PCI Interrupt Status				
This register represents the status of direct access to Document title variable and PCI slave events.				
31:27	Reserved	R	0	
26	pcii_wr_err	R	0	Interrupt on PCI DTL initiator write error flag
25	pcii_rd_err	R	0	Interrupt on PCI DTL initiator read error flag
24	xio_wr_err	R	0	Interrupt on XIO DTL target write error flag
23	xio_rd_err	R	0	Interrupt on XIO DTL target read error flag
22	pcir_wr_err	R	0	Interrupt on mmio register DTL target write error flag
21	pcir_rd_err	R	0	Interrupt on mmio register DTL target read error
20	pwrstate_chg	R	0	Power management register has been changed
19	Reserved	R	0	
18	pci2_wr_err	R	0	Interrupt on PCI2 DTL target write error flag
17	pci2_rd_err	R	0	Interrupt on PCI2 DTL target read error flag
16	pci1_wr_err	R	0	Interrupt on PCI1 DTL target write error flag
15	pci1_rd_err	R	0	Interrupt on PCI1 DTL target read error flag
14	pci_xio_ack_done	R	0	Rising edge of xio_ack has been observed
13:12	Reserved	R	0	
11	serr_seen	R	0	SERR observed on PCI bus
10	Reserved	R	0	

Table 8: Registers Description

Bit	Symbol	Access	Value	Description
9	pci_err	R	0	PCI master transaction attempted when not enabled by config register
8	err_base10_subword	R	0	Subword attempt to base10 aperture when restrained to word only (not used on the PNX15xx Series)
7	err_base14_subword	R	0	Subword attempt to base14 aperture when restrained to word only
6	err_base18_subword	R	0	Subword attempt to base18 aperture when restrained to word only (not used on PNX15xx Series)
5	pci_mstr_parity_err	R	0	PCI master set or observed parity error (PERR)
4	err_pci_parity	R	0	PCI Detected parity error (PERR)
3	sig_serr	R	0	Signaled system error (SERR)
2	pci_r_mabort	R	0	PCI Received Master Abort
1	pci_r_tabor	R	0	PCI Received Target Abort
0	pci_s_tabort	R	0	PCI Signaled Target Abort
Offset 0x04 0FE4	***PCI Interrupt Enable***			
31:27	Reserved	R	0	
26	en_int_pcii_wr_err	R/W	0	Enable interrupt on PCI DTL initiator write error flag
25	en_int_pcii_rd_err	R/W	0	Enable interrupt on PCI DTL initiator read error flag
24	en_int_xio_wr_err	R/W	0	Enable interrupt on XIO DTL target write error flag
23	en_int_xio_rd_err	R/W	0	Enable interrupt on XIO DTL target read error flag
22	en_int_pcir_wr_err	R/W	0	Enable interrupt on mmio register DTL target write error flag
21	en_int_pcir_rd_err	R/W	0	Enable interrupt on mmio register DTL target read error
20	en_int_pwrstate_chg	R	0	Enable interrupt on change of power state register
19	Reserved	R	0	
18	en_int_pci2_wr_err	R/W	0	Enable interrupt on PCI2 DTL target write error flag
17	en_int_pci2_rd_err	R/W	0	Enable interrupt on PCI2 DTL target read error flag
16	en_int_pci1_wr_err	R/W	0	Enable interrupt on PCI1 DTL target write error flag
15	en_int_pci1_rd_err	R/W	0	Enable interrupt on PCI1 DTL target read error flag
14	en_int_pci_xio_ack_done	R/W	0	Enable interrupt on rising edge of xio_ack done
13:12	Reserved	R	0	
11	en_int_serr_seen	R/W	0	Enable interrupt on SERR observed on PCI bus
10	Reserved	R	0	
9	en_int_pci_err	R/W	0	Enable interrupt on pci_err flag
8	en_int_base10_subword	R/W	0	Enable interrupt on Subword Attempt to Base10 Error Status
7	en_int_base14_subword	R/W	0	Enable interrupt on Subword Attempt to Base14 Error Status
6	en_int_base18_subword	R/W	0	Enable interrupt on Subword Attempt to Base18 Error Status
5	en_int_pci_mstr_parity err	R/W	0	Enable interrupt on PCI Master Parity Error
4	en_int_pci_parity	R/W	0	Enable interrupt on PCI Parity Error Status

Table 8: Registers Description

Bit	Symbol	Access	Value	Description
3	en_int_sig_serr	R/W	0	Enable interrupt on System Error Status
2	en_int_pci_r_mabort	R/W	0	Enable interrupt on PCI Received Master Abort Status
1	en_int_pci_r_tabort	R/W	0	Enable interrupt on PCI Received Target Abort Status
0	en_int_pci_s_tabort	R/W	0	Enable interrupt on PCI Signaled Target Abort Status
Offset 0x04 0FE8	***PCI Interrupt Clear***			
31:27	Reserved	R	0	
26	clr_pcii_wr_err	W	0	Clear PCI DTL initiator write error flag
25	clr_pcii_rd_err	W	0	Clear PCI DTL initiator read error flag
24	clr_xio_wr_err	W	0	Clear XIO DTL target write error flag
23	clr_xio_rd_err	W	0	Clear XIO DTL target read error flag
22	clr_pcir_wr_err	W	0	Clear mmio register DTL target write error flag
21	clr_pcir_rd_err	W	0	Clear mmio register DTL target read error
20	clr_pwrstate_chg	W	0	Clear power state change register flag
19	Reserved	R	0	
18	clr_pci2_wr_err	W	0	Clear PCI2 DTL target write error flag
17	clr_pci2_rd_err	W	0	Clear PCI2 DTL target read error flag
16	clr_pci1_wr_err	W	0	Clear PCI1 DTL target write error flag
15	clr_pci1_rd_err	W	0	Clear PCI1 DTL target read error flag
14	clr_pci_xio_ack_done	W	0	Clear pci_xio_ack done flag
13:12	Reserved	R	0	
11	clr_serr_seen	W	0	Clear serr_seen flag
10	Reserved	R	0	
9	clr_pci_err	W	0	Clear pci_err flag
8	clr_base10_subword	W	0	Clear Subword Attempt to Base10 Error Status
7	clr_base14_subword	W	0	Clear Subword Attempt to Base14 Error Status
6	clr_base18_subword	W	0	Clear Subword Attempt to Base18 Error Status
5	clr_pci_mstr_parity_err	W	0	Clear PCI Master Parity Error
4	clr_pci_parity	W	0	Clear PCI Parity Error Status
3	clr_sig_serr	W	0	Clear System Error Status
2	clr_pci_r_mabort	W	0	Clear PCI Received Master Abort Status
1	clr_pci_r_tabort	W	0	Clear PCI Received Target Abort Status
0	clr_pci_s_tabort	W	0	Clear PCI Signaled Target Abort Status
Offset 0x04 0FEC	***PCI Interrupt Set***			
31:27	Reserved	R	0	
26	set_pcii_wr_err	W	0	Set PCI DTL initiator write error flag
25	set_pcii_rd_err	W	0	Set PCI DTL initiator read error flag
24	set_xio_wr_err	W	0	Set XIO DTL target write error flag
23	set_xio_rd_err	W	0	Set XIO DTL target read error flag

Table 8: Registers Description

Bit	Symbol	Access	Value	Description
22	set_pcir_wr_err	W	0	Set mmio register DTL target write error flag
21	set_pcir_rd_err	W	0	Set mmio register DTL target read error
20	set_pwrstate_chg	W	0	Set change of power state register flag
19	Reserved	R	0	
18	set_pci2_wr_err	W	0	Set PCI2 DTL target write error flag
17	set_pci2_rd_err	W	0	Set PCI2 DTL target read error flag
16	set_pci1_wr_err	W	0	Set PCI1 DTL target write error flag
15	set_pci1_rd_err	W	0	Set PCI1 DTL target read error flag
14	set_pci_xio_ack_done	W	0	Set pci_xio_ack done flag
13:12	Reserved	R	0	
11	set_serr_seen	W	0	Set serr_seen flag
10	Reserved	R	0	
9	set_pci_err	W	0	Set pci_err flag
8	set_base10_subword	W	0	Set Subword Attempt to Base10 Error Status
7	set_base14_subword	W	0	Set Subword Attempt to Base14 Error Status
6	set_base18_subword	W	0	Set Subword Attempt to Base18 Error Status
5	set_pci_mstr_parity_err	W	0	Set PCI master Parity Error
4	set_pci_parity	W	0	Set PCI Parity Error Status
3	set_sig_serr	W	0	Set System Error Status
2	set_pci_r_mabort	W	0	Set PCI Received Master Abort Status
1	set_pci_r_tabort	W	0	Set PCI Received Target Abort Status
0	set_pci_s_tabort	W	0	Set PCI Signaled Target Abort Status
Offset 0x04 0FFC	***Module ID***			
31:16	Module ID	R	0xA051	Module ID
15:12	Major Revision number	R	0	Major Revision number
11:8	Minor revision number	R	1	Minor revision number
7:0	mod_size	R	0	Module size is 4 kB.

Table 9: PCI Configuration Registers

Bit	Symbol	Access	Value	Description
Offset 0x0000	***Device ID/Vendor ID***			
31:16	Device ID	R	0x5405	The ID assigned by the PCI SIG representative. The value will be hard coded.
15:0	Vendor ID	R	0x1131	Value 0x1131 is the ID assigned to Philips Semiconductors by the PCI SIG representative.
Offset 0x0004	***Command/Status***			
31	Parity Error	R/W	0	This bit will be set whenever the device detects a parity error. Write 1 to clear.

Table 9: PCI Configuration Registers

Bit	Symbol	Access	Value	Description
30	Signaled System Error	R/W	0	This bit is set whenever the device asserts SERR. Write 1 to clear.
29	Received Master Abort	R/W	0	Set by the PCI master when its transaction is terminated with a master abort. Write 1 to clear.
28	Received Target Abort	R/W	0	Set by the PCI master when its transaction is terminated with a target abort. Write 1 to clear.
27	Signaled Target Abort	R/W	0	Set by the PCI target when it terminates a transaction with a target abort. Write 1 to clear.
26:25	Devsel Timing	R	01	The PCI target uses medium DEVSEL timing.
24	Master Data Parity Error	R/W	0	Set by the PCI master when PERR is observed.
23	Fast Back-to-Back Capable	R	1	The PCI supports fast back-to-back transactions.
22	Reserved	R	0	
21	66 MHz Capable	R	cfg*	0 = 33 MHz PCI (The PNX15xx Series is 33 MHz). *Value determined by pci_setup register.
20	Capabilities List	R	1	Indicates a new Capabilities linked list is available at offset 40h.
19:10	Reserved	R	0000	
9	Fast back-to-back enable	R/W	0	Enable fast back-to-back transactions for PCI master.
8	SERR enable	R/W	0	Enable SERR to report system errors.
7	Stepping Control	R	0	Address stepping is not supported.
6	Parity Error Response	R/W	0	0 = No parity error response 1 = Enable parity error response.
5	VGA Palette Snoop	R	0	VGA is not supported.
4	Memory Write & Invalidate	R/W	0	Enable use of memory write and invalidate.
3	Special Cycles	R	0	Special cycles are not supported.
2	Enable Bus Master	R/W	0	Enable the PCI bus master.
1	Enable Memory space	R/W	0	Enable all memory apertures.
0	IO Space	R	0	The PCI module does not respond to IO transactions.
Offset 0x0008	***Class Code/Revision ID***			
31:8	Class Code	R/W*	048000	The PNX15xx Series is defined as a multimedia device. *The boot loader may change the class code to an alternate value if done before writing to the pci_setup register.
7:0	Revision ID	R	1	Revision ID. Will initially be assigned to 0. Revision ID must not be synthesized. It will need to be changed with revised silicon, whether for bug fixes or enhancements.
Offset 0x000C	***Latency Timer/Cache Line Size***			
31:16	Reserved	R	0x0000	Note: BIST is not implemented. Header is 0.
15:8	Latency Timer	R/W	0	Latency Timer
7:0	Cache Line Size	R/W	0	Cache Line Size
Offset 0x0010	***Base10 Address Register***			

Table 9: PCI Configuration Registers

Bit	Symbol	Access	Value	Description
This aperture is for the SDRAM on the PNX15xx Series. The following memory sizes are supported: 128 MB, 64 MB, 32 MB, or 16 MB.				
31:28	Base10 Address	R/W	0	Upper 4 bits of base10 address of the first memory aperture
27:21	Base10 Address	R/W*	0	*The base 10 can be configured to various aperture sizes from 2 MB to 256 MB. (See pci_setup register). Depending on aperture size selected, various bits will be R/W or Read Only. Bit: 27262524232221 256M:RORORORORORORO 128M:RWRORORORORORO 64M:RWRWRORORORORO 32M:RWRWRWRORORORO 16M:RWRWRWRWRORORO 8M:RWRWRWRWRWRORO 4M:RWRWRWRWRWRWRO 2M:RWRWRWRWRWRWRW RO = Read-only bits read back as zero.
20:4	Reserved	R	0	
3	Prefetchable	R	*cfg	Value is determined at boot time by the pci_setup register.
2:0	Type	R	0	Indicates type 0 memory space (locatable anywhere in 32-bit address space).
Offset 0x0014	***Base14 Address Register***			
This aperture will be set to 2 MB for MMIO on the PNX15xx Series.				
31:28	Base14 Address	R/W	0001	Upper 4 bits of base14 address of the first memory or IO aperture
27:21	Base14 Address	R/W*	1011111	*The base 14 can be configured to various aperture sizes from 2 MB to 256 MB. (See pci_setup register). Depending on aperture size selected, various bits will be R/W or Read Only. **Bit: 27262524232221** 256M:RORORORORORORO 128M:RWRORORORORORO 64M:RWRWRORORORORO 32M:RWRWRWRORORORO 16M:RWRWRWRWRORORO 8M:RWRWRWRWRWRORO 4M:RWRWRWRWRWRWRO 2M:RWRWRWRWRWRWRW RO = Read-only bits read back as zero.
20:4	Reserved	R	0	
3	Prefetchable	R	*cfg	Value is determined at boot time by the pci_setup register.
2:0	Type	R	0	Indicates type 0 memory space (locatable anywhere in 32-bit address space).
Offset 0x0018	***Base18 Address Register***			
This aperture is for the XIO on the PNX15xx Series, which supports up to 128 MB of XIO memory space.				
31:28	Base18 Address	R/W	0001	Upper 18 bits of base address of the first memory or IO aperture

Table 9: PCI Configuration Registers

Bit	Symbol	Access	Value	Description
27:21	Base18 Address	R/W*	1100000	*The base 18 can be configured to various aperture sizes from 2 MB to 256 MB. (See pci_setup register). Depending on aperture size selected, various bits will be R/W or Read Only. **Bit: 27262524232221** 256M:RORORORORORORO 128M:RWRORORORORORO 64M:RWRWRORORORORO 32M:RWRWRWRORORORO 16M:RWRWRWRWRORORO 8M:RWRWRWRWRWRORO 4M:RWRWRWRWRWRWRO 2M:RWRWRWRWRWRWRW RO = Read-only bits read back as zero.
20:4	Reserved	R	0	
3	Prefetchable	R	cfg*	Prefetchable if configured as 1. *Value is determined by pci_setup register.
2:0	Memory	R	0	This bit indicates type 0 memory aperture.
Offset 0x002C	***Subsystem ID/Subsystem Vendor ID***			
The values used in this register will be loaded into the register before entertaining any transactions on the PCI bus. The boot loader will initialize control register address 0x006C with the correct values.				
31:16	Subsystem ID	R	0	Subsystem ID. The value for this field is provided by Philips PCI SIG representative for Philips internal customers. External customers will provide their own number.
15:0	Subsystem Vendor ID	R	0	Subsystem Vendor ID. The value for this field is 1131 for Philips internal customers. External customers need to apply to the PCI SIG to obtain a value if they do not have one already.
Offset 0x0030	***Reserved***			
Offset 0x0034	***Capabilities Pointer***			
31:8	Reserved	R	0	
7:0	cap_pointer	R	0x40	Indicates extended capabilities are present starting at 40.
Offset 0x003C	***Max_Lat, Min_Gnt, Interrupt pin, Interrupt Line***			
31:24	max_lat	R/W1	0x18	Indicates the max latency tolerated in 1/4 microsecond for PCI master. This value may be changed if written to before the pci_setup register.
23:16	min_gnt	R/W1	0x09	Indicates how long the PCI master will need to use the bus. This value may be changed if written to before the pci_setup register.
15:8	interrupt_pin	R	0x01	Indicates which interrupt pin is used.
7:0	interrupt_line	R/W	0x00	Interrupt routing information
Offset 0x0040	***Power Management Capabilities***			
31:27	Reserved	R	0x0000	
26	d2_support	R	cfg*	1 = Device supports D2 power management state *Value is determined by pci_setup register.
25	d1_support	R	cfg*	1 = Device supports D1 power management state *Value is determined by pci_setup register.
24:19	Reserved	R	0	

Table 9: PCI Configuration Registers

Bit	Symbol	Access	Value	Description
18:16	version	R	010	Indicates compliance with version 1.1 of PM.
15:8	Next Item Pointer	R	00	There are no other extended capabilities.
7:0	Cap_ID	R	01	Indicates this is power management data structure.
Offset 0x0044	***PMCSR***			
31:2	Reserved	R		
1:0	pwr_state	RW		power_state. These bits are writable only when the corresponding bit in the PMC register is enabled

Chapter 8: General Purpose Input Output Pins

PNX15xx Series Data Book – Volume 1 of 1

Rev. 2 — 1 December 2004 Product data sheet

1. Introduction

The PNX15xx Series has 61 pins that are capable of operating as General Purpose software Input Output (GPIO) pins. 16 of them are dedicated GPIO pins. The other 45 pins are assigned to the other PNX15xx Series modules, like the Audio Out module, but they can be re-used as GPIO pins if they are not being used for their normal functional behavior. So these are designated as optional GPIO pins that can either operate in regular mode or in GPIO mode. All 61 pins support common features:

- software I/O - set a pin or pin group, enable a pin (or a pin group) and inspect pin values
- precise timestamping of internal and external events (up to 12 signals simultaneous)
- signal event sequence monitoring or signal generation (up to 4 signals simultaneous)
- timer source selection for TM3260

The 61 pins have the same GPIO capabilities. However some of the dedicated GPIO pins have additional features like:

- clocks - these pins are possible clock source for pattern generation or sampling mode. Or they are simply used to provide a clock to peripherals on the PNX15xx Series system board.
- wake-up event - used to wake-up PNX15xx Series from deep sleep mode, see Chapter 5 The Clock Module.
- boot option - determines the boot settings of PNX15xx Series, see Chapter 6 Boot Module.
- watchdog - this is a subset of the software I/O mode since the TM3260 CPU would toggle this pin at regular intervals in order to prevent an external watchdog to reset the entire system. Alternately the internal watchdog timer of PNX15xx Series system can be used, see Chapter 4 Reset.

After a PNX15xx Series system reset all the GPIO pins start in GPIO mode and in input mode.

2. Functional Description

A simplified block diagram of the GPIO module can be found in Figure 1. It presents the major interfaces of the GPIO module.

- the GPIO pins
- the MTL interface used to fetch data when operating in pattern generation mode or used to store data when the GPIO module is used in sampling mode. In both cases up to 4 First In First Out (FIFO) memory buffers are available for one of the modes.
- the DCS bus interface used to convey the MMIO register read and writes issued by the TM3260 CPU or any other master connected to PNX15xx Series through the PCI bus interface.
- the 5 interrupt lines which are routed directly to the TM3260 CPU. 4 lines are associated with the signal monitoring while the last interrupt line is linked to the event monitoring.

The following sections describe in more details the GPIO module behavior.

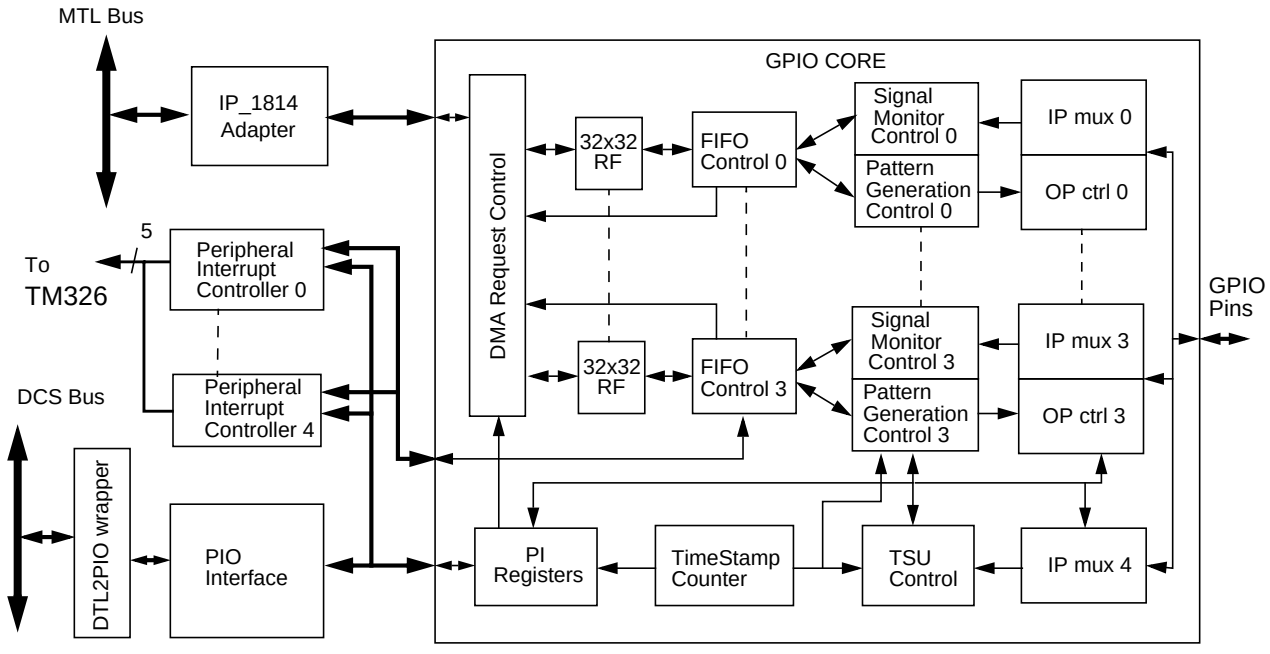


Figure 1: GPIO Module Block Diagram

2.1 GPIO: The Basic Pin Behavior

The pins that can be set as GPIO pins is available in Chapter _ in Section 2.3 on page 1-3 under the column 'GPIO #'. The following Table 1 duplicates the GPIO pin assignment. It also adds the inactive state value of the functional signal when the pin is switched from its functional mode to GPIO mode. The inactive state is used to

avoid unpredictable behavior in a module when the pin is being used as a GPIO pin. Figure 2 illustrates the basic functional diagram of a GPIO pin in the PNX15xx Series system.

Table 1: GPIO Pin List

Primary Function	GPIO Number	PNX15xx Series Module	Inactive State
FGPO_REC_SYNC	60	FGPO	0
VDI_V2	59	Input Video/Data Router	1
VDI_V1	58		1
SPDO	57	SPDIF Output	n/a
SPDI	56	SPDIF Input	1
VDO_AUX	55	Output Video/Data Router	n/a
VDO_D[33:32]	54 - 53		n/a
VDI_D[33:32]	52 - 51	Input Video/Data Router	1
LAN_MDC	50	LAN 10/100 MAC	n/a
LAN_MDIO	49		0
LAN_RX_ER	48		0
LAN_RX_DV	47		0
LAN_RXD[3:0]	46 - 43		0
LAN_COL	42		0
LAN_CRS	41		0
LAN_TX_ER	40		n/a
LAN_TXD[3:0]	39 - 36		n/a
LAN_TX_EN	35		n/a
XIO_D[15:8]	34 - 27	PCI-XIO	1
XIO_ACK	26		1
AO_SD[3:0]	25 - 22	Audio Out	n/a
AO_WS	21		n/a
AI_SD[3:0]	20 - 17	Audio In	0
AI_WS	16		0
GPIO	15 - 0	GPIO	n/a

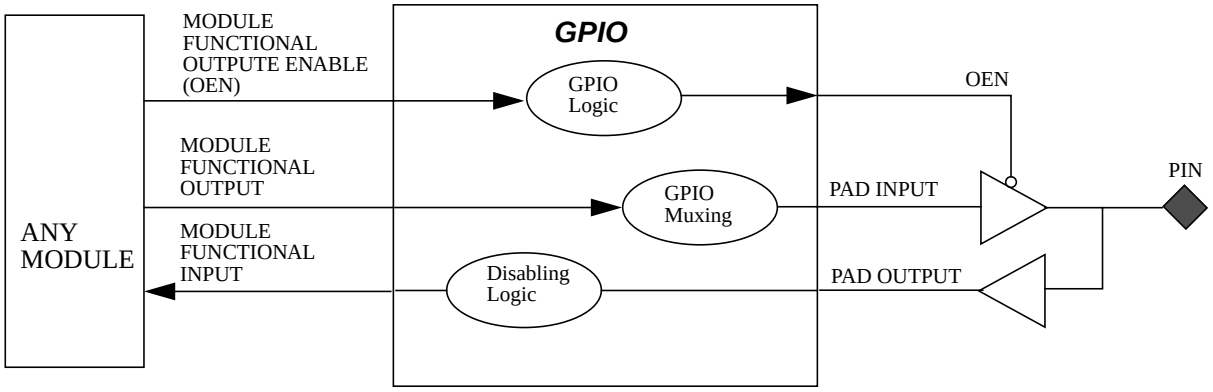


Figure 2: Functional Block Diagram of a GPIO Pin

The GPIO pins are controlled by software through MMIO register reads and writes. The MMIO registers allow to control the operating mode of the GPIO pin (on a pin-by-pin basis) but also set its value or read its value.

2.1.1 GPIO Mode settings

Each GPIO pin operates in 1 of 3 following modes:

- primary function
- open drain output
- tri-state output.

There are four GPIO Mode Control registers allocated to control the operating mode of the 61 PNX15xx Series GPIO pins. Each pin uses a 2-bit mode field located in one of the 4 Mode Control registers. Register MC0 controls GPIO pins [15:0], MC1 controls pins [31:16], etc. The 2-bit control values function is described in Table 2. The complete MMIO register layouts are in Section 4.1.

Table 2: GPIO Mode Select

GPIO Mode	Description
00	Retain pin mode of operation. A write with this mode does not overwrite current mode.
01	Switch pin mode to primary operating mode.
10	Switch pin mode to GPIO mode.
11	Switch pin mode to open-drain GPIO (this prevents active high drive).

2.1.2 GPIO Data Settings MMIO Registers

When a pin is set for GPIO mode, the data can be read and written by accessing one of four MASK and I/O Data (IOD) registers. Each of these registers accesses 16 of the 61 GPIO signals. Each register is composed of 16 MASK bits and 16 IOD bits. The MASK and IOD field make up a 2-bit value: the MASK bit is located in the upper 16 bits (31:16) and the IOD bit is located in the lower 16 bits (15:0) of the corresponding 32-bit MMIO register (groups 16 GPIO pins). For example, MASK

bit[16] is paired with IOD bit[0] and [17]...[1], [18]...[2], etc. This pairing makes up the 2-bit value for programming the GPIO data setting. The pairing allows to control 16 GPIO pins with a single 32-bit MMIO write/read from the TM3260 CPU. The available data settings are documented in Table 3. The complete MMIO register layouts are in Section 4.2.

Table 3: Settings for MASK[xx] and IOD[xx] Bits

MASK[xx] Bit	IOD[xx] Bit	Description
0	0	Retain current stored data (a write of 00 does not overwrite current data). Not Readable.
0	1	Data Input Mode, i.e. set the corresponding GPIO Pin in tri-state mode.
1	0	GPIO Output Mode. Drive a '0' onto the corresponding GPIO Pin or a generated pattern (see Section 2.3)
1	1	GPIO Output Mode. Drive a '1' onto the corresponding GPIO Pin or a generated pattern (see Section 2.3). **Note:** if open-drain mode is selected, drive to '1' is disabled.

Note: The xx portion of MASK[xx] or IOD[xx] identifies the GPIO number of the particular pin. Refer to Table 1 for the number allocation and the GPIO Data Control Register table on page 8-3.

Remark: Software should treat with care these MMIO registers since they do not behave as regular registers and some electrical problem can occur at board level since:

- writing to these bits may switch I/O signals between input & output mode.
- the IOD field of these registers reflects the state of the actual pad of the signal. This implies that depending on the mode of the GPIO pin values written to the IOD bits may not affect the pin state, and therefore cannot be read back.
- writing a 00 (binary) value to a MASK and IOD field pair causes no changes to the 2-bit field.

Writing Data on a GPIO Pin

A specific data can be written to a GPIO pin by executing a single MMIO register write. This is achieved by setting a '1' to the corresponding MASK[xx] bit and set IOD bit to the desired pin value, as described in Table 3.

Remark: The IOD bits may not reflect the value written to them since these bits are used to always represent the actual signal values at the pin side.

Remark: After reset every GPIO pin is in GPIO mode. The GPIO mode settings need to be programmed in order to switch the GPIO into its primary operating mode. It should be noted that if the primary operating mode for a GPIO is an active-low output a glitch can occur on the output if the data reaches the IO logic before the output enable. Therefore the software should always program it to GPIO mode first and then switch it to primary operating mode as follows:

1. Program Mode Select register in GPIO mode, i.e. 10 (binary).
2. Program Mode Select register in primary operating mode, i.e. 01 (binary).

2.1.3 GPIO Pin Status Reading

Each GPIO pin can be read by software using an MMIO read of the proper MASK and IOD register. In the 32-bit register, the lower 16 bits are the GPIO pin data values.

Software reading of the GPIO input pins is always possible, even when the GPIO pin is operating in its primary function mode.

Remark: For open drain or tri-state output values, the input value read by software is the pad value, not the driven value.

2.2 GPIO: The Event Monitoring Mode

The GPIO module allows to monitor events on all 61 GPIO pins but also on some PNX15xx Series internal signals coming from the different modules of PNX15xx Series. These signals are usually signals indicating the end or the start of the capture of a buffer. Documentation on the following signals can be found on each module documentation.

- VIP timestamp: vip1_eow_vbi, vip_eow_vid
- AI timestamp: ai1_tstamp
- AO timestamp: ao1_tstamp
- SDPI: spdi_tstamp1, spdi_tstamp2 (See SPDI MUX in Section 8.1 on page 3-27)
- SPDO: spdo_tstamp
- GPIO timestamps: LAST_WORD[3:0]
- QVCP timestamp: qvcp_tstamp

The state of these internal signals can be observed by software at any time by consulting the Internal Signals MMIO register documented in Section 4.3.

PNX15xx Series integrates a total of 12 timestamp units for event monitoring. An event is defined by a change on the monitored signals, i.e. a high to low or a low to high transition is an event. The operating mode of the timestamp units is simple:

- The software running on TM3260 selects the internal signals or the GPIO pins to be event monitored by setting properly the GPIO_EV[15:4] MMIO registers. These 12 control registers (one per timestamp unit) are used to select the source to monitor, the type of the event (rising, falling edge or both) as well as enabling the capture of the event.
- Every time an event occurs a DATA_VALID interrupt is generated. Therefore the DATA_VALID interrupt condition needs to be enabled by writing to the INT_ENABLE4 MMIO register (the GPIO generates the interrupt through interrupt line 4 which is connected to the TM3260, see Table 5 on page 3-12 for SOURCE number allocation). The INT_STATUS4 MMIO register indicates which of the 12 units has data ready to consume. The GPIO module expects then an interrupt clear by writing to the INT_CLEAR4 MMIO register.

- An overrun error interrupt is generated whenever new data is received before the DATA_VALID interrupt has been cleared. The old data is not overwritten, the new data is lost. The overrun interrupt shares the same interrupt MMIO registers as the VALID_DATA interrupt. The interrupt is enabled with INT_ENABLE4, cleared through INT_CLEAR4 and consulted through INT_STATUS4 MMIO registers.
- Upon a DATA_VALID interrupt the corresponding 32-bit TimeStamp Unit (TSU) MMIO register is stable to be read by software when the relevant DATA_VALID_[11:0] flag in the INT_STATUS4 MMIO register is raised. The TSU register contains the timestamp information, a direction bit and a 31-bit timestamp value, see Section 2.2.2.

Event monitoring is commonly used for low frequency events (less than a 100 per second) while signal monitoring can be used for more frequent events. Therefore the timestamp units are shared with the Signal Monitoring logic, Section 2.3.1.

2.2.1 Timestamp Reference clock

The timestamp reference clock is based on a 34-bit counter running at 108 MHz. However the frequency used for all timestamping in PNX15xx Series is 13.5 MHz (i.e., 108 MHz/8) which gives a better than 75 ns event resolution, i.e. only the upper 32-bit of the counter is visible by software. The counter can be observed with the TIME_CTR MMIO register.

The counter is reset by the PNX15xx Series system reset.

2.2.2 Timestamp format

Any change (according to the monitored edge event) generates a 31-bit timestamp and a 1 bit edge direction in a 32-bit word. The 1-bit direction indicator is a logic ‘1’ if a rising edge has occurred and a logic ‘0’ if a falling edge has occurred. The direction bit is the MSB of the 32-bit word generated. This is pictured in Figure 3.

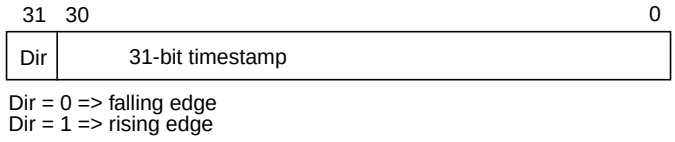


Figure 3: 32-bit Timestamp Format

Remark: The event timestamps can be written (per monitored signal) to a memory buffer, Section 2.3.1, or to a timestamp unit register, which is software readable.

2.3 GPIO: The Signal Monitoring & Pattern Generation Modes

There are 4 FIFO queues available to perform signal monitoring or pattern generation (mutually exclusive). Each FIFO queue can be programmed to operate in either of these modes for a selected group of GPIO pins.

The FIFO has DMA capability to allow efficient CPU access to large event lists (in opposite to the event monitoring described in Section 2.2).

A double DMA buffer scheme is used. The base start addresses for both DMA buffers in every queue is programmable as is the size of the DMA buffers. The SIZE parameter allows DMA buffers to be up to 1 Megabyte. Both DMA buffer work in a ping-pong fashion which forces the use of both of them.

Any of the GPIO pins or the internal signals listed in Section 2.2 can be selected for signal monitoring. Only the GPIO pins can be selected for pattern generation.

The layout of the MMIO register is found in Section 4.4, Section 4.5 and Section 4.11. Selection of the GPIO or internal signals to monitor can be found in Section 4.15.

2.3.1 The Signal Monitoring Mode

The signal monitoring mode is an extension of the event monitoring that uses the 12 timestamp units. The signals, i.e. GPIO pins and the internal signals) can be monitored in two different ways:

- Event Timestamping: Using an event timestamps whenever a signal changes state. In this case the FIFO queues are filled with 32-bit timestamp values as defined in Section 2.2.2.
- Signal Sampling: Sampling the signal value at a programmable frequency. In this mode up to 4 signals per FIFO can be grouped for sampling. The FIFO are filled up with the signal values at each sampling clock edge.

GPIO MMIO Description for Signal Monitoring FIFO queues

The FIFO queues are controlled by the GPIO_EV[3:0] MMIO registers. The status of the sampling and the interrupt control MMIO registers are INT_STATUS[3:0], INT_ENABLE[3:0] and INT_CLEAR[3:0]. INT_SET[3:0] is only meant for software debug (used to trigger the hardware interrupt but using software). In the following text a ‘x’ may be used to refer to one of the 4 MMIO registers, e.g. GPIO_EVx or one of the two flags, like BUFx_RDY for BUF2_RDY or BUF2_RDY.

Upon reset, signal monitoring is disabled (GPIO_EV[3:0].FIFO_MODE and GPIO_EV[3:0].EVENT_MODE = 00), and the DMA buffer 1 is the active DMA buffer. Software initiates signal monitoring by providing, per FIFO, two equal size empty DMA buffers and putting their base address and size in the relevant BASE1_PTRx, BASE2_PTRx and SIZEx MMIO registers. Once two valid DMA buffers are assigned, monitoring can be enabled by programming the relevant GPIO_EVx.FIFO_MODE and GPIO_EVx.EVENT_MODE. For the enabled FIFOs, the GPIO hardware will proceed to fill the DMA buffer 1 with timestamps or samples. Once DMA buffer 1 fills up, INT_STATUSx.BUF1_RDY is asserted, and monitoring continues a seamless transfer in DMA buffer 2. If INT_ENABLEx.BUF1_RDY_EN is enabled, an interrupt request is generated to the chip level interrupt controller, the VIC block in TM3260. The interrupt should be configured in the VIC block in level triggered mode.

When INT_STATUSx.BUF1_RDY is high, software is required to assign a new empty buffer to BASE1_PTRx and then clear the INT_STATUSx].BUF1_RDY flag (by writing a ‘1’ to INT_CLEARx.BUF1_RDY_CLR), before buffer 2 fills up which prevents an overrun.

Monitoring continues in DMA buffer 2, until it fills up. At that time, INT_STATUSx.BUF2_RDY is asserted, monitoring continues in the new DMA buffer 1, and the interrupt needs to be acknowledged as for DMA buffer 1.

If the software fails to read the full DMA buffers in time (i.e BUF1_RDY or BUF1_RDY is not cleared in time), the overrun error flag, INT_STATUSx.FIFO_OE, is raised and data may be lost. The INT_STATUSx.FIFO_OE error flag can only be cleared by an explicit write of '1' to the INT_CLEARx.FIFO_OE_CLR bit. The interrupt if seen by the TM3260 CPU if the bit INT_ENABLEx.FIFO_OE_EN is set.

If enabled, an interval of silence, GPIO_EVx.INTERVAL, can cause a BUFx_RDY flag to be asserted before all locations in the DMA buffer have been filled. Therefore, whenever BUFx_RDY is asserted, software is required to read the relevant INT_STATUSx register to know exactly how many valid 32-bit words of data are in the DMA buffer. The INT_STATUSx holds the VALID_PTR field which gives this information.

The number of valid 32-bit data words written to the DMA buffers is loaded by the GPIO module to the VALID_PTR field of the INT_STATUSx register immediately before the GPIO sets the relevant BUFx_RDY flag. If a second BUFx_RDY is activated before the first flag was cleared, VALID_PTR cannot be updated by the GPIO until the first activated BUFx_RDY flag is cleared by software. This clear will allow the GPIO to load the new VALID_PTR value for the second buffer.

If both BUFx_RDY flags are cleared at the same time, i.e if the value of VALID_PTR is not needed, the VALID_PTR value points back to the first buffer whose BUFx_RDY flag was raised. If the VALID_PTR value is required to be read, each BUFx_RDY must be cleared individually and in the correct order.

VALID_PTR is stable to be read by software when a BUFx_RDY flag is raised. BASE1_PTRx should be stable to be loaded by the GPIO module when BUF1_RDY is cleared by software and BASE2_PTRx should be stable to be loaded by the GPIO module when BUF2_RDY is cleared by software.

Remark: A DMA buffer can 'fill up' in two ways: all available locations are written to, or, in monitoring timestamped event mode, an interval of silence occurred.

Remark: SIZE must be a multiple of 64 bytes. SIZE is a static configuration register and should not change during GPIO operation.

The Interval of Silence in Event Timestamping Sampling Mode

If events occur on a monitored signal and an interval of silence follows, the relevant internal buffer contents are flushed to the DMA buffers.

When the contents of the internal buffer are flushed to the DMA buffer the relevant BUFx_RDY flag is set. The BUFx_RDY interrupt indicates that the DMA buffer is ready to be read by software and writing is switched to the second DMA buffer.

When an interval of silence occurs all the 64 bytes of the internal buffer are flushed even though there may not be 64 bytes of valid data in the internal buffer. Software must then read the module status to read the address where the last valid 32-bit data word, INT_STATUSx.VALID_PTR, was written.

The length of the interval duration is programmed using the GPIO_EV[3:0].INTERVAL fields.

Remark: If there is no internal buffer data to be flushed and no valid data in the DMA buffers the interval of silence will not cause BUFx_RDY to be asserted.

Remark: timestamping always works, even if the pin selected for monitoring is operating in its functional mode.

More About the Sampling Mode

In 'signal sampling' a signal, Figure 4, or a group of signals can be monitored at a programmed frequency or by a selected clock input.

The programmed sampling frequency is divided down from 108 MHz using a 16-bit divider. The sampling frequency is programmed in the DIVIDER[3:0].FREQ_DIV fields. The generated clock has a 50% duty cycle if the divider is an even number. In the case of an odd value the duty cycle is 33-66 or 66-33.

Instead of using the internal 108 MHz sampling clock it is possible to use one of the GPIO[6:0] inputs as the sampling clock. This is enabled using the bit fields EN_CLOCK_SEL and CLOCK_SEL in the GPIO_EV[3:0] registers. Some of the GPIO[6:0] pins can receive a clock coming from a PNX15xx Series DDS clock generators, see Section 2.5. If this feature is used it is important to know that these clocks need to be turned on by programming the clock module, refer to Chapter 5 The Clock Module. Alternately the clocks can be generated at board level.

Signal sampling, should be done with a clock that is at least twice the signal frequency.

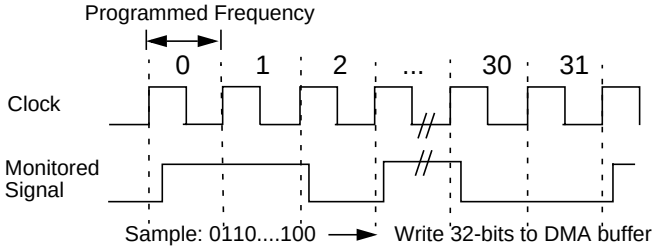


Figure 4: 1-bit Signal Sampling

The input signals to sample can be grouped together and sampled at once in the same FIFO queue. It is possible to sample 1, 2 or 4 GPIO inputs in one FIFO queue. The sampled 1, 2 or 4 bits fill a 32-bit word full of 32, 16 or 8 samples as pictured in Figure 5. The resulting 32-bit word of the sampled signals is written to the DMA

buffer. The numbers of signals to sample together per FIFO queue is programmed by setting the GPIO_EV[3:0].EN_IO_SEL fields. The signal selection for sampling is programmed in the IO_SEL[3:0] registers.

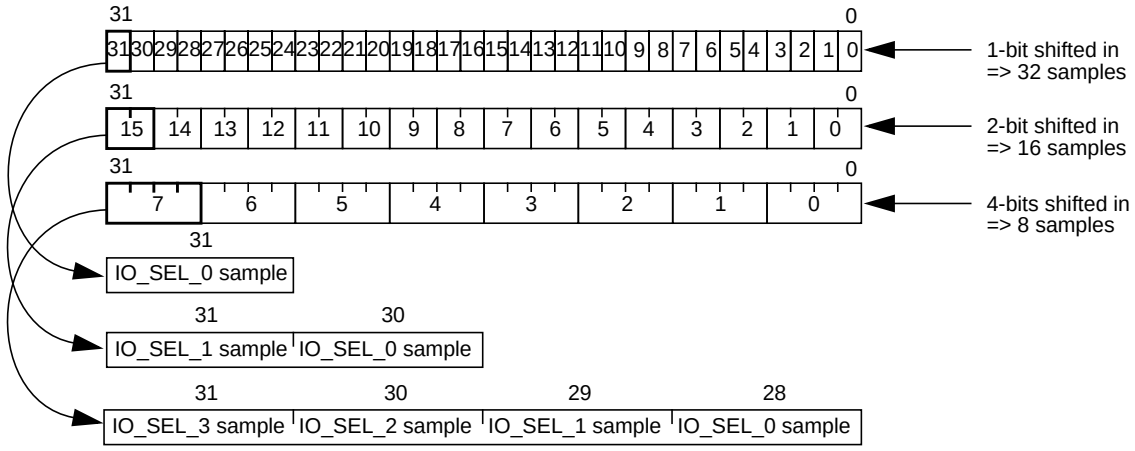


Figure 5: Up to 4-bit Signal Sampling

2.3.2 The Signal Pattern Generation Mode

The signal pattern generation mode is the dual of the signal sampling mode. The software builds in memory DMA buffers that are fetched by the GPIO module. The data is then transferred to a selected group of GPIO pins. Similarly to the sampling mode the pattern generation mode offers two different ways to output signals:

- Timestamp mode: The software creates DMA buffers that contain 32-bit values as defined in Section 2.2.2. The direction bit and the timestamp information is used to drive the GPIO pins with the correct polarity and to emit the sample at the correct time, i.e. when the software computed timestamped matches the internal timestamp counter.
- Pattern mode: The GPIO module outputs the DMA buffer content on a select group of GPIO pins. In this mode up to 4 signals per FIFO can be grouped for pattern generation.

Pattern generation can start once the software has filled the DMA buffers.

GPIO MMIO Description for Pattern Generation FIFO queues

The FIFO queues are controlled by the GPIO_EV[3:0] MMIO registers. The status of the sampling and the interrupt control MMIO registers are INT_STATUS[3:0], INT_ENABLE[3:0] and INT_CLEAR[3:0]. INT_SET[3:0] is only meant for software debug (used to trigger the hardware interrupt but using software). In the following text a ‘x’ may be used to refer to one of the 4 MMIO registers, e.g. GPIO_EVx or one of the two flags, like BUFx_RDY for BUF2_RDY or BUF2_RDY.

Upon reset, transmission is disabled (GPIO_EVx.FIFO_MODE and GPIO_EVx.EVENT_MODE is reset to 00), and the DMA buffer 1 is the active buffer. The system software initiates transmission by providing two DMA buffers containing

valid data and by putting their base addresses in the two BASEx_PTR registers, their maximum size into the SIZE register and the number of valid words for DMA buffer 1 into the PG_BUF_CTRLx.BUF_LEN bits.

When the FIFO queue is programmed into Pattern Generation mode, i.e. FIFO_MODE[1]=1, BUF1_RDY and BUF2_RDY flags will get set, indicating that it is ready for a new DMA buffer containing valid data to be assigned.

Once two valid buffers are assigned and FIFO queue has been enabled the BUF1_RDY flag must be cleared by software so that the GPIO module can load the BASE1_PTR and the PG_BUF_CTRLx.BUF_LEN values. After BUF1_RDY has been cleared the software can program the BUF_LEN value for DMA buffer 2. When the BUF2_RDY flag is cleared the BASE2_PTR and BUF_LEN values for DMA buffer 2 are loaded by the GPIO modules.

Remark: If the BUF_LEN values for DMA buffer1 and DMA buffer 2 are identical both BUF1_RDY and BUF2_RDY can be cleared at the same time.

The GPIO hardware now proceeds to empty DMA buffer 1 and transmitting the samples/timestamps on the selected GPIO pins. Once DMA buffer 1 is empty, BUF1_RDY is asserted. If BUF2_RDY has been cleared, transmission continues without interruption from DMA buffer 2. If BUF1_RDY_EN is enabled, a level triggered system level interrupt request is generated.

While BUF1_RDY is high, the system software is required to assign a new buffer to BASE1_PTRx, the number of valid words in the new buffer by setting PG_BUF_CTRLx.BUF_LEN and then clear BUF1_RDY (write a ‘1’ to BUF1_RDY_CLR) before DMA buffer 2 fills up to avoid an Underrun condition.Transmission continues from buffer 2, until it is empty. At that time, BUF2_RDY is asserted, and transmission continues from the new buffer 1, and so on. If an Underrun condition is reached the GPIO module stops the transmission, holds current values on the pins and does not warn the CPU that an underrun condition occurred.

Remark: The BASEx_PTRx and PG_BUF_CTRLx.BUF_LEN values for a DMA buffer are only loaded into the GPIO pattern generation logic when the relevant BUFx_RDY signal has been cleared. Since the PG_BUF_CTRLx.BUF_LEN register is shared between both DMA buffers it important that the value in BUF_LEN when BUFx_RDY is being cleared is the correct value for that DMA buffer.

The BASEx_PTRx and BUF_LEN values should be stable before software clears BUFx_RDY.

Remark: The DMA buffer sizes must be a multiple of 64 bytes. SIZE is a static configuration register and must not be changed during GPIO operation.

Pattern Generation using timestamps

This form of pattern generation is the inverse of event timestamping. Software fills a (per signal) DMA buffer with timed events (31-bit timestamp + 1-bit direction). The hardware performs the scheduled event on a selected GPIO pin when the reference timestamp clock reaches this value.

Pattern Generation using signal samples

In this type of pattern generation software fills a DMA buffer with sampled values. Patterns can be generated, Figure 6, at a programmed frequency or by a selected clock input.

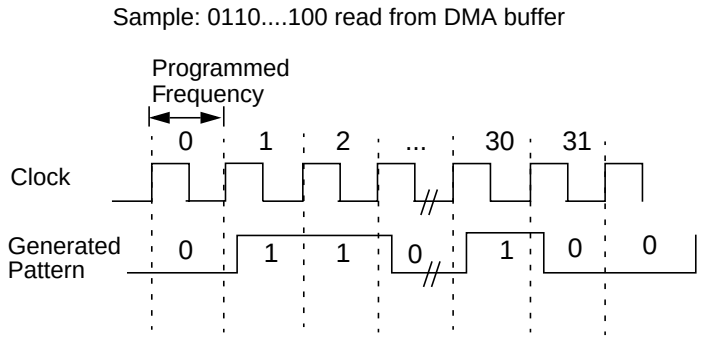


Figure 6: 1-bit Pattern Generation

The programmed sampling frequency is divided down from an internal 108 MHz clock using a 16-bit divider. The divider is programmed in the DIVIDER[3:0].FREQ_DIV fields. The generated clock has a 50% duty cycle if the divider is an even number. In the case of an odd value the duty cycle is 33-66 or 66-33.

Instead of using the internal 108 MHz clock it is also possible to use one of the GPIO[6:0] input pins as the pattern generation clock. This is enabled using the bit fields EN_CLOCK_SEL and CLOCK_SEL in the GPIO_EV[3:0] registers. Some of these GPIO[6:0] can receive a clock coming from a PNX15xx Series DDS clock generators, see Section 2.5. If this feature is used it is important to know that these clocks need to be turned on by programming the clock module, refer to Chapter 5 The Clock Module. Alternately the clocks can be generated at board level. The signal pattern is then generated at the given frequency present on the selected GPIO clock.

GPIO outputs can be grouped together in one FIFO queue. One FIFO queue can drive 1, 2 or 4 outputs. The number of outputs that can be driven by each queue is selected by programming GPIO_EV[3:0].EN_IO_SEL. The driven GPIO output pins are selected by programming the IO_SEL[3:0] registers. The 32-bit sample read from the DMA buffer is either 32 1-bit samples if 1 output is being driven by the FIFO

queue, 16 2-bit samples if 2 outputs are being driven by the FIFO queue or 8 4-bit samples if 4 outputs are being driven by the FIFO queue. This is illustrated in Figure 7.

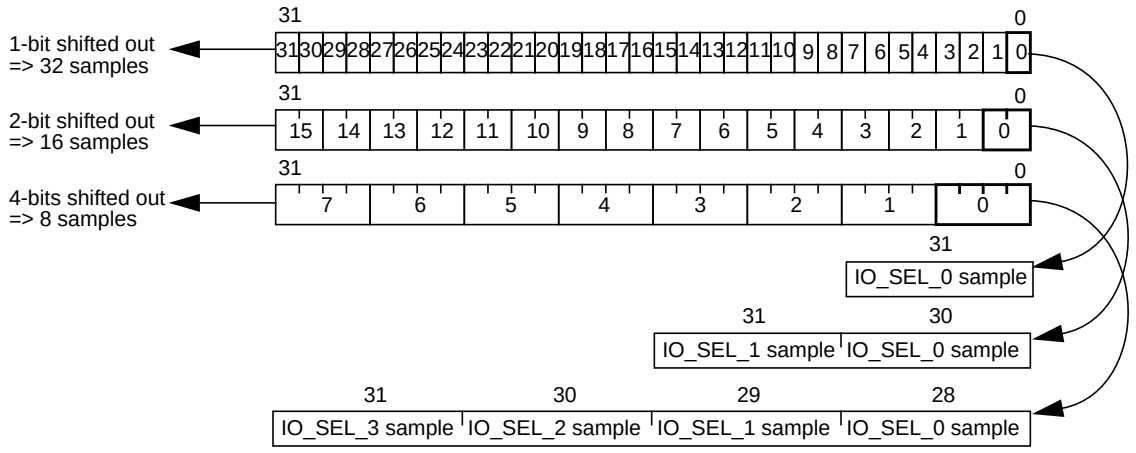


Figure 7: Up to 4-bit Samples per FIFO in Pattern Generation Mode

Remark: The number of samples to be generated is specified in a multiple of 32-bit word being sent by the GPIO module. Therefore, there is no fine-grain way to specify the exact amount of samples to be sent.

Additional GPIO Pattern Generation Feature: Timestamp Signal Generation

In pattern generation modes the GPIO can be programmed to generate events which signal that the last 32-bit word read from a DMA buffer has arrived at a GPIO output pin.

The event will be a positive edge pulse with the duration of the event to be greater than or equal to 148 ns (2 x [1/13.5 MHz]). The specific event generated for each FIFO queue is LAST_WORD.

LAST_WORD[3:0] have the same properties as the other internal signals as described in Section 2.2 and listed in Section 4.15.

The generation of the LAST_WORD[3:0] internal signals is enabled by setting the EN_EV_TSTAMP field of the relevant GPIO_EV[3:0] register.

2.4 GPIO Error Behaviour

A DMA buffer overrun, FIFO_OE, occurs if a new DMA buffer is not supplied by software in time, i.e if BUF1_RDY and BUF2_RDY are both active.

Similarly to the software double DMA buffering scheme, the GPIO module also implements a double internal buffering scheme per FIFO. This double buffering scheme allows to hide the latency of the accesses to the system memory. Each internal buffer is composed of an internal 64-byte memory. In sampling mode, the GPIO uses one of the internal 64-byte memories to store the being sampled data while the second 64-byte memory is being stored into memory. In pattern generation mode, the 64-byte memories are used in the opposite direction. In both cases the

system memory must have consumed the 64-byte memory before the GPIO logic needs it again. If the system memory latency is too long then the GPIO logic does not have an internal 64-byte memory to store in-coming data. In that case GPIO module generates an internal overrun, INT_OE, interrupt. If pattern generation mode the Underrun condition is not flagged to the CPU.

If either a FIFO_OE or INT_OE or Underrun error occurs, signal monitoring is temporarily halted, and incoming timestamps/samples will be lost. In the case of FIFO_OE, sampling resumes as soon as the control software makes one or more new buffers available by clearing the relevant BUFx_RDY. In the case of INT_OE, the GPIO module resumes normal operation as soon as the system memory allows it. INT_OE and FIFO_OE are 'sticky' error flags meaning they will remain set until an explicit software write of logic '1' to FIFO_OE_CLR or INT_OE_CLR is performed. In the case of Underrun the GPIO module resumes as soon as data is available.

2.4.1 GPIO Frequency Restrictions

The GPIO module has two frequency limitations:

- A hardware limitation: the maximum clock used to sample signals or generate patterns is 108 MHz.
- A hardware/software limitation: the system memory latency prevents to fill or empty the internal 64-byte memories on time. This is not only a hardware limitation. Indeed the memory latency is dependant on the memory clock speed, the amount of bandwidth used by the other modules of the PNX15xx Series system and ultimately by the central internal arbiter settings.

One FIFO Enabled

The calculations below show the maximum frequencies allowed for signals to be monitored and patterns to be generated if only one FIFO queue is enabled and the minimum latency guarantied by the system is 40 µs.

Remark: Sampling calculations assume 1-bit sampling (EN_IO_SEL = 00 or 11).

Timestamping: 1 edge -> 32-bits

=> 16 edges = 64 bytes of data

=> 16 edges can occur every 40 µs

=> 1 edge can occur every 2.5 µs = 400 kHz maximum frequency.

Sampling: 1 edge -> 1bit

=> 512 edges = 64 bytes of data

=> 512 edges can occur every 40 µs

=> 1 edge can occur every 78.125 ns = 12.8 MHz maximum frequency.

Several FIFOs Enabled

There is one DMA read channel and one DMA write channel available for the 4 FIFO queues. Each FIFO queue only makes 64 byte DMA requests to one of the channels. The bandwidth allocated by the central arbiter is done separately for the read channel and for the write channel. The 4 FIFOs compete to access to the same DMA channel. The arbitration between the 4 FIFOs is a priority encoded scheme. Every time there is a slot available in the DMA channel the local arbiter looks for the request coming from the 4 FIFOs in the order 0, 1, 2, and 3. There is up to 3 slots available in the DMA channel. Each FIFO does ping-pong requests, i.e. a FIFO cannot have two pending requests.

If the total system bandwidth available for the 4 FIFO queues in DMA read or DMA write is 64 bytes per 40 µs and if all FIFOs are in read mode or write mode then each FIFO gets one 64-byte request per 4 times 40 µs. If 2 FIFOs are in read mode and the other two in write mode and, at system level, the read DMA channel can get one 64-byte request per 40 µs and the write DMA channel can also get one 64-byte request per 40 µs, then each FIFO can get one 64-byte request per 2x40 µs.

So, in this situation the monitored/generated signal frequencies that can be tolerated are:

Remark: The following sampling calculations assume 1-bit sampling (EN_IO_SEL = 00 or 11).

Timestamping: 1 edge -> 32 bits

=> 16 edges = 64 bytes of data

=> 16 edges can occur every 2x40 µs

=> 1 edge can occur every 5 µs = 200 kHz maximum frequency.

Sampling: 1 edge -> 1 bit

=> 512 edges = 64 bytes of data

=> 512 edges can occur every 2x40 µs

=> 1 edge can occur every 156.25 ns = 6.4 MHz maximum frequency.

Similar calculations for frequency tolerances can be made for 2 or 3 queues requesting DMA in the same direction and at the same time and for queues which use multi-bit sampling, i.e. EN_IO_SEL set to binary code 01 or 10.

Remark: The computation can be made to answer a different question: if the signal to sample is running at 12 MHz, then a sampling frequency of more than 24 MHz is required then what is the minimum latency requirement for my system memory? Similarly, if several FIFOs are operating simultaneously with different operating frequencies (to sample different types of signals) then the different FIFOs will get different maximum operating frequencies because of the local arbitration.

2.5 The GPIO Clock Pins

GPIO[14:12,6:4] pins can be assigned to drive a clock generated from the clock module. These are clocks generated by DDS clock generators. Table 4 shows the mapping between DDS clocks and the GPIO pins through which they are routed to.

The clocks on pins 4, 5 and 6 can be used as clock sources for the FIFO queues. In this case the clocks are first routed to the pins, GPIO[4], GPIO[5] and GPIO[6], and then brought back inside the chip as any other external clock source would be. To use this feature the GPIO_EV register should be programmed in the following way:

GPIO_EV.EN_CLOCK_SEL = enabled, i.e. set to binary code 01 or 11

GPIO_EV.EN_DDS_SOURCE = enabled, i.e. set to ‘1’.

GPIO_EV.CLOCK_SEL = select between pins 4, 5 or 6

The clocks are selectable individually.

The clocks on pins 12, 13 and 14 are only routed to the PNX15xx Series pins and can be used as clock sources for some external devices, or loop back on the system board to GPIO[3:0]. They are not directly used as internal clock sources for the FIFO queues. In order to route the clocks on these GPIO[14:12] pins, the DDS_OUT_SEL MMIO register should be programmed appropriately.

Table 4: GPIO clock sources

GPIO[x] pin	Possible Clock Source
14	DDS0 or DDS2 (The selection is made in the clock module)
13	DDS5 or DDS1 (The selection is made in the clock module)
12	DDS6
6	DDS6
5	DDS7
4	DDS8

2.6 GPIO Interrupts

Each operating FIFO queue can generate 4 types of interrupts:

- BUF1_READY: DMA buffer 1 ready for reading or writing
- BUF2_READY: DMA buffer 2 ready for reading or writing
- FIFO_OE: DMA buffer overrun error
- INT_OE: Internal buffering overrun error.

Each timestamp unit has 2 types of interrupts:

- DATA_VALID: TSU has data ready to be read
- INT_OE: Internal buffering overrun error

Each FIFO queue has its own interrupt line to the TM3260 CPU, see Table 5 on page 3-12 for SOURCE number allocation.

The 12 timestamp unit interrupts are ORed together (if enabled) to produce one interrupt. Therefore the 12 TSUs produce only one interrupt, see Table 5 on page 3-12 for SOURCE number allocation.

All the interrupt status bits are ‘sticky’ bits and can only be cleared by writing a ‘1’ to the relevant interrupt clear register. The GPIO status MMIO register, VIC_INT_STATUS Section 4.9, stores information about whether a FIFO queue or a TSU caused the interrupt.

2.7 Timer Sources

Any of the GPIO pins or internal signals can be selected as a timer source for TM3260, see Table 6 on page 3-14. The selection is done by programming the TIMER_IO_SEL MMIO register, see Section 4.8.

2.8 Wake-up Interrupt

An interrupt called ‘gpio_interrupt’ is generated whenever the GPIO module requests an interrupt. This event is a ‘wake-up’ interrupt for the clock module to turn back on the system clocks once the PNX15xx Series has been sent into deep sleep mode.

2.9 External Watchdog

Any of the GPIO pin can be used in case of an external watchdog style reset generator as the output which is pulsed regularly by software to keep a reset from occurring. WDOG_OUT pin is a regular GPIO pin without any special properties, and can be used as an extra GPIO if no watchdog reset is present.

3. IR Applications

Table 5: Example of IR Characteristics

PROTOCOL	MIN. PULSE	REQ. FREQ	FREQ_DIV[15:0]	CARRIER_DIV[4:0]	Error (%)
CIR (IrDA Control)	6.67 µs	150 kHz	0x2D0	0x1 or Disabled	0
CIR with Sub-Carrier (TX)	0.667 µs	1.5 MHz	0x24	0x14	0
RC-MM	27.77 µs	36 kHz	0xBB8	0x1 or Disabled	0
RC-MM Sub-Carrier	9.26 µs[a]	108 kHz	0x1F4	0x6	0

[a] RF sub-carrier is 36kHz, ONTIME should be between 25-50% of 27.77us period. (108KHz = 33%)

For each FIFO queue programmed in signal monitoring or pattern generation modes, it is possible to divide the 108 MHz clock to obtain suitable frequencies for Ir applications.

As well as the 16-bit divider to divide the 108 MHz clock, each FIFO queue has a further 5-bit divider which can be enabled if sub-carrier frequencies are required for transmission. Therefore, in Ir applications, a FIFO queue can produce Ir signals at a

required TX frequency and have the option to multiplex a sub-carrier frequency onto the TX frequency if required. Figure 8 Figure 9 and Figure 10 illustrate the signal requirements.

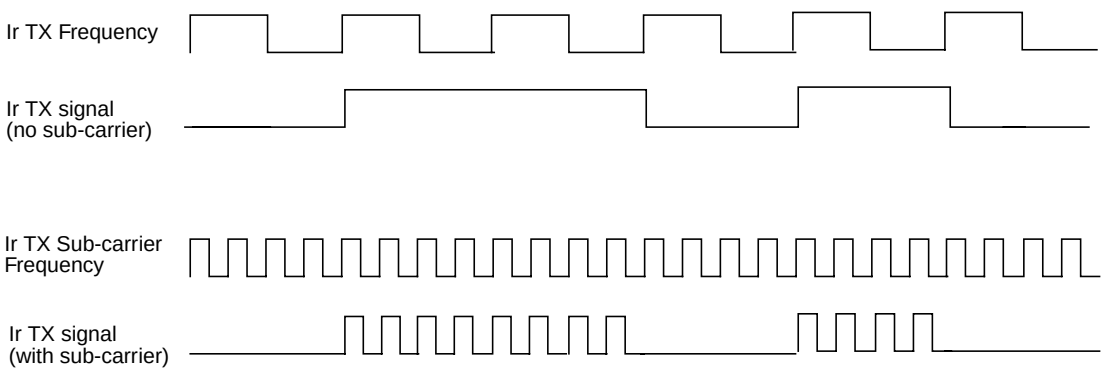


Figure 8: Example of Ir TX Signals with and without Sub-Carrier

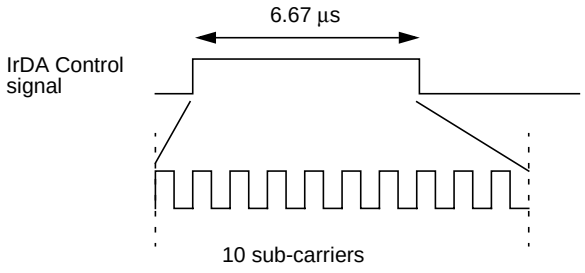


Figure 9: IrDA Control TX with Sub-Carrier Enabled

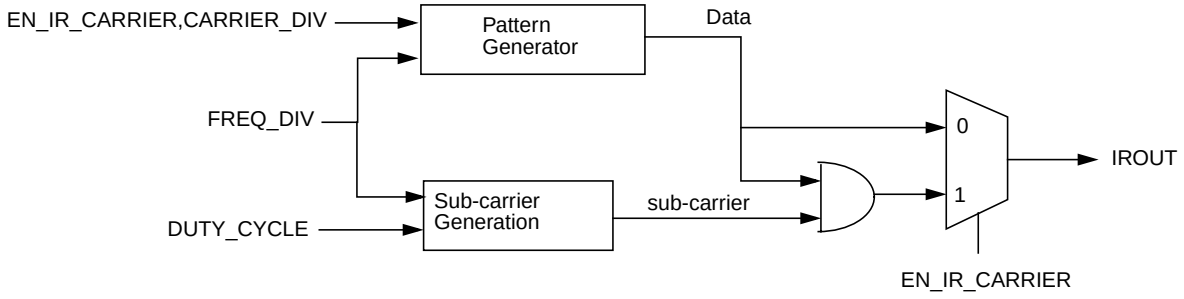


Figure 10: Sub-Carrier Multiplexing for TX

3.1 Duty-cycle programming

In the RC-MM IR protocol the duty-cycle of the sub-carriers must be between 25-50%. To accommodate this protocol and others it is possible to program the duty-cycle to be either 33%, 50%, or 66%.

In the 33% and 66% duty-cycle cases FREQ_DIV must be programmed such that the resulting frequency is 3 times the actual sub-carrier frequency, i.e the minimum pulse width (high or low) is programmed. Similarly, in the 50% duty-cycle case FREQ_DIV must be programmed such that the resulting frequency is 2 times the actual sub-carrier frequency.

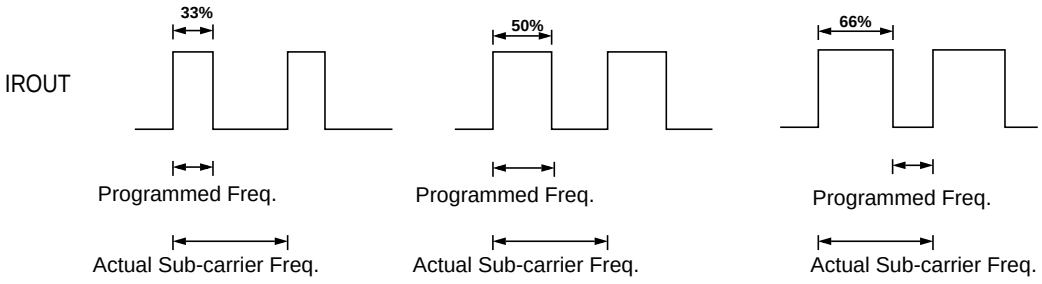


Figure 11: Examples of Duty Cycles for Ir TX Signals

3.2 Spike Filtering

When signal sampling at a programmed frequency a filtering feature is available in the GPIO module which filters out spikes which may occur on a Ir RX signal. This feature is enabled by programming the EN_IR_FILTER and IR_FILTER registers. The IR_FILTER value represents the spike filter pulse width, i.e all pulses less than the IR_FILTER pulse width are considered spikes and not passed through to the signal monitoring control.

4. MMIO Registers

Table 6: Register Summary

	Name	Description
0x10,4000	Mode Control 0	The Mode Control bit pairs which control GPIO pins 15-0.
0x10,4004	Mode Control 1	The Mode Control bit pairs which control GPIO pins 31-16.
0x10,4008	Mode Control 2	The Mode Control bit pairs which control GPIO pins 47-32.
0x10,400C	Mode Control 3	The Mode Control bit pairs which control GPIO pins 60-48.
0x10,4010	MASK and IO Data 0	MASK and IO data for GPIO pins 15-0.
0x10,4014	MASK and IO Data 1	MASK and IO data for GPIO pins 31-16.
0x10,4018	MASK and IO Data 2	MASK and IO data for GPIO pins 47-32.
0x10,401C	MASK and IO Data 3	MASK and IO data for GPIO pins 60-48.
0x10,4020	Internal Signals	Internal signals to be timestamped, software readable.
0x10,4024	GPIO_EV0	GPIO signal monitoring OR pattern generation control register for FIFO queue 0.
0x10,4028	GPIO_EV1	GPIO signal monitoring OR pattern generation control register for FIFO queue 1.
0x10,402C	GPIO_EV2	GPIO signal monitoring OR pattern generation control register for FIFO queue 2.
0x10,4030	GPIO_EV3	GPIO signal monitoring OR pattern generation control register for FIFO queue 3.
0x10,4034	GPIO_EV4	GPIO signal monitoring control register for timestamp unit 0
0x10,4038	GPIO_EV5	GPIO signal monitoring control register for timestamp unit 1
0x10,403C	GPIO_EV6	GPIO signal monitoring control register for timestamp unit 2
0x10,4040	GPIO_EV7	GPIO signal monitoring control register for timestamp unit 3
0x10,4044	GPIO_EV8	GPIO signal monitoring control register for timestamp unit 4
0x10,4048	GPIO_EV9	GPIO signal monitoring control register for timestamp unit 5
0x10,404C	GPIO_EV10	GPIO signal monitoring control register for timestamp unit 6
0x10,4050	GPIO_EV11	GPIO signal monitoring control register for timestamp unit 7
0x10,4054	GPIO_EV12	GPIO signal monitoring control register for timestamp unit 8
0x10,4058	GPIO_EV13	GPIO signal monitoring control register for timestamp unit 9
0x10,405C	GPIO_EV14	GPIO signal monitoring control register for timestamp unit 10
0x10,4060	GPIO_EV15	GPIO signal monitoring control register for timestamp unit 11
0x10,4064	IO_SEL0	IO Select register for FIFO queue 0
0x10,4068	IO_SEL1	IO Select register for FIFO queue 1
0x10,406C	IO_SEL2	IO Select register for FIFO queue 2
0x10,4070	IO_SEL3	IO Select register for FIFO queue 3
0x10,4074	PG_BUF_CTRL0	Pattern Generation DMA buffer control register. for FIFO queue 0
0x10,4078	PG_BUF_CTRL1	Pattern Generation DMA buffer control register. for FIFO queue 1
0x10,407C	PG_BUF_CTRL2	Pattern Generation DMA buffer control register for FIFO queue 2.
0x10,4080	PG_BUF_CTRL3	Pattern Generation DMA buffer control register for FIFO queue 3.
0x10,4084	BASE1_PTR0	Base address for DMA buffer 1 of FIFO queue 0.
0x10,4088	BASE1_PTR1	Base address for DMA buffer 1 of FIFO queue 1.
0x10,408C	BASE1_PTR2	Base address for DMA buffer 1 of FIFO queue 2.

Table 6: Register Summary *…Continued*

	Name	Description
0x10,4090	BASE1_PTR3	Base address for DMA buffer 1 of FIFO queue 3.
0x10,4094	BASE2_PTR0	Base address for DMA buffer 2 of FIFO queue 0.
0x10,4098	BASE2_PTR1	Base address for DMA buffer 2 of FIFO queue 1.
0x10,409C	BASE2_PTR2	Base address for DMA buffer 2 of FIFO queue 2.
0x10,40A0	BASE2_PTR3	Base address for DMA buffer 2 of FIFO queue 3.
0x10,40A4	SIZE0	Size of queue 0 in bytes.
0x10,40A8	SIZE1	Size of queue 1 in bytes.
0x10,40AC	SIZE2	Size of queue 2 in bytes.
0x10,40B0	SIZE3	Size of queue 3 in bytes.
0x10,40B4	DIVIDER_0	Frequency divider for FIFO queue 0
0x10,40B8	DIVIDER_1	Frequency divider for FIFO queue 1
0x10,40BC	DIVIDER_2	Frequency divider for FIFO queue 2
0x10,40C0	DIVIDER_3	Frequency divider for FIFO queue 3
0x10,40C4	TSU0	Timestamp Unit 0.
0x10,40C8	TSU1	Timestamp Unit 1
0x10,40CC	TSU2	Timestamp Unit 2
0x10,40D0	TSU3	Timestamp Unit 3
0x10,40D4	TSU4	Timestamp Unit 4
0x10,40D8	TSU5	Timestamp Unit 5
0x10,40DC	TSU6	Timestamp Unit 6
0x10,40E0	TSU7	Timestamp Unit 7
0x10,40E4	TSU8	Timestamp Unit 8
0x10,40E8	TSU9	Timestamp Unit 9
0x10,40EC	TSU10	Timestamp Unit 10.
0x10,40F0	TSU11	Timestamp Unit 11
0x10,40F4	TIME_CTR	31-bit timestamp master time counter. Runs at 13.5 MHz (108 MHz/8).
0x10,40F8	TIMER_IO_SEL	Selects GPIO pins or internal signals to be use as inputs for internal TM3260 timers.
0x10,40FC	VIC_INT_STATUS	Combined Interrupt status register for the VIC interrupts
0x10,4100	DDS_OUT_SEL	Enables GPIO[14:12] pins to output clocks coming from the clock module.
0x10,4FA0	INT_STATUS0	Interrupt status register, combined with module status for FIFO queue 0
0x10,4FA4	INT_ENABLE0	Interrupt enable register for FIFO queue 0
0x10,4FA8	INT_CLEAR0	Interrupt clear register (by software) for FIFO queue 0
0x10,4FAC	INT_SET0	Interrupt set register (by software) for FIFO queue 0
0x10,4FB0	INT_STATUS1	Interrupt status register, combined with module status for FIFO queue 1
0x10,4FB4	INT_ENABLE1	Interrupt enable register for FIFO queue 1
0x10,4FB8	INT_CLEAR1	Interrupt clear register (by software) for FIFO queue 1
0x10,4FBC	INT_SET1	Interrupt set register (by software) for FIFO queue 1
0x10,4FC0	INT_STATUS2	Interrupt status register, combined with module status for FIFO queue 2

Table 6: Register Summary *…Continued*

	Name	Description
0x10,4FC4	INT_ENABLE2	Interrupt enable register for FIFO queue 2
0x10,4FC8	INT_CLEAR2	Interrupt clear register (by software) for FIFO queue 2
0x10,4FCC	INT_SET2	Interrupt set register (by software) for FIFO queue 2
0x10,4FD0	INT_STATUS3	Interrupt status register, combined with module status for FIFO queue 3
0x10,4FD4	INT_ENABLE3	Interrupt enable register for FIFO queue 3
0x10,4FD8	INT_CLEAR3	Interrupt clear register (by software) for FIFO queue 3
0x10,4FDC	INT_SET3	Interrupt set register (by software) for FIFO queue 3
0x10,4FE0	INT_STATUS4	Interrupt status register, combined with module status for TSUs
0x10,4FE4	INT_ENABLE4	Interrupt enable register for TSUs
0x10,4FE8	INT_CLEAR4	Interrupt clear register (by software) for TSUs
0x10,4FEC	INT_SET4	Interrupt set register (by software) for TSUs
0x10,4FF4	POWERDOWN	Powerdown mode, module clock switched off.
0x10,4FFC	Module ID	Module Identification and revision information

Remark: ALL programmable fields related to FIFO queue or TSU operation described next are assumed static when the relevant FIFO queue or TSU is enabled. This excludes the registers BASE1_PTRx and BASE2_PTRx and the PG_BUF_CTRLx.BUF_LEN field and the Interrupt control registers.

4.1 GPIO Mode Control Registers

Table 7: GPIO Mode Control Registers

Bit	Symbol	Access	Value	Description
Offset 0x10,4000	*Mode Control for GPIO pins 15—0*			
31:30	MC for GPIO number 15	R/W	0b11	The Mode Control (MC) bit pairs control the mode of the corresponding GPIO pin. The number portion of MCxx identifies the GPIO number. Refer to Table 1 on page 8-3. The following values apply to writing to all of the bit pairs: • 00 - retain current GPIO Mode of operation (will not overwrite current mode). Not readable • 01 - place Pin in primary function mode • 10 - place Pin in GPIO function mode • 11 - place Pin in GPIO function with Open-Drain Output mode
29:28	MC for GPIO number 14	R/W	0b11	
27:26	MC for GPIO number 13	R/W	0b11	
25:24	MC for GPIO number 12	R/W	0b11	
23:22	MC for GPIO number 11	R/W	0b11	
21:20	MC for GPIO number 10	R/W	0b11	
19:18	MC for GPIO number 09	R/W	0b11	
17:16	MC for GPIO number 08	R/W	0b11	
15:14	MC for GPIO number 07	R/W	0b11	
13:12	MC for GPIO number 06	R/W	0b11	
11:10	MC for GPIO number 05	R/W	0b11	
9:8	MC for GPIO number 04	R/W	0b11	
7:6	MC for GPIO number 03	R/W	0b11	
5:4	MC for GPIO number 02	R/W	0b11	
3:2	MC for GPIO number 01	R/W	0b11	
1:0	MC for GPIO number 00	R/W	0b11	
Offset 0x10,4004	*Mode Control for GPIO pins 31—16*			
31:30	MC for GPIO number 31	R/W	0b11	The Mode Control (MC) bit pairs control the mode of the corresponding GPIO pin. The number portion of MCxx identifies the GPIO number. Refer to Table 1 on page 8-3. The following values apply to writing to all of the bit pairs: • 00 - retain current GPIO Mode of operation (will not overwrite current mode). Not readable • 01 - place Pin in primary function mode • 10 - place Pin in GPIO function mode • 11 - place Pin in GPIO function with Open-Drain Output mode
29:28	MC for GPIO number 30	R/W	0b11	
27:26	MC for GPIO number 29	R/W	0b11	
25:24	MC for GPIO number 28	R/W	0b11	
23:22	MC for GPIO number 27	R/W	0b11	
21:20	MC for GPIO number 26	R/W	0b11	
19:18	MC for GPIO number 25	R/W	0b11	
17:16	MC for GPIO number 24	R/W	0b11	
15:14	MC for GPIO number 23	R/W	0b11	
13:12	MC for GPIO number 22	R/W	0b11	
11:10	MC for GPIO number 21	R/W	0b11	
9:8	MC for GPIO number 20	R/W	0b11	
7:6	MC for GPIO number 19	R/W	0b11	
5:4	MC for GPIO number 18	R/W	0b11	
3:2	MC for GPIO number 17	R/W	0b11	
1:0	MC for GPIO number 16	R/W	0b11	

Table 7: GPIO Mode Control Registers

Bit	Symbol	Access	Value	Description
Offset 0x10,4008	*Mode Control for GPIO pins 47—32*			
31:30	MC for GPIO number 47	R/W	0b11	The Mode Control (MC) bit pairs control the mode of the corresponding GPIO pin. The number portion of MCxx identifies the GPIO number. Refer to Table 1 on page 8-3. The following values apply to writing to all of the bit pairs: • 00 - retain current GPIO Mode of operation (will not overwrite current mode). Not readable • 01 - place Pin in primary function mode • 10 - place Pin in GPIO function mode • 11 - place Pin in GPIO function with Open-Drain Output mode
29:28	MC for GPIO number 46	R/W	0b11	
27:26	MC for GPIO number 45	R/W	0b11	
25:24	MC for GPIO number 44	R/W	0b11	
23:22	MC for GPIO number 43	R/W	0b11	
21:20	MC for GPIO number 42	R/W	0b11	
19:18	MC for GPIO number 41	R/W	0b11	
17:16	MC for GPIO number 40	R/W	0b11	
15:14	MC for GPIO number 39	R/W	0b11	
13:12	MC for GPIO number 38	R/W	0b11	
11:10	MC for GPIO number 37	R/W	0b11	
9:8	MC for GPIO number 36	R/W	0b11	
7:6	MC for GPIO number 35	R/W	0b11	
5:4	MC for GPIO number 34	R/W	0b11	
3:2	MC for GPIO number 33	R/W	0b11	
1:0	MC for GPIO number 32	R/W	0b11	
Offset 0x10,400C	*Mode Control for GPIO pins 60—48*			
31:26	Unused			The Mode Control (MC) bit pairs control the mode of the corresponding GPIO pin. The number portion of MCxx identifies the GPIO number. Refer to Table 1 on page 8-3. The following values apply to writing to all of the bit pairs: • 00 - retain current GPIO Mode of operation (will not overwrite current mode). Not readable • 01 - place Pin in primary function mode (see MUX table) • 10 - place Pin in GPIO function mode (see MUX table) • 11 - place Pin in GPIO function with Open-Drain Output mode
25:24	MC for GPIO number 60	R/W	0b11	
23:22	MC for GPIO number 59	R/W	0b11	
21:20	MC for GPIO number 58	R/W	0b11	
19:18	MC for GPIO number 57	R/W	0b11	
17:16	MC for GPIO number 56	R/W	0b11	
15:14	MC for GPIO number 55	R/W	0b11	
13:12	MC for GPIO number 54	R/W	0b11	
11:10	MC for GPIO number 53	R/W	0b11	
9:8	MC for GPIO number 52	R/W	0b11	
7:6	MC for GPIO number 51	R/W	0b11	
5:4	MC for GPIO number 50	R/W	0b11	
3:2	MC for GPIO number 49	R/W	0b11	
1:0	MC for GPIO number 48	R/W	0b11	

4.2 GPIO Data Control

Table 8: GPIO Data Control

Bit	Symbol	Access	Value	Description
Offset 0x10,4010	*MASK and IO Data for GPIO pins 15—0*			
31:16	MASK[15:0]	R/W	0x0000	See Table 3 on page 8-5 for descriptions of bit values.
15:0	IOD[15:0]	R/W	0xFFFF	
Offset 0x10,4014	*MASK and IO Data for GPIO pins 31—16*			
31:16	MASK[31:16]	R/W	0x0000	See Table 3 on page 8-5 for descriptions of bit values.
15:0	IOD[31:16]	R/W	0xFFFF	
Offset 0x10,4018	*MASK and IO Data for GPIO pins 47—32*			
31:16	MASK[47:32]	R/W	0x0000	See Table 3 on page 8-5 for descriptions of bit values.
15:0	IOD[47:32]	R/W	0xFFFF	
Offset 0x10,401C	*MASK and IO Data for GPIO pins 60—48*			
31:29	Unused			See Table 3 on page 8-5 for descriptions of bit values.
28:16	MASK[60:48]	R/W	0x0000	
15:13	Unused			See Table 3 on page 8-5 for descriptions of bit values.
12:0	IOD[60:48]	R/W	0xFFFF	

4.3 Readable Internal PNX15xx Series Signals

Table 9: Readable Internal PNX1500 Signals

Bit	Symbol	Access	Value	Description
Offset 0x10,4020	*Internal Signals*			
31:12	Unused	-	-	
11	last_word_q3	R	0	Reads value of GPIO last 32-bit word timestamp for Queue 3. This is also referenced as LAST_WORD3.
10	last_word_q2	R	0	Reads value of GPIO last 32-bit word timestamp for Queue 2. This is also referenced as LAST_WORD2.
9	last_word_q1	R	0	Reads value of GPIO last 32-bit word timestamp for Queue 1. This is also referenced as LAST_WORD1.
8	last_word_q0	R	0	Reads value of GPIO last 32-bit word timestamp for Queue 0. This is also referenced as LAST_WORD0.
7	vip1_eow_vbi	R	0	Reads value of VIP end of VBI window timestamp signal.
6	vip1_eow_vid	R	0	Reads value of VIP end of VID window timestamp signal.
5	spdi_tstamp2	R	1	Reads value of SPDIF IN timestamp 2 signal (Section 8.1 on page 3-27).
4	spdi_tstamp1	R	0	Reads value of SPDIF IN timestamp 1 (Word Select timestamp) signal.
3	spdo_tstamp	R	0	Reads value of SPDIF OUT timestamp signal.
2	ai1_tstamp	R	0	Reads value of Audio IN timestamp signal.
1	ao1_tstamp	R	0	Reads value of Audio OUT timestamp signal.
0	qvcp_tstamp	R	0	Reads value of QVCP timestamp signal.

4.4 Sampling and Pattern Generation Control Registers for the FIFO Queues

Table 10: Sampling and Pattern Generation Control Registers for the FIFO Queues

Bit	Symbol	Access	Value	Description
Offset 0x10,4024 -> 0x030 GPIO_EV<0-3>				
31	Unused		-	
30	EN_EV_TSTAMP	R/W	0	Enables an event timestamp signal to be generated whenever the last 32-bit word from a DMA buffer reaches the GPIO output pins. This field is only valid in Pattern Generating modes, i.e. FIFO_MODE[1] set to '1'.
29	EN_IR_CARRIER	R/W	0	This bit enables a sub-carrier for Ir transmission. FREQ_DIV[15:0] is combined with CARRIER_DIV[4:0] to generate sub-carrier and TX frequencies: 0 - Ir Carrier disabled, CARRIER_DIV[4:0] not used. 1 - Ir Carrier enabled, CARRIER_DIV[4:0] used. **Note:** This field is only valid in Pattern Generation using samples mode (FIFO_MODE=11) with EN_CLOCK_SEL disabled.
28	EN_IR_FILTER	R/W	0	This bit enables a received Ir signal to be filtered. No signal pulses less than the period programmed in IR_FILTER are passed through to the monitoring logic. **Note:** This field is only valid in Signal Sampling mode (FIFO_MODE=01) with EN_CLOCK_SEL disabled.
27:26	EN_CLOCK_SEL	R/W	0	Enables an input signal selected by CLOCK_SEL to be used as the external clock source: 00 - CLOCK_SEL disabled 10 - CLOCK_SEL disabled 01 - CLOCK_SEL enabled, sample on positive edge 11 - CLOCK_SEL enabled, sample on negative edge **Note:** This field is only valid in Signal Sampling mode (FIFO_MODE=01) and Pattern Generation using samples mode (FIFO_MODE=11).
25:24	EN_PAT_GEN_CLK	R/W	0	Enables the clock generated by the frequency divider to be sent out of the chip during pattern generation using samples and frequency divider mode: 00 - EN_PAT_GEN_CLK disabled 10 - EN_PAT_GEN_CLK disabled 01 - EN_PAT_GEN_CLK enabled, output the clock as is 11 - EN_PAT_GEN_CLK enabled, output the inverted clock **Note:** This field is only valid in Pattern Generation using samples (FIFO_MODE=11) and frequency divider (EN_CLOCK_SEL - disabled) mode
23:22	Unused			

Table 10: Sampling and Pattern Generation Control Registers for the FIFO Queues

Bit	Symbol	Access	Value	Description
21	EN_DDS_SOURCE	R/W	0	Enables the use of a DDS clock for Signal Sampling or Pattern Generation using samples and external clock (EN_CLOCK_SEL[26] = 1) mode: 0 - disabled 1 - enabled **Note:** This field is only valid in Signal Sampling mode (FIFO_MODE=01) and Pattern Generation using samples mode (FIFO_MODE=11)
20:18	CLOCK_SEL	R/W	0	In Signal Sampling / Pattern Generation using samples and external clock (EN_CLOCK_SEL [26] = 1) mode: This field selects the GPIO input pin to be used as the external clock. Refer to Section 4.15 for field values. **Note:** Only the GPIO[6:0] can be used. **Note:** If EN_DDS_SOURCE = 1, then, depending on the content of DDS_OUT_SEL register, one of the GPIO[6:4] pins may receive an internally generated DDS clock. This clock can then be selected with CLOCK_SEL. In Pattern Generation using samples and the frequency divider (EN_CLOCK_SEL[26] = 0) mode: This field selects which GPIO output pin to output the sampling frequency clock on. Refer to Section 4.15 for field values (note only GPIO[6:0] pins can be used). **Note:** This field is only valid in Signal Sampling mode (FIFO_MODE=01) and Pattern Generation using samples mode (FIFO_MODE=11). **Note:** The GPIO clock used for sampling or pattern generation must not be greater than 108 MHz.
17:16	EN_IO_SEL	R/W	0	This field selects how many GPIO pins should be sampled in one FIFO queue: 00 - IO_SEL_0 enabled: 1-bit samples 11 - IO_SEL_0 enabled: 1-bit samples 01 - IO_SEL_[1:0] enabled: 2-bit samples 10 - IO_SEL_[3:0] enabled: 4-bit samples **Note:** This field is only valid in Signal Sampling mode (FIFO_MODE=01) or Pattern Generation using samples mode (FIFO_MODE=11). In all other modes only IO_SEL_0 is enabled.

Table 10: Sampling and Pattern Generation Control Registers for the FIFO Queues

Bit	Symbol	Access	Value	Description
15:4	INTERVAL	R/W	0	Interval of silence. If a change is monitored on a signal and no more signal activity is monitored for a time equal to the interval of silence, writing to the current buffer is halted and a BUFx_RDY interrupt is generated. Writing continues to the alternate buffer. This field is only valid if FIFO_MODE[1:0] = 00. 0x000 - Disabled 0x001 - 1x128 13.5 MHz period, 9.48 µs 0x002 - 2x128 13.5 MHZ periods, 18.96 µs 0x003 - 3x128 13.5 MHZ periods, 28,44 µs .... 0x3FF - 1023x128 13.5 MHz periods, 9.69 ms .... 0xFFF - 4095x128 13.5 MHz periods, 38.8 ms **Note:** This field in only valid in Event Timestamping mode (FIFO_MOD E = 00 and EVENT_MODE != 00)
3:2	EVENT_MODE	R/W	0	Timestamping event mode: 00 - event detection disabled 01 - capture negative edge 10 - capture positive edge 11 - capture either edge NOTE: This field is valid in Event Timestamping mode (FIFO_MODE[1:0]=00)
1:0	FIFO_MODE	R/W	0	This bit selects what mode of operation the FIFO queue is in: **00** - Event Timestamping (or Disabled if EVENT_MODE[1:0] = 00) **01** - Signal Sampling **10** - Pattern Generation using timestamps. **11** - Pattern Generation using samples.
Offset 0x10,4064 -> 0x070 IO_SEL[a]<0-3>				
31:24	IO_SEL_3	R/W	0	This field selects a GPIO pin which should be merged with the GPIO pin selected by IO_SEL_0, IO_SEL_1 and IO_SEL_2 to enable 4-bit samples in one FIFO queue. **Note:** This field is only used in Signal Sampling mode and Pattern Generation using samples mode and is enabled by EN_IO_SEL
23:16	IO_SEL_2	R/W	0	This field selects a GPIO pin which should be merged with the GPIO pins selected by IO_SEL_0, IO_SEL_1 and IO_SEL_3 to enable 4-bit samples in one FIFO queue. **Note:** This field is only used in Signal Sampling mode and Pattern Generation using samples mode and is enabled by EN_IO_SEL
15:8	IO_SEL_1	R/W	0	This field selects a GPIO pin which should be merged with the GPIO pin selected by IO_SEL_0 to enable 2-bit samples in one FIFO queue. **Note:** This field is only used in Signal Sampling mode and Pattern Generation using samples mode and is enabled by EN_IO_SEL

Table 10: Sampling and Pattern Generation Control Registers for the FIFO Queues

Bit	Symbol	Access	Value	Description
7:0	IO_SEL_0	R/W	0	- In Signal Monitoring modes (FIFO_MODE[1]=0) this field selects the GPIO pin or internal global signal to be observed. Refer to Section 4.15 for field values. - In Pattern Generation modes (FIFO_MODE[1]=1) this field selects the GPIO pin which is to be driven. Refer to Section 4.15 for field values.
Offset 0x10,4074-> 0x080 PG_BUF_CTRL<0-3>				
31:18	Unused		-	
17:0	BUF_LEN	R/W	0	This field indicates how many valid 32-bit words s/w has written to a DMA buffer. When BUF1_RDY is cleared the BUF_LEN value is loaded for DMA buffer 1. When BUF2_RDY is cleared the BUF_LEN value is loaded for DMA buffer 2. The 18-bit field allows DMA buffer lengths as large as 1MB. 0x00000 - 1 32-bit word 0x00001 - 2 32-bit words ..... 0x3FFFF - 262143 32-bit words **Note:** This field is valid in Pattern Generation modes (FIFO_MODE[1]=1)
Offset 0x10,4084 -> 0x090 BASE1_PTR<0-3>				
31:2	BASE1_PTR	R/W	0	Start byte address for DMA buffer 1 of FIFO queue. The base address must be 64-byte aligned.
1:0	Unused		-	
Offset 0x10,4094-> 0x0A0 BASE2_PTR<0-3>				
31:2	BASE2_PTR	R/W	0	Start byte address for DMA buffer 2 of FIFO queue. The base address must be 64-byte aligned.
1:0	Unused		-	
Offset 0x10,40A4-> 0x0 B0SIZE<0-3>				
31:20	Unused		-	
13:0	SIZE	R/W	0	Size, in 64 bytes multiples, of each of the 2 DMA buffers: 0x0001 = 64 bytes 0x0002 = 128 bytes .... 0x3FFF = 1 Megabytes
Offset 0x10,40B4 -> 0x0C0 DIVIDER<0-3>				
31:23	Unused		-	

Table 10: Sampling and Pattern Generation Control Registers for the FIFO Queues

Bit	Symbol	Access	Value	Description
22:21	DUTY_CYCLE	R/W	0	This field selects the duty cycle for sub-carriers. 00 - 33% duty-cycle 01 - 50% duty-cycle 10 - 66% duty cycle 11 - Illegal **Note:** This field is only valid in pattern generation modes and when EN_IR_CARRIER = 1

Table 10: Sampling and Pattern Generation Control Registers for the FIFO Queues

Bit	Symbol	Access	Value	Description
20:16	CARRIER_DIV (when FIFO_MODE[1] =1)	R/W	00	Used in Ir TX applications if a sub-carrier is required for transmission. To enable this divider EN_IR_CARRIER=1. If enabled, the Ir sub-carrier frequency is defined by programming FREQ_DIV and the 'ONTIME' is defined by FREQ_DIV x CARRIER_DIV. 0x00 - Disabled. 0x01 - Disabled. 0x02 - Sampling frequency is FREQ_DIV/2 ..... 0x1F - Sampling frequency is FREQ_DIV/31
18:16	IR_FILTER (when FIFO_MODE[1] =0)		00	Used in Ir RX applications to filter a received Ir signal. To enable this divider EN_IR_FILTER=1. If enabled, Ir pulses greater than IR_FILTER are passed through to the signal monitoring logic. 0x0 - 54/108 MHz, 0.5 µs 0x1 - 108/108 MHz, 1.0 µs 0x2 - 162/108 MHz, 1.5 µs 0x3 - 216/108 MHz, 2.0 µs 0x4 - 270/108 MHz, 2.5 µs 0x5 - 324/108 MHz, 3.0 µs 0x6 - 378/108 MHz, 3.5 µs 0x7 - 432/108 MHz, 4.0 µs **Note:** The filter operates on one input per queue, this bit is the input selected by IO_SEL[7:0]. If used in multi-bit sampling modes (IO_SEL_EN = 01 or 10) be aware that the filtered signal is delayed by the selected IR_FILTER value with respect to the other signals sampled in the queue.

Table 10: Sampling and Pattern Generation Control Registers for the FIFO Queues

Bit	Symbol	Access	Value	Description
15:0	FREQ_DIV	R/W	0000	16-bit Frequency Divider for signal sampling and pattern generation using samples. If EN_CARRIER_FREQ = 0 Sampling Freq. = 108MHz/FREQ_DIV 0x0000 - Disabled. 0x0001 - Sampling frequency is 108 MHz, 0x0002 - Sampling/Carrier frequency is 54 MHz ..... 0xFFFF - Sampling/Carrier frequency is 1.648 kHz If EN_CARRIER_FREQ = 1 Carrier Freq. = 54 MHz/FREQ_DIV 0x0000 - Disabled. 0x0001 - Carrier frequency is 54 MHz 0x0002 - Carrier frequency is 26 MHz ..... 0xFFFF - Carrier frequency is 824 Hz

[a] IO_SEL cannot be written to unless all signal generation and monitoring is disabled (FIFO_MODE = 00 and EVENT_MODE = 00).

4.5 Signal and Event Monitoring Control Registers for the Timestamp Units

Table 11: Signal and Event Monitoring Control Registers for the Timestamp Units

Bit	Symbol	Access	Value	Description
Offset 0x10,4034-> 0x060	*GPIO_EV<4-15>*			
31:10	Unused		-	
9:2	IO_SEL[a]	R/W	0	This field selects the GPIO pin or internal global signal to be monitored. Refer to Section 4.15 for field values.
1:0	EVENT_MODE	R/W	0	Timestamping event mode: 00 - event detection disabled 01 - capture negative edge 10 - capture positive edge 11 - capture either edge

[a] IO_SEL cannot be written to unless timestamping is disabled (EVENT_MODE=00).

4.6 Timestamp Unit Registers

Table 12: Timestamp Unit Registers

Bit	Symbol	Access	Value	Description
Offset 0x10,40C4->0x0F0	*TSU<0-11>*			
31	Direction	R	0	This field indicates the direction of the event which occurred: 0 - a falling edge 1 - a rising edge
30:0	Timestamp	R	0	This field holds the 31-bit timestamp.

4.7 GPIO Time Counter

Table 13: GPIO Time Counter

Bit	Symbol	Access	Value	Description
Offset 0x10,40F4	*TIME_CTR*			
31	Unused		-	
30:0	TIME_CTR	R	0	GPIO master time counter. This counter is incremented at a frequency of 13.5 MHz.

4.8 GPIO TM3260 Timer Input Select

Table 14: GPIO TM3260 Timer Input Select

Bit	Symbol	Access	Value	Description
Offset 0x10,4OF8	*TIMER_IO_SEL*			
31:16	Unused		-	
15:8	TIMER_IO_SEL1	R/W	0	Selects a GPIO pin or an internal signal to be output onto gpio_timer[1] signal which is connected to the GPIO_TIMER1 TM3260 timer source. See Section 4.15 for valid field values, i.e signal selection.
7:0	TIMER_IO_SEL0	R/W	0	Selects a GPIO pin or an internal signal to be output onto gpio_timer[0] signal which is connected to the GPIO_TIMER0 TM3260 timer source. See Section 4.15 for valid field values, i.e signal selection.

4.9 GPIO Interrupt Status

Table 15: GPIO Interrupt Status

Bit	Symbol	Access	Value	Description
Offset 0x10,40FC	*VIC_INT_STATUS*			
31:5	Unused		-	
4	TSU Status	R	0	TSU status bit for all interrupts of all 12 TSUs (ORed together)
3	FIFO Queue 3 status	R	0	FIFO Queue 3 status bit for all interrupts (ORed together)
2	FIFO Queue 2 status	R	0	FIFO Queue 2 status bit for all interrupts (ORed together)
1	FIFO Queue 1 status	R	0	FIFO Queue 1 status bit for all interrupts (ORed together)
0	FIFO Queue 0 status	R	0	FIFO Queue 0 status bit for all interrupts (ORed together)

4.10 Clock Out Select

Table 16: Clock Out Select

Bit	Symbol	Access	Value	Description
Offset 0x10,4100	*DDS_OUT_SEL*			
31:3	Unused		-	
2:0	DDS_OUT_SEL	R/W	0	Controls if the GPIO[14:12] pins are used as clock outputs for the system board. By default these pins are configured as inputs and act as regular GPIO pins: 0x000 - Disabled DDS_OUT_SEL[0] 0 - GPIO[12] is a GPIO pin 1 - DDS6 clk is sent out on GPIO[12] DDS_OUT_SEL[1] 0 - GPIO[13] is a GPIO pin 1 - DDS1 or DDS5 clk is sent out on GPIO[13] DDS_OUT_SEL[3] 0 - GPIO[14] is a GPIO pin 1 - DDS0 or DDS2 clk is sent out on GPIO[14]

4.11 GPIO Interrupt Registers for the FIFO Queues (One for each FIFO Queue)

Table 17: GPIO Interrupt Registers for the FIFO Queues (One for each FIFO Queue)

Bit	Symbol	Access	Value	Description
Offset 0x10,4FA0+[4*<0-3>]	***INT_STATUS<0-3>***			
31:14	VALID_PTR	R	0	This field indicates how many valid 32-bit words of data have been written by the GPIO module to the current DMA buffer: 0x00000 - 1 32-bit word 0x00001 - 2 32-bit words ..... 0x3FFFF - 262143 32-bit words When a BUFx_RDY signal occurs the address to read from can be calculated using VALID_PTR. This field is only updated by the GPIO after the relevant BUFx_RDY flag is cleared by software. This field is valid in Signal Monitoring modes only.
5:4	Reserved	R	0	
3	INT_OE	R	0	Internal overrun error. Internal GPIO data buffer has overrun before data has been written out to external DMA buffer. Data has been lost. ONLY USED in signal monitoring modes.
2	FIFO_OE	R	0	FIFO overrun error. A new, empty, DMA buffer was not supplied in time. ONLY USED in signal monitoring modes.
1	BUF2_RDY	R	0	-In Signal Monitoring modes: DMA buffer 2 is ready to be read. It is either full or an interval of silence has occurred. -In Pattern Generation modes: All contents of DMA buffer 2 have been read.
0	BUF1_RDY	R	0	-In Signal Monitoring modes: DMA buffer1 is ready to be read. It is either full or an interval of silence has occurred. -In Pattern Generation modes: All contents of DMA buffer 1 have been read.
Offset 0x10,4FA4+[4*<0-3>]	***INT_ENABLE<0-3>***			
31:4	Unused		-	
3	INT_OE_EN	R/W	0	Active high Internal overrun error interrupt enable for queue <0-3>.
2	FIFO_OE_EN	R/W	0	Active high FIFO overrun error interrupt enable for queue <0-3>.
1	BUF2_RDY_EN	R/W	0	Active high Buffer 2 ready interrupt enable for queue <0-3>.
0	BUF1_RDY_EN	R/W	0	Active high Buffer 1 ready interrupt enable for queue <0-3>.
Offset 0x10,4FA8+[4*<0-3>]	***INT_CLEAR<0-3>***			
31:4	Unused		-	
3	INT_OE_CLR	W	0	Active high internal overrun error interrupt clear for queue <0-3>.
2	FIFO_OE_CLR	W	0	Active high FIFO overrun error interrupt clear for queue <0-3>.
1	BUF2_RDY_CLR	W	0	Active high Buffer 2 ready interrupt clear for queue <0-3>.
0	BUF1_RDY_CLR	W	0	Active high Buffer 1 ready interrupt clear for queue <0-3>.
Offset 0x10,4FAC+[4*<0-3>]	***INT_SET<0-3>***			
31:4	Unused		-	

Table 17: GPIO Interrupt Registers for the FIFO Queues (One for each FIFO Queue)

Bit	Symbol	Access	Value	Description
3	INT_OE_SET	W	0	Active high internal overrun error interrupt set for queue <0-3>.
2	FIFO_OE_SET	W	0	Active high FIFO overrun error interrupt set for queue <0-3>.
1	BUF2_RDY_SET	W	0	Active high Buffer 2 ready interrupt set for queue <0-3>.
0	BUF1_RDY_SET	W	0	Active high Buffer 1 ready interrupt set for queue <0-3>.

4.12 GPIO Module Status Register for all 12 Timestamp Units

Table 18: GPIO Module Status Register for all 12 Timestamp Units

Bit	Symbol	Access	Value	Description
Offset 0x10,4FE0	*INT_STATUS4*			
31:24	Unused		-	
23	INT_OE_11	R	0	Internal overrun error in TSU[a] 11. Data in TSU overwritten before read by CPU (i.e before DATA_VALID interrupt was cleared).
22	INT_OE_10	R	0	Internal overrun error in TSU 10. Data in TSU overwritten before read by CPU (i.e before DATA_VALID interrupt was cleared).
21	INT_OE_9	R	0	Internal overrun error in TSU 9. Data in TSU overwritten before read by CPU (i.e before DATA_VALID interrupt was cleared).
20	INT_OE_8	R	0	Internal overrun error in TSU 8. Data in TSU overwritten before read by CPU (i.e before DATA_VALID interrupt was cleared).
19	INT_OE_7	R	0	Internal overrun error in TSU 7. Data in TSU overwritten before read by CPU (i.e before DATA_VALID interrupt was cleared).
18	INT_OE_6	R	0	Internal overrun error in TSU 6. Data in TSU overwritten before read by CPU (i.e before DATA_VALID interrupt was cleared).
17	INT_OE_5	R	0	Internal overrun error in TSU 5. Data in TSU overwritten before read by CPU (i.e before DATA_VALID interrupt was cleared).
16	INT_OE_4	R	0	Internal overrun error in TSU 4. Data in TSU overwritten before read by CPU (i.e before DATA_VALID interrupt was cleared).
15	INT_OE_3	R	0	Internal overrun error in TSU 3. Data in TSU overwritten before read by CPU (i.e before DATA_VALID interrupt was cleared).
14	INT_OE_2	R	0	Internal overrun error in TSU 2. Data in TSU overwritten before read by CPU (i.e before DATA_VALID interrupt was cleared).
13	INT_OE_1	R	0	Internal overrun error in TSU 1. Data in TSU overwritten before read by CPU (i.e before DATA_VALID interrupt was cleared).
12	INT_OE_0	R	0	Internal overrun error in TSU 0. Data in TSU overwritten before read by CPU (i.e before DATA_VALID interrupt was cleared).
11	DATA_VALID_11	R	0	Data in TSU 11 is ready to be read.
10	DATA_VALID_10	R	0	Data in TSU 10 is ready to be read.
9	DATA_VALID_9	R	0	Data in TSU 9 is ready to be read.
8	DATA_VALID_8	R	0	Data in TSU 8 is ready to be read.
7	DATA_VALID_7	R	0	Data in TSU 7 is ready to be read.
6	DATA_VALID_6	R	0	Data in TSU 6 is ready to be read.

Table 18: GPIO Module Status Register for all 12 Timestamp Units

Bit	Symbol	Access	Value	Description
5	DATA_VALID_5	R	0	Data in TSU 5 is ready to be read.
4	DATA_VALID_4	R	0	Data in TSU 4 is ready to be read.
3	DATA_VALID_3	R	0	Data in TSU 3 is ready to be read.
2	DATA_VALID_2	R	0	Data in TSU 2 is ready to be read.
1	DATA_VALID_1	R	0	Data in TSU 1 is ready to be read.
0	DATA_VALID_0	R	0	Data in TSU 0 is ready to be read.
Offset 0x10,4FE4	***INT_ENABLE4***			
31:24	Unused		-	
23	INT_OE_11_EN	R/W	0	Internal overrun interrupt enable register for TSU 11 0 - Interrupt disabled 1 - Interrupt enabled
22	INT_OE_10_EN	R/W	0	Internal overrun interrupt enable register for TSU 10 0 - Interrupt disabled 1 - Interrupt enabled
21	INT_OE_9_EN	R/W	0	Internal overrun interrupt enable register for TSU 9 0 - Interrupt disabled 1 - Interrupt enabled
20	INT_OE_8_EN	R/W	0	Internal overrun interrupt enable register for TSU 8 0 - Interrupt disabled 1 - Interrupt enabled
19	INT_OE_7_EN	R/W	0	Internal overrun interrupt enable register for TSU 7 0 - Interrupt disabled 1 - Interrupt enabled
18	INT_OE_6_EN	R/W	0	Internal overrun interrupt enable register for TSU 6 0 - Interrupt disabled 1 - Interrupt enabled
17	INT_OE_5_EN	R/W	0	Internal overrun interrupt enable register for TSU 5 0 - Interrupt disabled 1 - Interrupt enabled
16	INT_OE_4_EN	R/W	0	Internal overrun interrupt enable register for TSU 4 0 - Interrupt disabled 1 - Interrupt enabled
15	INT_OE_3_EN	R/W	0	Internal overrun interrupt enable register for TSU 3 0 - Interrupt disabled 1 - Interrupt enabled
14	INT_OE_2_EN	R/W	0	Internal overrun interrupt enable register for TSU 2 0 - Interrupt disabled 1 - Interrupt enabled
13	INT_OE_1_EN	R/W	0	Internal overrun interrupt enable register for TSU 1 0 - Interrupt disabled 1 - Interrupt enabled

Table 18: GPIO Module Status Register for all 12 Timestamp Units

Bit	Symbol	Access	Value	Description
12	INT_OE_0_EN	R/W	0	Internal overrun interrupt enable register for TSU 0 0 - Interrupt disabled 1 - Interrupt enabled
11	DATA_VALID_11_EN	R/W	0	Data valid interrupt enable register for TSU 11 0 - Interrupt disabled 1 - Interrupt enabled
10	DATA_VALID_10_EN	R/W	0	Data valid interrupt enable register for TSU 10 0 - Interrupt disabled 1 - Interrupt enabled
9	DATA_VALID_9_EN	R/W	0	Data valid interrupt enable register for TSU 9 0 - Interrupt disabled 1 - Interrupt enabled
8	DATA_VALID_8_EN	R/W	0	Data valid interrupt enable register for TSU 8 0 - Interrupt disabled 1 - Interrupt enabled
7	DATA_VALID_7_EN	R/W	0	Data valid interrupt enable register for TSU 7 0 - Interrupt disabled 1 - Interrupt enabled
6	DATA_VALID_6_EN	R/W	0	Data valid interrupt enable register for TSU 6 0 - Interrupt disabled 1 - Interrupt enabled
5	DATA_VALID_5_EN	R/W	0	Data valid interrupt enable register for TSU 5 0 - Interrupt disabled 1 - Interrupt enabled
4	DATA_VALID_4_EN	R/W	0	Data valid interrupt enable register for TSU 4 0 - Interrupt disabled 1 - Interrupt enabled
3	DATA_VALID_3_EN	R/W	0	Data valid interrupt enable register for TSU 3 0 - Interrupt disabled 1 - Interrupt enabled
2	DATA_VALID_2_EN	R/W	0	Data valid interrupt enable register for TSU 2 0 - Interrupt disabled 1 - Interrupt enabled
1	DATA_VALID_1_EN	R/W	0	Data valid interrupt enable register for TSU 1 0 - Interrupt disabled 1 - Interrupt enabled
0	DATA_VALID_0_EN	R/W	0	Data valid interrupt enable register for TSU 0 0 - Interrupt disabled 1 - Interrupt enabled
Offset 0x10,4FE8	***INT_CLEAR4***			
31:24	Unused		-	
23	INT_OE_11_CLR	W	0	Active high clear for internal overrun interrupt for TSU 11.
22	INT_OE_10_CLR	W	0	Active high clear for internal overrun interrupt for TSU 10.

Table 18: GPIO Module Status Register for all 12 Timestamp Units

Bit	Symbol	Access	Value	Description
21	INT_OE_9_CLR	W	0	Active high clear for internal overrun interrupt for TSU 9.
20	INT_OE_8_CLR	W	0	Active high clear for internal overrun interrupt for TSU 8.
19	INT_OE_7_CLR	W	0	Active high clear for internal overrun interrupt for TSU 7.
18	INT_OE_6_CLR	W	0	Active high clear for internal overrun interrupt for TSU 6.
17	INT_OE_5_CLR	W	0	Active high clear for internal overrun interrupt for TSU 5.
16	INT_OE_4_CLR	W	0	Active high clear for internal overrun interrupt for TSU 4.
15	INT_OE_3_CLR	W	0	Active high clear for internal overrun interrupt for TSU 3.
14	INT_OE_2_CLR	W	0	Active high clear for internal overrun interrupt for TSU 2.
13	INT_OE_1_CLR	W	0	Active high clear for internal overrun interrupt for TSU 1.
12	INT_OE_0_CLR	W	0	Active high clear for internal overrun interrupt for TSU 0.
11	DATA_VALID_11_CLR	W	0	Active high clear for data valid interrupt for TSU 11.
10	DATA_VALID_10_CLR	W	0	Active high clear for data valid interrupt for TSU 10.
9	DATA_VALID_9_CLR	W	0	Active high clear for data valid interrupt for TSU 9.
8	DATA_VALID_8_CLR	W	0	Active high clear for data valid interrupt for TSU 8.
7	DATA_VALID_7_CLR	W	0	Active high clear for data valid interrupt for TSU 7.
6	DATA_VALID_6_CLR	W	0	Active high clear for data valid interrupt for TSU 6.
5	DATA_VALID_5_CLR	W	0	Active high clear for data valid interrupt for TSU 5.
4	DATA_VALID_4_CLR	W	0	Active high clear for data valid interrupt for TSU 4.
3	DATA_VALID_3_CLR	W	0	Active high clear for data valid interrupt for TSU 3.
2	DATA_VALID_2_CLR	W	0	Active high clear for data valid interrupt for TSU 2.
1	DATA_VALID_1_CLR	W	0	Active high clear for data valid interrupt for TSU 1.
0	DATA_VALID_0_CLR	W	0	Active high clear for data valid interrupt for TSU 0.
Offset 0x10,4FEC	***INT_SET4***			
31:24	Unused		-	
23	INT_OE_11_SET	W	0	Active high set for internal overrun interrupt for TSU 11.
22	INT_OE_10_SET	W	0	Active high set for internal overrun interrupt for TSU 10.
21	INT_OE_9_SET	W	0	Active high set for internal overrun interrupt for TSU 9.
20	INT_OE_8_SET	W	0	Active high set for internal overrun interrupt for TSU 8.
19	INT_OE_7_SET	W	0	Active high set for internal overrun interrupt for TSU 7.
18	INT_OE_6_SET	W	0	Active high set for internal overrun interrupt for TSU 6.
17	INT_OE_5_SET	W	0	Active high set for internal overrun interrupt for TSU 5.
16	INT_OE_4_SET	W	0	Active high set for internal overrun interrupt for TSU 4.
15	INT_OE_3_SET	W	0	Active high set for internal overrun interrupt for TSU 3.
14	INT_OE_2_SET	W	0	Active high set for internal overrun interrupt for TSU 2.
13	INT_OE_1_SET	W	0	Active high set for internal overrun interrupt for TSU 1.
12	INT_OE_0_SET	W	0	Active high set for internal overrun interrupt for TSU 0.
11	DATA_VALID_11_SET	W	0	Active high set for data valid interrupt for TSU 11.
10	DATA_VALID_10_SET	W	0	Active high set for data valid interrupt for TSU 10.

Table 18: GPIO Module Status Register for all 12 Timestamp Units

Bit	Symbol	Access	Value	Description
9	DATA_VALID_9_SET	W	0	Active high set for data valid interrupt for TSU 9.
8	DATA_VALID_8_SET	W	0	Active high set for data valid interrupt for TSU 8.
7	DATA_VALID_7_SET	W	0	Active high set for data valid interrupt for TSU 7.
6	DATA_VALID_6_SET	W	0	Active high set for data valid interrupt for TSU 6.
5	DATA_VALID_5_SET	W	0	Active high set for data valid interrupt for TSU 5.
4	DATA_VALID_4_SET	W	0	Active high set for data valid interrupt for TSU 4.
3	DATA_VALID_3_SET	W	0	Active high set for data valid interrupt for TSU 3.
2	DATA_VALID_2_SET	W	0	Active high set for data valid interrupt for TSU 2.
1	DATA_VALID_1_SET	W	0	Active high set for data valid interrupt for TSU 1.
0	DATA_VALID_0_SET	W	0	Active high set for data valid interrupt for TSU 0.

[a] TSU = Timestamp Unit

4.13 GPIO POWERDOWN

Table 19: GPIO POWERDOWN

Bit	Symbol	Access	Value	Description
Offset 0x10,4FF4	*POWERDOWN*			
31	POWERDOWN	R/W	0	0 = Normal operation of peripheral. This is the reset value 1 = Module is powerdown and module clock can be removed. Module must respond to all reads. Generate e.g. 0xDEADABBA (except for reads of the powerdown bit!) Module should generate ERR ack on writes. (except for writes to the powerdown bit!)
30:0	Unused		-	

4.14 GPIO Module ID

Table 20: GPIO Module ID

Bit	Symbol	Access	Value	Description
Offset 0x10,4FFC	*MODULE_ID*			
31:16	MODULE ID	R	0xA065	16-bit Module identification ID.
15:12	MAJOR_REV	R	0x0	8-bit Major Revision identification ID.
11:8	MINOR_REV	R	0x1	8-bit Minor Revision identification ID.
7:0	APERTURE	R	0x0	Encoded as: Aperture size = 4K*(bit_value+1). The bit value is reset to 0 meaning a 4K aperture.

4.15 GPIO IO_SEL Selection Values

Table 21: GPIO IO_SEL Selection Values

Signal	CLOCK_SEL/ IO_SEL (hex)	MODE
LAST_WORD3	0x48	Signal Monitoring
LAST_WORD2	0x47	Signal Monitoring
LAST_WORD1	0x46	Signal Monitoring
LAST_WORD0	0x45	Signal Monitoring
vip1_eow_vbi	0x44	Signal Monitoring
vip1_eow_vid	0x43	Signal Monitoring
spdi_tstamp2	0x42	Signal Monitoring
spdi_tstamp1	0x41	Signal Monitoring
spdo_tstamp	0x40	Signal Monitoring
ai1_tstamp	0x3F	Signal Monitoring
ao1_tstamp	0x3E	Signal Monitoring
qvcp_tstamp	0x3D	Signal Monitoring
FGPO_REC_SYNC	0x3C	Signal Monitoring; Pattern Generation
VDI_V2	0x3B	Signal Monitoring; Pattern Generation
VDI_V1	0x3A	Signal Monitoring; Pattern Generation

Table 21: GPIO IO_SEL Selection Values

Signal	CLOCK_SEL/ IO_SEL (hex)	MODE
SPDIF_O	0x39	Signal Monitoring; Pattern Generation
SPDIF_I	0x38	Signal Monitoring; Pattern Generation
VDO_AUX	0x37	Signal Monitoring; Pattern Generation
VDO_D[33]	0x36	Signal Monitoring; Pattern Generation
VDO_D[32]	0x35	Signal Monitoring; Pattern Generation
VDI_D[33]	0x34	Signal Monitoring; Pattern Generation
VDI_D[32]	0x33	Signal Monitoring; Pattern Generation
LAN_MDC	0x32	Signal Monitoring; Pattern Generation
LAN_MDIO	0x31	Signal Monitoring; Pattern Generation
LAN_RX_ER	0x30	Signal Monitoring; Pattern Generation
LAN_RX_DV	0x2F	Signal Monitoring; Pattern Generation
LAN_RXD[3]	0x2E	Signal Monitoring; Pattern Generation
LAN_RXD[2]	0x2D	Signal Monitoring; Pattern Generation
LAN_RXD[1]	0x2C	Signal Monitoring; Pattern Generation
LAN_RXD[0]	0x2B	Signal Monitoring; Pattern Generation
LAN_COL	0x2A	Signal Monitoring; Pattern Generation
LAN_CRS	0x29	Signal Monitoring; Pattern Generation
LAN_TX_ER	0x28	Signal Monitoring; Pattern Generation
LAN_TXD[3]	0x27	Signal Monitoring; Pattern Generation
LAN_TXD[2]	0x26	Signal Monitoring; Pattern Generation
LAN_TXD[1]	0x25	Signal Monitoring; Pattern Generation
LAN_TXD[0]	0x24	Signal Monitoring; Pattern Generation
LAN_TX_EN	0x23	Signal Monitoring; Pattern Generation
XIO_D[7]	0x22	Signal Monitoring; Pattern Generation
XIO_D[6]	0x21	Signal Monitoring; Pattern Generation
XIO_D[5]	0x20	Signal Monitoring; Pattern Generation
XIO_D[4]	0x1F	Signal Monitoring; Pattern Generation
XIO_D[3]	0x1E	Signal Monitoring; Pattern Generation
XIO_D[2]	0x1D	Signal Monitoring; Pattern Generation
XIO_D[1]	0x1C	Signal Monitoring; Pattern Generation
XIO_D[0]	0x1B	Signal Monitoring; Pattern Generation
XIO_ACK	0x1A	Signal Monitoring; Pattern Generation
AO_SD[3]	0x19	Signal Monitoring; Pattern Generation
AO_SD[2]	0x18	Signal Monitoring; Pattern Generation
AO_SD[1]	0x17	Signal Monitoring; Pattern Generation
AO_SD[0]	0x16	Signal Monitoring; Pattern Generation
AO_WS	0x15	Signal Monitoring; Pattern Generation

Table 21: GPIO IO_SEL Selection Values

Signal	CLOCK_SEL/ IO_SEL (hex)	MODE
AI_SD[3]	0x14	Signal Monitoring; Pattern Generation
AI_SD[2]	0x13	Signal Monitoring; Pattern Generation
AI_SD[1]	0x12	Signal Monitoring; Pattern Generation
AI_SD[0]	0x11	Signal Monitoring; Pattern Generation
AI_WS	0x10	Signal Monitoring; Pattern Generation
GPIO[15]	0x0F	Signal Monitoring; Pattern Generation
GPIO[14]	0x0E	Signal Monitoring; Pattern Generation
GPIO[13]	0x0D	Signal Monitoring; Pattern Generation
GPIO[12]	0x0C	Signal Monitoring; Pattern Generation
GPIO[11]	0x0B	Signal Monitoring; Pattern Generation
GPIO[10]	0x0A	Signal Monitoring; Pattern Generation
GPIO[9]	0x09	Signal Monitoring; Pattern Generation
GPIO[8]	0x08	Signal Monitoring; Pattern Generation
GPIO[7]	0x07	Signal Monitoring; Pattern Generation
GPIO[6]	0x06	Signal Monitoring; Pattern Generation
GPIO[5]	0x05	Signal Monitoring; Pattern Generation
GPIO[4]	0x04	Signal Monitoring; Pattern Generation
GPIO[3]	0x03	Signal Monitoring; Pattern Generation
GPIO[2]	0x02	Signal Monitoring; Pattern Generation
GPIO[1]	0x01	Signal Monitoring; Pattern Generation
GPIO[0]	0x00	Signal Monitoring; Pattern Generation

Chapter 9: DDR Controller

PNX15xx Series Data Book – Volume 1 of 1

Rev. 2 — 1 December 2004 Product data sheet

1. Introduction

The DDR Controller is used to interface to off-chip DDR memory.

The primary features of the DDR SDRAM Controller include:

- 16- or 32-bit data bus width on DDR SDRAM memory side
- two MTL ports (one for the DMA memory traffic, one for the CPU)
- Supports x8, x16 and x32 memory devices
- Supports 64-Mbit, 128-Mbit, 256-Mbit and 512-Mbit DDR SDRAM memory devices
- Supports up to 2 ranks (physical banks) of memory devices
- Maximum of 8 open pages
- Maximum address range of 256 MBytes
- Halt modes to allow for power consumption reduction
- Programmable DDR SDRAM timing parameters that support DDR SDRAM memory devices up to 200 MHz
- Programmable bank mapping scheme to potentially improve bandwidth utilization (see Section 2.3.1).

The DDR controller module includes an arbiter which arbitrates between the DDR burst commands coming from the two different MTL ports. After arbitration, the DDR burst command selected by the arbiter is put in a 5-entry FIFO. The DDR module has a refresh counter to keep track of the refresh timing. The DDR module keeps track of the open pages in the DDR memories. Up to two DDR ranks (with 4 banks each) are supported resulting in a total of eight pages. The DDR command generator decides upon which command (refresh, precharge, activate, read, or write) to generate based on the information in the 5-entry FIFO, the state of the refresh counter, and the state of the DDR memories as indicated by the open page table.

The PNX15xx Series DDR controller follows the JEDEC specifications, [1][2].

2. Functional Description

Refer to Chapter for electrical and load constraints.

2.1 Start and Warm Start

There are two different start modes for the DDR SDRAM Controller: start, and the warm start. MMIO register IP_2031_CTL provides the interface to start the DDR controller.

2.1.1 The Start Mode

The START field of MMIO register IP_2031_CTL is used to trigger the start mode of the DDR SDRAM Controller. This mode is the common start mode. It is used when neither the DDR controller nor the DDR devices are yet initialized. This is the normal condition after a system reset has occurred. The MMIO registers that determine the timing and characteristics of the DDR memories should be programmed prior to the start action is triggered, since these register values may be used to configure the external DDR memories. The normal sequence of actions to start the DDR controller is to program the MMIO registers that configure the different parameters of the DDR memory devices and then set the START field of MMIO register IP_2031_CTL to ‘1’. This mode is used by the boot scripts.

Sequence of Actions During the Start Mode

During start (not warm start), the DDR SDRAM Controller performs the following sequence of actions:

- Apply a NOP command
- Precharge all command
- Load extended mode register
- Load mode register, with DLL reset
- 256 cycles delay for DDL.
- Precharge all command
- Auto refresh command
- Auto refresh command
- Load mode register, with DLL reset deactivated
- 256 cycles delay

2.1.2 Warm Start

The Warm start mode is a special mode where the DDR controller initializes itself but does not initialize the DDR devices. This mode is used in applications where the power of the PNX15xx Series is shutdown after the DDR devices have been sent to self-refresh mode. In that state the DDR devices remained powered and therefore they retain the data and the configuration. Once the PNX15xx Series power supplies are back on and an external reset is applied, the DDR controller can be started by asserting the WARM_START field of MMIO register IP_2031_CTL. By doing so the DDR controller configures itself without configuring the DDR devices. Instead the

DDR controller, once configured, executes an exit of self-refresh mode which starts back on the DDR devices. There is no boot scripts provision for this mode, therefore an external eeprom is required to activate this mode.

2.1.3 Observing Start State

The START and WARM_START fields of MMIO register IP_2031_CTL will be set to '0' when the respective start action has completed. Do not perform a start action while the DDR controller is still busy performing a previous start action.

2.2 Arbitration

The DDR SDRAM Controller provides an arbiter between the DMA traffic (generated by the PNX15xx Series modules) and the TM3260 CPU as pictured in Section 1 on page 9-3.

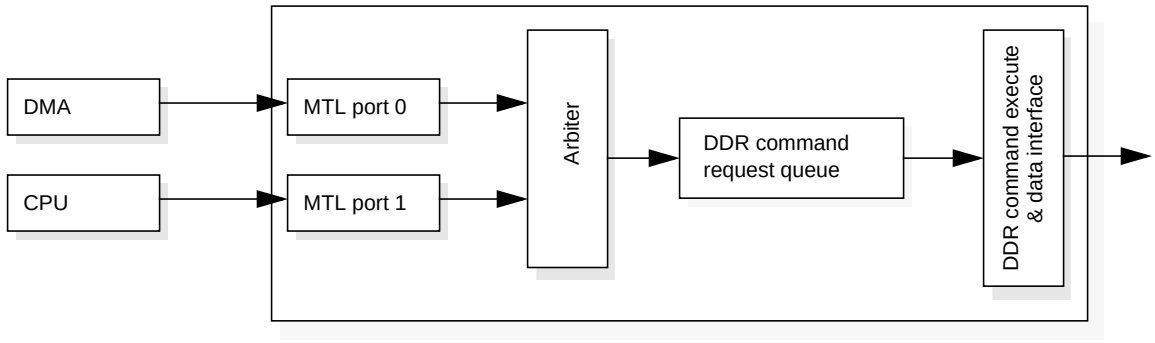


Figure 1: The two MTL Ports of the DDR SDRAM Controller

The DDR SDRAM Controller arbiter is responsible for scheduling between MTL transaction requests from the different MTL ports. The arbitration scheme has been optimized to achieve a high DDR bandwidth efficiency (at the cost of DDR latency).

2.2.1 The First Level of Arbitration: Between the DMA and the CPU

The arbitration flow is pictured in Figure 2.

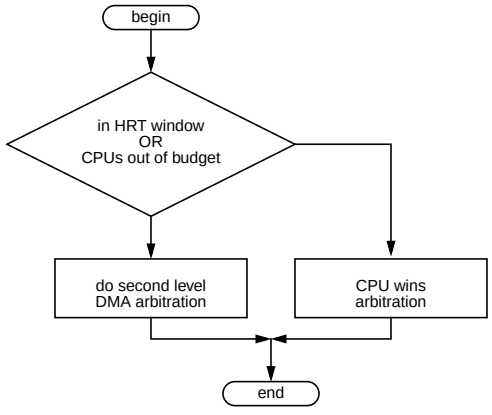


Figure 2: Arbitration in the DDR Controller

There are two mechanisms available in the arbitration: *windows* and *account* budgets.

Windows provide the basic means to allocate DDR bandwidth. A window is defined in terms of DDR controller clock cycles. Windows are defined for DMA traffic (HRT_WINDOW) and CPU traffic (CPU_WINDOW), and they alternate with each other in time. During an HRT_WINDOW, the DMA traffic is given priority by the arbitration scheme. During a CPU_WINDOW, the CPU traffic is given priority by the arbitration scheme.

As implied by the names CPU_WINDOW and HRT_WINDOW, windows have been introduced to divide DDR bandwidth between CPU traffic and Hard Real-Time (HRT) DMA traffic. Typically, in an SOC a third type of traffic is present as well: Soft Real_time (SRT) DMA traffic. This type of traffic usually has less hard real-time constraints than HRT DMA traffic i.e., the bandwidth requirements can be averaged over a much larger time period (several windows) than with HRT. However, it is still necessary to ensure that this type of traffic receives DDR memory bandwidth. To this end, a CPU account is introduced.

The CPU account limits (budgets) the memory bandwidth consumption by the CPU traffic to ensure that SRT DMA traffic receives enough memory bandwidth. The CPU account is defined by CPU_RATIO, CPU_LIMIT, CPU_CLIP and CPU_DECR.

The value CPU_RATIO controls how much bandwidth the CPU can get. The value CPU_LIMIT controls how many DDR bursts the CPU can take back-to-back before the CPU is out of budget. The value CPU_CLIP controls how much debt the CPU is allowed to build up. CPU_DECR is made programmable so that the accuracy of the accounting can be increased. This is especially needed when using dynamic ratios (see Section).

When the internal account exceeds CPU_LIMIT, DMA traffic is given higher priority than CPU traffic, independent of which window is active. The internal account is a saturated counter, that is, it will not wrap around on an underflow or overflow. For every DDR controller memory clock cycle, the internal counter is decremented by CPU_DECR. Whenever a CPU DDR burst is started, the internal counter is incremented by an amount equal to the amount of data transfer cycles, plus the value of CPU_RATIO, except when a CPU MTL transaction is ‘for free’.

A CPU MTL transaction is for free if it starts while the account value is above the CPU_CLIP value[1]. If a DDR burst is for free, then the account gets incremented by an amount equal to the amount of data transfer cycles, without the CPU_RATIO. The CPU_CLIP value should always be set equal or higher than CPU_LIMIT, otherwise CPU_LIMIT would never be reached.

By means of the accounting mechanism, the CPU bandwidth can be budgeted. In the CPU_WINDOW a CPU normally has priority over DMA. For every clock cycle the CPU account gets funded with CPU_DECR. For every CPU DDR burst, the costs of that burst, defined as CPU_RATIO plus data transfer cycles, are accounted for. When the CPU account runs out of budget (account value above CPU_LIMIT), then DMA will get priority over the CPU.

1. If pre-empting of the MTL transaction is not allowed, then all DDR bursts from one MTL transaction are treated the same. So if the first DDR burst is (not) for free then the other DDR bursts for the same MTL transactions will also be (not) for free. If pre-emption of the MTL transaction is allowed, then the ‘for free’ decision is made separately for each DDR burst.

If there is no DMA then the CPU can still get the BW which it has to pay for by allowing the CPU account to borrow from its future budget. If there is a longer time period where there is no DMA traffic, the CPU account could potentially build up a huge debt. As soon as DMA traffic restarts, the CPU could conceivably have an extended period of time where they have a lower priority than DMA (while paying off the debt). The CPU_CLIP value controls how much debt the CPU account is allowed to build up. After that value has been reached and there is still no DMA traffic the CPU will get the bandwidth for free. The number of data transfer cycles is accounted for to approximately (excluding overhead) get the same account value before and after the free transaction.

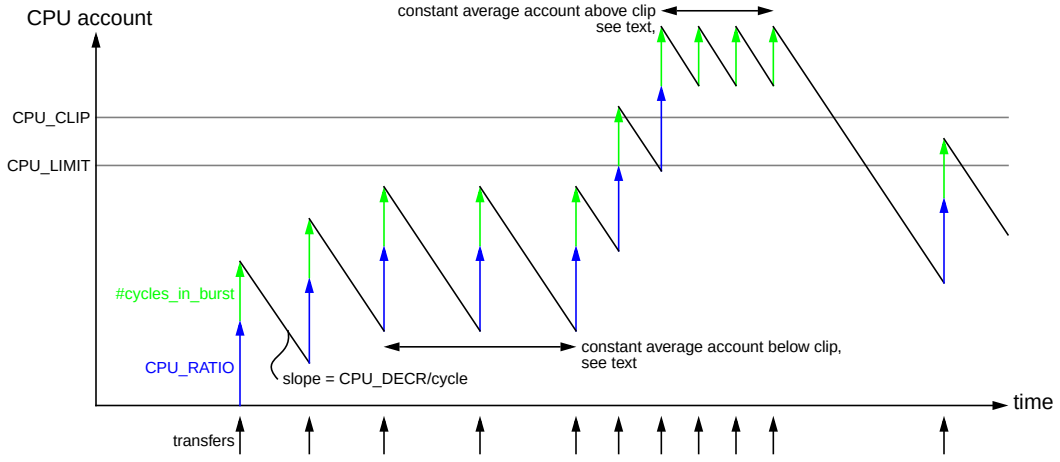


Figure 3: CPU account

In the time zone marked “constant average account below clip” in Figure 3, the transfer rate is such that the average value of the CPU account is constant. In this zone, we have the following equilibrium:

$$\text{CPU\_RATIO} + \#\text{cycles\_in\_burst} = \text{CPU\_DECR} \times \#\text{cycles\_between\_arbitration}$$

Where #cycles_in_burst is the nominal number of cycles it takes to complete a DDR burst, being half of the burst length, and #cycles_between_arbitration is the number of clock cycles between 2 successive CPU transfers win arbitration.

From this the CPU bandwidth (as percentage of maximum achievable) with constant average account is derived:

$$\text{CPU\_BW} = \frac{\#\text{cycles\_in\_burst}}{\#\text{cycles\_between\_arbitration}} = \frac{\text{CPU\_DECR}}{1 + \frac{\text{CPU\_RATIO}}{\#\text{cycles\_in\_burst}}}$$

In the time zone marked “constant average account above clip” in Figure 3, the transfer rate is such that the average value of the CPU account is constant. In this zone, we have the following equilibrium:

$$\#\text{cycles\_in\_burst} = \text{CPU\_DECR} \times \#\text{cycles\_between\_arbitration}$$

This equilibrium is only possible with CPU_DECR = 1 and all transfers back-to-back without any efficiency loss. In other words: when the CPU account exceeds the CPU_CLIP value, the account can only stay constant when the CPUs take 100% of the available bandwidth. This is unlikely to happen because the CPU account exceeds the CPU_LIMIT value, giving DMA a higher priority than the CPUs. Therefore, the CPU account will not stay above CPU_CLIP for long.

2.2.2 Second Level of Arbitration

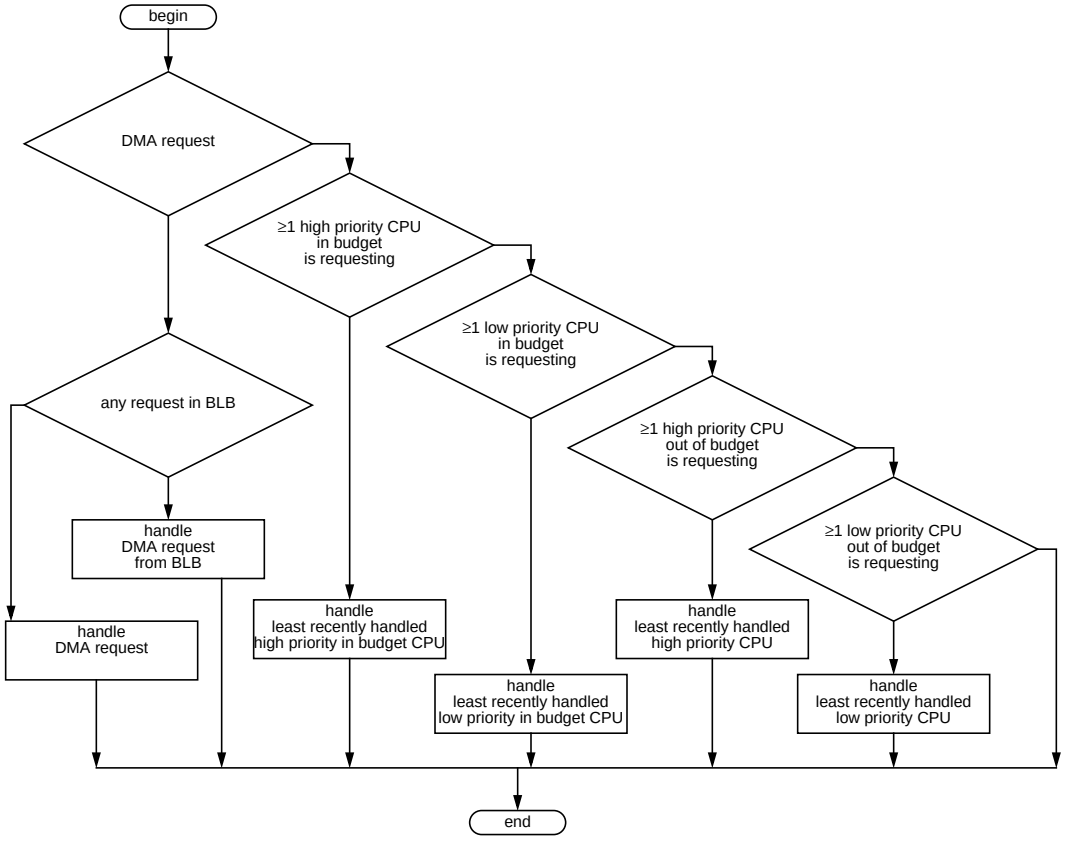


Figure 4: Arbitration when DMA has priority

2.2.3 Dynamic Ratios

The accounting mechanism described earlier is the static ratio variant. The problem with this approach is that the statically programmed CPU_RATIO that is used, per DDR burst, can not account for significantly different amounts of overhead by a DDR burst that can occur in real life. To fix that problem dynamic ratios have been introduced, which can be enabled through the ARB_CTL register.

Whenever a CPU DDR burst is started with dynamic ratios, the internal account is incremented by an amount equal to “the number of clock cycles spent on the previous CPU DDR burst” times the CPU_RATIO. This way the real overhead is measured and accounted for and therefore the accounting mechanism is much more accurate.

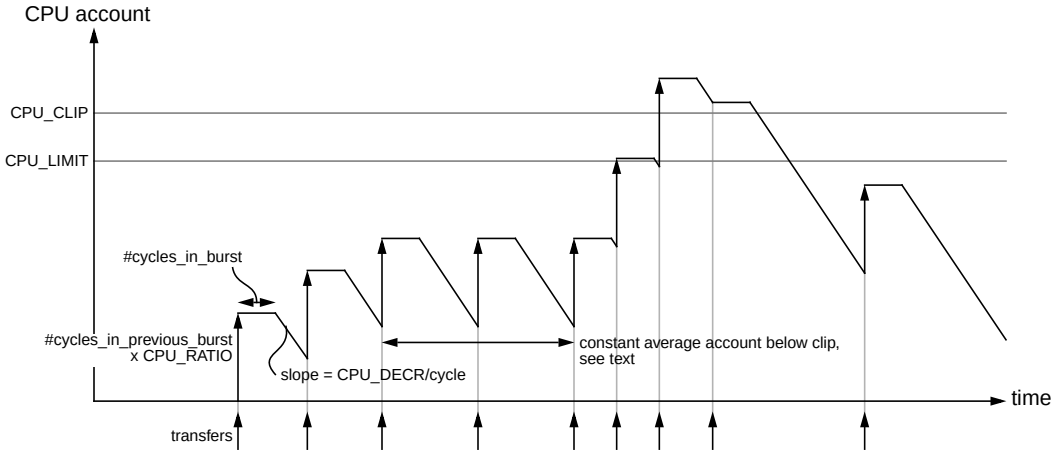


Figure 5: CPU account using dynamic ratios

In the time zone marked “constant average account below clip” in Figure 5, the transfer rate is such that the average value of the CPU account is constant. In this zone, we have the following equilibrium:

$$CPU\_RATIO \times \#cycles\_in\_previous\_burst = CPU\_DECR \times (\#cycles\_between\_arbitration - \#cycles\_in\_burst)$$

Where #cycles_in_burst is the actual number of cycles it takes to complete this DDR burst, and #cycles_in_previous_burst is the total number of clock cycles spent on executing CPU bursts since the previous CPU command won the arbitration.

From this the CPU bandwidth with constant average account is derived:

$$CPU\_BW = \frac{\#cycles\_in\_burst}{\#cycles\_between\_arbitration} = \frac{CPU\_DECR}{CPU\_RATIO \times \frac{\#cycles\_in\_previous\_burst}{\#cycles\_in\_burst} + CPU\_DECR}$$

Considering that on average, the number of cycles in a burst will be equal to the number of cycles in the previous burst, the average CPU bandwidth is:

$$average(CPU\_BW) = \frac{CPU\_DECR}{CPU\_RATIO + CPU\_DECR}$$

When the CPU account exceeds the CPU_CLIP value, the account can only stay constant when the CPUs take 100% of the available bandwidth. This is unlikely to happen because the CPU account exceeds the CPU_LIMIT value, giving DMA a higher priority than the CPUs. Therefore, the CPU account will not stay above CPU_CLIP for long.

With dynamic ratios enabled, the free bandwidth is also handled differently. When the bandwidth is for free i.e., the account is above the CPU_CLIP value, the internal account is not incremented at all. To ensure the internal account has the same value before and after the free bandwidth DDR burst, the account never decrements

whenever a clock cycle is spent on a CPU DDR burst, even if the burst is not for free. To account for this the CPU_RATIO should be set by the amount CPU_DECR lower as compared to the static ratios approach.

2.2.4 Pre-Emption

The arbitration scheme can be further fine tuned by specifying when arbitration is done. An MTL transaction is chopped up into one or more DDR bursts, as the arbiter operates on DDR bursts. Typically, the arbitration is done on an MTL transaction basis; i.e., once an MTL transaction has been selected by the arbitration scheme, all of its DDR bursts are processed before a new MTL transaction is selected. This approach tries to maximize bandwidth efficiency by exploiting locality assumed to be present within an MTL transaction. However, it might increase the expected latency of some MTL transactions.

When there is a CPU MTL transaction present while doing arbitration in an HRT window (and no DMA MTL transaction present). The CPU MTL transaction is selected by the arbiter. While the CPU transaction is being processed, a DMA MTL transaction becomes present (in the HRT window). The CPU MTL transaction is consuming HRT window bandwidth, while a DMA MTL transaction is waiting to be selected by the arbiter. From an overall bandwidth point of view, finishing the CPU MTL transaction to completion might be a good idea, but the programmed bandwidth partitioning is not fully applied. To address this issue, the concept of MTL transaction pre-emption is introduced.

MTL transaction pre-emption is programmable (via the MMIO register ARB_CTL) and can be used to interrupt an ongoing MTL transaction—before it is completed—to favor another MTL transaction. Pre-emption allows ongoing CPU MTL transactions to be interrupted by a DMA MTL transaction while in the HRT_WINDOW, and allows ongoing DMA MTL transactions to be interrupted by a CPU MTL transaction while in the CPU_WINDOW. Interruption of an MTL transaction of the same type will never happen. Any interruption will reduce the overall efficiency of the DDR Controller as it disallows exploiting locality assumed to be present within a MTL transaction.

The pre-emption field supports three different pre-emption settings. Table 1 describes the CPU pre-emption field.

Table 1: CPU Preemption Field

Preemption Field Value	Description
0	No preemption (once a CPU MTL command has started to enter the DDR arbitration buffer, it will go completely into the DDR arbitration buffer, uninterrupted by other (CPU or DMA) MTL commands).
1	Preempt a CPU MTL command as it starts to enter the DDR arbitration buffer while currently active in the DMA window. The CPU MTL command will only be interrupted by a DMA MTL command, not by another CPU MTL command. Default value
2	Undefined
3	Preempt a CPU MTL command that is currently active in the DMA window (independent of when it started to enter the DDR arbitration buffer).The CPU MTL command will only be interrupted by a DMA MTL command, not by another CPU MTL command.

2.2.5 Back Log Buffer (BLB)

The request for a DDR burst that wins the arbitration is always put in a FIFO queue. This FIFO is 5 levels deep to allow the DDR to look ahead and open and close pages in memory banks in order to increase DDR efficiency. Unfortunately this also means that a new high priority request that has immediately won the arbitration could possibly wait 5 full DDR bursts before it gets serviced. In a system in which almost all the available bandwidth is used (the FIFO is almost always full) this can significantly increase the latency.

Usually CPU traffic requires low latency and DMA traffic requires high bandwidth. In order to reduce latency for the CPUs, the back log buffer (BLB) has been implemented. When the BLB is enabled (through the ARB_CTL register), DMA DDR bursts that are in the FIFO can be temporarily moved to the BLB.

This is done under the following conditions:

- The FIFO entries hold a DMA DDR burst.
- No DDR burst of the same DMA MTL transaction has reached the top of the FIFO yet.
- the BLB is empty
- A CPU DDR burst request wins the arbitration.
- CPU traffic has higher priority than DMA traffic. (This is important in case the CPU wins arbitration, despite being lower priority than DMA, due only to the absence of DMA traffic.)

The BLB therefore allows the CPU transaction to overtake the DMA transaction already in the FIFO. Since the DDR Controller may have already opened/closed pages for the DMA DDR bursts, this feature will reduce the DDR efficiency.

As soon as DMA requests start winning the arbitration again, the DMA DDR bursts from the BLB get a higher priority than DMA requests from the MTL ports. Only when BLB is empty, DMA requests from the MTL ports can be serviced.

2.2.6 PMAN (Hub) versus DDR Controller Interaction

An additional factor that must be considered is the interaction of the Hub and the DDR Controller. The DDR Controller command FIFO (pipeline) is 5 entries, however the PMAN only allows 3 transactions to be outstanding. This means that the other two FIFO stages can (and will be) occupied by transactions from one of the CPUs. This can result in unexpected CPU bandwidth of up to 50%. This value is an extreme worst-case; a more realistic number (assuming some kind of video decoding) is around 15% of the gross DDR memory cycles.

Under the condition that the total required CPU budget is more than the maximum “leakage” of bandwidth it is possible to reduce the additional “leakage” (above and beyond budget) to zero by setting the value for CLIP = LIMIT + 2 * RATIO * <average transaction latency>.

The net result of this setting is that although “leakage” will still occur, it will be charged against the budget and compensated for immediately after occurrence.

It should also be noted that under some circumstances the PMAN will be granted a request even though there is a valid CPU request pending. This can only be detected within simulations and will be very difficult for a user to actually discern. This condition results from the particular optimizations that were performed on the logic and only delays a CPU by one DDR transaction. The overall bandwidth for the CPU is not affected.

2.3 Addressing

The DDR SDRAM Controller performs address mapping of MTL addresses onto DDR memory rank, bank, row and column addresses. The 32-bit MTL addresses, provided to the DDR controller, cover a 4-GB address range. Of these 32-bit addresses, the upper four bits are ignored by the DDR controller, reducing the addressable range to 256 MB. Note that the DDR controller only supports up to 256 MB of DDR memory (either implemented by a single rank or two ranks of size 128 MB).

2.3.1 Memory Region Mapping Scheme

For a 32-bit DDR interface, each column is 4 bytes wide. Therefore the 2 least significant bits of the MTL address are ignored.

For a 16-bit DDR interface (or a 32-bit DDR interface using the half width mode), each column is 2 bytes wide. Therefore the least significant bit of the MTL address is ignored.

The mapping is defined by the MMIO register DDR_DEF_BANK_SWITCH. 2^BANK_SWITCH defines the size of the interleaving. The addressing is then done as pictured in Figure 6.

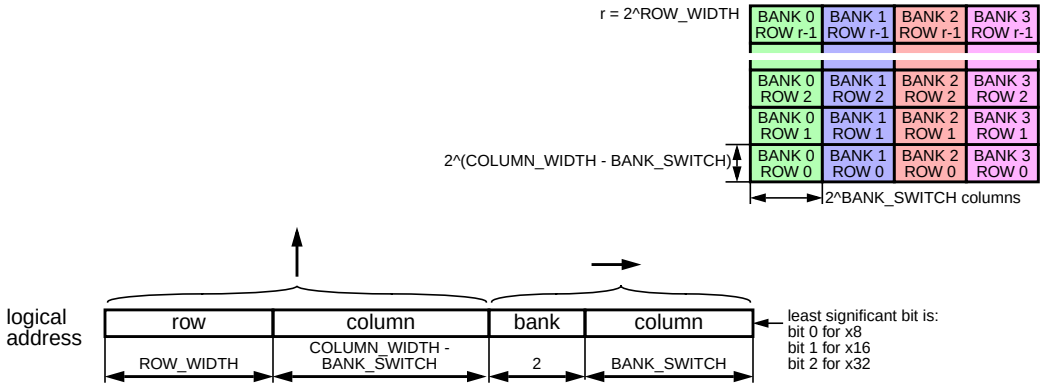


Figure 6: Address Mapping: Interleaved Mode

Changing the BANK_SWITCH value may improve/decrease performance. This is application specific. 32-byte and 1024-byte are the recommended operating modes.

This mapping can be illustrated in the following tables. In all of these examples a 32-bit DDR interface and a DDR burst length of 8 32-bit/4-byte elements (a full DDR burst transfers 8 * 4 bytes= 32 bytes).

Example 1: 32-Byte Interleaving

In 32-byte interleaving mode, the mapping scheme should change the DDR bank every other 2^3 = 8 columns. The BANK_SWITCH field is programmed to 3.

Table 2: 32-Byte Interleaving, 256 Columns

MTL Address Range	Row Address	Bank Address	Column Address
0x000:0000-0x000:001f	0x0000	0b00	0x0000-0x0007
0x000:0020-0x000:003f	0x0000	0b01	0x0000-0x0007
0x000:0040-0x000:005f	0x0000	0b10	0x0000-0x0007
0x000:0060-0x000:007f	0x0000	0b11	0x0000-0x0007
0x000:0080-0x000:009f	0x0000	0b00	0x0008-0x000f
0x000:0fe0-0x000:0fff	0x0000	0b11	0x00f8-0x00ff
0x000:1000-0x000:101f	0x0001	0b00	0x0000-0x0007
0x000:1020-0x000:103f	0x0001	0b01	0x0000-0x0007
0x000:1fe0-0x000:1fff	0x0001	0b11	0x00f8-0x00ff
0x000:2000-0x000:201f	0x0002	0b00	0x0000-0x0007
0x000:2020-0x000:203f	0x0002	0b01	0x0000-0x0007

Table 3: 32-Byte Interleaving, 512 Columns

MTL Address Range	Row Address	Bank Address	Column Address
0x000:0000-0x000:001f	0x0000	0b00	0x0000-0x0007
0x000:0020-0x000:003f	0x0000	0b01	0x0000-0x0007
0x000:0040-0x000:005f	0x0000	0b10	0x0000-0x0007
0x000:0060-0x000:007f	0x0000	0b11	0x0000-0x0007
0x000:0080-0x000:009f	0x0000	0b00	0x0008-0x000f
0x000:0feo-0x000:0fff	0x0000	0b11	0x00f8-0x00ff
0x000:1000-0x000:101f	0x0000	0b00	0x0100-0x0107
0x000:1020-0x000:103f	0x0000	0b01	0x0100-0x0107
0x000:1fe0-0x000:1fff	0x0000	0b11	0x01f8-0x01ff
0x000:2000-0x000:201f	0x0001	0b00	0x0000-0x0007
0x000:2020-0x000:203f	0x0001	0b01	0x0000-0x0007

Example 2: 1024-Byte Interleaving

In 1024-byte interleaving mode, the mapping scheme should change the DDR bank every other 2^8 = 256 columns. The BANK_SWITCH field is programmed to 8.

Table 4: Mapping scheme: 1024-Byte Interleaving, 256 Columns

MTL Address Range	Row Address	Bank Address	Column Address
0x000:0000-0x000:001f	0x0000	0b00	0x0000-0x0007
0x000:0020-0x000:003f	0x0000	0b00	0x0008-0x000f
0x000:0040-0x000:005f	0x0000	0b00	0x0010-0x0017
0x000:0060-0x000:007f	0x0000	0b00	0x0018-0x001f
0x000:0400-0x000:041f	0x0000	0b01	0x0000-0x0007

Table 4: Mapping scheme: 1024-Byte Interleaving, 256 Columns *...Continued*

MTL Address Range	Row Address	Bank Address	Column Address
0x000:0800-0x000:081f	0x0000	0b10	0x0000-0x0007
0x000:0c00-0x000:0c1f	0x0000	0b11	0x0000-0x0007
0x000:1000-0x000:101f	0x0001	0b00	0x0000-0x0007
0x000:1400-0x000:141f	0x0001	0b01	0x0000-0x0007
0x000:2000-0x000:201f	0x0002	0b00	0x0000-0x0007
0x000:2400-0x000:241f	0x0002	0b01	0x0000-0x0007

Table 5: 1024-Byte Interleaving, 512 Columns

MTL Address Range	Row Address	Bank Address	Column Address
0x000:0000-0x000:001f	0x0000	0b00	0x0000-0x0007
0x000:0020-0x000:003f	0x0000	0b00	0x0008-0x000f
0x000:0040-0x000:005f	0x0000	0b00	0x0010-0x0017
0x000:0060-0x000:007f	0x0000	0b00	0x0018-0x001f
0x000:0400-0x000:041f	0x0000	0b00	0x0100-0x0107
0x000:0800-0x000:081f	0x0000	0b01	0x0000-0x0007
0x000:0c00-0x000:0c1f	0x0000	0b01	0x0100-0x0107
0x000:1000-0x000:101f	0x0000	0b10	0x0000-0x0007
0x000:1400-0x000:141f	0x0000	0b10	0x0100-0x0107
0x000:2000-0x000:201f	0x0001	0b00	0x0000-0x0007
0x000:2400-0x000:241f	0x0001	0b00	0x0100-0x0107

2.3.2 DDR Memory Rank Locations

The DDR SDRAM Controller supports two DDR memory ranks. The location of these two memory ranks in the MTL address space is defined by means of MMIO registers RANK0_ADDR_LO, RANK0_ADDR_HI, and RANK1_ADDR_HI. Rank 1 starts where rank0 leaves off in the MTL address space; i.e. the ranks are successive. Programming of these MMIO registers should be consistent with the size of the memories. An attempt to address an address outside of the two DDR memory ranks will result in an error, which is registered by MMIO registers. Erroneous addressing will still result in DDR read or write operations being performed.

Rank 1 starts where rank0 leaves off in the MTL address space i.e., the ranks are successive. Programming these MMIO registers should be consistent with the memory size. An attempt to address an address outside of the two DDR memory ranks will result in an error, which is registered by MMIO registers. Erroneous addressing will still result in DDR read or write operations being performed.

The start addresses of the ranks should be a multiple of the respective rank sizes.

The following examples will illustrate rank addressing and error detection situations.

Some Examples

If RANK0_ADDR_LO is set to 0x0800:0000, RANK0_ADDR_HI to 0x0bff:ffff, and RANK1_ADDR_HI to 0x0dff:ffff. This implies a 64-MB rank 0 starting at address 0x0800:0000, and a 32 MB rank 1 starting at address 0x0c00:0000.

If the address 0x0900:0000 has to be mapped, the upper 4 bits of the 32-bit address are ignored. The address is located at byte offset 0x0100:0000 in rank 0.

If the address 0x3900:0000 has to be mapped, the upper 4 bits of the 32-bit address are ignored. The address is located at byte offset 0x0100:0000 in rank 0.

If the address 0x0c80:0000 has to be mapped, the upper 4 bits of the 32-bit address are ignored. The address is located at byte offset 0x0080:0000 in rank 1.

If the address 0x3c80:0000 has to be mapped, the upper 4 bits of the 32-bit address are ignored. The address is located at byte offset 0x0080:0000 in rank 1.

If the address 0x0500:0000 has to be mapped, the upper 4 bits of the 32-bit address are ignored. The address is not located within any of the two ranks, therefor an error flag is set in MMIO register ERR_VALID to indicate this. Furthermore, the DDR SDRAM Controller output signal “ip_2031_ddr_addr_err” is made ‘1’. The 28 lower bits of the address indicate a reference below rank 0. Therefor, this address is aliased to rank 0. The aliased address is located at byte offset 0x0100:0000 in rank 0.

If the address 0x0e80:0000 has to be mapped, the upper 4 bits of the 32-bit address are ignored. The address is not located within any of the two ranks, therefore an error flag is set in MMIO register ERR_VALID to indicate this. Furthermore, the IP_2031 output signal “ip_2031_ddr_addr_err” is made ‘1’. The 28 lower bits of the address indicate a reference above rank 1. Therefore, this address is aliased to rank 1 and located at byte offset 0x0080:0000 in rank 1.

2.4 Clock Programming

The DDR clock is managed by the Clock module. Both clk_mem and clk_dtl_mmio must be on.

2.5 Power Management

In order to reduce power consumption, the DDR SDRAM Controller can be turned into halt mode. During halt mode, the clock inputs to the DDR controller may be turned off to reduce dynamic power consumption. When the clock inputs to the DDR controller are turned off, it will be non-functional. The DDR controller assumes that during halt mode the clock inputs to the DLLs may be turned off as well. As a result, the DDR controller power up sequence includes resetting the DLLs.

Note that when the clock inputs to the DDR controller are turned off, no access to the DDR controller MMIO registers is possible.

Putting the DDR SDRAM Controller in halt mode, and keeping the clock inputs to the DDR controller turned on, allows for safe programming of the MMIO registers using the DTL MMIO interface. When MMIO registers DDR_MR and DDR_EMR are re-programmed, a start action has to be performed (after the DDR controller is unhalted), for the new DDR values to take effect.

2.5.1 Halting and Unhalting

There are three different ways in which halting can be achieved:

1. By means of writing the halt register-field of a software programmable MMIO register.
2. Telling the DDR SDRAM Controller to go into halt mode automatically after a certain period of inactivity.

In Halt mode, the DDR devices are sent into self-refresh mode.

2.5.2 MMIO Directed Halt

MMIO register IP_2031_CTL, field HALT can be written with a ‘1’ to indicate a request for halting. Write a ‘0’ to this field to indicate a request for taking the DDR controller out of halt mode. Direct Halt directives are meant to be used when the PNX15xx Series system is sent to sleep and therefore no request is supposed to happen on the MTL port. Direct Halt un-halt command is also used/required when changing the clock frequency of the DDR interface.

Software must wait for a time period equal to a minimum of 256 DDR SDRAM Controller clocks before clocks are changed or turned off.

2.5.3 Auto Halt

The DDR SDRAM Controller can turn itself in halt mode when it has observed a certain period of inactivity. By programming the MMIO registers HALT_COUNT and CTL a period can be defined and automatic halting can be activated. The DDR controller will automatically unhalt when a new MTL memory request is presented to one of its input ports. To ensure the IP_2031 can detect these MTL memory requests, the DDR controller clock inputs need to be turned on during auto halt (or at least have

to be turned on before the MTL memory request is presented to the DDR controller). This mode adds extra latency for requests to be served and should therefore be used adequately.

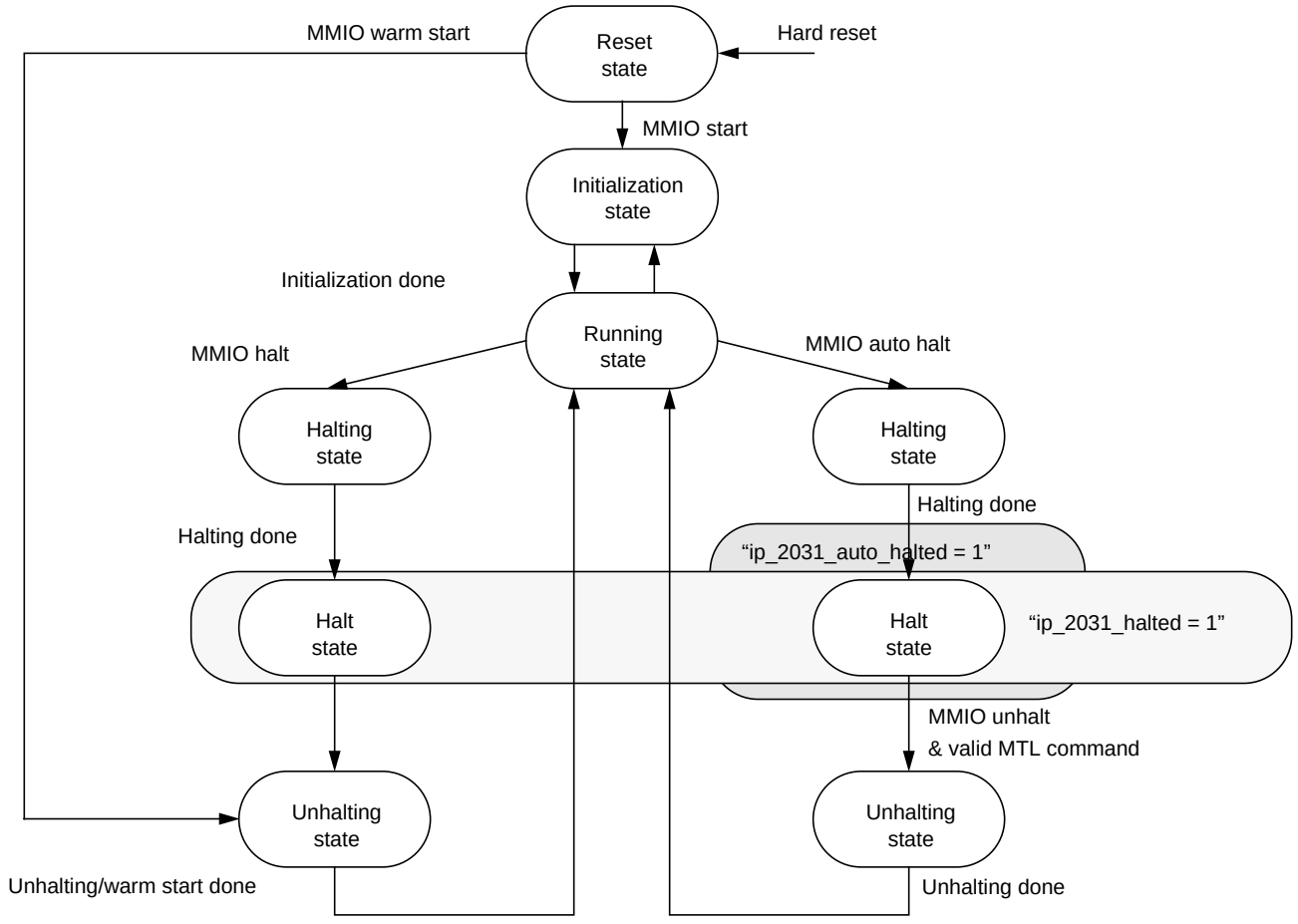


Figure 7: DDR SDRAM Controller Start and Halt State Machine

2.5.4 Observing Halt Mode

When the DDR SDRAM Controller entered halt mode due to an auto halt, it will only unhalt when a MTL memory request is presented to one of its input ports. To ensure the DDR controller can detect these MTL memory requests, the DDR controller clock inputs need to be turned on during auto-halt (or at least turned on before the MTL memory request is presented to the DDR controller). Therefore it is advised not to turn off the clock to the DDR controller when "ip_2031_auto_halted" is '1' since this is a dynamic mode controlled by the DDR controller not software.

The DDR SDRAM Controller is in halt mode if the HALT_STATUS bit in the MMIO register IP_2031_CTL is set to '1'. The clock, clk_mem and clk_mmio must be turned on to execute the MMIO read.

2.5.5 Sequence of Actions

To enter halt mode, the DDR SDRAM Controller performs the following sequence of actions:

1. Precharge all banks (of all ranks)
2. Apply a NOP command
3. Enter self refresh mode, with CKE low, deactivate internal DLL

To leave halt mode, the DDR controller performs the following action:

- 256 DDR SDRAM Controller memory cycles with CKE high, NOP commands, to activate DLL.

3. Application Notes

3.1 Memory Configurations

The DDR SDRAM Controller supports a wide range of DDR SDRAM memory configurations. Some examples of memory configurations that are supported for an external data bus of 32 bits are shown in Figure 8. On the left side a single physical bank of DDR devices is connected to the DDR controller. Throughout this document the term rank will be used for a physical bank in order to prevent any confusion with the logical banks inside the DDR devices. On the right hand side of Figure 8 two ranks of DDR devices are connected to the DDR controller. In single rank configurations, there is no need to drive the chip select inputs on the DDR devices from the DDR controller. In a multi-rank configuration, each rank will receive its own chip select signal from the DDR controller. The DDR controller offers a 1 to 1 match with the pin names of the DDR memory devices.

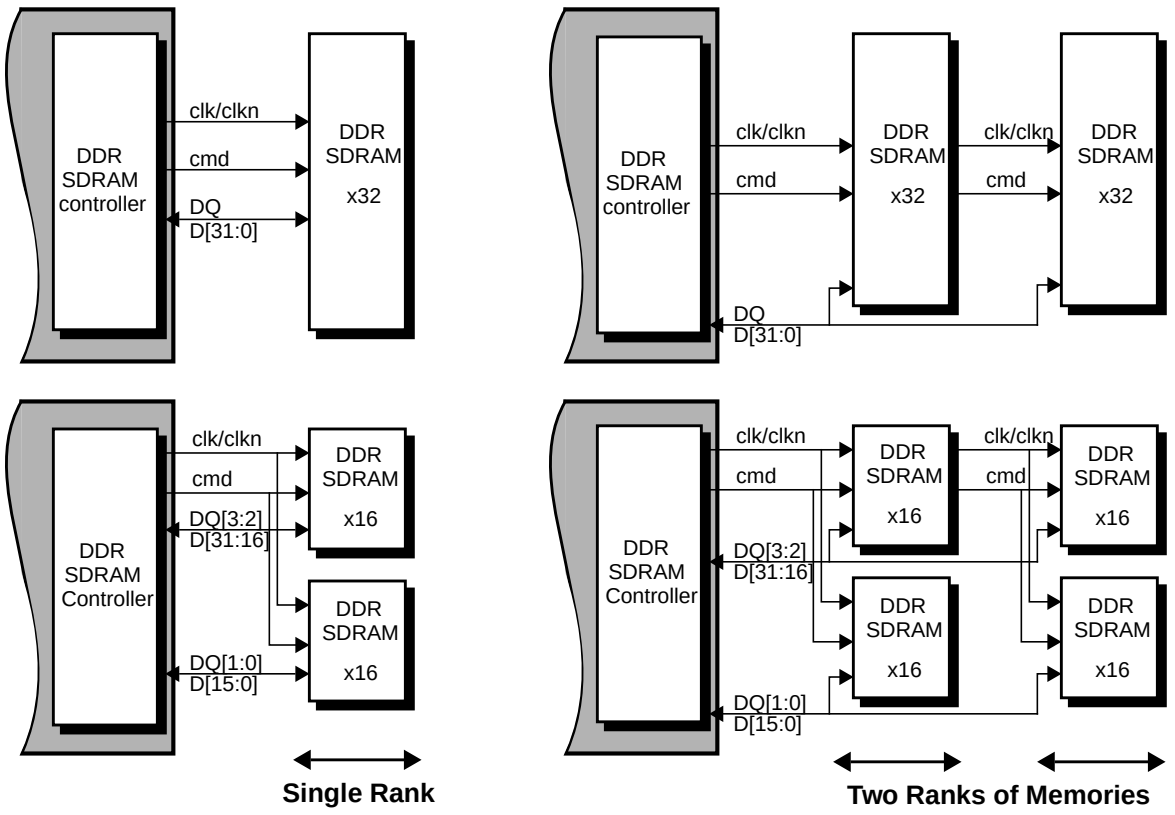


Figure 8: Examples of Supported Memory Configurations

3.2 Error Signaling

The MMIO port does not support error signaling. Reads from invalid addresses return the value “0”, writes to invalid addresses are ignored. The errors are not reported at system level.

Changing MMIO registers of an initiated DDR SDRAM Controller may cause incorrect behavior. Forcing the DDR controller into halt mode, programming MMIO registers while in halt mode, then unhalting the DDR controller when the MMIO registers have been programmed is the suggested series of actions to take.

3.3 Latency

The DDR SDRAM Controller uses two pipeline stages to calculate the command(s) that will be issued to the DDR memories after a MTL command is accepted by the DDR controller.

We will describe the latency of a MTL read command. Assume we have a MTL read command on one of the MTL ports in cycle 0 which is accepted by the DDR controller. In cycle 1, the DDR controller will determine the first DDR burst for the MTL read command. In cycle 2, the DDR SDRAM Controller will determine the DDR commands that need to be sent out on the DDR interface (we assume we do not have

any other MTL transactions pending in the DDR controller). When the read was to an already activated row, in cycle 3 a DDR read command will appear on the DDR interface. Given a CAS latency of n cycles (typically the CAS latency is 2, 2.5, 3, 3.5, or 4 cycles), the first read data element will be presented by the memory device in cycle 3 + n. To allow for safe clock domain transfer (from the “dqs” clock domain to the “clk_mtl” clock domain) and to combine two DDR read data elements into a single MTL read data element, the DDR controller takes two extra cycles before presenting the read data on the MTL interface in cycle 3 + n + 2. As a result, the lowest latency from MTL read command accept to first MTL read data element valid on the MTL interface is 3 +
n + 2 cycles. In case of pending MTL transactions in the DDR controller, and in case of required DDR precharge and activate commands, the latency will increase.

3.4 Data Coherency

Memory requests at an MTL port of the DDR controller are processed and executed in the order that they are received. The DDR controller does not re-order the commands on a MTL interface. From this point of view data coherency between memory bus agents that connect to a single port on the DDR controller is guaranteed.

However, the memory requests that are made to different MTL interfaces on the DDR SDRAM Controller in general will not be serviced in the order that they appeared. The order in which these requests are serviced depends on the state of the DDR SDRAM device(s) and how the internal arbiter is programmed. The user needs to take care of data coherency between memory agents that connect to different MTL ports of the DDR controller.

3.5 Programming the Internal Arbiter

The window is defined by a 16-bit value that represents the size of the window in terms of IP_2031 memory clock cycles. By choosing a certain ratio between the HRT_WINDOW and the CPU_WINDOW, the available DDR bandwidth can be divided between DMA traffic on MTL port 0, and CPU traffic on MTL port 1. The window size may affect the latency of traffic. By choosing a large value for HRT_WINDOW, the CPU traffic may get a large latency. However, for small window sizes the DDR controller may not be able to divide the available DDR bandwidth between DMA traffic and CPU traffic as expected by the programmed window sizes. Window sizes between values 20 and 100 are advised to ensure acceptable traffic latencies and proper dividing of available DDR bandwidth. E.g. to achieve a DDR bandwidth division of 25% CPU traffic and 75% DMA traffic, a CPU_WINDOW of 25, and a HRT_WINDOW of 75 could be programmed. Using a CPU_WINDOW of 100, and a HRT_WINDOW of 300 would probably be able to achieve a more accurate division of the bandwidth, but may result in unacceptable traffic latencies.

To program the parameters for the internal arbiter, follow the steps below:

1) Determine the total available bandwidth (tot_bw), based on the board DDR setup (frequency and bus width). **Note a Mega in Hz is not a M in Bytes**! Use an average DDR efficiency of 73%, determine required peak hard real time bandwidth (hrt_pk), average hard real time bandwidth (hrt_avg), average soft real time (srt) and average total CPU bandwidth (CPU).

2) Select the minimum (min) window size allowed; 20 or 40 are good examples. Higher minimum values increase the latency, but can also slightly increase the DDR efficiency (because more requests of one type (DMA or CPU) are handled in sequence). The value 20 is based on a 128-byte transfer that takes 16 data transfer cycles on a 32-bit DDR interface. Assuming an average DDR efficiency of 80% a total of 20 cycles will be needed to start and finish this transfer.

3) Use the minimum window value for the port with the least traffic (DMA or CPU) and calculate the other window according to the following formula:

if (hrt_pk < cpu)

hrt_window = min;

cpu_window = ((tot_bw / hrt_pk) -1)* hrt_window;

else

cpu_window = min;

hrt_window = (hrt_pk / (tot_bw - hrt_pk)) * cpu_window;

endif

4) If the selected minimum value is low and the calculated window size is much bigger than the minimum value, setting ‘always’ pre-empt (0x3) on the high bandwidth traffic (and maybe even ‘never’ pre-empt (0x0) on the low bandwidth traffic) will be needed to make sure the low bandwidth modules get enough traffic.

5) The next parameter to calculate is cpu_ratio. To do this, first account for the fact that normally not all available bandwidth will be used. It is a good idea to distribute the headroom proportionally between the CPU and the soft real time DMA, as shown in the following formulas:

srt2 = (tot_bw - hrt_avg) * srt / (srt + cpu);

cpu2 = (tot_bw - hrt_avg) * cpu / (srt + cpu);

6) The cpu_ratio and cpu_limit make sure that when the CPU is asking too much bandwidth that it gets blocked out in the CPU window and soft real time DMA is allowed access instead. The cpu_ratio determines how many cycles the CPU gets blocked (versus the DMA) for each cycle the DDR spends on CPU data transfers. The cpu_ratio is added to the account for each DDR burst, a DDR burst length is 4 cycles. So the formula for cpu_ratio is:

cpu_ratio = 4 * (hrt_avg + srt2) / cpu2;

7) Finally the cpu_limit needs to be estimated, as it basically determines how many consecutive CPU transfers are allowed to finish before the CPU gets blocked out. A typical value is one data cache line replacement (copy back and fetch) and one instruction fetch. Assuming a data and instruction cache line size of 64 bytes, that is a total of 3*64 = 192 bytes.

For each DDR burst (4 clock cycles) the DDR transfers (for a dual data rate, 32-bit DDR interface) 4*2*4=32 bytes, so 192/32 = 6 bursts are needed. The cpu_limit needs to be at least 6*cpu_ratio.

In general, setting the cpu_limit too low will block the CPU too frequently causing a too high latency (execution time). Setting the cpu_limit too high can completely block the soft real time DMA for a long time when the hard real time DMA and CPU bandwidth are peaking. But perhaps the long latency that causes the soft real time may not be a problem.

3.6 The DDR Controller and the DDR Memory Devices

The DDR SDRAM Controller is compatible with most of the DDR SDRAM vendors. This is achieved when the correct timing parameters are programmed in the MMIO registers holdings the timing parameters has presented in the two following sections.

4. Timing Diagrams and Tables

This section shows how programmable timing parameters direct the operation of the DDR SDRAM Controller. It is not the intention of this section to give a complete overview of all DDR interface signaling. Only the main ones are described.

Table 6 presents the values that are used for the different timing parameters in the timing diagrams.

Table 6: DDR Timing Parameters

Parameter	Symbol	Value (Clock Cycles)
CAS latency	t_{CAS}	2.5
Minimum time between two active commands to different banks	t_{RRD}	3
Minimum time between two active commands to same bank	t_{RC}	8
Minimum time between auto refresh and active command	t_{RFC}	8
Minimum time after last data write and precharge to same bank	t_{WR}	1
Minimum time between active and precharge command	t_{RAS}	8
Minimum time between precharge and active command	t_{RP}	4
Minimum time between active and read command	t_{RCD_RD}	4
Minimum time between active and write command	t_{RCD_WR}	2

Throughout all timing diagrams a DDR burst size of eight data elements is used.

In the timing diagrams, symbols are used to indicate the DDR commands that are issued by the DDR controller. An overview of these commands and their symbol convention are shown in Table 7.

Table 7: DDR Commands

DDR Commands	Symbol
Any DDR command	Any
Activate command	Act
Precharge command	Pre
Read command	Read
Write command	Write
Auto refresh command	A. rf.

4.0.1 Tcas Timing Parameter

Figure 9 shows two consecutive read bursts with a Tcas delay of 2.5 cycles.

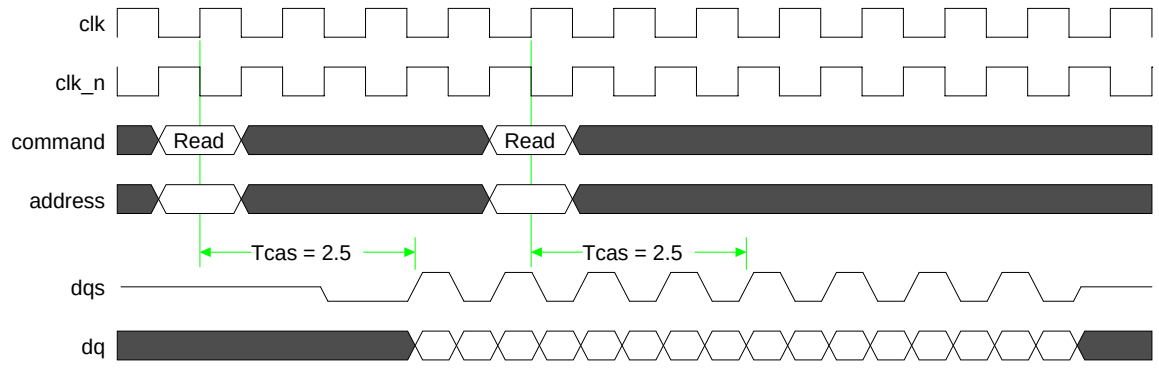


Figure 9: Tcas Timing Parameter

4.1 Trrd and Trc Timing Parameters

Figure 10 shows three active commands with a Trrd delay of 3 cycles and a Trc delay of 8 cycles. The first two activated commands are to different banks, the third activated command is to the same bank as the second command.

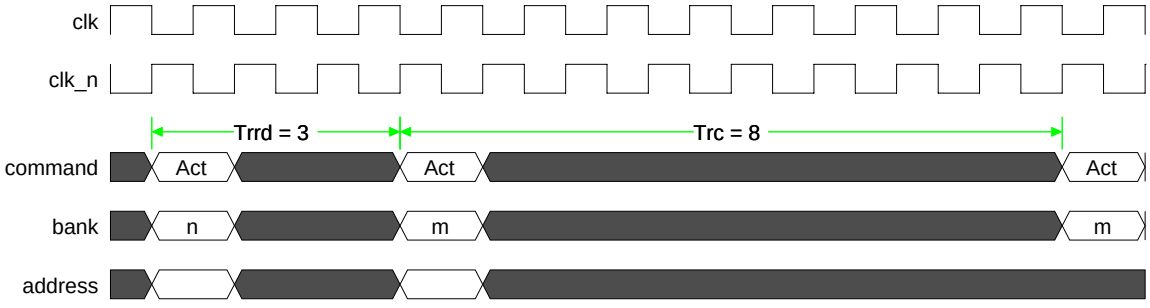


Figure 10: Trrd and Trc Timing Parameters

4.2 Trfc Timing Parameter

Figure 11 shows a Tcas of 8 cycles.

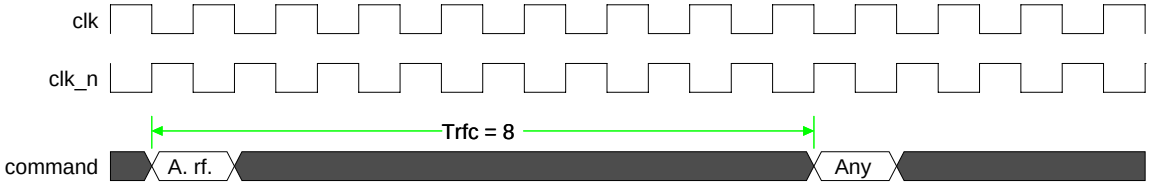


Figure 11: Trfc Timing Parameter

4.3 Twr Timing Parameter

Figure 12 shows a Twr of 1 cycle.

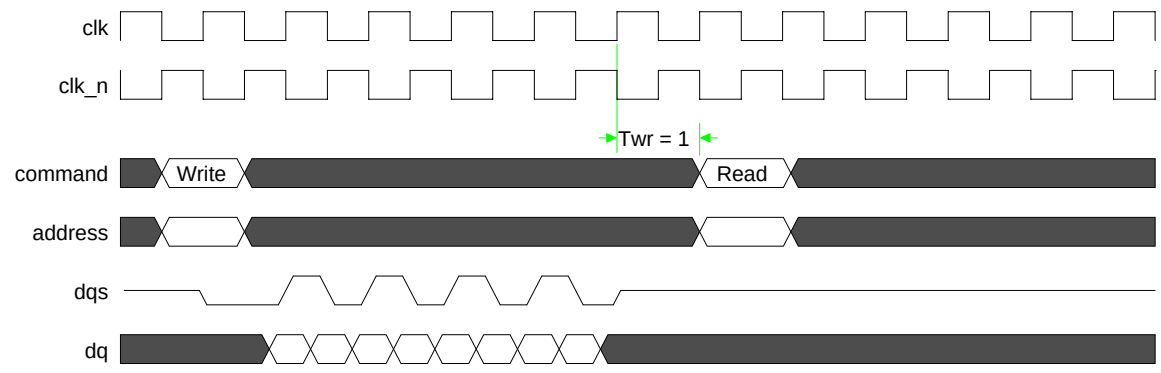


Figure 12: Twr Timing Parameter

4.4 Tras Timing Parameter

Figure 13 shows a Tras of 8 cycles.

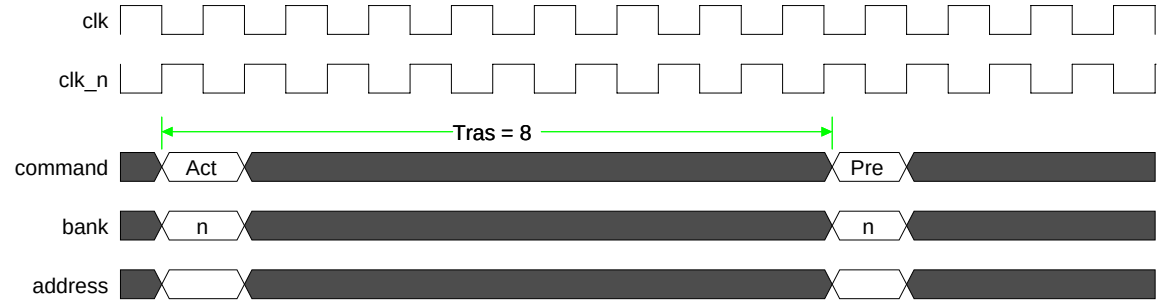


Figure 13: Tras Timing Parameter

4.5 Trp Timing Parameter

Figure 14 shows a Trp of 4 cycles.

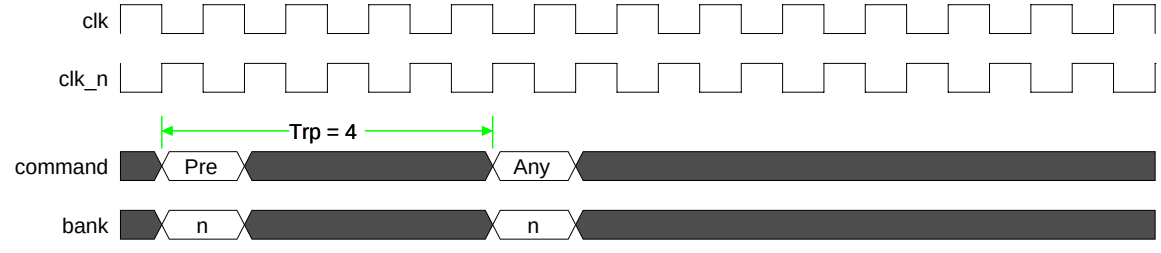


Figure 14: Trp Timing Parameter

4.6 Trcd_rd Timing Parameter

Figure 15 shows a Trcd_rd of 4 cycles.

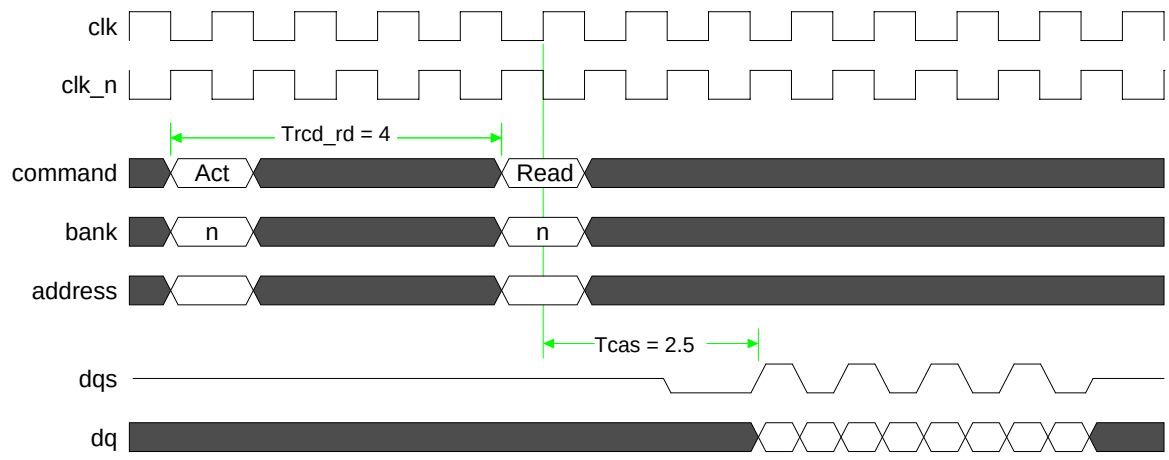


Figure 15: Trcd_rd Timing Parameter

4.7 Trcd_wr Timing Parameter

Figure 16 shows a Trcd_wr of 2 cycles.3.2 Asynchronous Reset Synchronization

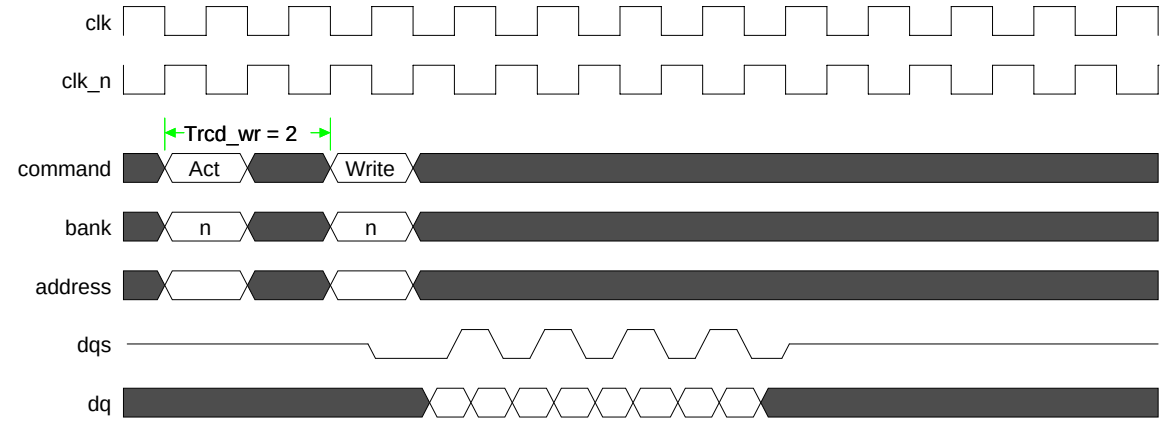


Figure 16: Trcd_wr Timing Parameter

5. Register Descriptions

The DDR SDRAM Controller contains a number of MMIO registers that are used to:

- set generic control and read generic status information
- set dimensions of the DDR memories
- set timing characteristics of the DDR memories
- set arbitration parameters

- observe the performance of the DDR SDRAM Controller
- observe specifics about errors

Turning the DDR controller into halt mode, programming MMIO registers while in halt mode, and un-halting the DDR controller when the MMIO registers have been programmed, is the suggested series of actions to change MMIO register values of a started DDR controller.

5.1 Register Summary

The offsets reported in the following table are absolute offset with respect to the MMIO_BASE value.

Table 8: Register Summary

Offset	Symbol	Description
0x06 5000	IP_2031_CTL	DDR GENERAL CONTROL
0x06 5004	DDR_DEF_BANK_SWITCH	DDR BANK SWITCH ADDRESSING
0x06 5008	AUTO_HALT_LIMIT	DDR AUTO HALT LIMIT
0x06 5010	RANK0_ADDR_LO	DDR RANK0 ADDRESS LOW LIMIT
0x06 5014	RANK0_ADDR_HI	DDR RANK0 ADDRESS HIGH LIMIT
0x06 5018	RANK1_ADDR_HI	DDR RANK1 ADDRESS HIGH LIMIT
0x06 5080	DDR_MR	DDR MODE REGISTER
0x06 5084	DDR_EMR	DDR EXTEND MODE REGISTER
0x06 5088	DDR_PRECHARGE_BIT	DDR PRECHARGE BIT FIELD
0x06 50C0	RANK0_ROW_WIDTH	DDR RANK0 ROW BIT WIDTH
0x06 50C4	RANK0_COLUMN_WIDTH	DDR RANK0 COLUMN BIT WIDTH
0x06 50D0	RANK1_ROW_WIDTH	DDR RANK1 ROW BIT WIDTH
0x06 50D4	RANK1_COLUMN_WIDTH	DDR RANK1 COLUMN BIT WIDTH
0x06 5100	DDR_TRCD	DDR ACTIVE to READ or WRITE DELAY
0x06 5104	DDR_TRC	DDR ACTIVE to ACTIVE/AUTO REFRESH DELAY
0x06 5108	DDR_TWTR	DDR INTERNAL WRITE to READ COMMAND DELAY
0x06 510C	DDR_TWR	DDR WRITE RECOVERY TIME
0x06 5110	DDR_TRP	DDR PRECHARGE COMMAND PERIOD
0x06 5114	DDR_TRAS	DDR ACTIVE to PRECHARGE COMMAND PERIOD
0x06 511C	DDR_TRRD	DDR ACTIVE BANK A to ACTIVE BANK B COMMAND
0x06 5120	DDR_TRFC	DDR AUTO REFRESH COMMAND PERIOD
0x06 5124	DDR_TMRD	DDR LOAD MODE REGISTER COMMAND CYCLE
0x06 5128	DDR_TCAS	DDR CAS READ LATENCY
0x06 512C	DDR_RF_PERIOD	DDR REFRESH PERIOD
0x06 5180	ARB_CTL	DDR ARBITER CONTROL
0x06 5184	ARB_HRT_WINDOW	DDR ARBITER HARD REAL TIME WINDOW
0x06 5188	ARB_CPU_WINDOW	DDR ARBITER CPU WINDOW

Table 8: Register Summary

Offset	Symbol	Description
0x06 51C0	ARB_CPU_LIMIT	DDR ARBITER CPU LIMIT
0x06 51C4	ARB_CPU_RATIO	DDR ARBITER CPU RATIO
0x06 5200	PF_MTL0_RD_VALID	DDR PERFORMANCE MTL0 READ VALID
0x06 5204	PF_MTL0_WR_ACCEPT	DDR PERFORMANCE MTL0 WRITE ACCEPT
0x06 5208	PF_MTL1_RD_VALID	DDR PERFORMANCE MTL1 READ VALID
0x06 520C	PF_MTL1_WR_ACCEPT	DDR PERFORMANCE MTL1 WRITE ACCEPT
0x06 5240	PF_IDLE	DDR PERFORMANCE IDLE
0x06 5280	ERR_VALID	DDR ERROR VALID
0x06 5284	ERR_MTL_PORT	DDR ERROR MTL PORT
0x06 5288	ERR_MTL_CMD_ADDR	DDR ERROR MTL COMMAND ADDRESS
0x06 528C	ERR_MTL_CMD_READ	DDR ERROR MTL COMMAND READ
0x06 5290	ERR_MTL_CMD_ID	DDR ERROR MTL COMMAND ID
0x06 0FFC	MODULE_ID	DDR MODULE ID

5.2 Register Table

Table 9: Register Description

Bit	Symbol	Access	Value	Description
Generic Control and Status				
Offset 0x06 5000	***IP_2031_CTL***			
31	HALT_STATUS	R	0	‘0’: Not in halt mode. ‘1’: Halt mode.
30	AUTO_HALT_STATUS	R	0	‘0’: Not in halt mode. ‘1’: Halt mode.
29:16	Unused	R	-	These bits should be ignored when read and written as 0s.
15	HALT	R/W	0	‘0’: Unhalt when in halt mode. ‘1’: Halt when not in halt mode.
14	AUTO_HALT	R/W	0	‘0’: No automatic halt. ‘1’: Allow automatic halt.
13	WARM_START	R/W	0	‘1’: Perform a warm start of the controller. This will behave as a unhalt operation. This can be used to start the DDR controller without effecting the state of the external DDR memory.
12:5	Unused	R	-	These bits should be ignored when read, and written as 0s.
4	DIS_WRITE_INT	R	1	‘1’: DDR write burst cannot be interrupted by following read command.
3	DDR_DQS_PER_BYTE	R/W	0	‘0’: A single “dqs” signal is provided for all “dq” byte lane. Output pin MM_DQS[0] must be used for all byte lanes. ‘1’: A separate “dqs” signal is provided for every “dq” byte lane. These strobe signals are used to register “dq” byte lanes.

Table 9: Register Description

Bit	Symbol	Access	Value	Description
2	DDR_HALVE_WIDTH	R/W	0	'0': The complete "dq" bus of the DDR interface is used. '1': Only the lower halve of the data bus of the DDR interface is used. Only DDR data bits "MM_DATA[15:0]" are in use.
1	SPEC_AUTO_PR	R/W	0	'0': Speculative auto precharge is off. '1': Speculative auto precharge is on.
0	START	R/W	0	'1': Start DDR controller. When started, controller will return the start bit to '0'.
Offset 0x06 5004	***DDR_DEF_BANK_SWITCH***			
Note: Addressing modes 2048_MODE, 1024_MODE, and the interleaving mode defined by the BANK_SWITCH field are mutually exclusive. Setting 2048_MODE to '1' sets the IP_2031 into 2048 byte stride mode, and makes the values of 1024_MODE and BANK_SWITCH "don't cares" for the IP_2031. When 2048_MODE is '0' and 1024_MODE is '1', the IP_2031 is set into 1024 byte stride mode, which makes the value of BANK_SWITCH a "don't care" for the IP_2031.				
31:4	Unused	R	-	These bits should be ignored when read and written as 0s.
3:0	BANK_SWITCH	R/W	3	Switch banks every 2^BANK_SWITCH columns (each column has a width of 4 bytes). For 32-byte interleaving set this value equal to 0x3. For full page/row interleaving set this value equal to the column width value. Only the following values are supported: 0x3, 0x4, 0x5, 0x6, 0x7, 0x8, 0x9, 0xa, and 0xb. Recommended value is 3.
Offset 0x06 5008	***AUTO_HALT_LIMIT***			
31	PON	R/W	0	Controls PON signal of the SSTL_2 PADs: '1': May be set to '1' when the DDR devices are sent into self-refresh mode, i.e. after a HALT command. '0': Normal operation: must be set back to '0' before enabling again the DDR devices, i.e. before the UNHALT command.
30:0	LIMIT	R/W	-	After LIMIT amount of IP_2031 idle cycles, automatic halt kicks in.
The address locations of the DDR memory ranks are determined by registers RANK0_ADDR_LO, RANK0_ADDR_HI, and RANK1_ADDR_HI. Addresses in [RANK0_ADDR_LO, RANK0_ADDR_HI] are directed to rank 0, addresses in [RANK0_ADDR_HI, RANK1_ADDR_HI] are directed to rank 1. Addresses outside the two ranks are said to cause an address error.				
Offset 0x06 5010	***RANK0_ADDR_LO***			
31:0	ADDR_LO	R/W	0x0000 0000	Address at which the DDR rank 0 address space starts.
Offset 0x06 5014	***RANK0_ADDR_HI***			
31:0	ADDR_HI	R/W	0xFFFF FFFF	Address at which the DDR rank 0 address space ends.
Offset 0x06 5018	***RANK1_ADDR_HI***			
31:0	ADDR_HI	R/W	0xFFFF FFFF	Address at which the DDR rank 1 address space ends.
Dimension of DDR Memories				
Offset 0x06 5080	***DDR_MR***			
31:13	Unused	R	-	These bits should be ignored when read, and written as 0's.

Table 9: Register Description

Bit	Symbol	Access	Value	Description
12:0	MR	R/W	0x043	Mode register. The assumption is the DLL reset bit is at location 8. Use the datasheet of the DDR memory to determine the value of this register. The reset value of this register represents a CAS latency of 3.0 cycles, and a burst length of 8. **Make sure to select a burst size of 8, and a sequential burst type to ensure correct IP_2031 operation.** The following is taken from a DDR datasheet and describes the different bits of the mode register. Bits 0 up to 2: burst length Bit 3: burst type (‘0’: sequential, ‘1’: interleaved) Bits 4 up to 6: CAS latency Bits 7 and up: operating mode (‘0’: normal operation, ‘2’: normal operation/reset DLL)
Offset 0x06 5084	***DDR_EMR***			
31:13	Unused	R	-	These bits should be ignored when read, and written as 0s.
12:0	EMR	R/W	0x000	Extended Mode Register. Use the datasheet of the DDR memory to determine the value of this register. For emulation purposes it may be required to disable the DLL. To this end, make sure that bit 0 of this register contains a ‘1’. In normal (non-emulation) mode, make sure that bit 0 of this register contains a ‘0’. The following is taken from a DDR datasheet and describes the different bits of the extended mode register. Bit 0: DLL (‘0’: enable, ‘1’: disable). Bit 1: drive strength (‘0’: normal, ‘1’: reduced) Bit 2: QFC mode Bits 3 and up: operating mode
Offset 0x06 5088	***DDR_PRECHARGE_BIT***			
31:4	Unused	R	-	These bits should be ignored when read, and written as 0s.
3:0	PRECHARGE_BIT	R/W	0xa	Column bit responsible for precharge. Only the values 0x8 (bit 8) and 0xa (bit 10) are supported.
Offset 0x06 50C0	***RANK0_ROW_WIDTH***			
31:4	Unused	R	-	These bits should be ignored when read, and written as 0s.
3:0	ROW_WIDTH	R/W	0xd	Row dimension: 2^ROW_WIDTH rows i.e., a value of 0xC specifies 2^12 = 4096 rows. Only the following values are supported: 0x8, 0x9, 0xa, 0xb, 0xc, and 0xd (supporting 256 up to 8192 rows).
Offset 0x06 50C4	***RANK0_COLUMN_WIDTH***			
31:4	Unused	R	-	These bits should be ignored when read, and written as 0s.
3:0	COLUMN_WIDTH	R/W	0xa	Column dimension: 2^COLUMN_WIDTH columns (each column has a width of 32 bit). I.e., a value of 0xa specifies 2^10 = 1024 columns of 32 bit each. Only the following values are supported: 0x8, 0x9, 0xa, and 0xb (supporting 256 up to 2048 columns).
Offset 0x06 50D0	***RANK1_ROW_WIDTH***			
31:4	Unused	R	-	These bits should be ignored when read, and written as 0s.

Table 9: Register Description

Bit	Symbol	Access	Value	Description
3:0	ROW_WIDTH	R/W	0xd	Row dimension: 2^ROW_WIDTH rows. I.e., a value of 0xc specifies 2^12 = 4096 rows. Only the following values are supported: 0x8, 0x9, 0xa, 0xb, 0xc, and 0xd (supporting 256 up to 8192 rows).
Offset 0x06 50D4	***RANK1_COLUMN_WIDTH***			
31:4	Unused	R	-	These bits should be ignored when read, and written as 0's.
3:0	COLUMN_WIDTH	R/W	0xa	Column dimension: 2^COLUMN_WIDTH columns (each column has a width of 32 bit). I.e., a value of 0xa specifies 2^10 = 1024 columns of 32 bit each. Only the following values are supported: 0x8, 0x9, 0xa, and 0xb (supporting 256 up to 2048 columns).
Timing Characteristics				
Offset 0x06 5100	***DDR_TRCD***			
31:20	Unused	R	-	These bits should be ignored when read, and written as 0s.
19:16	TRCD_WR	R/W	2	Minimum time between active and write command (RAS to CAS delay). When the datasheet of the DDR memory does not specify a value for this timing parameter, use the value as specified for TRCD. Must be greater or equal than tRAP.
15:4	Unused	R	-	These bits should be ignored when read, and written as 0s.
3:0	TRCD_RD	R/W	4	Minimum time between active and read command (RAS to CAS delay). When the datasheet of the DDR memory does not specify a value for this timing parameter, use the value as specified for TRCD. Must be greater or equal than tRAP.
Offset 0x06 5104	***DDR_TRC***			
31:4	Unused	R	-	These bits should be ignored when read, and written as 0's.
3:0	TRC	R/W	0xd	Minimum time between two active commands to the same bank.
Offset 0x06 5108	***DDR_TWTR***			
31:4	Unused	R	-	These bits should be ignored when read, and written as 0's.
3:0	TWTR	R/W	2	Write to read command delay
Offset 0x06 510C	***DDR_TWR***			
31:4	Unused	R	-	These bits should be ignored when read, and written as 0's.
3:0	TWR	R/W	3	Write recovery time. Must be greater or equal than tWR_A. TWR+TRP must be greater or equal than tDAL.
Offset 0x06 5110	***DDR_TRP***			
31:4	Unused	R	-	These bits should be ignored when read, and written as 0's.
3:0	TRP	R/W	4	Precharge command period. TWR+TRP must be greater or equal than tDAL.
Offset 0x06 5114	***DDR_TRAS***			
31:4	Unused	R	-	These bits should be ignored when read, and written as 0s.
3:0	TRAS	R/W	9	Minimum delay from active to precharge.
Offset 0x06 511C	***DDR_TRRD***			
31:4	Unused	R	-	These bits should be ignored when read, and written as 0's.
3:0	TRRD	R/W	2	Active bank a to active bank b command

Table 9: Register Description

Bit	Symbol	Access	Value	Description
Offset 0x06 5120	***DDR_TRFC***			
31:4	Unused	R	-	These bits should be ignored when read, and written as 0's.
3:0	TRFC	R/W	0xf	Auto refresh command period.
Offset 0x06 5124	***DDR_TMRD***			
31:4	Unused	R	-	These bits should be ignored when read, and written as 0's.
3:0	TMRD	R/W	2	Load mode register command cycle time.
Offset 0x06 5128	***DDR_TCAS***			
31:4	Unused	R	-	These bits should be ignored when read, and written as 0s.
3:0	TCAS	R/W	8	CAS read latency, specified in halve cycles. I.e., a value of 0b0111 (7) represents a CAS delay of 3.5 cycles (7 halve cycles).
Offset 0x06 512C	***DDR_RF_PERIOD***			
31:6	Unused	R	-	These bits should be ignored when read, and written as 0s.
15:0	RF_PERIOD	R/W	3515	Refresh period expressed in terms of cycles. Typically a refresh is required at an average interval of 15.625 us. For a 100 MHz. device this translates into a RF_PERIOD value of 1562. For a 200 MHz. device this translates into a RF_PERIOD value of 3125.
Arbitration Parameters				
Offset 0x06 5180	***ARB_CTL***			
31	CPU_DMA_DECR	R/W	1	'0': Do not decrement CPU counters when in a DMA_WINDOW. '1': Do decrement CPU counters when in a DMA_WINDOW.
30	CPU_HRT_SRT_ENABLE	R/W	0	'0': Controller will interpret that DMA port contains only Hard Real Time DMA requests. '1': Controller will interpret that DMA port contains Hard Real Time or Soft Real Time DMA requests.
29	BLB_ENABLE	R/W	0	'0': Disable Back Log Buffer '1': Enable Back Log Buffer.
28	DYN_RATIOS	R/W	0	'0': Use Static Ratios. This means accounts are incremented by the value (RATIO+ DDR burst size (in terms of cycles)) whenever a CPU DDR burst is performed. '1': Enable Dynamic Ratios. This means accounts are incremented by the value RATIO every clock cycle that is spent on servicing a CPU DDR burst. This feature also causes account not to decrement during clock cycles that are spent on CPU DDR bursts.
27:18	Reserved	R	-	These bits should be ignored when read, and written as 0s.

Table 9: Register Description

Bit	Symbol	Access	Value	Description
17:16	CPU_PREEMPT[1]	R/W	0x1	0x0: No preemption (once a CPU MTL command has started to enter the DDR arbitration buffer, it will go completely into the DDR arbitration buffer, uninterrupted by other (CPU or DMA) MTL commands. 0x1: Preempt a CPU MTL command when it started to enter the DDR arbitration buffer while inside of the DMA window, and is currently active in the DMA window. The CPU MTL command will only be interrupted by a DMA MTL command, not by another CPU MTL command. 0x2: Undefined 0x3: Preempt a CPU MTL command that is currently active in the DMA window (independent of when it started to enter the DDR arbitration buffer).The CPU MTL command will only be interrupted by a DMA MTL command, not by another CPU MTL command. Recommended value is 0.
15:2	Unused	R	-	These bits should be ignored when read, and written as 0s.
1:0	DMA_PREEMPT[2]	R/W	0x1	0x0: No preemption (once a DMA MTL command has started to enter the DDR arbitration buffer, it will go completely into the DDR arbitration buffer, uninterrupted by other (CPU or DMA) MTL commands. 0x1: Preempt a DMA MTL command when it started to enter the DDR arbitration buffer while inside of the CPU window, and is currently active in the CPU window. The DMA MTL command will only be interrupted by a CPU MTL command, not by another DMA MTL command. 0x2: Undefined 0x3: Preempt a DMA MTL command that is currently active in the CPU window (independent of when it started to enter the DDR arbitration buffer).The DMA MTL command will only be interrupted by a CPU MTL command, not by another DMA MTL command. If enabled recommended value is 3.

[1] The preemption field determines the aggressiveness with which MTL commands are preempted when they are active in a window that was not meant for the MTL command. Value 0 represents low aggressiveness, value 0x1 represents medium aggressiveness, and value 0x3 represents high aggressiveness. The more aggressive, the better the time multiplexing by means of windows is accomplished. However, aggressive preemption may result in lower overall bandwidth.

[2] See above footnote.

Offset 0x06 5184	***ARB_HRT_WINDOW***			
31:16	Unused	R	-	These bits should be ignored when read, and written as 0s.
15:0	WINDOW	R/W	0x003f	Window size for Hard Real-Time (HRT) MTL requests (in terms of clock cycles). Add 1 for the real effective window size.
Offset 0x06 5188	***ARB_CPU_WINDOW***			
31:16	Unused	R	-	These bits should be ignored when read, and written as 0s.
15:0	WINDOW	R/W	0x003F	Window for Central Processor Unit (CPU) MTL requests (in terms of clock cycles). Add 1 for the real effective window size
Offset 0x06 51C0	***ARB_CPU_LIMIT***			
31:16	Unused	R	-	These bits should be ignored when read, and written as 0s.

Table 9: Register Description

Bit	Symbol	Access	Value	Description
15:0	LIMIT	R/W	0xFFFF	When the DDR controller internal CPU account exceeds this value, no CPU DDR burst will be performed when DMA traffic is present (CPU traffic has lower priority than DMA traffic). The internal CPU account is decremented by DECR every clock cycle. For increment information see DYN_RATIOS description.

[1] See register ARB_CPU_RATIO for a description of the RATIO value.

[2] When transferring a burst of n 32-bit data elements at a double data rate, the burst size in terms of clock cycles is n/2.

Offset 0x06 51C4	***ARB_CPU_RATIO***			
31:8	Unused	R	-	These bits should be ignored when read, and written as 0s.
7:0	RATIO	R/W	0x04	If DYN_RATIOS are disabled the value is added to the internal account for each CPU DDR burst. If DYN_RATIOS are enabled then this value is added to the internal account for each clock cycle spent on a CPU DDR burst.
Offset 0x06 51C8	***ARB_CPU_CLIP***			
31:16	Reserved	R	-	These bits should be ignored when read, and written as 0s.
15:0	CLIP	R/W	0xFFFF	CPU account clip. When the internal account goes above this value the CPU DDR bursts are ‘for free’. This value should always be equal or higher than LIMIT.
Offset 0x06 51CC	***ARB_CPU_DECR***			
31:8	Reserved	R	-	These bits should be ignored when read, and written as 0s.
7:0	DECR	R/W	0x01	CPU account decrement. This value is used to decrement the internal account of each clock cycle (with some exceptions).

Performance Measurement

To allow for performance evaluation, the DDR SDRAM Controller includes a set of registers that measures data traffic. Incremental 32-bit counters are used to measure the read and write traffic on every MTL port separately.

Offset 0x06 5200	***PF_MTL0_RD_VALID***			
31:0	MTL_RD_VALID	R/W	-	Counter for valid MTL read data elements.
Offset 0x06 5204	***PF_MTL0_WR_ACCEPT***			
31:0	MTL_WR_ACCEPT	R/W	-	Counter for valid MTL write data elements.
Offset 0x06 5208	***PF_MTL1_RD_VALID***			
31:0	MTL_RD_VALID	R/W	-	Counter for valid MTL read data elements.
Offset 0x06 520C	***PF_MTL1_WR_ACCEPT***			
31:0	MTL_WR_ACCEPT	R/W	-	Counter for valid MTL write data elements.
Offset 0x06 5240	***PF_IDLE***			
31:0	IDLE	R/W	-	Counts cycles in which the DDR memory controller is considered to be idle (not valid entries on the top of the DDR arbitration queue).

Errors

These registers can be used to observe DDR memory addressing errors. If an MTL command is referring to an address outside the DDR addressable region, the MTL command specifics are registered in the error registers, and an interrupt to the TM3260 is raised to indicate the error. In the case of multiple successive errors, the MTL command that caused the first error is registered, but successive errors are not registered (until the VALID field of ERR_VALID is set to ‘0’).

Offset 0x06 5280	***ERR_VALID***			
31:1	Unused	R	-	These bits should be ignored when read, and written as 0s.

Table 9: Register Description

Bit	Symbol	Access	Value	Description
0	VALID	R/W	0x0	'0': no error '1': error This is used to acknowledge the interrupt error indication.
Offset 0x06 5284	***ERR_MTL_PORT***			
31:2	Unused	R	-	These bits should be ignored when read, and written as 0s.
1:0	MTL_PORT	R	-	MTL port that caused the error.
Offset 0x06 5288	***ERR_MTL_CMD_ADDR***			
31:0	MTL_CMD_ADDR	R	-	MTL command address.
Offset 0x06 528C	***ERR_MTL_CMD_READ***			
31:1	Unused	R	-	These bits should be ignored when read, and written as 0s.
0	MTL_CMD_READ	R	-	MTL command read.
Offset 0x06 5290	***ERR_MTL_CMD_ID***			
31:10	Unused	R	-	These bits should be ignored when read, and written as 0's.
9:0	MTL_CMD_ID	R	-	MTL command identifier. See Section 2.2 on page 26-2 for ID codes.
Offset 0x06 5FFC	***MODULE_ID***			
31:16	MODULE_ID	R	0x2031	DDR memory controller module ID
15:12	MAJOR_REV	R	1	Major revision number
11:8	MINOR_REV	R	1	Minor revision number
7:0	APERTURE	R	0	Aperture size is 4 KB ((APERTURE+1)* 4 KB).

6. References

[1] Double Data Rate (DDR) SDRAM specification, JEDEC standard JESD79, June 2000, JEDEC Solid State Technology Association

[2] EIA/JEDEC Standard, Stub Series Terminated Logic for 2.5 Volts (SSTL_2), EIA/JESD8-9, September 1998, Electronic Industries Association, JEDEC Solid State Technology Division

Chapter 10: LCD Controller

PNX15xx Series Data Book – Volume 1 of 1

Rev. 2 — 1 December 2004 Product data sheet

1. Introduction

The LCD controller is required to control the power sequencing of the LCD panel. All the other functions required for an LCD controller like color expansion, screen timing generation is taken care by the QVCP (Quality Video Composition Processor). QVCP also does some video enhancement functions. Please refer to the QVCP documentation for the details about these functions.

1.1 LCD Controller Features

The following feature allows PNX15xx Series to be connected to many LCD models:

- Automatic power on/off sequencing
- Programmable delays for the power sequencing
- Polarity control for power enable and back light control signals
- Data Enable signal generation

2. Functional Description

2.1 Overview

The LCD controller receives the parallel video out data from the QVCP along with the timing signals (HSYNC, VSYNC, CBLANK and CLK_LCD) and applies power sequencing before sending it out to the LCD interface. It converts CBLANK signal into

DE as required by the LCD panel. Apart from these timing signals, it also generates the power enable (TFTVDDON) and back light control (TFTBKLTON) signals that are required for some LCD's. Figure 1 presents the LCD controller block diagram.

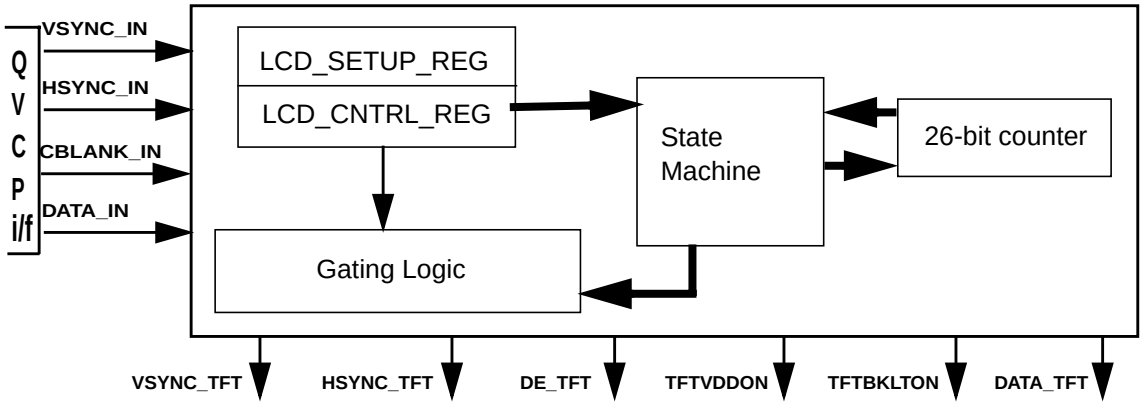


Figure 1: Block diagram of the LCD Controller

2.2 Power Sequencing

LCDs are very sensitive to the power sequencing. Not following these rules may create a latch-up or DC effect that would damage the LCD panel. Figure 2 pictures the generic power sequence constraints.

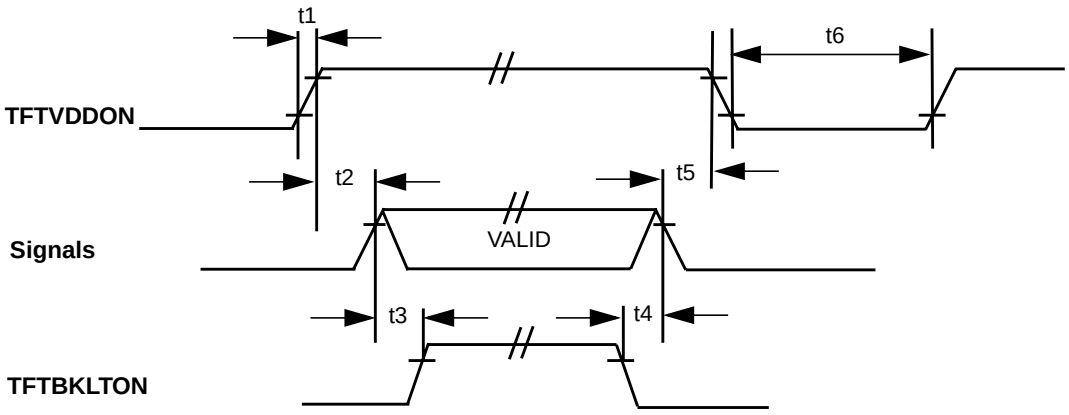


Figure 2: Generic Power Sequence for TFT LCD Panels

At power up of the system, the LCD panel remains without any power supply applied. The LCD controller provides a signal, TFTVDDON, to power the LCD panel. There is some constraint on the ramping up of the power supply (time constraint **t1**). But this time constraint is board related and hence the LCD controller does not provide any support. Once the power is stable, the data/control signals (Data/HSync/VSync/ DataEnable) may be driven after **t2** (before the TFT power supply is on, the data/ control signals must be at 0 V). After the data/control signals have become valid, a minimum time, **t3**, is required before the BackLight of the panel can be turned on.

Similar power sequence applies for the power down sequence, resulting in the defined **t4** and **t5** parameters.

After a power down sequence is completed, a minimum time, **t6**, is necessary before the next power up sequence can be started.

These delay values, t2, t3, t4, t5, and t6, are programmable in the LCD controller.

3. Operation

3.1 Overview

After reset, an initialization program (like an LCD driver) sets up the values in the LCD_SETUP register. This register is used to enable the LCD interface and to specify the power sequencing delay values needed for the particular LCD panel. Refer to Section 4. on page 10-6 for the MMIO register layout details. This register is implemented as a ‘write once’ register to prevent a software application from changing the delay values after the initialization program has set the correct values. Programming incorrect values may damage the LCD panel.

When the software is ready to send data to the LCD panel, it sets START_PUD_SEQ bit in the LCD_CONTROL register. This starts the power up sequencing. Similarly, when the software wants to shut down the LCD panel, it resets the bit. This starts the power down sequencing.

The power sequencing is controlled by a state machine to guaranty all the critical timing parameters.

3.2 Power Sequencing State Machine

The state machine in the LCD controller generates the control signals to gate the data/control signals for the LCD interface. On reset these signals are de-asserted so that the LCD interface is disabled. Once the power up sequence is started, these signals are asserted in the order required for the power up sequence. The delays are

calculated using a 26-bit counter that is controlled by the state machine. This counter runs on the 27 MHz clock (the input PNX15xx Series crystal). The state machine is shown in Figure 3.

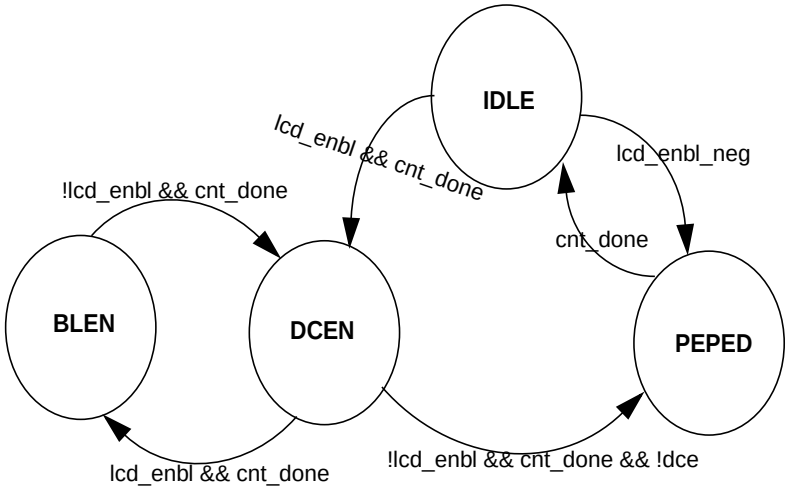


Figure 3: Power Sequencing State Machine Block Diagram

3.2.1 IDLE state

After reset, the state machine comes up in the IDLE state. In this state, when the lcd_enbl signal (which is asserted when both lcd_if_en and start_pud_seq bits are set) is asserted, the power up sequence is started by asserting the TFTVDDON signal and loading the counter with PWREN_DCE_DELAY that is set in the LCD_SETUP register. The counter starts to count down after it is loaded.

If the lcd_enbl is de-asserted in the IDLE state, then the state machine goes to the PEPED state, de-asserts the TFTVDDON signal and loads the counter with PWR_EN_PWREN_DELAY value.

If the lcd_enbl is still asserted when the counter decrements to zero, then the state machine goes to DCEN state and asserts the ‘dce’ signal. It also loads the counter with DCE_BKLT_DELAY value.

3.2.2 DCEN state

In the DCEN state, when the counter reaches zero and lcd_enbl is still asserted, then the state machine transitions to the BLEN state and asserts the TFTBKLTON signal. This completes the power up sequence.

If the lcd_enbl signal is de-asserted when ‘dce’ signal is still asserted, then the ‘dce’ signal is de-asserted and the counter is loaded with DCE_PWREN_DELAY value. There is no state transition.

If the counter reaches zero with both the dce and lcd_enbl signal de-asserted, the state machine transitions to the PEPED state. During this transition, the TFTVDDON signal is de-asserted and the counter is loaded with PWREN_PWREN_DELAY value.

3.2.3 BLEN state

In the BLEN state, when the lcd_enbl signal is de-asserted, the TFTBKLTON signal is de-asserted and the counter is loaded with BKLT_DCE_DELAY value. There is no state transition. When the counter reaches zero with lcd_enbl signal still de-asserted, the state machine moves to the DCEN state de-asserting the dce signal. During this transition, the counter is loaded with DCE_PWREN_DELAY value.

If the lcd_enbl signal is asserted in the BLEN state, the TFTBKLTON signal is asserted and there is no state change.

3.2.4 PEPED state

In the PEPED state, the state machine waits for the counter to reach zero to force the PWREN_PWREN_DELAY and goes back to the IDLE state. This completes the power down sequencing. If the lcd_enbl signal is asserted when the counter reaches zero, a new power up sequencing is started.

3.3 Counter

The counter used to calculate the delays is a 26-bit down counter. It starts counting down as soon as it is loaded with a delay value and asserts the cnt_done signal when the counter reaches zero. It runs on the 27 MHz clock (input PNX15xx Series crystal).

3.4 Gating Logic

The control signals from the state machine are in the clk_lcd_tstamp clock domain. They are first synchronized to the clk_lcd clock domain before using them to gate the data/control signals from the QVCP. The clk_lcd clock is also gated without any glitch in the gating logic. The clock gating circuit is shown in Figure 4.

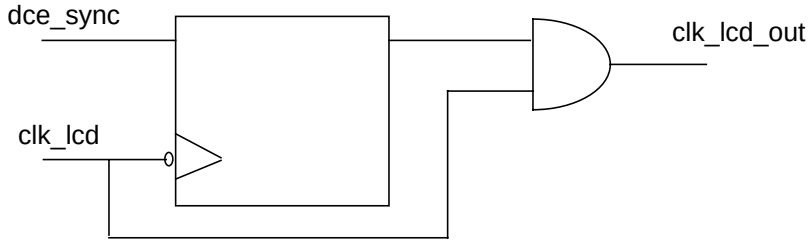


Figure 4: Clock Gating Logic

4. Register Descriptions

A summary of the LCD controller MMIO register is presented in Table 1 and the layout of the MMIO registers is described in Section 4.1.

Table 1: LCD Controller Register Summary

Offset	Name	Description
0x07,3000	LCD_SETUP	Supports programmable delay for the power sequencing.
0x07,3004	LCD_CNTRL	Control register to start the power on/off sequencing.
0x07,3008	LCD_STATUS	Gives the status of power up/down sequencing.
0x07,300C—07,3FF0	Reserved	
0x07,3FF4	LCD_DISABLE_IF	To disable the MMIO interface for power management.
0x07,3FF8	Reserved	
0x07,3FFC	LCD_MODULE_ID	Module ID number, including major and minor revision levels.

4.1 LCD MMIO Registers

Table 2: LCD CONTROLLER Registers

Bit	Symbol	Access	Value	Description
Offset 0x07,3000	***LCD_SETUP***			
Note 1: This is a special register with respect to write. **This is a "write once" register.** It is implemented this way to prevent a software application from altering the delay values after the setup software initialized them correctly. This protects the LCD panel from being damaged by an incorrect write by a software application. **Even if default values are desired, do a write to the register so that the write-once protection mechanism takes effect.** **Note 2:** **The delay values are based on a 27 MHz clock.** **Note 3:** Please refer to Figure 22-2 to correlate the delay values.				
31	LCD_IF_EN	R/W1	1	This bit enables the LCD interface. If this bit is set, then power sequencing will be applied to the data/control signals based on the value of START_PUD_SEQ bit in LCD_CNTRL register. If this bit is not set, then all the LCD interface signals will remain de-asserted. When this bit is set, the output router block will select the LCD interface overriding any other programming of the mux in the output router.
30	VDD_POL	R/W1	1	1 = TFTVDDON is of positive polarity. 0 = TFTVDDON is of negative polarity.
29	BKLT_POL	R/W1	1	1 = TFTBKLTON is of positive polarity. 0 = TFTBKLTON is of negative polarity.
28:20	Unused	-	-	
19:16	PWREN_PWREN_DELAY	R/W1	11	Delay from the end of a power down sequence to the start of the next power up sequence. Delay value t6 in steps of 100 ms with 0 corresponding to 100 ms and 15 corresponding to 1600 ms.
15:12	DCE_PWREN_DELAY	R/W1	3	Delay from data/control signals de-assertion to the de-assertion of TFTVDDON signal. Delay value t5 in steps of 10 ms with 0 corresponding to 10 ms and 15 corresponding to 160 ms.
11:8	BKLT_DCE_DELAY	R/W1	7	Delay from de-assertion of TFTBKLT signal to the de-assertion of data/control signals. Delay value t4 in steps of 100 ms with 0 corresponding to 100 ms and 15 corresponding to 1600 ms.
7:4	DCE_BKLT_DELAY	R/W1	2	Delay from assertion of data/control signals to TFTBKLT signal assertion. Delay value t3 in steps of 100 ms with 0 corresponding to 100 ms and 15 corresponding to 1600 ms.
3:0	PWREN_DCE_DELAY	R/W1	3	Delay from assertion of TFTVDDON signal to assertion of data/control signals. Delay value t2 in steps of 10 ms with 0 corresponding to 10 ms and 15 corresponding to 160 ms.
Offset 0x07,3004	***LCD_CNTRL***			
Note: A board level power sequencing must also be observed to ensure that the LCD panel is powered and ready to accept the power-up sequence before the power-up sequence is started from PNX15xx Series.				
31:1	Unused	-	-	
0	START_PUD_SEQ	R/W	0	Writing a 1 (when the bit is 0) will start a power up sequencing and writing a 0 (when the bit is 1) will start a power down sequencing. When the LCD interface is not enabled, this bit always stays 0.
Offset 0x07,3008	***LCD_STATUS***			
31:1	Unused	-	-	

Table 2: LCD CONTROLLER Registers *…Continued*

Bit	Symbol	Access	Value	Description
0	PUD_STATUS	R	1	This is a read only bit that gives the status of power up or down sequence, If this bit is set to '1' and START_PUD_SEQ bit is '1', then the power up sequence has been completed. If this bit is set to '1' and START_PUD_SEQ bit is '0', then the power down sequence has been completed. If this bit is '0', then either a power up or down (depending on START_PUD_SEQ bit) sequence is in progress.
Offset 0x07,300C—0x07,3FF0			***Reserved***	
Offset 0x07,3FF4		***LCD_DISABLE_IF***		
31:1	Unused		-	
0	DISABLE_IF	R/W	0	Setting this bit to '1' disables the MMIO interface. The only valid transactions are to the interface disable register. All other transactions are not valid, write transactions will generate a write error and read transactions will return 0xDEADABBA. The bit is used to provide power control status for system software power management.
Offset 0x07,3FF8		***Reserved***		
Offset 0x07,3FFC		***LCD_MODULE_ID***		
31:16	ID	R	0xA050	Module ID. This field identifies the module as the LCD controller.
15:12	MAJ_REV	R	0x0	Major Revision ID. This field is incremented by 1 when changes introduced in the module result in software incompatibility with the previous version of the module. First version default = 0.
11:8	MIN_REV	R	0x0	Minor Revision ID. This field is incremented by 1 when changes introduced in the module result in software compatibility with the previous version of the module. First version default = 0.
7:0	APERTURE	R	0x00	Aperture size. Identifies the MMIO aperture size in units of 4 KB for the LCDC module. LCDC has an MMIO aperture size of 4 KB.

Chapter 11: QVCP

PNX15xx Series Data Book – Volume 1 of 1

Rev. 2 — 1 December 2004 Product data sheet

1. Introduction

The QVCP (Quality Video Composition Processor) is a high-resolution image composition and processing pipeline that facilitates both graphics and video processing. In combination with several other modules, it provides a new generation of graphics and video capability, far exceeding the older standards. QVCP provides its advanced functionality using a series of layers and mixers; a series of display-data layers (pixel streams) are created and logically mixed in sequence to render the composite output picture.

The PNX15xx Series hosts one QVCP module. The display processor (QVCP) contains a total number of two layers and is mainly intended to be connected to a TV, a monitor or an LCD panel. Due to the independence of the layers, a number of different scenarios is possible. However, in general, the QVCP has been designed to mix one video plane and one graphic plane. It can therefore be used to display a fully composited video image consisting of PIP(s), menu(s), and other graphical information.

In this document, the words surface or plane are used to replace layer depending on the context.

QVCP supports a whole range of progressive and interlaced display standards: for televisions, from standard-definition resolutions such as PAL or NTSC to all eighteen ATSC display formats such as 1080i or 720p, and for computer and LCD displays, from VGA to W-XGA resolutions at 60 Hz. The wide variety of output modes guarantees the compatibility with most display-processor chips.

In order to achieve high-quality video and graphics as demanded by future consumer products, a number of complex tasks need to be performed by the QVCP. The main functions of the video and graphics output pipeline are listed below:

- Fetching of up to two image surfaces from memory
- Color expansion in case of non-full color or indexed data formats
- Reverse-gamma correction
- Video quality enhancement such as luminance sharpening, chroma transient improvement, histogram modification, skin-tone correction, blue stretch, and green enhancement.
- Horizontal up-scaling for video and graphics images in both linear and panorama mode.

- Screen timing generation adopted to the connected display requirements (SD-TV standards, HD-TV standards, progressive and interlaced formats)
- Color space uniqueness of all the display surfaces
- Merging of the image surfaces (blend, invert, exchange)
- Positioning of the various surfaces (including finer positioning)
- Brightness and contrast control on a per-surface basis
- Gamma correction and noise shaping of the final composited image
- Output format generation

1.1 Features

QVCP comprises various processing **layers**, a hierarchical **mixer** cascade (where an image surface is always associated with a layer and a mixer structure), and an **output pipeline**. A top-level block diagram is shown in Figure 1.

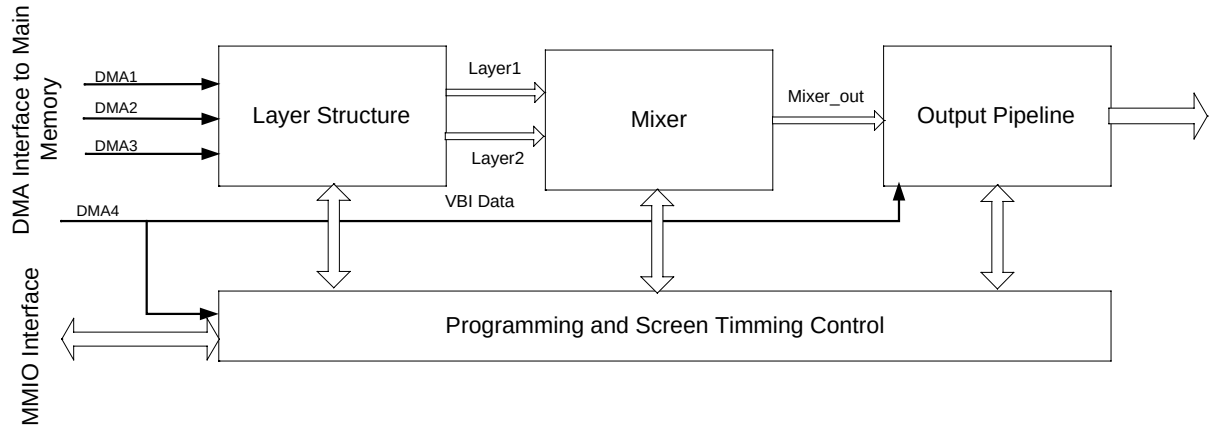


Figure 1: QVCP Top Level Diagram

A *layer* contains various video and graphics processing functions (which are necessary to accomplish the tasks mentioned above) *and* obtains data from a particular data source. The data may provide a desktop image, a motion video image, a cursor, or a sprite image. Registers in the layer select the data source and set the display and the image-processing parameters.

A *mixer* is a functional block that selects between and manipulates data streams from two sources: the pixel stream from its companion layer and the pixel stream output of the previous mixer (in the hierarchy). The mixing functions include pixel inversion, pixel selection (between sources), and alpha blending. The mixers operate on a per-pixel basis using programmable logical raster operations (ROPs) to determine which

functions to apply. The keys used in the raster operations include chroma keying (color-keying on a color range) and color keying. The output of the mixer is a continuous stream of pixels at the video-clock rate going to the display device.

The output of the mixer hierarchy is connected to the *output pipeline*. The output pipeline comprises gamma correction, dithering (noise shaping), and reformatting. The formatter inserts VBI data in the horizontal blanking interval and re-formats the final output-data stream according to the requirements of the connected device.

2. Functional Description

2.1 QVCP Block Diagram

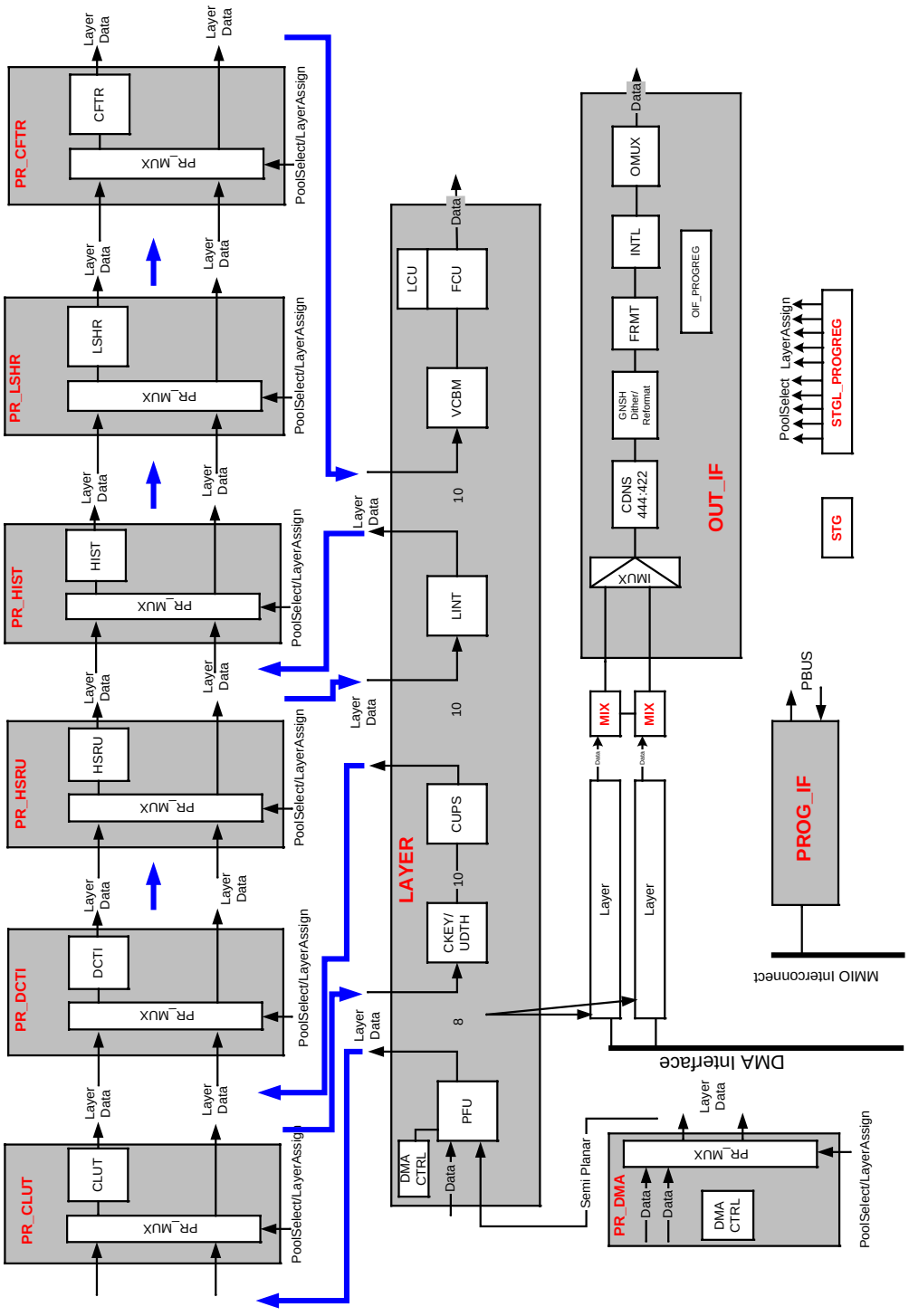


Figure 2: QVCP BLock Diagram

The basic block diagram of the QVCP is illustrated in Figure 2. The front-end part accommodates 2 symmetrical layers, which suggest 2 image surfaces with independent characteristics such as pixel data format, color space, size and position on the final composite image. Each of these layers is tied into the memory access infrastructure of the PNX15xx Series via an independent DMA read interface. A layer (or a layer module, as it is called) is responsible for producing a valid pixel for every display coordinate. Both layer modules, as mentioned before, are identical, and so, there are no restrictions as to how each layer may be utilized: 2 video layers, 2 graphics layers, or 1 graphic + 1 video layer are examples of some of the combinations that can be achieved. However, as described later, there are some restrictions on the image improvements that can be applied per surface. A wide variety of RGB, YUV, and alpha blend formats are supported. Each layer, as detailed later, can perform a variety of video-processing functions such as color space conversion, 4:4:4 to 4:2:2 down-filtering and 4:2:2 to 4:4:4 up-sampling, color- and chroma key extraction, etc. It should be noted that “region-based graphics” is not supported at the hardware level; software must generate one uniform color depth surface if an application requires region based graphics.

2.2 Architecture

The QVCP is architected using the concept of **virtual identical layers with a common resource pool**. Each layer is built as a skeleton which contains only the essentialprocessing blocks. The remaining processing blocks --- the more exotic ones responsible for picture enhancement, for example --- are part of the pool resources.The principal assumption, in defining the QVCP architecture, is that there is no use-case which requires all of the features to be active in all of the layers at the same time. For any particular use-case, there will be a specific selection of features required in each layer. All layers can make use of pool resources. There is no specific order or assignment of the pool resources to the specific layers. However, prior to using QVCP in the context of a specific application scenario, it is required to assign resources from the pool to specific layers. This is done via a set of global QVCP resource-allocation registers.

In addition to the symmetric layer structure, the QVCP contains, as mentioned before, a set of (special) image processing functions which are located outside of the layers in a resource pool. The pool-resource concept takes into account the fact that software drivers would like to access the layers in a symmetrical unified fashion. The features used in a certain display scenario are however not symmetrically distributed among the layers at all. For a given application scenario, there is no case when every layer uses all its resources. Therefore theses features which are never used by all layers at the same time are located in the resource pool. Hence, the pool contains only a number of functional units of the same kind which is smaller than the total number of layers.

Attached to each layer is a mixer which acts as a three-port image combination module, combining the image coming from the layer attached to the mixer with the image coming from the previous mixer. The resultant image is forwarded to the next mixer. This mixer cascade implies a certain layer, and therefore a certain image, order on the final display surface.

The outputs of all mixers are connected to the back-end part of the QVCP --- the output formatter. The output formatter performs all necessary functions to adopt the final composited image to the display requirements. Among the functions performed in the output formatter are: gamma correction, chrominance downsampling, output formatting, and VBI insertion.

2.3 Layer Resources and Functions

This section focuses on the elements comprising a layer. Note that all of the described modules are present in each layer exactly once, the justification being that they (elements) are either always needed for the basic operation of a layer or they are so small (in design size) that assigning them to the resource pool would be inefficient due to the multiplexing and routing overhead associated with the pool elements.

2.3.1 Memory Access Control (DMA CTRL)

QVCP has 4 DMA agents, each of which connects to a 512-Byte buffer in the DMA adapter. DMA agents 1-2 are hard coded to layer1-2 respectively. DMA4 is used to fetch a VBI packet or a data packet for DMA-based control-register programming. DMA3 can be assigned to any of the two layers for supporting the semi-planar input format.

For video data fetches, the request block size is equal to the initial layer width (before horizontal scaling). If start_fetch is disabled (i.e., Enable bit 31 of register 0x10E2C8 is programmed to zero), the first DMA request starts right after the layer_enable is asserted and QVCP works as if prefetch is enabled. However, if start_fetch is enabled (i.e., Enable bit 31 of register 0x10E2C8 is programmed to one), then the DMA starts fetching only when QVCP s internal line counter reaches the 12-bit line threshold programmed in the Fetch Start bits [11:0] of register 0x10E2C8. Data fetched for the first field (interlaced) or frame (progressive) is not used and is flushed at the FCU (Flow Control Unit) FIFO. Thereafter, the pixels for the second field/frame start marching into the FCU FIFO, waiting for the correct layer position. The FCU FIFO releases pixels only if the x,y coordinates generated by the Screen Timing Generator (STG) match the layer position. In case of an interlaced output, the field ID is also checked. The DMA fetch request for the next active video line starts as soon as the last active pixel of the current line moves from the adapter FIFO into the processing pipeline and this request must be served in time to guarantee that the first active pixel of this new line is ready at the FCU FIFO before the STG signals the start of active video for the new line.

The DMA-based register-control programming only needs to be done once for a particular display scenario; thus, DMA4 is mainly used for VBI data fetch. QVCP is designed such that a VBI packet will only be inserted in the horizontal blanking interval and only one VBI packet is allowed in any one horizontal blanking interval. To insert this packet, there are two DMA requests. The first request has a block size of 1 since it is used to fetch only the header (which contains the size information). The second request is meant to fetch actual data of the required size and so, the maximum DMA request size (for the second request) is equal the length of horizontal blanking interval. The VBI data for the current horizontal blanking interval is always fetched in advance and stored in the DMA buffer (in the adapter). After sending out this prefetched data, the VBI DMA control unit (DMA4) requests a prefetch of the next packet (and correct operation requires that the sequence of the next two fetches must

complete before the start of the next horizontal blanking interval). In the current design, the VBI packet can be inserted only between the EAV and the SAV time codes.

2.3.2 Pixel Formatter Unit (PFU)

The PFU retrieves the raw data stream, for a particular image source, from the system memory and formats it according to the specified pixel format. Table 1 summarizes the various native pixel formats supported by the PFU.

Table 1: Summary of Native Pixel Formats

Format	Description
1 bpp indexed	CLUT entry = 24-b color + 8-b alpha
2 bpp indexed	CLUT entry = 24-b color + 8-b alpha
4 bpp indexed	CLUT entry = 24-b color + 8-b alpha
8 bpp indexed	CLUT entry = 24-b color + 8-b alpha
RGBa 444	16-b unit, containing 1 pixel with alpha
RGBa 4534	16-b unit, containing 1 pixel with alpha
RGB565	16-b unit, containing 1 pixel, no alpha
RGBa 8888	32-b unit, containing 1 pixel with alpha
Packed YUVa 4:4:4	32-b unit, containing 1 pixel with alpha
Packed YUV 4:2:2 (UYVY)	16-b unit, 2 successive units contain 2 horizontally-adjacent pixels, no alpha
Packed YUV 4:2:2 (YUY2, 2vuy)	16-b unit, 2 successive units contain 2 horizontally-adjacent pixels, no alpha
Semiplanar YUV 4:2:2	Separate memory planes for Y and UV (both 8 and 10 bpp)

Remark: Semiplanar YUV 4:2:0 is supported by the software API. However the support is achieved by duplicating the UV samples. Therefore it is not a true 4:2:0 support. True 4:2:0 support can be achived by using the MBS module to convert the 4:2:0 pixels into 4:2:2 or 4:4:4 pixels before entering QVCP.

Remark: The PFU does not do the pixel conversion it just formats properly the raw data into the QVCP pipeline, therefore it requires to know the input pixel format.

2.3.3 Chroma Key and Undither (CKEY/UDTH) Unit

Chroma keying allows overlaying of video and/or graphic layers, depending on whether the considered pixel lies within a specified color range. This feature allows video transparency or “green screening” — a technique used for compositing a foreground imagery with a background. Undithering allows recovery of 9 bits of precision from the dithered 8 bits stored in memory (where QVCP fetches the pixels for video processing).

CKEY

Chroma key matching is usually performed on a range of values rather than on a single value as in *color keying;* a Key Mask is provided to allow chroma keying on specific bits within the data word considered. In QVCP, the *color key* is generated in each layer in the source color space. Every layer can use up to four color keys.

Chroma keying allows overlaying video depending on whether the pixel lies within a range of color values. This feature allows video transparency or “green screening”—a technique for compositing foreground imagery with a background.

Color keying is considered to be a subset of chroma keying and can be achieved by setting the color range accordingly; because of the similarity between Chroma and Color keying (the difference being a color range instead of a fixed single color), this document will use the terms interchangeably, even though QVCP implements both functionality. The chroma/color key result is used for mixing functions downstream. For example, color keying can be used to determine which pixels should contain motion video. The color key function will compare the pixels of a layer with the color key register and place motion video from another layer in those pixels that match. The Color Key is generated in each layer (before color space conversion). Each byte lane of the expanded RGB 8:8:8 or YUV 4:4:4 new pixel is passed through an 8-bit Color Key AND mask. (For the YUV 4:2:2 input format, the U and V samples are repeated for every other pixel). The result in each channel is compared to the register values for Color Key Lower Range and Color Key Upper Range. Every layer can use up to four color keys. When the pixel component is equal to a range register value or is within the range, the result of the comparison for that byte lane is true (1). If it’s outside the range, the result is 0. The results from the three byte lanes are used as keys in the 8-bit Color Key Combining ROP. This ROP is a mechanism used for extreme flexibility in keying and is located in the layer. (It is not to be confused with ROP in the mixer which follows each layer.)

CnKey0_Layer is determined by checking if the B(V) color component is within the chroma key range.

CnKey1_Layer is determined by checking if the G(U) color component is within the chroma key range.

CnKey2_Layer is determined by checking if the R(Y) color component is within the chroma key range.

Cn represents one of the four possible key colors. Since four colors are possible per layer, four different ROPs exist for determining which component of which key color should lead to a color key hit. If the color component is within the range, the key is set to 1. If it’s not, the key is set to 0.

Table 2: Color Key Combining ROPs

ROP Bit	Key2	Key1	Key0
[0]	0	0	0
[1]	0	0	1
[2]	0	1	0
[3]	0	1	1
[4]	1	0	0
[5]	1	0	1
[6]	1	1	0
[7]	1	1	1

The ROP should be programmed to enable each key combination for which one wants to trigger a “color key=true.” For each pixel in the stream, the ROP checks the keys to determine if they match one of the selected combinations. When there is a match, the result is true (1). If there is no match, the result is false (0). The result of the Color Key Combining ROPs is combined in the mixer and used as a key (Key1) in the Invert, Pixel Select, Alpha Blend Select and Pass Through Key ROPs.

The examples in Table 3 describe how to program the ROP for various results.

Table 3: Chroma Key ROP Examples

ROP	Description
0xFF	All ROP bits are enabled, so all of the key combinations are included. One of them must be true; therefore, the result will always be true (chroma key=1).
0x00	None of the ROP bits is enabled, so the ROP will never get a match. The result will always be false (chroma key=0).
0x80	Bit 7 is enabled. This is the combination where all three keys=1. The result will be true only when all the pixel’s YUV (RGB) are in the YUV (RGB) chroma key range.
0xF0	Bits 7,6,5 and 4 are enabled. These are the combinations where Key2=1. For any pixel where the Y(R) component (Key2) matches the chroma key range, the ROP result will be 1.
0xE8	Bits 7, 6, 5, and 3 are enabled (11101000=E8). These are the combinations where at least two of the keys are true. In this case, the ROP will return true whenever any two of the color components match their respective chroma key range.

The result of the four ROPs within a layer is fed into the mixer associated with the layer. The four keys are called: Current_Color_Key_0-3.

UDTH

UDTH is the undither unit that is used to recover 9 bits of precision from an 8-bit dithered data. It is the reverse of the dither operation in the VIP as an 8-bit-wide video is usually not very practical (since some head room is lost because there is not a good automatic gain control). For quality reasons, therefore, one has to process 9 or 10-bit video.

However, if every stage in the processing chain—after its required processing—has to round to 8 bits, accumulated quantization errors (quantization noise) occurs. The local video data paths can be made wider (e.g., 10 bits), but since there are only 8-bit wide field memories, compression from 9 bits or more to 8 bits or less will have to take place. Furthermore, if pixels are blindly rounded and/or quantized to 8 bits, particularly for low-frequency (small) signals, the quantization noise is not evenly distributed but remains correlated to the input signal, and contouring effects occur.

So, what is wanted is 9-bit video quality for an 8-bit (memory) price. With this goal in mind, it is primarily to de-correlate the quantization error and also to retain some precision, that dithering is used. Dithering distributes the error across the entire spectrum simply by adding some random noise prior to quantization via a random or semi-random perturbation of the pixel values. The 8-bit values, with 9-bit precision, are stored in memory where they are fetched from and processed by the QVCP. For the part of a picture that is almost constant (or flat) in the horizontal direction, one should try to recover the full 9 bits from the stored 8 bits because the quantization error is more noticeable; However, when there are more high-frequency components, the full 9 bits cannot be recovered, but the quantization error is less visible anyway.

The UDTH unit, however, does more than just undithering. It does format conversion as well. QVCP has its own data format protocol (10-bit signed data -- UP(Y/R), MI(U/G), LO(V/B) --- and 8-bit unsigned alpha) and is designed to process data with a nominal range of 9 bits; one extra bit is reserved for overshoots and undershoots. Input data gets converted, from an external format to the native QVCP format, inside the UDTH module. Figure 3 shows the data-flow diagram of the UDTH unit. In the figure, the nominal data range is shown for each conversion step for both 8- (bottom part) and 10- (top part) bit inputs. Note that an 8-bit input to QVCP will be converted to a 9+1 format by undithering or by left shifting. A 10-bit input, however, can be either true 10 bits or 9+1 bits; for the former, there is not too much head-room left and so, the nominal range of the various layers should be equalized at the VCBM unit.

The function of Alpha Processing unit is to insert a fixed alpha or pre-multiply each pixel with a per-pixel alpha. For a 10-bit input, only fixed alpha is supported.

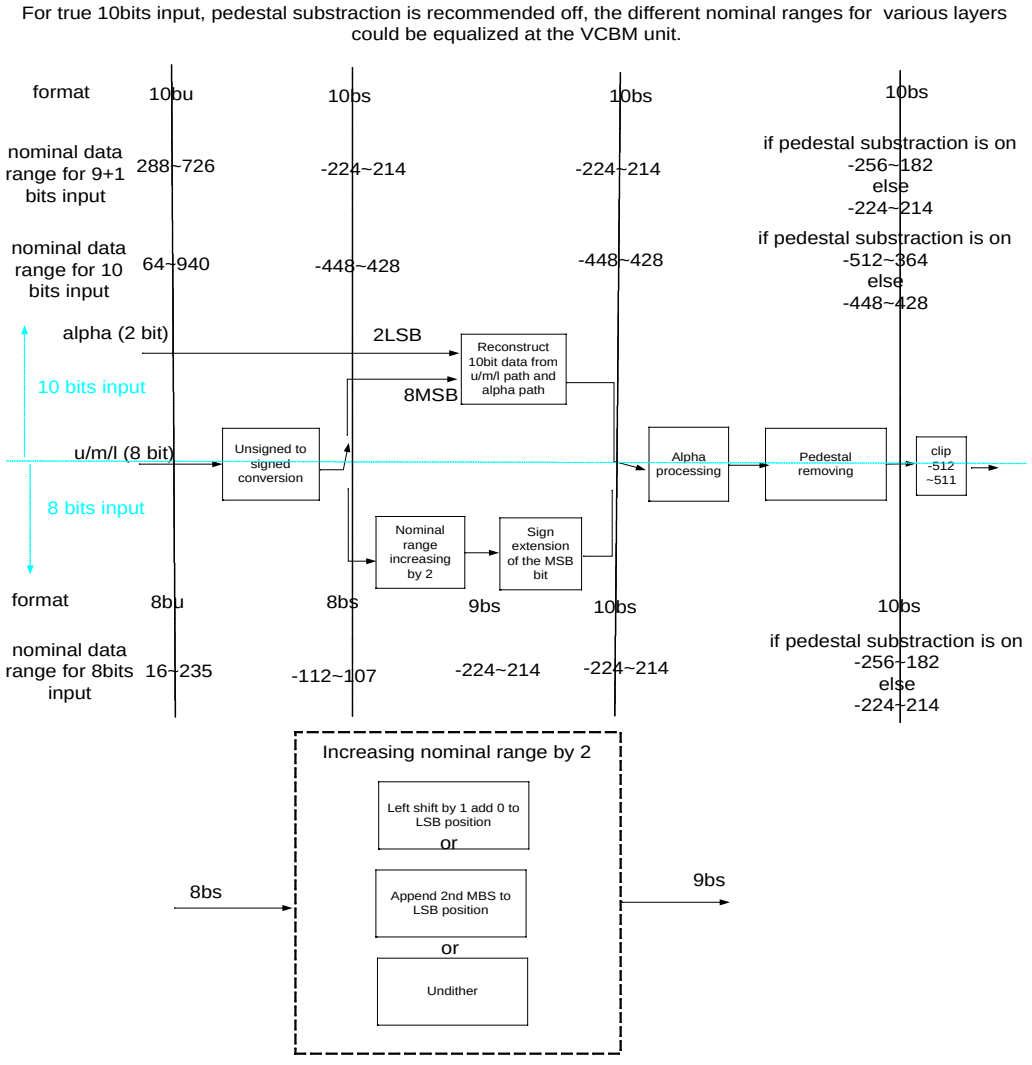


Figure 3: Undithering and Pedestal Manipulation

2.3.4 Chroma Upsample Filter (CUPS)

The chroma (chrominance) upsampling module allows conversion of YUV 4:2:2 data streams to 4:4:4 data streams by interpolating the missing chrominance information. There exist two forms of 4:2:2 co-sited and interspersed. It depends on how the U&V data was down-sampled from 4:4:4 to 4:2:2. Since both co-sited and interspersed 4:2:2 formats (as shown in Figure 4) are supported, the up-sampling method of CUPS should be adapted accordingly. This feature is necessary for supporting RGB/YUV outputs in full-color resolution.

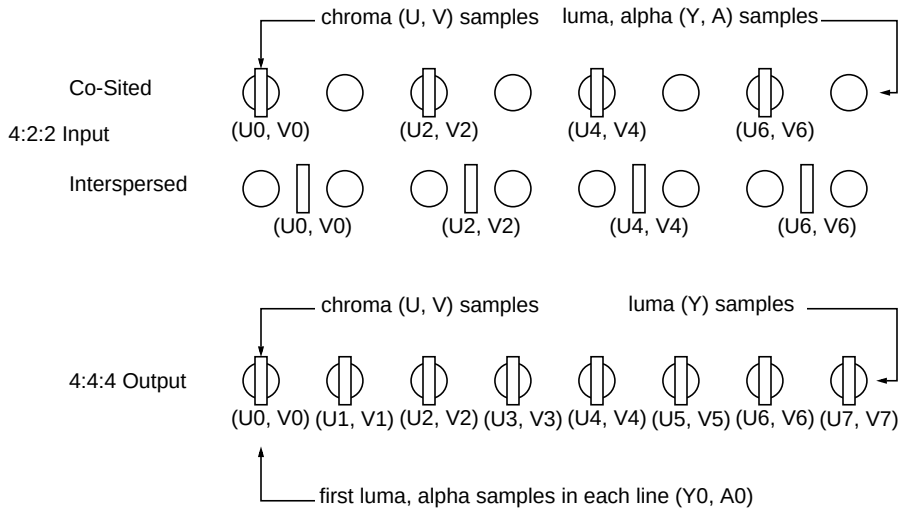


Figure 4: 4:2:2 and 4:4:4 Formats

2.3.5 Linear Interpolator (LINT)

The linear interpolator is used for horizontal up-scaling of graphics images. It is specifically used for graphics images because its nature makes it unsuitable for quality scaling of video material. However, due to its small size, the block is present in both layers of QVCP. This unit supports up-scaling only, and can handle both YUV and RGB data streams. It also supports scaling of the alpha channel as well as any potential previously-extracted color keys. The output samples are calculated from the input samples via a piece-wise linear interpolator. All pixel components are treated equally.

Remark: Layer Size (final)) register has to be updated to match the scaled width, if LINT scaling is changed.

2.3.6 Video/Graphics Contrast Brightness Matrix (VCBM)

The first purpose of VCBM is contrast (gain) and brightness (offset) control (in that order). The contrast and brightness controls are for normalizing the amplitude (black and white level) of all sources. They also permit balancing the visibility of all video and graphics layers. An important benefit of having separate controls for all video and graphics layers is that the user interface never needs to disappear, even if the user tries to make (the video part of) the picture invisible. This benefit should be achieved by control software: limit the control range for the user interface layers.

The second function of the VCBM unit is to take YUV (for video) or RGB (for graphics) inputs and produce RGB or YUV outputs. The color space conversion is effected by multiplying a 3x1column vector (with 3 components: Y, U, V or R, G, B) in the source color space with a 3x3 matrix (of coefficients) in order to obtain a 3x1 column vector in the destination color space. The programmable coefficients of the 3x3 matrix can be altered to modify contrast, saturation, and hue.

$$\begin{bmatrix} UP_{out} \\ MI_{out} \\ LO_{out} \end{bmatrix} = \begin{bmatrix} C_{00} & C_{01} & C_{02} \\ C_{10} & C_{11} & C_{12} \\ C_{20} & C_{21} & C_{22} \end{bmatrix} \times \begin{bmatrix} UP_{in} + b \\ MI_{in} + b' \\ LO_{in} + b' \end{bmatrix} \quad (10)$$

where

- UP, MI, and LO = R,G,B or Y,U,V
- Cij= the standard matrix coefficients multiplied by the desired contrast ratio
- b = the brightness value divided by the desired contrast ratio
- For YUV input b’ = 0 and for RGB input b’=b
- For RGB -> RGB, there is no support for brightness control.

Thus, the main functions performed by the VCBM unit are: contrast and brightness control and color-space conversion with white-point control; this is achieved by using one or more operations from the following sequence:

- contrast control by multiplying the 9 matrix coefficients by the same desired contrast ratio (where a ratio of 1 means a gain of unity).
- brightness control on YUV (add offset "b", as shown above, to values at the input of matrix, where b signifies the desired brightness offset scaled by the contrast gain)
- YUV-to-RGB matrix, also usable for YUV-to-YUV
- optional saturation control via the YUV-to-RGB (etc.) matrix (by adjusting the contribution of U and V by the same ratio, while keeping the contribution of Y as constant, done by modifying the six matrix coefficients for U and V.
- optional white-D control via the YUV-to-RGB (etc.) matrix
- offset addition for making unsigned U&V
- guaranteed hard-clipping to 10-bit unsigned formats

2.3.7 Layer and Fetch Control

The layer fetch control receives the global screen coordinates from the STG and takes care of extracting the pixels from a layer when needed (i.e., when the programmed position for the layer has reached). Another task (of the layer fetch control unit) is to clip layers exceeding the screen coordinates.

2.4 Pool Resources and Functions

The following sections describe the pool elements. These elements are never needed in all of the layers at the same time.

All of the pool units comprise three basic sub-modules (and so do the layer units):

Functional Unit: This is where the data is processed. It contains the data path and the logic to control the flow of data.

Register File: The register file contains the registers which are used to control the pool resource. These registers are programmed via the pbus and are read/write.

Push-Pull Interface: This unit is used to control the flow of data into and out of the pool resource. The push pull interface allow the flow of data to be stalled by and block in the video pipe. If a stall occurs then all processing of the inputs stops and all data is stored. When the stall is released then the data is processed as before.

2.4.1 CLUT (Color Look Up Table)

The resource pool contains one set of component-based color-look-up tables for each color component and for the potential alpha value of a pixel. The look-up table has a depth of 256 words, with each word being 8 bits wide.

The basic function of a CLUT is to expand indexed-color formats. An 8-bit indexed color would be applied to all component LUTs as an address, whereby each of the LUTs will provide on its data ports the previously-programmed data word belonging to that address. However, since the addresses of the LUTs are not linked together, gamma correction on a component basis is also possible.

2.4.2 DCTI (Digital Chroma/Color Transient Improvement)

The Digital Color Transient Improvement (DCTI) block improves the steepness of color transients. It is a form of delay modulation: around transients the signal is time-compressed. This is a non-linear operation, and it increases the bandwidth of the color signal. DCTI can not increase the number of transients per line (that would require real bandwidth in the signal path), it can only increase the steepness of transients that are already there.

DCTI modifies the U & V data paths. Horizontal transients are detected and enhanced (without overshoots) by shifting the color values The amount of color shift is controlled by values generated via differentiating the original signal, taking absolute values of the differential, and differentiating the absolute values once again. To prevent the third harmonic distortion, the so-called over the hill protection is applied. This prevents peak signals from being distorted.

2.4.3 HSRU (Horizontal Sample Rate Upconverter)

The main purpose of HSRU is horizontal up-sampling, where the re-sampling function obeys a third-order difference equation for the phase of the sample positions. This creates more space in the spectrum for LTI to fill. This extra room can also be used by other non-linear operations, like HIST and VCBM, so that they will create less undesired aliasing. Up-sampling is good before any non-linear operation, and all blocks behind the HSRU will run on the higher sample-rate. We can also choose to up-sample the left and right edges of the image more than the centre. This is called

"Panorama mode" or "Superwide mode", and it is recommended for showing 4:3 or 14:9 images on a 16:9 display. Besides a linear constant upscaling ratio, the HSRU also supports special non-linear upscaling ratios for the panorama mode.

Remark: Layer size(final) register has to be updated to match the scaled output size of HSRU

2.4.4 HIST (Histogram Modification) Unit

The Histogram modification block complements the Histogram measurement blocks in the MBS. It modifies the Y, U & V data according to a non-linear transfer curve. The Y transfer curve is described by the 32 values of a look up table. The color difference signals are also coupled to this curve by a calculation derived from the original Y and its value after the Histogram modification. The final aim is to provide a greater contrast by increasing the range of intensities in the input signal.

2.4.5 LSHR (Luminance/Luma Sharpening) Unit

The LSHR module is used to improve (increase) the sharpness impression. Inputs to the LSHR unit are all three of the Y, U and V signals. They are 10 bit signed values (-512 to 511 range). The LSHR unit modifies the Y component only; U and V remain untouched. Outputs of the LSHR unit are also 10-bit signed values.

The LSHR unit has a latency of 33 cycles when it is enabled. In the bypass mode, it has no latency. For proper operation, at least 7 dummy cycles are required between any two lines. The unit also performs sharpness measurements on the luma signal and the results are stored in two status register: LSHR_E_max and LSHR_E_sum. They are updated at the end of each frame.

2.4.6 Color Features (CFTR) Unit

The Color Features block performs a sequential combination of three functions:

- Skin Tone Correction
- Blue Stretch
- Green Enhancement

These features are intended to correct the errors caused by the transmission and the phosphors used in CRT displays. The reason that skin tone, blue-stretch (alters the white temperature) and green enhancement are chosen is that the human eye is most sensitive to unnatural errors in these colors. In practice, they can also be used to enhance the vividness of certain colors in the picture. Skin tone correction will be useful to compensate for a small phase error in the demodulation of analog NTSC (hue error). Blue stretch and green enhancement just look nice.

Remark: It may not be fully correct to state above that the color features are intended to correct the errors caused by the transmission and the phosphors used in CRT displays. The real cause of the tint problem may well be that people are trying to correct for an error that no longer exists. The current practice in NTSC countries has been for a long time to encode for phosphors that are similar to EBU. If the TV makes no corrections for presumed original NTSC phosphors, then there will be no error.

2.4.7 PLAN (Semi Planar DMA) Unit

This pool element contains one DMA channel which can be independently assigned to any layer. By default, this DMA channels is assigned to the first layer. The DMA channel is meant for fetching UV data in parallel to the fetching of Y-data by the DMA unit that is already present inside each layer.

2.5 Screen Timing Generator

The Screen Timing Generator (STG) creates the required synchronization signals for the monitor or other display devices. The screen timing generator usually operates as the timing master in the system. However, it is also possible to synchronize the operation of the screen timing generator to external events i.e., a vertical synchronization signal. The screen timing generator also defines the active display region. The coordinate system for the STG is (x, y), with (0, 0) referring to the top left of the screen. The coordinate (Horizontal Total, Vertical Total) defines the bottom right of the screen. Horizontal and vertical blanking intervals, synchronization signals, and the visible display are within these boundaries.

Some of the control parameters that need to be set for the screen timing are:

- HTOTAL = Total no. of pixels per line minus one
- VTOTAL = Total no. of lines per field minus one
- HSYNCS/E = Start/End pixel position of horizontal sync (Hsync)
- VSYNCS/E = Start/End line position of vertical sync (Vsync)
- HBLNKS/E = Start/End pixel position of horizontal blanking interval
- VBLNKS/E = Start/End line position of vertical blanking interval

The following rules apply to the register settings specifying the screen timing using the above control parameters:

- total number of pixel per line: HTOTAL + 1
- total number of lines per field: VTOTAL + 1
- 0,1 < HBLNKS <=HTOTAL
- 0,1 < HSYNCS <= HTOTAL
- 0 < VBLNKS <= VTOTAL
- 0 < VSYNCS <= VTOTAL
- Hsync - must be asserted or negated for at least one clock
- Vsync - must be asserted or negated for at least one scanline
- Hblank - must be asserted or negated for at least two clocks
- Vblank - must be asserted or negated for at least two scanlines

The state change of the odd_even signal is always tied to the rising edge of the vsync signal. Figure 14 identifies screen display parameters controlled by fields in the STG registers

Other restrictions for the screen timing generation are as follows:

invalid HSYNCS/E settings: 0, 1, 2 > HTOTAL

invalid HBLNKS/E settings: 0, 1, > HTOTAL

invalid VSYNCS/E settings: 0, > VTOTAL

invalid VBLNKS/E settings: 0, > VTOTAL

invalid difference HSYNCE-HSYNCS: -1

invalid difference HBLNKE-HBLNKS: 0, -1, -2

In interlaced modes these differences are not allowed:1, 2, 3, 4 to guarantee sufficiently long horizontal blanking:

invalid difference VSYNCE-VSYNCS: -1

invalid difference VBLNKE-VBLNKS: 0, -1, -2

invalid difference VBLNKE-VBLNKS: 0, -1, -2

2.6 Mixer Structure

The properly formatted pixels from each layer are combined in a cascaded series of mixer units. There is one mixer unit associated with each layer unit within the QVCP. For a given screen position, each mixing unit can select the pixel from the layer, the pixel from the previous mixer, or a blend of the two pixels. If a layer does not generate a valid pixel for a specific screen position, then the mixer will pass the pixel from the previous mixer. If no layers are producing valid pixels, a background color will be displayed. The mixer selection criteria are based on a number of functions ROPs that can be used to create such common effects as color keying. There are no restrictions on window size, position, or overlap. A mode such as PIP is simple to implement by setting layer_N for full screen video and layer_N+1 to the PIP. PIP size and position may be changed on a frame by frame basis. Effects such as blending a PIP in and out of the full screen video are easy to achieve using the 256 level alpha blend capability of each mixer.

The main functionality of the mixer stages is to compose the outgoing pixel streams from each layer to the final display image. The mixer data path operates on 10 bits. This includes clipping, alpha blending, inverting colors. Which of those functions is applied and how, is defined in a set of raster operations (ROPs). A raster operation is always a logical combination of several input keys and a specific ROP register which enables one or more of the different key combinations. It (ROP operation) is like a function that generates the output based on a logical combination of several input signals and the programmable ROP register.

Each mixer knows 4 different keys (Key0, Key1, Key2, Key3) as illustrated in the mixer block diagram.

- Key0, output key from previous layer, KeyPass ROP

- Key1, (current color/chroma key pixel key OR new pixel key 1)
- Key2 (New Pixel Key 2)
- Key3 color key of the previous pixel

Since each layer can have four color keys, the output is a 4-bit vector specifying which keys match the pixel. This 4-bit vector is masked by the ColorKeyMASK. The result is Key1. For Key3 the procedure is similar, with the only difference being the color key vector is passed from the previous mixer. The result of masking the color key vector with the ColorKeyMask register is Key3. However, one can do selective color keying for the current pixel. A ColorKeyROP specifies whether color keying is performed on the current layer pixel or not. Inputs for this ROP are Key0,1,2, 3.

The ROP block decides if a certain operation is done on the pixel or not.

The outputs of the Select, Alpha, Invert, Key_Pass and Alpha_pass ROPs are based on Raster operations as shown in Table 4.

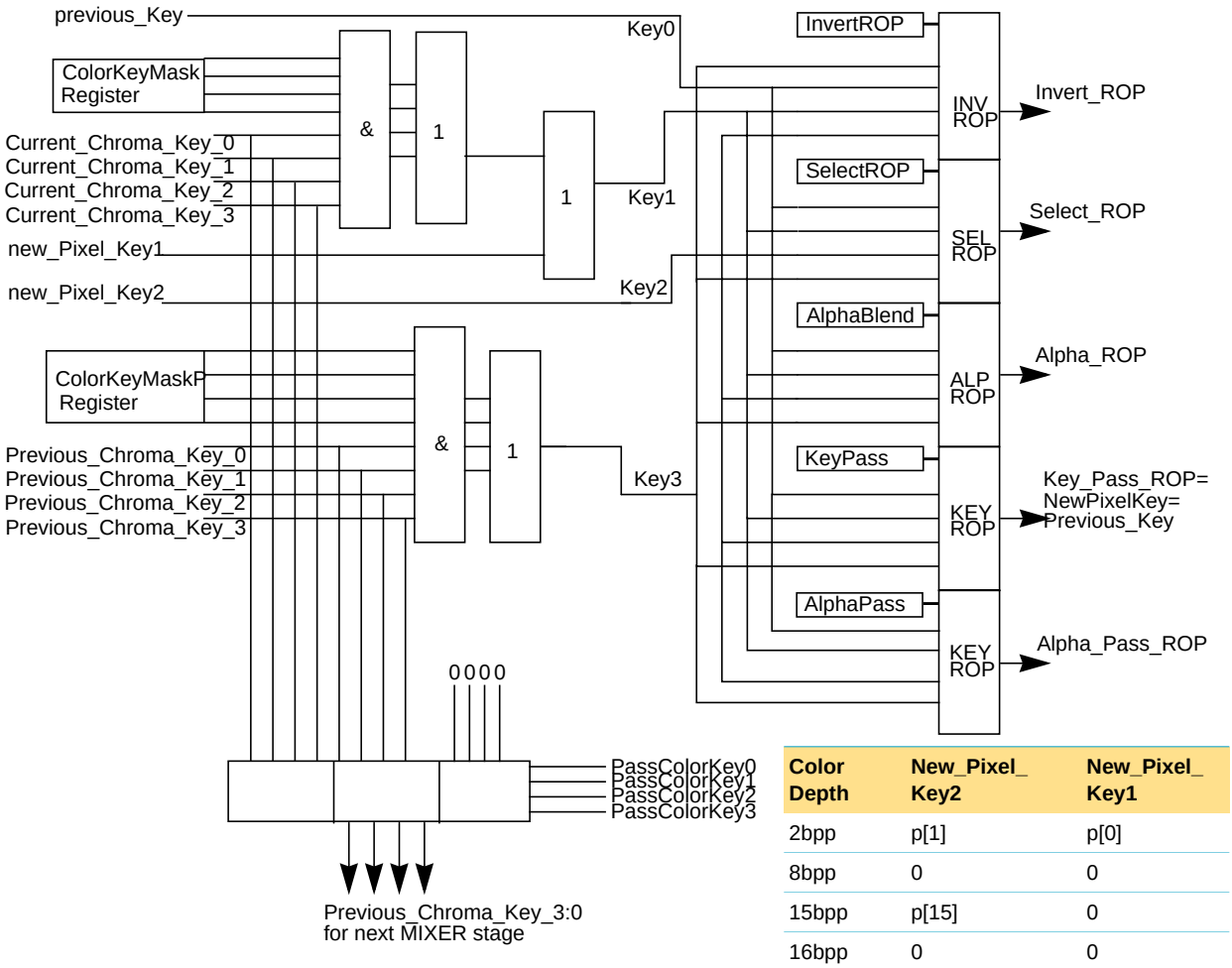
Table 4: ROP Table for Invert/Select/Alpha/KeyPass/AlphaPass ROPs

ROP BIT	Key3	Key2	Key1	Key0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

Figure 5 illustrates the Mixer function. The upper part shows the generation of the signals which are used to control the actual pixel manipulation functions shown in Table 4.

Remark: For mixer 1 there is no previous mixer and therefore the corresponding inputs are set to 0.

2.6.1 Key Generation



PassColorKey0-3 register settings

00 pass zeros to the next mixer
01 pass current color key to next mixer
10 pass previous color key to next mixer
00 reserved

Color Depth	New_Pixel_ Key2	New_Pixel_ Key1
2bpp	p[1]	p[0]
8bpp	0	0
15bpp	p[15]	0
16bpp	0	0
24bpp	0	0
32bpp	p[31:24] && Pixel KeyAnd[31:24]	0

Note: The generation of the new_Pixel_Key happens in the pixel formatter block, not in the mixer.

Figure 5: Mixer Block Diagram—Pixel Selection

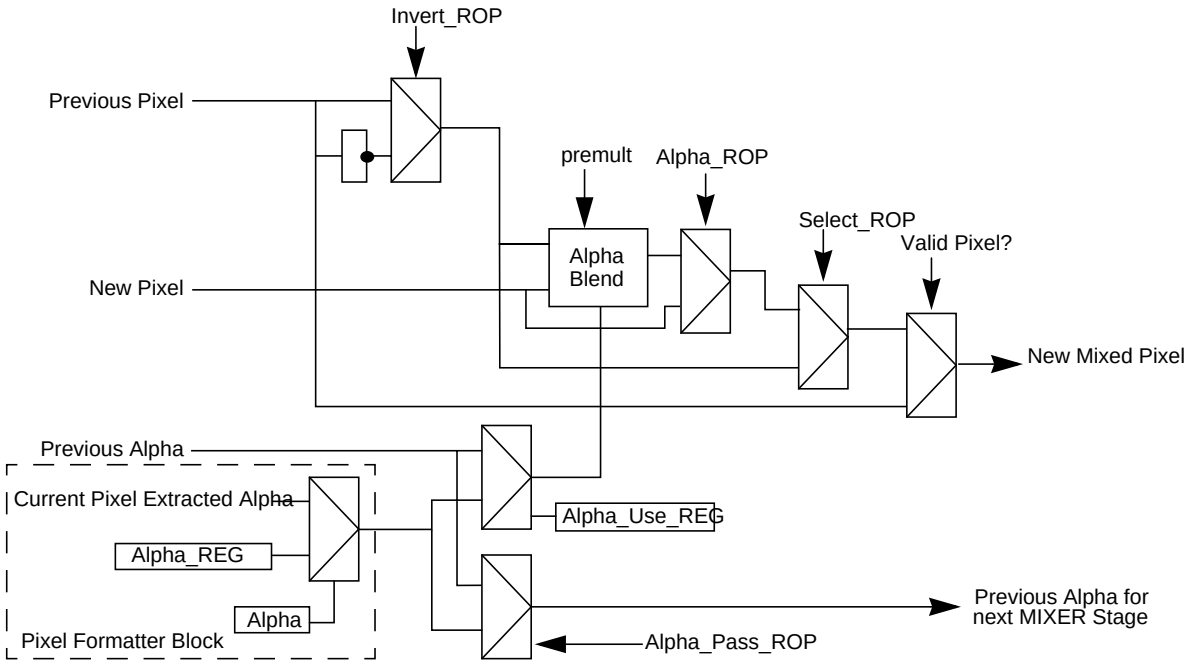


Figure 6: Mixer Block Diagram—Pixel Processing

2.6.2 Alpha Blending

Blending allows video and graphics to be combined with varying levels of transparency. Blending is possible only when both current and previous Layer pixels are valid. Either 16 or 256 levels of blend from one layer to another and vice versa are available. The blend value may come from a layer's alpha register or from the upper 4 or 8 bits of an incoming pixel or is a multiplication of both.

The blending is done according to the following equation:

```
Pixel_result = alpha x Pixel_current + (1-alpha) x Pixel_previous
```

Pre-multiplied pixel formats are supported. The Premult bit is set, which means the incoming pixel stream is already pre-multiplied with the per-pixel alpha value. The resulting alpha blend equation is as follows:

```
Pixel_result = Pixel_current + (1-alpha) x Pixel_previous
```

An additional per-component pre-multiply with a constant can be achieved by proper programming of the color space matrix. Fading of alpha values is controlled by the alpha_mix bit. If it is set, the pixel alpha gets multiplied by the fixed alpha value/256.

Remark: Alpha=255 has the effect, in hardware, of making alpha equal to 1 in the above equations.

2.7 Output Pipeline Structure

The input to the output pipeline comes from the mixers. The output pipeline houses the formatter (FRMT) that produces the final output stream in the required output format.

Besides two instances of the formatter, the output-interface unit also contains two instances of a chroma-downsampling unit (CDNS), one instance of a gamma-correction and noise-shaping unit (GNSH) and several input-output muxes; the gamma-lookup-table allows possible gamma-correction on the final composited image stream, whereas the noise shaper logic reduces (dithers) the number of bits per pixel (in the gamma-corrected image) via error propagation. The insertion of VBI data into the D1 or D1-like output streams is also supported. In order to support 4:2:2 output formats, two chroma-down-sample filters (CDNS) are included at the beginning of the input-data chain. The input multiplexer (IMUX) is used to appropriately select the input stream depending on the partitioned layer boundary. The interleaving unit (INTL) is to be programmed in a pass-through mode.

2.7.1 Supported Output Formats

The output formatter supports the following output formats:

- 30-, 24- or 18-bit parallel YUV or RGB + external H- and V-sync and composite blank
- 10- or 8-bit D1 or D1-like 4:2:2 YUV
- 10- or 8-bit D1-like 4:4:4 YUV/RGB
- 20- or 16-bit double-interface semi-planar 4:2:2 D1 mode with 10- or 8-bit Y and 10- or 8-bit U/V multiplexed data

Remark: The PNX15xx Series digital video interface has assigned up to 30 data pins to the video output interface. Refer to Chapter 3 System On Chip Resources for the different pin assignment.

2.7.2 Layer Selection

In the Mixers, the final image (to be displayed) is composed (composited) from the images obtained from the various layers.

2.7.3 Chrominance Downsampling (CDNS)

Chroma down-sampling is necessary for creating a YUV 4:2:2 output signal. It uses a (1,2,1)/4 low-pass filter to create co-sited U&V samples (because ITU-R.601 specifies only co-sited sub-sampling). Every second U&V output sample is discarded, but then the other sample is repeated. Consequently, the output stream is still 4:4:4 and the repeated samples have to be discarded later.

2.7.4 Gamma Correction and Noise Shaping (GNSH& ONSH)

The gamma-lookup-table allows possible gamma-correction on the final composited image stream, whereas the noise shaper logic reduces (dithers) the number of bits per pixel (in the gamma-corrected image) via error propagation.

In the GNSH unit, the QVCP-internal data format is converted into the desired output format. The overshoots and undershoots which are generated by the QVCP layer units are also removed if noise shaping is off and the desired output is not the 9+1 mode; the data will be left-shifted by one bit and the MSB bit will be lost.

The main purpose of gamma correction is to adapt the gamma prescribed by the transmission standard to a particular display device. Gamma may also be adapted to ambient light conditions (a lower gamma for a brighter environment). Gamma correction should be done on RGB signals going to the display, but never on a YUV signal.

The gamma correction in QVCP is based on linear interpolation.The input signal is divided into 64 segments of 16 values each. For each component, there are 2 independent look-up tables, one for the base and the other for the slope. For a strong correction, there is an optional squarer after the linear interpolator, which is recommended to be switched on when the gamma value is larger than about 1.4. The output of the gamma corrector is a 14b unsigned value.

QVCP supports four output data formats: 6 bits, 8 bits, 9+1 bits, and 10 bits. They are generated by the dither unit. The dither unit uses an error-propagation algorithm, which minimize the average error caused by losing bits.

2.7.5 Output Interface Modes

The supported output interface modes are described below. Note that Chapter 3 System On Chip Resources presents how the QVCP module pins are mapped to the PNX15xx Series pins.

30-, 24- or 18-Bit Parallel Mode

All video data pins are used to transport the digital video data stream without any component multiplexing. The output data format can be either YUV or RGB and 10-, 8-, or 6-bit (as determined by the noise shaping) per color component.

30-bit Mode:

QVCP_DATA[29:20]: Y[9:0] or R[9:0]

QVCP_DATA[19:10]: U[9:0] or G[9:0]

QVCP_DATA[9:0]: V[9:0] or B[9:0]

24-bit Mode:

QVCP_DATA[29:22]: Y[7:0] or R[7:0]

QVCP_DATA[19:12]: U[7:0] or G[7:0]

QVCP_DATA[9:2]: V[7:0] or B[7:0]

18-bit Mode:

QVCP_DATA[29:24]: Y[5:0] or R[5:0]

QVCP_DATA[19:14]: U[5:0] or G[5:0]

QVCP_DATA[9:4]: V[5:0] or B[5:0]

D1 Mode

10-bit Mode:

- QVCP_DATA[9:0]: image stream with multiplexed components

8-bit Mode:

- QVCP_DATA[9:2]: image stream with multiplexed components

6-bit Mode:

- QVCP_DATA[9:4]: image stream with multiplexed components

Double D1 Mode

In this mode, twenty video data pins are used out of the 30. These twenty pins are used to stream out one video data stream. QVCP outputs YUV 4:2:2 by splitting Y and UV into two separate streams, Y uses 10 pins and the interleaved UV uses the other 10 pins. The data stream can be generated by splitting up the QVCP into two sets of layers.

10-bit Mode:

- QVCP_DATA_OUT[19:10]: UV component of image stream
- QVCP_DATA_OUT[9:0]: Y component of image stream

8-bit Mode:

- QVCP_DATA_OUT[19:12]: UV component of image stream
- QVCP_DATA_OUT[9:2]: Y component of image stream

6-bit Mode:

- QVCP_DATA_OUT[19:14]: UV component of image stream
- QVCP_DATA_OUT[9:4]: Y component of image stream

2.7.6 Auxiliary Pins

QVCP has two auxiliary (AUX) output pins, each of which can be independently programmed for:

1. Composite blanking, where the value on the pin is asserted HIGH (1) if either Vblanking is TRUE or Hblanking is TRUE or both are TRUE. The complement of the value on the pin can, therefore, be used as the indicator of valid/active pixels.
2. Odd/even indication, where the field polarity is indicated in the interlaced mode. The value on the pin is 0 in progressive mode.
3. Video/graphics indication, where the color keyn (n = 1, 2, 3, 4) of the mixer 2 is used. The corresponding color key can serve as video/graphics indicator.

The two auxiliary are referenced as QVCP_AUX (QVCP auxiliary 1) and VDO_AUX (QVCP auxiliary 2) in Chapter 3 System On Chip Resources.

3. Programming and Resource Assignment

3.1 MMIO and Task Based Programming

In order for the QVCP to function properly its various block have to be configured. Each functional unit contains a set of programming registers. A more detailed description of the various registers can be found in the register description section of this document.

The registers are divided in layer specific registers and global registers. Layer specific registers are used to set up the layer related functions such as layer position, size, pixel format and various conversion functions. The global register space accommodates functions such as screen timing and output format related functions. Another important part of the global register space are the resource assignment registers which allow to assign the pool resources to specific layers.

There are two ways to access the QVCP registers:

1. The first and primary way to get read/write access to the registers is via the MMIO bus, which maps the registers into the overall PNX15xx Series address space.
2. The second way to get write-only access to the registers is via data structures fetched through the VBI DMA access port (used to fetch VBI data which get inserted into the output data stream). Differentiation between VBI and programming data is accomplished via a different header.

The data structure to be used contains a header consisting of a pointer to the next packet in memory. A null pointer indicates the last packet in a linked list. The header also contains a field ID field which allows field synchronized insertion of VBI or re-programming packets. Packet insertion can cause an interrupt if the appropriate header flag is set. A detailed view of the packet format can be found in Figure 7.

Each data packet consists of an 8-byte descriptor followed by data (see Table 5.)

Table 5: Data Packet Descriptor

Bit	Description
12:0	Data byte count
13	Unused
14	1=wait for proper vertical field 0=send data on current field without considering the field ID (for a series of packets to be inserted in the same field, this bit should only be set for the first packet and not for subsequent ones. If this bit is set for all packets, they will be inserted with one field delay each).
15	1=generate interrupt when this packet is transmitted 0=don’t generate packet interrupt
27:16	Screen line in which to insert the data packet 0=first line after rising edge of VSYNC 0xFFF=line compare disabled. The packet is inserted without consideration of the line counter.
30:28	Field ID for this packet to be sent on

Table 5: Data Packet Descriptor *...Continued*

Bit	Description
31	Data type 0=VBI data 1=register reprogram data for internal QVCP registers
59:32	Next packet address
63:60	Unused
61:...	Data if bit 31=0, the data block consists of byte VBI data if bit 31=1, each qword in the data portion is: 15:0 QVCP register address 31:16 unused 63:32 register data

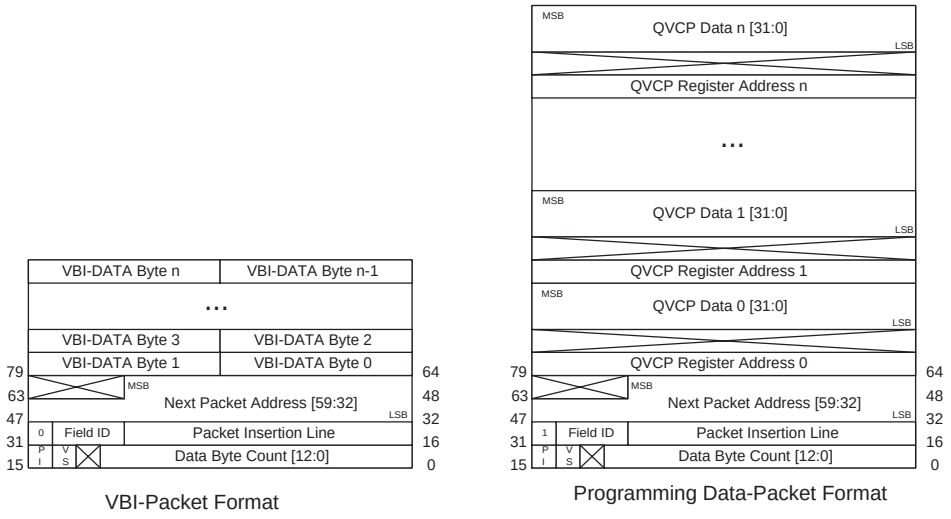


Figure 7: VBI/Programming Data Packet Formats

3.2 Setup Order for the QVCP

The following order is recommended for setting up the QVCP for a particular display scenario:

- The screen timing generator setup should be performed first since this is usually fixed for the target display. Once the screen timing is set up the screen timing generator should still stay disabled until all other settings are complete.
- In a second step the resource assignment for a particular display scenario should be set up. This involves two sub-steps:
  - Each functional unit, whether it is located inside a layer or in the resource pool, should be assigned to an aperture slot in the overall QVCP aperture.

– Once the aperture assignment has been determined, a matching resource assignment to the data path has to be performed. This assures that the data flow through the QVCP is switched through the proper resources assigned to a specific layer and a specific-layer aperture. For details and an example about resource and aperture allocation see Section 3.3 on page 11-30.

After completing the pool resource allocation and assigning each functional unit a spot in the QVCP aperture map, the layer-specific functions can be configured. If a pool resource has been assigned to a layer, its programming registers will occupy a spot in this layer's aperture map. For a layer without pool resource usage, the particular spot in the aperture map will stay empty making sure that there is symmetry in the programming register location for all the layers. If writes are performed to an unoccupied spot they will be discarded. Reads will return zero.

Once all layer specific functions are set up, the output interface needs to be programmed in order to correctly interface with the display controller chip. After performing all these tasks the screen timing generator may be enabled. Once running, the layers needed for the specific display scenario can be enabled. This concludes the QVCP setup. Once the QVCP is set up for a certain scenario and images are displayed, a number of operations can be reconfigured on the fly. Among those functions are layer size and position, alpha blending and mixing functions as well as color key and various other features.

3.2.1 Shadow Registers

Whenever any picture setting needs to be changed, it is always a good idea to make it a seamless transition i.e., no noticeable artifacts should be observed by the general audience. For most use cases, the goal is to change settings in between fields/frames or during non-active video lines (e.g., VBI).

The QVCP provides two mechanisms (programmable/selectable via a register bit) for register shadowing, whereby certain registers are shadowed to prevent screen artifacts during the reconfiguration operations.

One method allows all new setting changes to really take effect at any line location assigned by user. By using a duplicated set of the acting registers — the shadow registers as they are commonly called — any register changes will first get buffered, will wait for the correct time (i.e., the programmed line is the current line being processed), and then be passed to acting registers. The contents of the shadow registers are transferred to the corresponding active registers at the line location indicated by 0x10E1F0[11:0]. The user has R/W access only to the shadow registers, but not the active registers.

Besides a trigger from the line location indicated by the register at 0x10E1F0[11:0], the second method comprises shadowing on a positive edge of the layer-enable signal (i.e., when a disabled layer is enabled). The "positive edge" of layer enable implies "when the layer_enable register changes from 0->1". However, this positive edge triggers only the shadow registers within that specific layer, all other layers' shadow registers are not affected.

In conclusion, a shadow register transfers its content to an active register at

the positive edge of layer-enable, or

when the output line number is equal to the line number specified in the register at 0x10E1F0[11:0]

An unwanted situation can arise when shadowing starts (as a result of the trigger) before the user has finished programming a complete sequence of register changes. To prevent this from happening, the complete register reprogramming must be followed by a “FINISH” which should really trigger the shadowing. The “FINISH” is activated via rewriting a value of “1” to the LayerN_Enable, which originally has value “1”. This is sometimes called rehitting the LayerN_Enable bit. By making use of this mechanism, any “UNFINISHED” programming will only be ready for shadowing when the LayerN_Enable is rehit. Figure 13 below depicts the intended programming procedure for QVCP.

Remark: Once the shadowing is complete, the Layer upload bit is set again.

Figure 8 and Figure 9 illustrate the shadowing procedure.

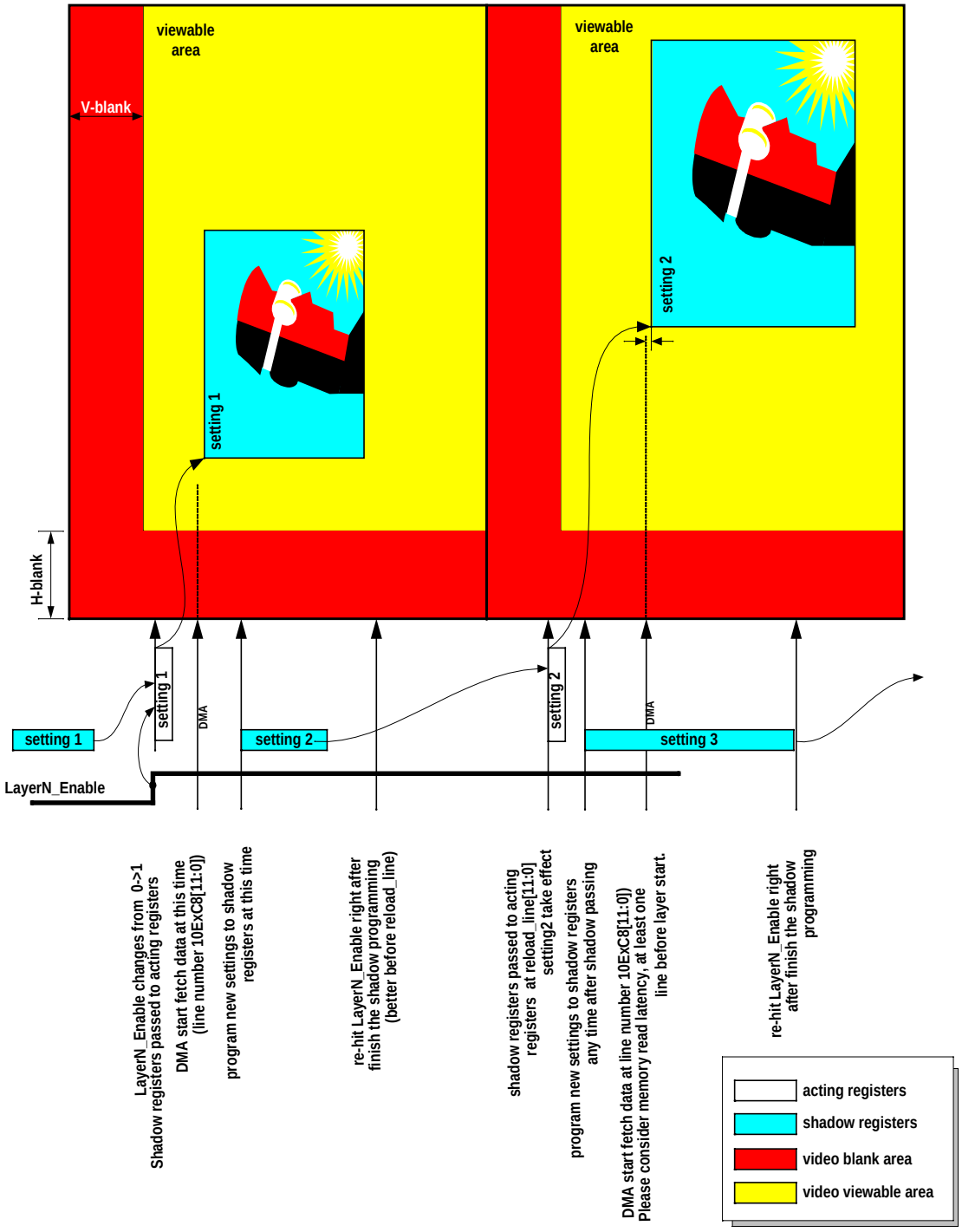


Figure 8: Shadow Mechanism

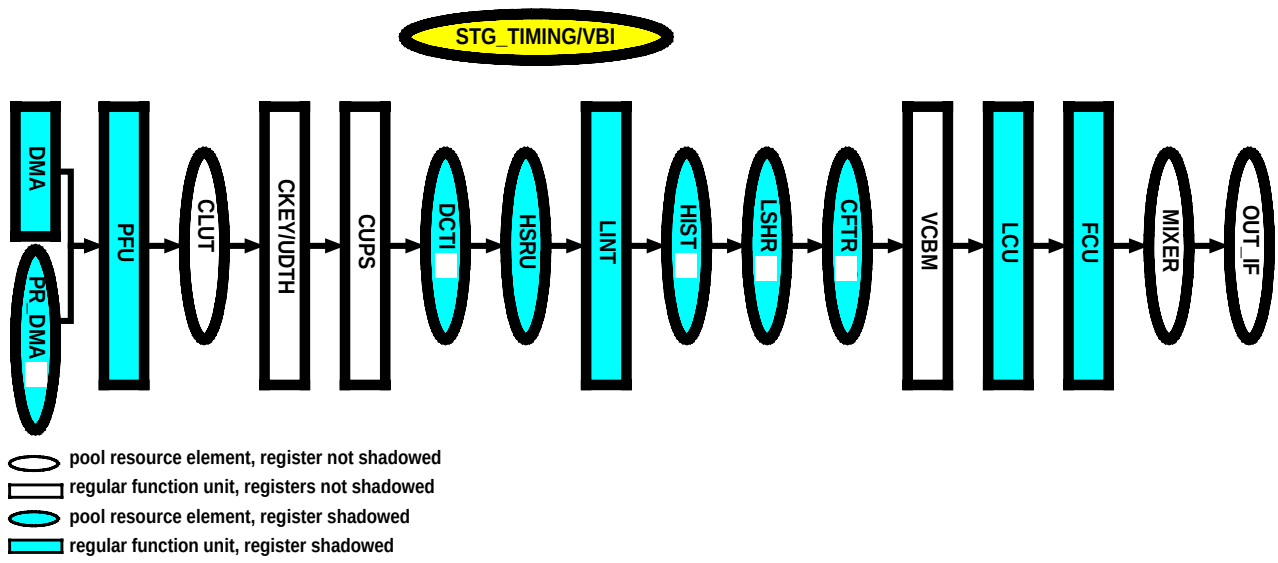


Figure 9: Shadowing of Registers

Table 6 lists all shadowed registers (where LAPT stands for Layer APerTure):

Table 6: Shadow Registers

Register	Used by
Dummy Pixel Count (0x10E[LAPT]14)	Pixel formatter
Layer Size (0x10E[LAPT]34)	Pixel formatter
Pixel Key AND Register (0x10E[LAPT]4C)	Pixel formatter
Output and Alpha manipulation (0x10E[LAPT]B8)	Pixel formatter
Formats (0x10E[LAPT]BC)	Pixel formatter
Variable Format register (0x10E[LAPT]C4)	Pixel formatter
Layer Source Address A (0x10E[LAPT]00)	DMA
Layer Pitch A (0x10E[LAPT]04)	DMA
Layer Source Width (0x10E[LAPT]08)	DMA
Layer Source Address B (0x10E[LAPT]0C)	DMA
Layer Pitch B (0x10E[LAPT]10)	DMA
Dummy Pixel Count (0x10E[LAPT]14)	Pixel formatter
Layer Start (0x10E[LAPT]30)	Layer Control
Layer Size (0x10E[LAPT]34)	Layer Control
Layer Pixel Processing (0x10E[LAPT]3C) (except bits 0 and 1)	Pixel Formatter
INTR (0x10E[LAPT]A8)	Linear Interpolator
HSRU Phase (0x10E[LAPT]AC)	HSRU
HSRU Delta Phase (0x10E[LAPT]B0)	HSRU
Layer Size (final) (0x10E[LAPT]B4)	Layer Control / Scalers

Table 6: Shadow Registers …Continued

Register	Used by
Output and Alpha manipulation (0x10E[LAPT]B8)	Pixel formatter
Formats (0x10E[LAPT]BC)	Pixel formatter
Variable Format Register (0x10E[LAPT]C4)	Pixel formatter
Start Fetch (0x10E[LAPT]C8)	Pixel formatter
LSHR_PAR_0 (0x10E[LAPT]E8)	LSHR
LSHR_PAR_1 (0x10E[LAPT]EC)	LSHR
LSHR_PAR_2 (0x10E[LAPT]F0)	LSHR
LSHR_PAR_3 (0x10E[LAPT]F4)	LSHR
LSHR_E_max (0x10E[LAPT]F8)	LSHR
LSHR_E_sum (0x10E[LAPT]FC)	LSHR
LUT-HIST (0x10E[LAPT]124)	HIST
LUT-HIST (0x10E[LAPT]128)	HIST
LUT-HIST (0x10E[LAPT]12C)	HIST
LUT-HIST (0x10E[LAPT]130)	HIST
LUT-HIST (0x10E[LAPT]134)	HIST
LUT-HIST (0x10E[LAPT]138)	HIST
LUT-HIST (0x10E[LAPT]13C)	HIST
LUT-HIST (0x10E[LAPT]140)	HIST
Layer Histogram control(0x10E[LAPT]144) (enable bit only)	HIST
Layer CFTR blue (0x10E[LAPT]48)	CFTR
Layer CFTR green (0x10E[LAPT]4C)	CFTR
Layer DCTI control(0x10E[LAPT]50) (enable bit only)	DCTI
Layer DCTI control(0x10E[LAPT]50) (enable bit only)	DCTI

3.2.2 Fast Access Registers

The architecture of the QVCP MMIO access results in module dependent latencies for the various configuration registers. For most of the registers this does not present a problem since their content is usually rather static or only updated once per field/frame. Some registers, however require access with relatively low latency. Table 7 lists the QVCP configuration registers which can be accessed with low latency.

Table 7: Fast Access Registers

Register
Field_Info (0x10 E1F8)
XY_Position (0x10 E1FC)
Interrupt Status (0x10 EFE0)
Interrupt Enable (0x10 EFE4)
Interrupt Clear (0x10 EFE8)

Table 7: Fast Access Registers …Continued

Register
Interrupt Set (0x10 EFEC)
Powerdown (0x10 EFF4)
Module_ID (0x10 EFFC)

3.3 Programming of Layer and Pool Resources

This section describes in detail the resource pool concept and the aperture assignment for pool and non-pool resources. A resource in general is a functional unit which performs a certain independent task in the video display chain of the QVCP.

3.3.1 Resource Assignment and Selection

If no pool resources are used, the data flow for a single image surface through the QVCP is strictly horizontal i.e., the pixel stream flows through the layer and does not leave it. The pool resources are assigned by default to one of the layers. However the pool resources are bypassed by default, which results in all layers becoming identical in their function.

To assign a pool resource to a different layer it requires the following:

- Ensure that the resource shows up in the assigned layer aperture.
- Ensure that the pixel data stream of the particular layer is directed through the selected resource.

3.3.2 Aperture Assignment

Each functional unit (resource) used in a QVCP layer has a unique identifier. It is used to control the assignment of this resource to a specific QVCP layer aperture location.

Table 8 lists the resource ID assignment for the functional units currently present in the QVCP. A 32-bit identifier is used for the resource ID which allows for the addition of functional units in future derivatives.

Table 8: Resource ID Assignment

ID	Functional Unit
1	PFU (Pixel Formatter Unit)
2	CKEY (Color Key and Undither Unit)
3	CUPS (Color Upsampling Unit)
4	LINT (Linear Interpolator)
5	VCBM (Video Contrast Brightness Matrix)
7	LCU (Layer Control Unit)
8	DMA (Control Unit)
9	CLUT (Color Look Up Table Unit)
10	HSRU (Horizontal Sample Rate Converter)
11	LSHR (Luminance Sharpening Unit)
12	HIST (Histogram Modification Unit)

Table 8: Resource ID Assignment *...Continued*

ID	Functional Unit
13	CFTR (Color Features)
14	DCTI (Dynamic Color Transient Improvement)
15	PLAN (Semi Planar Channel)

The location of a layer in the overall QVCP address map is shown in Table 9.

Table 9: Register Space Allocation

Address Range	Function
0x0H - 0x1FFH	Global QVCP register space
0x200H - 0x3FFH	Layer 1 register space
0x400H - 0x5FFH	Layer 2 register space

Each functional unit which belongs to a layer occupies a fixed spot within the layer address range of 0x200H bytes. However the layer assignment of this functional unit is programmable. It should usually follow the pixel data flow for a specific image surface through the functional units involved.

Two 32-bit registers, RESOURCE_ID and FU_ASSIGNMENT, are used to assign all resources to a specific layer address space. One is used to identify the resource to be assigned. The other register is split up into 4-bit chunks which contain the specific assignment for the resource identified in the first register. This allows for up to 8 resources of the same kind per functional unit. In this specific QVCP implementation only a maximum of two of the same kind of each resource is needed for non-pool resources and only one location is needed for the pooled resource. The remaining slots are reserved for future implementations. The two registers act as access points to an internal table which keeps the programmed values. All resources are programmed through the same two registers. The ID register has to be written first.

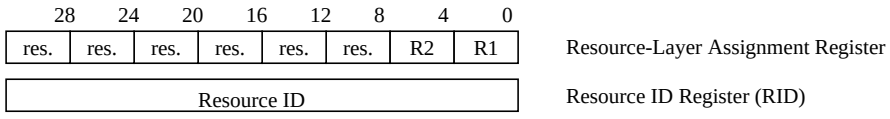


Figure 10: Resource Layer and ID

Table 10 outlines the association of a given Rn {n=0..5} value to an address space. The value of Rn {n=0..5} is equivalent to the MMIO offset bits [12:9].

Table 10: Rn Association

Rn	Address Space
0	Reserved for global QVCP addresses
1	0x200H - 0x3FFH (Layer 1)
2	0x400H - 0x5FFH (Layer 2)

The usage of address bit 12 would indicate a 8 kB aperture space for the QVCP. This is however not the case, in the current implementation only bits 11:9 are used because the QVCP occupies only a 4 kB address space. The 12'th bit is reserved for future extensions and alignment purposes.

3.3.3 Data Flow Selection

Pool resources are functional units which do not have a fixed assignment to a specific layer. Depending on the use case a resource is assigned to a specific layer.

Two 32-bit registers, POOL_RESOURCE_ID and POOL_RESOURCE_LAYER_ASSIGNMENT, are used to assign a specific resource to a layer. One is used to identify the resource to be assigned. The other register is split up into 4-bit chunks which contain the specific assignment for the resource identified in the first register. This allows for up to 8 resources of the same kind per functional unit. In this specific QVCP implementation, only two of the same kinds of pool resources are needed. The remaining 6 slots are reserved for future implementation. These two registers act in the same way as described earlier for the aperture assignment. They are used to perform the assignment for all resources, which is again stored in an internal table in which the two registers are the access point. The ID register has to be programmed first.

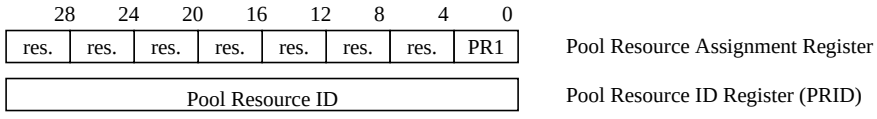


Figure 11: Resource Layer and ID

The resource layer assignment for the 2-layer, 1-pool resource scenario is shown in Figure 12.

Table 11: Resource-Layer Assignment for Pool Resource

PR1	Assignment
4'b000	Resource is assigned to layer 1.
4'b001	Resource is assigned to layer 2.

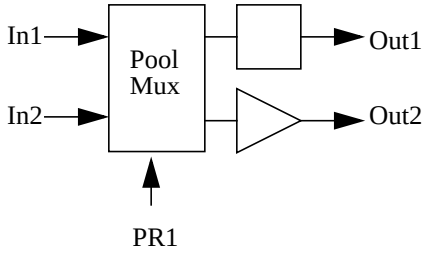


Figure 12: 2-Layer 1 Resource Elements Scenario

Note that due to resource assignment, the layers where the data stream leaves a pool element are potentially at a different layer than where it entered, hence the horizontal data flow structure no longer exists.

If for instance in Figure 12, layer 2 is configured to use resource 1, the data that will enter at In2 will leave the pool element at Out1 and the data stream entering at In1 will leave at Out2. The consequence is that all subsequent functional units will see layers 1 and 2 swapped. It should be obvious that the subsequent units will have to be reconfigured to show up in a different layer aperture.

If further down in the display pipe another swap is needed, the subsequent blocks again have to be aperture reassigned. If a configuration requires the pixel stream for a particular image surface to leave in the same layer as it entered, there is a cross-bar implementation at the far end of the layer. This allows arbitrary assignment of an input layer to a different output layer. This cross-bar can also be used to implement a “z” reordering of image surfaces.

As a general guideline for the aperture map assignment, one should assign all functional units through which a pixel data stream originating from pixel formatter N flows, to the aperture space N regardless of whether the data flow is horizontal or zig-zag due to pool resource reassignments.

3.3.4 Pool Resource Assignment Example

Figure 13 illustrates a programming example for pool and aperture reassignments.

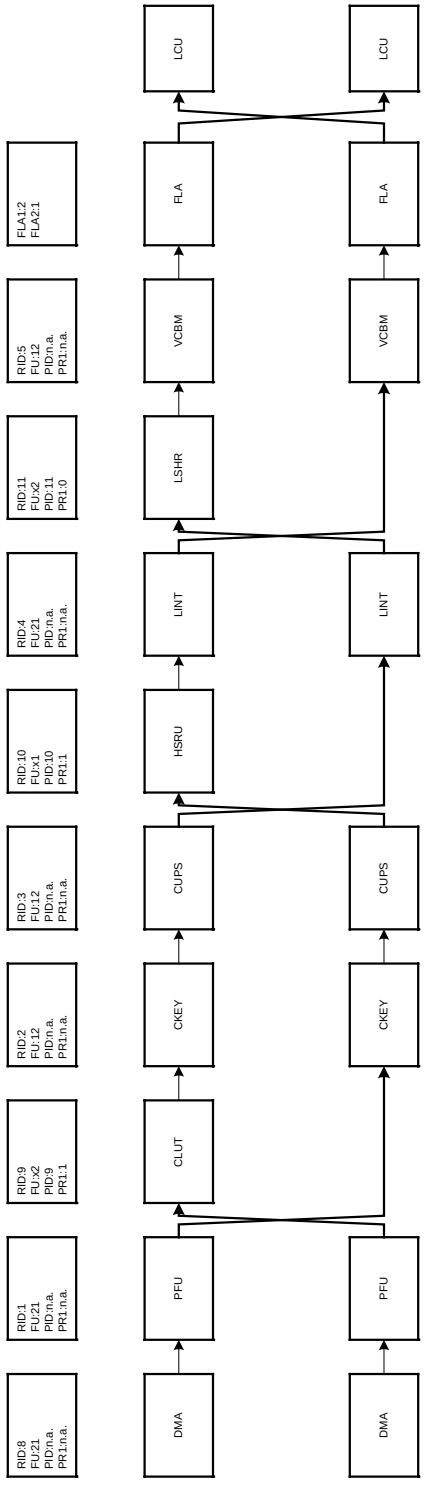


Figure 13: Pool and Aperture Reassignments

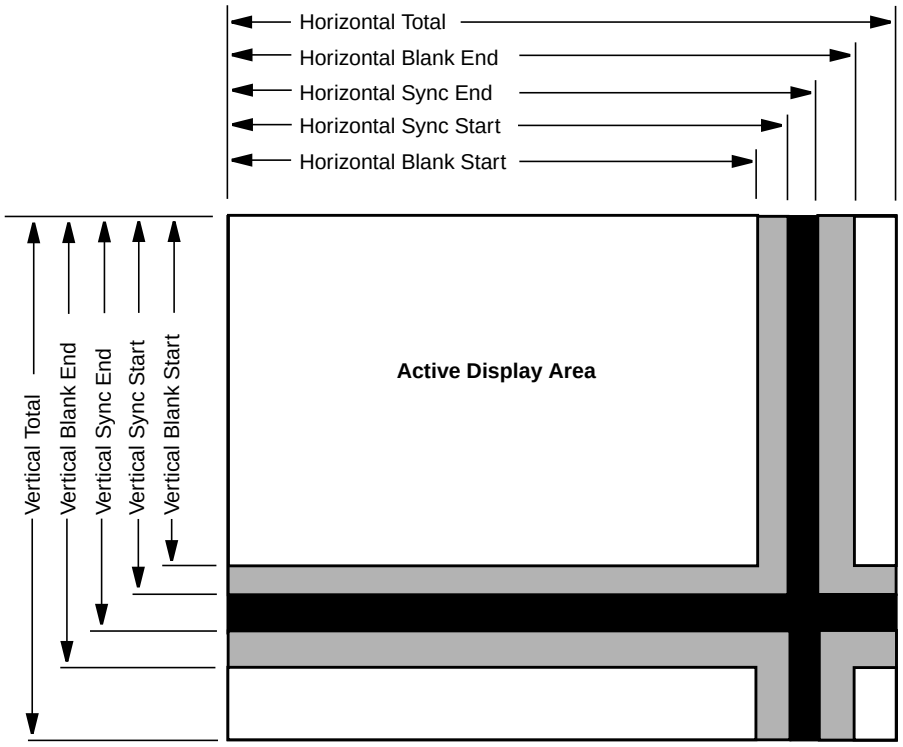


Figure 14: Video Frame Screen Timing

3.4 Programming the STG

Because the STG coordinate system begins at (0,0), it's necessary to program certain registers to one less than the desired value. For example, a scan line has 800 pixels total. The horizontal total should be set to 799 because 0—799 is a total of 800. The same applies to programming the vertical total.

In the vertical domain, there are three main timing intervals to set: vertical active time, vertical blank time, and vertical sync time. The position of the vertical sync defines the vertical front and back porches. Note that the vertical sync interval (and therefore vertical blanking) must be a minimum of one line in duration.

The STG has no specific requirement for horizontal blank and sync. The location, duration and even existence of horizontal blank and sync times is entirely display surface dependent. If the display surface does not require horizontal blanking, it's not necessary to program it into the STG.

Non-blanked area occurs when the currently active line is not within the vertical blanking interval or in the column of the horizontal blanking interval. Display layers can be programmed to reside on any portion of the screen. Any non-blanked screen position that does not have an active display layer pixel assigned to it will result in the background color or the previous layer pixel being displayed.

The QVCP provides clipping support at the edge of the defined H- and V-total. If a layer is positioned in a way that some part of it would exceed the overall screen dimensions, no wrapping occurs but the pixel layering in this area is marked as invalid, hence they are not being displayed.

The QVCP also supports negative screen positions i.e., top and left side clipping of layers. For negative x and y layer start positions, the following equations must be used:

```
if StartX < 0 then
StartX = Xtotal + 1 - ABS(StartX)
set StartX sign bit
StartY = Ydisplay - 1
else
StartX = StartX
StartY = StartY

if StartY < 0 then
StartY = Ytotal + 1 - ABS(StartY)
set StartY sign bit
else
StartY = StartY
```

In addition to the standard progressive QVCP display mode, another mode called “interlaced” can be switched on by setting the Interlaced control bit. In this mode the VTotal register no longer specifies the height of a frame but the height of a field. The field height alternates by one line depending on whether an odd or even field needs to be processed by the QVCP. Four registers are provided for this mode to specify the actual location of the vsync signal within a line in odd and even fields.

3.4.1 Changing Timing

All register settings to the timing generator take effect immediately and are not clock re- synchronized. (The start/stop bit is the exception. It takes effect immediately and is clock re-synchronized.) The only safe way to change screen timing is as follows:

1. Turn off the timing generator.
2. Program all registers needed in the new display mode.
3. Turn the timing generator back on. In the process, the entire display pipeline is reset. All display layers are reset, and the screen timing starts at the vertical total, which guarantees a complete vertical blank period and vertical sync signal at the start of any mode change.

3.5 Programming QVCP for Different Output Formats

Table 12 shows the programmer how to obtain the desired output formats. The programming bits reside in the “Control” register (offset 0x03C).

Table 12: Programming Values for Supported PNX15xx Series Output Formats

Format	Out_mode	d1_mode	Interleave	Oversample	clk_ratio	Ten-bit	Output format
Parallel	2	1	x	x	1:1	1/0	up: YYYYYY md: UUUUUU lo: VVVVVV
D1 interface (422)	0	1	0	1	1:2	1/0	lo: UYVYUY...stream
D1 Interface (444)	0	0	0	1	1:3	1/0	lo: YUVYUV...stream
Double D1	1	1	0	1	1:1	1/0	md: UVUVUVUVUVU lo: YYYYYYYYYYY

4. Application Notes

4.1 Special Features

4.1.1 Signature Analysis

Signature analysis is a feature where QVCP calculates a 16-bit signature on the upper 8 bits of each mixer output separately and sends it to a register which can be read easily once “sig_done” of the Signature3 register (offset 0x10E058) is set. QVCP follows a specific CRC algorithm to calculate the signature.

The signature analysis can be done independently on each layer output. Also, the signatures of data (Alpha, upper, middle, Lower path) and control (misc path, which consists of vsync, hsync, blank etc.) can be read separately. See the registers with offsets 0x10E050, 0x10E054 and 0x10E058 for more details.

4.2 Programming Help

The tables below attempt to provide some help in choosing the programming parameters for some of the video enhancement modules. It is to be noted, however, that the parameter settings shown below are just example settings that worked well on a particular experimental picture. One particular setting will not optimize the enhancement effects for different images (even when they are part of a sequence). For the real application, the parameters need to be adjusted, by the control software, according to the measurement results (obtained from assorted measurement units in other video modules of the parent chip).

4.3 LINT Parameters

The phase of the first output pixel is programmed using PCoeff (reg 0x10,E2A8[21:16]). The upscaling ratio is programmed in DPCoeff (reg 0x10,E2A8[11:0]). Table 13 shows the min-max values of the LINT programming parameters.

Table 13: LINT programming

Register bits	Value type	Minimum value	Maximum value
PCoeff (0x10 E2A8[21:16])	unsigned	0.0000_00	0.1111_11
DPCoeff (0x10 E2A8[11:0])	unsigned	0.0000_0000_0001	1.0000_0000_0000

If Scaling ratio == 1

DPCoeff = 0 // the decimal bits for 1.0000_0000_0000

else

DPCoeff = (1000 * H) / Scaling ratio

4.4 HSRU Parameters

The phase of the first output pixel is programmed via HSRU_phase (reg 0x10 E2AC[5:0]). HSRU_d_phase (reg 0x10E2AC[27:16]) is the first-order phase difference of the first output pixel, while HSRU_dd_phase (reg 0x10 E2B0[11:0]) and HSRU_ddd_phase (reg 0x10 E2B0[25:16]) are the second and the third-order phase differences, respectively, of the first output pixel. Table 14 shows the min-max values of the programmable HSRU parameters. Note that the decimal points are actually aligned for the different values and “s: stands of the sign extension that is automatically performed by hardware (only the lower *non-s* values need to be programmed).

For bypassing the HSRU, all the control parameters have to be set to 0.

Table 14: HSRU programming

Register bits	Value type	Minimum value	Maximum value
HSRU_phase (0x10 E2AC[5:0])	unsigned	0.0000_00	0.1111_11
HSRU_d_phase (0x10E2AC[27:16])	unsigned	0.0000_0000_0001	1.0000_0000_0000
HSRU_dd_phase (0x10 E2B0[11:0])	signed	s.ssss_ss10_0000_0000_00	s.ssss_ss01_1111_1111_11
HSRU_ddd_phase (0x10 E2B0[25:16])	signed	s.ssss_ssss_ssss_ss10_0000_0000	s.ssss_ssss_ssss_ss01_1111_1111

4.5 LSHR Parameters

Table 15: LSHR Programming Parameters

Register	Bits	Reset Value	Setting	
			Gentle	Strong
LSHR_PAR_0	31 - Enable LSHR	0	1	1
	30:24 - HDP_CORING_THR	0	8	4
Offset: 0x10 E2E8	23:21 - HDP_NEG_GAIN	0	4	4
	20:18 - HDP_DELTA	0	2	2
	17:14 - HDP_HPF_GAIN	0	4	7
	13:10 - HDP_BPF_GAIN	0	4	4
	9:6 - HDP_EPF_GAIN	0	4	8
	5:3 - KAPPA	0	0	0
	2 - ENABLE_LTI	0	1	1
	1 - ENABLE_CDS	0	1	1
	0 - ENABLE_HDP	0	1	1
LSHR_PAR_1	31 - WIDE_FORMAT	0	0	0
	30:19 - Unused	0	0	0
Offset: 0x10 E2F0	18:12 - LTI_CORING_THR	0	16	8
	11:8 - LTI_HPF_GAIN	0	0	0
	7:4 - LTI_HPF_GAIN	0	0	2
	3:0 - LTI_HPF_GAIN	0	4	6
LSHR_PAR_2	31:0 - ENERGY_SEL	0	2	2
	29:25 - Unused	0		
Offset: 0x10 E2F4	24:18 - LTI_MAX_GAIN	0	64	127
	17:14 - LTI_STEEP_GAIN	0	12	12
	13:6 - LTI_BASE_GAIN	0	-16	-16
	5:3 - LTI_STEEP_TAPS	0	3	3
	2:0 - LTI_MINMAX_TAPS	0	2	2

4.6 DCTI Parameters

Table 16: DCTI Programming Parameters

Register	Bits	Reset value	Setting	
			Gentle	Strong
Layer DCTI control	31:16 - Unused			
Offset: 0x10 E350	15 - Superhill	1	1	1
	14:11 - Threshold	4	8	4
	10 - Separate	0	0	0
	9 - Protection	1	1	1
	8:6 - Limit	7	2	7
	5:2 - Gain	8	2	8
	1 - Ddx sel	1	1	1
	0 - Enable	0	1	0

4.7 CFTR Parameters

Table 17: CFTR Programming Parameters

Register	Bits	Reset value	Setting	
			Gentle	Strong
Layer CFTR Blue/Skin Tone	31:25: - Unused			
Offset: 0x10 E348	24 - Blueycomp	1	0	1
	23:20 - Bluegain	10	10	15
	19:17 - Bluesize	4	2	0
	16: Blue_enable	0	1	1
	15:9 - Unused			
	8:6 - Skingain	2	2	3
	5:3 - Skintone	2	2	7
	2:1 - Skinsize	1	1	3
	0 - Skin_enable	0	1	1
Layer CFTR Green	31:15 - Unused			
Offset: 0x10 E34C	14:11 - Greenmax	9	9	15
	10:8 - Greensat	4	4	7
	7:4 - Greengain	7	7	15
	3:1 - Greensize	0	2	0
	0 - Green_enable	0	1	1

4.8 Underflow Behavior

This section briefs on the underflow handling in QVCP.

4.8.1 Layer Underflow

Any time the layer position has reached but the small 16-pixel FIFO at the end of every layer pipe has run out of available pixels, underflow occurs.

4.8.2 Underflow Symptom

- Only portion of a picture is displayed or occasional blinking of picture happens
- Underflow interrupt bit is set.

4.8.3 Underflow Recovery

Should an underflow occur, the layer would fetch and dump remaining data for the current field/frame. The next field/frame would be fetched and displayed as normal.

4.8.4 Underflow Trouble-shooting

- Check if the DMA source width settings (0x10,Ex08) matches the initial layer width (0x10,Ex34)
- Check if the initial layer width (0x10,Ex34) matches the final layer width (0x10,ExB4) for the non-scaled layer.
- Check if the final layer width (0x10,ExB4) is within acceptable cropping range for LINT or HSRU scaling.
- Check whether the DMA start fetch (0x10,ExC8) is at line number too close to the display position. Note that about 64 pixels is QVCP's input-to-output latency. So, depending on the system-memory latency, the DMA fetch should start as early as possible, in order to make up for the request-to-data latency.
- Check if the system memory arbiter is giving high priority to QVCP.
- Check if QVCP demands exceed allocated memory bandwidth.

4.8.5 Underflow Handling

The underflow interrupt status would stay asserted until an interrupt-status-clear is programmed.

4.9 Clock Calculations

Table 18 below cites a few examples for pixel and output clock calculations for some target resolutions.

Table 18: Interface Characteristics for Some Target Resolutions

Display Modes	Interface Mode	Sync	Interface Speed
4:2:2 CVBS or Y/C PAL/NTSC/SECAM resolution 4:2:2 Example PAL: 864 pixel/line x 312.5 lines/field x 50Hz = 13.5MHz/Y samples 7.5 MHz/U samples 7.5 MHz/V samples	4:2:2 Muxed components	embedded SAV/EAV D1 style	27 MHz
4:4:4 RGB or YUV PAL/NTSC/SECAM resolution 4:4:4 Example PAL: 864 pixel/line x 312.5 lines/field x 50Hz = 13.5 MHz/ component	4:4:4 Muxed Components	embedded SAV/EAV D1 style or external H/V/ Blank	40.5 MHz
4:4:4 RGB or YUV 2FH (double line frequency -> double refresh rate) Example PAL: 864 pixel/line x 312.5 lines/field x 50Hz x2 = 27 MHz/ component	4:4:4 Muxed Components	embedded SAV/EAV D1 style or external H/V/ Blank	81 MHz
4:4:4 RGB or YUV 480P (PAL/NTSC resolution, progressive) Example PAL: 864 pixel/line x 625 lines/field x 50Hz = 27 MHz/component	4:4:4 Muxed Components	embedded SAV/EAV D1 style external H/V/ Blank	81 MHz
4:4:4 RGB or YUV 1920x1080@60I Example: ·1920 active pixels per line (2200 total), 1080 active lines per frame (1125 total), 30 frames (60 fields) per second = 2200x 1125x 30 = 74.25 MHz	4:4:4 3x10 bits planar output	external H/V/ Blank	74.25 MHz

5. Register Descriptions

5.1 Register Summary

Table 19 summarizes the MMIO registers of QVCP. The offset are absolute offset relative to the MMIO_BASE. Only Layer 1 MMIO registers are displayed. Layer 2 MMIO registers are similar to Layer 1 MMIO registers but are located at offset 0x10E400 instead of 0x10E200.

Table 19: Register Module Association

Offset	Symbol	Module
0x10 E000	TOTAL	
0x10 E004	HBLANK	
0x10 E008	VBLANK	
0x10 E00C	HSYNC	
0x10 E010	VSYNC	
Control and Interrupt Registers		
0x10 E014	VINTERRUPT	
0x10 E018	FEATURES	
0x10 E01C	DEFAULT BACKGROUND COLOR	
0x10 E020	CONTROL	
0x10 E024	FINAL_LAYER_ASSIGNMENT	
0x10 E028	INTLCTRL1	
0x10 E02C	Reserved	
0x10 E030	VBI SRC Address	
0x10 E034	VBI_CTRL	
0x10 E038	VBI_SENT_OFFSET	
0x10 E03C	OUT_CTRL	
0x10 E040	POOL_RESOURCE_ID	
0x10 E044	POOL_RESOURCE_LAYER_ASSIGNMENT	
0x10 E048	RESOURCE_ID	
0x10 E04C	FU_ASSIGNMENT	
0x10 E050	Signature1	
0x10 E054	Signature2	
0x10 E058	Signature3	
0x10 E05C	Output pedestals1	
0x10 E060	Output pedestals2	
0x10 E064	Output GNSH LUT Data Upper	
0x10 E068	Output GNSH LUT Data Middle	
0x10 E06C	Output GNSH LUT Data Lower	
0x10 E070	Output ONSH Ctrl	
0x10 E074	Output GAMMA Ctrl	

Table 19: Register Module Association *...Continued*

Offset	Symbol	Module
0x10 E1F0	Shadow reload	
0x10 E1F8	Field_Info	
0x10 E1FC	XY_Position	
Layer & Mixer Registers		
0x10 E200	Layer Source Address A (Packed/Semi Planar Y)	DMA
0x10 E204	Layer Source Pitch A (Packed/Semi Planar Y)	DMA
0x10 E208	Layer Source Width (Packed/Semi Planar Y)	DMA
0x10 E20C	Layer Source Address B (Packed/Semi Planar Y)	DMA
0x10 E210	Layer Source Pitch B (Packed/Semi Planar Y)	DMA
0x10 E214	Dummy Pixel Count	PFU
0x10 E218	Layer Source Address A (Semi Planar UV)	DMA
0x10 E21C	Layer Source Address B (Semi Planar UV)	DMA
0x10 E228	Layer Source Pitch (Semi Planar UV)	DMA
0x10 E22C	Layer Source Width (Semi Planar UV)	DMA
0x10 E230	Layer Start	LCU
0x10 E234	Layer Size	LCU/DMA
0x10 E238	Pedestal and O/P format	CKEY(UDTH)
0x10 E23C	Layer Pixel Processing	DMA/LCU(MIX)/CUPS
0x10 E240	Layer Status/Control	LCU/PFU/DMA
0x10 E244	LUT Programming	LUT
0x10 E248	LUT Addressing	LUT
0x10 E24C	Pixel Key AND Register	PFU
0x10 E250	Color Key1 AND Mask	CKEY
0x10 E254	Color Key Up1	CKEY
0x10 E258	Color Key Low1	CKEY
0x10 E25C	Color Key Replace1	CKEY
0x10 E260	Color Key2 AND Mask	CKEY
0x10 E264	Color Key Up2	CKEY
0x10 E268	Color Key Low2	CKEY
0x10 E26C	Color Key Replace2	CKEY
0x10 E270	Color Key3 AND Mask	CKEY
0x10 E274	Color Key Up3	CKEY
0x10 E278	Color Key Low3	CKEY
0x10 E27C	Color Key Replace3	CKEY
0x10 E280	Color Key4 AND Mask	CKEY
0x10 E284	Color Key Up4	CKEY
0x10 E288	Color Key Low4	CKEY
0x10 E28C	Color Key Replace4	CKEY

Table 19: Register Module Association *…Continued*

Offset	Symbol	Module
0x10 E290	Color Key Mask/ROP	LCU(MIX)
0x10 E294	Pixel Invert/Select ROP	LCU(MIX)
0x10 E298	Alpha Blend/Key Pass	LCU(MIX)
0x10 E29C	Alpha Pass	LCU(MIX)
0x10 E2A0	Color Key ROPs 1/2	LCU(MIX)
0x10 E2A4	Color Key ROPs 3/4	LCU(MIX)
0x10 E2A8	INTR	INTR
0x10 E2AC	HSRU Phase	HSRU
0x10 E2B0	HSRU Delta Phase	HSRU
0x10 E2B4	Layer Size (final)	INTR/HSRU/LCU
0x10 E2B8	Output and Alpha manipulation	PFU/LCU(MIX)/CKEY
0x10 E2BC	Formats	PFU/CKEY
0x10 E2C0	Layer Background Color	LCU(MIX)
0x10 E2C4	Variable Format register	PFU
0x10 E2C8	Start Fetch	DMA/PFU
0x10 E2CC	Brightness & Contrast	VCBM
0x10 E2D0	Matrix Coefficients 1	VCBM
0x10 E2D4	Matrix Coefficients 2	VCBM
0x10 E2D8	Matrix Coefficients 3	VCBM
0x10 E2DC	Matrix Coefficients 4	VCBM
0x10 E2E0	Matrix Coefficients 5	VCBM
0x10 E2E8	LSHR_PAR_0	LSHR
0x10 E2EC	LSHR_PAR_1	LSHR
0x10 E2F0	LSHR_PAR_2	LSHR
0x10 E2F4	LSHR_PAR_3	LSHR
0x10 E2F8	LSHR_E_max	LSHR
0x10 E2FC	LSHR_E_sum	LSHR
0x10 E300	LSHR Measurement Window Start	LSHR
0x10 E304	LSHR Measurement Window End	LSHR
0x10 E320	Layer Solid Color	LCU(MIX)
0x10 E324	Layer LUT-HIST bins 00 to 03	HIST
0x10 E328	Layer LUT-HIST bins 04 to 07	HIST
0x10 E32C	Layer LUT-HIST bins 08 to 011	HIST
0x10 E330	Layer LUT-HIST bins 12 to 15	HIST
0x10 E334	Layer LUT-HIST bins 16 to 19	HIST
0x10 E338	Layer LUT-HIST bins 20 to 23	HIST
0x10 E33C	Layer LUT-HIST bins 24 to 027	HIST
0x10 E340	Layer LUT-HIST bins 28 to 31	HIST

12NC 9397 750 14321

Table 19: Register Module Association *...Continued*

Offset	Symbol	Module
0x10 E344	Layer Histogram control	HIST
0x10 E348	Layer CFTR Blue	CFTR
0x10 E34C	Layer CFTR Green	CFTR
0x10 E350	Layer DCTI Control	DCTI

5.2 Register Tables

Table 20: QVCP 1 Registers

Bit	Symbol	Access	Value	Description
Screen Timing Generator Registers				
Offset 0x10 E000	***TOTAL***			
31:28	Unused		-	
27:16	HTOTAL	R/W	0	Horizontal Total sets the number of horizontal pixels for the display. Total # of pixels per line = HTOTAL+1.
15:12	Unused		-	
11:0	VTOTAL	R/W	0	Vertical Total sets the number of vertical pixels for the display. Total # of lines per frame = VTOTAL +1. Total # of lines for odd field = VTOTAL +1. Total # of lines for even field = VTOTAL +2
Offset 0x10 E004	***HBLANK***			
31:28	Unused		-	
27:16	HBLANKS	R/W	0	Horizontal Blank Start sets the pixel location where horizontal blanking starts. Limitation: HTOTAL+2 >= HBLANKS >= 2.
15:12	Unused		-	
11:0	HBLNKE	R/W	0	Horizontal Blank End sets the pixel location where horizontal blanking ends. Limitation: HTOTAL >= HBLNKE >=0.
Offset 0x10 E008	***VBLANK***			
31:28	Unused		-	
27:16	VBLANKS	R/W	0	Vertical Blank Start sets the pixel location where vertical blanking starts. Limitation: VTOTAL+1 >=VBLANKS >=1.
15:12	Unused		-	
11:0	VBLANKE	R/W	0	Vertical Blank End sets the pixel location where vertical blanking ends. Limitation: VTOTAL >= VBLANKE >=0.
Offset 0x10 E00C	***HSYNC***			
31:28	Unused		-	
27:16	HSYNCS	R/W	0	Horizontal Sync Start sets the pixel location where horizontal sync starts. Limitation: HTOTAL+2 >= HSYNCS >=2.
15:12	Unused		-	
11:0	HSYNCE	R/W	0	Horizontal Sync End sets the pixel location where horizontal sync ends. Limitation: HTOTAL >= HSYNCE >=0.
Offset 0x10 E010	***VSYNC***			
31:28	Unused		-	

Table 20: QVCP 1 Registers *...Continued*

Bit	Symbol	Access	Value	Description
27:16	VSYNCS	R/W	0	Vertical Sync Start sets pixel location where Vertical sync starts. Limitation: VTOTAL+1 >= VSYNCS >=1.
15:12	Unused		-	
11:0	VSYNCE	R/W	0	Vertical Sync End sets pixel location where Vertical sync ends. Limitation: VTOTAL >= VSYNCE >=0.
Control and Interrupt Registers				
Offset 0x10 E014	***VINTERRUPT***			
31:28	Unused		-	
27:16	VLINTA	R/W	0	Vertical Line Interrupt A sets a vertical line number where an interrupt will be generated when the scan line matches this value. The interrupt is monitored by the Event Monitor (EVM). Limitation: VTOTAL >= VLINTA >=0.
15:12	Unused		-	
11:0	VLINTB	R/W	0	Vertical Line Interrupt B sets a vertical line number where an interrupt will be generated when the scan line matches this value. The interrupt is monitored by the Event Monitor (EVM). Limitation: VTOTAL >= VLINTB >=0.
Offset 0x10 E018	***FEATURES***			
31:30	NOOUT	R	0x1	Number of Output channels
29:27	Unused		-	
26:24	NOGNSH	R	0x1	Number of GNSHs
23:21	NOPLAN	R	0x1	Number of PLANs (semi planar channels)
20:18	NOLSHR	R	0x1	Number of LSHRs
17:15	NOHSRU	R	0x1	Number of HSRUs
14:12	NOHIST	R	0x1	Number of HISTs
11:9	NOCTI	R	0x1	Number of CTIs
8:6	NOCFTR	R	0x1	Number of CFTRs
5:3	NOCLUTS	R	0x1	Number of CLUTs
2:0	NOLAYERS	R	0x2	Number of layers
Offset 0x10 E01C	***DEFAULT BACKGROUND COLOR***			
31:24	Unused		-	
23:16	Upper	R/W	0	Background color of the upper channel (R/Y) (two's complement)
15:8	Middle	R/W	0	Background color of the middle channel (G/U) (two's complement)
7:0	Lower	R/W	0	Background color of the lower channel (B/V) (two's complement)

Table 20: QVCP 1 Registers *...Continued*

Bit	Symbol	Access	Value	Description
Offset 0x10 E020	***CONTROL***			
31:30	Unused		-	
29	Interlaced	R/W	0	Interlaced mode bit 0 = Non-interlaced mode; VTotal=frame height. 1 = Interlaced mode Field height = VTotal for odd fields. Field height = VTotal+1 for even fields. O_E flag = 0 for odd (bottom) fields O_E flag =1 for even (top) fields
28	BlankPol	R/W	0	BLANK Polarity 0 = Positive blank 1 = Negative blank
27	Unused		-	
26	HSYNCPol	R/W	0	HSYNC Polarity 0 = Positive going 1 = Negative going
25	Unused		-	
24	VSYNCPol	R/W	0	VSYNC Polarity 0 = Positive going 1 = Negative going
23:21	Unused		-	
20	BlankCtl	R/W	0	Blank Control allows either normal blanking or forces blanking to occur immediately. 0 = Blank output is equivalent to BlankPol setting 1 = Normal Blank
19	Unused		-	
18	HSYNCCtl	R/W	0	HSYNC Control enables or disables the horizontal sync output of the chip. 0 = HSYNC output is equivalent to HSYNCPol setting 1 = Enable
17	Unused		-	
16	VSYNCCtl	R/W	0	VSYNC Control enables or disables vertical sync output of the chip. 0 = VSYNC output is equivalent to VSYNCPol setting 1 = Enable
15:12	Unused		-	

Table 20: QVCP 1 Registers *...Continued*

Bit	Symbol	Access	Value	Description
11:8	AUXCTRL2	R/W	0	[9:8] = 2'b00 => output acts like a composite blanking signal controlled with BlankPol and BlankCtl [9:8] = 2'b01 => pouts Odd/Even signal in interlaced modes, zero in progressive modes [9:8] = 2'b10 => [11:10] = 2'b00 => outputs colorkey1 of mixer 2 [11:10] = 2'b01 => outputs colorkey2 of mixer 2 [11:10] = 2'b10 => outputs colorkey3 of mixer 2 [11:10] = 2'b11 => outputs colorkey4 of mixer 2 [9:8] = 2'b11 => reserved
7:4	AUXCTRL1	R/W	0	[5:4] = 2'b00 => output acts like a composite blanking signal controlled with BlankPol and BlankCtl [5:4] = 2'b01 => pouts Odd/Even signal in interlaced modes, zero in progressive modes [5:4] = 2'b10 => [7:6] = 2'b00 => outputs colorkey1 of mixer 2 [7:6] = 2'b01 => outputs colorkey2 of mixer 2 [7:6] = 2'b10 => outputs colorkey3 of mixer 2 [7:6] = 2'b11 => outputs colorkey4 of mixer 2 [5:4] = 2'b11 => reserved
3	DATA_OEN	R/W	0	Output enable control for video data bus 0=Data outputs enabled (normal operation) 1=Data outputs disabled (tri-state)
2	TRIGGER_POL	R/W	1	External trigger, i.e. VSYNC, polarity for the slave mode. 1 = Positive edge (default) 0 = Negative edge
1	MASTER	R/W	0	STG master/slave/operation 0 = Master mode 1 = Slave mode
0	TGRST	R/W	0	Timing generator reset 0 = Disable 1 = Enable Disable will reset all layer_enable bits (global QVCP reset).
Offset 0x10 E024	***FINAL_LAYER_ASSIGNMENT***			
31:8	Unused		-	
7:4	FLA2	R/W	1	Layer assignment to mixer 2 3'b000: Input layer1 => Mixer 2 3'b001: Input layer 2=> Mixer 2 all other settings are reserved
3:0	FLA1	R/W	0	Layer assignment to mixer 1 3'b000: Input layer1 => Mixer 1 3'b001: Input layer 2=> Mixer 1 all other settings are reserved
Offset 0x10 E028	***INTLCTRL1***			
31:28	Unused		-	

Table 20: QVCP 1 Registers *...Continued*

Bit	Symbol	Access	Value	Description
27:16	INT_START_E	R/W	0	Horizontal offset for VSYNC start even field (interlaced mode only) Vsync appears at INT_START_E + 1.
15:12	Unused		-	
11:0	INT_START_O	R/W	0	Horizontal offset for VSYNC start odd field (interlaced mode only) Vsync appears at INT_START_O + 1.
Offset 0x10 E030	***VBI SRC Address***			
31:28	Unused		-	
27:0	VBI_SRC_ADDR	R/W	0	VBI data source address
Offset 0x10 E034	***VBI_CTRL***			
31:1	Unused		-	
0	VBI_EN	R/W	0	Enable VBI data fetch engine.
Offset 0x10 E038	***VBI_SENT_OFFSET***			
31:12	Unused		-	
11:0	VBI_SENT_OFFSET	R/W	0	This programming value specifies the number of lines to add to the linecnt value in the packet identifier.
Offset 0x10 E03C	***OUT_CTRL***			
31:25	Unused		-	
24	TC_outS1R/Y	R/W	1	Set to unsigned format for the Y/R channel of the D1 slice: 1 = invert the MSB of the Y/R channel for the D1 slice 0 = leave D1 slice untouched
23	TC_outS1G/U	R/W	1	Set to unsigned format for the U/G channel of the D1 slice: 1 = invert the MSB of the U/G channel for D1 slice 0 = leave D1 slice untouched
22	TC_outS1B/V	R/W	1	Set to unsigned format for the V/B channel of the D1 slice: 1 = invert the MSB of the V/B channel for D1 slice 0 = leave D1 slice untouched
21	DNS1	R/W	0	444:422 down sample enable 1 = down sample filter enabled 0 = down sample filter bypassed
20:19	Unused		-	
18	Qualifier	R/W	0	1 = slice qualifier is put out 0 = hsync is put out
17:16	Outmode	R/W	0	00 = output interface runs is d1 mode 01 = output interface runs in double-d1 mode 10 = output interface operates in up to 30 bit parallel mode 11 = unused
15	parallel_mode	R/W	0	This bit controls the sync delay compensation. 1 = syncs are delayed (needed for 24-bit parallel output mode) 0 = no additional sync delay
14:13	Unused		-	

Table 20: QVCP 1 Registers *...Continued*

Bit	Symbol	Access	Value	Description
12	Oversample	R/W	1	This bit enables the output state machine for oversampling. This bit should be 0 for interleaved output modes. If one (only supported for a single d1 stream, either 444 or 422 or 444x) the output clock should be 2x the streaming clock i.e. 422 SD mode: streaming clock 27 MHz, output clock 54 MHz results in a 2x oversampling of the data stream. 1 = oversampling enabled 0 = no oversampling
11	Unused		-	
10	D1_MODE	R/W	1	1 = 4:2:2 D1 mode 0 = 4:4:4 D1 mode
9:3	Unused		-	
2:0	MUX_SEL_1	R/W	0	Tap off selection for first slice 0 = tap off after first mixing stage 1 = tap off after second mixing stage all other values are reserved
Offset 0x10 E040	***POOL_RESOURCE_ID***			
31:4	Unused		-	
3:0	PID	R/W	0	Pool resource ID register 9 = Color Look Up Table 10 = Horizontal Sample Rate Converter 11 = Luminance Sharpening Unit 12 = Histogram Modification Unit 13 = Color Features 14 = Dynamic Color Transient Improvement 15 = Semi Planar Channels
Offset 0x10 E044	***POOL_RESOURCE_LAYER_ASSIGNMENT***			
31:4	Unused		-	
3:0	PR1	R/W	0	Resource 1 assignment 4'b0000=layer 1 4'b0001=layer 2 all other values are reserved
Offset 0x10 E048	***RESOURCE_ID***			
31:4	Unused		-	
3:0	RID	R/W	0	Resource ID register
Offset 0x10 E04C	***FU_ASSIGNMENT***			
31:20	Unused		-	
19:16	R5	R/W	0	4'b0001=layer 1 4'b0010=layer 2 all other values are reserved, this register is not applicable for pool resources
15:12	R4	R/W	0	4'b0001=layer 1 4'b0010=layer 2 all other values are reserved, this register is not applicable for pool resources

Table 20: QVCP 1 Registers *...Continued*

Bit	Symbol	Access	Value	Description
11:8	R3	R/W	0	4'b0001=layer 1 4'b0010=layer 2 all other values are reserved, this register is not applicable for pool resources
7:4	R2	R/W	0	4'b0001=layer 1 4'b0010=layer 2 all other values are reserved
3:0	R1	R/W	0	4'b0001=layer 1 4'b0010=layer 2 all other values are reserved
Offset 0x10 E050	***Signature1***			
31:16	middle signature	R	0	Middle path signature
15:0	lower signature	R	0	Lower path signature
Offset 0x10 E054	***Signature2***			
31:16	alpha signature	R	0	Alpha path signature
15:0	upper signature	R	0	Upper path signature
Offset 0x10 E058	***Signature3***			
31:16	misc signature	R	0	Other signature
15:9	Unused		-	
8	sig_done	R	0	Signature done
7:6	Unused		-	
5:3	sig_select	R/W	0	Signature select 3'b000 = layer 1 output selected for signature analysis 3'b001 = layer 2 output selected for signature analysis all other values are reserved
2:1	Unused		-	
0	sig_enable	R/W	0	Signature enable
Offset 0x10 E05C	***Output pedestals1***			
31:24	Unused		-	
29:20	UPPER_PED1	R/W	0	Pedestal added to upper value (Signed value from -512 to 511)
19:10	MIDDLE_PED1	R/W	0	Pedestal added to upper value (Signed value from -512 to 511)
9:0	LOWER_PED1	R/W	0	Pedestal added to upper value (Signed value from -512 to 511)
Offset 0x10 E064	***Output GNSH LUT Data Upper***			
31:20	Unused		-	
19:10	BASE_UPPER	R/W	0	Upper data for Gamma Base table
9:0	DELTA_UPPER	R/W	0	Upper data for Gamma Delta table
Offset 0x10 E068	***Output GNSH LUT Data Middle***			
31:20	Unused		-	
19:10	BASE_MIDDLE	R/W	0	Middle data for Gamma Base table

Table 20: QVCP 1 Registers *...Continued*

Bit	Symbol	Access	Value	Description
9:0	DELTA_MIDDLE	R/W	0	Middle data for Gamma Delta table
Offset 0x10 E06C	***Output GNSH LUT Data Lower***			
31:20	Unused		-	
19:10	BASE_LOWER	R/W	0	Lower data for Gamma Base table
9:0	DELTA_LOWER	R/W	0	Lower data for Gamma Delta table
Offset 0x10 E070	***Output ONSH Ctrl***			
31:21	Unused	R/W	0	
4:3	GNSH_ERROR	R/W	0	GNSH Error mode 00= Truncation 01= Rounding 10= Error Propagation - not initialized except power-on 11= Error Propagation - initialized with hblank
2:1	GNSH_SIZE	R/W	0	GNSH Output resolution 00 = 9 bits 01 = 8 bits 10 = 10 bits 11 = 6bits
0	GNSH_422	R/W	0	GNSH mode 1 = YUV 4:2:2 0 = YUV 4:4:4 / RGB
Offset 0x10 E074	***Output GAMMA Ctrl***			
31	GAMMA_ENABLE	R/W	0	Gamma correction enable bit (also disables signal range adjustment & clipping if non-(9+1) bit mode selected) 1=active 0=bypass
30	HOST_ENABLE	R/W	0	This enables the HOST read/write access to GNSH 1=host access enabled 0=host access disabled, no access possible
29:25	Unused	-	-	
24	GNSH_SQUARE	R/W	0	Gamma correction with squaring enable bit 1=active 0=bypass
23:22	Unused	-	-	
21:16	UPPER_ADDR	R/W	0	Internal address in the upper Gamma Delta and base tables
15:14	Unused	-	-	
13:8	MIDDLE_ADDR	R/W	0	Internal address in the middle Gamma Delta and base tables
7:6	Unused	-	-	
5:0	LOWER_ADDR	R/W	0	Internal address in the lower Gamma Delta and base tables
Offset 0x10 E1F0	***Shadow_Reload***			
31	reserved	R/W	0	should always set to 0
30	reload_mode	R/W	0	0: reload all together at line location indicated by reload_line. 1: always reload at end pixel of the layer (old mode, do not use)

Table 20: QVCP 1 Registers *...Continued*

Bit	Symbol	Access	Value	Description
29:12	Unused	R	0	
11:0	reload_line	R/W	0	line count number where shadow reload occurs. Please make sure reload line is set to a position earlier than layer start Y position given in 0x10,E230.
Offset 0x10 E1F8	***Field_Info***			
31:3	Unused		-	
2:0	Field_ID	R	-	Field_ID is reset by disabling the screen timing generator Field_ID is incremented with each rising edge of VSYNC and wraps around after reaching the value 0x7 which yields a sequence of 8 fields which could be differentiated by using the Field_ID register.
Offset 0x10 E1FC	***XY_Position***			
31	O_E_STAT	R	0	Odd/Even flag status (interlaced mode) 0 = First field (odd/top field) 1 = Second field (even/bottom field)
30:28	Unused		-	
27:16	STG_Y_POS	R	-	Current vertical position of screen timing generator
15:12	Unused		-	
11:0	STG_X_POS	R	-	Current horizontal position of screen timing generator
Layer & Mixer Registers				
The structure of each layer function block is identical. The register for a function such as Source Address in Layer 1, has the same structure as the corresponding register in Layer 2. Layer one starts at offset 0x200 from the QVCP base address. Layer two starts at offset 0x400 from the QVCP base address.				
Offset 0x10 E200	***Layer Source Address A (Packed/Semi Planar Y)***			
31:28	Unused		-	
27:0	Layer N Source Address A	R/W	0	Layer N Source Data Start Address A in bytes. This sets starting address A for data transfers from the linear Frame Buffer memory to Layer N. For semi planar and planar modes this address points to the Y plane. **Note:** It should be aligned on a 128-byte boundary for memory performance reasons. It has to be 8-byte aligned.
Offset 0x10 E204	***Layer Source Pitch A (Packed/Semi Planar Y)***			
31:23	Unused		-	
22:0	Layer N Pitch A	R/W	0	Layer N Source Data Pitch B in bytes. This sets pitch A for data transfers from the linear Frame Buffer memory to Layer N. For semi planar and planar modes this determines the pitch for the Y plane. The value has to be rounded up to the next 64-bit word.
Offset 0x10 E208	***Layer Source Width (Packed/Semi Planar Y)***			
31:23	Unused		-	
12:0	Layer N Source Width	R/W	0	Layer N source width in bytes. For semi planar and planar modes this determines the source data with in bytes for the Y plane. The value has to be rounded up to the next 64-bit word.
Offset 0x10 E20C	***Layer Source Address B (Packed/Semi Planar Y)***			
31:28	Unused		-	

Table 20: QVCP 1 Registers *...Continued*

Bit	Symbol	Access	Value	Description
27:0	Layer N Source Address B	R/W	0	Layer N Source Data Start Address B in bytes. This sets starting address B for data transfers from the linear Frame Buffer memory to Layer N. For semi planar and planar modes this address points to the Y plane. **Note:** It should be aligned on a 128-byte boundary. It has to be 8-byte aligned.
Offset 0x10 E210	***Layer Source Pitch B (Packed/Semi Planar Y)***			
31:23	Unused		-	
22:0	Layer N Pitch B	R/W	0	Layer N Source Data Pitch B in bytes sets pitch B for data transfers from the linear Frame Buffer memory to Layer N. For semi planar and planar modes this determines the pitch for the Y plane. The value has to be rounded up to the next 64-bit word.
Offset 0x10 E214	***Dummy Pixel Count***			
31:8	Unused		-	
7:0	DCnt	R/W	0	Number of dummy pixels to be inserted between layer video lines
Offset 0x10 E218	***Layer Source Address A (Semi Planar UV)***			
31:28	Unused		-	
27:0	Layer Source Address A Semi Planar UV	R/W	0	Layer N Source Data Start Address A in bytes. This sets starting address A for data transfers from the linear Frame Buffer memory to Layer N. This Register holds the source address for the UV plane in semi planar modes. Note: It should be aligned on a 128-byte boundary. It has to be 8-byte aligned.
Offset 0x10 E21C	***Layer Source Address B (Semi Planar UV)***			
31:28	Unused		-	
27:0	Layer Source Address B Semi Planar UV	R/W	0	Layer N Source Data Start Address B in bytes. This sets starting address B for data transfers from the linear Frame Buffer memory to Layer N. This Register holds the source address for the UV plane in semi planar modes. Note: It should be aligned on a 128-byte boundary. It has to be 8-byte aligned.
Offset 0x10 E220	***Line Increment (Packed)***			
31:16	Unused		-	
15:0	Line Increment Packed	R/W	0xFFFFh	This register determines whether a layer line is repeatedly fetched from memory or not. Round Down(2^{16}/(Line Increment Packed))= #of times the same line is fetched i.e., 0x8000H would fetch each line exactly twice (line doubling).
Offset 0x10 E224	***Line Increment (Semi Planar)***			
31:16	Unused		-	
15:0	Line Increment Semi Planar	R/W	0xFFFFh	This register determines whether a layer line is repeatedly fetched from memory or not. Round Down(2^{16}/(Line Increment Semi Planar))= #of times the same line is fetched i.e., 0x8000H would fetch each line exactly twice (line doubling).
Offset 0x10 E228	***Layer Source Pitch (Semi Planar UV)***			

Table 20: QVCP 1 Registers *...Continued*

Bit	Symbol	Access	Value	Description
31:23	Unused		-	
22:0	Layer N Pitch Semi Planar	R/W	0	Layer N Source Data Pitch in bytes. This sets pitch for data transfers from the linear Frame Buffer memory to Layer N for semi planar modes. The value is used independent of whether buffer A or B is used. The value has to be rounded up to the next 64-bit word.
Offset 0x10 E22C	***Layer Source Width (Semi Planar UV)***			
31:23	Unused		-	
12:0	Layer N Source Width Semi Planar	R/W	0	Layer N source width in bytes for semi planar modes. The value is used independent of whether buffer A or B is used. The value has to be rounded up to the next 64-bit word.
Offset 0x10 E230	***Layer Start***			
31	Fine	R/W	0	Fine positioning enable for interlaced modes (layer size needs to be set to odd + even number of lines). Fine=0 : LayerNStartY is always relative to frame position, ie, LayerNStartY=100 will display the layer at STG_Y_POS=100 position. Fine=1 : LayerNStartY is always relative to field position, ie. LayerNStartY=100 will be translated to display layer at STG_Y_POS=100/2=50 position. Fine=1 is recommanded in interlaced mode. Fine=0 is recommanded in progressive mode.
30:29	Unused		-	
28:16	LayerNStartX	R/W	0	Layer N Start x position (from zero at left edge) in pixels. Negative X start position is possible.
15:13	Unused		-	
12:0	LayerNStartY	R/W	0	Layer N Start y position (from zero at top) in lines. Negative Y position is allowed. Note: In interlaced modes the following rules apply: Fine=0 : LayerNStartY is always relative to frame position i.e., LayerNStartY=100 will display the layer at STG_Y_POS=100 position. Fine=1 : LayerNStartY is always relative to field position i.e., LayerNStartY=100 will be translated to display layer at STG_Y_POS=100/2=50 position. Fine=1 is recommanded in interlaced mode. Fine=0 is recommanded in progressive mode. Whenever layer y position is changed, please make sure other y position sensitive register settings are still satisfied, such as : start fetch register 10E2C8, shadow reload position 10E1F0 layer start field register 10E23C (for interlaced mode)
Offset 0x10 E234	***Layer Size***			
31:28	Unused		-	
27:16	LayerNHeight	R/W	0	Layer N height in lines.
15:12	Unused		-	

Table 20: QVCP 1 Registers *...Continued*

Bit	Symbol	Access	Value	Description
11:0	LayerNWidth	R/W	0	initial (before scaling) Layer N width, in pixels.
Offset 0x10 E238	***Pedestal and O/P format***			
31:24	Pedestal_up	R/W	0	Pedestal to be added to Upper input (pedestal_up is a 2's complement number from -128 to 127) The pedestal removal is performed after the color key unit, before dithering.
23:16	Pedestal_mid	R/W	0	Pedestal to be added to Middle input (pedestal_mid is a 2's complement number from -128 to 127) The pedestal removal is performed after the color key unit, before dithering
15:8	Pedestal_low	R/W	0	Pedestal to be added to Lower input (pedestal_low is a 2's complement number from -128 to 127) The pedestal removal is performed after the color key unit, before dithering
7:3	Unused		-	
2:1	OP_format	R/W	0	Output type selector 0 = data expansion from 8 to 9 bit through multiply by two (zero in LSB) 1 = data expansion from 8 to 9 bit through multiply by two (MSB in LSB position) 2,3 = data expansion from 8 to 9 bit through multiply by two (undither operation)
0	FRMT_4xx	R/W	0	Input format indicator 0 = Input is in 4:4:4 format 1 = Input is in 4:2:2 format
Offset 0x10 E23C	***Layer Pixel Processing***			
31:6	Unused		-	
5	Buffer toggle	R/W	0	This bit controls the DMA buffer mode: 1 = Always toggle between buffer A and B (A=odd field, B=even field). 0 = No buffer toggle, always fetch from buffer spec A.
4	Layer_Start_Field	R/W	1	Field in which the layer gets actually enabled once the LayerN_Enable bit is set. This bit is used to invert the internal odd/even signal. If the result of the operation Layer_Start_Field xor OE is true the layer is enabled, otherwise the layer stays disabled until the OE signal changes. In non-interlaced modes: this bit must be set to 1'b1 since the internal odd/even signal is forced to zero. In interlaced modes: LayerNStartY (0x10,E230) >= 0, set this bit to 0 LayerNStartY (0x10,E230) < 0, set this bit to 1
3	Premult	R/W	0	If this bit is set, the incoming pixels are premultiplied with alpha. That disables the new x alpha multiplication in the mixer stage if alpha blending is enabled.

Table 20: QVCP 1 Registers *...Continued*

Bit	Symbol	Access	Value	Description
2	Alpha_use	R/W	0	Controls which alpha value is used for blending in the layer mixer stage 1 = Use previous alpha 0 = Use alpha of current layer
1	422:444_Interspersed	R/W	0	Chroma upsample filter operation mode 1 = use this mode if input samples are arranged interspersed 0 = use this mode if input samples are arranged co-sited
0	422:444_Enable	R/W	0	Chroma upsample filter enable 1 = chroma upsample filter is enabled 0 = chroma upsample filter is in bypass mode
Offset 0x10 E240	***Layer Status/Control***			
31:10	Unused		-	
9	Layer upload	R	-	This bit indicates if the register upload into the shadow area is still in progress. 1 = New register upload possible, previous upload is complete 0 = Upload in progress, DO NOT reprogram any registers as the results are undetermined
8:1	Unused		-	
0	LayerN_Enable	R/W	0	0 = Disable layer N 1 = Enable layer N This register reads always 0 if the screen timing generator is not enabled
Offset 0x10 E244	***LUT Programming***			
31:24	Alpha	R/W	0	Alpha value for LUT programming
23:16	Red	R/W	0	Red value for LUT programming
15:8	Green	R/W	0	Green value for LUT programming
7:0	Blue	R/W	0	Blue value for LUT programming
Offset 0x10 E248	***LUT Addressing***			
31:24	LUTAddress	R/W	0	Address register for LUT programming, no auto-increment is supported.
23:9	Unused		-	
8	Host_Enable	R/W	0	This enables read/write access by the host: 1 = Host access enabled. 0 = Host access disabled.
7:2	Unused		-	
1	Lut_enable	R/W	0	LUT enable signal 0 = bypass LUT 1 = Allow data to flow through LUT
0	Unused		-	
Offset 0x10 E24C	***Pixel Key AND Register***			
31:24	PixelKeyAND	R/W	0xFF	The bits 31:24 in 32 bpp mode are ANDed with this mask (input for KEY2). Not available when PF_10B_MODE(see 0x10 E2BC)is on.

Table 20: QVCP 1 Registers *...Continued*

Bit	Symbol	Access	Value	Description
23:0	Unused		-	
Offset 0x10 E250	***Color Key1 AND Mask***			
31:24	Unused		-	
23:0	ColorKeyAND1	R/W	0xFFFFFF	Defines a 24-bit Mask where the pixel is ANDed before it's keyed with the ColorKeyAND value.
Offset 0x10 E254	***Color Key Up1***			
31:24	Unused		-	
23:0	ColorKeyup1	R/W	0	Defines a 24-bit color key used for color keying inside the layer.
Offset 0x10 E258	***Color Key Low1***			
31:24	Unused		-	
23:0	ColorKeylow1	R/W	0	Defines a 24-bit color key used for color keying inside the layer.
Offset 0x10 E25C	***Color Key Replace1***			
31	Colorkeyreplaceen	R/W	0	Enables color replacement.
30:24	Unused		-	
23:0	ColorKeyreplace1	R/W	0	Defines a 24-bit color to be put into the data path if the color key matches. The data format to be used is an expanded 24-bit RGB/YUV format. If the data was fetched unsigned from memory, an unsigned value has to be used. Signed pixel data formats in memory require signed values in this register.
Offset 0x10 E260	***Color Key2 AND Mask***			
31:24	Unused		-	
23:0	ColorKeyAND2	R/W	0xFFFFFF	Defines a 24-bit Mask where the pixel is ANDed before it's keyed with the COLORKEY value.
Offset 0x10 E264	***Color Key Up2***			
31:24	Unused		-	
23:0	ColorKeyup2	R/W	0	Defines a 24-bit color key used for color keying inside the layer.
Offset 0x10 E268	***Color Key Low2***			
31:24	Unused		-	
23:0	ColorKeylow2	R/W	0	Defines a 24-bit color key used for color keying inside the layer.
Offset 0x10 E26C	***Color Key Replace2***			
31	Colorkeyreplaceen	R/W	0	Enables color replacement.
30:24	Unused		-	
23:0	ColorKeyreplace2	R/W	0	Defines a 24-bit color to be put into the data path if the color key matches. The data format to be used is an expanded 24-bit RGB/YUV format. If the data was fetched unsigned from memory, an unsigned value has to be used. Signed pixel data formats in memory require signed values in this register.
Offset 0x10 E270	***Color Key3 AND Mask***			
31:24	Unused		-	

Table 20: QVCP 1 Registers *...Continued*

Bit	Symbol	Access	Value	Description
23:0	ColorKeyAND3	R/W	0xFFFFFF	Defines a 24-bit Mask where the pixel is ANDed before it's keyed with the COLORKEY value.
Offset 0x10 E274	***Color Key Up3***			
31:24	Unused		-	
23:0	ColorKeyup3	R/W	0	Defines a 24-bit color key used for color keying inside the layer.
Offset 0x10 E278	***Color Key Low3***			
31:24	Unused		-	
23:0	ColorKeylow3	R/W	0	Defines a 24-bit color key used for color keying inside the layer.
Offset 0x10 E27C	***Color Key Replace3***			
31	Colorkeyreplaceen	R/W	0	Enables color replacement.
30:24	Unused		-	
23:0	ColorKeyreplace3	R/W	0	Defines a 24-bit color to be put into the data path if the color key matches. The data format to be used is an expanded 24-bit RGB/YUV format. If the data was fetched unsigned from memory, an unsigned value has to be used. Signed pixel data formats in memory require signed values in this register.
Offset 0x10 E280	***Color Key4 AND Mask***			
31:24	Unused		-	
23:0	ColorKeyAND4	R/W	0xFFFFFF	Defines a 24-bit Mask where the pixel is ANDed before it's keyed with the COLORKEY value.
Offset 0x10 E284	***Color Key Up4***			
31:24	Unused		-	
23:0	ColorKeyup4	R/W	0	Defines a 24-bit color key used for color keying inside the layer.
Offset 0x10 E288	***Color Key Low4***			
31:24	Unused		-	
23:0	ColorKeylow4	R/W	0	Defines a 24-bit color key used for color keying inside the layer.
Offset 0x10 E28C	***Color Key Replace4***			
31	Colorkeyreplaceen	R/W	0	Enables color replacement.
30:24	Unused		-	
23:0	ColorKeyreplace4	R/W	0	Defines a 24-bit color to be put into the data path if the color key matches. The data format to be used is an expanded 24-bit RGB/YUV format. If the data was fetched unsigned from memory, an unsigned value has to be used. Signed pixel data formats in memory require signed values in this register.
Offset 0x10 E290	***Color Key Mask/ROP***			
31:24	Unused		-	
23:20	ColorKeyMask	R/W	0	This mask specifies which color to key in for the current pixel coming out of the layer.
19:16	ColorKeyMaskP	R/W	0	This color Mask is used to decide which color key to use for the incoming previous pixel.

Table 20: QVCP 1 Registers *...Continued*

Bit	Symbol	Access	Value	Description
15:8	Unused		-	
7:6	PassColorKey3	R/W	0	This register determines how to handle color key forwarding 00 pass zeros to the next mixer 01 pass current color key 3 to next mixer 10 pass previous color key 3 to next mixer 11 reserved
5:4	PassColorKey2	R/W	0	This register determines how to handle color key forwarding 00 pass zeros to the next mixer 01 pass current color key 2 to next mixer 10 pass previous color key 2 to next mixer 11 reserved
3:2	PassColorKey1	R/W	0	This register determines how to handle color key forwarding 00 pass zeros to the next mixer 01 pass current color key1 to next mixer 10 pass previous color key1 to next mixer 11 reserved
1:0	PassColorKey0	R/W	0	This register determines how to handle color key forwarding 00 pass zeros to the next mixer 01 pass current color key 0 to next mixer 10 pass previous color key 0 to next mixer 11 reserved
Offset 0x10 E294	***Pixel Invert/Select ROP***			
31:16	InvertROP	R/W	0	This ROP decides if the previous pixel is inverted or not. ROP output: 1 = Invert previous pixel. 0 = Do not invert previous pixel.
15:0	SelectROP	R/W	0	This ROP determines which pixel to select for the current mixer output. ROP output: 1 = Select previous pixel. 0 = Select new pixel.
Offset 0x10 E298	***Alpha Blend/Key Pass***			
31:16	AlphaBlend	R/W	0	This ROP value determines whether or not to do an alpha blend. ROP output: 1 = Do alpha blending. 0 = No alpha blending
15:0	KeyPass	R/W	0	This ROP generates the key which is passed to the next layer mixer and is used as KEY0 in those ROPs.
Offset 0x10 E29C	***Alpha Pass***			
31:16	AlphaPass	R/W	0	This ROP value determines which alpha is passed to the next mixer stage. ROP output: 1 = Alpha of previous pixel 0 = Alpha of current pixel
15:0	Unused		-	
Offset 0x10 E2A0	***Color Key ROPs 1/2***			
31:16	Unused		-	

Table 20: QVCP 1 Registers *...Continued*

Bit	Symbol	Access	Value	Description
15:8	ColorKeyROP1	R/W	0	This ROP determines if results of component color keying are true or not. Keys to the ROP are range_match upper, middle, lower. Upper match is key2, middle match is key1, lower match is key0. 0 = Color key didn't match. 1 = Color key matched.
7:0	ColorKeyROP2	R/W	0	This ROP determines if results of component color keying are true or not. Keys to the ROP are range_match upper, middle, lower. Upper match is key2, middle match is key1, lower match is key0. 0 = Color key didn't match. 1 = Color key matched.
Offset 0x10 E2A4	***Color Key ROPs 3/4***			
31:16	Unused		-	
15:8	ColorKeyROP3	R/W	0	This ROP determines if results of component color keying are true or not. Keys to the ROP are range_match upper, middle, lower. Upper match is key2, middle match is key1, lower match is key0. 0 = Color key didn't match. 1 = Color key matched.
7:0	ColorKeyROP4	R/W	0	This ROP determines if the results of component color keying are true or not. Keys to the ROP are range_match upper, middle, lower. Upper match is key2, middle match is key1, lower match is key0. 0 = Color key didn't match. 1 = Color key matched.
Offset 0x10 E2A8	***INTR***			
31:22	Unused		-	
21:16	PCoeff	R/W	-	Phase coefficient
15:12	Unused		-	
11:0	DPCoeff	R/W	-	Differential phase coefficient For the interpolator to work in bypass mode this register has to be programmed to 0
Offset 0x10 E2AC	***HSRU Phase***			
31:28	Unused		-	
27:16	HSRU_d_phase	R/W	0	Unsigned. This delta phase is added with phase with every output data. Once phase is added with a certain number of d_phases to get overflowed, then it's time shift input sample signals. For the HSRU to work in bypass mode this register has to be programmed to 0. Example: 8000 (hex) => upscaling by 2 4000 (hex) => upscaling by 4
15:7	Unused		-	
5:0	HSRU_phase	R/W	0	Unsigned. This is the initial phase of input pixel phase. It determines the portions of the first input samples used to generate output pixels.
Offset 0x10 E2B0	***HSRU Delta Phase***			
31:26	Unused		-	

Table 20: QVCP 1 Registers *...Continued*

Bit	Symbol	Access	Value	Description
25:16	HSRU_ddd_phase	R/W	0	Signed. This delta-delta-delta phase is added with delta-delta phase to make it change. This is used for non-linear scaling ratios.For the HSRU to work in bypass mode this register has to be programmed to 0.
15:12	Unused		-	
11:0	HSRU_dd_phase	R/W	0	Signed. This is the initial delta-delta phase. It is added with delta phase to make it change. This is used for non-linear scaling ratios. For the HSRU to work in bypass mode this register has to be programmed to 0. Note: Layer Size(final) register has to be modified if HSRU scale ratio is changed.
Offset 0x10 E2B4	***Layer Size (final)***			
31:12	Unused		-	
11:0	LayerNWidth	R/W	0	final (after scaling) Layer N width, in pixels. **Note**: This register has to be programmed to match the final width after scaling, as given by the equation below. • Final width = (input width)*scaling ratio LINT and HSRU can only crop at most 5 pixels off a scaled image. Setting this register to a width which is more than 5 pixels smaller than the scaled width can result in data underflow. On the other hand, if the final width is greater than the scaled image, the last pixel will be repeated to fill the final width. Always remember to update this register if LINT or HSRU scale values are changed.
Offset 0x10 E2B8	***Output and Alpha manipulation***			
31:24	Unused		-	
23:16	LAYER_FIXED_ALPHA	R/W	0	Alpha blend value to be applied to mixer. Provides 256 levels of fixed alpha blending: The AlphaSelect ROP must be set appropriately to use this feature.
15:14	PF_ALPHA_MODE	R/W		Control how Alpha channel data is generated 00: Fixed Alpha 01: Fixed Alpha 10: Per pixel alpha 11: Per pixel alpha is multiplied with (fixed_alpha)/256 Mode 11 is not effective when PF_10B_MODE is on since the Alpha value is set to zero.
13	Unused			
12	PF_A2C	R/W	0	Controls alpha channel format within layer 0 = data untouched 1 = data conversion two's compliment <-> binary offset The conversion takes place after the color key unit before the undither unit.

Table 20: QVCP 1 Registers *...Continued*

Bit	Symbol	Access	Value	Description
11	PF_U2C	R/W	0	Controls upper channel format within layer 0 = data untouched 1 = data conversion two's compliment <-> binary offset The conversion takes place after the color key unit before the undither unit.
10	PF_M2C	R/W	0	Controls middle channel format within layer 0 = data untouched 1 = data conversion two's compliment <-> binary offset The conversion takes place after the color key unit before the undither unit.
9	PF_L2C	R/W	0	Controls lower channel format within layer 0 = data untouched 1 = data conversion two's compliment <-> binary offset The conversion takes place after the color key unit before the undither unit
8:6	Unused		-	
5:3	PF_OFFSET2	R/W	0	Defines pixel offset (in bytes) within a multi-pixel 64-bit word for channel 2 for semi-planar and planar modes. 0, 2 or 4 for 10-bit YUV 4:2:2 semi-planar format 0 to 7 for 8-bit YUV 4:2:2 semi-planar format The number will be truncated to the closest even number for channel 2
2:0	PF_OFFSET1	R/W	0	Defines pixel offset (in bytes) within a multi-pixel 64-bit word. 0, 2 or 4 for 10-bit YUV 4:2:2 semi-planar format 0 or 4 for 10-bit (20 bpp) packed YUV 4:2:2 format 0, 2, 4 or 6 for 8-bit (16 bpp) packed YUV 4:2:2 or 16-bit varible format 0 or 4 for 32-bit varible format 0 to 7 for all the other formats
Offset 0x10 E2BC Formats				
31:14	Unused		-	
13	PF_ENDIAN	R/W	0	Input format endian mode 0: Same as system endian mode 1: Opposite of system endian mode Not available when PF_10B_MODE is on.
12	Unused			
11:10	PF_PIX_MODE	R/W	0	Pixel key output modes 00: Both keys '0' 01: Bits [1:0] of V/B output 10: Key 2 Bit [7] of alpha output 11: Key 2 AND of pixel key and alpha is not zero Mode 11 is not available when PF_10B_MODE is on.
9	Unused		-	

Table 20: QVCP 1 Registers *...Continued*

Bit	Symbol	Access	Value	Description
8	PF_10B_MODE	R/W	0	10-bit Input format modes 0: 8-bit input format mode 1: 10-bit input format mode
7:0	PF_IPFMT	R/W	0	Input Formats 08 (hex) = YUV 4:2:2 semi-planar 24 (hex) = 1-bit indexed[Note1] 45 (hex) = 2-bit indexed[Note1] 66 (hex) = 4-bit indexed[Note1] 87 (hex) = 8-bit indexed[Note1] A0 (hex) = packed YUY2 4:2:2 A1 (hex) = packed UYVY 4:2:2 AC (hex) = 16 bits variable contents 4:4:4 CC (hex) = 24 bits variable contents 4:4:4 E8 (hex) = 32 bits variable contents 4:2:2 EC (hex) = 32 bits variable contents 4:4:4 **Note1:** For indexed modes Variable format register should be set 'E7E7E7E7' **Note 2**: Only YUV 4:2:2 semi-plana format (08) and packed formats (A0 & A1) are available when PF_10B_MODE is on
Offset 0x10 E2C0	***Layer Background Color***			
31	BG_enable	R/W	0	This bit enables the replacement of the previous input by the specified background color. 1 = Replace 0 = Use previous mixer output.
30:24	Unused		-	
23:16	Upper	R/W	0	Upper channel of the background color (R/Y) (two's complement)
15:8	Middle	R/W	0	Middle channel of the background color (G/U) (two's complement)
7:0	Lower	R/W	0	Lower channel of the background color (B/V) (two's complement)
Offset 0x10 E2C4	***Variable Format register***			
31:29	PF_SIZE_A[2:0]	R/W	0	Size component for alpha Number of bits minus 1 (e.g. 7 => 8 bits per component) Not available when PF_10B_MODE is on.
28:24	PF_OFFS_A[4:0]	R/W	0	Offset component for alpha Index of MSB position within 32-bit word (0-31) Not available when PF_10B_MODE is on.
23:21	PF_SIZE_L[2:0]	R/W	0	Size component for V or B Number of bits minus 1 (e.g. 7 => 8 bits per component) Not available when PF_10B_MODE is on.
20:16	PF_OFFS_L[4:0]	R/W	0	Offset component for V or B Index of MSB position within 32-bit word (0-31) Not available when PF_10B_MODE is on.
15:13	PF_SIZE_M[2:0]	R/W	0	Size component for U or G Number of bits minus 1 (e.g. 7 => 8 bits per component) Not available when PF_10B_MODE is on.
12:8	PF_OFFS_M[4:0]	R/W	0	Offset component for U or G Index of MSB position within 32-bit word (0-31) Not available when PF_10B_MODE is on.

Table 20: QVCP 1 Registers *...Continued*

Bit	Symbol	Access	Value	Description
7:5	PF_SIZE_U[2:0]	R/W	0	Size component for Y or R Number of bits minus 1 (e.g. 7 => 8 bits per component) Not available when PF_10B_MODE is on.
4:0	PF_OFFS_U[4:0]	R/W	0	Offset component for Y or R Index of MSB position within 32-bit word (0-31) Not available when PF_10B_MODE is on.
Offset 0x10 E2C8	***Start Fetch***			
31	Enable	R/W	0	Set this bit to delay the DMA data fetch timing until line number specified in bit 11:0 is reached. If disabled, DMA will pre-fetch data for the next field at the end of current field.
27:16	FlushCount	R/W	0x30h	The number of flush pixels to be inserted after the end of a field. If Start Fetch is enabled this register must contain a large enough value to flush all pixels out of the pipeline after the last pixel entered the pixel formatter. (approx. 50)
15:12	Unused	R/W	-	
11:0	Fetch Start	R/W	0	If enabled (by setting bit 31 to 1), the data fetched from memory will be delayed until line number set here is reached, ie. the data pre-fetch is disabled. The number given here must be set to a value earlier in Y position than LayerNStartY in 10E230 to prevent from layer underflow. In non-interlaced mode : this value is relative to FRAME position. For example, if LayerNStartY=100, a start fetch position of 98 is deemed earlier position. In interlaced mode: this value is relative to FIELD position. For example, if LayerNStartY=100, a start fetch position of 52 is deemed one line too late to start the fetch, because LayerNStartY=100 is equivalent to field position 100/2=50. Therefore, a start fetch positon of 48 is a proper one.
Offset 0x10 E2CC	***Brightness & Contrast***			
31:29	Unused		-	
28	VCBM_U2B	R/W	0	Brightness control bit for upper channel. VCBM_U2B = 1 if brightness control is activated for the upper channel.
27	VCBM_M2B	R/W	0	Brightness control bit for middle channel. VCBM_M2B = 1 if brightness control is activated for the middle channel.
26	VCBM_L2B	R/W	0	Brightness control bit for lower channel. VCBM_L2B = 1 if brightness control is activated for the lower channel.

Table 20: QVCP 1 Registers *...Continued*

Bit	Symbol	Access	Value	Description
25:16	VCBM_BRIGHTNESS	R/W	0	Brightness Setting (Signed value ranging from -512 to 511 which is equivalent to -100% to +100% brightness change for nominal signals having a range from -256 to 255 with a contrast setting of 256) Nominal value: 0 Brightness control is performed after e color space conversion in associated with the contrast control. Y“ = Y‘ + VCBM_BRIGHTNESS if VCBM_U2B flag is raised.
15	VCBM_U2C	R/W	0	Two’s complement to binary offset conversion for contrast control (upper channel Y’/R’ = Y/R + VCBM_BLK_OFFSET) Needs to be set in case the data format entering the VCBM is YUV or RGB
14	VCBM_M2C	R/W	0	Two’s complement to binary offset conversion for contrast control (middle channel U/G) Needs to be set in case the data format entering the VCBM is RGB
13	VCBM_L2C	R/W	0	Two’s complement to binary offset conversion for contrast control (lower channel V/B) Needs to be set in case the data format entering the VCBM is RGB.
12	VCBM_U2CO	R/W	0	Back-end reverse offset (for two’s complement mentioned above) control bit for upper channel. (Yout/Rout = Yout/Rout - VCBM_BLK_OFFSET described below)
11	VCBM_M2CO	R/W	0	Back-end reverse offset (for two’s complement mentioned above) control bit for middle channel
10	VCBM_L2CO	R/W	0	Back-end reverse offset (for two’s complement mentioned above) control bit for lower channel
9:0	VCBM_BLK_OFFSET	R/W	0x100	Signed 10-bit two's complement to binary offset (or Black-level offset). +256 is the default value.
Offset 0x10 E2D0	***Matrix Coefficients 1***			
31:27	Unused		-	
26:16	C_11	R/W	0x100	Color space conversion matrix coefficient - C11 matrix component
15:11	Unused		-	
10:0	C_12	R/W	0	Color space conversion matrix coefficient - C12 matrix component
Offset 0x10 E2D4	***Matrix Coefficients 2***			
31:27	Unused		-	
26:16	C_13	R/W	0	Color space conversion matrix coefficient - C13 matrix component
15:11	Unused		-	
10:0	C_21	R/W	0	Color space conversion matrix coefficient - C21 matrix component
Offset 0x10 E2D8	***Matrix Coefficients 3***			
31:27	Unused		-	
26:16	C_22	R/W	0x100	Color space conversion matrix coefficient - C22 matrix component
15:11	Unused		-	
10:0	C_23	R/W	0	Color space conversion matrix coefficient - C23 matrix component
Offset 0x10 E2DC	***Matrix Coefficients 4***			
31:27	Unused		-	

Table 20: QVCP 1 Registers *...Continued*

Bit	Symbol	Access	Value	Description
26:16	C_31	R/W	0	Color space conversion matrix coefficient - C31 matrix component
15:11	Unused		-	
10:0	C_32	R/W	0	Color space conversion matrix coefficient - C32 matrix component
Offset 0x10 E2E0	***Matrix Coefficients 5***			
31:27	Unused		-	
26:16	C_33	R/W	0x100	Color space conversion matrix coefficient - C33 matrix component
15:10	Unused		-	
9	DIV_BY_512	R/W	0	Matrix coefficient fraction precision setting 0 = 8-bit fraction format; Matrix product is divided by 256. 1 = 9-bit fraction format; Matrix product is divided by 512.
8	VCBM_ENABLE	R/W	0	Operate on inputs or bypass block 0 = Bypass block 1 = Allow data flow through block
7:0	Unused		-	
Offset 0x10 E2E8	***LSHR_PAR_0***			
31	ENABLE_LSHR	R/W	0	Enable or disable LSHR, if disable LSHR will operate in bypass mode.
30:24	HDP_CORING_THR	R/W	0	HDP coring threshold Coring threshold for HDP adjustment (0..127)
23:21	HDP_NEG_GAIN	R/W	0	HDP negative overshoot adjustment factor Look-up table step size adjustment factor for negative overshoots (0..4)
20:18	HDP_DELTA	R/W	0	HDP LUT step size factor Step size factor for HDP look-up table (0..4)
17:14	HDP_HPF_GAIN	R/W	0	HDP HPF filter gain Weighting factor for HPF filter in HDP (0..15: sum of HDP filter gains must be 32 or less)
13:10	HDP_BPF_GAIN	R/W	0	HDP BPF filter gain Weighting factor for BPF filter in HDP (0..15: sum of HDP filter gains must be 32 or less)
9:6	HDP_EPF_GAIN	R/W	0	HDP EPF filter gain Weighting factor for EPF filter in HDP (0..15: sum of HDP filter gains must be 32 or less)
5:3	KAPPA	R/W	0	EPF filter selector Determines response of EPF filter (0,1,2,4)
2	ENABLE_LTI	R/W	0	Enable luma transient improvement 1:include LTI; 0:not LTI
1	ENABLE_CDS	R/W	0	Enable color dependent sharpness 1:include CDS; 0:not CDS
0	ENABLE_HDP	R/W	0	Enable horizontal dynamic peaking 1:include HDP; 0:not HDP
Offset 0x10 E2EC	***LSHR_PAR_1***			
31:26	CDS_CORING_THR	R/W	0	CDS coring threshold Coring threshold for CDS adjustment (0..63)

Table 20: QVCP 1 Registers *...Continued*

Bit	Symbol	Access	Value	Description
25:22	CDS_GAIN	R/W	0	CDS gain factor Strength of CDS adjustment (0..15)
21:18	CDS_SLOPE	R/W	0	CDS transition slope Determines size in UV plane of CDS adjustment transition from onset to maximum (0..15)
17:14	CDS_AREA	R/W	0	CDS onset area Determines location in UV plane of onset of CDS adjustment (0..15)
13	Unused		-	
12:9	HDP_LUT_GAIN	R/W	0	HDP LUT gain factor Gain factor for HDP look-up table scaling (0..15)
8:0	Unused		-	
Offset 0x10 E2F0	***LSHR_PAR_2***			
31	WIDE_FORMAT	R/W	0	Wide format modeSwitches internal filters, adapting to narrow and wide output horizontal resolutions 0:less than or equal to 1280 pixels per line 1: greater than 1280 pixels per line
30:19	Unused		-	
18:12	LTI_CORING_THR	R/W	0	LTI coring threshold Coring threshold for LTI adjustment (0..127)
11:8	LTI_HPF_GAIN	R/W	0	LTI HPF filter gain Weighting factor for HPF filter in LTI(0..15;sum of LTI filters gain must be 32 or less)
7:4	LTI_BPF_GAIN	R/W	0	LTI BPF filter gain Weighting factor for BPF filter in LTI(0..15;sum of LTI filters gain must be 32 or less)
3:0	LTI_EPF_GAIN	R/W	0	LTI EPF filter gain Weighting factor for EPF filter in LTI(0..15;sum of LTI filters gain must be 32 or less)
Offset 0x10 E2F4	***LSHR_PAR_3***			
31:30	ENERGY_SEL	R/W	0	Energy measurement sharpening filter selector 0: HPF (maximum), 4*HPF(sum); 1: BPF; 2: EPF; 3: HPF;
29:25	Unused		-	
24:18	LTI_MAX_GAIN	R/W	0	LTI gain factor limit Maximum LTI gain factor (0..127)
17:14	LTI_STEEP_GAIN	R/W	0	LTI gain factor steepness slope Slope of steepness influence on LTI gain factor (0..15)
13:6	LTI_BASE_GAIN	R/W	0	LTI basic gain factor Basic LTI gain (strength) factor, no steepness(-128..127)
5:3	LTI_STEEP_TAPS	R/W	1	LTI luma steepness filter width Single-sided width of LTI luma steepness filter (1..7)
2:0	LTI_MINMAX_TAPS	R/W	1	LTI luma minimum, maximum filter width Single-sided widths of LTI luma minimum and maximum filters (1..7)

Table 20: QVCP 1 Registers *...Continued*

Bit	Symbol	Access	Value	Description
Offset 0x10 E2F8	***LSHR_E_max***			
31:10	Unused		-	
9:0	LSHR_E_max	R	0	Statistics on one of the sharpness filter max measurement value = 10bU
Offset 0x10 E2FC	***LSHR_E_sum***			
31:26	Unused		-	
25:0	LSHR_E_sum	R	0	Statistics on one of the sharpness filter sum of abs max energy value = 26bU
Offset 0x10 E300	***LSHR Measurement Window Start***			
31:27	Reserved			
26:16	LSHR_MW_START_Y	R/W	0	LSHR measurement window start line (The first line included in the measurement window, the layer start position is (0,0)).
15:11	Reserved			
10:0	LSHR_MW_START_X	R/W	0	LSHR measurement window start pixel
Offset 0x10 E304	***LSHR Measurement Window End***			
31:27	Reserved			
26:16	LSHR_MW_END_Y	R/W	7FF	LSHR measurement window end line (The last line included in the measurement window)
15:11	Reserved			
10:0	LSHR_MW_END_X	R/W	7FF	LSHR measurement window end pixel
Offset 0x10 E320	***Layer Solid Color***			
31	SC_enable	R/W	0	This bit enables the replacement of the layer input by the specified color. 1 = Replace 0 = Use layer input
30:24	Unused		-	
23:16	Upper	R/W	0	Upper channel of the replacement color (R/Y) (two's complement)
15:8	Middle	R/W	0	Middle channel of the replacement color (G/U) (two's complement)
7:0	Lower	R/W	0	Lower channel of the replacement color (B/V) (two's complement)
Offset 0x10 E324	***Layer LUT-HIST Bins 00 to 03***			
31:24	bin03	R/W	0	8-bit signed offset from a Yout=Yin Curve for Yin=-192+ped register
23:16	bin02	R/W	0	8-bit signed offset from a Yout=Yin Curve for Yin=-208+ped register
15:8	bin01	R/W	0	8-bit signed offset from a Yout=Yin Curve for Yin=-224+ped register
7:0	bin00	R/W	0	8-bit signed offset from a Yout=Yin Curve for Yin=-240+ped register
Offset 0x10 E328	***Layer LUT-HIST Bins 04 to 07***			
31:24	bin07	R/W	0	8-bit signed offset from a Yout=Yin Curve for Yin=-128+ped register
23:16	bin06	R/W	0	8-bit signed offset from a Yout=Yin Curve for Yin=-144+ped register
15:8	bin05	R/W	0	8-bit signed offset from a Yout=Yin Curve for Yin=-160+ped register
7:0	bin04	R/W	0	8-bit signed offset from a Yout=Yin Curve for Yin=-176+ped register

Table 20: QVCP 1 Registers *...Continued*

Bit	Symbol	Access	Value	Description
Offset 0x10 E32C	***Layer LUT-HIST Bins 08 to 011***			
31:24	bin11	R/W	0	8-bit signed offset from a Yout=Yin Curve for Yin=-64+ped register
23:16	bin10	R/W	0	8-bit signed offset from a Yout=Yin Curve for Yin=-80+ped register
15:8	bin09	R/W	0	8-bit signed offset from a Yout=Yin Curve for Yin=-96+ped register
7:0	bin08	R/W	0	8-bit signed offset from a Yout=Yin Curve for Yin=-112+ped register
Offset 0x10 E330	***Layer LUT-HIST Bins 12 to 15***			
31:24	bin15	R/W	0	8-bit signed offset from a Yout=Yin Curve for Yin= 0+ped register
23:16	bin14	R/W	0	8-bit signed offset from a Yout=Yin Curve for Yin=-16+ped register
15:8	bin13	R/W	0	8-bit signed offset from a Yout=Yin Curve for Yin=-32+ped register
7:0	bin12	R/W	0	8-bit signed offset from a Yout=Yin Curve for Yin=-48+ped register
Offset 0x10 E334	***Layer LUT-HIST Bins 16 to 19***			
31:24	bin19	R/W	0	8-bit signed offset from a Yout=Yin Curve for Yin=64+ped register
23:16	bin18	R/W	0	8-bit signed offset from a Yout=Yin Curve for Yin=48+ped register
15:8	bin17	R/W	0	8-bit signed offset from a Yout=Yin Curve for Yin=32+ped register
7:0	bin16	R/W	0	8-bit signed offset from a Yout=Yin Curve for Yin=16+ped register
Offset 0x10 E338	***Layer LUT-HIST Bins 20 to 23***			
31:24	bin23	R/W	0	8-bit signed offset from a Yout=Yin Curve for Yin=128+ped register
23:16	bin22	R/W	0	8-bit signed offset from a Yout=Yin Curve for Yin=112+ped register
15:8	bin21	R/W	0	8-bit signed offset from a Yout=Yin Curve for Yin=96+ped register
7:0	bin20	R/W	0	8-bit signed offset from a Yout=Yin Curve for Yin=80+ped register
Offset 0x10 E33C	***Layer LUT-HIST Bins 24 to 027***			
31:24	bin27	R/W	0	8-bit signed offset from a Yout=Yin Curve for Yin=192+ped register
23:16	bin26	R/W	0	8-bit signed offset from a Yout=Yin Curve for Yin=176+ped register
15:8	bin25	R/W	0	8-bit signed offset from a Yout=Yin Curve for Yin=160+ped register
7:0	bin24	R/W	0	8-bit signed offset from a Yout=Yin Curve for Yin=144+ped register
Offset 0x10 E340	***Layer LUT-HIST Bins 28 to 31***			
31:24	bin31	R/W	0	8-bit signed offset from a Yout=Yin Curve for Yin=256+ped register
23:16	bin30	R/W	0	8-bit signed offset from a Yout=Yin Curve for Yin=240+ped register
15:8	bin29	R/W	0	8-bit signed offset from a Yout=Yin Curve for Yin=224+ped register
7:0	bin31	R/W	0	8-bit signed offset from a Yout=Yin Curve for Yin=208+ped register
Offset 0x10 E344	***Layer Histogram Control***			
31:14	Unused		-	
13	enable	R/W	0	Histogram and black stretch enabled 1 = enabled 0 = bypassed

Table 20: QVCP 1 Registers *...Continued*

Bit	Symbol	Access	Value	Description
12:11	uv_gain	R/W	2	Gain Factor for UV correction 00 = factor of 0 01 = factor of 0.5 10 = factor of 1 11 = factor of 2
10	uv_pos	R/W	1	UV corrections only in positive direction
9	ratio_limit	R/W	1	Minimum denominator for UV processing 0 = denominator larger than or equal to 64 1 = denominator larger than or equal to 128
8	round	R/W	0	Round or Truncate in interpolation for Y transfer function 1 = Round 0 = Truncate
7:0	black_off	R/W	0	8-bit signed offset for black stretch value This is the 33rd histogram variable and this is the only way to add an offset from a Yout=Yin curve for Yin= -256+ped register. But it affects all other 32 values too.
Offset 0x10 E348	***Layer CFTR Blue***			
31:25	Unused		-	
24	blueycomp	R/W	1	Compensate Y in order to prevent illegal colors in RGB space 1 = Y compensation on 0 = Y compensation off
23:20	bluegain	R/W	A	Strength of blue stretch effect (0..15) higher value = greater effect
19:17	bluesize	R/W	4	Blue stretch detection area (0..7) lower value = greater detection area
16	blue_enable	R/W	0	Blue stretch functionality 1 = Enable 0 = Bypass
15:9	Unused		-	
8:6	skingain	R/W	2	Strength of skin tone correction effect (0..4) higher value greater effect
5:3	skintone	R/W	2	Direction of correction (0..4), lower value = towards “yellow” higher value = towards “red”
2:1	skinsize	R/W	1	Skin tone detection area size (0..2) higher value = greater detection area
0	skin_enable	R/W	0	Skin tone correction functionality 1 = Enable 0 = Bypass
Offset 0x10 E34C	***Layer CFTR Green***			
31:15	Unused		-	
14:11	greenmax	R/W	9	Maximum correction(0..15), higher value = stronger correction allowed
10:8	greensat	R/W	4	Green detection area maximum saturation (0..7) higher value = effect extends to higher saturations

Table 20: QVCP 1 Registers *...Continued*

Bit	Symbol	Access	Value	Description
7:4	greengain	R/W	7	Strength of green enhancement effect (0..15) higher value = greater effect
3:1	grrensize	R/W	0	Green detection area minimum saturation (0..7) lower value = greater detection area
0	green_enable	R/W	0	Enable green enhancement functionality 1 = Enable 0 = Bypass
Offset 0x10 E350	***Layer DCTI Control***			
31:16	Unused		-	
15	superhill	R/W	1	Superhill mode, avoid discolorations in transients within a colour component.
14:11	threshold	R/W	4	Immunity against noise
10	separate	R/W	0	Common or separate processing of U and V signals 1 = Separate 0 = Common (1 is the recommended value as it works better)
9	protection	R/W	1	Hill protection mode, no discolorations in narrow colour gaps
8:6	limit	R/W	7	Limit for pixel shift range 0-6 = (limit+1)*2 7 = 15
5:2	gain	R/W	8	Gain Factor on sample shift gain/16 (0/16..15/16)
1	ddx_sel	R/W	1	Selection of simple or improved first differentiating filter 1 = Improved 0 = Simple
0	enable	R/W	0	Enable DCTI functionality 1 = Enable DCTI 0 = Bypass DCTI
Offset 0x10 EFE0	***Interrupt Status QVCP***			
31:12	Unused		-	
11	LAYER_DONE	R	0	The layer has been completely displayed (layer 2)
10	BUF_DONE	R	0	DMA channel is done fetching all data for the current layer (layer 2)
9	FCU_UNDERFLOW	R	0	Underflow in FCU FIFO for layer 2
8	Unused			
7	LAYER_DONE	R	0	The layer has been completely displayed (layer 1)
6	BUF_DONE	R	0	DMA channel is done fetching all data for the current layer (layer 1)
5	FCU_UNDERFLOW	R	0	Underflow in FCU FIFO for layer 1
4	Unused			
3	VINTB	R	0	Vertical line interrupt issued if Y position matches VLINTB
2	VINTA	R	0	Vertical line interrupt issued if Y position matches VLINTA
1	VBI_DONE_INT	R	0	VBI/Register load is done with the current packet list

Table 20: QVCP 1 Registers *...Continued*

Bit	Symbol	Access	Value	Description
0	VBI_PACKET_INT	R	0	VBI/Register reload has sent a packet with the IRQ request bit set in the packet header
Offset 0x10 EFE4	***Interrupt Enable QVCP***			
31:24	Unused		-	
23:0	Interrupt Enables	R/W	0	A '1' in the appropriate bit will enable the interrupt according to the specification in register 0xFE0.
Offset 0x10 EFE8	***Interrupt Clear QVCP***			
31:24	Unused		-	
23:0	Interrupt Clears	W	0	A '1' in the appropriate bit will clear the interrupt according to the specification in register FE0.
Offset 0x10 EFEC	***Interrupt Set QVCP***			
31:24	Unused		-	
23:0	Interrupt Sets	W	0	A '1' in the appropriate bit will set the interrupt according to the specification in register FE0.
Offset 0x10 EFF4	***Powerdown***			
31	Powerdown	R	0	This bit has no effect i.e., there is no powerdown implemented for this module.
30:0	Unused		-	
Offset 0x10 EFFC	***Module ID***			
31:16	Module ID	R	0xA052	Unique revision number
15:12	REV_MAJOR	R	0	Major revision counter
11:8	REV_MINOR	R	1	Minor revision counter
7:0	APP_SIZE	R	00	Aperture Size 0 = 4 kB

Chapter 12: Video Input Processor

PNX15xx Series Data Book – Volume 1 of 1

Rev. 2 — 1 December 2004 Product data sheet

1. Introduction

The Video Input Processor (VIP) handles incoming digital video and processes it for use by other components of the PNX15xx Series. This enables applications such as picture-in-picture and video teleconferencing on the TV screen.

1.1 Features

The VIP provides the following functions:

- Receives digital video data from the video port. The data stream may come from a device such as the TDA8751, which can digitize analog video from any source and convert a digital signal from a DVI interface/source into parallel YUV format
- Features 8/10-bit single channel (single-stream) and 16/20-bit dual channel (dual-stream) capture of CCIR601 YUV 4:2:2 video input with embedded or explicit syncs, supported by a maximum clock frequency of 81 MHz. The DUAL_STREAM mode is used to capture a 16 or 20-bit HD stream where 8/10-bit Y and 8/10-bit multiplexed U/V data are received and captured on two separate channels.
- Provides video and auxiliary (AUX, ANC, or RAW) data acquisition and capture.
  - Provides separate acquisition windows for video and for VBI data (cannot be used if the output format is planar data.
  - Implements two identical Dither units capable of either dithering or rounding 9- or 10-pixel components in video mode.
  - Enables raw data capture in either 8 or 10 bits for single_stream mode and 8 bits of DUAL_STREAM mode.
  - Enables ANC header decoding or window mode for VBI data extraction.
- Performs horizontal scaling, cropping and pixel packing on video data from a continuous video data stream or a single field or frame.
  - Performs horizontal down-scaling or zoom-up by 2x, the upscaling being possible only in the single-stream mode.
  - Enables linear horizontal aspect ratio conversion using normal or transposed 6-tap polyphase filter.
  - Enables non-linear horizontal aspect ratio conversion using normal 6-tap polyphase filter.

  - Permits optional linear phase interpolation / nonlinear phase interpolation (as in MBS).

- Allows color-space conversion (mutually exclusive with scaling) on the video path.
- Allows 4:2:2 to 4:4:4 conversions on the video path.
- Provides last-pixel-in signals, for VBI and video data, to the GPIO block for Timestamping.
- Features interrupt generation, for VBI or video data written to memory.
- Provides an internal Test Pattern Generator with NTSC, PAL, and variable format support.
- Features a wide variety of output formats such as planar YUV 4:4:4, planar YUV 4:2:2, planar RGB, semi-planar YUV 4:2:2 packed UYVY, etc. Planar formats are mutually exclusive with the VBI capture.

2. Functional Description

2.1 VIP Block Level Diagram

The main functional blocks of the VIP and the primary data paths (not including syncs etc.) are shown in Figure 1.

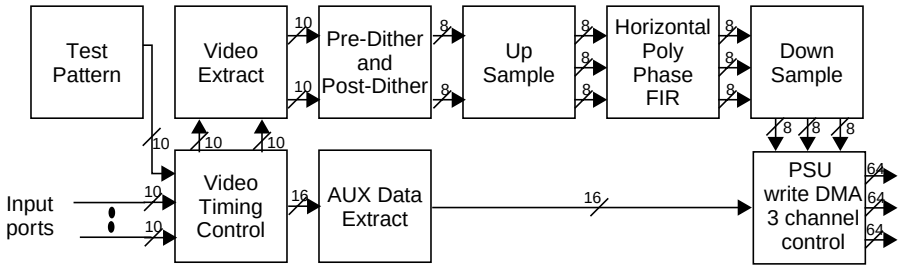


Figure 1: Simplified VIP Block Diagram

A brief description of each of the submodules is given in Table 1.

Table 1: VIP Submodule Descriptions

Submodule	Brief Description of Functionality
Test Pattern	An internal generator that produces 4:2:2 NTSC/PAL video streams
Video Timing Control	This submodule receives incoming data samples from either the Test Pattern Generator or the Digital Video Port. A tally of the sample is maintained when it conforms to the ITU-R 656 or ITU-R 1364. Video and Aux samples are forwarded to Video Extract and Aux Data Extract respectively.
Video Extract	Video input pipe windower. This submodule: receives video samples from Video Port Input module. captures desired samples in a programmable size rectangular area (window). forwards captured samples to the Pre-Dither unit.

Table 1: VIP Submodule Descriptions *...Continued*

Submodule	Brief Description of Functionality
Pre-Dither and Post-Dither	• There are two identical Dither units: Pre-Dither and Post-Dither, capable of 10->9, 10->8 and 9->8 dithering/rounding of the video data only. The recommended mode is to have rounding for 10->9 and dithering for 9->8
Up Sample	• 4:2:2 to 4:4:4 Interpolation FIR Filter for chroma upsampling 8-bit video samples are received from Post-Dither.
Horizontal Poly Phase FIR	• Horizontal scaler pipeline
Down Sample	• 4:4:4 to 4:2:2 Decimation FIR Filter for chroma down sampling
AUX Data Extract	Video input pipe AUX windower. This submodule: receives aux samples from Video Port Input module. captures desired samples in a programmable captured window and/or within a buffer space. captures ANC packet with matching DID captures all valid input samples forwards the captured samples to Pixel Packer.
3 Channel Write DMA Control	• An interface to the memory agent

2.2 Chip I/O and Connections

Figure 2 sketches the input pins of the VIP module. Refer to Chapter 3 System On Chip Resources, Section 7. on page 3-16 for the mapping of the VIP I/O signal with the PNX15xx Series I/O pins.

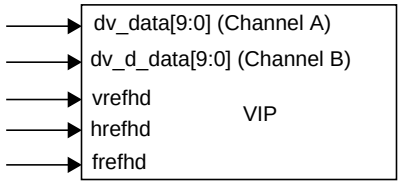


Figure 2: VIP Module Interface

2.2.1 Data Routing and Video Modes

The VIP can be operated in three different modes.

SD Video Mode

The interleaved data (YUV) is captured from the dv_data[9:0] input, also called Channel A. The dv_d_data, also called Channel B, is not used in the SD mode.

HD Mode

The Y data is expected on dv_d_data[9:0] (Channel B) and U/V data is expected on dv_data[9:0] (Channel A).

RAW MODE

In RAW mode the data can be captured from Channel A or B.

2.2.2 Input Timing

A separate signal, dv_valid, is provided to validate all incoming data. The relationship between dv_valid and data, with reference to clock, is shown in Figure 3.

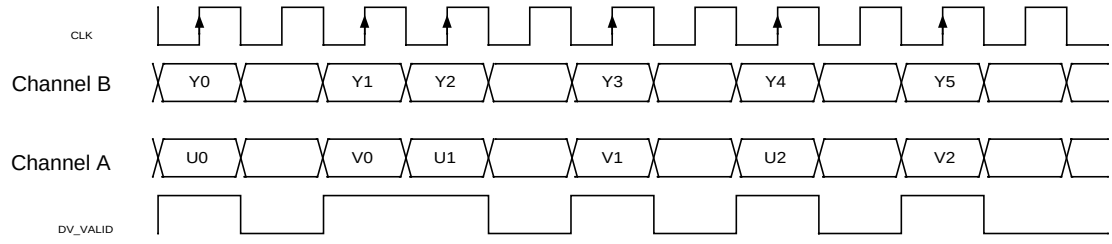


Figure 3: Digital Video Input Port Timing Relationships in HD Mode

2.3 Test Pattern Generator

The Test Pattern Generator produces a video stream with a pixel frequency of half the VIP input clock e.g., the 27 MHz encoder clock by programming the clock selection block accordingly.

The sync generation is NTSC-like, with 525 lines per frame and 858 pixels per line. The active video range is 720x462 bordered by a white frame.

The test pattern is shown in Figure 4, and contains the following elements:

- A white 2-pixel wide frame—size 720x462
- A color bar—white 100%, yellow 75%, cyan 75%, green 75%, magenta 75%, red 75%, blue 75%, and black 0%.
- A grey ramp—full value range 0–255
- A vertical multiburst
- A horizontal multiburst—first rectangle solid in odd, second solid in even field
- Vertical lines
- A moving cursor
- Test pattern

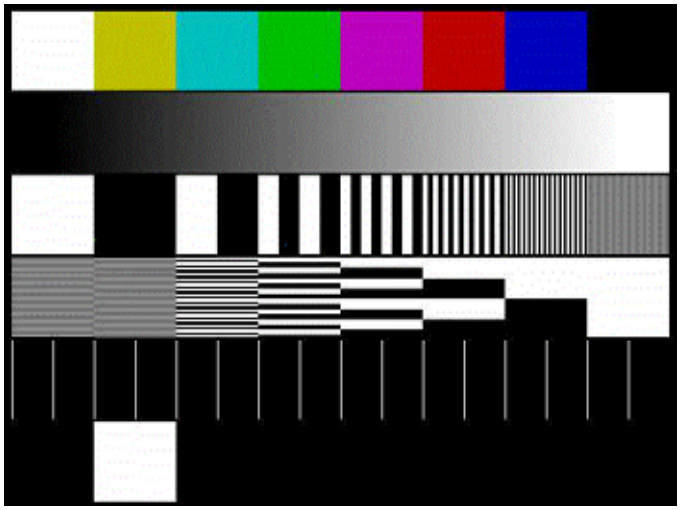


Figure 4: Test Pattern

To capture a picture using the build-in test pattern generator (odd and even field), set up the registers as shown in Table 2 to start capturing at the upper left corner of the white frame.

Table 2: Test Pattern Generator Setup

Mode	Reference	Window Start (x,y)	Window End (x,y)
NTSC	HREF- / VREF+	8a,0 (138,0)	359,f1 (857,241)
PAL	HREF- / VREF+	90,0 (144,0)	35f,11f (863,287)

2.4 Input Formats

The VIP accepts the following external video input streams:

- 8/10-bit data with encoded [EAV/SAV] syncs YUV 4:2:2 (alias D1 mode)
- 8-bit data with external [HREF, VREF] syncs YUV 4:2:2 (alias VMI mode)
- 8/10-bit or 16/20-bit raw data samples (alias RAW mode)
- 16/20-bit video data on 2 groups of pins for Y and multiplexed U/V with both encoded [SAV/EAV] and explicit [hrefhd, vrefhd, frefhd] syncs (alias DUAL_STREAM or HD mode)

The YUV 4:2:2 sampling scheme assumed by all modes is defined by CCIR 601.

D1 Mode

The D1 Mode expects an 8/10-bit 4:2:2 video data stream (defined by CCIR 656) with syncs encoded in the video data stream.[1] Timing reference codes recognized are 80h, 9Dh, ABh, B6h, C7h, DAh, ECh and F1h. Single bit errors in the reference codes are corrected, but double bit errors are rejected. The supported mode is shown in Figure 5.

1. For compatibility with 8-bit D1 interfaces the two LSBs are not used for timing reference extraction (as defined in CCIR 656-2).

This is strictly a single-stream mode, where VIP captures on Channel A (dv_data[9:0]) either 10-bit or 8-bit (MSB aligned, with dv_data[1:0] unused) multiplexed YUV video data with embedded syncs. The DUAL_STREAM register should be programmed to 0 in this mode.

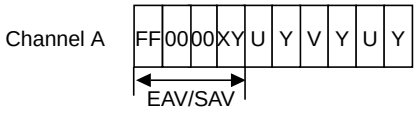


Figure 5: D1 Data Stream

VMI Mode

The VMI Mode is an 8-bit YUV (4:2:2) mode with external horizontal and vertical reference signals, which follows the VMI protocol. Chrominance and luminance input samples are multiplexed into a single 8-bit input data stream on Channel A. The Field Identifier (FID) is derived from the horizontal and vertical sync timing relation.

This is also a single-stream mode, where VIP captures on Channel A (dv_data[9:0]) 8-bit VMI data with explicit syncs, where dv_data[9:2] correspond to VMI data and dv_data[1:0] correspond to VREF and HREF respectively. The DUAL_STREAM register should be programmed to 0 in this mode.

RAW Mode

In Raw Mode, valid 8-bit or 10-bit data are continuously captured and written into system memory. Both single and dual streams are supported in this mode. The DUAL_STREAM register can, therefore, be programmed to either 1 or 0 in this mode.

In the single stream mode (DUAL_STREAM register = 0), 8-bit data (dv_data[9:2]) is captured as it is but 10-bit data (dv_data[9:0]) is extended to 16 bits by either adding leading zeros or by sign-extension.

In the dual stream mode (DUAL_STREAM register = 1), only 8 MSBs of the 10-bit data are valid for each of the 2 channels: A and B. Two 8-bit data, dv_data[9:2] and dv_d_data[9:2], are captured simultaneously from the 8 upper bits of both the channels, for both 8-bit or 10-bit modes, and packed into one 16-bit entity. Channel A and Channel B data occupy the 8 LSBs and 8 MSBs respectively, of the packed 16-bit result.

Raw Mode is only available in the auxiliary capture path of the VIP. It can be enabled independent of D1 or VMI mode.

Remark: RAW mode may not be supported in next PNX15xx Series generations.

HD Mode

This is strictly a DUAL_STREAM mode (DUAL_STREAM register = 1), where VIP expects 10-bit Y and 10-bit U/V data on 2 separate inputs (Channels A and B); the U/V data is time multiplexed. In order to support a number of external decoders, this mode has been implemented to work not only with embedded or encoded syncs where EAV and SAV codes are embedded in the data, but also with explicit syncs where the synchronization reference is provided explicitly via HREF, VREF, and FREF

signals (as specified in the implementation of the TDA8751 decoder from Philips and the HMP8117 decoder from Intersil). In HD mode HREF, VREF, and FREF are respectively connected to the VIP module pins hrefhd, vrefhd an frefhd. The supported mode is shown in Figure 6. Note that for detecting the embedded sync in this HD mode, the code is expected to be in the U/V stream; to this end, the current design checks only one of the streams, the U/V stream, for the presence of the embedded codes, assuming that any information embedded in the Y stream is identical (see ITU BT 1120, SMPTE 274M standards). The DUAL_STREAM register must be programmed to 1 in this mode.

Remark: The explicit sync signals are used only in the HD or DUAL_STREAM mode.

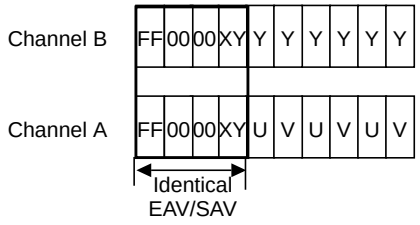


Figure 6: HD Dual Data Stream

Table 3 tries to capture the above discussion into a quick checklist of implemented input formats, where an X designates the presence (support) of the corresponding feature.

Table 3: Video Input Formats

		Single Stream (YUV)			Dual Stream (Y and U/V)		
Video Modes		**Embedded Sync**	**Explicit Sync**	**No Sync**	**Embedded Sync**	**Explicit Sync**	**No Sync**
D1	8-bit	X					
	10-bit	X					
VMI	8-bit		X				
	10-bit						
RAW	8-bit			X			X
	10-bit			X			X
HD	8-bit				X	X	
	10-bit				X	X	

2.5 Video Data Path

The relation between the video input formats and the supported data stream is shown in Table 4.

Table 4: Relationship Between Input Formats and Video Data Capture

Input Modes		Single Stream (YUV) Video Data	Dual Stream (Y and U/V) Video Data
D1	8-bit	X	
	10-bit	X	
VMI	8-bit	X	
HD	8-bit		X
	10-bit		X

2.5.1 Video Data Flow

The video datapath dataflow for the VIP is shown in Figure 7.

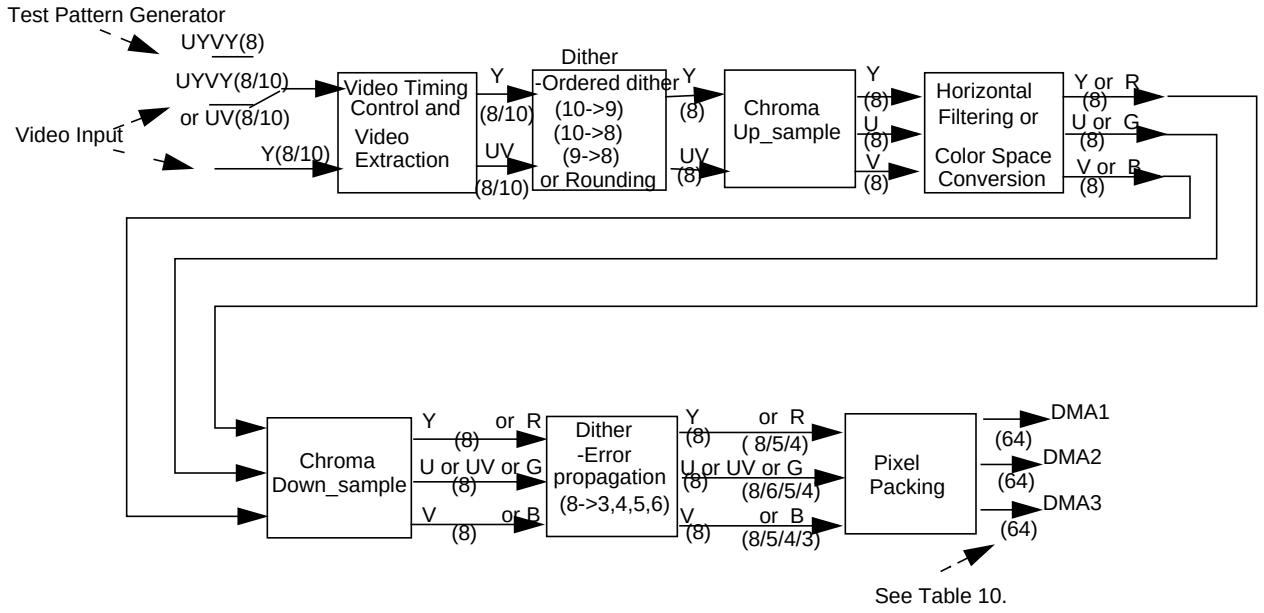


Figure 7: Video Data Flow

2.5.2 Video Data Acquisition

The Video and Auxiliary Data Extract block, shown in Figure 1, receives a continuous pixel stream from the Video Timing Control block and outputs active window data and synchronization signals. Bit fields in the windowing registers specify the start and end of the source windows relative to the reference edges of H and V syncs and size of the target windows.

2.5.3 Internal Timing

Window start is defined relative to either the rising or falling edges of the VREF and HREF inputs (or similar D1 events), as shown in Figure 8.

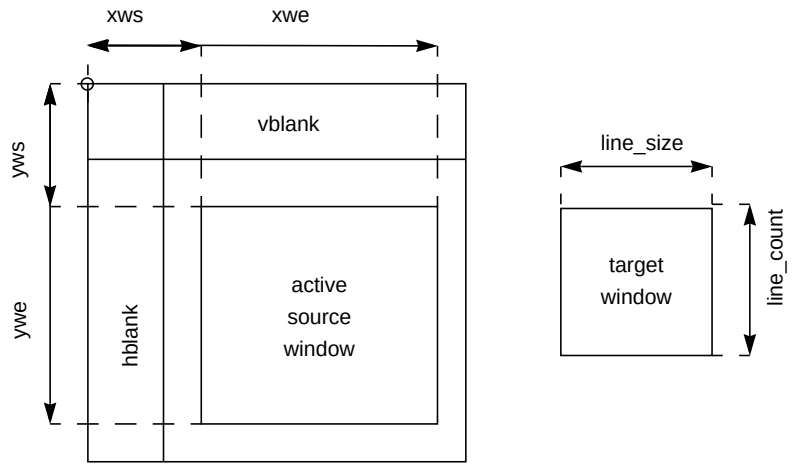


Figure 8: Source and Target Window Parameters

The first qualified data aligned with the REHS reference edge is interpreted as a U sample. If the UYVY data stream is out of sync, it can be realigned with the *vsra* bits in register 00100.

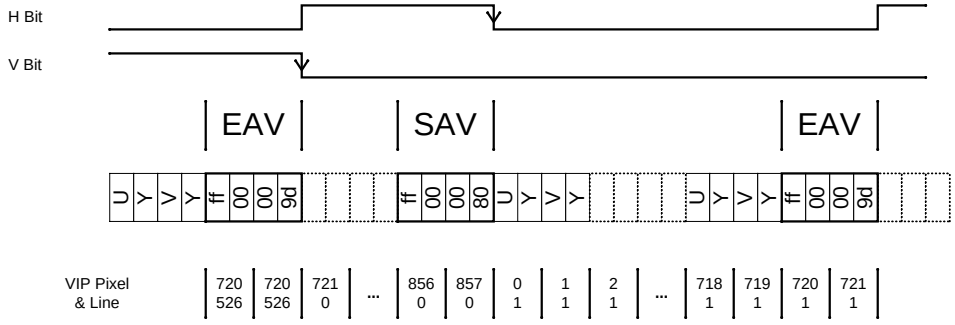


Figure 9: Acquisition Window Counter Reference

For an example showing how to setup the windower and scaler to capture the entire test pattern, refer to Table 2, Figure 8, and Figure 9.

2.5.4 Field Identifier Generation

The Field Identifier in the D1 mode is extracted from the F bit in every valid video header, whereas in the VMI mode, the same is derived from the value of the HREF signal during the negative edge of the VREF signal. The Field Identifier timing is illustrated in Figure 10, and Table 5 shows various Field Identified generation modes. Note that instead of using the Field Identifier derived from the video stream, it can also be forced to zero or forced to toggle after each new incoming field; the forced

value takes effect after the selected vertical reference edge occurs at the input. The *SF* bit controls how the Field Identifier value is interpreted. Any change of the Field Identifier interpretation takes effect immediately.

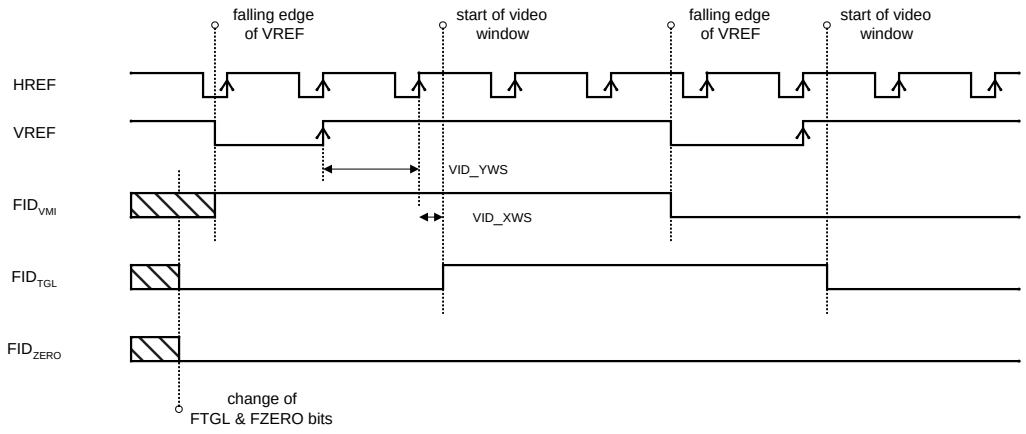


Figure 10: Field Identifier Timing

Table 5: Field Identifier Generation Modes

VDI8	HREF	VSEL	FZERO	FTGL	Change at	FID
x	f	VMI	0	0	negedge VREF	!f
f	x	D1	0	0	valid D1 Header	f
x	x	x	1	0	immediately	0
x	x	x	x	1	immediately*	0,1,0,..

* FID toggles after detection of video window start.

FID	FSWP	Meaning
0	0	Odd
1	0	Even
0	1	Even
1	1	Odd

Video Acquisition Window

The start location of the window to be captured, relative to the input stream, is specified in the Window Start registers, 00140 (*VID_XWS, VID_YWS*).

The stop location of the window to be captured, relative to the input stream, is specified in the Window End register, 00144 (*VID_XWE, VID_YWE*).

Additional Target window cropping, which might be necessary after scaling, can be done with the *LINE_SIZE* and *LINE_COUNT* values in the Target Window Size register, 00304.

Dithering of the Video Data

There are two identical Dither units, Pre-Dither and Post-Dither, capable of 10->9, 10->8 and 9->8 dithering/rounding with saturating values. These two dither units are cascaded together on the video data path. The two dither units are independent of each other, and controlled with separate MMIO registers. Input samples are assumed to be left (MSB) aligned on the 10 bit input bus. Output samples are left aligned (MSB) on the 10 bit output bus. Both Dither units need to be disabled for an 8-bit input data stream, to avoid unexpected results.

The Dither units can be used when the bit precision of samples needs to be limited, while preventing quantization effects in areas with almost uniform levels of shades in a picture.

The following discussion refers to a single Dither unit, either Pre- or Post-Dither

The dither unit processes up to 10 bit inputs. It receives all the three components, Y, U and V, on two 10 bit input buses, and dithers/rounds them down to 8 or 9 bits. Dithering can be enabled separately for luma (Y) and chroma (U and V) components. If the dither unit is enabled but dithering is disabled, rounding, instead of dithering, is performed.

Whenever dithering is enabled, the dithering process alternates its activity between adjacent pixels: either every pixel or every two pixels. Furthermore, any combination of three alternation patterns can be selected: line, field, and frame alternations.

The Dither units are controlled by QVI_PRE_DITHER_CTRL and QVI_POST_DITHER_CTRL MMIO registers, for the pre- and post-dither units, respectively. Immediately after the unit is enabled, it waits for the beginning of the following captured image before it actually starts to operate.

Enabling the dither unit resets its internal state.

Dither Mechanism

The operation mode is programmable via MODE in the dither-unit control register. The three available modes are:

- 10-bit input down to 8 bits of output
- 10-bit input down to 9 bits of output
- 9 bit input down to 8 bits of output.

Remark: 8-bit input samples are not changed when passed through the Dither unit (the 8 output MSBs are identical to the 8 input MSBs, but the 2 output LSBs are changed by the dither unit).

The Dither unit independently dithers all the three components Y, U, and V in the same way. Each input pixel is processed independently in the sense that the value of the other inputs do not affect the processing of the current input.

The unit is enabled with DITHER_ENABLE. The programmer can select which components are dithered; with DITHER_Y for the luma components, and DITHER_UV, independent of Y, for the chroma components. When the dither unit is enabled, a component that is not selected for dithering goes through rounding. The final value of all components is saturated at 1023, which is the largest value represented by the 10 bit output.

Whenever the dithering operation is enabled, the process of dithering alternates between successive pixel-components, either every pixel or every two pixels, in the same image line. This option is programmable with DOUBLE_PIXEL_ALT for Single or Double pixel alternation.

There are another three dithering options that can be enabled or disabled independently: alternate processing between successive lines, fields and frames.

Enabling the Dither Units

Immediately after the Dither unit is enabled or after a reset, the unit waits for the beginning of a newly-captured image. Only then the unit starts dithering.

Once the Dither unit is operational (enabled), it keeps track of the order in which the images arrive: we refer to the very first image at the unit dither as the *even* image, the second image as the *odd* image, and so on. A frame here is defined as two images: an even image followed by an odd image. This maintained state does not depend on the selected alternation options, it is maintained as long as the Dither unit is enabled. Any alternative activity corresponds to the internally-maintained state of a frame and field (even or odd) and has nothing to do if the signal is coming from the top or the bottom field.

Dithering operation also distinguishes between even and odd pixel-components of the same type (either Y, U or V) in a line.

The first occurrence of a Y or U or V component in the first line in the first received image is considered to be an **even** occurrence (or *set*).

2.5.5 Horizontal Video Filters (Sampling, Scaling, Color Space Conversion)

Interpolation Filter (Upsampling)

All horizontal video processing is based on equidistant sampled components. All 4:2:2 video streams, therefore, have to be upsampled before being scaled horizontally. The interpolation FIR filter used can interpolate interspersed or co-sited chroma samples. Mirroring of samples at the field boundaries compensate for run-in and run-out conditions of the filter.

The following coefficients are used:

- co-sited: A=(1) and B=(-3,19,19,-3)/32
- interspersed: C=(-1,5,13,-1)/16 and D=(-1,13,5,-1)/16

Decimation Filter (Downsampling)

After horizontal processing, the chrominance may be down-sampled to reduce memory bandwidth or allow a higher-quality vertical processing not available otherwise. Mirroring of samples at the field boundaries compensate for run in and run out conditions of the filter.

The following coefficients are used:

- co-sited: low pass A=(1,2,1)/4 or sub-sample A=(0,1,0)
- interspersed: B=(-3,19,19,-3)/32

Normal Polyphase Filter (Horizontal Scaling)

The normal polyphase filter can be used to zoom up (upscale) or downscale a video image. Depending on the number of components, the filter is used with 6 taps (three-component mode) or 3 taps (four-component mode).

Color Space Matrix Mode

In addition to normal and transposed polyphase filtering (scaling), the FIR filter structure can instead be programed to perform color space-conversion. A dedicated set of registers holds the coefficients for the color-space matrix. Horizontal scaling and color space conversion are mutually exclusive.

2.5.6 Video Data Write to Memory

The VIP can produce a variety of output formats. Video formats range from a single-component up to three-component formats (like a 4:4:4 YUV). Up to three write planes can be defined. On the input, the video format is restricted to YUV 4:2:2 as defined in ITU-R-656 or 8/10 raw data. On the output, true color and compressed formats are supported. For a complete list of supported video formats, refer to Section 3. Register Descriptions.

The Pixel Packing Unit takes care of quantization and packing of the color components into 64-bit units. A list of the most common video formats supported is shown in Table 6. Packing of a pixel into 64-bit units is always done from right to left while bytes within one pixel unit are ordered according to the endian mode settings (specified by the global endian mode register; endian mode bit in the output format register can, however, invert that signal).

Table 6: Output Pixel Formats

Format	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
planar YUV (4:4:4, 4:2:2) or RGB	plane #1																								Y8 or R8							
	plane #2																								U8 or G8							
	plane #3																								V8 or B8							
semi planar YUV (4:2:2)	plane #1																								Y8 or R8							
	plane #2																								U8/V8							
packed 4/4/4 RGBa																	alpha				R4				G4				B4			
packed 4/5/3 RGBa																	alpha				R4				G5					B3		
packed 5/6/5 RGB																	R5					G6						B5				
packed YUY2 4:2:2																	U8 or V8								Y8							
packed UYVY 4:2:2																	Y8								U8 or V8							
packed 888 RGB(a)	(alpha)								R8 or Y8								G8 or U8								B8 or V8							
packed 4:4:4 VYU(a)	(alpha)								V8								Y8								U8							

Table 6 shows the location of the first 'pixel unit' within a 64-bit word in the little endian mode. The selected endian mode will affect the position of the components within a multi-byte pixel unit!

Remark: VIP does not explicitly support a 4:2:0 memory format. Such a format can be obtained by discarding partial data written to memory.

Capture Enable Mode

Using the *cfen* bits, video capture can be limited to odd or even or both fields. If both fields are to be captured, the capture starts with the next odd field.

The status of the *osm* (one-shot) bit in the mode-control-register specifies the capture mode (one-shot or continuous):

- If *osm*=0, the corresponding incoming video stream is captured continuously. For example, in a video conference application the vanity image would be a continuous stream to the frame buffer.
- If *osm*=1, the corresponding incoming video stream is captured one field or frame at a time (depending on the *cfen* bits).

Programming hint: In a video conference application the captured image would be a one-shot stream to the host memory. If you write *osm*=1 and select field/frame in the register, it is captured on the next VSYNC and *cfen* bits are cleared to 0. To capture the next image, the *cfen and osm* bits must be reprogrammed.

Address Generation

The line address is generated by loading the base address from the corresponding register set at the beginning of each field and adding the line pitch to it at the beginning of every new line.The lower three bits of the first three base address registers are used as an intra-long-word offset for the left-most pixel components of each line. The offset has to be a multiple of the number of bytes per component.

Double Buffer Mode

To avoid line tear caused by trying to display a frame at the same time that it is being updated, a double buffer mode is available. In this double buffer mode, a second set of DMA base addresses is available. After capturing and storing one complete frame in the location described by one set, the other set is used for the next frame. The idea is illustrated in Figure 11.

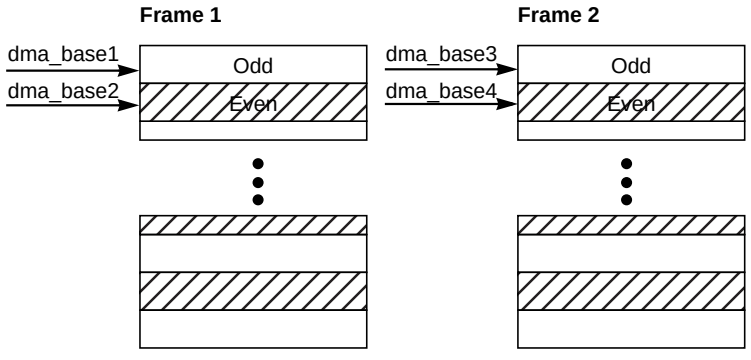


Figure 11: Double Buffer Mode

2.5.7 Auxiliary Data Path

The relationship between the input data modes and the supported auxiliary capture modes is shown in Table 7.

Table 7: Relationship Between Input Formats and Data Capture

		Single-Stream Auxiliary Data			Dual-Stream Auxiliary Data		
Input modes		**AUX**	**ANC**	**RAW**	**AUX**	**ANC**	**RAW**
D1	8-bit	X	X				
	10-bit	X	X				
VMI	8-bit	X			X		
RAW	8-bit			X			X
	10-bit			X			X
HD	8-bit				X		
	10-bit				X		

Auxiliary Data Flow

The auxiliary data flow is shown in Figure 12.

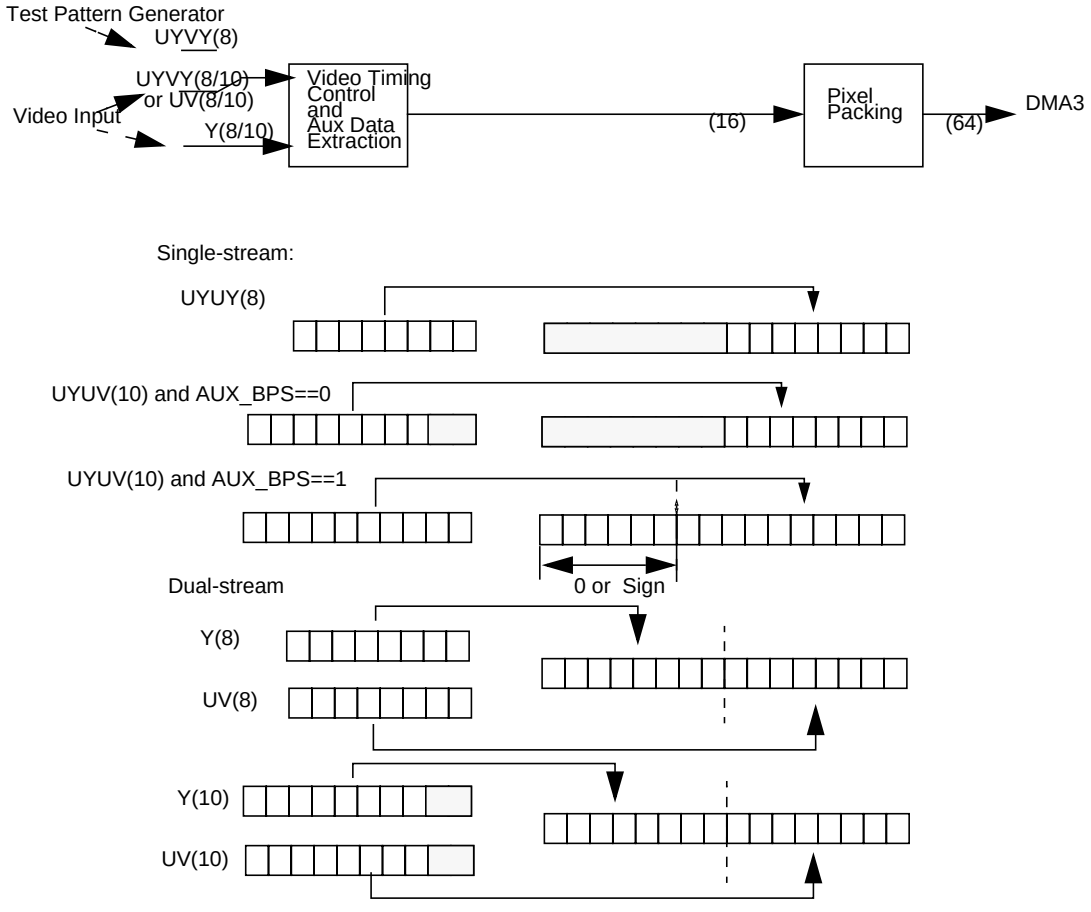


Figure 12: Auxiliary Data Flow

Auxiliary Data Acquisition

Capturing auxiliary data utilizes the same DMA engine used for the third video plane. Capture of overlapping Video and Auxiliary regions is, therefore, only possible when semi-planar or packed formats are being used. Data can be captured in either 8- or 10-bit modes. In the single-stream mode, 10-bit data is extended to 16 bits by either adding leading zeros or by sign-extension. In dual stream mode, only 8 MSBs of a 10-bit data are valid; two such MSB sets (2x8-bit data) are captured simultaneously, for either 8-bit or 10-bit modes, and packed to form a resultant 16-bit unit. thus, Channel A data (8 bits) and channel B data (8 bits) are located at the 8 LSBs and the 8 MSBs respectively, of the packed 16-bit data.

Three different types of auxiliary data capture are defined:

- Ancillary Data Capture (ANC)
- Auxiliary data acquisition window (AUX)
- Raw data capture (RAW)

A buffer-size register can be used to limit data acquisition by size (one shot mode) or define a ring-buffer length.

Even though ANC and AUX capture can be enabled separately, simultaneous capture of ANC and AUX is not advisable. Timing and sequence of ANC and AUX data are not necessarily related and therefore are likely to lead to unpredictable results if simultaneous capture is attempted!

Ancillary Data Capture

Ancillary Data, embedded in the stream and marked by ITU-R-1364 header codes, can be decoded and extracted for software processing (see Figure 13). AUX_BPS register specifies the number of bits to be captured per ANC sample. In the 8-bit mode, two LSBs of the 10-bit data bus are ignored. Ancillary data capture is not supported in the dual stream mode.

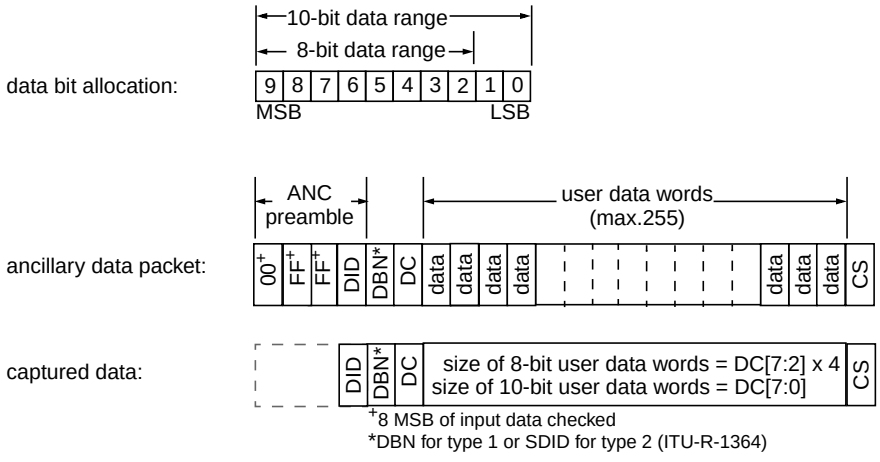


Figure 13: ANC Data Structure

Four sequential ANC preamble bytes, 00, FF, FF and a qualified DID word, enable ANC data capture. A qualified DID word is defined:

- masked AUX_ANC-enabled ID matched (see Figure 13)
- bit 8 is even parity for bit 7-0(10-bit data mode) / bit 7-2(8-bit data mode)
- bit 9 = not bit 8

2 LSBs of both ID_MASK_0 and ID_MASK_1 need to be programmed to 2'b00 in 8-bit data mode to prevent unexpected results.

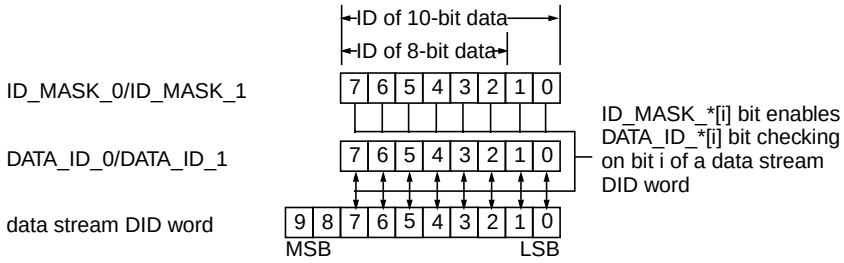


Figure 14: ANC Masked ID Checking

In the type 1 case, the data block number (DBN) distinguishes successive data packets with a common data ID. In the type 2 case, the DID is followed by the secondary identification (SDID). The captured packed length is taken form the data count (DC) byte.

A value of DC=0 will capture exactly four data words consisting of DID, DBN (or SDID), DC and checksum (CS). If DC is not equal to 0, additional user data words defined by DC are captured. Parity bits for DBN (or SDID) and DC bytes are not checked.

Auxiliary Data Acquisition Window

The auxiliary data acquisition window can be used to capture either VBI data or an additional region of video data. It provides yet another capture-window. The field identifier is compared against *AUX_CFEN* bits at the start of the programmed window to control whether a field is captured or not. The start and end points of the auxiliary window are defined by the AUX Window Start and End registers at offsets 00180 and 00184 (*AUX_XWS, AUX_YWS, AUX_XWE, AUX_YWE*).

The *AUX_XWS parameter* specifies the number of the first pixel to be captured after the HREF reference edge.The *AUX_YWS* parameter specifies the number of the first line to be captured after AUX reference edge.

The *AUX_XWE* parameter specifies the number of the last pixel to be captured.The *AUX_YWE* parameter specifies the number of the last line to be captured.

Pixel and lines start counting at 0.

Raw Data Capture

Raw data capture overrides ANC or AUX data capture modes when enabled. In this raw capture mode, any validated data at the video port is captured regardless of external or embedded synchronization signals.

AUX Data Write to Memory

Auxiliary data capture formats, for writing into the frame buffer, are limited to raw luma and chroma samples in 8 or 10 bit formats (extended to 16 bit). Optionally, writing of chroma samples can be omitted.

Capture Enable Mode

The *aux_cfen* bits specify the fields from which the device is to capture the AUX data. If cfen=0, no auxiliary data is captured. Once the capture of an auxiliary window has started, resetting these bits has no effect until the end of the video window.

The *aux_anc* bits specify the type of ancillary data blocks to be fetched. Once the capture of an ANC block has started, resetting of these bits has no effect until the end of the data block.

The *aux_raw* bit enables continuous capturing of raw samples regardless of external or internal syncs. If raw capture is enabled, *aux_cfen and aux_anc* bit settings are ignored.

The *aux_bsize* value specifies, in number of bytes, the size of the buffer available for AUX data.

The *aux_pitch* bits specify the AUX line pitch, *i.e.*, the difference in the address from a pixel on a line to the same pixel on the next line, when pitch mode is enabled. Pitch mode is also defined for the ANC data capture, where each packet is treated as a new video line.

The *aux_osm* bit can be used to automatically limit capture by stopping after any wrap condition is reached. *End of AUX window* wrap condition also applies to ANC capture, even if AUX capture is disabled.

The data buffered in the local FIFOs is flushed when a wrap condition is reached or in pitch mode at the end of each video line or ANC packed. For the raw mode, the data is also flushed when disabling raw capture.

Address Generation

The AUX DMA base address register provides 28 bits to specify a destination address for storing the AUX data in the frame buffer. The address generation is similar to that of video capture, except for the missing double-buffer and field modes.

The relationship between the input data formats and the different video and auxiliary data capture scenarios is shown in Table 8. It is important to note, however, that the current VIP design does not support mixing of AUX and ANC data.

Table 8: Relationship Between Input Formats and Data Capture

Video Modes		Single stream (YUV)				Dual stream (Y and U/V)			
			Auxiliary Data				Auxiliary Data		
		Video Data	AUX	ANC	RAW	Video Data	AUX	ANC	RAW
D1	8-bit	X	X	X					
	10-bit	X	X	X					
VMI	8-bit	X	X			X	X		
RAW	8-bit				X				X
	10-bit				X				X
HD	8-bit					X	X		
	10-bit					X	X		

2.5.8 Interrupt Generation

The VIP contains a DVP-compliant interrupt generation mechanism. Interrupts can be generated for the following events:

- start of video
- end of video (written to memory)
- start of AUX in
- end of AUX out (written to memory)
- line threshold
- pipeline error (due to illegal scaling ratio e.g. >2x scaling or memory bus bandwidth error (fifo overflow))

In addition to these interrupts, the VIP module also provides *last-pixel-in* signals, for the VBI and video capture modes, to the GPIO block for timestamping.

3. Register Descriptions

3.1 Register Summary

The base address for VIP MMIO registers begins at absolute offset (with respect to MMIO_BASE) of 0x10 6000.

Table 9: VIP MMIO Register Summary

Offset	Name	Description
0x10 6000	VIP_MODE	VIP operation mode
0x10 6020	ANC_DID_EVEN	ANC DID for even field
0x10 6024	ANC_DID_ODD	ANC DID for odd field
0x10 6040	VIP_LINETHR	Video line count threshold
0x10 6100	VIN_FORMAT	Video input format and mode

Table 9: VIP MMIO Register Summary *…Continued*

Offset	Name	Description
0x10 6104	VIN_TESTPGEN	Video test pattern generator
0x10 6140	WIN_XYSTART	Video horizontal and vertical acquisition window start
0x10 6144	WIN_XYEND	Video horizontal and vertical acquisition window end
0x10 6160	PRE_DIT_CTRL	Pre-Dither control
0x10 6164	POST_DIT_CTRL	Post_Dither control
0x10 6180	AUX_XYSTART	Auxiliary horizontal and vertical acquisition window start
0x10 6184	AUX_XYEND	Auxiliary horizontal and vertical acquisition window stop
0x10 6200	HSP_ZOOM_0	Initial zoom for 1st pixel in line (unsigned)
0x10 6204	HSP_PHASE	Horizontal phase control
0x10 6208	HSP_DZOOM_0	Initial zoom delta for 1 pixel in line (signed)
0x10 620C	HSP_DDZOOM	Zoom delta change (signed)
0x10 6220	CSM_COEFF0	Color space matrix coefficients C_{00} - C_{02}
0x10 6224	CSM_COEFF1	Color space matrix coefficients C_{10} - C_{12}
0x10 6228	CSM_COEFF2	Color space matrix coefficients C_{20} - C_{22}
0x10 622C	CSM_OFFS1	Color space matrix offset coefficients D_0-D_2
0x10 6230	CSM_OFFS2	Color space matrix rounding coefficients E_0-E_2
0x10 6284	CSM_CKEY	Color key components
0x10 6300	PSU_FORMAT	Output format and mode
0x10 6304	PSU_WINDOW	Target window size
0x10 6340	PSU_BASE1	Target base address DMA #1
0x10 6344	PSU_PITCH1	Target line pitch component 1
0x10 6348	PSU_BASE2	Target base address DMA #2
0x10 634C	PSU_PITCH2	Target line pitch component 2 and 3
0x10 6350	PSU_BASE3	Target base address DMA #3
0x10 6354	PSU_BASE4	Target base address DMA #4
0x10 6358	PSU_BASE5	Target base address DMA #5
0x10 635C	PSU_BASE6	Target base address DMA #6
0x10 6380	AUX_FORMAT	auxiliary capture output format and mode
0x10 6390	AUX_BASE	Auxiliary capture base address
0x10 6394	AUX_PITCH	Auxiliary capture line pitch
0x10 6800—69FC	COEFF_TABLE	Coefficient table for horizontal filter (0-5)
0x10 6FE0	INT_STATUS	Interrupt status register
0x10 6FE4	INT_ENABLE	Interrupt enable register
0x10 6FE8	INT_CLEAR	Interrupt clear register
0x10 6FEC	INT_SET	Interrupt set register
0x10 6FFC	MODULE_ID	Module Identification and revision information

3.2 Register Table

Table 10: Video Input Processor (VIP) 1 Registers

Bit	Symbol	Access	Value	Description
Operating Mode Control Registers				
Offset 0x10 6000	***VIP Mode Control***			
31:30	VID_CFEN[1:0]	R/W	0	Video window capture field enable 00 = capture disabled 01 = capture odd only 10 = capture even only 11 = capture both
29	VID_OSM	R/W	0	Video capture one shot mode 0 = continuously capture fields selected by CFEN 1 = capture fields selected by CFEN only once
28	VID_FSEQ	R/W	0	video capture field sequence 0 = capture fields starting with any field 1 = capture fields starting with odd field setting has no effect unless VID_CFEN is set to capture both
27:26	AUX_CFEN[1:0]	R/W	0	Auxiliary window capture enable 00 = capture disabled 01 = capture odd only 10 = capture even only 11 = capture both
25	AUX_OSM	R/W	0	Auxiliary capture one shot mode 0 = when auxiliary wrap event is reached, buffer wraps around 1 = when auxiliary wrap event is reached, capturing stops
24	AUX_FSEQ	R/W	0	Auxiliary capture field sequence 0 = capture fields starting with any field 1 = capture fields starting with odd field setting has no effect unless AUX_CFEN is set to capture both
23:22	AUX_ANC[1:0]	R/W	0	ANC data capture enable 00 = no ANC data captured 01 = odd ANC field blocks. (masked DATA_ID_0 bit matched) 10 = even ANC field blocks. (masked DATA_ID_1 bit matched) 11 = odd/even ANC field blocks. (masked DATA_ID_* bit matched)
21	AUX_RAW	R/W	0	Auxiliary raw capture enable 0 = raw capture disabled 1 = raw capture enabled, all samples will be captured when enabled, AUX_ANC and AUX_CFEN settings are ignored
20:18	reserved			
17	RST_ON_ERR	R/W	0	Reset on error Writing a one into this bit will automatically reset the block in case of a pipeline error (e.g. illegal scaling ratio / FIFO overflow)
16	SOFT_RESET	W	0	Soft reset Writing a one into this bit will reset the block
15	reserved			

Table 10: Video Input Processor (VIP) 1 Registers *…Continued*

Bit	Symbol	Access	Value	Description
14	IFF_CLAMP	R/W	0	Clamp mode for IFF (affects U/V only) 0: clamp to 0-255 1: clamp to 16 - 240 (CCIR range)
13:12	IFF_MODE	R/W	0	Interpolation mode 00: bypass 01: reserved 10: co-sited 11: interspersed
11	reserved			
10	DFF_CLAMP	R/W	0	Clamp mode for DFF (affects U/V only) 0: clamp to 0-255 1: clamp to 16 - 240 (CCIR range)
9: 8	DFF_MODE	R/W	0	Decimation mode 00: bypass 01: co-sited (sub sample) 10: co-sited (low pass) 11: interspersed
7:4	reserved			
3	HSP_CLAMP	R/W	0	Clamp mode for HSP 0: clamp to 0-255 1: clamp to CCIR range defined by bit 2
2	HSP_RGB	R/W	0	Color space mode, defines CCIR clamping range for HSP 0: processing in YUV color space 1: processing in RGB color space
1:0	HSP_MODE	R/W	0	Horizontal processing mode 00: bypass mode 01: color space matrix mode 10: normal polyphase mode 11: transposed polyphase mode
ANC Identifier Codes (DID)				
Offset 0x10 6020	***ANC Identifier Codes - Odd Field***			
31:16	reserved			
15:8	ID_MASK_0[7:0]	R/W	0xFC	Mask for enabling bit checking on ANC identifier code For each ID_MASK_0[i] bit, 1: enable DATA_ID_0[i] bit checking 0: disable DATA_ID_0[i] bit checking
7:0	DATA_ID_0[7:0]	R/W	0x44	ANC identifier code
Offset 0x10 6024	***ANC Identifier Codes - Even Field***			
31:16	reserved			
15:8	ID_MASK_1[7:0]	R/W	0xFC	Mask for enabling bit checking on ANC identifier code For each ID_MASK_1[i] bit, 1: enable DATA_ID_1[i] bit checking 0: disable DATA_ID_1[i] bit checking
7:0	DATA_ID_1[7:0]	R/W	0x54	ANC identifier code

Table 10: Video Input Processor (VIP) 1 Registers *…Continued*

Bit	Symbol	Access	Value	Description
Video Informations Registers				
Offset 0x10 6040	***VIP Line Threshold***			
31:11	reserved			
10:0	LCTHR[10:0]	R/W	0	Video line count threshold Line threshold status bit is set if video line count (SVLC) reaches this value Note: It is possible to have multiple interrupts per field at different line counts, by re-programming the threshold value in this register from the ISR.
Input Format Control Registers				
Offset 0x10 6100	***Video Input Format***			
31:30	VSRA[1:0]	R/W	0	Video stream realignment 00 = normal 01 = ignore 1st sample after HREF 1x = reserved
29:26	reserved		-	
25	SYNCHD	R/W	0	HD sync select 0 = embedded sync 1 = explicit sync
24	DUAL_STREAM	R/W	0	Dual video data stream enable 0 = single video data stream mode 1 = dual video data stream mode
23:21	reserved		-	
20	NHDAUX	R/W	0	header detect during AUX window 0 = D1 header detection enabled inside AUX window 1 = D1 header detection disabled inside AUX window
19	NPAR	R/W	0	Parity check disable 0 = parity check enabled for D1 header detection 1 = parity check disabled for D1 header detection
18:16	reserved		-	
15:14	VSEL[1:0]	R/W	0	Video source select 00 = reserved 01 = video port, encoded sync (D1-Mode) 10 = video port, external sync (VMI-Mode) 11 = reserved
13	TWOS	R/W	0	UV data type 0 = offset binary 1 = two's complement
12	TPG	R/W	0	Test pattern generator 0 = video stream selected by VSEL 1 = internal test pattern generator
11:10	reserved		-	
10	FREF	R/W	0	Field toggle reference mode 0 = normal, use VREF 1 = toggling Field bit is used as vertical reference

Table 10: Video Input Processor (VIP) 1 Registers *…Continued*

Bit	Symbol	Access	Value	Description
9	FTGL	R/W	0	Field toggle mode 0 = normal 1 = free toggle (sequence starts with FID = 0)
8:4	reserved		-	
3	SF	R/W	0	Swap field interpretation 0: odd (first) field = 0, even (second) field = 1 1: odd (first) field = 1, even (second) field = 0
2	FZERO	R/W	0	Force FID value to zero 0 = field identifier derived from input stream 1 = force field identifier value to 0
1	REVS	R/W	0	Vertical sync reference edge 0 = falling edge / start of active video 1 = rising edge / end of active video
0	REHS	R/W	0	Horizontal sync reference edge 0 = falling edge / SAV 1 = rising edge / EAV
Offset 0x10 6104	***Video Test Pattern Generator Control***			
31	PAL	R/W	0	Field generation mode 0 = NTSC timing 1 = PAL timing
30	reserved		0	
29	VSEL	R/W	0	Vertical timing signal select (will be removed) 0 = generate VREF 1 = generate VS
28	HSEL	R/W	0	Horizontal timing signal select (will be removed) 0 = generate HREF 1 = generate HS
27	SWAP	R/W	0	Alternative test pattern 0 = normal test pattern 1 = test pattern with diagonal patterns, etc.
26	MOVE	R/W	0	Scrolling enable for alternative test pattern 0 = no scrolling 1 = scrolling enabled
25:0	reserved		0	
Video Acquisition Window Control Registers				
Offset 0x10 6140	***Video Acquisition Window Start***			
31:27	reserved		-	
26:16	VID_XWS[10:0]	R/W	0	Horizontal video window start The pixel co-sited with the reference edge REHS is numbered 0.
15:11	reserved		-	
10:0	VID_YWS[10:0]	R/W	0	Vertical video window start The first line indicated by the reference edge REVS is numbered 0.
Offset 0x10 6144	***Video Acquisition Window End***			
31:27	reserved		-	

Table 10: Video Input Processor (VIP) 1 Registers *…Continued*

Bit	Symbol	Access	Value	Description
26:16	VID_XWE[10:0]	R/W	0	Horizontal video window end pixels from XWS up to and including XWE are processed
15:11	reserved		-	
10:0	VID_YWE[10:0]	R/W	0	Vertical video window end lines from YWS up to and including YWE are processed
Offset 0x10 6160	***Pre-Dither Control***			
Offset 0x10 6164	***Post-Dither Control***			
31	DITHER_ENBLE	R/W	0	Dither Control Enable / disable the dither unit. DITHER_ENABLE = 0 : disable; DITHER_ENABLE = 1 : enable.
30	DITHER_Y	R/W	-	Dither Y Components Enable / disable dithering of Y pixel-components. DITHER_Y = 0 : disable (round and clip); DITHER_Y = 1 : enable (dither).
29	DITHER_UV	R/W	-	Dither U and V Components Enable / disable dithering of U and V pixel-components. DITHER_UV = 0 : disable (round and clip); DITHER_UV = 1 : enable (dither).
28:27	MODE[1:0]	R/W	-	Mode of Operation Select input and output sizes and the computations carried out: MODE = 0 : 10->9; MODE = 1 : 10->8; MODE = 2 : 9->8; MODE = 3: Reserved.
26:4	reserved	R	-	
3	FRAME_ALT	R/W	-	Frame Alternate Enable/disable frame alternation while dithering. FRAME_ALT = 0 : disable; FRAME_ALT = 1 : enable.
2	FIELD_ALT	R/W	-	Field Alternate Enable/disable field alternation while dithering. FIELD_ALT = 0 : disable; FIELD_ALT = 1 : enable.
1	LINE_ALT	R/W	-	Line Alternate Enable/disable line alternation while dithering. LINE_ALT = 0 : disable; LINE_ALT = 1 : enable.
0	DOUBLE_PIXEL_ALT	R/W	-	Single or Double Pixel Alternate Select single or double pixel alternation while dithering. DOUBLE_PIXEL_ALT = 0 : single pixel alternation; DOUBLE_PIXEL_ALT = 1 : double pixel alternation.
VBI Acquisition Window Control Registers				
Offset 0x10 6180	***Auxiliary Acquisition Window Start***			
31:27	reserved		-	
26:16	AUX_XWS[10:0]	R/W	0	Horizontal auxiliary window start The pixel cosited with the reference edge REHS is numbered 0.

Table 10: Video Input Processor (VIP) 1 Registers *…Continued*

Bit	Symbol	Access	Value	Description
15:11	reserved		-	
10:0	AUX_YWS[10:0]	R/W	0	Vertical auxiliary window start The line cosited with the reference edge REVS is numbered 0.
Offset 0x10 6184	***Auxiliary Acquisition Window End***			
31:27	reserved		-	
26:16	AUX_XWE[10:0]	R/W	0	Horizontal auxiliary window end pixels from XWS up to and including XWE are processed
15:11	reserved		-	
10:0	AUX_YWE[10:0]	R/W	0	Vertical auxiliary window end lines from YWS up to and including YWE are processed
Horizontal Video Processing Control Registers				
Offset 0x10 6200	***Initial Zoom***			
31:29	HSP_PHASE_MODE[2:0]	R/W	0	Phase mode 0: 64 phases 1: 32 phases 2: 16 phases 3: 8 phases 4: 4 phases 5: 2 phases 6: fixed phase 7: linear phase interpolation (only valid for4 component mode)
28:27	reserved		-	
26	HSP_FIR_COMP[1:0]	R/W		Horizontal filter components 0: three components, 6 tap FIR each 1: four components, 3 tap FIR each (4th component unused) In color space matrix mode this value has to remain zero
25:20	reserved		-	
19: 0	HSP_ZOOM_0[19:0]	R/W	0	Initial zoom for 1st pixel in line (unsigned, LSB = 2^{-16}) 2 0000 (hex): downscale 50% 1 0000 (hex): no scaling = 2^{0} 0 8000 (hex): zoom 2 x (transposed: downscale 50%)
Offset 0x10 6204	***Phase Control***			
31	reserved		-	
30:28	HSP_QSHIFT[2:0]	R/W	0	Quantization shift control used to change quantization before being multiplied with HSP_MULTIPLY. 100 (bin): divide by 16 101 (bin): divide by 8 110 (bin): divide by 4 111 (bin): divide by 2 000 (bin): multiply by 1 001 (bin): multiply by 2 010 (bin): multiply by 4 011 (bin): multiply by 8 Warning: A value range overflow caused by an improper quantization shift can not be compensated later by multiplying with a HSP_MULTIPLY value below 0.5!

Table 10: Video Input Processor (VIP) 1 Registers *…Continued*

Bit	Symbol	Access	Value	Description
27:26	reserved		-	
25	HSP_QSIGN	R/W	0	Quantization sign bit
24:16	HSP_QMULTIPLY[8:0]	R/W	0	Quantization multiply control used to compensate for different weight sum in transposed polyphase or color space matrix mode, remaining bits are fraction (largest number is 511/512) Value range: $0 \le m < 1.0$. Instead of using values in the range of $m < 0.5$ the quantization shift HSP_QSHIFT should be modified to gain more precision in the truncated result.
15:13	reserved		-	
12: 0	HSP_OFFSET_0	R/W	0	Initial start offset for DTO
Offset 0x10 6208	***Initial Zoom delta***			
31:26	reserved		-	
25: 0	HSP_DZOOM_0[25:0]	R/W	0	Initial zoom delta for 1 pixel in line (signed, LSB = 2^{-27}) used for non constant scaling ratios
Offset 0x10 620C	***Zoom delta change***			
31:29	reserved		-	
28: 0	HSP_DDZOOM[28:0]	R/W	0	Zoom delta change (signed, LSB = 2^{-40}) used for non constant scaling ratios
Color Space Matrix Registers				
Offset 0x10 6220	***Color space matrix coefficients C_{00} - C_{02}***			
31:30	Unused		-	
29:20	CSM_C02[9:0]	R/W	0	Coefficient C02, two's complement
19:10	CSM_C01[9:0]	R/W	0	Coefficient C01, two's complement
9:0	CSM_C00[9:0]	R/W	0	Coefficient C00, two's complement
Offset 0x10 6224	***Color space matrix coefficients C_{10} - C_{12}***			
31:30	Unused		-	
29:20	CSM_C12[9:0]	R/W	0	Coefficient C12, two's complement
19:10	CSM_C11[9:0]	R/W	0	Coefficient C11, two's complement
9:0	CSM_C10[9:0]	R/W	0	Coefficient C10, two's complement
Offset 0x10 6228	***Color space matrix coefficients C_{20} - C_{22}***			
31:30	Unused		-	
29:20	CSM_C22[9:0]	R/W	0	Coefficient C22, two's complement
19:10	CSM_C21[9:0]	R/W	0	Coefficient C21, two's complement
9:0	CSM_C20[9:0]	R/W	0	Coefficient C20, two's complement
Offset 0x10 622C	***Color space matrix offset coefficients D_0 - D_2***			
31:29	Unused		-	
28	CSM_D2_TWOS	R/W	0	Offset coefficient D_2 type 0 = unsigned 1 = signed
27:20	CSM_D2[7:0]	R/W	0	Offset coefficient D_2

Table 10: Video Input Processor (VIP) 1 Registers *…Continued*

Bit	Symbol	Access	Value	Description
19	Unused		-	
18	CSM_D1_TWOS	R/W	0	Offset coefficient D_1 type 0 = unsigned 1 = signed
17:10	CSM_D1[7:0]	R/W	0	Offset coefficient D_1
9	Unused		-	
8	CSM_D0_TWOS	R/W	0	Offset coefficient D_0 type 0 = unsigned 1 = signed
7:0	CSM_D0[7:0]	R/W	0	Offset coefficient D_0
Offset 0x10 6230	***Color space matrix offset coefficients E_0 - E_2***			
31:30	Unused		-	
29:20	CSM_E2[9:0]	R/W	0	Offset coefficient E2, two's complement
19:10	CSM_E1[9:0]	R/W	0	Offset coefficient E1, two's complement
9:0	CSM_E0[9:0]	R/W	0	Offset coefficient E0, two's complement
Color Keying Control Registers				
Offset 0x10 6284	***Color Key Components***			
31: 24	CKEY_ALPHA	R/W	0	Alpha value Defines the alpha value to be used for keyed samples.
23: 0	reserved			
Video Output Format Control Registers				
Offset 0x10 6300	***Video Output Format***			
31:30	PSU_BAMODE	R/W	0	Base address mode 00 = single set (e.g. progressive video source) base 1-3 according to number of planes (plane 1-3) 01 = reserved 10 = alternate sets each field (e.g. interlaced video source) base 1-3, odd field (plane 1-3) base 4-6, even field (plane 1-3) 11 = alternate sets each field and frame (e.g. double buffer mode) packed modes only, frame index is set to 1 if cfen=0, frame index is incremented after capturing even field before capturing odd, base address byte offset is defined in PSU_OFFSET1 base 1, odd field 1st frame (plane 1 only) base 2, even field 1st frame (plane 1 only) base 3, odd field 2nd frame (plane 1 only) base 4, even field 2nd frame (plane 1 only)
29:14	reserved		-	
13	PSU_ENDIAN	R/W	0	Output format endianess 0: same as system endianess 1: opposite of system endianess
12	reserved		-	

Table 10: Video Input Processor (VIP) 1 Registers *…Continued*

Bit	Symbol	Access	Value	Description
11:10	PSU_DITHER	R/W	0	Output format dither mode 00: no dithering 01: error dispersion (never reset pattern) 10: error dispersion (reset pattern at first capture enable) 11: error dispersion (reset pattern every field)
9:8	PSU_ALPHA	R/W	0	Output format alpha mode 00 = no alpha (alpha byte not written) 01 = alpha byte written, value from CKEY_ALPHA (offset 284) 10 = reserved 11 = reserved setting 00 is ignored if size of alpha component is less than 8 bits
7:0	PSU_OPFMT	R/W	0	Output formats 08 (hex) = YUV 4:2:2, semi-planar 0B (hex) = YUV 4:2:2, planar 0F (hex) = RGB or YUV 4:4:4, planar A9 (hex) = compressed 4/4/4 + (4 bit alpha) AA (hex) = compressed 4/5/3 + (4 bit alpha) AD (hex) = compressed 5/6/5 A0 (hex) = packed YUY2 4:2:2 A1 (hex) = packed UYVY 4:2:2 E2 (hex) = YUV or RGB 4:4:4 + (8 bit alpha) E3 (hex) = VYU 4:4:4 + (8 bit alpha)
Offset 0x10 6304	***Target Window Size***			
31:27	reserved		-	
26:16	PSU_LSIZE	R/W	0	Line size Used for horizontal cropping after scaling 0 = cropping disabled 1 = one pixel
15:11	reserved		-	
10:0	PSU_LCOUNT	R/W	0	Line count Used for vertical cropping after scaling 0 = cropping disabled 1 = one line
Video Output Address Generation Control Registers				
Offset 0x10 6340	***Target Base Address #1***			
31:28	reserved		-	
27: 3	PSU_BASE1	R/W		Base address DMA #1 used depending on PSU_BAMODE setting
2:0	PSU_OFFSET1	R/W		Base address byte offset plane 1 bits define pixel offset within multi pixel 64 bit words (e.g. a 16bit pixel can be placed on any 16 bit boundary)
Offset 0x10 6344	***Target Line Pitch #1***			
31:15	Unused		-	
14: 3	PSU_PITCH1	R/W		Line pitch DMA #1, signed value (two's complement) used for all packed formats and for plane 1
2:0	Unused		-	
Offset 0x10 6348	***Target Base Address #2***			

Table 10: Video Input Processor (VIP) 1 Registers *…Continued*

Bit	Symbol	Access	Value	Description
31:28	Unused		-	
27: 3	PSU_BASE2	R/W		Base address DMA #2 used depending on PSU_BAMODE setting
2:0	PSU_OFFSET2	R/W		Base address byte offset plane 2 bits define pixel offset within multi pixel 64 bit words (e.g. a 16bit pixel can be placed on any 16 bit boundary)
Offset 0x10 634C	***Target Line Pitch #2***			
31:15	Unused		-	
14: 3	PSU_PITCH2	R/W		Line pitch DMA #2, signed value (two's complement) used for planes 2 and 3
2:0	Unused		-	
Offset 0x10 6350	***Target Base Address #3***			
31:28	Unused		-	
27: 3	PSU_BASE3	R/W		Base address DMA #3 used depending on PSU_BAMODE setting
2:0	PSU_OFFSET3	R/W		Base address byte offset plane 3 bits define pixel offset within multi pixel 64 bit words (e.g. a 16bit pixel can be placed on any 16 bit boundary)
Offset 0x10 6354	***Target Base Address #4***			
31:28	Unused		-	
27: 3	PSU_BASE4	R/W		Base address DMA #4 used depending on PSU_BAMODE setting
2: 0	Unused		-	
Offset 0x10 6358	***Target Base Address #5***			
31:28	Unused		-	
27: 3	PSU_BASE5	R/W		Base address DMA #5 used depending on PSU_BAMODE setting
2: 0	Unused		-	
Offset 0x10 635C	***Target Base Address #6***			
31:28	Unused		-	
27: 3	PSU_BASE6	R/W		Base address DMA #6 used depending on PSU_BAMODE setting
2: 0	Unused		-	
Auxiliary Data Output Format Control Registers				
Offset 0x10 6380	***Auxiliary Capture Output Format***			
31:30	AUX_BAMODE		0	Base address mode 00 = pitch mode, wrap at end of buffer or window 01 = pitch mode, wrap at end of buffer 10 = append mode, wrap at end of buffer or window 11 = append mode, wrap at end of buffer
29:27	reserved		-	

Table 10: Video Input Processor (VIP) 1 Registers *…Continued*

Bit	Symbol	Access	Value	Description
26	AUX_SGNEX	R/W	0	Auxiliary capture sign extension 0 = no sign extension 1 = sign extension enabled for 10 bit samples
25	AUX_BPS	R/W	0	Auxiliary capture bits per sample 0 = 8 bit samples 1 = 10 bit samples
24	AUX_SUBSAMPLE	R/W	0	Auxiliary capture sub-sample 0 = all samples 1 = luma (even) samples only Not available for ANC data capture
23:22	reserved		-	
21:0	AUX_BZSIZE[21:0]	R/W	0	Auxiliary capture ringbuffer size Size of ringbuffer in bytes, 0 = unlimited buffer size
Auxiliary Data Output Address Generation Control Registers				
Offset 0x10 6390	***Auxiliary Capture Base Address***			
31:28	Unused		-	
27: 0	AUX_BASE	R/W	0	Auxiliary capture base address Lower 3 bits define byte offset within 64 bit words, offset has to be a multiple of the byte per unit size (e.g. a 16bit unit can be placed on any 16 bit boundary)
Offset 0x10 6394	***Auxiliary Capture Line Pitch***			
31:15	Unused		-	
14: 3	AUX_PITCH	R/W	0	Auxiliary capture line pitch Signed value
2:0	Unused		-	
Miscellaneous Registers				
Offset 0x10 6800 - 69FC	***Coefficient Table #1 Taps 0-5 (Horizontal)***			
63:62	Unused		-	
61:52	TAP_5[X][9:0]	W	-	Inverted coefficient, tap #5, two's complement
51:42	TAP_4[X][9:0]	W	-	Inverted coefficient, tap #4, two's complement
41:32	TAP_3[X][9:0]	W	-	Inverted coefficient, tap #3, two's complement
31:30	Unused		-	
29:20	TAP_2[X][9:0]	W	-	Inverted coefficient, tap #2, two's complement
19:10	TAP_1[X][9:0]	W	-	Inverted coefficient, tap #1, two's complement
9:0	TAP_0[X][9:0]	W	-	Inverted coefficient, tap #0, two's complement
Interrupt and Status Control Registers				
Offset 0x10 6FE0	***Interrupt Status***			
31	STAT_FID_AUX	R	1	Field identifier at start of auxiliary window
30	STAT_FID_VID	R	0	Field identifier at start of video window
29	STAT_FID_VPI	R	0	Field identifier at video input port
28	Unused		-	

Table 10: Video Input Processor (VIP) 1 Registers *…Continued*

Bit	Symbol	Access	Value	Description
27:16	STAT_LINE_COUNT [11:0]	R	0	Source video line count
15:10	Unused		-	
9	STAT_AUX_OVRFLW	R	0	Auxiliary buffer overflow event
8	STAT_VID_OVRFLW	R	0	Video buffer overflow event
7	STAT_WIN_SEQBRK	R	0	Windower sequence break event
6	STAT_FID_SEQBRK	R	0	Field identifier sequence break event
5	STAT_LINE_THRESH	R	0	Line counter threshold reached event
4	STAT_AUX_WRAP	R	0	Auxiliary capture write pointer wrap around event
3	STAT_AUX_START_IN	R	0	Start of auxiliary data acquisition event
2	STAT_AUX_END_OUT	R	0	End of auxiliary data write to memory event
1	STAT_VID_START_IN	R	0	Start of video data acquisition event
0	STAT_VID_END_OUT	R	0	End of video data write to memory event
Offset 0x10 6FE4	***Interrupt Enable***			
31:10	Unused		-	
9	IEN_AUX_OVRFLW	R/W	0	Auxiliary buffer overflow event
8	IEN_VID_OVRFLW	R/W	0	Video buffer overflow event
7	IEN_WIN_SEQBRK	R/W	0	Windower sequence break event
6	IEN_FID_SEQBRK	R/W	0	Field identifier sequence break event
5	IEN_LINE_THRESH	R/W	0	Line counter threshold reached event
4	IEN_AUX_WRAP	R/W	0	Auxiliary capture write pointer wrap around event
3	IEN_AUX_START_IN	R/W	0	Start of auxiliary data acquisition event
2	IEN_AUX_END_OUT	R/W	0	End of auxiliary data write to memory event
1	IEN_VID_START_IN	R/W	0	Start of video data acquisition event
0	IEN_VID_END_OUT	R/W	0	End of video data write to memory event
Offset 0x10 6FE8	***Interrupt Clear***			
31:10	Unused		-	
9	CLR_AUX_OVRFLW	W	0	Auxiliary buffer overflow event
8	CLR_VID_OVRFLW	W	0	Video buffer overflow event
7	CLR_WIN_SEQBRK	W	0	Windower sequence break event
6	CLR_FID_SEQBRK	W	0	Field identifier sequence break event
5	CLR_LINE_THRESH	W	0	Line counter threshold reached event
4	CLR_AUX_WRAP	W	0	Auxiliary capture write pointer wrap around event
3	CLR_AUX_START_IN	W	0	Start of auxiliary data acquisition event
2	CLR_AUX_END_OUT	W	0	End of auxiliary data write to memory event
1	CLR_VID_START_IN	W	0	Start of video data acquisition event
0	CLR_VID_END_OUT	W	0	End of video data write to memory event
Offset 0x10 6FEC	***Interrupt Set***			

Table 10: Video Input Processor (VIP) 1 Registers *…Continued*

Bit	Symbol	Access	Value	Description
31:10	Unused		-	
9	SET_AUX_OVRFLW	W	0	Auxiliary buffer overflow event
8	SET_VID_OVRFLW	W	0	Video buffer overflow event
7	SET_WIN_SEQBRK	W	0	Windower sequence break event
6	SET_FID_SEQBRK	W	0	Field Identifier sequence break event
5	SET_LINE_THRESH	W	0	Line counter threshold reached event
4	SET_AUX_WRAP	W	0	Auxiliary capture write pointer wrap around event
3	SET_AUX_START_IN	W	0	Start of auxiliary data acquisition event
2	SET_AUX_END_OUT	W	0	End of auxiliary data write to memory event
1	SET_VID_START_IN	W	0	Start of video data acquisition event
0	SET_VID_END_OUT	W	0	End of video data write to memory event
Offset 0x10 6FF4	***Powerdown***			
31	Power_Down	RW	0	0 = normal operation 1 = Powerdown mode
30:0	Reserved			
Offset 0x10 6FFC	***Module ID***			
31: 16	MOD_ID	R	011A	Module ID Unique 16-bit code
15: 12	REV_MAJOR	R	3	Major revision counter
11: 8	REV_MINOR	R	0	Minor revision counter
7: 0	APP_SIZE	R	00	Aperture Size 0 = 4 kB

Chapter 13: FGPO: Fast General Purpose Output

PNX15xx Series Data Book – Volume 1 of 1

Rev. 2 — 1 December 2004 Product data sheet

1. Introduction

The Fast General Purpose Output (FGPO) module is a high-bandwidth (up to 400 MBytes/sec) output data channel. The FGPO outputs data in 8, 16, and 32-bit widths.

The FGPO operates in two main modes: record output or message passing

- May be used as a versatile interface with streaming data receivers at rates from DC to 100 MHz
- May be used as a transmitter port for inter-TriMedia unidirectional message passing
- Allows optional synchronization with external control signals
- Allows optional generation of external control signals
- Allows optional output at selected timestamp times
- Allows optional output of variable message/record lengths
- Allows continuous data output transfers using DMA transfers from two main memory buffers

1.1 FGPO Overview

Figure 1 shows the top level connection of the FGPO module to the MMIO and MTL Busses within the PNX15xx Series. All external FGPO signals are registered and routed through the Output Router module before leaving the PNX15xx Series. Latency buffering of data and endian conversion is done in the MTL DTL Adapter. All FGPO register access is through the MMIO DTL adapter.

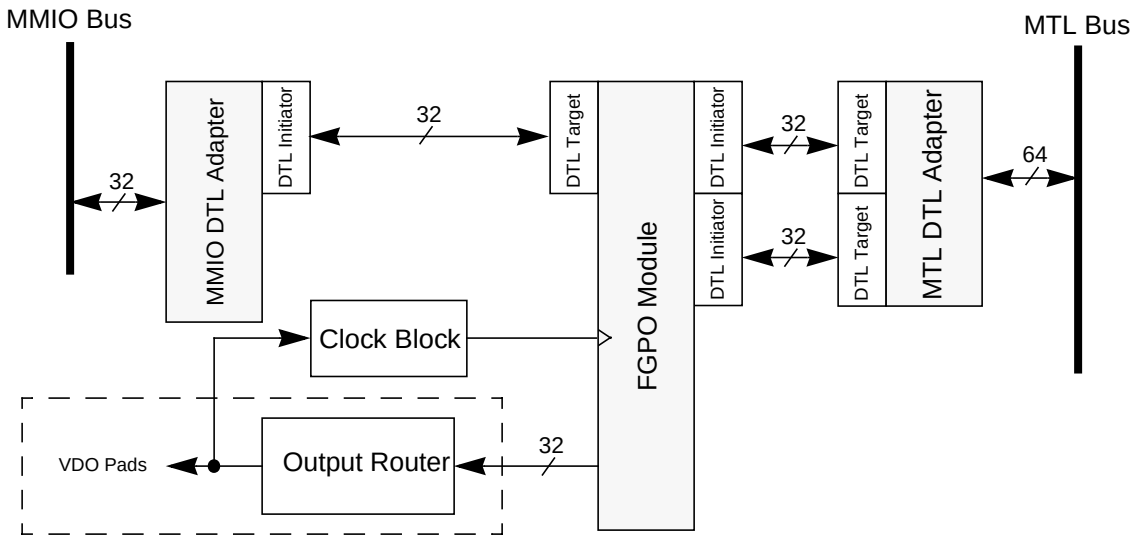


Figure 1: Top Level Block Diagram

Figure 2 shows the basic sections of the FGPO module.

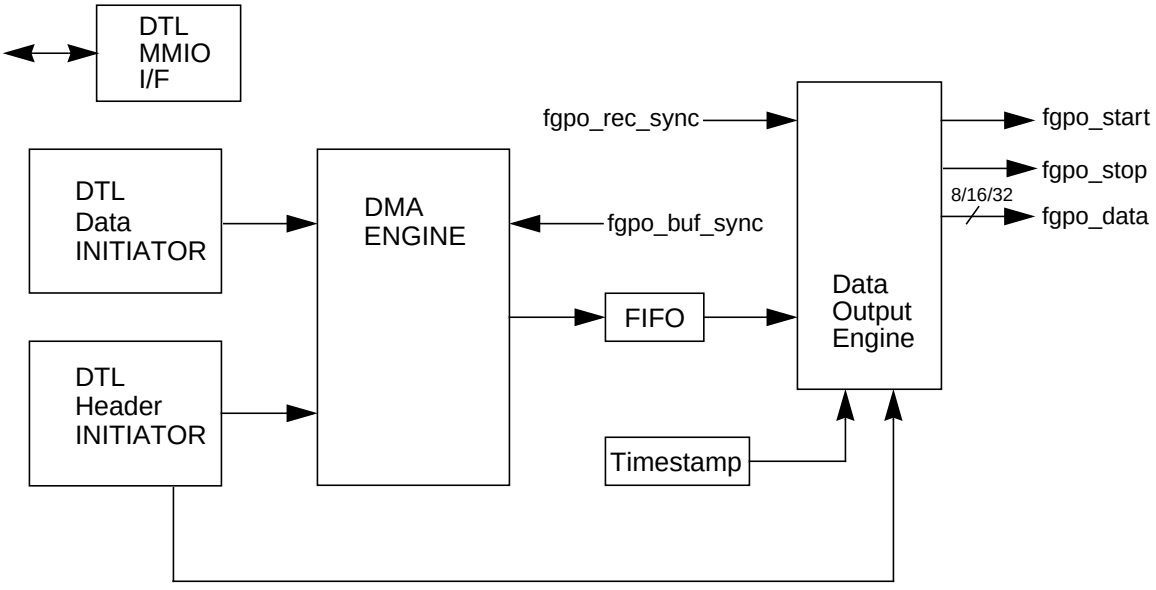


Figure 2: FGPO Module Block Diagram

1.2 FGPO to VDO pin mapping

fgpo_start (fgpo_rec_start) maps to VDO_D[32]

fgpo_stop (fgpo_buf_start) maps to VDO_D[32]

fgpo_clk from clock module maps to VDO_C2

VDO_D[31:0] mapping depends on the VDO_MODE (Output Router) register settings, see Chapter 3 System On Chip Resources.

1.3 DTL MMIO Interface

This block contains all of the programmable registers used by the FGPO module accessed through the MMIO bus. Refer to Section 4. for registers description. This block also handles clock domain crossing between the MMIO bus clock and the FGPO module clock.

1.4 Header Initiator

If either FGPO_CTL.TSTAMP_SELECT or FGPO_CTL.VAR_LENGTH bits are set this DTL Initiator will read the record/message Timestamp and Variable Length fields. The Variable Length information is passed on to the DMA Engine to issue a read request from memory. The Timestamp information is passed to the Data Output Engine for a timestamp trigger point. The MTL DTL Adapter for this DTL port contains a 2x8 (16 byte) FIFO.

1.5 Data Initiator

Issues main memory read requests for all data samples. The MTL DLT Adapter for this DTL port contains a 128x8 (1024 byte) FIFO.

1.6 Record Output Mode

This mode allows the FGPO to read and transmit structured record data from main memory to the outside world. The start of a record may be triggered by reaching an absolute time (Timestamp), by expiration of a counted gap between records, or by a synchronized external transition on the fgpo_rec_sync pin.

The switching of buffers may also be triggered by a synchronized external transition on the fgpo_buf_sync pin.

A record start control signal is generated at the start of each record on the fgpo_start (fgpo_rec_start) pin.

Output starts from a new location in the buffer for each record. Successive records are output until the programmed number of records in a buffer is exhausted, then the alternate buffer is used.

A buffer start control signal is generated at the start of each new buffer on the fgpo_stop (fgpo_buf_start) pin.

This allows the output of video frames consisting of multiple line records, synchronized by a frame or field synchronization signal.

1.7 Message Passing Mode

This mode allows the FGPO to read and transmit messages from main memory to either an FGPI unit on another PNX15xx Series or a PNX1300 Series VI in message passing mode.

One FGPO can broadcast to multiple receiving FGPI's. No data interpretation is done. Each message from the sender is read from a separate message location in the memory buffer.

Message start and stop is signaled by the sender by separate fgpo_start and fgpo_stop control signals.

2. Functional Description

Table 1: Module signal pins

Signal	Type	Description
clk_fgpo	input	From Clock Module. External FGPO clock on VDO_C2 pin is connected to the Clock Module. FGPO data and control signals are output at each rising edge on clk_fgpo. Use the PNX15xx Series Clock Module to change clk_fgpo characteristics.
fgpo_rec_sync	input	From External PAD. In external record/message sync mode a programmable transition on this pin will trigger the output of a record or message after a synchronization delay of 4 FGPO clock cycles. If the transition occurs before the FGPO has finished the output of a previous record or message, the transition will be ignored.
fgpo_buf_sync	input	From External PAD. In external buffer sync mode a programmable transition on this pin will start a new buffer after a synchronization delay of 4 FGPO clock cycles. If the transition occurs before the FGPO has finished the current buffer, the transition will be ignored.
fgpo_start or fgpo_rec_start	output	To External PAD VDO_D[32] via Output Router. Message Passing Mode: A positive pulse output on this pin indicates the start of a message. The pulse may be programmed to occur one clock before or at the same clock with the first valid data sample. Record Mode: A positive pulse output on this pin indicates the start of a record. The pulse may be programmed to occur one clock before or at the same clock with the first valid data sample or A positive pulse lasting as long as valid data samples are output.

Table 1: Module signal pins *...Continued*

Signal	Type	Description
fgpo_stop or fgpo_buf_start	output	To External PAD VDO_D[33] via Output Router. Message Passing Mode: A programmable pulse on fgpo_stop indicates the end of a message. This pulse may be programmed to be a one clock pulse concurrent with the last data sample, or a pulse lasting as long as valid data samples are output. Record Capture Mode: A programmable pulse on fgpo_buf_start indicates the start of a new buffer. The pulse may be programmed to occur one clock before or at the same clock with the first valid data sample for the buffer or A positive pulse lasting as long as each buffer is active. or A positive pulse lasting as long as buffer 2 is active.
fgpo_data	output	To External PAD VDO_D[31:0] via Output Router. General Purpose high speed sample data output changing on each active edge of clk_fgpo. In 8-bit mode data is placed on fgpo_data7:0]. In 16-bit mode data is placed on fgpo_data[15:0].
fgpo_interrupt	output	Interrupt status connects to the TriMedia Processor in the PNX15xx Series.

2.1 Stopping clk_fgpo for output flow control

Some users may wish to supply the FGPO clock externally. The Clock Module can be set up to receive the clock on VDO_C2 and drive clk_fgpo with that signal. This would allow the external world to stop the FGPO clock (when low) for flow control. All FGPO registers can be accessed (read/write) while clk_fgpo is inactive without any bus time-outs.

2.2 Reset

FGPO is reset by any PNX15xx Series system reset or by setting the SOFTWARE_RESET bit FGPO_SOFT_RST register.

Remark: SOFTWARE_RESET does not reset MMIO bus interface registers. Any DMA transfers will be aborted during a SOFTWARE_RESET. All registers reset to the Reset Value shown in the Register Description section.

2.3 Base Addresses

Two base address registers are used to point to main memory buffers in a double buffering scheme. Addresses are forced into 32-bit address alignment.

2.4 Sample (data) Size

Data size (width) per sample is set to either 8, 16, or 32-bit using FGPO_CTL.DATA_SIZE bit field. For 8-bit samples, four samples are packed into one 32-bit word. For 16-bit samples, two samples are packed 2 into one 32-bit word. Packed data is read from memory in full 32-bit words.

Byte order, with which the data is read from memory, is controlled by the global PNX15xx Series endian mode. The endian state only affects 16 and 32-bit sample sizes.

2.5 Record or Message Size

The number of samples per record is set by FGPO_REC_SIZE field. This is amount of data that will be output after each record or message start event unless the FGPO_CTL.VAR_LENGTH bit is set. If the FGPO_CTL.VAR_LENGTH bit is set the length of a record or message is set by the value of the second 32-bit word read from the header attached to the record or message.

Valid values are in the range of 2 to 2^{24} - 1.

Remark: The FGPI has a minimum message size of 2 or 3. See FGPI Module specification for more information.

2.6 Records or Messages Per Buffer

The number of records or messages per buffer is set by FGPO_SIZE register.

Valid values are in the range of 1 to 2^{24} - 1.

2.7 Stride

If the number of records or messages per buffer is greater than one, the address stride has to be programmed into the FGPO_STRIDE register.

Output starts at a new location in the current buffer on each record or message start event. After output starts a new address is generated by adding the contents of the FGPO_STRIDE register to the previous starting address.

Care must be taken that FGPO_STRIDE is greater than or equal to FGPO_REC_SIZE. Add 8 if either TSTAMP_SELECT or VAR_LENGTH bits are set.

2.8 Interrupt Events

The FGPO_IR_STATUS register contains status and interrupt event status. To generate an interrupt to the TriMedia processor the corresponding FGPO_IR_ENA bit must be set. To clear an interrupt event (acknowledge the interrupt) a ‘1’ must be written to the corresponding FGPO_IR_CLR bit. The FGPO_IR_SET register can be used to generate software interrupts.

12NC 9397 750 14321

2.8.1 BUF1DONE and BUF2DONE Interrupts

When the number of records or messages output from a main memory buffer equals the value in the FGPO_SIZE register an associated Buffer Done interrupt will be generated.

Remark: This interrupt is generated when the FGPO Engine finishes sending the last sample from the last record/message from the associated main memory buffer.

2.8.2 THRESH1_REACHED and THRESH2_REACHED Interrupts

When FGPO_NRECn (the number of records or messages output from memory buffer n) equals the contents of the FGPO_THRESHn register then the associated THRESHn_REACHED bit will be set in the FGPO_IR_STATUS register.

The THRESHn_REACHED condition is ‘sticky’ and can only be cleared by software writing a ‘1’ to the FGPO_IR_CLR.THRESHn_REACHED_ACK bit.

Remark: This interrupt is generated when the FGPO Engine finishes reading the last sample from the threshold record/message number from main memory and NOT when the last sample from the threshold record/message number is output.

2.8.3 UNDERRUN Interrupt

If software fails to assign a new buffer (update FGPO_BASEn register) and perform an interrupt acknowledge (clear BUFnDONE interrupt) before both buffers are done, the interrupt event FGPO_IR_STATUS.UNDERRUN will be set and the output of samples will stop.

This happens when the FGPO switches to a buffer for which:

- a buffer done event has occurred and
- the buffer done interrupt has not been acknowledged and
- the corresponding enable bit is set and
- a new record or message start event has arrived

Output continues upon receipt of either BUF1DONE_ACK or BUF2DONE_ACK or both. Refer to Figure 4 on page 13-11 to see which buffer output resumes from. The UNDERRUN condition is ‘sticky’ and can only be cleared by software writing a ‘1’ to the UNDERRUN_ACK bit.

2.8.4 MBE Interrupt

A Memory Bandwidth Error (MBE) interrupt is generated when no data samples care available during a record or message transfer. During the time MBE state exists the last valid data sample will be output on the fgpo_data pins. Therefore one or more data samples will be added to the message until the adapter FIFO contains valid data samples. Then sample output resumes. For example if FGPO is set to send a message with 6 samples and an MBE occurs before D3 is output, i.e. D3 has not yet

reached the FGPO from the memory, then D2 remains on the data bus until D3 is available. Therefore extra samples are sent and could be detected by FGPI with an OVERFLOW condition is the message has a know lenght.

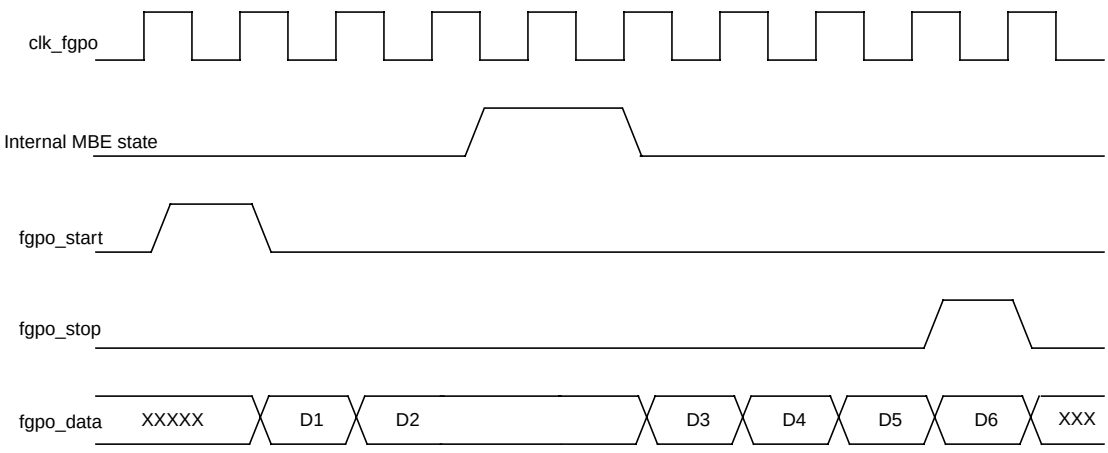


Figure 3: Back-to-back Message Passing Example

The MBE condition is ‘sticky’ and can only be cleared by software writing a ‘1’ to the FGPO_IR_CLR.MBE_ACK bit. However inside FGPO it is edge triggerred, i.e. if the CPU clears the MBE interrupt while FGPO is still in MBE state the sticky bit is indeed cleared and will not get set again unless FGPO gets out of the MBE state and comes back to it. In the later case, yet another MBE interrupt will be generated.

Software must not disable FGPO upon and MBE occuring. It should let FGPO reach the BUF{1,2}DONE state before disabling FGPO.

2.9 Record or Message Counters

The registers FGPO_NREC1 and FGPO_NREC2 count the number of complete records or messages output. The counters are incremented when a record or message stop event is seen. The counters are cleared to zero when the associated FGPO_BASEn register is updated.

Reading a FGPO_NRECn register while the associated buffer is active **MAY NOT RETURN THE ACTUAL TRANSFER COUNT** (can be less than or equal to the actual count) due to clock domain crossing logic. The best time to read a FGPO_NRECn register is during the associated BUFnDONE interrupt service routine as the counter is not updated during this time.

See Section 2.8.2 THRESH1_REACHED and THRESH2_REACHED Interrupts for information on how to use FGPO_NRECn while the associated buffer is active.

2.10 Timestamp

If enabled, by setting FGPO_CTL.TSTAMP_SELECT bit to ‘1’, an 8-byte header is read from memory before the data. The first 32-bit word contains the start time (timestamp) of the record or message. The second 32-bit word may contain the length of the record or message if the VAR_LENGTH bit is set, else the contents are ignored.

The timestamp clock is derived from the main timestamp clock which runs at 13.5 MHz when the GPIO module module is clocked by the 108 MHz clock.

Remark: The length of the header is NOT INCLUDED in the FGPO_REC_SIZE value but MUST be included in the FGPO_STRIDE value.

If both FGPO_CTL.TSTAMP_SELECT and FGPO_CTL.VAR_LENGTH bits are set, then the timestamp word is read from memory before the length word.

Enabling timestamp mode overrides all other buffer and record synchronization.

2.11 Variable Length

If enabled, by setting FGPO_CTL.VAR_LENGTH bit to ‘1’, an 8-byte header is read from memory before the data. The first 32-bit word may contain the start time (timestamp) of the record or message if TSTAMP_SELECT is set, else the contents are ignored. The second 32-bit word contains the length of the record or message.

Remark: The length of the header is NOT INCLUDED in the FGPO_REC_SIZE value but MUST be included in the FGPO_STRIDE value.

If both FGPO_CTL.TSTAMP_SELECT and FGPO_CTL.VAR_LENGTH bits are set, then the timestamp word is read from memory before the length word.

In message mode, if the message length read from the header is greater than the value programmed into the FGPO_REC_SIZE register then the message will be truncated to the length contained in the FGPO_REC_SIZE register.

2.12 Output Time Registers

To help determine the actual time when a transfer took place there are the FGPO_TIME1 and FGPO_TIME2 registers. These registers hold the time when the last sample from a buffer is sent out. This serves to observe the actual departure time in non-timestamp operation modes.

2.13 Double Buffer Operation

Figure 4 shows the major states associated with double buffering. In the following discussion a buffer start event means either the current buffer is done or that an external buffer sync event tells the FGPO to terminate the current buffer and switch to the next buffer. The exact semantics depends on the operating mode of the FGPO.

Upon reset (hardware or software), all status and control bits are placed in the reset condition and no buffer is active. Once software has programmed the required parameters, it is safe to enable output by setting OUTPUT_ENABLE_1 and

OUTPUT_ENABLE_2. Buffer 1 will become the active buffer first. Starting with the next record or message start event samples will be output from buffer 1 until either output is disabled or buffer 1 is terminated by a buffer start event.

Double buffer operation may be terminated by disabling the next buffer to which the FGPO will switch to. This is done by clearing the associated FGPO_CTL.OUTPUT_ENABLE_n bit.

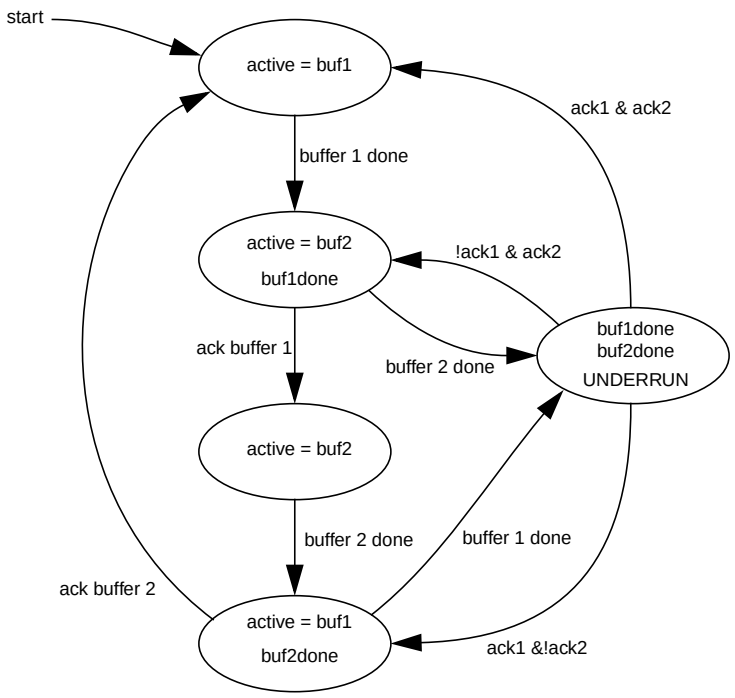


Figure 4: Double Buffer Major States

2.14 Single Buffer Operation

Single buffer operation may be enabled by only setting the FGPO_CTL.OUTPUT_ENABLE_1 bit. When buffer 1 is done, sample output will stop until the BUF1DONE_ACK is received.

3. Operation

3.1 Both Operating Modes

3.1.1 Setup

Initialize all registers except the FGPO_CTL register, first, then load the FGPO_CTL register with the OUTPUT_ENABLE_1 and OUTPUT_ENABLE_2 bits set.

3.1.2 Interrupt Service Routines

Software must update the FGPO_BASEn register value (where n is the number of the buffer that interrupted with a buffer done interrupt) **BEFORE** clearing the buffer done interrupt flag. **This must be done even if the base address of the buffer does not change.**

3.1.3 Optimized DMA Transfers

The DDR Memory controller used in the PNX15xx Series is optimized for 128-byte block transfers on 128-byte address boundaries. To keep Main Memory bus traffic at a minimum the programmer should program the FGPO_BASE1 and FGPO_BASE2 with bits [6:0] = 0000000 and program the FGPO_STRIDE to multiples of 128.

3.1.4 Terminating DMA Transfers

During the next-to-last BUFnDONE interrupt service routine turn off (set to ‘0’) the associated FGPO_CTL.OUTPUT_ENABLE_n bit.

During the last BUFnDONE interrupt service routine turn off (set to ‘0’) the associated FGPO_CTL.OUTPUT_ENABLE_n bit, the FGPO is now IDLE

3.1.5 Signal Edge Definitions

The FGPO uses only the rising edge of clk_fgpo. If the negative edge of an external clock needs to be used, program the PNX15xx Series clock module to invert the external clock for the FGPO.

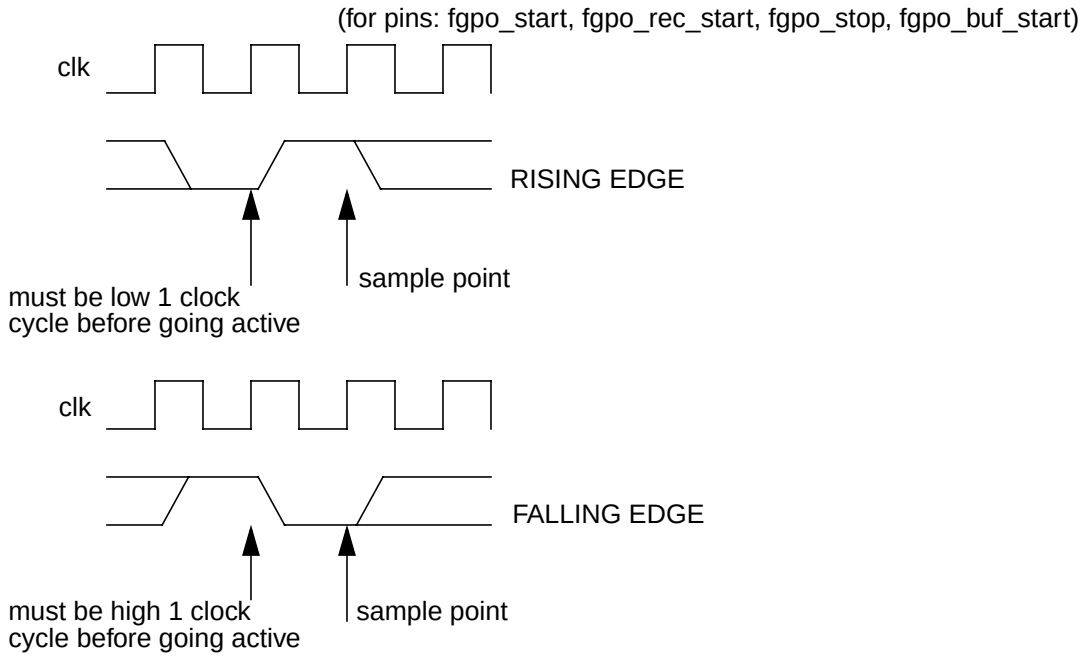


Figure 5: Signal Edge Definition

3.2 Message Passing Mode

If FGPO_REC_SIZE is not a multiple of 4 bytes, the message will be read from main memory as a series of 32-bit words. Only the last word is partially used.

Message start is signaled on the fgpo_start pin and message stop is signaled on the fgpo_stop pin. See FGPO_CTL.MSG_START and FGPO_CTL.MSG_STOP for selecting which edge will be active.

Figure 6 illustrates an example of a two 8-sample message transfer. The message start event is set to the falling edge of fgpo_start and the message stop event is set to the rising edge of fgpo_stop.

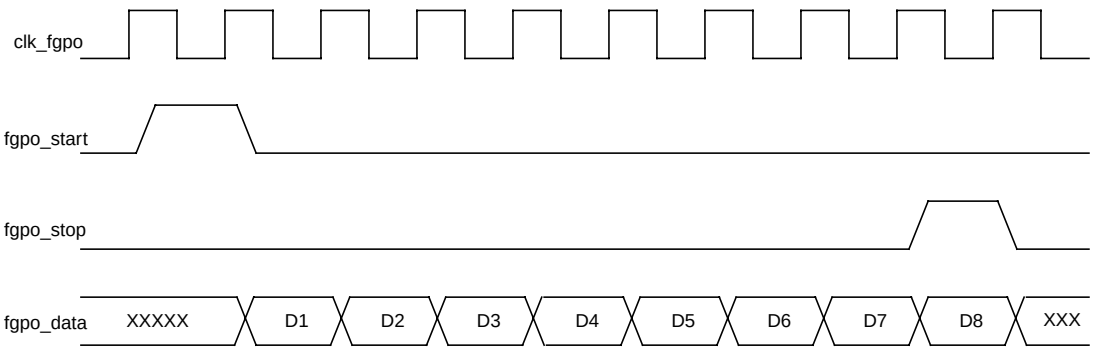


Figure 6: Back-to-back Message Passing Example

Message mode requires both fgpo_start and fgpo_stop signals to operate. External buffer sync is not used with message mode. Buffers are switched when the number of messages sent equals the value programmed into the FGPO_SIZE register.

THE MINIMUM MESSAGE SIZE is 2. FGPO_REC_SIZE must be programmed greater than 1.

If the outgoing message length is greater than the value programmed into the FGPO_REC_SIZE register, the message is truncated.

3.3 PNX1300 Series Message Passing Mode

PNX1300 Series Message Passing mode can be emulated by setting FGPO_SIZE to 1 and only enabling buffer 1 (FGPO_CTL.OUTPUT_ENABLE_1 = ‘1’).

3.4 Record Output Mode

If FGPO_REC_SIZE is not a multiple of 4 bytes, the record will be read from main memory as a series of 32-bit words. Only the last word is partially used.

Record start is signaled on the fgpo_start (fgpo_rec_start) pin. See FGPO_CTL.MSG_START (REC_SYNC) for selecting which edge will be active.

Buffer switching is signaled on the fgpo_stop (fgpo_buf_start) pin. See FGPO_CTL.MSG_STOP (BUF_SYNC) for selecting which buffer sync method will be used.

3.4.1 Record Synchronization Events

Starting output of sample data for each record is signaled by a output start event (selected by the FGPO_CTL.REC_SYNC bits):

- a rising edge on fgpo_rec_sync pin
- a falling edge on fgpo_rec_sync pin
- wait FGPO_REC_GAP clock cycles before starting next record, start first record immediately
- wait for timestamp event
- occur immediately after the previous buffer is sent or when output is enabled

The record ends by reaching the programmed record size in the FGPO_REC_SIZE register or by the next record start event, whichever comes first.

It takes 4 FGPO clock cycles to synchronize and react to events on the fgpo_rec_sync pin in external record sync mode. If timestamps are enabled the output is started on the next FGPO clock tick after the timestamp event.

3.4.2 Buffer Synchronization Events

Each buffer is started by a buffer start event. (selected by the FGPO_CTL.BUF_SYNC bits):

- a rising edge on fgpo_buf_sync pin
- a falling edge on fgpo_buf_sync pin
- alternating rising and falling edges on fgpo_buf_sync pin, starting with the next rising edge on fgpo_buf_sync pin
- alternating rising and falling edges on fgpo_buf_sync pin, starting with the next falling edge on fgpo_buf_sync pin
- wait FGPO_BUF_SYNC clock cycles before starting next buffer, start first buffer immediately
- occur immediately after the previous buffer is sent or when output is started

The fgpo_buf_sync signal will only be observed after the current buffer is finished. It takes 4 FGPO clock cycles to synchronize and react to events on the fgpo_buf_sync pin in external buffer sync mode.

4. Register Descriptions

Table 2: Register Summary

Offset	Name	Clock Domain	Description
0x07,1000	FGPO_CTL	fgpo	Controls operational mode and enables/disables DMA transfers
0x07,1004	FGPO_BASE1	mmio	Starting address for first buffer
0x07,1008	FGPO_BASE2	mmio	Starting address for second buffer
0x07,100C	FGPO_SIZE	fgpo	Number of records/messages per buffer
0x07,1010	FGPO_REC_SIZE	fgpo	Size of record/message in samples
0x07,1014	FGPO_STRIDE	fgpo	Address stride between records/messages
0x07,1018	FGPO_NREC1	mmio	Number of records/messages transferred from buffer 1
0x07,101C	FGPO_NREC2	mmio	Number of records/messages transferred from buffer 2
0x07,1020	FGPO_THRESH1	fgpo	Interrupt Threshold for Buffer 1
0x07,1024	FGPO_THRESH2	fgpo	Interrupt Threshold for Buffer 2
0x07,1028	FGPO_REC_GAP	fgpo	Delay between records/messages
0x07,102C	FGPO_BUF_GAP	fgpo	Delay between buffers
0x07,1030	FGPO_TIME1	fgpo	Timestamp when buffer 1 was finished
0x07,1034	FGPO_TIME2	fgpo	Timestamp when buffer 2 was finished
0x07,1038 - 0x07,1FDC	reserved	n/a	
0x07,1FE0	FGPO_IR_STATUS	mmio	Module Interrupt Status
0x07,1FE4	FGPO_IR_ENA	mmio	Module Interrupt Enables
0x07,1FE8	FGPO_IR_CLR	mmio	Module Interrupt Clear (Interrupt Acknowledge)
0x07,1FEC	FGPO_IR_SET	mmio	Module Interrupt Set (Debug)
0x07,1FF0	FGPO_SOFT_RST	mmio	Module Software Reset
0x07,1FF4	FGPO_IF_DIS	mmio	Module Interface Disable
0x07,1FF8	FGPO_MOD_ID_EXT	mmio	Module ID Extension
0x07,1FFC	FGPO_MOD_ID	mmio	Module ID

4.1 Mode Register Setup

Table 3: Fast general purpose output (FGPO)

Bit	Symbol	Access	Value	Description
FPGO Registers				
Offset 0x07,1000	**FGPO_CTL**			
31:22	Reserved	R	0	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
21	POLARITY_IN	R/W	0	Determines **clk_fgpo** clock sampling edge for **fgpo_rec_sync** and **fgpo_buf_sync** inputs: **0** = use same active edge as for outputs **1** = use alternate active edge as for outputs

Table 3: Fast general purpose output (FGPO) *…Continued*

Bit	Symbol	Access	Value	Description
20:19	BUF_START / MSG_STOP	R/W	00	In Record Mode: Selects the buffer sync output on **fgpo_stop** at the start of a new buffer: **00** = a one clock positive pulse concurrently with the first data sample of each buffer output. **01** = a one clock positive pulse one clock before the first data sample of each buffer output. **10** = a positive pulse asserted when any buffer is active, negated while waiting for a buffer sync. **11** = a positive pulse asserted when data from buffer 2 is valid. In Message Mode: Selects message stop output on **fgpo_stop** at the end of a message: **00** = a one clock positive pulse concurrently with the last data sample for each message **01** = a positive pulse asserted when message data is valid. **10** = same as **00** above **11** = same as **01** above
18	Reserved	R	0	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
17:16	REC_START / MSG_START	R/W	00	Selects record/message start output on **fgpo_start** at the beginning of a record/message: **00** = a one clock positive pulse concurrently with the first data sample of each record/message output. **01** = a one clock positive pulse one clock before the first data sample of each record/message output. **10** = a positive pulse asserted as long as valid data is output, negated while waiting for record/message sync event on **fgpo_rec_sync**. **11** = same as **00** above.
15	Reserved	R	0	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.

Table 3: Fast general purpose output (FGPO) *…Continued*

Bit	Symbol	Access	Value	Description
14	POLARITY_CLK	R/W	0	Externally selects active clock edge for **clk_fgpo** via **fgpo_clk_pol** **0** = rising edge **1** = falling edge **Note:** This bit not used in the PNX15xx Series. All FGPO clock control is in the clock module.
13	OUTPUT_ENABLE_2	R/W	0	Enable output from buffer 2. This bit, along with bit 12 below, start and stop FGPO DMA activity.
12	OUTPUT_ENABLE_1	R/W	0	Enable output from buffer 1. This bit, along with bit 13 above, start and stop FGPO DMA activity. If output from only one buffer is desired, this bit must be used to start/stop DMA.
11	MODE	R/W	0	**0** = record mode **1** = message mode
10	Reserved	R	0	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
9:8	SAMPLE_SIZE	R/W	00	Encodes size of data samples output on **fgpo_data**: **00** = 8-bit data samples **01** = 16-bit data samples **10** = 32-bit data samples **11** = same as **10** above

Table 3: Fast general purpose output (FGPO) *…Continued*

Bit	Symbol	Access	Value	Description
7:5	BUF_SYNC	R/W	000	Encodes function of **fgpo_buf_sync** in record mode. Encodes to 000 in message mode: **000** = No buffer sync, ignores **fgpo_buf_sync** input. Switch to alternate buffer at EOB (End-Of-Buffer). Start first buffer immediately after OUTPUT_ENABLE_1 (bit 12 above) is set. **001** = Wait FGPO_BUF_GAP clock pulses before switch to alternate buffer. Ignores **fgpo_buf_sync** input. Start first buffer immediately after OUTPUT_ENABLE_1 (bit 12 above) is set. **010** = same as **000** above. **011** = same as **001** above. **100** = Switch buffers on rising edge on **fgpo_buf_sync** input. Wait for next rising edge on **fgpo_buf_sync** input, after OUTPUT_ENABLE_1 (bit 12 above) is set, to start first buffer. **101** = Switch buffers on rising edge on **fgpo_buf_sync** input. Wait for next rising edge on **fgpo_buf_sync** input, after OUTPUT_ENABLE_1 (bit 12 above) is set, to start first buffer. **110** = Switch buffers on rising edge on **fgpo_buf_sync** input. Wait for next rising edge on **fgpo_buf_sync** input, after OUTPUT_ENABLE_1 (bit 12 above) is set, to start first buffer. **111** = Switch buffers on rising edge on **fgpo_buf_sync** input. Wait for next rising edge on **fgpo_buf_sync** input, after OUTPUT_ENABLE_1 (bit 12 above) is set, to start first buffer.
4	Reserved	R	0	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
3:2	REC_SYNC	R/W	00	Encodes function of **fgpo_rec_sync** in record/message mode. **00** = No record/message sync, ignores **fgpo_rec_sync** input. Switch to next record at EOR/EOM (End-Of-Record / End-Of-Message). Start first record/message immediately after OUTPUT_ENABLE_1 (bit 12 above) is set. **01** = Wait FGPO_REC_GAP clock pulses before starting next record/message. Ignores **fgpo_buf_sync** input. Start first record/message immediately after OUTPUT_ENABLE_1 (bit 12 above) is set. **10** = Start record/message on **fgpo_rec_sync** rising edge. **11** = Start record/message on **fgpo_rec_sync** falling edge.

Table 3: Fast general purpose output (FGPO) *…Continued*

Bit	Symbol	Access	Value	Description
1	TSTAMP_SELECT	R/W	0	**0** = The REC_SYNC and BUF_SYNC fields control record/message sync and buffer sync events. **1** = Overrides BUF_SYNC to “No Sync mode: 000”, REC_SYNC field is ignored. Causes the FGPO to read an 8-byte record/ message header where the first 4-bytes contains the 32-bit timestamp word. Record/message will start when the internal timestamp matches the timestamp in the header. **Note**: The length of the header (8 bytes) **IS NOT** included in the record/message size in FGPO_SIZE register but **IS** included in the stride (FGPO_STRIDE register).
0	VAR_LENGTH	R/W	0	**0** = The length of each record/message is contained in the FGPO_REC_SIZE register. **1** =Causes the FGPO to read an 8-byte record/message header where the second 4-bytes contains the 32-bit record/message length word. **Record mode**: The value read from the header overrides the contents of the FGPO_REC_SIZE register. **Message mode**: If the message length read from the header is greater than the value in the FGPO_REC_SIZE register then the message is truncated to FGPO_REC_SIZE samples. **Note**: The length of the header (8 bytes) **IS NOT** included in the record/message size in FGPO_SIZE register but **IS** included in the stride (FGPO_STRIDE register).
Offset 0x07,1004	***FGPO_BASE1***			
31:2	BASE1	R/W	0	32-bit word aligned address pointing to Buffer 1 base.
1:0	Reserved	R	0	Always 0.
Offset 0x07,1008	***FGPO_BASE2***			
31:2	BASE2	R/W	0	32-bit word aligned address pointing to Buffer 2 base.
1:0	Reserved	R	0	Always 0.
Offset 0x07,100C	***FGPO_SIZE***			
31:24	Reserved	R	0	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
23:0	SIZE	R/W	0	Number of records/messages per buffer. Range: **1 to 2^{24}-1**
Offset 0x07,1010	***FGPO_REC_SIZE***			
31:24	Reserved	R	0	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
23:0	REC_SIZE	R/W	0	Number of samples per record/message. Range: **2 to 2^{24}-1**
Offset 0x07,1014	***FGPO_STRIDE***			
31:2	STRIDE	R/W	0	Address stride between records/messages

Table 3: Fast general purpose output (FGPO) *…Continued*

Bit	Symbol	Access	Value	Description
1:0	Reserved	R	0	Always 0.
Offset 0x07,1018	***FGPO_NREC1***			
31:24	Reserved	R	0	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
23:0	NREC1	R	0	Number of records/messages output from buffer 1. Cleared to zero when FGPO_BASE1 register is written to.
Offset 0x07,101C	***FGPO_NREC2***			
31:24	Reserved	R	0	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
23:0	NREC2	R	0	Number of records/messages output from buffer 2. Cleared to zero when FGPO_BASE1 register is written to.
Offset 0x07,1020	***FGPO_THRESH1***			
31:24	Reserved	R	0	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
23:0	THRESH1	R/W	0	THRESH1_REACHED interrupt generated when FGPO_NREC1 count equals this register value. Range: **1 to 2^{24}-1**
Offset 0x07,1024	***FGPO_THRESH2***			
31:24	Reserved	R	0	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
23:0	THRESH2	R/W	0	THRESH2_REACHED interrupt generated when FGPO_NREC2 count equals this register value. Range: **1 to 2^{24}-1**
Offset 0x07,1028	***FGPO_REC_GAP***			
31:24	Reserved	R	0	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
23:0	REC_GAP	R/W	0	Clock delay after a record/message is output before the next record/message is output. Range: **1 to 2^{24}-1**
Offset 0x07,102C	***FGPO_BUF_GAP***			
31:24	Reserved	R	0	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
23:0	BUF_GAP	R/W	0	Clock delay after a buffer is output before the next buffer is output. Range: **1 to 2^{24}-1**
Offset 0x07,1030	***FGPO_TIME1***			
31:0	TIME1	R	0	Holds timestamp when buffer 1 completed.
Offset 0x07,1034	***FGPO_TIME2***			
31:0	TIME2	R	0	Holds timestamp when buffer 2 completed.

4.2 Status Registers

Table 4: Status Registers

Bit	Symbol	Access	Value	Description
Standard Registers				
Offset 0x07,1FE0	***FGPO_IR_STATUS***			
31:8	Reserved	R	0	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
7	BUF1_ACTIVE	R	0	1 when Buffer 1 is active
6	Reserved	R	0	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
5	MBE	R	0	Memory Bandwidth Error detected.
4	UNDERRUN	R	0	Buffer Underrun detected.
3	THRESH2_REACHED	R	0	Buffer 2 Threshold reached.
2	THRESH1_REACHED	R	0	Buffer 1 Threshold reached.
1	BUF2_DONE	R	0	Buffer 2 done.
0	BUF1_DONE	R	0	Buffer 1 done.
Offset 0x07,1FE4	***FGPO_IR_ENA***			
31:6	Reserved	R	0	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
5	MBE_ENA	R/W	0	Memory Bandwidth Error Interrupt Enable
4	UNDERRUN_ENA	R/W	0	Buffer Underrun Interrupt Enable
3	THRESH2_REACHED_ENA	R/W	0	Buffer 2 Threshold Interrupt Enable
2	THRESH1_REACHED_ENA	R/W	0	Buffer 2 Threshold Interrupt Enable
1	BUF2_DONE_ENA	R/W	0	Buffer 2 done Interrupt Enable
0	BUF1_DONE_ENA	R/W	0	Buffer 1 done Interrupt Enable
Offset 0x07,1FE8	***FGPO_IR_CLR***			
31:6	Reserved	R	0	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
5	MBE_ACK	R/W	0	Memory Bandwidth Error Interrupt Acknowledge
4	UNDERRUN_ACK	R/W	0	Buffer Underrun Interrupt Acknowledge
3	THRESH2_REACHED_ACK	R/W	0	Buffer 2 Threshold Interrupt Acknowledge
2	THRESH1_REACHED_ACK	R/W	0	Buffer 2 Threshold Interrupt Acknowledge
1	BUF2_DONE_ACK	R/W	0	Buffer 2 done Interrupt Acknowledge
0	BUF1_DONE_ACK	R/W	0	Buffer 1 done Interrupt Acknowledge
Offset 0x07,1FEC	***FGPO_IR_SET***			
31:6	Reserved	R	0	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
5	MBE_SET	R/W	0	Set Memory Bandwidth Error Interrupt
4	UNDERRUN_SET	R/W	0	Set Buffer Underrun Interrupt

Table 4: *…Continued* **Status Registers**

Bit	Symbol	Access	Value	Description
3	THRESH2_REACHED_SET	R/W	0	Set Buffer 2 Threshold Interrupt
2	THRESH1_REACHED_SET	R/W	0	Set Buffer 2 Threshold Interrupt
1	BUF2_DONE_SET	R/W	0	Set Buffer 2 done Interrupt
0	BUF1_DONE_SET	R/W	0	Set Buffer 1 done Interrupt
Offset 0x07,1FF0	***FGPO_SOFT_RST***			
31:1	Reserved	R	0	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
0	SOFTWARE_RESET	R/W	0	**1** = Asserts an internal FGPO reset The effects are: • All FGPO registers are reset • All pending interrupts are removed • Any pending DMA reads are aborted • All state machines return to their reset state ALL CLOCKS MUST BE RUNNING before the soft reset can complete. This bit will clear after the reset completes.
Offset 0x07,1FF4	***FGPO_IF_DIS***			
31:1	Reserved	R	0	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
0	DISABLE_BUS_IF	R/W	0	**1** = All writes to FGPO MMIO space (except this register) will be ignored. All reads (except this register) will return 0x00000000. The FGPO module clock can be stopped low to save power when this bit is set.
Offset 0x07,1FF8	***FGPO_MOD_ID_EXT***			
31:0	MODULE_ID_EXT	R	0	32-bit Module ID Extension
Offset 0x07,1FFC	***FGPO_MOD_ID***			
31:16	MOD_ID	R	0x014C	16-bit Module ID code.
15:12	MAJOR_REV	R	0	4-bit Major Revision code
11:8	MINOR_REV	R	0x2	4-bit Minor Revision code
7:0	APERATURE	R	0	8-bit Aperture code. 0x00 = 4K byte aperture.

Chapter 14: FGPI: Fast General Purpose Interface

PNX15xx Series Data Book – Volume 1 of 1

Rev. 2 — 1 December 2004 Product data sheet

1. Introduction

The Fast General Purpose Input (FGPI) module is a high-bandwidth (up to 400 MBytes/sec) input data channel. The FGPI packs either four 8-bit, or two 16-bit, or one 32-bit data sample(s) into one 32-bit word which is sent to main memory via DMA.

The FGPI operates in two main modes: record capture or message passing

- May be used as a versatile interface with streaming data sources at rates from DC to 100MHz
- May be used as a receiver port for inter-TriMedia unidirectional message passing
- Allows optional synchronization with external control signals
- Allows optional insertion of timestamp into message/record packet sent to memory
- Allows optional insertion of message/record length into message/record packet sent to memory
- Allows continuous data transfer using DMA transfers to two main memory buffers

1.1 FGPI Overview

Refer to Figure 1. This block diagram shows the top level connection of the FGPI module to the MMIO and MTL Busses within the PNX15xx Series. All external FGPI signals are registered and routed through the Input Router module before being presented to the FPGI module. Latency buffering of data and endian conversion is done in the MTL DTL Adapter. All FGPI register access is through the MMIO DTL adapter.

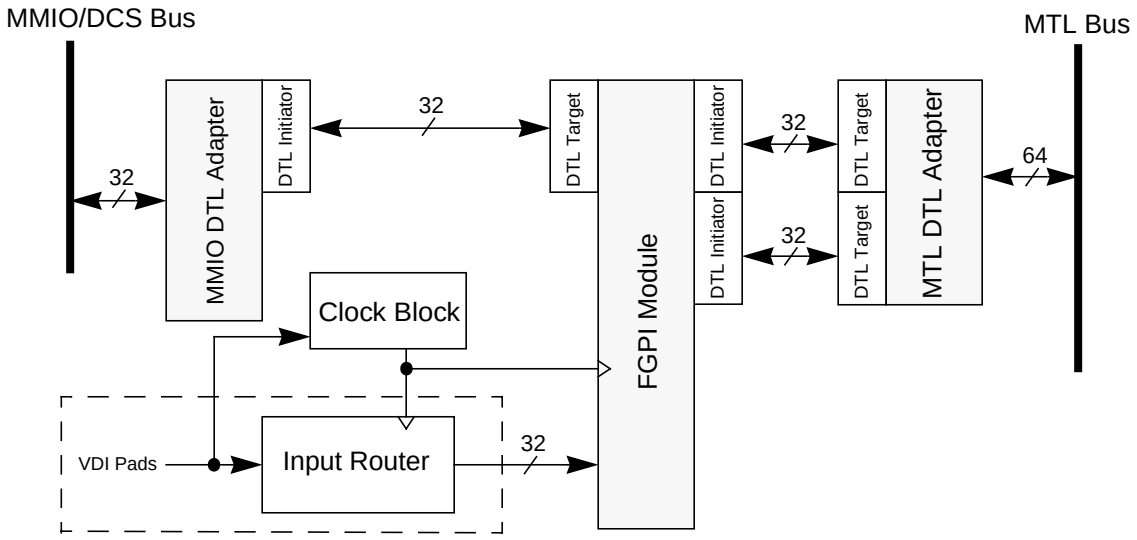


Figure 1: Top Level Block Diagram

Refer to Figure 2. This block diagram shows the basic sections of the FGPI module.

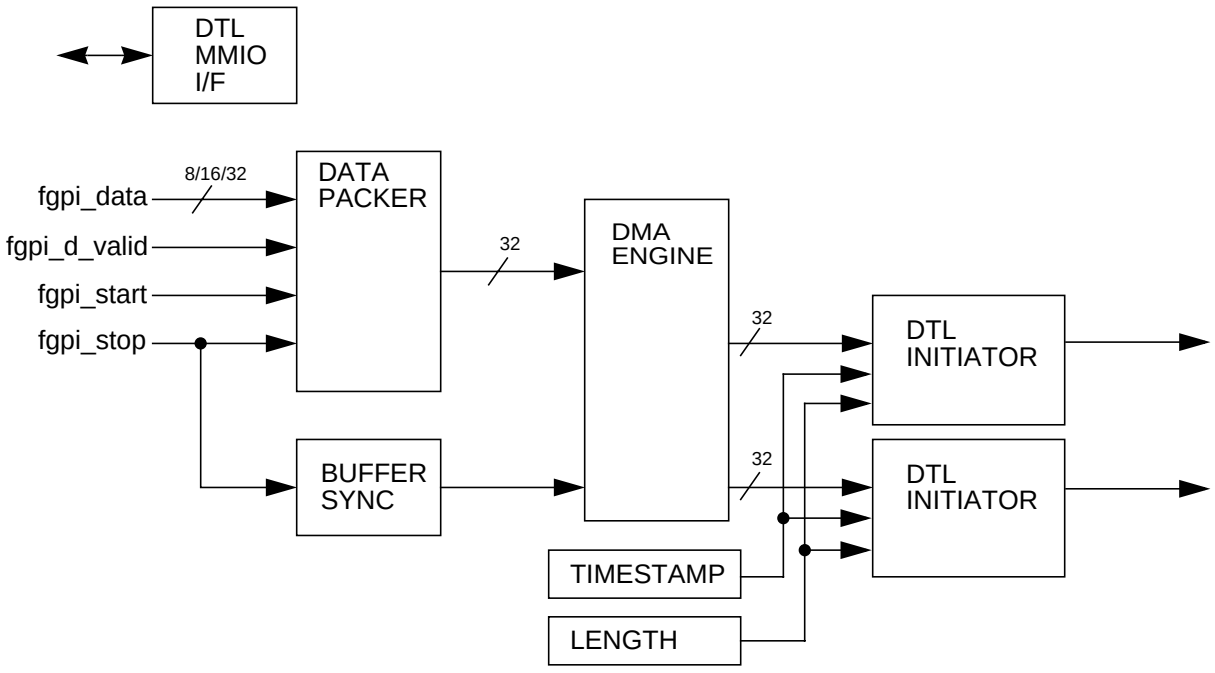


Figure 2: FGPI Module Block Diagram

1.2 VDI to FGPI pin mapping

VDI_D[32] maps to fgpi_start (fgpi_rec_start)

VDI_D[33] maps to fgpi_stop (fgpi_buf_start)

VDI_V2 maps to fgpi_d_valid

VDI_C2 maps to clock module fgpi_clk input

VDI_D[31:0] mapping depends on the VDI_MODE (Input Router) register settings as described in the Chapter 3 System On Chip Resources.

1.3 DTL MMIO Interface

This block contains all of the programmable registers used by the FGPI module accessed through the MMIO bus. Refer to Section 4. on page 14-18 for register descriptions. This block also handles clock domain crossing between the MMIO bus clock and the FGPI module clock.

1.4 Data Packer

This block is used to pack incoming data samples into 32-bit words to be sent to main memory. This module also informs the DMA Engine when a valid 32-bit data word is ready to be loaded into the MTL DTL adapters FIFO via the DTL Initiators.

1.4.1 8-Bit Sample Packing Mode

Sample data received on fgpi_data[7:0] will be packed into one 32-bit word as follows:

sample 1 to word[7:0]

sample 2 to word[15:8]

sample 3 to word[23:16]

sample 4 to word [31:24]

1.4.2 16-bit Sample Packing Mode

Sample data received on fgpi_data[15:0] will be packed into one 32-bit word as follows:

sample 1 to word[15:0]

sample 2 to word[31:16]

1.4.3 32-bit Sample Mode

Sample data received on fgpi_data[31:0] will pass to word[31:0].

1.5 Record Capture Mode

This mode allows the FGPI to receive and store structured record data. The start of a record may be triggered by a transition on the fgpi_start (fgpi_rec_start) pin. The active transition is programmed in the FGPI_CTL register bits [3:2].

Recording starts at a new location in the current buffer for each record received. Successive records are stored in the current buffer until the programmed number of records a buffer contains is reached. At this time the next buffer is activated and subsequent records are loaded into that buffer.

A buffer sync recording mode is available. This mode will switch between alternate buffers on each active transition on the fgpi_stop (fgpi_buf_sync) pin. The active transition is programmed in the FGPI_CTL register bits [7:5]. This allows recording of video frames consisting of multiple line records synchronized by a frame sync of field sync signal.

A continuous raw capture mode is available when FGPI_CTL register bits [7:5] == 100 and bits [4:3] = 10. In this mode data is captured when ever fgpi_d_valid is asserted high.

1.6 Message Passing Mode

In message passing mode the FGPI can act as a receiver of data from the FGPO output from another PNX15xx Series processor. The FGPI can also receive data from a PNX1300 Series VO output in message passing mode.

One FGPO can broadcast to multiple FGPI's by controlling the fgpi_d_valid pin. No data interpretation is done. Each message from the sender is written to a separate memory location in the current buffer. Message start is signaled by fgpi_start pin and message stop is signaled by the fpgi_stop pin.

2. Functional Description

Table 1: Module signal pins

Signal	Type	Description
clk_fgpi	input	From Clock Module. External FGPI clock on VDI_C2 pin is connected to the Clock Module. FGPI data and control signals are sampled at each rising edge on clk_fgpi when fgpi_d_valid is asserted high. Use the PNX15xx Series Clock Module to change clk_fgpi characteristics.
fgpi_d_valid	input	From External PAD, VDI_V2 via Input Router. In all operating modes fgpi_d_valid is used to qualify data & control signals. fgpi_start (fgpi_rec_start), fgpi_stop (fgpi_buf_start), and fgpi_data will only be sampled when fgpi_d_valid is high during the rising edge of clk_fgpi.
fgpi_start or fgpi_rec_start	input	From External PAD, VDI_D[32] via Input Router. Message Passing Mode: A programmable transition on fgpi_start (see FGPI_CTL register bits 3:2) indicates the start of a message. The message starts on the clk_fgpi edge when the transition was detected. Record Capture Mode: A programmable transition on fgpi_rec_start (see FGPI_CTL register bits 3:2) indicates the start of a record. The record starts on the clk_fgpi edge when the transition was detected.
fgpi_stop or fgpi_buf_start	input	From External PAD, VDI_D[33] via Input Router. Message Passing Mode: A programmable transition on fgpi_stop (see FGPI_CTL register bits 7:5) indicates the end of a message. The message ends on the clk_fgpi edge when the transition was detected. Record Capture Mode: A programmable transition on fgpi_buf_start (see FGPI_CTL register bits 7:5) indicates the start of a new buffer. The new buffer starts on the clk_fgpi edge when the transition was detected.
fgpi_data	input	From External PAD's, VDI_D[31:0] via Input Router. General Purpose high speed data input sampled on the rising edge of clk_fgpi when fgpi_d_valid is high.
fgpi_interrupt	output	Interrupt status connects to the TriMedia Processor in the PNX15xx Series.
fgpi_intr_active	output	Not used in the PNX15xx Series.
fgpi_clk_pol	output	Not used in the PNX15xx Series.
fgpi_resetn	output	Goes to the PNX15xx Series Input Router module to reset it's registers used in routing data to the FGPI module.

2.1 Reset

FGPI is reset by any PNX15xx Series system reset or by setting the SOFTWARE_RESET bit in the FGPI_SOFT_RST register.

Remark: SOFTWARE_RESET does not reset MMIO bus interface registers. Any DMA transfers will be aborted during a SOFTWARE_RESET. All registers are reset to the Reset Value shown in the Register Description section.

2.2 Base Addresses

Two base address registers are used to point to main memory buffers in a double buffering scheme. Addresses are forced into 32-bit address alignment.

2.3 Sample (data) Size

Data size (width) per sample is set to either 8, 16, or 32 bits using FGPI_CTL.SAMPLE_SIZE bit field. For 8-bit samples, four samples are packed into one 32-bit word. For 16-bit samples, two samples are packed 2 into one 32-bit word.

Byte order, with which the data is written to memory, is controlled by the global PNX15xx Series endian mode. The endian state only affects 16 and 32-bit sample sizes.

Figure 3 shows how data is stored in memory if data input to the FGPI does not match the setting of the FGPI_CTL.SAMPLE_SIZE bit field. Settings for the PNX15xx Series Input Router will affect the “unknown data” received.

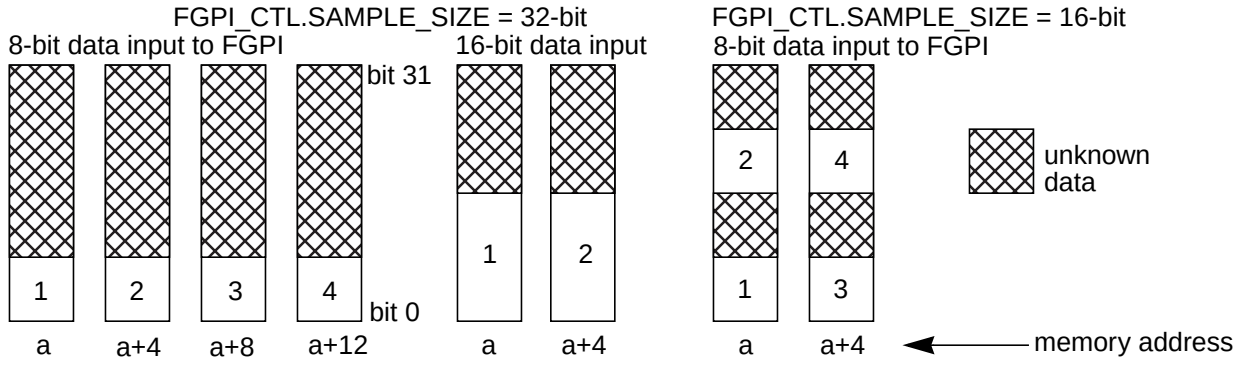


Figure 3: Input data width not equal to sample size setting

2.4 Record or Message Size

In record mode:

The number of samples per record is set by FGPI_REC_SIZE field. This is the amount of samples that will be captured after each record start event.

In message mode:

Maximum number of samples per message is set by FGPI_REC_SIZE field. The end of a message is signaled by the active fgpi_stop edge. If the message length is greater than the programmed value in the FGPI_REC_SIZE register, the message is truncated and a OVERFLOW interrupt is generated.

2.5 Records or Messages Per Buffer

The number of records or messages per buffer is set by FGPI_SIZE register.

2.6 Stride

If the number of records or messages per buffer is greater than one, the address stride has to be programmed into the FGPI_STRIDE register.

Recording starts at a new location in the current buffer on each record or message start event. After recording starts a new address is generated by adding the contents of the FGPI_STRIDE register to the previous starting address.

Care must be taken that FGPI_STRIDE is greater than or equal to FGPI_REC_SIZE. Add 4 if TSTAMP_SELECT is set. Add 4 if VAR_LENGTH is set.

2.7 Interrupt Events

The FGPI_IR_STATUS register contains buffer status and interrupt event status. To generate an interrupt to the TriMedia processor the corresponding FGPI_IR_ENA bit must be set. To clear an interrupt event (acknowledge the interrupt) a ‘1’ must be written to the corresponding FGPI_IR_CLR bit. The FGPI_IR_SET register can be used to generate software interrupts.

2.7.1 BUF1FULL and BUF2FULL Interrupts

When the number of records or messages received and stored in a main memory buffer equals the value in the FGPI_SIZE register an associated Buffer Full interrupt will be generated.

Remark: Received records or messages ARE GUARANTEED to be in main memory when the BUFnDONE interrupt is received.

2.7.2 THRESH1_REACHED and THRESH2_REACHED Interrupts

When FGPI_NRECn (the number of records or messages stored in memory buffer n) equals the contents of the FGPI_THRESHn register then the associated THRESHn_REACHED bit will be set in the FGPI_IR_STATUS register.

The THRESHn_REACHED condition is ‘sticky’ and can only be cleared by software writing a ‘1’ to the FGPI_IR_CLR.THRESHn_REACHED_ACK bit.

WARNING: Received records or messages ARE NOT GUARANTEED to be in main memory when the THRESHn_REACHED interrupt is received. The only interrupt that guarantees that the records or messages are in main memory are the BUFnDONE interrupts.

2.7.3 OVERRUN Interrupt

If software fails to assign a new buffer (update FGPI_BASEn register) and perform an interrupt acknowledge (clear BUFnFULL interrupt) before both buffers fill up, the interrupt event FGPI_IR_STATUS.OVERRUN will be set and capture of samples will stop.

This happens when the FGPI switches to a buffer for which:

- a buffer full event has occurred **and**
- the buffer full interrupt has not been acknowledged **and**
- the corresponding enable bit is set **and**
- a new record or message start event has arrived

Capture continues upon receipt of either BUF1FULL_ACK or BUF2FULL_ACK or both. Refer to Figure 4 on page 14-11 to see which buffer capture resumes in. The OVERRUN condition is ‘sticky’ and can only be cleared by software writing a ‘1’ to the FGPI_IR_CLR.OVERRUN_ACK bit.

2.7.4 MBE Interrupt

A Memory Bandwidth Error (MBE) interrupt is generated when received samples can not be loaded into the main memory adapter FIFO. One or more data samples will be lost until the adapter FIFO can accept samples. Sample capture resumes at the correct address.

The MBE condition is ‘sticky’ and can only be cleared by software writing a ‘1’ to the FGPI_IR_CLR.MBE_ACK bit.

2.7.5 OVERFLOW Interrupt (Message Passing Mode Only)

If the message length overflows the value programmed in the FGPI_REC_SIZE register, the message is truncated and the FGPI_IR_STATUS.OVERFLOW interrupt will be generated.

The OVERFLOW condition is ‘sticky’ and can only be cleared by software writing a ‘1’ to the FGPI_IR_CLR.OVERFLOW_ACK bit.

2.8 Record or Message Counters

The registers FGPI_NREC1 and FGPI_NREC2 count the number of complete records or messages transferred to memory. The counters are incremented when a record or message stop event is seen. The counters are cleared to zero when the associated FGPI_BASEn register is updated.

Reading a FGPI_NRECn register while the associated buffer is active **MAY NOT RETURN THE ACTUAL TRANSFER COUNT** (can be less than or equal to the actual count) due to clock domain crossing logic. The best time to read a FGPI_NRECn register is during the associated BUFnFULL interrupt service routine as the counter is not updated during this time.

See Section 2.7.2 THRESH1_REACHED and THRESH2_REACHED Interrupts for information on how to use FGPI_NRECn while the associated buffer is active.

2.9 Timestamp

If enabled, by setting FGPI_CTL.TSTAMP_SELECT bit to ‘1’, a 4-byte time-of-arrival word giving the record or message start event time is written to main memory before sample data.

The timestamp clock is derived from the main timestamp clock which runs at 13.5 MHz when the GPIO module module is clocked by the 108 MHz clock.

NOTE: The length of the timestamp word is NOT INCLUDED in the FGPI_REC_SIZE value but MUST be included in the FGPI_STRIDE value.

If both FGPI_CTL.TSTAMP_SELECT and FGPI_CTL.VAR_LENGTH bits are set, then the timestamp word is written to memory before the length word.

2.10 Variable Length

If enabled, by setting FGPI_CTL.VAR_LENGTH bit to '1', a 4-byte length of record or message (number of samples received) word is written to main memory before sample data but after the timestamp word (if enabled).

Remark: The length of the VAR_LENGTH word is NOT INCLUDED in the FGPI_REC_SIZE value but MUST be included in the FGPI_STRIDE value.

2.11 Double Buffer Operation

Figure 4 presents the major states associated with double buffering. In the following discussion a buffer start event means either the current buffer is full or that an external buffer sync event tells the FGPI to terminate the current buffer and switch to the next buffer. The exact semantics depends on the operating mode of the FGPI.

Upon a system reset, all status and control bits are placed in the reset condition and no buffer is active. Once software has programmed the required parameters, it is safe to enable capture by setting CAPTURE_ENABLE_1 and CAPTURE_ENABLE_2. Buffer 1 will become the first active buffer. Starting with the next record or message start event samples will be captured in buffer 1 until either capture is disabled or buffer 1 is terminated by a buffer start event. Refer to Figure 5 on page 14-12 for how the FGPI handles buffer termination and switching during a transfer.

When a buffer fills, or is stopped by a buffer start event, the last data sample is tagged by the FGPI so the memory controller will inform the FGPI when the buffer is written to main memory. When the tag is acknowledged the FGPI will issue a BUF1FULL interrupt (if enabled). During this time buffer 2 will be capturing data samples.

Double buffer operation may be terminated by disabling the next buffer to which the FGPI will switch to. This is done by clearing the associated FGPI_CTL.CAPTURE_ENABLE_n bit.

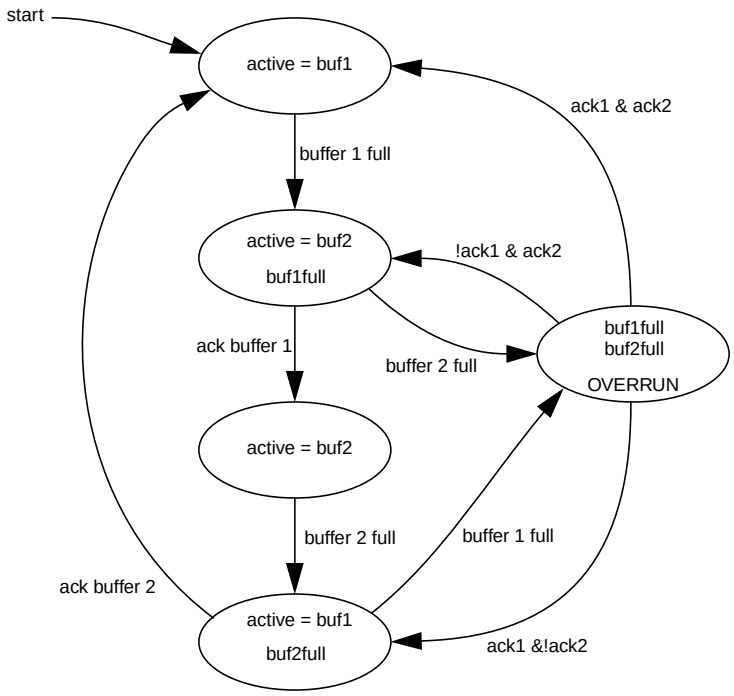


Figure 4: Double Buffer Major States

2.12 Single Buffer Operation

Single buffer operation may be enabled by only setting the FGPI_CTL.CAPTURE_ENABLE_1 bit. When buffer 1 is full, sample capture will stop until the BUF1FULL_ACK is received.

2.13 Buffer Synchronization

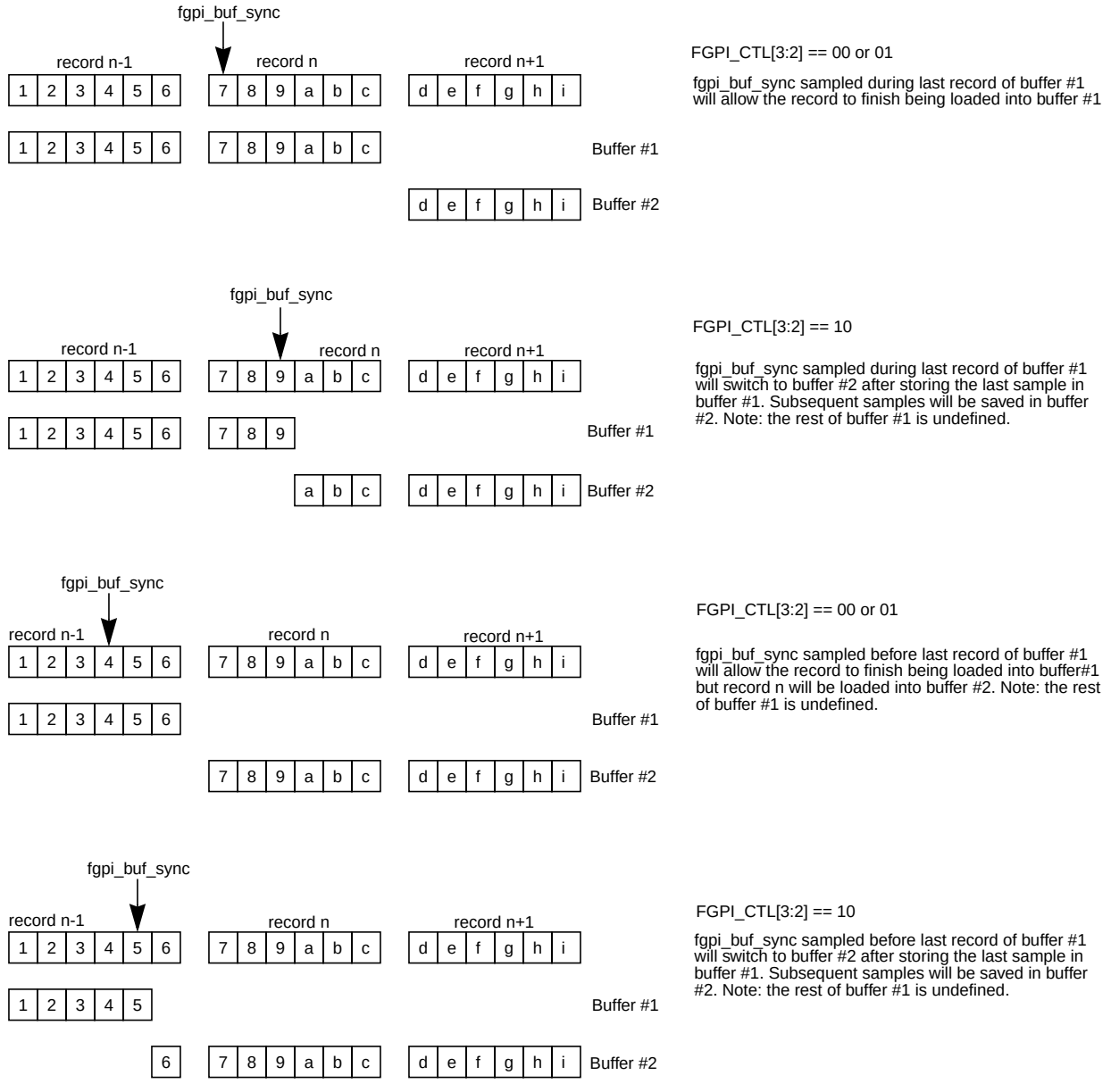


Figure 5: Buffer Sync Actions

3. Operation

3.1 Both Operating Modes

3.1.1 Setup

Initialize all registers, except the FGPI_CTL register, first. Then load the FGPI_CTL register with the CAPTURE_ENABLE_1 and CAPTURE_ENABLE_2 bits set.

3.1.2 Interrupt Service Routines

Software **must** update the FGPI_BASEn register (where n is the number of the buffer that interrupted with a buffer full interrupt) **BEFORE** clearing the buffer full interrupt flag. **This must be done even if the base address of the buffer does not change.**

3.1.3 Optimized DMA Transfers

The DDR Memory controller used in the PNX15xx Series is optimized for 128-byte block transfers on 128-byte address boundaries. To keep Main Memory bus traffic at a minimum the programmer should program the FGPI_BASE1 and FGPI_BASE2 with bits [6:0] = 0000000 and program the FGPI_STRIDE to multiples of 128.

3.1.4 Terminating DMA Transfers

During the next-to-last BUFnFULL interrupt service routine turn off (set to ‘0’) the associated FGPI_CTL.CAPTURE_ENABLE_n bit.

During the last BUFnFULL interrupt service routine turn off (set to ‘0’) the associated FGPI_CTL.CAPTURE_ENABLE_n bit, the FGPI is now IDLE.

3.1.5 Signal Edge Definitions

The FGPI uses only the rising edge of clk_fgpi. If the negative edge of an external clock needs to be used, program the PNX15xx Series clock module to invert the external clock for the FGPI.

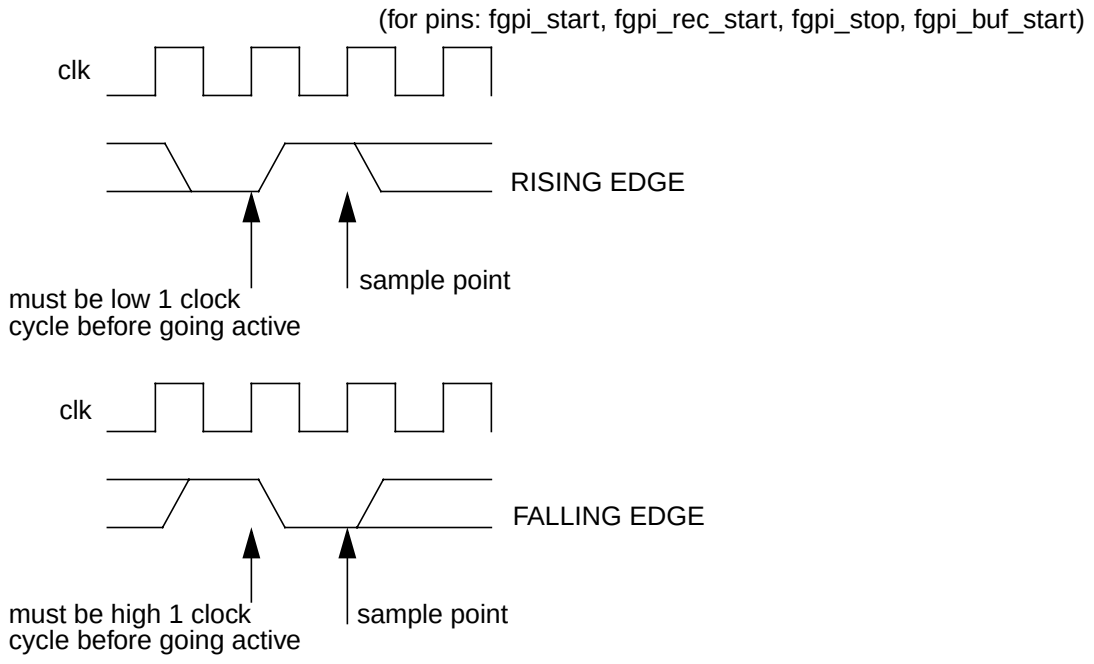


Figure 6: Signal Edge Definition

3.2 Message Passing Mode

Only active clock edges where fgpi_d_valid is asserted ‘1’ allow data samples to be captured.

If FGPI_REC_SIZE is not a multiple of 4 bytes, the message will be written to main memory as a series of 32-bit words. Only the last word is padded with zeroes in the unused bit positions.

Message start is signaled on the fgpi_start pin and message stop is signaled on the fgpi_stop pin. See FGPI_CTL.MSG_START and FGPI_CTL.MSG_STOP for selecting which edge will be active.

Figure 7 illustrates an example of a two 8-sample message transfer. The message start event is set to the falling edge of fgpi_start and the message stop event is set to the rising edge of fgpi_stop.

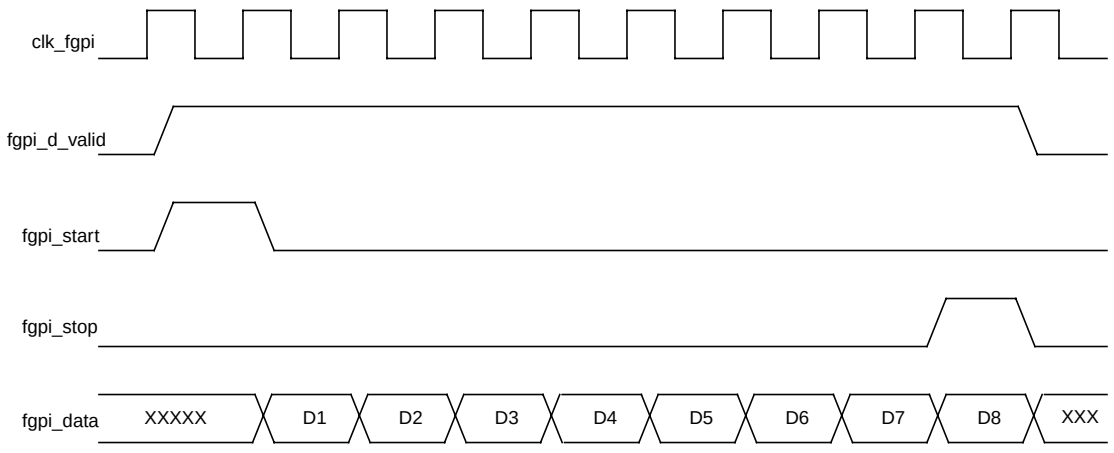


Figure 7: Back-to-back Message Passing Example

Message mode requires both fgpi_start and fgpi_stop signals to operate. There is no external buffer sync available. Buffers are switched when the number of messages received equals the value programmed into the FGPI_SIZE register.

If the incoming message length is greater than the value programmed into the FGPI_REC_SIZE register, the message is truncated and the OVERFLOW interrupt is generated (if enabled).

3.2.1 Minimum Message/Record Size

THE MINIMUM MESSAGE SIZE is 2, that is, FGPI_REC_SIZE must be programmed greater than 1.

There is a limit to the message size when back-to-back messages are received. If the sample size is 32-bits AND TSTAMP_SELECT = ‘1’ AND VAR_LENGTH = ‘1’ then the minimum message size is 3 (due to the insertion of 2 32-bit words into the message packet).

3.3 PNX1300 Series Message Passing Mode

PNX1300 Series Message Passing mode can be emulated by setting FGPI_SIZE to 1 and only enabling buffer 1 (FGPI_CTL.CAPTURE_ENABLE_1 = ‘1’).

3.4 Record Capture Mode

Only active clock edges where fgpi_d_valid is asserted ‘1’ allow data samples to be captured.

If FGPI_REC_SIZE is not a multiple of 4 bytes, the record will be written to main memory as a series of 32-bit words. Only the last word is padded with zeroes in the unused bit positions.

Record start is signaled on the fgpi_start (fgpi_rec_start) pin. See FGPI_CTL.MSG_START (REC_SYNC) for selecting which edge will be active.

Buffer switching is signaled on the fgpi_stop (fgpi_buf_start) pin. See FGPI_CTL.MSG_STOP (BUF_SYNC) for selecting which buffer sync method will be used.

3.4.1 Record Synchronization

Starting capture of sample data for each record is signaled by a record start event (selected by the FGPI_CTL.REC_SYNC bits):

- a rising edge on fgpi_start (fgpi_rec_start) pin
- a falling edge on fgpi_start (fgpi_rec_start) pin
- occur immediately after the previous buffer is filled or when capture is started

(see Section 3.1.5 on page 14-13 for signal definitions for rising and falling edges).

The record ends by reaching the programmed record size in the FGPI_REC_SIZE register or by the next record start event, whichever comes first.

For rising and falling edge record start events the record may reach the programmed record size before the next record start event starts a new record. In this case the FGPI will stop sampling data and wait for the next record start event to continue filling the current buffer. The record may also be terminated by receiving the next record start event before the end of the current record. In this case the record is only partially filled, the content of the rest of the record is undefined.

When no record sync is used data is sampled continuously. This leads to back-to-back recording of consecutive records, independent of the fgpi_start (fgpi_rec_start) pin state. Each record is exactly the programmed size in the FGPI_REC_SIZE register.

3.4.2 Buffer Synchronization

Refer to Figure 5 on page 14-12 for the following section.

It is possible to further synchronize the recording of sample data to a buffer start event. (selected by the FGPI_CTL.BUF_SYNC bits):

- a rising edge on fgpi_stop (fgpi_buf_start) pin
- a falling edge on fgpi_stop (fgpi_buf_start) pin
- alternating rising & falling edges on fgpi_stop (fgpi_buf_start) pin, starting with a rising edge
- alternating rising & falling edges on fgpi_stop (fgpi_buf_start) pin, starting with a falling edge
- occur immediately after the previous buffer is filled or when capture is started

(see Section 3.1.5 on page 14-13 for signal definitions for rising and falling edges).

The buffer start event switches to a new buffer as soon as a concurrent or following record start event is detected IF double buffering is enabled.

For rising, falling, and alternate buffer start events the programmed number of records (FGPI_SIZE) for the current buffer may be reached before the next buffer start event. If this happens the capture of data is stopped until the next buffer and record start events. If a record start event occurs before the buffer start event, it will be ignored. A buffer start event may occur before the next record start event.

For rising, falling, and alternate buffer start events the current buffer may also be terminated by receiving a buffer start event before the programmed number of records (FGPI_SIZE) is reached. If record start event is rising or falling then the current record will finish being loaded into the current buffer before switching to the next buffer. If record start event is ignored the current buffer is only partially filled and the content of the remaining buffer is undefined. Subsequent samples will be stored in the next buffer.

When no buffer sync is used the fgpi_stop (fgpi_buf_start) pin is ignored. In this mode each buffer will contain FGPI_SIZE records and buffer switching occurs immediately after a buffer fills.

When no buffer or record sync is used data samples are sampled and stored continuously. This mode is called “free running” or “raw capture”.

3.4.3 Setup and Operation with Input Router VDI_MODE[7] = 1

See the Global Register specification for more information. Setting the VDI_MODE bit 7 activates an fgpi_data stream pre-processor that extracts the SAV/EAV sync signals (as defined in the video CCIR 656 standard) out of an 8-bit D1 stream and generates the fgpi_start (fgpi_rec_start) and fgpi_stop (fgpi_buf_start) control signals.

FGPI and Input Router Setup Requirements:

- The 8-bit data stream must be applied to the fgpi_data[7:0] pins
- VDI_MODE[7] = 1, VDI_MODE[4:3] = xx (don’t care), VDI_MODE[1:0] = 01
- FGPI_CTL_MODE = 0 (Record Mode)
- FGPI_CTL_SAMPLE_SIZE = 00 (8-bit samples packed 4 samples per 32-bit word)

- FGPI_CTL_BUF_SYNC = 010 (Switch buffers on alternating edges on fgpi_buf_start)
- FGPI_CTL_REC_SYNC = 01 (Start record on falling edge on fgpi_rec_start)

4. Register Descriptions

Table 2: Register Summary

Offset	Name	Clock Domain	Description
0x07,0000	FGPI_CTL	fgpi	Controls operational mode and enables/disables DMA transfers
0x07,0004	FGPI_BASE1	mmio	Starting address for first buffer
0x07,0008	FGPI_BASE2	mmio	Starting address for second buffer
0x07,000C	FGPI_SIZE	fgpi	Number of records/messages per buffer
0x07,0010	FGPI_REC_SIZE	fgpi	Size of record/message in samples
0x07,0014	FGPI_STRIDE	fgpi	Address stride between records/messages
0x07,0018	FGPI_NREC1	mmio	Number of records/messages transferred into buffer 1
0x07,001C	FGPI_NREC2	mmio	Number of records/messages transferred into buffer 2
0x07,0020	FGPI_THRESH1	fgpi	Interrupt Threshold for Buffer 1
0x07,0024	FGPI_THRESH2	fgpi	Interrupt Threshold for Buffer 2
0x07,0028 - 0x07,0FDC	reserved	n/a	
0x07,0FE0	FGPI_IR_STATUS	mmio	Module Interrupt Status
0x07,0FE4	FGPI_IR_ENA	mmio	Module Interrupt Enables
0x07,0FE8	FGPI_IR_CLR	mmio	Module Interrupt Clear (Interrupt Acknowledge)
0x07,0FEC	FGPI_IR_SET	mmio	Module Interrupt Set (Debug)
0x07,0FF0	FGPI_SOFT_RST	mmio	Module Software Reset
0x07,0FF4	FGPI_IF_DIS	mmio	Module Interface Disable
0x07,0FF8	FGPI_MOD_ID_EXT	mmio	Module ID Extension
0x07,0FFC	FGPI_MOD_ID	mmio	Module ID

4.1 Mode Registers

Table 3: Fast general purpose INput (FGPI)

Bit	Symbol	Access	Value	Description
FPGI Registers				
Offset 0x07,0000	**FGPI_CTL**			
31:15	Reserved	R	0	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
14	POLARITY_CLK	R/W	0	Externally selects active clock edge for **clk_fgpi** via **fgpi_clk_pol** **0** = rising edge **1** = falling edge Note: This bit not used in the PNX15xx Series. All FGPI clock control is in the clock module.
13	CAPTURE_ENABLE_2	R/W	0	Enable input capture into buffer 2. This bit, along with bit 12 below, start and stop FGPI DMA activity.

Table 3: Fast general purpose INput (FGPI) *…Continued*

Bit	Symbol	Access	Value	Description
12	CAPTURE_ENABLE_1	R/W	0	Enable input capture into buffer 1. This bit, along with bit 13 above, start and stop FGPI DMA activity. If input capture into only one buffer is desired, this bit must be used to start/stop DMA.
11	MODE	R/W	0	**0** = record/raw capture mode **1** = message capture mode
10	Reserved	R	0	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
9:8	SAMPLE_SIZE	R/W	00	Encodes size of data samples input on **fgpi_data:** **00** = 8-bit data samples **01** = 16-bit data samples **10** = 32-bit data samples **11** = same as **10** above
7:5	BUF_SYNC / MSG_STOP	R/W	000	Encodes function of **fgpi_stop / fgpi_buf_start** pin Record Mode (fgpi_buf_start): **000** = Switch buffers on rising edge on fgpi_buf_start. Wait for next rising edge on fgpi_buf_start to start first buffer. **001** = Switch buffers on falling edge on fgpi_buf_start. Wait for next falling edge on fgpi_buf_start to start first buffer. **010** = Switch buffers on alternating edges on fgpi_buf_start. Wait for next rising edge on fgpi_buf_start to start first buffer. **011** = Switch buffers on alternating edges on fgpi_buf_start. Wait for next falling edge on fgpi_buf_start to start first buffer. **100** = No buffer sync. Switch to alternate buffer at current buffer **E**nd-**O**f-**B**uffer (EOB). Start first buffer immediately after enable. **101 - 111** = same as **100** above. Message Mode (fgpi_stop): **000** = Stop message on rising edge of fgpi_stop **001** = Stop message on falling edge of fgpi_stop **010**, **100**, and **110** = same as **000** above. **011**, **101**, and **111** = same as **001** above.
4	Reserved	R	0	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.

Table 3: Fast general purpose INput (FGPI) *…Continued*

Bit	Symbol	Access	Value	Description
3:2	REC_START / MSG_START	R/W	00	Encodes function of **fgpi_start / fgpi_rec_start** pin Record Mode (fgpi_rec_start): **00** = Start record on rising edge on fgpi_rec_start **01** = Start record on falling edge on fgpi_rec_start **10** = No record sync. Switch to next record if current buffer is full. Start first record immediately after enable. **11** = same as **10** above Message Mode (fgpi_start): **00** = Start message on rising edge of fgpi_start **01** = Start message on falling edge of fgpi_start **10** = same as **00** above. **11** = same as **01** above.
1	TSTAMP_SELECT	R/W	0	**1** = Write Timestamp before record/message data. Timestamp is a 4-byte time-of-arrival for fgpi_rec_start / fgpi_start event. If this bit and bit-0 below (VAR_LENGTH) are both set the Timestamp is written before the length. **Note**: The length of the Timestamp (4 bytes) **IS NOT** included in the record/message size in FGPI_SIZE register but **IS** included in the stride (FGPI_STRIDE register).
0	VAR_LENGTH	R/W	0	**1** = The length of each record/message (in samples) is written before record/message data. Var_length is a 4-byte count. If this bit and bit-1 above (TSTAMP_SELECT) are both set the Timestamp is written before the length. **Note**: The length of var_length (4 bytes) **IS NOT** included in the record/message size in FGPI_SIZE register but **IS** included in the stride (FGPI_STRIDE register).
Offset 0x07,0004	**FGPI_BASE1**			
31:2	BASE1	R/W	0	32-bit word aligned address pointing to Buffer 1 base.
1:0	Reserved	R	0	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
Offset 0x07,0008	**FGPI_BASE2**			
31:2	BASE2	R/W	0	32-bit word aligned address pointing to Buffer 2 base.
1:0	Reserved	R	0	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.

Table 3: Fast general purpose INput (FGPI) *…Continued*

Bit	Symbol	Access	Value	Description
Offset 0x07,000C	**FGPI_SIZE**			
31:24	Reserved	R	0	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
23:0	SIZE	R/W	0	Number of records/messages per buffer. Range: **1 to 2^{24}-1**
Offset 0x07,0010	**FGPI_REC_SIZE**			
31:24	Reserved	R	0	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
23:0	REC_SIZE	R/W	0	Number of samples per record/message. Range: **2 to 2^{24}-1** (See Section 3.2.1 on page 14-14 on minimum REC_SIZE)
Offset 0x07,0014	**FGPI_STRIDE**			
31:2	STRIDE	R/W	0	Address stride between records/messages
1:0	Reserved	R	0	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
Offset 0x07,0018	**FGPI_NREC1**			
31:24	Reserved	R	0	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
23:0	NREC1	R	0	Number of records/messages sent to buffer 1. Cleared to zero when FGPI_BASE1 register is written to.
Offset 0x07,001C	**FGPI_NREC2**			
31:24	Reserved	R	0	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
23:0	NREC2	R	0	Number of records/messages sent to buffer 2. Cleared to zero when FGPI_BASE1 register is written to.
Offset 0x07,0020	**FGPI_THRESH1**			
31:24	Reserved	R	0	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
23:0	THRESH1	R/W	0	THRESH1_REACHED interrupt generated when FGPI_NREC1 count equals this register value. Range: **1 to 2^{24}-1**
Offset 0x07,0024	**FGPI_THRESH2**			
31:24	Reserved	R	0	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
23:0	THRESH2	R/W	0	THRESH2_REACHED interrupt generated when FGPI_NREC2 count equals this register value. Range: **1 to 2^{24}-1**

4.2 Status Registers

Table 4: Status Registers

Bit	Symbol	Access	Value	Description
Standard Registers				
Offset 0x07,0FE0	***FGPI_IR_STATUS***			
31:8	Reserved	R	0	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
7	BUF1_ACTIVE	R	0	1 when Buffer 1 is active
6	OVERFLOW	R	0	Message Overflow Error detected.
5	MBE	R	0	Memory Bandwidth Error detected.
4	UNDERRUN	R	0	Buffer Underrun detected.
3	THRESH2_REACHED	R	0	Buffer 2 Threshold reached.
2	THRESH1_REACHED	R	0	Buffer 1 Threshold reached.
1	BUF2_FULL	R	0	Buffer 2 is full.
0	BUF1_FULL	R	0	Buffer 1 is full.
Offset 0x07,0FE4	***FGPI_IR_ENA***			
31:7	Reserved	R	0	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
6	OVERFLOW_ENA	R/W	0	Message Overflow Error Interrupt Enable
5	MBE_ENA	R/W	0	Memory Bandwidth Error Interrupt Enable
4	UNDERRUN_ENA	R/W	0	Buffer Underrun Interrupt Enable
3	THRESH2_REACHED_ENA	R/W	0	Buffer 2 Threshold Interrupt Enable
2	THRESH1_REACHED_ENA	R/W	0	Buffer 2 Threshold Interrupt Enable
1	BUF2_FULL_ENA	R/W	0	Buffer 2 full Interrupt Enable
0	BUF1_FULL_ENA	R/W	0	Buffer 1 full Interrupt Enable
Offset 0x07,0FE8	***FGPI_IR_CLR***			
31:7	Reserved	R	0	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
6	OVERFLOW_ACK	R/W	0	Message Overflow Error Interrupt Acknowledge
5	MBE_ACK	R/W	0	Memory Bandwidth Error Interrupt Acknowledge
4	UNDERRUN_ACK	R/W	0	Buffer Underrun Interrupt Acknowledge
3	THRESH2_REACHED_ACK	R/W	0	Buffer 2 Threshold Interrupt Acknowledge
2	THRESH1_REACHED_ACK	R/W	0	Buffer 2 Threshold Interrupt Acknowledge
1	BUF2_DONE_ACK	R/W	0	Buffer 2 done Interrupt Acknowledge
0	BUF1_DONE_ACK	R/W	0	Buffer 1 done Interrupt Acknowledge
Offset 0x07,0FEC	***FGPI_IR_SET***			
31:7	Reserved	R	0	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
6	OVERFLOW_SET	R/W	0	Set Message Overflow Error Interrupt

Table 4: Status Registers

Bit	Symbol	Access	Value	Description
5	MBE_SET	R/W	0	Set Memory Bandwidth Error Interrupt
4	UNDERRUN_SET	R/W	0	Set Buffer Underrun Interrupt
3	THRESH2_REACHED_SET	R/W	0	Set Buffer 2 Threshold Interrupt
2	THRESH1_REACHED_SET	R/W	0	Set Buffer 2 Threshold Interrupt
1	BUF2_DONE_SET	R/W	0	Set Buffer 2 done Interrupt
0	BUF1_DONE_SET	R/W	0	Set Buffer 1 done Interrupt
Offset 0x07,0FF0	***FGPI_SOFT_RST***			
31:1	Reserved	R	0	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
0	SOFTWARE_RESET	R/W	0	**1** = Asserts an internal FGPI reset (also output on **fgpi_resetn** pin) The effects are: • All FGPI registers are reset • All pending interrupts are removed • Any pending DMA writes are aborted (FIFO's are cleared) • All state machines return to their reset state ALL CLOCKS MUST BE RUNNING before the soft reset can complete. This bit will clear after the reset completes.
Offset 0x07,0FF4	***FGPI_IF_DIS***			
31:1	Reserved	R	0	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
0	DISABLE_BUS_IF	R/W	0	**1** = All writes to FGPI MMIO space (except this register) will be ignored. All reads (except this register) will return 0x00000000. The FGPI module clock can be stopped low to save power when this bit is set.
Offset 0x07,0FF8	***FGPI_MOD_ID_EXT***			
31:0	MODULE_ID_EXT	R	0	32-bit Module ID Extension
Offset 0x07,0FFC	***FGPI_MOD_ID***			
31:16	MOD_ID	R	0x014B	16-bit Module ID code
15:12	MAJOR_REV	R	0	4-bit Major Revision code
11:8	MINOR_REV	R	0x1	4-bit Minor Revision code
7:0	APERATURE	R	0	8-bit Aperture code. 0x00 = 4K byte aperture

Chapter 15: Audio Output

PNX15xx Series Data Book – Volume 1 of 1

Rev. 2 — 1 December 2004 Product data sheet

1. Introduction

The Audio Output module provides a DMA-driven serial interface designed to support stereo audio D/A converters. The Audio Out module can support up to eight PCM audio channels by driving up to four external stereo D/A converters. The Audio Out unit provides a glueless interface to high quality, low cost oversampling D/A converters. A precise programmable oversampling clock is featured.

1.1 Features

The Audio Out unit, along with the associated external D/A converters, provides the following capabilities:

- Up to 8 channels of audio output
- 16-bit or 32-bit samples per channel
- Programmable 1 Hz to 100 kHz sampling rate
  (Note: This is a practical range. The actual sample rate is application dependent.)
- Internal or external bit clock source
- Autonomously retrieves processed audio data from dual DMA buffers in memory
- 16-bit mono and stereo PC standard memory data formats
- Control capability for highly integrated PC codecs

Remark: AC-97 codecs are not supported.

2. Functional Description

The Audio Out module has four major subsystems: a programmable sample clock generator, a DMA engine, a parallel to serial converter and memory mapped registers for configuration and control. The Audio Out connectors provide the digital audio stream, clock and control signals to external D/A converters. A block diagram of Audio Out is illustrated in Figure 1.

The DMA engine reads 16 or 32-bit samples from memory using dual DMA buffers in memory. Software initially assigns two full sample buffers in memory containing an integral number of samples for all active channels. The DMA engine retrieves samples from the first buffer in memory until exhausted and continues from the

second buffer in memory as a request for a new first sample buffer in memory is issued to the system controller. This action is repeated as the buffers in memory are sent out.

The samples are given to the data serializer (parallel-to-serial converter), which sends them out in a MSB first or LSB first serial frame format that can also contain one or two codec control words of up to 16 bits. The output frame structure is programmable.

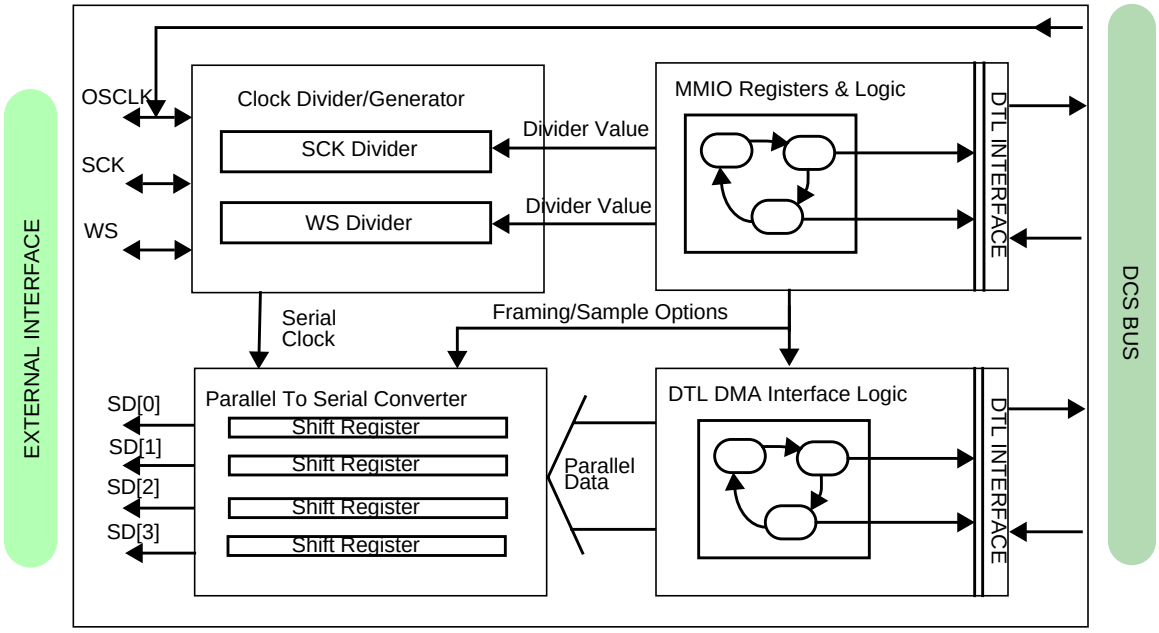


Figure 1: Audio Out Block Diagram

2.1 External Interface

The Audio Out module has seven signals connected to the external world:

- AO_OSCLK referenced as OSCLK in this chapter.
- AO_SCK referenced as SCK in this chapter.
- AO_WS referenced as WS in this chapter.
- AO_SD[3:0] referenced as SD[3:0] in this chapter.

Table 1: Audio Out Unit External Signals[1]

Signal Name	Type	Description
AO_OSCLK	OUT	Oversampling Clock. This output can be programmed to emit any frequency up to 40 MHz. It is intended for use as the 256 Fs or 384 Fs oversampling clock by the external D/A conversion subsystem.
AO_SCK	I/O	Serial Clock. When Audio Out is programmed to act as the serial interface timing slave (RESET default), SCK acts as input. It receives the Serial Clock from the external audio D/A subsystem. The clock is treated as fully asynchronous to the chip main clock. When Audio Out is programmed to act as serial interface timing master, SCK acts as output. It drives the Serial Clock for the external audio D/A subsystem. The clock frequency is a programmable integral divide of the OSCLK frequency. SCK is limited to the frequency of the OSCLK or lower.
AO_WS	I/O	Word Select. When Audio Out is programmed as the serial-interface timing slave (RESET default), WS acts as an input. WS is sampled on the opposite SCK edge at which SD is asserted. When Audio Out is programmed as serial-interface timing master, WS acts as an output. WS is asserted on the same SCK edge as SD. WS is the word select or frame synchronization signal from/to the external D/A subsystem. Each audio channel receives one sample for every WS period. WS can be set to change on OSCLK positive or negative edges by the CLOCK_EDGE bit.
AO_SD[0]	OUT	Serial Data for channel 1. Connect to stereo external audio D/A subsystem. SD[0] can be set to change on OSCLK positive or negative edges by the CLOCK_EDGE bit.
AO_SD[1]	OUT	Serial Data for channel 2. Connect to stereo external audio D/A subsystem. SD[1] can be set to change on OSCLK positive or negative edges by the CLOCK_EDGE bit.
AO_SD[2]	OUT	Serial Data for channel 3. Connect to stereo external audio D/A subsystem. SD[2] can be set to change on OSCLK positive or negative edges by the CLOCK_EDGE bit.
AO_SD[3]	OUT	Serial Data for channel 4. Connect to stereo external audio D/A subsystem. SD[3] can be set to change on OSCLK positive or negative edges by the CLOCK_EDGE bit.

[1] These signals are external to the chip, after the pad cells.

The OSCLK output is an accurate, programmable clock output intended to be used as the master system clock for the external D/A subsystem. The other pins constitute a flexible serial output interface.

SCK - Serial Clock
WS - Word Select

0 = Left Channel
1 = Right Channel

SD[3:0] - Serial Data

Using the Audio Out MMIO registers, these connectors can be configured to operate in a variety of serial interface framing modes, including but not limited to:

- Standard stereo I^2S (MSB first, one bit delay from WS, left and right data in a frame). For further details on I^2S, refer to the "I^2S Bus Specification" dated June 5 1996, in the Multimedia ICs Data Handbook IC22 by Philips Semiconductors, 1998.
- LSB first with 1- to 16-bit data per channel

- Complex serial frames of up to 512 bits/frame

2.2 Memory Data Formats

The Audio Out unit autonomously obtains samples from memory in 16 or 32-bit per sample memory formats, as shown in Figure 2. Successive samples are always read from increasing memory address locations.

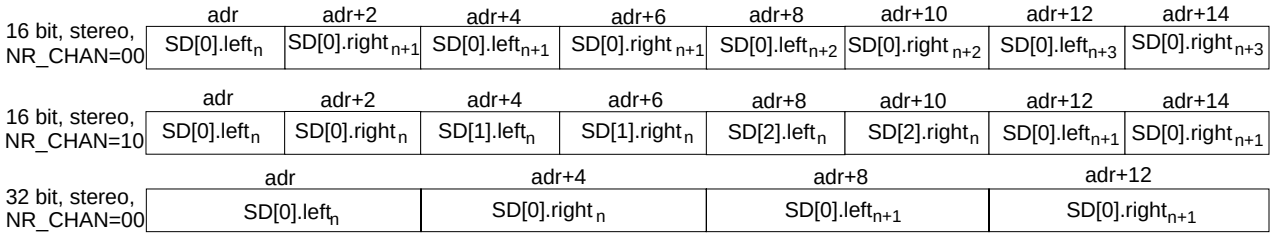


Figure 2: Examples of Audio Out Memory DMA Formats

The Audio Out implements a double buffering scheme in memory to ensure that there are always samples available to transmit, even if the system controller is highly loaded and slow to respond to interrupts. The software assigns two equal size buffers in memory by writing 2 base addresses and a sample size to the corresponding MMIO registers described in Section 4. on page 15-15.

Table 2: Operating Modes and Memory Formats

NR_CHAN	MODE	Destination of Successive Samples
00	mono	SD[0].left
00	stereo	SD[0].left, SD[0].right
01	mono	SD[0].left, SD[1].left
01	stereo	SD[0].left, SD[0].right, SD[1].left, SD[1].right
10	mono	SD[0].left, SD[1].left, SD[2].left
10	stereo	SD[0].left, SD[0].right, SD[1].left, SD[1].right, SD[2].left, SD[2].right
11	mono	SD[0].left, SD[1].left, SD[2].left, SD[3].left
11	stereo	SD[0].left, SD[0].right, SD[1].left, SD[1].right, SD[2].left, SD[2].right, SD[3].left, SD[3].right.

Prior to output transmission, if SIGN_CONVERT = 1, the MSB of the memory data is inverted. This allows the use of external two's complement 16-bit D/A converters to generate audio from 16-bit unsigned samples. This MSB inversion also applies to the '0' values transmitted to non-active output channels.

Note that the Audio Out hardware does **not** support A-law or m-law data formats. If such formats are desired, the system DSP processor should be used to convert from A-law or m-law data to 16-bit linear data.

2.2.1 Endian Control

Audio Out expects data to be supplied in little endian byte ordering mode. In little endian byte ordering mode, the least significant byte has the lowest memory address.

2.3 Audio Out Data DMA Operation

Upon reset, transmission is disabled (TRANS_ENABLE = 0), and buffer 1 in memory is the active buffer (BUF1_ACTIVE = 1). The system software initiates transmission by providing two equal size buffers in memory (filled with valid audio data) and putting the base addresses and sample size in the two AO_BASE*x* registers and the AO_SIZE register. Once two valid buffers are assigned, transmission can be enabled by writing a '1' to TRANS_ENABLE. Note that serial frame configurations should not be changed when transmission is enabled. The Audio Out unit hardware now proceeds to empty buffer 1 in memory by transmission of output samples. The buffers in memory are requested in blocks of up to 1024 32-bit word maximums. When buffer 1 in memory is empty, BUF1_EMPTY is asserted, and transmission continues without interruption from buffer 2 in memory. If BUF1_INTEN is enabled, a level triggered system interrupt request is generated.

Note that the buffers in memory must be 64-byte aligned (the six LSBs of AO_BASE1 and AO_BASE2 are zero). Memory buffer sizes must be in multiples of 64 samples (the six LSBs of AO_SIZE are zero).

The system software is required to assign a new, full buffer to AO_BASE1 and perform an ACK1 before buffer 2 in memory is empty. Transmission continues from buffer 2 in memory until it is empty. At that time, BUF2_EMPTY is asserted, and transmission continues from the new buffer 1 in memory.

An ACK*x* by the interrupt service routine performs two functions:

- It notifies Audio Out that the corresponding AO_BASE*x* register now points to a buffer filled with samples.
- It clears BUF*x*_EMPTY. Upon receipt of an ACK*x*, the Audio Out hardware de asserts the interrupt request line at the next system controller clock edge.

2.3.1 TRANS_ENABLE

The TRANS_ENABLE bit of the AO_CTL register controls the transfer of data from memory and the transmission of data from the serial output pins. While this control bit is enabled, configuration parameters in the register set should not be changed.

When TRANS_ENABLE is disabled after it had been enabled, an abort is sent out to the DMA interface adapter to cancel all DMA transactions associated with the Audio Out and serial transmission as well as WS transmission is stopped.

Before enabling the TRANS_ENABLE bit after an enable, disable sequence, care must be taken that the correct address values are stored in the base address registers. Upon a restart of TRANS_ENABLE, DMA transfers will start from the address given by AO_BASE1. In some modes, a restart of existing values of the base address registers may not produce coherent results.

When TRANS_ENABLE is disabled after a period of being enabled, the status flags for BUF1_ACTIVE, UNDERRUN, HBE, BUF2_EMPTY and BUF1_EMPTY within the AO_STATUS register will retain the values they were at when the disable effect took place. If the TRANS_ENABLE is enabled once again, these flags will be reset to their initial states by the hardware before starting up transmission again.

2.4 Interrupts

Audio Out has a level triggered interrupt request signal. An interrupt is asserted as long as any of the UNDERRUN, HBE, BUF1_EMPTY or BUF2_EMPTY condition flags are asserted. and the corresponding INTEN bit is enabled. Interrupts are sticky (i.e., an interrupt remains asserted until the software explicitly clears the condition flag by an ACK action).

Remark: For legacy reasons, the MMIO interrupt mechanism for Audio Out has not been changed.

2.4.1 Interrupt Latency

During normal error free transmission, the source of an Audio Out interrupt will be from BUF1_EMPTY and BUF2_EMPTY. The DMA buffer size configured in the AO_SIZE register will directly affect the frequency of these ‘empty’ interrupts. Software should set up the AO_SIZE register with a large enough value so that interrupts occur no more frequently than 1 ms in order to meet system interrupt latency requirements.

2.5 Timestamp Events

Audio Out exports event signals associated with audio transmission to the central timestamp/timer function in the system. The central timestamp/timer function can be used to count the number of occurrences of each event or timestamp, the occurrence of the event, or both. The event will be a positive edge pulse with the duration of the event to be greater than or equal to 200 nS. The specific event exported is:

- BUF_DONE — The occurrence of this TSTAMP event indicates that the last word of the current DMA memory buffer has started to be sent out on the serial data port.

This event represents a precise, periodic time interval which can be used by system software for audio/video synchronization.

2.6 Serial Data Framing

The Audio Out unit can generate data in a wide variety of serial data framing conventions. Figure 3 illustrates the notion of a serial frame. If POLARITY = 1, a frame starts with a positive edge of the WS signal. If POLARITY = 0, a serial frame starts with a negative edge on WS. (See Section 4. on page 15-15.) If

CLOCK_EDGE = 0, the parallel-to-serial converter samples WS on a positive clock edge transition and outputs the first bit (bit 0) of a serial frame on the next falling edge of SCK.

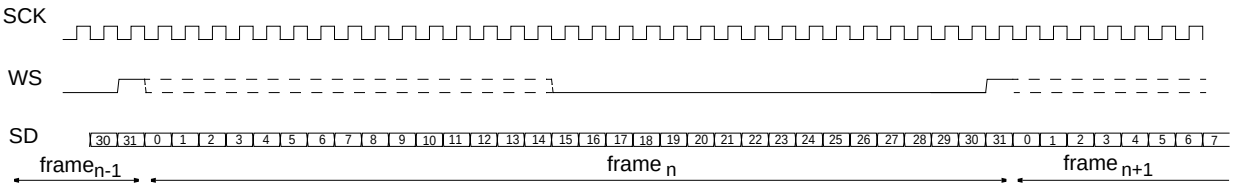


Figure 3: Definition of Serial Frame Bit Positions (POLARITY = 1, CLOCK_EDGE = 0)

If CLOCK_EDGE = 1, the parallel-to-serial converter samples WS on the negative edge of SCK, while audio data is output on the positive edge. That is, the SCK polarity would be reversed with respect to Figure 3.

Every serial frame transmits a single left and right channel sample and optional codec control data to each D/A converter. The sample data can be in an LSB first or MSB first form at an arbitrary serial frame bit position and with an arbitrary length. (See Section 4..)

In MSB first mode (DATAMODE = 0), the parallel-to-serial converter sends the value of LEFT[MSB] in bit position LEFTPOS in the serial frame. Subsequently, bits from decreasing bit positions in the LEFT dataword, up to and including LEFT[SSPOS], are transmitted in order.

In LSB first mode (DATAMODE = 1), the parallel-to-serial converter sends the value of LEFT[SSPOS] in bit position LEFTPOS in the serial frame. Subsequent bits from the LEFT data word, up to and including LEFT[MSB], are transmitted in order. The exact bits transmitted for a data item ‘S’ are shown in Table 3.

Table 3: Bits Transmitted for Each Memory Data Item

Operating Mode	First Bit	Last Bit	Valid SSPOS Values
16 bit/sample, MSB first	S[15]	S[SSPOS]	0...15
16 bit/sample, LSB first	S[SSPOS]	S[15]	0...15
32 bit/sample, MSB first	S[31]	S[SSPOS]	0...31
32 bit/sample, LSB first	S[SSPOS]	S[31]	0...31

Frame bits that do not belong to either LEFT[MSB:SSPOS] or RIGHT[MSB:SSPOS] or a codec control field (Section 2.7 on page 15-9) are shifted out as zero. This zero extension ensures that Audio Out can be used in combination with D/A converters which expect more bits than the actual number of transmitted bits in the current operating mode (e.g., 18-bit D/As operating with 16-bit memory data).

If a starting position for a field is defined at a position such that the end of a serial frame is reached before the end of the data field is reached, the data beyond the last bit of the serial frame is truncated.

2.6.1 Serial Frame Limitations

Due to the implementation, there is a minimum serial frame length requirement that is operating-mode dependent according to Table 4.

Table 4: Minimum Serial Frame Length in Bits

Operating Mode	Minimum Serial Frame Length
16 bit/sample, mono	13 bits
32 bit/sample, mono	13 bits
16 bit/sample, stereo	13 bits
32 bit/sample, stereo	36 bits

2.6.2 WS Characteristics

The WS signal is used to define the start point of a serial frame. The start of a frame can be marked by a transition on WS 1 clock before the start of the first bit of a new frame. This WS transition can be programmed to be either a positive or a negative edge transition.

In addition, the WS signal can be programmed to be a 50% duty cycle wave form or a pulse that is 1 clock cycle wide. If the WS is configured to be 1 single clock wide pulse, the pulse spans the 2 clock cycles preceding the first bit of the new frame. If the WS is configured to be a 50% duty cycle waveform, the active edge of the WS signal occurs 1 clock cycle before the first bit of the new frame. If the serial frame is of an even bit count, then the second transition of the WS signal occurs in the clock cycle before the halfway point of the serial frame. If the serial frame is of an odd bit count, then the portion of the WS wave that is in the low state has the extra clock cycle.

With an odd bit count for a serial frame and with a frame starting with a negative edge of the WS, a 50% duty cycle WS signal would have the first part of the WS signal 1 clock longer than the second half. With a positive edge of the WS signal marking the start of a serial frame, the second half of the serial frame would have a WS signal longer by 1 clock cycle.

2.6.3 I^2S Serial Framing Example

Figure 4 and Table 5 show how the Audio Out module MMIO registers should be set to transmit 16 or 32 bits of stereo data via an I^2S serial standard to an 18-bit D/A converter with a 64-bit serial frame.

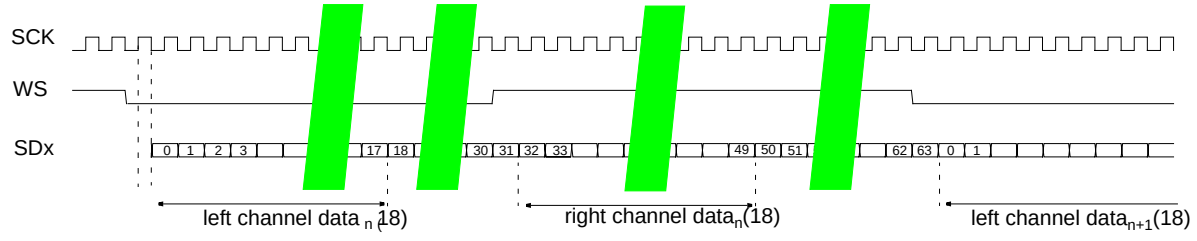


Figure 4: Serial Frame (64 Bits) of a 18-Bit Precision I^2S D/A Converter

Table 5: Example Setup For 64-Bit I²S Framing

Field	Value	Explanation
POLARITY	0	Frame starts with negative edge Audio Out WS.
LEFTPOS	0	LEFT[MSB] will go to serial frame position 0.
RIGHTPOS	32	RIGHT[MSB] will go to serial frame position 32.
DATAMODE	0	MSB first.
SSPOS	0	Stop with LEFT/RIGHT[0], send 0s after. (For 32-bit/sample mode, this field could be set to 14 to ensure zeroes in all unused bit positions.)
CLOCK_EDGE	0	Audio Out SD change on negative edge Audio Out SCK
WSDIV	63	Serial frame length = 64
WS_PULSE	0	Emit 50% duty cycle Audio Out WS.

Remark: The transfer of data from SDRAM into the Audio Out module's transmit FIFOs is initiated by the transition of WS, not by transmit enable. As a consequence, there is a delay between the receipt of the **first** WS pulse and the transmission of the **first** data. The length of this delay is dependent on system load and is not easily predictable.

2.7 Codec Control

In addition to the left and right data fields that are generated based on autonomous DMA action, a serial frame generated by Audio Out can be set to contain one or two control fields of up to 16 bits in length. Each control field can be independently enabled or disabled by the CC1_EN, CC2_EN bits in AO_CTL.

The content shifted into the frame is taken from the CC1 and CC2 field in the AO_CC register. The CC1_POS and CC2_POS fields in the AO_CFC register determine the first bit position in the frame where the control field is emitted observing the setting of DATAMODE (i.e., LSB or MSB first).

The CC_BUSY bit in AO_STATUS indicates if the Audio Out unit is ready to receive another CC1, CC2 value pair. Writing a new value pair to AO_CC writes the value into a buffer register and raises the CC_BUSY status. (See Section 4. on page 15-15.) As soon as both CC1 and CC2 values have been copied to a shadow register in preparation for transmission, CC_BUSY is negated, indicating that the Audio Out logic is ready to accept a new codec control pair. The old CC1/CC2 data is transmitted continuously (i.e., software is not required to provide new CC1 and CC2 data).

Software **must ensure** that the CC_BUSY status is negated before writing a new CC1, CC2 pair. The user, by the process of waiting on CC_BUSY, can reliably emit a sequence of individual audio frames with distinct control field values. This can, for example, be used during codec initialization. Note that no provision is made for interrupt-driven operation of such a sequence of control values. It is assumed, after initialization, that the value of control fields determines slowly changing, asynchronous parameters such as output volume.

It is legal to program the control field positions within the frame such that CC1 and CC2 overlap each other and/or left and right data fields. If two fields are defined to start at the same bit position, the priority is left (highest), right, CC1 then CC2. The field with the highest priority will be emitted starting at the conflicting bit position. If a field *f2* is defined to start at a bit position *i* that falls within a field *f1* starting at a lower bit position, *f2* will be emitted starting from *i* and the rest of *f1* will be lost. Any bit positions not belonging to a data or control field will be emitted as zero.

If a field is defined to start at a bit position such that the end of the field goes beyond the end of the frame, the data beyond the end of the frame (as defined by the active edge of WS) is lost.

Figure 5 shows a 64-bit frame suitable for use with the CS4218 codec. It is obtained by setting POLARITY=1, LEFTPOS=0, RIGHTPOS=32, DATAMODE=0, SSPOS=0, CLOCK_EDGE=1, WS_PULSE=1, CC1_POS= 16, CC1_EN=1, CC2_POS=48 and CC2_EN=1.

Note that frames are generated (externally or internally) even when TRANS_ENABLE is de-asserted. Writes to CC1 and CC2 should only be done after TRANS_ENABLE is asserted. The ‘first’ CC values will then go out on the next frame.

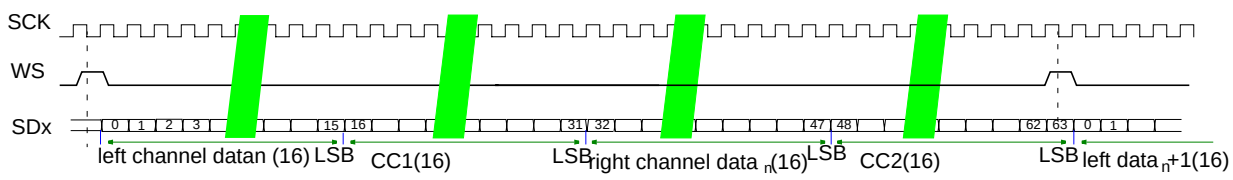


Figure 5: Example Codec Frame Layout for a Crystal Semiconductor CS4218

2.8 Data Bus Latency and HBE

The Audio Out unit relies on the FIFO buffers within the DMA interface adapter as well as an output holding register that holds a single mono sample or single stereo sample pair. For Audio Out there are four separate stereo output channels and each output channel has one output holding register. The holding register width is 64 bits.

Under normal operation, the DMA interface adapter provides samples from memory fast enough to avoid any missing samples. Meanwhile, data is being emitted from one 64-byte hardware buffer and holding register. If the data bus arbiter is set up with an insufficient latency guarantee, the situation can arise that the hardware FIFO buffer within the DMA interface adapter is not refilled in time and the buffer and holding register are exhausted by the time a new output sample is due. In that case the HBE flag is raised to indicate a bandwidth error. The last sample for each channel will be repeated until the buffer is refreshed. The HBE condition is sticky and can only be cleared by an explicit ACK_HBE. This condition indicates an incorrect setting of the data bus bandwidth arbiter.

Table 6 shows the maximum tolerable latency for a number of common operating modes. The right most column in the table indicates the maximum tolerable latency for Audio Out under normal operating condition. To sustain error free audio playback, one 64-byte DMA transfer must be completed within the maximum latency period

indicated for each operating mode. Note that for high sample rates with more than four channels (96 kHz, 32-bit, >4ch), Audio Out cannot guarantee error-free operation due to data bus latency restrictions.

Table 6: Audio Out Latency Tolerance Examples

Transfer Mode	Fs (kHz)	T (nSec)	* Max Latency (uSec) (with T=1/Fs)
2 ch stereo 16 bit/sample	48.0	20833	354.17
4 ch stereo 16 bit/sample	48.0	20833	187.50
6 ch stereo 16 bit/sample	48.0	20833	104.17
8 ch stereo 16 bit/sample	48.0	20833	104.17
2 ch stereo 32 bit/sample	48.0	20833	187.50
4 ch stereo 32 bit/sample	48.0	20833	104.17
6 ch stereo 32 bit/sample	48.0	20833	62.50
8 ch stereo 32 bit/sample	48.0	20833	62.50
2 ch stereo 16 bit/sample	96.0	10417	177.08
4 ch stereo 16 bit/sample	96.0	10417	93.75
6 ch stereo 16 bit/sample	96.0	10417	52.08
8 ch stereo 16 bit/sample	96.0	10417	52.08
2 ch stereo 32 bit/sample	96.0	10417	93.75
4 ch stereo 32 bit/sample	96.0	10417	52.08

* Max Latency (uSec) (with T=1/Fs)

2ch 16it stereo: (17 * T)2ch 32bit stereo:(9 * T)
4ch 16bit stereo:(9 * T)4ch 32bit stereo:(5 * T)
6ch 16bit stereo: (5 * T)6ch 32bit stereo:(3 * T)
8ch 16bit stereo:(5 * T)8ch 32bit stereo:(3 * T)

2.9 Error Behavior

In normal operation, the system controller and Audio Out hardware continuously exchange buffers without ever failing to transmit a sample. If the system controller fails to provide a new DMA buffer address in time, the UNDERRUN error flag is raised and the last valid sample or sample pair is repeated until a new buffer of data is assigned by an ACK1 or ACK2. The UNDERRUN flag is *not affected* by ACK1 or ACK2; it can only be cleared by an explicit ACK_UDR.

If an HBE error occurs, the last valid sample or sample pair is repeated until the Audio Out hardware receives enough data to generate a new serial data frame from the DMA interface adapter.

3. Operation

3.1 Clock Programming

3.1.1 Sample Clock Generator

The sample clock generator is programmable to support various sample frequencies. Figure 6 illustrates the different clock capabilities of the Audio Out unit. A square wave Direct Digital Synthesizer (DDS) drives the clock system. This DDS is external to the Audio Out block.

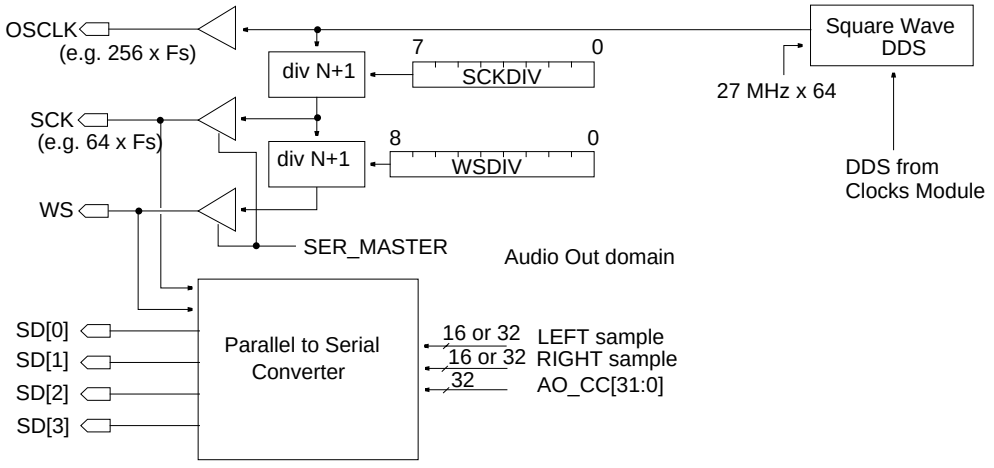


Figure 6: Audio Out Clock System and I/O Interface

Using the DDS as a clock source allows software to control the coarse and fine clock rate so that complex forms of synchronization can be implemented without external hardware. Examples include locking the audio to a broadcast clock, or locking it to an SPDIF input without changing the system's hardware.

The DDS output is always sent to the OSCLK output pin. This output is intended to be used as the 256 Fs or 384 Fs system clock source for oversampling D/A converters.

Software may change the oversampling clock frequency dynamically (via the DDS controls) to adjust the outgoing audio sample rate. In ATSC transport stream decoding, this is the method used by which the system software locks the audio output sample rate to the original program provider sample rate.

Table 7 presents several sample rates with the SCKDIV setting necessary to achieve a bit clock of 64 Fs.

Table 7: Clock System Setting

Fs	OSCLK	SCKDIV	SCK
44.1 kHz	256 Fs	3	64 Fs
48.0 kHz	256 Fs	3	64 Fs
44.1 kHz	384 Fs	5	64 Fs
48.0 kHz	384 Fs	5	64 Fs

The values of SCKDIV given assume the oversampling clock supplied to Audio Out is either 256 Fs or 384 Fs. The value of SCKDIV is determined by Equation 11.

$$f_{SCK} = \frac{f_{OSCLK}}{SCKDIV + 1} \quad (11)$$

Remark: SCKDIV is in the range of 0-255.

3.1.2 Clock System Operation

Word Select (WS) and Serial Clock (SCK) are sent to the external D/A converter in the master mode. WS determines the sample rate: each active channel receives one sample for each WS period. SCK is the data bit clock. The number of SCK clocks in a WS period is the number of data bits in a serial frame required by the attached D/A converter.

WS is derived from the SCK bit clock. It is controlled by the value of WSDIV and it sets the serial frame length. The number of bits per frame is equal to WSDIV+1. There are some minimum length requirements for a serial frame. Refer to Section 2.6.1 on page 15-8 for details.

SCK and WS can be configured as input or output by the SER_MASTER control field in the AO_SERIAL register. If configured as an output, SCK can be set to a divider of the OSCLK output frequency. (See Section 4. on page 15-15 for more details.)

Whether set as input or output, the SCK connector is always used as the bit clock for parallel to serial conversion. The WS signal always acts as the trigger to start the generation of a serial frame. WS can also be programmed using WSDIV to control the serial frame length. The number of bits per frame is equal to WSDIV+1.

If the serial frame length is set to be an odd number of bits and the WS pulse is programmed to be 50% duty cycle, the portion of the WS waveform that is in the low state will have the extra clock bit.

The preferred configuration of the clock system options is to use OSCLK as the D/A subsystem master clock and let the D/A subsystem be a timing slave to the serial interface (SER_MASTER = 1).

Some D/A converters provide somewhat better SNR properties if they are configured as serial masters, so the Audio Out should be configured as a slave in this case (SER_MASTER = 0). As illustrated by Figure 6, the internal parallel to serial converter that constructs the serial frame is indifferent as to who is the serial master.

3.2 Reset-Related Issues

Audio Out is reset by the system reset signal or by writing AO_CTL.RESET = 1. Either reset method sets all MMIO fields to their default values as indicated in the Section 4. Register Descriptions.

When software reset is activated (AO_CTL.RESET = 1) the clock generation using the values programmed in the internal MMIO register (SER_MASTER, AO_SERIAL, ...) is stopped since these values get reset. This causes the AO module to do not have a clock anymore which does not complete the reset. Therefore before resetting the AO module, the AO module should be clocked by the crystal 27 MHz clock.

Software must follow a series of steps to ensure that Software Reset happens correctly:

1. Check to see if there is a valid clock present on the Audio Out external clock input.

2. If there is no clock, then write to the clocks block to switch the Audio Out block clock input to the 27 MHz oscillator.

3. Apply Software Reset and poll the Reset bit until it is cleared.

4. Program the Clocks block to switch back to the external clock mode for the Audio Out block.

Clocks are required to be running during HW/SW reset because synchronous reset is used to initialize the logic. The unique feature with Audio is that unlike all other blocks in the system, the Audio blocks default to the external clock source on any reset. If the external clock does not exist when a HW reset is applied, then the logic is left uninitialized without any indication.

3.3 Register Programming Guidelines

- When configuring the Audio Out, data framing options and DMA buffer parameters must be programmed before enabling transmission.

3.4 Power Management

Audio Out has no internal powerdown functionality, however the chip power management software can remove the main block level clock to Audio Out. If the Audio Out module enters a powerdown state, SCK, WS and SD[*x*] hold their value stable but OSCLK continues to provide a D/A converter oversampling clock.

Once the system wakes up, the signals resume their original transitions at the point where they were halted. The external D/A converter subsystem will most likely be confused by this behavior, so it is recommended that Audio Out transmissions be stopped (by deactivating TRANS_ENABLE) prior to software enable of Audio Out powerdown.

4. Register Descriptions

The following tables illustrate the register set of the Audio Out block. Access to these registers is via the DTL port. These registers are distributed between the DTL clock domain (DCS/MMIO bus), the DMA clock domain (MTL/DDR bus) and the IP clock domain, i.e. the AO module clock. The access time to these registers is proportional to the clock domain frequency with respect to the CPU speed.

4.1 Register Summary

Table 8: Register Summary

Offset	Name	Description
0x11 0000	AO_STATUS	Provides status of buffers and other Audio Out components/situations.
0x11 0004	AO_CTL	Control register to configure Audio Out options
0x11 0008	AO_SERIAL	Control register to configure Audio Out serial timing and data options
0x11 000C	AO_FRAMING	Control register to configure data framing
0x11 0010	Reserved	
0x11 0014	AO_BASE1	Base address of DMA buffer 1 in memory
0x11 0018	AO_BASE2	Base address of DMA buffer 2 in memory
0x11 001C	AO_SIZE	The DMA Buffer size in samples
0x11 0020	AO_CC	Codec Control data content
0x11 0024	AO_CFC	Codec data position
0x11 0028—0FF0	Reserved	
0x11 0FF4	AO_PWR_DWN	Powerdown function
0x11 0FFC	AO_MODULE_ID	Provides module ID number, including major and minor revision levels.

4.2 Register Table

Table 9: Audio Output Port Registers

Bit	Symbol	Access	Value	Description
Offset 0x11 0000	***AO_STATUS—DTL Clock Domain***			
31:6	Unused		-	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
5	CC_BUSY	R	0	0 = Audio Out is ready to receive a CC1, CC2 pair. 1 = Audio Out is not ready to receive a CC1, CC2 pair. Try again in a few SCK clock intervals.
4	BUF1_ACTIVE	R	1	1 =DMA buffer 1 in memory will be used for the next sample to be transmitted. 0 =DMA buffer 2 in memory will contain the next sample.

Table 9: Audio Output Port Registers *…Continued*

Bit	Symbol	Access	Value	Description
3	UNDERRUN	R	0	An UNDERRUN error has occurred i.e., the system controller/ software failed to provide a full DMA buffer in memory in time and no new samples were transmitted. The last sample or sample pairs were sent to the D/A converter. If UDR_INTEN is also '1', an interrupt request is pending. The UNDERRUN flag can ONLY be cleared by writing an '1' to ACK_UDR.
2	HBE	R	0	Bandwidth Error indicates that no new data was transmitted due to an inability of the DMA interface adapter to provide the required data in time for the start of a new frame. The last data set is repeated. If HBE_INTEN is an '1', then an interrupt is also sent to the system. The HBE flag stays set until an '1' is written to ACK_HBE.
1	BUF2_EMPTY	R	0	1= buffer 2 is empty. If BUF2_INTEN is also '1', an interrupt request is asserted. BUF2_EMPTY is cleared by writing a '1' to ACK2, at which point the Audio Out hardware will assume that AO_BASE2 and AO_SIZE describe a new full DMA buffer in memory.
0	BUF1_EMPTY	R	0	1= buffer 1 is empty. If BUF1_INTEN is also '1', an interrupt request is asserted. BUF1_EMPTY is cleared by writing a '1' to ACK1, at which point the Audio Out hardware will assume that AO_BASE1 and AO_SIZE describe a new full DMA buffer in memory.
Offset 0x11 0004	***AO_CTL—DTL Clock Domain***			
31	RESET	R/W	0	Resets the Audio Out logic. See Section 3.2 on page 15-14 for a description of software reset.
30	TRANS_ENABLE	R/W	0	Transmission Enable flag 0 = Audio Out is inactive. 1 = Audio Out transmits samples and acts as DMA master to read samples from local memory. Do not change any of the framing configurations while transmission is enabled.
29:28	TRANS_MODE	R/W	00	00 = Mono, 32 bits/sample. Left and right data sent to each active output are the same. 01 = Stereo, 32 bits/sample 10 = Mono, 16 bits/sample. Left and right data are the same. 11 = Stereo, 16 bits/sample
27	SIGN_CONVERT	R/W	0	0 =Leave MSB unchanged. 1 = Invert MSB (not applied to codec control fields).
26:25	Unused		-	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read
24	CC1_EN	R/W	0	0 = CC1 emission disabled. 1 = CC1 emission enabled.
23	CC2_EN	R/W	0	0 = CC2 emission disabled. 1 = CC2 emission enabled.
22	WS_PULSE	R/W	0	0 = Emit 50% WS. 1 = Emit single SCK cycle WS.

Table 9: Audio Output Port Registers *...Continued*

Bit	Symbol	Access	Value	Description
21:8	Unused		-	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read
7	UDR_INTEN	R/W	0	UNDERRUN Interrupt Enable. 0 = No interrupt for UNDERRUN condition 1 = Interrupt if an UNDERRUN error occurs.
6	HBE_INTEN	R/W	0	HBE Interrupt Enable: 0 = No interrupt for HBE condition 1 = Interrupt if a data bus bandwidth error occurs.
5	BUF2_INTEN	R/W	0	Buffer 2 Empty Interrupt Enable: 0 = No interrupt when DMA buffer is empty. 1 = Interrupt if DMA buffer 2 in memory is empty.
4	BUF1_INTEN	R/W	0	Buffer 1 Empty Interrupt Enable: 0 = No interrupt when DMA buffer 1 is empty. 1 = Interrupt if DMA buffer 1 in memory is empty.
3	ACK_UDR	R/W	0	Write a 1 to clear the UNDERRUN flag and remove any pending UNDERRUN interrupt request. ACK_UDR always reads 0.
2	ACK_HBE	R/W	0	Write a 1 to clear the HBE flag and remove any pending HBE interrupt request. ACK_HBE always reads as 0.
1	ACK2	R/W	0	Write a 1 to clear the BUF2_EMPTYflag and remove any pending BUF2_EMPTY interrupt request. AO_BASE2 is then used to fetch buffer1 data from memory. ACK2 always reads 0.
0	ACK1	R/W	0	Write a 1 to clear the BUF1_EMPTY flag and remove any pending BUF1_EMPTY interrupt request. AO_BASE1 is then used to fetch buffer1 data from memory. ACK1 always reads 0.
Offset 0x11 0008	***AO_SERIAL—DTL Clock Domain***			
31	SER_MASTER	R/W	0	0 = The D/A subsystem is the timing master over the Audio Out serial interface. SCK and WS act as inputs. 1 = AO is the timing master over serial interface. SCK and WS act as outputs. This mode is required for 4, 6 or 8 channel operation. The SER_MASTER bit should only be changed while Audio Out is disabled i.e.,TRANS_ENABLE = 0.
30	DATAMODE	R/W	0	0 = Data is transmitted MSB first. 1 = Data is transmitted LSB first.
29	CLOCK_EDGE	R/W	0	0 = The parallel-to-serial converter samples WS on positive edges of SCK and outputs data on the negative edge of SCK. 1 = The parallel-to-serial converter samples WS on negative edges of SCK and outputs data on positive edges of SCK.
28:19	Unused		-	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
18:17	NR_CHAN	R/W	00	00 = Only SD[0] is active. 01 = SD[0] and SD[1] are active. 10 = SD[0], SD[1], and SD[2] are active. 11 = SD[0], SD[1], SD[2] and SD[3] are active. Each SD output receives either 1 or 2 channels depending on TRANS_MODE. Non-active channels receive 0 value samples. In mono modes, each channel of a SD output receives identical left and right samples.

Table 9: Audio Output Port Registers *...Continued*

Bit	Symbol	Access	Value	Description
16:8	WSDIV	R/W	0x0	Sets the divider used to derive WS from SCK. Set to 0...511 for a serial frame length of 1...512. Note the frame size limitations discussed in Section 23.4.6.1 on page 23-16.
7:0	SCKDIV	R/W	0x0	Sets the divider used to derive SCK from OSCLK. Set to 0...255 for division by 1...256.
Offset 0x11 000C	***AO_FRAMING—DMA Clock Domain***			
31	POLARITY	R/W	0	0 = Serial frame starts with a WS negedge. 1 = Serial frame starts with a WS posedge. This bit should not be changed during operation of Audio Out i.e., only update this bit when TRANS_ENABLE = 0.
30	SSPOS4	R/W	0	Start/Stop bit position MSB. Note that SSPOS is a 5-bit field, while bit SSPOS4 is non-adjacent for backwards compatibility in 16-bit/ sample modes. Program this field along with AO_FRAMING[3:0].
29:22	Unused		-	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
21:13	LEFTPOS	R/W	0x0	Defines the bit position within a serial frame where the first data bit of the left channel is placed. The first bit of a serial frame is bit 0.
12:4	RIGHTPOS	R/W	0x0	Defines the bit position within a serial frame where the first data bit of the right channel is placed.
3:0	SSPOS	R/W	0x0	Start/Stop bit position. Note that SSPOS is a 5-bit field, while bit SSPOS4 is non-adjacent for backwards compatibility in 16-bit/ sample modes. Program this field along with AO_FRAMING[30]. If DATAMODE = MSB first, transmission starts with the MSB of the sample i.e., bit 15 for 16-bit/sample modes or bit 31 for 32-bit/ sample modes. SSPOS determines the bit index (0..31) in the parallel input word of the last transmitted data bit. If DATAMODE = LSB first, SSPOS determines the bit index (0..31) in the parallel word of the first transmitted data bit. Bits SSPOS up to and including the MSB are transmitted i.e., up to bit 15 in 16-bit/ sample mode and bit 31 in 32-bit/sample mode.
Offset 0x11 0010	***Reserved—DTL Clock Domain***			
31:0	Reserved		-	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
Offset 0x11 0014	***AO_BASE1—DTL Clock Domain***			
31:6	BASE1	R/W	0x0	Base Address of DMA buffer1 in memory - must be a 64-byte aligned address in local memory. If changed it must be set before ACK1.
5:0	Unused		-	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
Offset 0x11 0018	***AO_BASE2—DTL Clock Domain***			
31:6	BASE2	R/W	0x0	Base Address of DMA buffer2 in memory - must be a 64-byte aligned address in local memory. If changed it must be set before ACK2.
5:0	Unused		-	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.

Table 9: Audio Output Port Registers *...Continued*

Bit	Symbol	Access	Value	Description
Offset 0x11 001C	***AO_SIZE—DMA Clock Domain***			
31:6	SIZE	R/W	0	DMA buffer size in samples. The number of mono samples or stereo sample pairs is read from a DMA buffer in memory before switching to the other DMA buffer in memory. Buffer size in bytes is as follows: 16 bps, mono: 2 * SIZE 32 bps, mono: 4 * SIZE 16 bps, stereo: 4 * SIZE 32 bps, stereo: 8 * SIZE
5:0	Unused		-	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
Offset 0x11 0020	***AO_CC—DMA Clock Domain***			
31:16	CC1	R/W	0x0	The 16-bit value of CC1 is shifted into each emitted serial frame starting at bit position CC1_POS, as long as CC1_EN is asserted.
15:0	CC2	R/W	0x0	The 16-bit value of CC2 is shifted into each emitted serial frame starting at bit position CC2_POS, as long as CC2_EN is asserted.
Offset 0x11 0024	***AO_CFC—DMA Clock Domain***			
31:18	Unused		-	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
17:10	CC1_POS	R/W	0x0	Defines the bit position within a serial frame where the first data bit of CC1 is placed.
9:0	CC2_POS	R/W	0x0	Defines the bit position within a serial frame where the first data bit of CC2 is placed.
Offset 0x11 0028—0FF0	***Reserved—DTL Clock Domain***			
31:0	Reserved		-	A read returns 0xDEADABBA.
Offset 0x11 0FF4	***AO_PWR_DWN—DTL Clock Domain***			
31:1	Unused		-	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
0	PWR_DWN	R/W	0	The bit is used to provide power control status for system software block power management.
Offset 0x11 0FFC	***AO_MODULE_ID—DTL Clock Domain***			
31:16	ID	R	0x0120	Module ID. This field identifies the block as type Audio Out.
15:12	MAJ_REV	R	0	Major Revision ID. This field is incremented by 1 when changes introduced in the block result in software incompatibility with the previous version of the block. First version default = 0.
11:8	MIN_REV	R	0x2	Minor Revision ID. This field is incremented by 1 when changes introduced in the block result in software *compatibility* with the previous version of the block. First version default = 0.
7:0	APERTURE	R	0	Aperture size. Identifies the MMIO aperture size in units of 4 KB for the AO block. AO has an MMIO aperture size of 4 KB. Aperture = 0: 4 KB.

Chapter 16: Audio Input

PNX15xx Series Data Book – Volume 1 of 1

Rev. 2 — 1 December 2004 Product data sheet

1. Introduction

The Audio Input block can have up to four audio input ports. Each audio input port supports single or dual-channel sources. Hence the Audio In block can support up to 8 channels of audio input (4 stereo channels).

The Audio In module provides a DMA-driven serial interface to an off-chip stereo A/D converter, I^2S subsystem or other serial data source. Audio In provides all signals needed to connect to high quality, low cost oversampling A/D converters. The Audio In module and external A/D converter (or I^2S subsystem) together are capable of generating a programmable sample clock by dividing a precise oversampling clock, which is an input to this block. It is assumed that a programmable clock generator for a precise oversampling A/D system clock is present elsewhere in the chip.

1.1 Features

- Four channels of audio input per port
- 16 or 32-bit samples per channel
- Programmable 1 Hz to 100 kHz sampling rate
  (Note: This is a practical range. The actual sample rate is application dependent.)
- Internal or external sampling clock source
- Audio In autonomously writes sampled audio data to memory using buffering (DMA)
- 16-bit and 32-bit mono and stereo PC standard memory data formats
- Raw mode where the bits from all the active inputs are sampled by bit clock along with the frame sync signal (WS) and packed into one 8 bit byte in memory for software to tear apart.
- Little or big-endian memory formats.

Remark: AC-97 codecs are not supported.

2. Functional Description

The Audio In module has four major subsystems: a programmable sample clock generator, a serial-to-parallel converter, a DTL initiator interface that initiates transfer of parallel data to a DTL-to-memory bus adapter and a MMIO type low latency DTL target interface for MMIO configuration registers.

The sampling clock can be used as either master or slave to the external A/D device. The sampling clock synchronizes the serial-to-parallel converter with the source data stream. The samples enter the serial-to-parallel converter, which reformats the data for the initiator. The initiator streams the parallel data in to the DTL-to-memory bus adapter. All the buffering of data is done in this adapter. The adapter also acts as the DMA engine and bursts data to memory using the address provided by the initiator. Since buffering of data is heavily dependent on system level latency issues, it is best done in the adapter. Hence the buffer is not present in this block and instead will be in the adapter and sized according to system requirements.

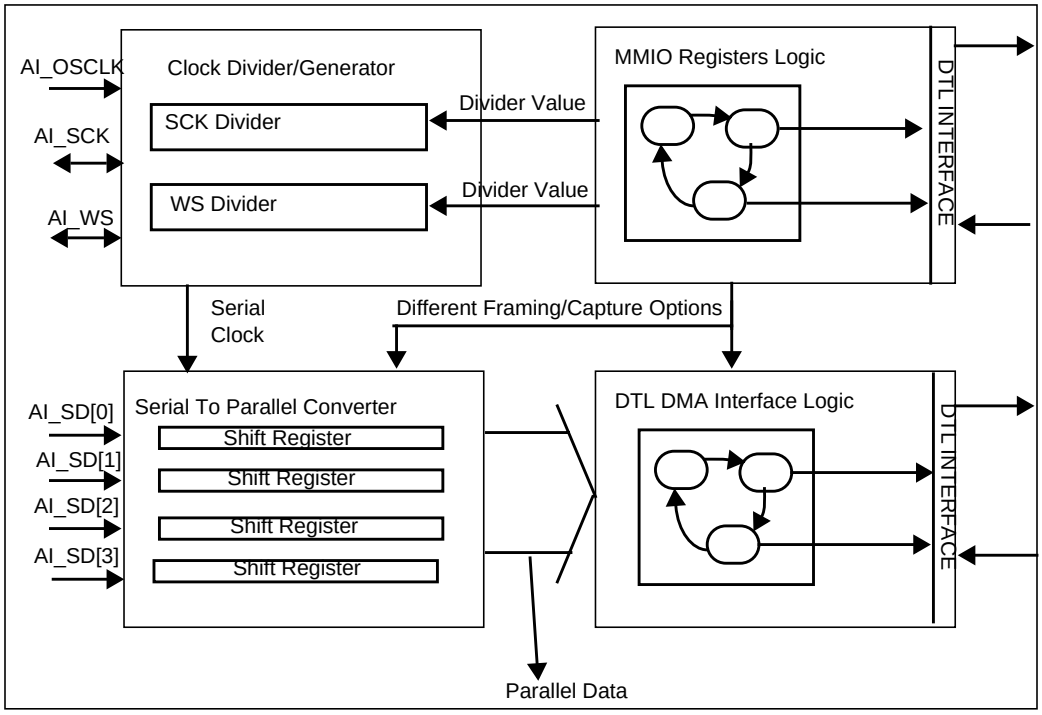


Figure 1: Audio In Block Diagram

2.1 Chip Level External Interface

Table 1: Audio-In I2S Related Ports

Signal	Type	Description
AI_OSCLK	Input	Oversampling Clock. This can be programmed to emit any frequency up to 40 MHz with a resolution of better than 0.3 Hz. It is intended for use as the 256 f_s or 384 f_s oversampling clock by external A/D subsystem.It is also used by the Audio in block to generate AI_SCK when it is in master mode. This is generated from the clock block, outside the Audio In module. It is an output from the chip, but also an input to the Audio Input block.
AI_SCK	In/out	When Audio In is programmed as the serial-interface timing slave (power-up default), SCK is an input. SCK receives the serial bit clock from the external A/D subsystem. This clock is treated as fully asynchronous to the main chip level clock. When Audio In is programmed as the serial-interface timing master, SCK is an output. SCK drives the serial clock for the external A/D subsystem. The frequency is a programmable integral divide of the OSCLK frequency. SCK is limited to 30 MHz. The sample rate of valid samples embedded within the serial stream is limited to 100 kHz.
AI_SD[3:0]	Input	Serial Data from external A/D subsystem. Data on these pins are sampled on positive or negative edges of SCK as determined by the CLOCK_EDGE bit in the AI_SERIAL register.
AI_WS	In/out	When Audio In is programmed as the serial-interface timing slave (power-up default), WS acts as an input. WS is sampled on the same edge as selected for SD. When Audio In is programmed as the serial-interface timing master, WS acts as an output. It is asserted on the opposite edge of the SD sampling edge. WS is the word-select or frame-synchronization signal from/to the external A/D subsystem.

The Audio In chip level I2S related external interface has seven pins AI_OSCLK, AI_SCK, AI_WS and AI_SD[3:0]. These pins may be also referenced as OSCLK, SCK, WS and SD[3:0].

The AI_OSCLK is a precise, programmable clock output intended to serve as the master system clock for the external A/D subsystem. The AI_OSCLK is generated from the Clock module which is outside the Audio In block. Although conceptually the oversampling clock is an output at the chip level, at the Audio In block level, this is an input. Six other pins constitute a flexible serial input interface IP.

AI_SCK = Audio In Serial Clock
AI_WS = Audio In Word Select

0 = Left Channel
1 = Right Channel

AI_SD[3:0] = Audio In Serial Data

Using the Audio In MMIO registers, these pins can be configured to operate in a variety of serial interface framing modes, including but not limited to the following:

- Standard stereo I^2S (MSB first, 1-bit delay from WS, left and right data in a frame). (For further details on I^2S, refer to the "I^2S Bus Specification" dated June 5 1996, in the *Multimedia ICs Data Handbook IC22* by Philips Semiconductors, 1998.)
- LSB first with 1- to 32-bit data per channel
- Complex serial frames of up to 512 bits/frame with "valid sample" qualifier bit

- Raw sample mode where the serial data for each active serial channel is sampled at each sampling clock edge along with the AI_WS and transferred to memory as a byte.

2.2 General Operations

Software initiates capture by providing two equal size empty buffers and putting their base address and size in the BASE1, BASE2 and SIZE registers. Once two valid (local memory) buffers are assigned, capture can be enabled by writing a ‘1’ to CAP_ENABLE. The Audio In unit hardware will proceed to fill buffer 1 with input samples. Once buffer 1 fills up, BUF1_FULL is asserted, and capture continues without interruption in buffer 2. If BUF1_INTEN is enabled, a level triggered interrupt request is generated to the chip level interrupt controller.

Note that the buffers must be 64-byte aligned and must be a multiple of 64 samples in size (the six LSBits of AI_BASE1, AI_BASE2 and AI_SIZE are always zero).

Software is required to assign a new, empty buffer to BASE1 and perform an ACK1, before buffer 2 fills up. Capture continues in buffer 2, until it fills up. At that time, BUF2_FULL is asserted and capture continues in the new buffer 1, etc.

Upon receipt of an ACK, the Audio In hardware removes the related interrupt request line assertion at the next main clock edge.

In normal operation, the chip level system controller and Audio In hardware continuously exchange buffers without ever losing a sample. If the system controller fails to provide a new buffer in time, the OVERRUN error flag is raised. This flag is *not affected* by ACK1 or ACK2; it can only be cleared by an explicit write of logic ‘1’ to ACK_OVR.

Remark: Reserved bits in MMIO registers should be ignored when read and written as zeros. See Section 4. on page 16-15.

3. Operation

3.1 Clock Programming

Figure 2 illustrates the clocking capabilities of the Audio Input unit. Driving the system is a square wave Direct Digital Synthesizer (DDS). The DDS can be programmed to emit frequencies from approximately 1 Hz to 40 MHz with a resolution of better than 0.3 Hz. The DDS and its control registers reside in the Clocks module outside the Audio in Unit.

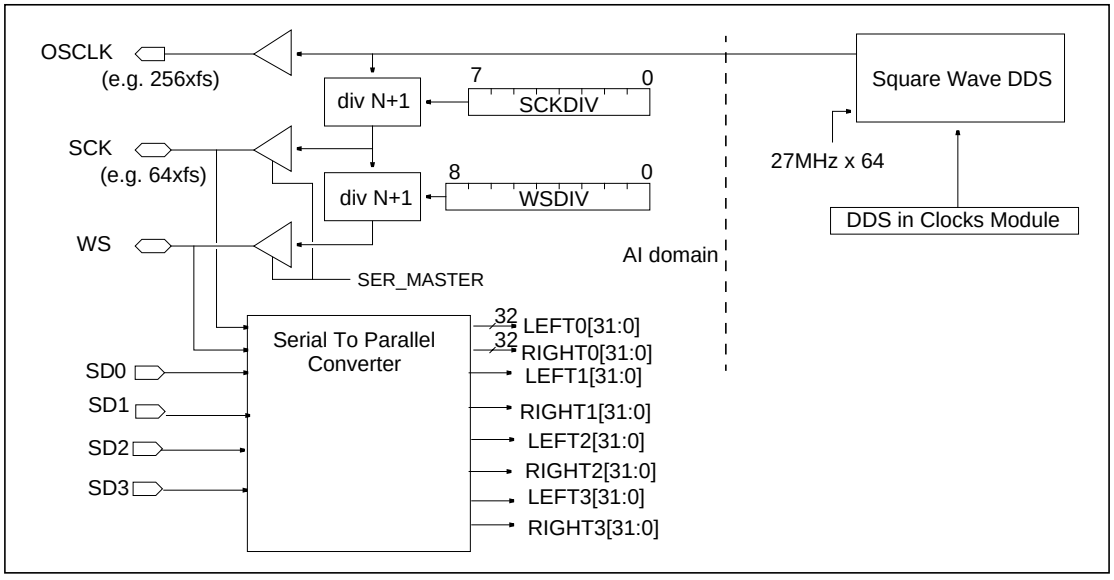


Figure 2: Audio In Clock System and I/O Interface

The output of the DDS is always sent on the OSCLK output pin. This output is intended to be used as the 256 f_S or 384 f_S system clock source for oversampling A/D converters.

Software may change the DDS frequency setting dynamically, so as to adjust the input sampling rate to track an application dependent master reference. Using the DDS function, a high quality, low-jitter OSCLK is generated.

3.1.1 Clock System Operation

SCK and WS can be configured as input or output, as determined by the SER_MASTER control field. As an output, SCK is a divided form of the OSCLK output frequency. The SCKDIV register value is used to divide down the OSCLK frequency. See Section 4. on page 16-15 for more details. Whether input or output, the SCK pin signal is used as the bit clock for serial-parallel conversion. The value of SCKDIV is determined by Equation 12:

$$f_{AISCK} = \frac{f_{AIOSCLK}}{SCKDIV + 1} \quad (12)$$

Remark: SCKDIV is in the range 0-255.

If set as output, WS can similarly be programmed using WSDIV to control the serial frame length from 1 to 512 bits. The number of bits per frame is equal to WSDIV + 1. Table 2 presents several sample rates with the appropriate SCKDIV necessary to achieve a bit clock of 64 f_s.

Table 2: Sample Rate Settings

f_s	OSCLK	SCKDIV	SCK
44.1 kHz	256 f_s	3	64 f_s
48.0 kHz	256 f_s	3	64 f_s
44.1 kHz	384 f_s	5	64 f_s
48.0 kHz	384 f_s	5	64 f_s

The preferred application of the clock system options is to use OSCLK as A/D master clock, and let the A/D converter be timing master over the serial interface (SER_MASTER = 0).

In case of an external codec for common audio input and audio output use, it may not be possible to independently control the A/D and D/A system clocks. It is recommended that the Audio Out clock system DDS is used to provide a single master A/D and D/A clock. The Audio Out or the D/A converter can be used as serial interface timing master, and Audio In is set to be slave to the serial frame determined by Audio Out (Audio In SER_MASTER = 0, SCK and WS externally wired to the corresponding Audio Out pins). In such systems, independent software control over A/D and D/A sampling rate is not possible, but component count is minimized.

3.2 Reset-Related Issues

The Audio In unit is reset by a chip level hardware reset or by writing logic ‘1’ to the AI_CTL.RESET register bit. As soon as the software reset bit is written, further MMIO commands are held off until the software reset has taken effect in the IP clock domain and the reset state restored. Upon RESET, capture is disabled (CAP_ENABLE = 0), and buffer1 is the active buffer (BUF1_ACTIVE = 1).

If the Audio In module was operating in clock master mode (SER_MASTER = 1) then a reset action prevents the SCK clock to be generated. This prevents the reset to complete. Therefore upon a software reset the AI module clock must be switched to the default 27 MHz (crystal input) in order to complete the reset.

Software should follow a series of steps to ensure that Software Reset happens correctly:

1. Check to see if there is a valid clock present on the Audio Input external clock input.

2. If there is no clock, then write to the Clocks block to switch the Audio Input clock to the 27 MHz oscillator.

3. Apply Software Reset and poll the RESET bit until it is cleared.

4. Program the Clocks block to switch the Audio Input external clock back to the external clock mode.

Clocks are required to be running during HW/SW reset because synchronous reset is used to initialize the logic. The unique feature with Audio is that unlike all other blocks in the system, the Audio blocks default to the external clock source on any reset. If the external clock does not exist when a HW reset is applied, then the logic is left uninitialized without any indication.

3.3 Register Programming Guidelines

Software needs to ensure that the cap_enable bit (bit 30, AI_CTL) is *programmed after* all the other registers have been programmed to ensure proper functionality.

Disabling and re-enabling capture

Here is a brief discussion on how the audio in block works if for some reason software needs to disable the capture and consequently re-enable it. The Audio Input module is continuously capturing and transferring data to memory through the adapter in the system. The adapter threshold should be suitably set to satisfy the system latency requirements. Once the adapter FIFO reaches the threshold, it will initiate a transfer to memory. This behavior will continue until the capture is disabled. Once the capture is disabled, the Audio In block will issue a FLUSH to the adapter so that it can flush its FIFO and hence all the pertinent data that would reach memory. However, it must be understood that disabling capture is not the same as applying software reset. Even though capture is disabled, all the internal DMA state machines have pointers pointing to addresses in memory corresponding to the transaction that was just completed. So if the software intends to re-enable capture from scratch with new pointers, there needs to be a software reset performed between disabling and re-enabling the capture, along with optional reprogramming of the registers. Failure to do a software reset will result in the Audio Input module behaving as though the previous transaction is continuing with all the previous pointers active.

3.4 Serial Data Framing

The Audio In unit can accept data in a wide variety of serial data framing conventions. Figure 3 illustrates the notion of a serial frame. If POLARITY = 1, CLOCK_EDGE = 0, and EARLYMODE=0, a frame is defined with respect to the positive transition of the WS signal as observed by a positive clock transition on SCK. (See Section 4..) Each data bit sampled on positive SCK transitions has a specific bit position—i.e., once the clock edge detects the WS transition, the next sample will be data bit position 0.

Each subsequent clock edge defines a new bit position. Other combinations of POLARITY and CLOCK_EDGE can be used to define a variety of serial frame bit position definitions. Further if the EARLYMODE = 1, then the first data bit (position 0) will be the data bit sampled in the same clock edge in which the WS signal transition is detected.(See Section 4..)

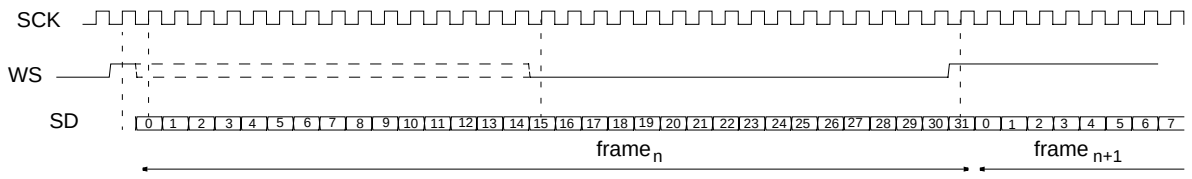


Figure 3: Audio In Serial Frame and Bit Position Definition (POLARITY = 1, CLOCK_EDGE = 0, EARLYMODE = 0)

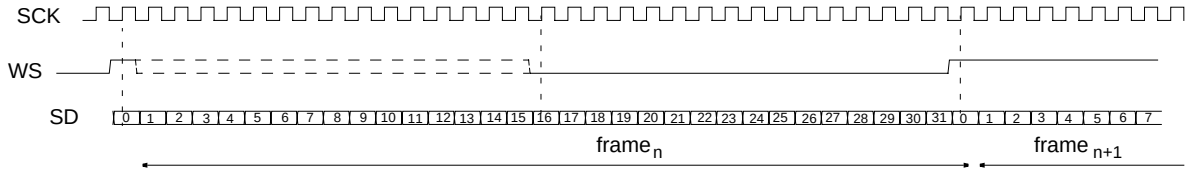


Figure 4: Audio In Serial Frame and Bit Position Definition (POLARITY = 1, CLOCK_EDGE = 0, EARLYMODE = 1)

The capturing of samples is governed by FRAMEMODE. If FRAMEMODE = 00, every serial frame results in one sample from the serial-parallel converter. A sample is defined as a left/right pair in stereo modes or a single left channel value in mono modes. If FRAMEMODE = 1y, the serial frame data bit in bit position VALIDPOS is examined. If it has value ‘y’, a sample is taken from the data stream (the valid bit is allowed to precede or follow the left or right channel data provided it is in the same serial frame as the data).

The left and right sample data can be in a LSB-first or MSB-first form at an arbitrary bit position and with an arbitrary length. (See Section 4..) In MSB-first mode, the serial-to-parallel converter assigns the value of the bit at LEFTPOS to LEFT[MSB]. Subsequent bits are assigned, in order, to decreasing bit positions in the LEFT data word, up to and including LEFT[SSPOS]. Bits LEFT[SSPOS–1:0] are cleared. Hence, in MSB-first mode, an arbitrary number of bits are captured. They are left-adjusted in the 16(32)-bit parallel output of the converter.

In LSB-first mode, the serial to parallel converter assigns the value of the bit at LEFTPOS to LEFT[SSPOS]. Subsequent bits are assigned, in order, to increasing bit positions in the LEFT data word, up to and including LEFT[MSB]. Bits LEFT[SSPOS–1:0] are cleared. Hence, in LSB-first mode, an arbitrary number of bits are captured. They are returned left-adjusted in the 16(32)-bit parallel output of the converter.

The following table shows the exact bit positions assigned for a data item ‘S’.

Table 3: Bit Positions Assigned for Each Data Item

Operating Mode	First Bit	Last Bit	Valid SSPOS Values
16 bit/sample, MSB-first	S[15]	S[SSPOS]	0..15
16 bit/sample, LSB-first	S[SSPOS]	S[15]	0..15
32 bit/sample, MSB-first	S[31]	S[SSPOS]	0..31
32 bit/sample, LSB-first	S[SSPOS]	S[31]	0..31

See Figure 5 and Table 4 for an example of how the Audio In module registers are set to collect 16-bit samples using the Philips SAA7366 I^2S 18-bit A/D converter. (See Section 4..) The setup assumes the SAA7366 acts as the serial master.

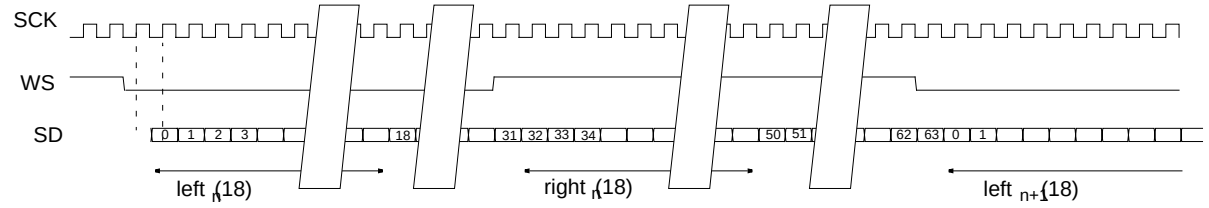


Figure 5: Serial Frame of the SAA7366 18-Bit I^2S A/D Converter (Format 2 SWS)

For example, if it were desired to use only the 12 MSBits of the A/D converter in Figure 5, use the settings of Table 4 with SSPOS set to four. This results in LEFT[15:4] being set with data bits 0..11 and LEFT[3:0] being set equal to zero. RIGHT[15:4] is set with data bits 32..43 and RIGHT[3:0] is set to zero.

Similarly, if it was desired to use only the 12 MSBits, but send 32 bit samples to memory, use the settings of Table 4 with SSPOS set to 20. This results in LEFT[31:20] being set with data bits 0...11 and LEFT[20:0] being set equal to zero. RIGHT[31:4] is set with data bits 32...43 and RIGHT[20:0] is set to zero.

Table 4: Example Setup For SAA7366

Field	Value	Explanation
SER_MASTER	0	SAA7366 is serial master.
SCKDIV	3	SCK set to OSCLK/4 (not needed since SER_MASTER = 0).
WSDIV	63	Serial frame length of 64 bits (not needed since SER_MASTER = 0)
POLARITY	0	Frame starts with negative WS.
FRAMEMODE	00	Take a sample of each serial frame.
VALIDPOS	n/a	Don’t care (Every frame is valid).
LEFTPOS	0	Bit position 0 is MSB of left channel and will go to LEFT[15].
RIGHTPOS	32	Bit position 32 is MSB of right channel and will go to RIGHT[15].
DATAMODE	0	MSB first
SSPOS	0	Stop with LEFT/RIGHT[0].
CLOCK_EDGE	0	Sample WS and SD on positive SCK edges for I^2S

3.5 Memory Data Formats

The Audio In unit autonomously writes samples to memory in mono and stereo 16 and 32-bit per sample formats, as shown in Figure 6. Successive samples are always stored at increasing memory address locations.

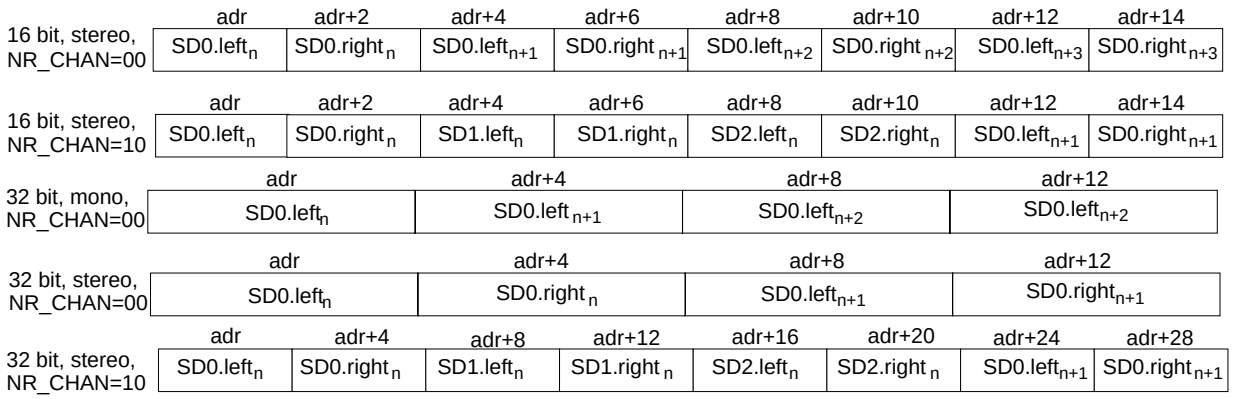


Figure 6: Audio In Memory DMA Formats

Table 5: Operating Modes and Memory Formats

NR_CHAN	MODE	Source of Successive Samples
00	mono (one channel)	SD0.left
00	stereo (one channel)	SD0.left, SD0.right
01	mono (two channels)	SD0.left, SD1.left
01	stereo (two channels)	SD0.left, SD0.right, SD1.left, SD1.right
10	mono (three channels)	SD0.left, SD1.left, SD2.left
10	stereo (three channels)	SD0.left, SD0.right, SD1.left, SD1.right, SD2.left, SD2.right
11	mono (four channels)	SD0.left, SD1.left, SD2.left, SD3.left
11	stereo (four channels)	SD0.left, SD0.right, SD1.left, SD1.right, SD2.left, SD2.right, SD3.left, SD3.right.

3.5.1 Endian Control

The following table illustrates exactly how the Audio Inout block writes data in memory (byte) locations precisely after the correct endian swapping done at the adapter depending on the polarity of the big-endian bit in the Global Registers Module.

Table 6: Endian Ordering of Audio Data in Main Memory

Operating Modes	m[adr]	m[adr+1]	m[adr+2]	m[adr+3]	m[adr+4]	m[adr+5]	m[adr+6]	m[adr+7]
16-bit mono - little endian	left$_n$[7:0]	left$_n$[15:8]	left$_{n+1}$[7:0]	left$_{n+1}$[15:8]	left$_{n+2}$[7:0]	left$_{n+2}$[15:8]	left$_{n+3}$[7:0]	left$_{n+3}$[15:8]
16-bit mono - big endian	left$_n$[15:8]	left$_n$[7:0]	left$_{n+1}$[15:8]	left$_{n+1}$[7:0]	left$_{n+2}$[15:8]	left$_{n+2}$[7:0]	left$_{n+3}$[15:8]	left$_{n+3}$[7:0]
16-bit stereo - little endian	left$_n$[7:0]	left$_n$[15:8]	right$_n$[7:0]	right$_n$[15:8]	left$_{n+1}$[7:0]	left$_{n+1}$[15:8]	right$_{n+1}$[7:0]	right$_{n+1}$[15:8]

Table 6: Endian Ordering of Audio Data in Main Memory *…Continued*

Operating Modes	m[adr]	m[adr+1]	m[adr+2]	m[adr+3]	m[adr+4]	m[adr+5]	m[adr+6]	m[adr+7]
16-bit stereo - big endian	$left_n$[15:8]	$left_n$[7:0]	$right_n$[15:8]	$right_n$[7:0]	$left_{n+1}$[15:8]	$left_{n+1}$[7:0]	$right_{n+1}$[15:8]	$right_{n+1}$[7:0]
32-bit mono - little endian	$left_n$[7:0]	$left_n$[15:8]	$left_n$[23:16]	$left_n$[31:24]	$left_{n+1}$[7:0]	$left_{n+1}$[15:8]	$left_{n+1}$[23:16]	$left_{n+1}$[31:24]
32-bit mono - big endian	$left_n$[31:24]	$left_n$[23:16]	$left_n$[15:8]	$left_n$[7:0]	$left_{n+1}$[31:24]	$left_{n+1}$[23:16]	$left_{n+1}$[15:8]	$left_{n+1}$[7:0]
32-bit stereo - little endian	$left_n$[7:0]	$left_n$[15:8]	$left_n$[23:16]	$left_n$[31:24]	$right_n$[7:0]	$right_n$[15:8]	$right_n$[23:16]	$right_n$[31:24]
32-bit stereo - big endian	$left_n$[31:24]	$left_n$[23:16]	$left_n$[15:8]	$left_n$[7:0]	$right_n$[31:24]	$right_n$[23:16]	$right_n$[15:8]	$right_n$[7:0]

3.6 Memory Buffers and Capture

The Audio Input unit hardware implements a double buffering scheme to ensure that no samples are lost, even if the chip level controller is highly loaded and slow to respond to interrupts. The software assigns buffers by writing a base address and size to the MMIO control fields (see Section 4. on page 16-15).

In 16-bit capture modes, the sixteen MSBits of the serial-to-parallel converter output data are written to memory. In 32-bit capture modes, all bits of the parallel data are written to memory. If SIGN_CONVERT is set to one, the MSB of the data is inverted, which is equivalent to translating from two's complement to offset binary representation. This allows the use of an external two's complement 16-32 bit A/D converter to generate 16-32 bit unsigned samples.

Remark: The Audio In hardware does *not* generate A-law or μ-law data formats. If such formats are desired, additional processing is necessary, via software, to convert from 16-bit linear data to A-law or μ-law data.

3.7 Data Bus Latency and HBE

Audio In uses a 128-byte buffer to capture the audio input samples. When 64-bytes of space is full, the next incoming samples are continuously stored in the remaining 64-byte space while the adapter issues DMA request for the first 64 bytes of data. Under normal operation, the first 64-bytes worth data gets written to memory while the second 64-bytes worth data is being filled up. This normal operation will be maintained as long as the data bus arbiter is set to guarantee a latency for Audio In that matches the incoming audio samples rate.

Given a sample rate f_S, and an associated sample interval T (in nSec), the bus latency should be at most 16*T nSec in the case of 16 bit samples and 8T nSec in the case of 32 bit samples, for 2 audio channels. Similarly for 4 channels (8T, 4T), 6 channels (16T/3, 8T/3) and 8 channels (4T, 2T) respectively. If this max latency is not satisfied, the HBE (Bandwidth Error) condition will result. This error flag gets set when the 128-byte buffer is full and a new sample arrives for that buffer. Thus HBE error condition occurs when the adapter is unable to transfer data from its FIFO to memory in time. Hence the adapter FIFO gets full and is not able to accommodate valid data coming from the Audio In block. This results in an HBE condition. Table 7

shows the required data bus arbitration latency requirements for a number of common operating modes, for 2 channels. The right column in Table 7 shows the nature of the resulting 64-byte burst data bus requests.

In the Raw Mode however, the sampling is much faster. One eight bit byte is sampled every SCK. Hence one 32 bit word (4 bytes) are transferred every four clocks. So for example if the sample clock SCK is about 25 MHz, then the bandwidth requirement would be 40 MBytes per second. Obviously this requirement is much higher than in the usual serial mode for this block.

Table 7: Audio In Data Bus Arbiter Latency Requirement Examples — 16-Bit Data Examples

CapMode 2 Channels	f_s (kHz)	T (nS)	Max Arbiter Latency (16*T) (uSec)	Access Pattern
Stereo 2x16 bits/sample	44.1	22,676	362.816	1 64-byte request minimally every 362.816 uS
Stereo 2x16 bits/sample	48.0	20,833	333.328	1 64-byte request minimally every 333.328 uS
Stereo 2x16 bits/sample	96.0	10,417	166.672	1 64-byte request minimally every 166.672 uS

Table 8: Audio In Data Bus Arbiter Latency Requirement Examples — 32-Bit Data Examples

CapMode 2 Channels	f_s (kHz)	T (nS)	Max Arbiter Latency (8*T) (uSec)	Access Pattern
Stereo 2x32 bits/sample	44.1	22,676	181.408	1 64-byte request minimally every 181.408uS
Stereo 2x32 bits/sample	48.0	20,833	166.66	1 64-byte request minimally every 166.66 uS
Stereo 2x32 bits/sample	96.0	10,417	83.34	1 64-byte request minimally every 83.34 uS

3.8 Error Behavior

If either an OVERRUN or HBE error occurs, input sampling is temporarily halted and incoming samples will be lost. In the case of OVERRUN, sampling resumes as soon as the control software makes one or more new buffers available through an ACK1 or ACK2 operation. In the case of HBE, sampling will resume as soon as the data in the FIFO can be written to memory. HBE and OVERRUN are ‘sticky’ error flags meaning they will remain set until an explicit software write of logic ‘1’ to ACK_HBE or ACK_OVR is performed. See Section 4. on page 16-15.

3.9 Interrupts

The AI_STATUS register provides all sources of Audio In generated interrupt: BUF1_FULL, BUF2_FULL, HBE and OVERRUN. All interrupts sourced by Audio In to the chip level interrupt controller are level triggered. An interrupt will be generated from Audio In only if the corresponding interrupt enable bit is set in the AI_CTL register. For example, to assert an interrupt to the system upon the occurrence of a bandwidth error (HBE asserted), set the HBE_INTEN bit to logic ‘1’. See Section 4..

Interrupt status bits within AI_STATUS are persistent, meaning that once the interrupt is triggered, it will remain asserted until cleared by setting the corresponding ACK bit in AI_CTL. Using the above example, assuming the HBE interrupt is asserted, write a logic ‘1’ to ACK_HBE to clear the AI_STATUS.HBE bit and deactivate the interrupt to the system.

3.10 Timestamp Events

Audio In exports event signals associated with audio capture to the central timestamp/timer function on-chip. The central timestamp/timer function can be used to count the number of occurrences of each event or timestamp the occurrence of the event or both. The event will be a positive edge pulse with the duration of the event to be greater than or equal to 160ns. The specific event exported is:

- The ai_tstamp_event event will occur when the last sample for the current DMA buffer reaches the internal buffer. The internal buffer referred to here is present in the adapter and not inside the Audio In block. One precise event will occur for each of the two DMA buffers.

The occurrence of this event represents a precise, periodic time interval which can be used by system software for audio/video synchronization.

3.11 Diagnostic Mode

This mode can be used during the diagnostic phase of system testing to verify the correct operation of both Audio In and Audio Out units. This loopback mode internally feeds the I2S Audio Out to the I2S Audio In. This test can be used to check the data flow from the Audio Out buffer to the Audio In buffer.

Audio In Operation

The Audio (I2S) Input Ports Registers (Section 4. on page 16-15) describe the function of the control and status fields of the AI unit. To ensure compatibility with future devices, undefined bits in MMIO registers should be ignored when read and written as 0s.

The AI unit is reset by a PNX15xx Series hardware reset, or by writing 0x80000000 to the AI_CTL register. Upon RESET, capture is disabled (CAP_ENABLE = 0), and buffer1 is the active buffer (BUF1_ACTIVE = 1). The CPU initiates capture by providing two equal size empty buffers and putting their base address and size in the BASEn and SIZE registers. Once two valid (local memory) buffers are assigned, capture can be enabled by writing a ‘1’ to CAP_ENABLE. The AI unit now proceeds to fill buffer 1 with input samples. Once buffer 1 fills up, BUF1_FULL is asserted, and capture continues without interruption in buffer 2. If BUF1_INTEN is enabled, a SOURCE 11 interrupt request is generated.

Note that the buffers must be 64-byte aligned and multiple of 64 samples in size (the six LSBits of AI_BASE1, AI_BASE2 and AI_SIZE are always ‘0’).

The CPU is required to assign a new, empty buffer to BASE1 and perform a ACK1 before buffer 2 fills up. Capture continues in buffer 2 until it fills up. At that time, BUF2_FULL is asserted and capture continues in the new buffer 1.

Upon receipt of an ACK, the AI hardware removes the related interrupt request line assertion at the next CPU clock edge. The AI interrupt should always be operated in level-sensitive mode since AI can signal multiple conditions that each need independent ACKs over the single internal SOURCE 11 request line.

In normal operation, the CPU and AI hardware continuously exchange buffers without ever losing a sample. If the CPU fails to provide a new buffer in time, the OVERRUN error flag is raised. The flag is not affected by ACK1 or ACK2; it can only be cleared by an explicit ACK-OVR.

3.12 Software Reset

Bit 31 of the AI_CTL register is the software reset bit for the Audio Input module.

The purpose of SW reset register bit is to cleanly abort DMA traffic, reset the Audio Input logic and reset all MMIO registers. SW reset must not hang the MMIO bus interface (i.e. MMIO bus state machine is not reset).

Although the IP clock should be running (i.e. not gated off to save power) when the SW reset bit is written, the system will not hang if a SW reset is executed when the clocks are off (this means in master mode that the clock needs to be switched to the 27 MHz crystal clock since SER_MASTER is also reset and therefore the clock is removed.

Remark: that SW reset makes no attempt to stop DMA data transfer in a precise manner.

The following sequence takes place following a software reset:

1. CPU sets the SW reset bit via an MMIO interface write and then polls that bit waiting for it to be cleared indicating reset completion.
2. Although this MMIO interface write completes immediately, accesses to all other registers are disabled until a round trip handshake with the Audio Input clock domain indicates that all SW reset action is complete. If a read to a register other than the SW reset register occurs while the interface is disabled then the error condition and 0xDEADABA data are immediately returned. If a read to the SW reset register occurs then a 1 is returned immediately with no error condition to indicate that the SW reset is still in progress. A toggle style handshake between the bus and Audio Input clock domains is already built into the new MMIO interface register for SW reset. The Audio Input completes the following actions before the handshake completion signal is asserted:
    a) Disable the streaming interface
    b) Abort DMA
    c) Reset IP logic
    d) Reset MMIO registers in both Audio Input and bus clock domains
3. The SW reset handshake is asserted to indicate that the above actions in both clock domains are complete. This causes the SW reset bit to be de-asserted and the MMIO interface to be enabled.

3.13 Raw Mode

Apart from the usual I^2S mode and the early mode, capture can also be enabled in the raw mode. At every sample clock (SCK) the data bit(s) from each active channel is capture along with the WS. This information is then formed as a byte. After four such bytes are formed, the resulting 32-bit data is transferred to memory. Hence every sample clock results in a byte of data for software to tear apart and manipulate.

The following table shows how the data bits, and the WS are sampled with respect to the different channels and formed in to a byte that gets transferred to memory.

Table 9: Raw Mode Format of Input Data and Word Select

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit2	Bit 1	Bit 0
0	0	0	SD[3]	SD[2]	SD[1]	SD[0]	WS

4. Register Descriptions

The register descriptions for the Audio In block are given below. The base address for the Audio In registers begins at offset 0x11 1000.

Table 10: Register Summary

Offset	Name	Description
0x11 1000	AI_STATUS	Provides status of Audio In components/situations.
0x11 1004	AI_CTL	Control register to configure Audio In options
0x11 1008	AI_SERIAL	Control register to configure Audio In serial timing and data options
0x11 100C	AI_FRAMING	Control register to configure data framing format
0x11 1010	Reserved	
0x11 1014	AI_BASE1	Base address of buffer 1
0x11 1018	AI_BASE2	Base address of buffer 2
0x11 101C	AI_SIZE	The DMA Buffer size in samples
0x11 1020—1FF0	Reserved	
0x11 1FF4	AI_PWR_DWN	Powerdown function. Implementation details not decided yet.
0x11 1FFC	AI_MODULE_ID	Module ID number, including major and minor revision levels

4.1 Register Table

Table 11: Audio (I^2S) Input Ports Registers

Bit	Symbol	Access	Value	Description
Note: The clock frequency emitted by the AI_OSCLK output is set in registers that control the Clock block in the chip.				
Offset 0x11 1000	***AI_STATUS***			
31:5	Unused		-	
4	BUF1_ACTIVE	R	1	1 = Buffer will be used for the next incoming sample. 0 = Buffer 2 will receive the next sample.
3	OVERRUN	R	0	An OVERRUN error has occurred i.e., software failed to provide an empty buffer in time and 1 or more samples have been lost.
2	HBE	R	0	Bandwidth Error

Table 11: Audio (I^2S) Input Ports Registers *...Continued*

Bit	Symbol	Access	Value	Description
1	BUF2_FULL	R	0	1 = Buffer 2 is full. If BUF2_INTEN is also 1, an interrupt request is pending.
0	BUF1_FULL	R	0	1 = Buffer 1 is full. If BUF1_INTEN is also 1, an interrupt request is pending.
Offset 0x11 1004 AI_CTL				
31	RESET	R/W	0	The Audio In logic is reset by writing a 0x80000000 to AI_CTL. This bit is set during software reset and is cleared at the completion of software reset. Software can poll this bit and when it reads a 0, it knows that the reset is done.
30	CAP_ENABLE	R/W	0	Capture Enable flag: 0 = Audio In is inactive. 1 = Audio In captures samples and acts as DMA master to write samples to local memory.
29:28	CAP_MODE	R/W	00	00 = Mono (left ADC only), 32 bits/sample 01 = Stereo, 2 times 32 bits/sample 10 = Mono (left ADC only), 16 bits/sample 11 = Stereo, 2 times 16 bits/sample
27	SIGN_CONVERT	R/W	0	0 = Leave MSB unchanged. 1 = Invert MSB.
26	EARLYMODE	R/W	0	Setting this bit will enable the Audio Input port to capture data in a mode where the first data bit is driven on the same clock edge during which WS is driven. So in this mode the data is sampled one clock early compared to the standard I2S mode. 0 = Standard I2S mode. First data bit expected the next clock after WS has been sampled. 1 = Early mode. First data bit expected the same clock during which WS has been sampled.
25	DIAGMODE	R/W	0	0 = Normal operation 1 = Diagnostic mode
24	RAWMODE	R/W	0	0 = Normal I2S mono/stereo capture formats. 1 = Serial stream is captured in a raw mode. At every sample clock (SCK) the data bit(s) from each active channel is capture along with the WS. This information is then transferred to memory as a byte. Hence every sample clock results in a byte of data transferred to memory for software to tear apart and manipulate.
23:8	Unused		-	
7	OVR_INTEN	R/W	0	Overrun Interrupt Enable: 0 = No interrupt 1 = Interrupt if an overrun error occurs.
6	HBE_INTEN	R/W	0	HBE Interrupt Enable: 0 = No interrupt 1 = Interrupt if a bandwidth error occurs.
5	BUF2_INTEN	R/W	0	Buffer 2 full interrupt Enable: 0 = No interrupt 1 = Interrupt if buffer 2 full.
4	BUF1_INTEN	R/W	0	Buffer 1 full Interrupt Enable: 0 = No interrupt 1 = Interrupt if buffer 1 full.

Table 11: Audio (I^2S) Input Ports Registers *...Continued*

Bit	Symbol	Access	Value	Description
3	ACK_OVR	R/W	0	Write a 1 to clear the OVERRUN flag and remove any pending OVERRUN interrupt request. This bit always reads as 0.
2	ACK_HBE	R/W	0	Write a 1 to clear the HBE flag and remove any pending HBE interrupt request. This bit always reads as 0.
1	ACK2	R/W	0	Write a 1 to clear the BUF2_FULL flag and remove any pending BUF2_FULL interrupt request. AI_BASE2 must be valid before setting ACK2. This bit always reads as 0.
0	ACK1	R/W	0	Write a 1 to clear the BUF1_FULL flag and remove any pending BUF1_FULL interrupt request. AI_BASE1 must be valid before setting ACK2.This bit always reads as 0.
Offset 0x11 1008	***AI_SERIAL***			
31	SER_MASTER	R/W	0	Sets clock ratios and internal/external clock generation. 0 = The A/D converter is the timing master over the serial interface. AI_SCK and AI_WS pins are set to be input. 1 = Audio In serial interface is the timing master over the external A/D. The AI_SCK and AI_WS pins are set to be outputs.
30	DATAMODE	R/W	0	0 = MSB first 1 = LSB first
29:28	FRAMEMODE	R/W	00	This mode governs capturing of samples. 00 = Accept a sample every serial frame. 01 = Unused, reserved 10 = Accept sample if valid bit = 0. 11 = Accept sample if valid bit = 1.
27	CLOCK_EDGE	R/W	0	0 = The SD and WS pins are sampled on positive edges of the SCK pin. If SER_MASTER = 1, WS is asserted on SCK negative edge. 1 = SD and WS are sampled on negative edges of SCK. As output, WS is asserted on SCK positive edge.
26:20	Unused		-	
19	SSPOS4	R/W	0	Start/Stop bit MSB. Note that SSPOS is actually a 5 bit field, and this is the MSB SSPOS[4]and is non-adjacent to the bits SSPO[3:0] due to software compatibility reasons. Program this field along with AI_FRAMING[3:0].
18:17	NR_CHAN	R/W	00	00 = Only SD[0] is active. 01 = SD[0] and [1] are active. 10 = SD[0], [1], and [2] are active. 11 = SD[0]..SD[3] are active. Each SD input receives either 1 or 2 channels depending on CAP_MODE. In mono modes, the samples are captured from the left channel.
16:8	WSDIV	R/W	0	Sets the divider used to derive AI_WS from AI_SCK. Set to 0..511 for a serial frame length of 1..512.
7:0	SCKDIV	R/W	0	Sets the divider used to derive AI_SCK from AI_OSCLK. Set to 0..255, for division by 1..256.
Offset 0x11 100C	***AI_FRAMING***			
31	POLARITY	R/W	0	Sets format of serial data stream. 0 = Serial frame starts on WS negative edge. 1 = Serial frame starts on WS positive edge.

Table 11: Audio (I^2S) Input Ports Registers *...Continued*

Bit	Symbol	Access	Value	Description
30:22	VALIDPOS	R/W	0	Defines bit position within a serial frame where the valid bit is found.
21:13	LEFTPOS	R/W	0	Defines bit position within a serial frame where the first data bit of the left channel is found.
12:4	RIGHTPOS	R/W	0	Defines bit position within a serial frame where the first data bit of the right channel is found.
3:0	SSPOS_3_0	R/W	0	Start/Stop bit position. If DATAMODE = MSB first, SSPOS determines the bit index (0..15) for 16 bit samples and (0...31) for 32 bit samples, in the parallel word of the *last* data bit. Bits 15 or 31 (MSB) up to and including SSPOS are taken in order from the serial frame data. All other bits are set to zero. If DATAMODE = LSB first, SSPOS determines the bit index (0..15) for 16 bit samples and (0...31) for 32 bit samples, in the parallel word of the first data bit. Bits SSPOS up to and including 15 or 31 (MSB) are taken in order from the serial frame data. All other bits are set to zero. Note: Refer also to AI_SERIAL[19]
Offset 0x11 1010	***Reserved***			
Offset 0x11 1014	***AI_BASE1***			
31:6	BASE1	R/W	0	Base Address of buffer 1 must be a 64-byte aligned address in local memory. If changed it must be set before ACK1.
5:0	Reserved			
Offset 0x11 1018	***AI_BASE2***			
31:6	BASE2	R/W	0	Base Address of buffer 1 must be a 64-byte aligned address in local memory. If changed it must be set before ACK2.
5:0	Reserved			
Offset 0x11 101C	***AI_SIZE***			
31:6	SIZE	R/W	0	Sets number of samples in buffers before switching to other buffers. In stereo modes, a pair of 16-bit or 32-bit data counts as 1 sample. In mono modes, a single value counts as a sample.Buffer size in bytes is as follows: 16 bps, mono: 2 * SIZE 32 bps, mono: 4 * SIZE 16 bps, stereo: 4 * SIZE 32 bps, stereo: 8 * SIZE During raw mode the sample size is always 8 bits. Hence Buffer size in bytes for the raw mode is SIZE.
5:0	Reserved			
Offset 0x11 1020—1FF0	***Reserved***			
Offset 0x11 1FF4	***AI_PWR_DWN***			
31:1	Unused		-	

Table 11: Audio (I^2S) Input Ports Registers *...Continued*

Bit	Symbol	Access	Value	Description
0	PWR_DWN	R/W	0	The bit is used to provide power control status for system software block power management. Implementation details yet to be worked out.
Offset 0x11 1FF8	***MODULE_ID_EXT***			
31:0	Unused	R/W	-	The audio in block does not use this register. This always reads 0.
Offset 0x11 1FFC	***AI_MODULE_ID***			
31:16	ID	R	0x010D	Module ID. This field identifies the block as type Audio In.
15:12	MAJ_REV	R	0x1	Major Revision ID. This field is incremented by 1 when changes introduced in the block result in software incompatibility with the previous version of the block. First version default = 0.
11:8	MIN_REV	R	0x1	Minor Revision ID. This field is incremented by 1 when changes introduced in the block result in software compatibility with the previous version of the block. First version default = 0.
7:0	APERTURE	R	0	Aperture size. Identifies the MMIO aperture size in units of 4 KB for the AI block. AI has an MMIO aperture size of 4 KB. Aperture = 0: 4 KB.

Chapter 17: SPDIF Output

PNX15xx Series Data Book – Volume 1 of 1

Rev. 2 — 1 December 2004 Product data sheet

1. Introduction

The SPDIF Output (SPDO) block generates a 1-bit high-speed serial data stream. The primary application is to generate SPDIF (Sony/Philips Digital Interface) data for use by external audio equipment.

1.1 Features

The SPDIF output has the following features:

- Fully compliant with IEC-60958, for both consumer and professional applications
- Supports two channel linear PCM audio, with 16-24 bit/sample
- Supports one or more Dolby Digital AC-3 6 channel data streams embedded as IEC-61937
- Supports one or more MPEG-1 or MPEG-2 audio streams embedded as per IEC-61937
- Allows arbitrary, programmable, sample rates from 1 Hz to 300 kHz
- SPDO can carry data with a sample rate independent of and asynchronous to the Audio Out sample rate
- SPDO hardware performs autonomous DMA of memory resident IEC-60958 sub-frames
- SPDO hardware performs parity generation and bi-phase mark encoding
- Software has full control over all data content, including User and Channel data

Remark: IEC-61937 is a generic specification for transmitting non PCM data with an IEC-60958 transport.It supports AC3, PTS, MPEG1 Layer3 (MP3). MPEG 2 BC, MPEG 2 AAC and others.

It is possible to use the SPDO signal as a general purpose high-speed data stream. Potential applications include use as a high-speed UART or high speed serial data channel. In this case, the features include:

- Up to 40 mbit/sec data rate
- Full software control over each bit cell transmitted
- LSB first or MSB first data format

2. Functional Description

2.1 Architecture

The SPDO module has two basic components: a DMA engine and an emitter. The emitter is clocked from the DDS (in the Clock module) and can be programmed to the desired sample rare. The emitter delivers the data stream to the SPDIF Output pin.

2.2 General Operations

Software initially gives SPDO two memory data buffers, then enables the SPDO block. As soon as the first memory buffer is drained, SPDO requests a new buffer from software while switching over the use the other memory buffer, and so on.

With the exception of the DDS operation, the SPDO block is generally software compatible with that of the PNX1300 Series.

3. Operation

3.1 Clock Programming

A programmable clock generated by the SPDO Direct Digital Synthesizer (DDS). Note that the DDS resides in the central Clocks module.

3.1.1 Sample Rate Programming

In SPDIF, the frame rate always equals f_s, the sample rate of embedded audio. This relation holds for PCM as well as for AC-3 and MPEG audio. Each frame consists of 128 Unit Intervals (UI's). The length of a UI is determined by the frequency setting of the SPDO Direct Digital Synthesizer (DDS) in the central clock module.

$$f_s = \frac{(f_{DDS})}{128} \quad (13)$$

The DDS can be programmed to emit on chip frequencies from approximately. 1 Hz to 80 MHz with a maximum jitter of less than 0.579 ns. Refer to Chapter 5 The Clock Module for details.

Table 1 shows settings for common sample rate and main clock values.

Table 1: SPDIF Out Sample Rates and Jitter

f_s (kHz)	UI (nSec)	jitter (nSec)
32.000	244.14	0.579
44.100	177.15	0.579
48.000	162.76	0.579
96.000	81.38	0.579

3.2 Register Programming Guidelines

Before enabling the SPDO block, software must assign two buffers with data to SPDO_BASE1, SPDO_BASE2 and a buffer size value (in bytes) to SPDO_SIZE. Each memory buffer size must be a multiple of 64 bytes, regardless of the operating mode.

The SPDO block is enabled by writing a ‘1’ to SPDO_CTL.TRANS_ENABLE. Once enabled, the first DMA buffer is sent out at the programmed sample rate. Once the first buffer is empty, BUF1_ACTIVE is negated and the BUF1_EMPTY flag in SPDO_STATUS is asserted. If BUF1_INTEN in SPDO_CTL is asserted, an interrupt to the chip level interrupt controller is generated.

The SPDO block continues by emitting the data in DMA buffer2. In normal operation, software assigns a new buffer 1 full of data to SPDO and signals this by writing a ‘1’ to ACK_BUF1. The SPDO block immediately negates the BUF1_EMPTY condition and the related interrupt request. Once buffer2 is empty, similar signalling occurs and the hardware switches back using buffer1. Transmission continues interrupted until the unit is disabled.

The SPDO module has two operating modes: SPDIF and transparent DMA mode.

IEC Mode

SPDIF driver software assembles SPDIF data in each memory data buffer. Each memory data buffer consists of groups of 32 bit words in memory. Each word describes the data to be transmitted for a single IEC-60958 sub-frame, including what type of preamble to include. Each sub-frame is transmitted in 64 clock intervals of the SPDO clock.

The SPDIF mode is also useful for providing a source input stream for an available SPDIF IN block.

Transparent DMA Mode

In transparent DMA mode, software prepares each data bit exactly as it is to be transmitted, in a series of 32 bit words in each memory data buffer. The 32 bit word is constructed according to the byte ordering rules of little or big-endian mode. Each 32-bit word is transmitted (LSB or MSB first) into 32 clock intervals of the DDS. The data is shifted out “as is,” without bi-phase mark encoding, parity generation or preamble insertion.

3.3 Data Formatting

3.3.1 IEC-60958 Serial Format

Figure 1 shows the serial format layout of an IEC-60958 block. A block starts with a special “B” preamble, and consists of 192 frames. The sample rate of all embedded audio data is equal to the frame rate. Each frame consists of 2 subframes. Subframe 1 always starts with an “M” preamble, except for subframe 1 in frame 0, which starts with a “B”. Subframe 2 always starts with a “W” preamble.

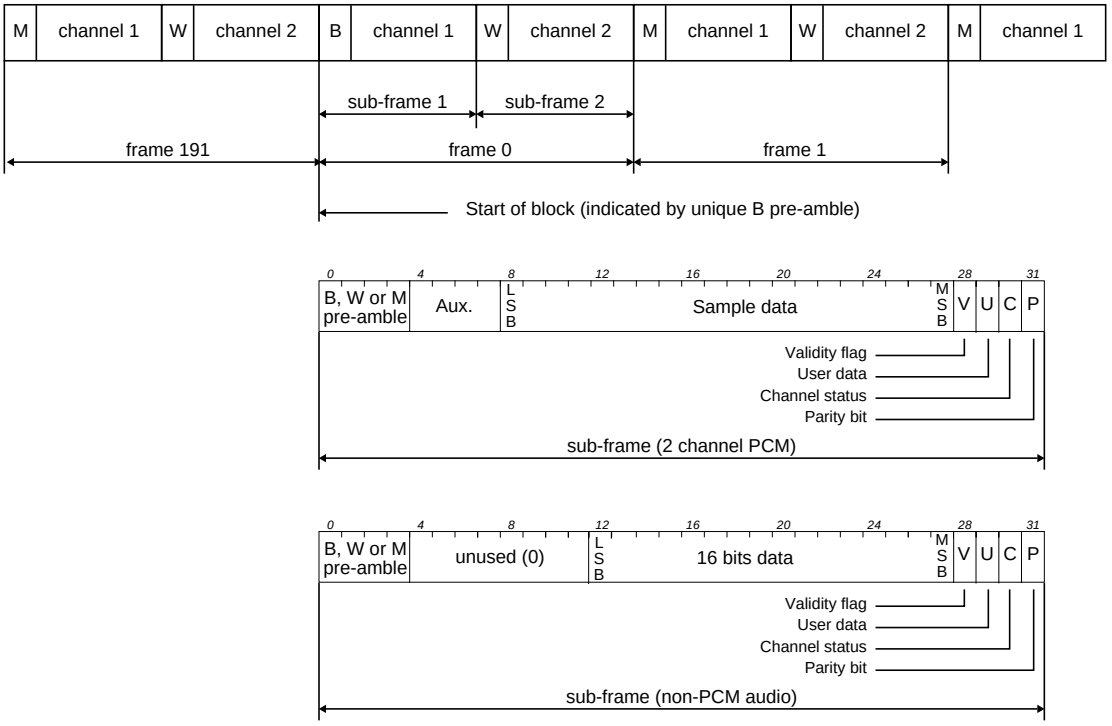


Figure 1: Serial Format of a IEC-60958 Block

When IEC-60958 data carries two-channel PCM data, one audio sample is transmitted in each sub-frame, “left” in sub-frame 1 and “right” in sub-frame 2. Each sample can be 16-24 bit, where the MSB is always aligned with bit slot 27 of the sub-frame. In case of more than 20 bits per sample, the Aux field is used for the 4 LSB bits.

When IEC-60958 data carries non-PCM audio, such as 1 or more streams of encoded AC-3 data and/or MPEG audio, each sub-frame carries 16 bits data. The data of successive frames adds up to a payload data-stream which carries its own burst-data.

This is described in the “Interface for non-PCM Encoded Audio Bitstreams Applying IEC958.” *Philips Consumer Electronics,* June 6 1997, IEC 100c/WG11 (preliminary for IEC-61937).

Programmers should refer to the IEC-60958 *Digital Audio Interface*, “Part 1:General; Part 2: Professional Applications; Part 3: Consumer Applications” for a precise description of the required values in each field for different types of consumer equipment.

The SPDO block hardware only generates B, W and M preambles as well as the P (Parity) bit. All other bits in the sub-frame are completely determined by software and copied “as is” from memory to output, subject only to bit-cell coding.

Software must construct valid IEC-60958 blocks by using the right sequence of 32-bit words as described in Section IEC-60958 Memory Data Format

IEC-60958 Bit Cell and Preamble

Each data bit in IEC-60958 is transmitted using bi-phase mark encoding. In bi-phase mark encoding, each data bit is transmitted as a cell consisting of two consecutive binary states. The first state of a cell is always inverted from the second state of the previous cell. The second state of a cell is identical to the first state if the data bit value is a “0”, and inverted if the data bit value is a “1”.

Preambles are coded as bi-phase mark violations, where the first state of a cell is not the inverse of the last state of the previous cell.

The duration of each state in a cell is called a UI (Unit Interval), so that each cell is 2 UIs long. In SPDO, the length of a UI is 1 SPDO clock cycle as determined by the settings of the DDS (see Section 3.1.1 Sample Rate Programming).

Figure 2 illustrates the transmission format of 8 bit data value “10011000”, as well as the transmission format of the three preambles. Note that each preamble always starts with a rising edge. This is made possible by the presence of the parity bit, which always guarantees an even number of ‘1’ bits in each subframe.

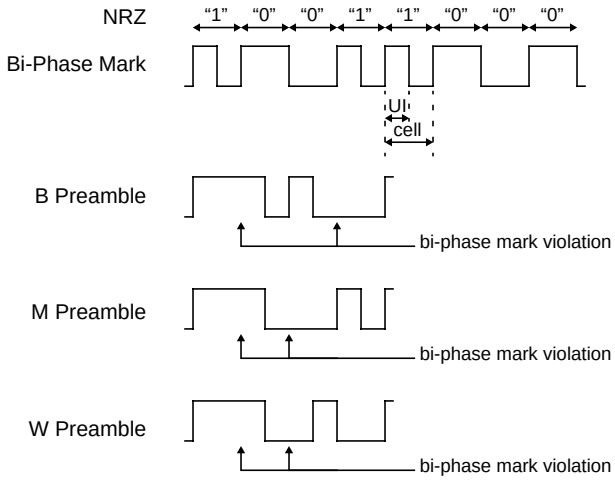


Figure 2: Bi-Phase Mark Data Transmission

IEC-60958 Parity

The parity bit, or P bit in Figure 1, is computed by the SPDIF Out hardware. The P bit value should be set such that bit cells 4 to 31 inclusive contain an even number of ones (and hence even number of zeroes). The P bit is bi-phase mark encoded using the same method as for all other bits.

IEC-60958 Memory Data Format

The system software must prepare a memory data structure that instructs the SPDIF block hardware to generate correct IEC-60958 blocks. This data structure consists of 32-bit words with the content described in Table 5.3:

Table 2: SPDIF Subframe Descriptor Word

Bits	Definition
31 (msb)	This bit must be a ‘0’ for future compatibility.
30..4	Data value for bits 4..30 of the subframe, exactly as they are to be transmitted. Hardware will perform the bi-phase mark encoding and Parity generation.
3..0 (lsb)	0000 - generate a B preamble 0001 - generate an M preamble 0010 - generate a W preamble 0011 .. 1111 reserved for future

The data structure for a block consists of 384 of these 32-bit descriptor words, one for each subframe of the block, with the correct B, M, W values. All data content, including the U, C and V flag are fully under control of the software that builds each block.

A DMA buffer handed to the hardware is required to be a multiple of 64 bytes in length. It can contain 1 or more complete blocks, or a block may straddle DMA buffer boundaries. The 64 byte length will result in DMA buffers that contain a multiple of 16 subframes.

3.3.2 Transparent Mode

When SPDO is set to operate in transparent mode, it takes all 32-bits of the memory data and shifts them out as is, without bi-phase mark encoding, parity generation or preamble insertion.

Two transparent modes are provided, as determined by the TRANS_MODE field in SPDO_CTL: LSB first and MSB first.

One bit of memory data is transmitted for each DDS clock.

The 32-bit memory word is constructed according to the same rules for byte ordering as in Section IEC-60958 Memory Data Format above.

3.4 Errors and Interrupts

3.4.1 DMA Error Conditions

Two types of errors can occur during DMA operation.

If the software fails to provide a new buffer of data in time, and both DMA buffers empty out, the SPDO hardware raises the UNDERRUN flag in SPDO_STATUS. Transmission switches over to the use of the next buffer, but the data transmitted is from the previously transmitted buffer. IF UDR_INTEN is asserted, an interrupt will be generated. The UNDERRUN flag is sticky - it will remain asserted until the software clears it by writing a '1' to ACK_UDR.

A lower level error can also occur when the limited size internal buffer empties out before it can be refilled across the data bus. This situation can arise only if insufficient bandwidth is allocated to SPDO from the bus arbiter. In this case, the Highway Bandwidth Error (HBE) error flag is raised.

3.4.2 HBE and Latency

If the arbiter is set up with an insufficient latency guarantee, a situation can arise that requested data will not arrive in time, (when a new output sample is due). In that case the HBE error is raised, and the last sample for each channel will be repeated until the new buffer is refreshed. The HBE condition is sticky, and can only be cleared by an explicit ACK_HBE. This condition indicates an incorrect setting of the arbiter.

The arbiter needs to guarantee that the maximum latency required by the SPDO block can always be met.

Given an output data rate f_S in samples/sec, 2 x 32 bits are required each sample interval. The arbiter should be set to have a latency so that the buffer is refilled before a sample interval expires. Refer to Table 3 for example latency requirements.

Table 3: SPDO Block Latency Requirements

f_S (kHz)	$1/f_S$ (nSec)	Max. latency ($9/f_S$) (uSec)
32.000	31250	281
44.100	22675	204
48.000	20833	187
96.000	10416	94

3.4.3 Interrupts

The SPDO block generates an interrupt if one of the following status bit flags, and its corresponding INTEN flag are set: BUF1_EMPTY, BUF2_EMPTY, HBE, UNDERRUN. See Offset 0x10 9000 SPDO_STATUS for details.

All these status flags are 'sticky', i.e. they are asserted by hardware when a certain condition occurs, and remain set until the interrupt handler explicitly clears them by writing a '1' to the corresponding ACK bit in SPDO_CTL. The SPDO hardware takes the flag away in the clock cycle after the ACK is received. This allows immediate return from interrupt once performing an ACK.

3.4.4 Timestamp Events

SPDO exports event signals associated with audio transmission to the central timestamp/timer function on-chip. The central timestamp/timer function can be used to count the number of occurrences of each event or timestamp the occurrence of the event or both. The event will be a positive edge pulse with the duration of the event to be greater than or equal to 200 ns. The specific event exported is as follows:

- BUF_DONE - signals when the last word of a memory buffer is being requested by the SPDO module. Note that this event is not dependent upon memory bus latency.

The occurrence of this event represents a precise, periodic time interval which can be used by system software for audio/video synchronization.

3.5 Endian Mode

The SPDIF descriptor is a 32-bit word memory data structure,

4. Signal Description

4.1 External Interface

Table 4: SPDIF Out External Signals

Signal	Type	Description
SPDO	O	SPDIF output. Self clocking interface carrying either two-channel PCM data with samples up to 24 bits, or encoded Dolby Digital (AC-3) or MPEG audio data for decoding by an external AC3 or MPEG capable audio amplifier.

An external circuit as shown in Figure 3 is required to provide an electrically isolated output and convert the 3.3 Volt output pin to a drive level of 0.5 V peak-peak into a 75 Ohm load, as required for consumer applications of IEC-60958.

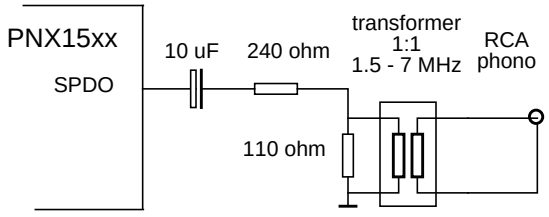


Figure 3: Suggested External SPDIF Output Interface Circuitry

5. Register Descriptions

The base address for the PNX15xx Series SPDIF Output Port module is 0x10 9000.

5.1 Register Summary

Table 5: SPDIF Output Module Register Summary

Offset	Name	Description
0x10 9000	SPDO_STATUS	SPDIF Out Status
0x10 9004	SPDO_CTL	SPDIF Out general control register
0x10 9008	Reserved	
0x10 900C	SPDO_BASE1	Base address of buffer1

Table 5: SPDIF Output Module Register Summary *…Continued*

Offset	Name	Description
0x10 9010	SPDO_BASE2	Base address of buffer2
0x10 9014	SPDO_SIZE	Size of the buffers
0x10 9018—9FF0	Reserved	
0x10 9FF4	SPDO_PWR_DWN	Powerdown
0x10 9FFC	SPDO_MODULE_ID	Module ID

Remark: The clock frequency emitted by the DDs output can be found in Chapter 5 The Clock Module.

5.2 Register Table

Table 6: SPDO Registers

Bit	Symbol	Access	Value	Description
Offset 0x10 9000	***SPDO_STATUS***			
31:5	Reserved			To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
0	BUF1_EMPTY	R	0	This flag gets set if DMA buffer 1 has been emptied by the SPDO hardware. The flag can be cleared only by a software write to ACK_BUF1.
1	BUF2_EMPTY	R	0	This flag gets set if DMA buffer 2 has been emptied by the SPDO hardware. The flag can be cleared only by a software write to ACK_BUF2.
2	HBE (bandwidth error)	R	0	Bandwidth Error. This flag gets set if the internal buffers in SPDO were emptied before new memory data was brought in. This flag can be cleared only by a software write to ACK_HBE.
3	UNDERRUN	R	0	This flag gets set if both DMA buffers were emptied before a new full buffer was assigned by software. The hardware has performed a normal buffer switch over and is emitting old data. It can only be cleared by software write to ACK_UDR.
4	BUF1_ACTIVE	R	1	This flag gets set if the hardware is currently emitting DMA buffer 1 data, and is negated when emitting DMA buffer 2 data.
Offset 0x10 9004	***SPDO_CTL***			
31	RESET	W	0	1 =Software Reset. Immediately resets the SPDO block. This should be used with extreme caution. Any ongoing transmission will be interrupted, and receivers may be left in a strange state.
30	TRANS_ENABLE	R/W	0	1 =Enables transmission as per the selected mode. 0 =Transmission Disabled. Stops any ongoing transmission after completing any actions related to the current data descriptor word.

Table 6: SPDO Registers *…Continued*

Bit	Symbol	Access	Value	Description
29:27	TRANS_MODE	R/W	000	Transmission mode. 000 =IEC-60958 mode. Hardware performs bi-phase mark encoding, preamble and parity generation, and transmits one IEC-60958 subframe for each data descriptor word. 010 =Transparent mode, LSB first. The 32 bit data descriptor words are transmitted as is, LSB first. 011 =Transparent mode, MSB first. The 32 bit data descriptor words are transmitted as is, MSB first. Other codes are reserved for future extensions. Note: The transmission mode should only be changed while transmission is disabled.
26:8	Reserved		-	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
7	UDR_INTEN	R/W	0	If UDR_INTEN = 1 and UNDERRUN = 1, an interrupt is asserted to the chip level interrupt controller.
6	HBE_INTEN	R/W	0	If HBE_INTEN = 1 and HBE = 1, an interrupt is asserted to the chip level interrupt controller.
5	BUF2_INTEN	R/W	0	If BUF2_INTEN = 1 and BUF2_EMPTY = 1, an interrupt is asserted to the chip level interrupt controller.
4	BUF1_INTEN	R/W	0	If BUF1_INTEN = 1 and BUF1_EMPTY = 1, an interrupt is asserted to the chip level interrupt controller.
3	ACK_UDR	W	0	1= Clear UNDERRUN. 0=No effect. Always reads as 0.
2	ACK_HBE	W	0	1= Clear HBE. 0= No effect.Always reads as 0.
1	ACK_BUF2	W	0	1= Clear BUF2_EMPTY. Informs SPDO that DMA buffer 2 is full. 0= No effect. Always reads as ‘0’. SPDO_BASE2 is then used to fetch buffer1 data from memory.
0	ACK_BUF1	W	0	1= Clear BUF1_EMPTY. Informs SPDO that DMA buffer 1 is 0= No effect. Always reads as 0. SPDO_BASE1 is then used to fetch buffer1 data from memory.
Offset 0x10 9008	***Reserved***			
31:0	Reserved	-	R/W	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
Offset 0x10 900C	***SPDO_BASE1***			
31:6	SPDO_BASE1	0x0	R/W	Contains the memory address of DMA buffer 1. If changed it must be set before ACK_BUF1.
5:0	Reserved	-	R	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
Offset 0x10 9010	***SPDO_BASE2***			
31:6	SPDO_BASE2	0x0-	R/W	Contains the memory address of DMA buffer 2. If changed it must be set before ACK_BUF2.
5:0	Reserved	-	R	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
Offset 0x10 9014	***SPDO_SIZE***			

Table 6: SPDO Registers *…Continued*

Bit	Symbol	Access	Value	Description
31:6	SPDO_SIZE	0x0	R/W	Determines the size, in bytes, of both DMA buffers
5:0	Reserved	-	R	To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
Offset 0x10 9018—9FF0	***Reserved***			
Offset 0x10 9FF4	***SPDO_PWR_DWN***			
31:1	Reserved	-		To ensure software backward compatibility unused or reserved bits must be written as zeros and ignored upon read.
0	PWR_DWN	0x0	R/W	Used to provide power control status for system software block power management.
Offset 0x10 9FFC	***SPDO_MODULE_ID***			
31:16	ID	0x0121	R	Module ID. This field identifies the block as type SPDO. ID=0x0121
15:12	MAJ_REV	0	R	Major Revision ID. This field is incremented by 1 when changes introduced in the block result in software *incompatibility* with the previous version of the block. First version default = 0
11:8	MIN_REV	0x1	R	Minor Revision ID. This field is incremented by 1 when changes introduced in the block result in software *compatibility* with the previous version of the block. First version default = 0.
7:0	APERTURE	0	R	Aperture size. Identifies the MMIO aperture size in units of 4 KB for the SPDO block. SPDO has an MMIO aperture size of 4KB. APERTURE = 0: 4KB

Chapter 18: SPDIF Input

PNX15xx Series Data Book – Volume 1 of 1

Rev. 2 — 1 December 2004 Product data sheet

1. Introduction

The SPDIF input block accepts digital serial input that complies with the IEC60958 format specification for audio bitstreams. The interface locks onto and decodes the incoming “biphase-mark” encoded signal and recognize all preambles associated with the IEC60958 audio format.

1.1 Features

Key functions include:

- Clock extraction and decode of incoming “biphase-mark” encoded serial bitstream
- Recognition of all preamble types: B, M, W
- Support for 17- to 24-bit PCM coded or non-PCM coded data types
- DMA of incoming audio samples into memory
- Raw mode 32-bit capture of incoming sub-frames
- Interrupt on parity or validity error as well as others
- Internal loopback with SPDIF out - Diagnostic mode
- Support for IEC61937 non-PCM bitstreams format
- Capture of channel status and user information to MMIO registers

2. Functional Description

2.1 SPDIF Input Block Level Diagram

The SPDIF high level block diagram is presented in Figure 1. The SPDIF receiver input samples the input bitstream at a much higher clock rate than the input source bitrate. During this process, the SPDIF bitclock and data are recovered. This sampled “synchronous” audio stream is then passed to an SPDIF decoder function. Internally, the SPDIF decoder produces a decoded binary representation of the ‘bi-phase’ data stream. At its output, the decoder produces a framed audio data format with separate framing, data and clock signals. This framed audio data is fed to the DMA unit for

storage in main memory using a hardware double buffered scheme. In addition, during the decode phase, the input stream is processed to extract parity, validity and selected channel status information for each IEC60958 block.

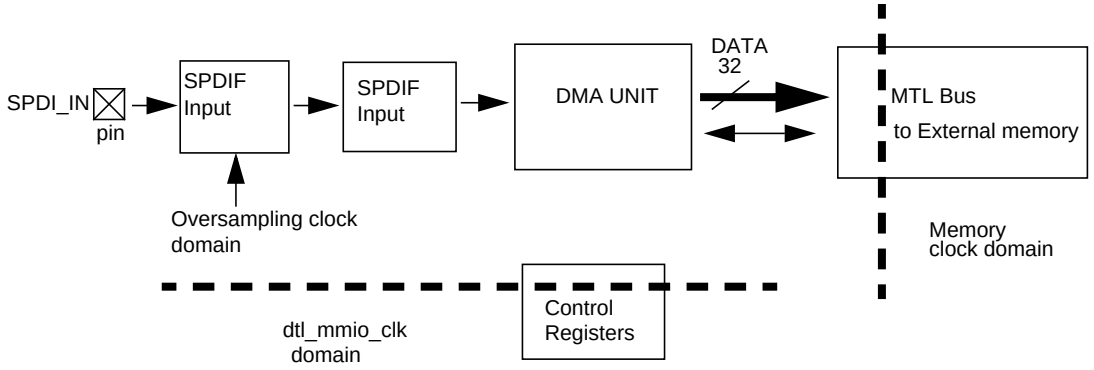


Figure 1: SPDIF Input Block Diagram

The SPDIF bitstream is composed of a single signal that is organized into a block structure of 192 frames. The signal has both data and an embedded clock present. Each frame is composed of 2 subframes each composed of 32 bits. The stream is encoded with a line code called “bi-phase mark” encoding. Figure 2 shows the organization of the IEC60958 SPDIF stream format.

The input stream is parsed by the hardware using an extracted bitclock that is synchronous to the oversampling clock. The audio data, validity flag, channel status and parity bits are extracted and the SPDI_STATUS and SPDI_CBITS registers are updated, see Figure 9. The audio portion of each subframe can contain samples that are up to 24 bits in length.

2.2 Architecture

2.2.1 Functional Modes

The SPDIF Input module has 3 major functional modes. All modes are configured via software programmable MMIO registers, see Figure 9. These modes are:

- **16-bit Mode**: Subframe bits [27:12] inclusive are selected and stored. All biphase encoded bits are decoded. In addition, the state of the parity bit and the validity bit of each subframe is sampled and the SPDI_STATUS register is updated with the results. This mode is useful when the stream contains either 16-bit PCM audio or 16-bit non-PCM samples.

32-bit Mode: Subframe bits [27:4] inclusive are selected and stored subject to a programmable bitmask. A 32-bit word is formed by padding ‘0’ bits to the *least significant* end of the masked audio samples. In addition, the state of the parity bit and the validity bit of each subframe is sampled and the SPDI_STATUS register is updated with the results. This mode provides for any audio sample size ranging from 17 to 24 bits.

- **Raw capture mode**: Subframe bits [31:0] inclusive are selected and stored. All “biphase” encoded bits are decoded and a binary representation of the IEC60958 subframe is generated. The entire 32-bit binary number is then stored as one unit in memory.

Section 3.2 provides detailed information regarding use of all functional modes.

2.3 General Operations

2.3.1 Received Serial Format

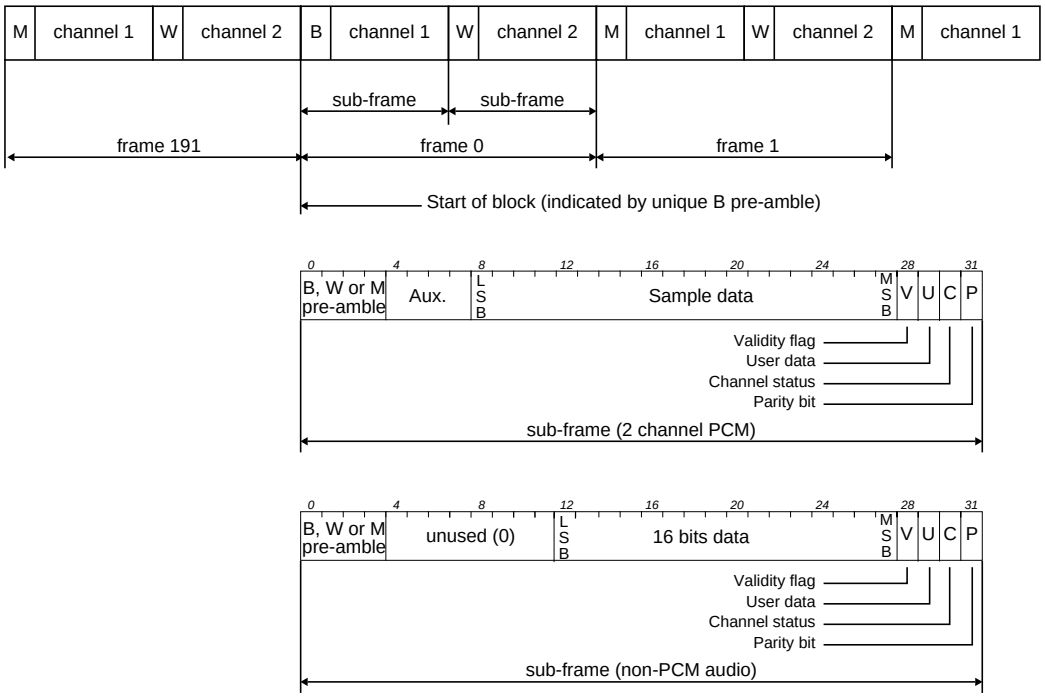


Figure 2: Serial Format of an IEC60958 Block

2.3.2 Memory Formats

The S/PDIF input block will copy the incoming samples into memory in the order that they occur in the input bitstream. The input bitstream is parsed and audio samples are captured and packed as 32-bit words prior to being written to memory. The SPDIF Input decoder always decodes the bi-phase encoded samples into binary prior to transfer to memory. Refer to Figure 4 for the memory formatting for each of the sample sizes supported by the SPDIF Input module.

- **16-bit mode**: The input samples are packed into 32-bit words consisting of two 16-bit samples per 32-bit word. This mode is compatible with the TM1100 and PNX2700 Audio Out memory formats.

- **32-bit mode**: For 17 through 24-bit audio, the samples are formatted into 32-bit words and placed in memory at consecutive 32-bit addresses. For these sample sizes, the sample is first combined with a programmable bitmask

SPDI_SMPMASK.SMASK. The result of the mask operation is zero extended, at the *least significant* end, to the full 32-bits before being placed in memory. The resultant 32 bit words are of the form: 0x*nnnnmm*00 where the *n*'s are the 16 msbits of the sample, the *m's* are the masked 8 lsbits subject to SMASK, see Section 3.2.9. This mode produces audio that is compatible with the PNX2700 Audio Out memory formats.

- **Raw capture mode**: Input subframes are captured and all 'bi-phase' encoding (bits [4:31]) is replaced with a binary representation. The preamble portion (bits [0:3]) of the subframe is replaced with a code indicating what preamble was present, see Figure 3. All parts of the subframe are assembled in order and this 32-bit word is then placed in memory. This mode can be used for "pass-through" of audio to an external SPDIF OUT block. The external SPDIF OUT block must be configured appropriately.

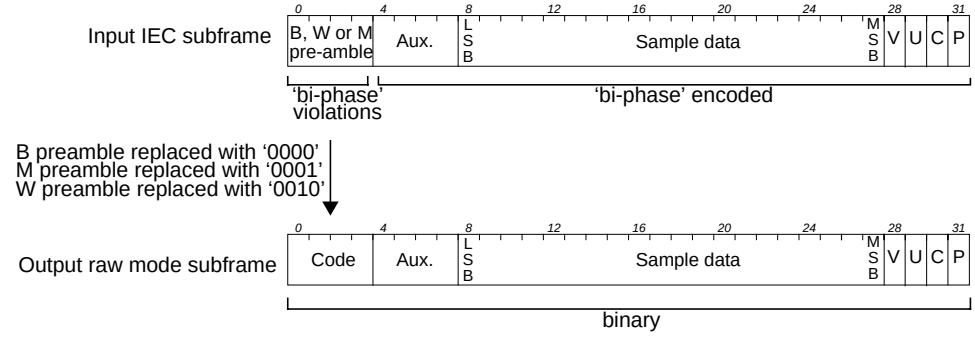


Figure 3: SPDIF Input: Raw Mode Format

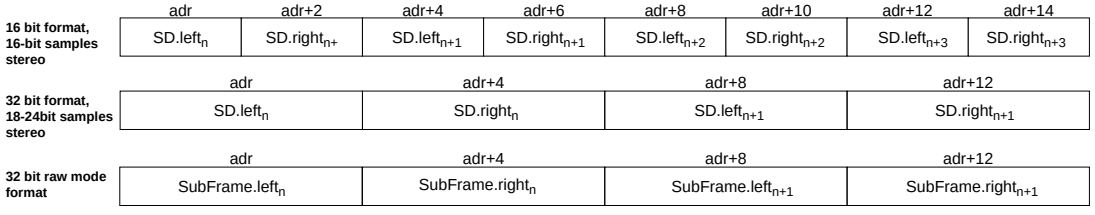


Figure 4: SPDIF Input Sample Order View of Memory

2.3.3 SPDIF Input Endian Mode

The SPDIF Input module can store data in memory using either big-endian or little-endian formatting.

The format in memory for both little and big-endian byte ordering is shown in Figure 5

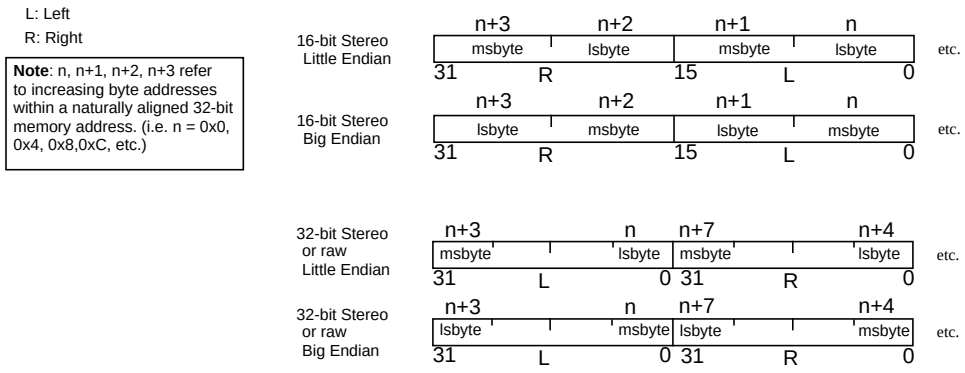


Figure 5: Endian Mode Byte Address Memory Format

2.3.4 Bandwidth and Latency Requirements

Normally, the rate of transmission of frames corresponds exactly to the source sampling frequency. The maximum latency requirement will be for 96 kHz streams (i.e. frame rate = 96 kHz) with the SPDIF Input input set up for any of the 32-bit capture modes:

(96K frames/sec) x (8bytes/ frame) = 0.768Mbytes/sec

The maximum latency allowed in order to sustain this transfer rate is (assuming data transfers are 64 bytes each):

64 bytes/N sec= 0.768 Mbytes/sec

Solving for N and providing a relation,

$$N \leq 83.33\text{uSec} \quad (14)$$

For error-free operation during sustained DMA, there needs to be one 64 byte DMA write transfer completed to memory every 83 usecs. This guarantees the latency requirement for the worst case input sample rate. If the latency requirement is not met, the hardware sets the HBE bit in the SPDI_STATUS register to logic ‘1’ indicating a bandwidth error. For this condition, one or more audio samples have been lost and are not recoverable. The bus arbitration for the SPDIF Input input block should be adjusted by the user to satisfy this latency requirement. Refer to section Section 3.2 for details on SPDI_STATUS and other registers.

3. Operation

3.1 Clock Programming

3.1.1 SPDIF Input Clock Domains

The SPDIF Input module operates using two clock domains. From Figure 1, the SPDIF receiver, decoder and DMA unit use an oversampling clock. The oversampling clock frequency is software selectable via the chip central clocking function. The registers blocks use a dtl_mmio clock and resynchronize what's needed into SPDIF receiver oversampling clock

The source input stream is oversampled by the SPDIF receiver and a representation of the bi-phase data bitstream is produced along with a separate internal 64*Fs* bitclock. The oversampling clock used to sample the input stream is a low jitter, divided form of the system PLL clock. The frequency of the oversampling clock must be within a certain frequency range described by the following equation

$$1220\mathrm{Fs} \le \mathrm{Fosclk} \le 2400\mathrm{Fs}\ . \quad (15)$$

where *Fs* is the incoming sample rate. Factors affecting this frequency range are beyond the scope of this document.

To guarantee error free capture for all sample rates, the oversampling frequency *Fosclk* must be set to a nominal value. The SPDIF receivers' internal oversampling clock frequency can be programmed by selecting a clock divider setting in the central clock functional block (see Chapter 5 The Clock Module for oversampling clock programming details). The divider selections and clocks that are produced are shown in Table 1.

Table 1: SPDIF Input Oversampling Clock Value Settings

Input Audio Sample Rate: fs (kHz)	Central PLL Base Frequency (64x27 MHz)/4	Central Clock Divider n	Fosclk: Oversampling Clock freq (MHz)
96	432 MHz	3	144.00
32.0, 44.1 and 48.0	432 MHz	6	72.0

3.1.2 SPDIF Receiver Sample Rate Tolerance and IEC60958

Three levels of sampling frequency accuracy are specified in the IEC60958 document. The SPDIF receiver will achieve lock onto a level III signal (*variable pitch shift of +/- 12.5% of Fs*) with respect to all the standard sampling frequencies; 32 kHz, 44.1 kHz and 48 kHz as well as the higher 96 kHz. For this design, the SPDIF receiver is classified as a level III compliant receiver.

3.1.3 SPDIF Input Receiver Jitter Tolerance

The maximum tolerable input jitter of the SPDIF Input receiver is described by the equation

$$\mathrm{Tmax(jitter)} = \pm 0.13 \times \frac{1}{128\mathrm{Fs}} \quad (16)$$

or 0.26UI pk-pk (1 UI = 1/128 *Fs*). For a particular *Fs*, the max jitter is shown in Table 2.

Table 2: Input Jitter for Different Sample Rates

Fs (kHz)	1 UI = 1/(128 Fs) (nsec)	Max Jitter = 0.26UI pk-pk (nsec)
32	244.1	31.7
44.1	177.2	23.0
48	162.8	21.2
96	81.4	10.6

The SPDIF Input receiver will reproduce the input data and clock without error if the maximum input jitter remains within the specified max jitter tolerance above. The receiver design meets and exceeds the IEC60958-3 consumer jitter requirements specification (i.e *0.25 UI pk-pk between 200 Hz and 400 kHz jitter freq.*).

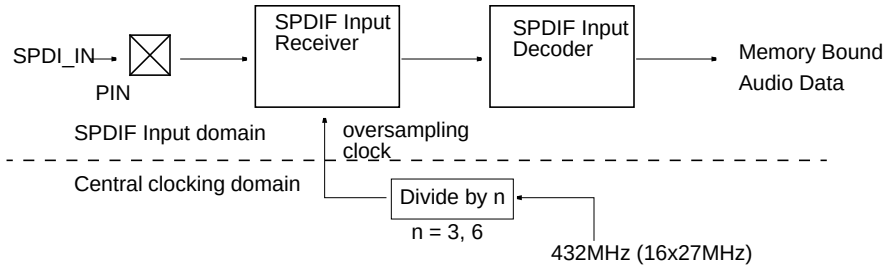


Figure 6: SPDIF Input Oversampling Clock Generation

3.1.4 SPDIF Input and the Oversampling Clock

The oversampling clock supplied to the input receiver is derived from a divider in the central clock control block. The divider value to select is determined by Table 1. Once the oversampling clock has been selected by programming a divider value, the condition of the LOCK bit status indicator in SPDI_STATUS provides feedback on whether the selected oversampling clock has allowed the interface to achieve lock onto the incoming SPDIF input stream. The settings provided by the divider for the oversampling clock are sufficient for capture of 32 kHz, 44.1 kHz, 48 kHz and 96 kHz sample rate input streams.

3.2 Register Programming Guidelines

3.2.1 SPDIF Input Register Set

The S/PDIF register set is outlined in Figure 9 on page 18-15 and Figure 10 on page 18-16. The register set is composed of status and control functions necessary to configure SPDIF Input data capture and DMA of audio data to main memory. To ensure compatibility with future devices, any reserved MMIO register bits in Figure 9 and Figure 10 should be *ignored when read, and written as zeroes*.

3.2.2 SPDI_STATUS Register Functions

The UCBITS bit indicates that a complete set of user data and channel status bits is available in the SPDI_CBITSx and SPDI_UBITSx registers. This bit is updated on a block basis.

The LOCK bit indicates whether the interface has locked onto the incoming SPDIF stream. Software must detect the lock condition asserted before enabling data capture. The LOCK bit is active as long as an oversampling clock is supplied. Capture does not have to be enabled for the LOCK bit to be active.

The UNLOCK bit indicates that the receiver has either: 1) Not attained the locked state -or- 2) Has fallen out of the locked state for some reason. At power on, the interface will indicate that it is NOT locked by asserting the UNLOCK status bit. During normal operation, when a valid SPDIF input stream is applied to the pin interface, the receiver will indicate a lock condition is attained by asserting the LOCK bit. If the receiver then looses lock, the UNLOCK bit will be asserted. Each of the UNLOCK and LOCK status bits can be used to generate an interrupt to the system. Note that the LOCK and UNLOCK status bits are sticky, meaning that once they are set in the status register, each must be cleared explicitly by writing to the appropriate bit in the SPDI_INTCLR register.

The VERR and PERR bits indicate validity and parity error conditions associated with the data contained within the input stream. Note that when validity errors occur, the associated payload data is passed unchanged. When parity errors occur, the associated payload data is muted (zeroed). This conditions apply to all modes except raw mode capture. For raw mode, the entire decoded subframe is passed to memory regardless of parity.

OVERRUN and HBE indicate data loss conditions in memory and internally to the hardware. BUF1_ACTIVE indicates whether memory buffer 1 is currently being filled with data. BUF1_FULL and BUF2_FULL indicate whether memory buffers are completely used. Refer to Table 6 for more details.

3.2.3 LOCK and UNLOCK State Behavior

Shortly after power on (or any reset), the receiver will indicate that it is not locked to an input stream by asserting the UNLOCK bit in SPDI_STATUS. Later, once a valid SPDIF input stream is applied to the interface, the receiver will assert the LOCK status bit. At this moment, the receiver has determined that the input stream is valid and that DMA capture can be started by the user. Note that these two status bits, LOCK and UNLOCK are sticky and may become activated regardless of the state of the SPDIF Input capture enable bit SPDI_CTL.CAP_ENABLE. Software must explicitly clear the LOCK and UNLOCK status bits using the appropriate SPDI_INTCLR register bits. The SPDIF Input receiver will *never* be in a locked state *and* in an unlocked state simultaneously. Also note that the LOCK and UNLOCK behavior applies to all capture modes. For raw mode processing, parity and validity bits are ignored. PERR or VERR status bits are not updated.

3.2.4 UNLOCK Error Behavior and DMA

If the receiver should encounter an error condition in the stream, it will indicate this by asserting the UNLOCK bit. During runtime, the conditions that can cause the UNLOCK state are: 1) an unexpected bi-phase error is encountered; 2) the input

stream is not present or is suddenly removed, and 3) the input signal contains too much jitter. If the UNLOCK_ENBL bit is set, the SPDIF Input will generate an interrupt. Note that parity and validity errors (PERR and VERR) do *not* cause out of lock conditions.

From the point where the error condition occurred, the contents of the currently filling internal 64 byte buffers are muted (zeroed). The external memory buffers will receive muted data from this point forward. If the receiver does not re-lock before the current external memory buffer is filled to completion with muted data, DMA will halt. DMA is halted in this way so that bus resources are not further utilized. Otherwise, DMA continues with valid data soon after lock is reacquired. From the error point onward, the last stable capture sample rate will be maintained by the hardware automatically during this error condition processing. The following is a start-up software process flow for capture of an SPDIF stream.

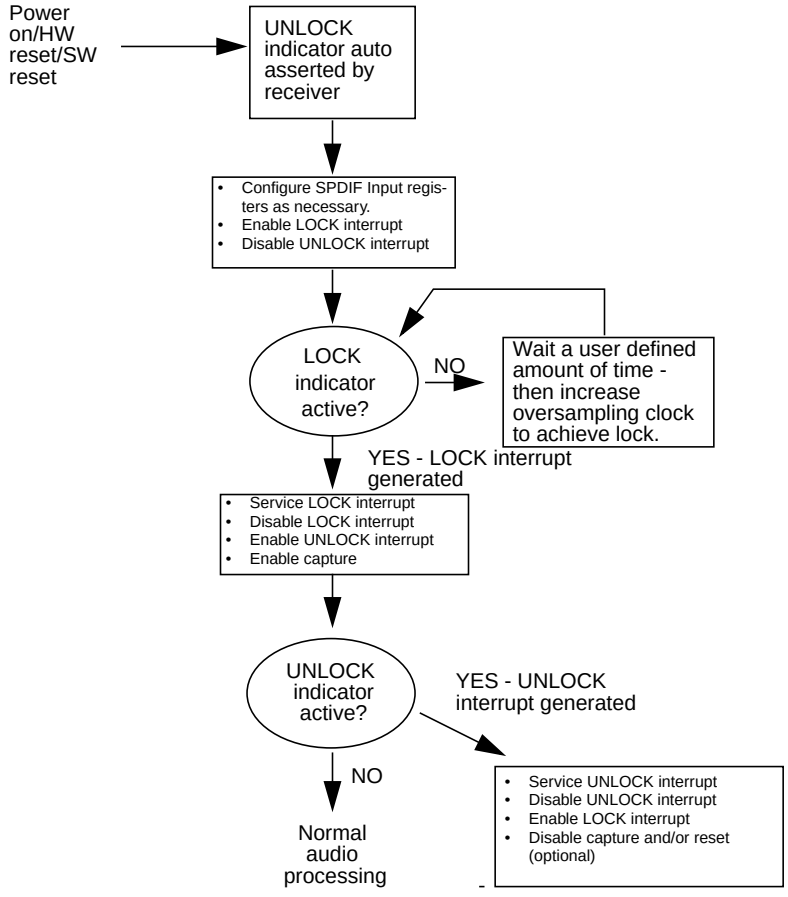


Figure 7: Lock/Unlock Processing for SPDIF Input

3.2.5 SPDI_CTL and Functions

The SPDI_CTL register provides system control of the SPDIF Input interface. The RESET bit is used to completely reset the interface and all registers. The result of asserting the RESET bit is all SPDIF Input capture activity stops and all registers are initialized to logic ‘0’s. In addition, any pending SPDIF Input interrupts are cleared and disabled. Any pending DMA activity is cancelled and active request are aborted.

The CAP_ENABLE and SAMP_MODE bits enable capture of audio data in a particular format as described in Table 6 on page 18-17 and Section 2.3.2. The CHAN_MODE bits allow capture in either stereo (left/right sample pairs) or mono (primary or secondary channel). The CHAN_MODE setting is independent of the UCBITS_SEL setting.

The DIAG_MODE bit is used in conjunction with an SPDIF Out block. The DIAG_MODE bit configures the SPDIF Input source to be the output pin of the SPDIF Out block. Programmers must configure SPDIF Input and SPDIF Output modules appropriately to use the loopback properly. The memory formats used while this bit is enabled are unaffected. The only difference is that the source of the input stream is internal rather that the external SPDIF Input pin.

3.2.6 SPDI_CBITSx and Channel Status Bits

The channel status block indicates the status of the currently received audio stream. Its structure is different for each of the consumer or professional IEC60958 formats. Within each 32-bit subframe is a channel status bit at location bit[30] in the 32-bit word. The two ‘C’ bits, one for each of the subframes within a frame, can be different. This can occur, for instance, while receiving a 2-channel stream. SPDI_CBITS1..SPDI_CBITS6 hold channel status bits embedded in the source SPDIF stream.

The SPDI_CBITSx registers are updated on a block basis. Upon the occurrence of each new B preamble in the source stream, the SPDI_CBITSx registers are updated with selected channel status information from the *previous* block. Programmers can use the SPDI_CBITSx registers to determine the state of the SPDIF source material. Information such as whether the stream is a consumer or professional type, sample rate and sample size can be determined as well as other information. The SPDI_CTL.UCBITS_SEL register determines which set (subframe 1 or 2) of 192 channel status bits will be captured. A selected set of the channel status bits captured by the SPDI_CBITS registers are shown in Table 3 and Table 4.

Table 3: SPDI_CBITS1 Channel Status Meaning

SPDI_CBITS1[n]	IEC Consumer Channel Status Bit No.	IEC Consumer Meaning	AES/EBU Professional Channel Status Bit No.	AES/EBU Professional Meaning
0	0	Consumer mode	0	Professional mode
1	1	PCM/non-PCM data	1	PCM/non-PCM data
2	2	Copyright	2	Emphasis
3	3	Format	3	Emphasis
4	4	Format	4	Emphasis
5	5	Format	5	Locked
6	6	Mode	6	Sample rate
7	7	Mode	7	Sample rate
8	8	Category code	8	Channel mode
9	9	Category code	9	Channel mode
10	10	Category code	10	Channel mode
11	11	Category code	11	Channel mode

Table 3: SPDI_CBITS1 Channel Status Meaning *...Continued*

SPDI_CBITS1[n]	IEC Consumer Channel Status Bit No.	IEC Consumer Meaning	AES/EBU Professional Channel Status Bit No.	AES/EBU Professional Meaning
12	12	Category code	12	User bit mgmt
13	13	Category code	13	User bit mgmt
14	14	Category code	14	User bit mgmt
15	15	Category code	15	User bit mgmt
16	16	Source Number	16	Use of aux bits
17	17	Source Number	17	Use of aux bits
18	18	Source Number	18	Use of aux bits
19	19	Source Number	19	Source word length - Source encode history
20	20	Channel number	20	Source word length - Source encode history
21	21	Channel number	21	Source word length - Source encode history
22	22	Channel number	22	Source word length - Source encode history
23	23	Channel number	23	Source word length - Source encode history
24	24	Sample rate	24	Multi-channel function
25	25	Sample rate	25	Multi-channel function
26	26	Sample rate	26	Multi-channel function
27	27	Sample rate	27	Multi-channel function
28	28	Clock accuracy	28	Multi-channel function
29	29	Clock accuracy	29	Multi-channel function
30	30	reserved	30	Multi-channel function
31	31	reserved	31	Multi-channel function

Table 4: SPDI_CBITS2 Channel Status Meaning

SPDI_CBITS2[n]	IEC Consumer Channel Status Bit No.	IEC Consumer Meaning	AES/EBU Professional Channel Status Bit No.	AES/EBU Professional Meaning
0	32	Word length	32	Digital audio reference signal
1	33	Word length	33	Digital audio reference signal
2	34	Word length	34	reserved
3	35	Word length	35	reserved
4:31	36:63		36:63	

3.2.7 SPDI_UBITSx and User Bits

The complete set of user data channel bits are available in the SPDI_UBITSx registers. The SPDI_UBITSx registers are updated on a block basis. Upon the occurrence of each new B preamble in the source stream, the SPDI_UBITSx

registers are updated with user data bits information from the *previous* block. The meaning of the user data is application dependent. The SPDI_CTL.UCBITS_SEL register determines which set (subframe 1 or 2) of 192 user data bits will be captured.

3.2.8 SPDI_BASEx and SPDI_SIZE Registers and Memory Buffers

SPDI_BASE1 and SPDI_BASE2 are used to select the main memory buffer starting addresses used for DMA of audio data samples. At the start of SPDIF Input capture, the hardware will begin filling the memory buffer beginning at the address specified in the SPDI_BASE1 register. Once this buffer is filled, the hardware will switch buffers and begin filling the memory buffer starting at the address specified in the SPDI_BASE2 register. The size of these DMA buffers is specified in the SPDI_SIZE register. Note that the hardware limits the buffer size and starting address to be aligned to 64 byte addresses. Assignment to SPDI_BASE1, SPDI_BASE2 and SPDI_SIZE have no effect on the state of the SPDI_STATUS flags. Any change to the SPDI_BASE1 or SPDI_BASE2 registers should only be done while a memory buffer is not being used by the hardware DMA.

3.2.9 SPDI_SMPMASK and Sample Size Masking

The SPDI_SMPMASK register allows per bit masking the *least significant 8 bits* of the incoming samples (corresponding to subframe bits [11:4]). The SMASK setting *only* applies to 32-bit capture mode (i.e. SAMP_MODE = 01). The 8 bits of SMASK will determine which subframe bits [11:4] will be captured and stored in memory. To reject a particular bit (*within subframe bits [11:4]*) in an audio sample, set the corresponding SMASK[7:0] bit to logic ‘1’. The default value of SMASK is 0x00.

Note the sense of the mask operation! Setting SMASK[7:0] bits to logic ‘1’ will zero the corresponding subframe bit [11:4]. Others will pass unchanged!

Ex 1. The capture mode is configured for 32-bit stereo capture (SAMP_MODE = 01 and CHAN_MODE = 00). It is desired to store only 20 bit sample pairs. The left/right channels incoming subframe bits [27:4] consist of a 24 bit samples of the form L = 0xABCDEF and R = 0x123456 respectively. To retain the most significant 20 bits (i.e. keep ‘ABCDE’ in the left sample and ‘12345’ in the right sample) and write them to memory, set the SMASK[7:0] bits to ‘0x0F’. During capture and once the samples are masked, the least significant ends are zero extended to 32 bits. The result is a 32 bit sample pair of the form L = ‘0xABCDE000’ and R = ‘0x12345000’. The samples are finally stored in memory subject to endian control.

The masking operation applies to all memory bound samples.

3.2.10 SPDI_BPTR and the Start of an IEC60958 Block

During SPDIF Input capture, memory buffers are continuously filled with input data. As input blocks are filling the memory buffers, the address of the first instance of a frame 0 in a particular memory buffer changes continuously. To aid software with the task of finding the start of a block in memory, the SPDI_BPTR contains the address of the first occurrence of a frame 0 (indicating the starting boundary of a complete 192 frame block) within the last filled memory buffer - either BUF1 or BUF2. This function is useful during capture of non-PCM coded data as found in IEC61937 data streams. The software driver can use SPDI_BPTR to find the beginning of an IEC60958 block and then quickly determine the location of any sync condition thereafter embedded in the non-PCM data structure.

3.2.11 Interrupts

The SPDI_STATUS register contains status flags that indicate certain conditions that may need the attention of the chip level controller. Each of these conditions can be used as interrupt sources.

The SPDI_INTEN register is used to provide the capability to enable any of the interrupt source bits in the SPDI_STATUS register. To enable one of the interrupts shown in the SPDI_STATUS register, the programmer must set the corresponding SPDI_INTEN bits for that interrupt source. For example, to allow an interrupt to be passed to the chip level interrupt controller upon the occurrence of a parity error in the incoming stream during capture, the user must write a logic ‘1’ to the PERR_ENBL bit in SPDI_INTEN. To disable an interrupt source, write a logic ‘0’ to the appropriate bit in SPDI_INTEN. The effect of writing a logic ‘0’ to an enable bit while the particular interrupt is active is that the interrupt is unconditionally de-asserted and disabled.

The status conditions in the SPDI_STATUS register will be ‘sticky’, meaning the SPDI_STATUS bit will remain active until explicitly cleared by setting the corresponding clear bit in the SPDI_INTCLR register. Using the same example above, if the parity error bit PERR is enabled (SPDI_INTEN.PERR_ENBL = 1) and the PERR interrupt is active currently, to clear the PERR bit and deactivate the interrupt the user must write a logic ‘1’ to the PERR_CLR bit in the SPDI_INTCLR register.

The SPDI_INTSET register is useful for software diagnostic generation of interrupts. Setting any of these bits to logic ‘1’ will generate an interrupt to the chip level interrupt controller. To use the SPDI_INTSET register to generate interrupts, the same enable rule applies as outlined above.

For SPDIF Input, the hardware interrupt signal is “level triggered”. This means the interrupt signal passed to the chip level interrupt controller will be logic ‘1’ when active and will remain so until cleared explicitly by the system interrupt handler software.

3.2.12 Event Timestamping

SPDIF Input has no timestamping internal to the block. Instead, SPDIF Input exports several event notification signals to the central timestamping function on chip, see Chapter 8 General Purpose Input Output Pins and Section 8.1 on page 3-27 in Chapter 3 System On Chip Resources. The central timestamp function includes timers and timestamp registers to provide event counts and event triggered “snapshot” clock values. The event signal is a positive edge going pulse with positive level duration greater than or equal to 160 ns.

The specific events that are exported to the central timestamp function are:

- WS (Word strobe) - this event signals the arrival of a sample pair on the SPDIF Input interface. The rising edge of the signal indicates the beginning of either the B or M subframe. The logic “high” duration is as stated above.

- SWS (Last subframe) - this event signal indicates that a single 32-bit subframe *corresponding to the last sample in the currently filling memory buffer* has been received at the input to SPDIF Input. The event is NOT qualified with a particular block boundary. This represents a precise, periodic event for use by system software to achieve audio/video synchronization.
- All SPDI_STATUS register bits (except LOCK) - these events can be used by software to either count or timestamp any interrupt generated by SPDIF Input. Refer to Table 6 for details regarding SPDIF Input interrupt sources.

4. Signal Descriptions

4.1 External Interface Pins

The SPDIF Input module has a single input pin. The signal applied to this pin must have TTL level voltage swing.

Table 5: SPDIF Input Pin Summary

Signal	Type	Description
SPDI_IN	IN	Single-ended SPDIF Input pin. Input sample rate can be 32 kHz, 44.1 kHz, 48 kHz or 96 kHz. Input signal must be TTL compatible.

For the commonly found 0.5 Vpp SPDIF signal (IEC60958 consumer mode), the user must externally restore the signal to TTL voltage levels. In all cases, external isolation of the input signal is recommended.

4.1.1 System Interface Requirements

IEC60958 specifies that consumer systems have a 0.5 Vpp signal driven from the transmitter into an unbalanced cable with a 75 ohm nominal impedance. The load side must present a 75-ohm resistive impedance over the frequency band of 0.1 to 6 MHz. Figure 8 presents an input circuit that satisfies the load requirements. The circuit presents a simple RS422 differential receiver. The chosen receiver should have good input hysteresis. Also, the signal applied to the SPDIF Input input pin should ideally have a 50% duty cycle. It is recommended that the system designer add an isolation transformer to the input circuit. Other consumer input circuits may be possible.

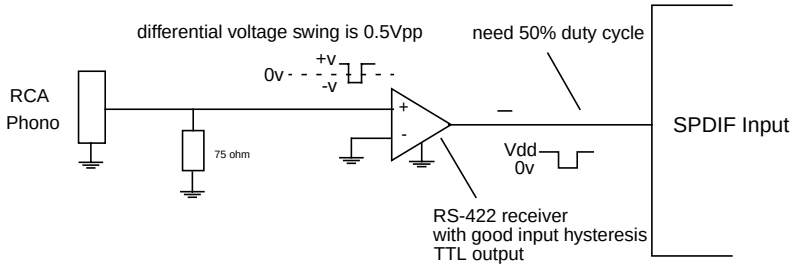


Figure 8: SPDIF Input Consumer interface

5. Register Descriptions

5.1 Register Summary

5.1.1 SPDIF Input Interrupt Registers

The registers SPDI_STATUS, SPDI_INTSET, SPDI_INTCLR and SPDI_INTEN support DVP block level interrupt processing. The SPDIF Input interrupt control mechanism is discussed in detail in section Section 3.2.11.

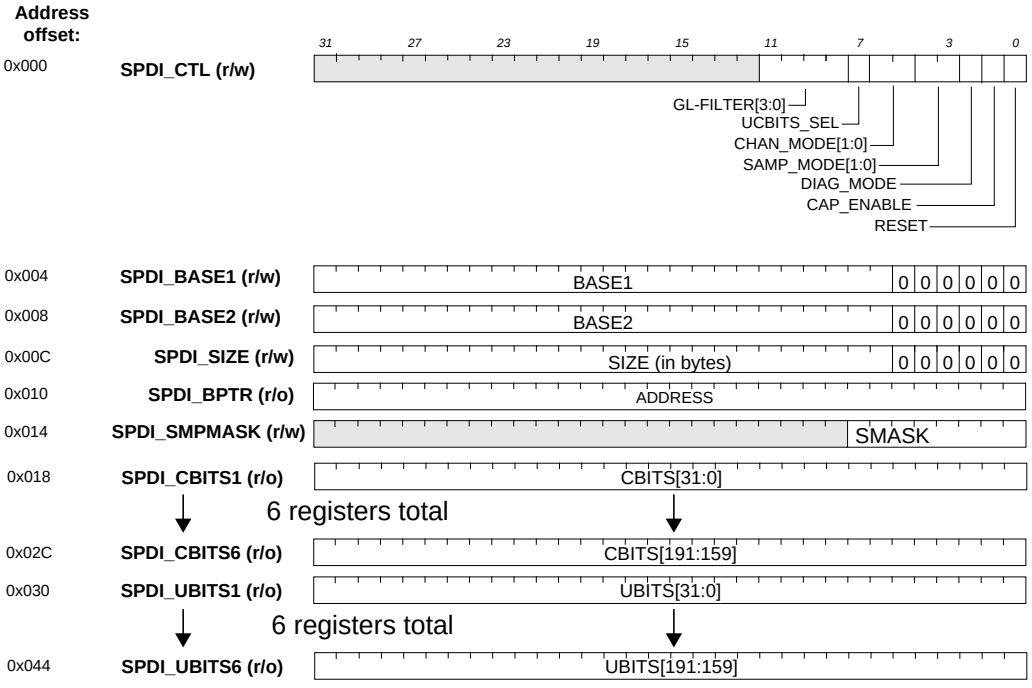


Figure 9: SPDIF Input MMIO Registers (1 of 2)

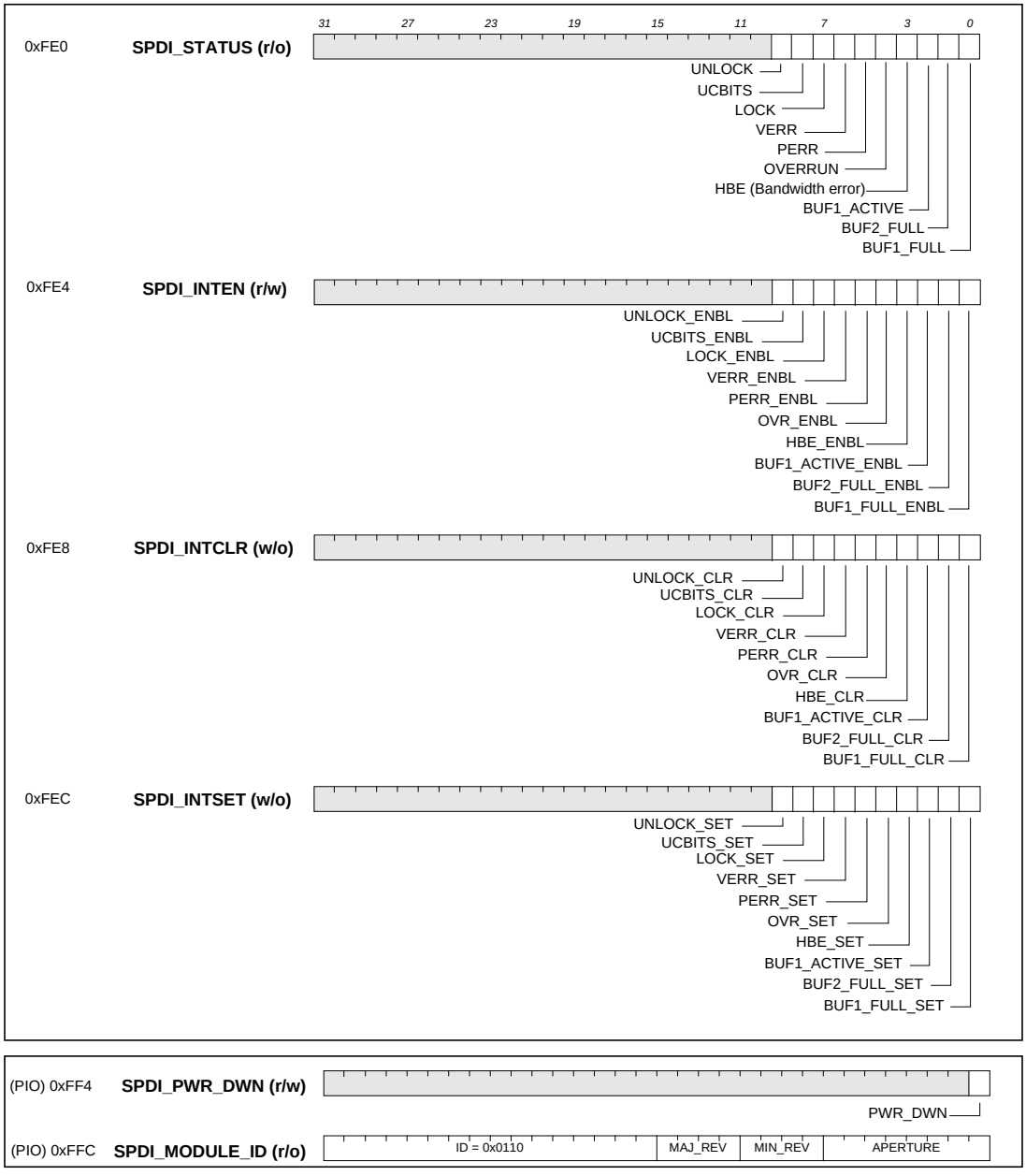


Figure 10: SPDIF Input MMIO Registers (2 of 2)

5.2 Register Table

Table 6: SPDIF Input Registers

Bit	Symbol	Access	Value	Description
Offset 0x10 A000	*SPDI_CTL*			
31:9	Unused		-	
11:8	GL_FILTER	R/W	0	Input glitch filter control. These bits control the rejection of a glitch on the SPDI interface. 0000 = The Glitch Rejection Filter is disabled. 0001 .. 1111 = An incoming signal transition must remain stable for (programmed value + 1) rising edges of OSC_CLK, otherwise it is rejected as a glitch.
7	UCBITS_SEL	R/W	0	User/Channel status bits select. Selects the set of subframes from which the user and channel status bits are extracted and written to the SPDI_UBITSx and SPDI_CBITSx registers. This bit is activated only on a block boundary, meaning that the bit can be changed at any time via software, but the update of the SPDI_UBITSx and SPDI_CBITSx registers with the new information will wait until a complete block has been received at the SPDIF Input. 0 = subframe 1 is selected, user and channel status bits are extracted/written to UBITSx and CBITSx registers. 1 = subframe 2 is selected. User and channel status bits are extracted/written to UBITSx and CBITSx registers.
6:5	CHAN_MODE[1:0]	R/W	0	00 = Capture stereo left/right sample pairs (Default). 01 = Capture mono primary (subframe 1) channel. 10 = Capture mono secondary (subframe 2) channel. 11 = Reserved Note: The channel mode should only be changed while capture is disabled (i.e. CAP_ENABLE = 0).
4:3	SAMP_MODE[1:0]	R/W	0	**00 = 16-bit mode**. Subframe bits [27:12] inclusive are selected and stored. Hardware stores a single 16-bit word per subframe. If audio samples are actually larger than 16 bits, the most significant 16 bits of the audio sample will be selected and stored. **01= 32-bit mode**. Subframe bits [27:4] inclusive are selected and stored subject to SMASK. A 32-bit word is formed by bitwise masking the sample (subject to the value of SPDI_SMPMASK.SMASK) and padding '0' bits to the least significant end of the 24 bits. The resultant 32-bit words are of the form: 0xnnnnmm00 where the *n*s are the 16 subframe bits [27:12] and the *m*s are the eight masked subframe bits [11:4]. This provides for any audio sample size from 17 to 24 bits. (See the SPDI_SMPMASK register description for operation of the SMASK feature). SMASK only applies for this sample mode. **10 = Raw capture mode**. The entire subframe is captured and stored. The bi-phase portion of the subframe (i.e., bits [31:4]) are decoded into binary. Bits [3:0] are replaced with a code. The entire 32 bits are then stored as one unit. 11 = Reserved Note: The sample mode should only be changed while capture is disabled (i.e., CAP_ENABLE = '0').

Table 6: SPDIF Input Registers *…Continued*

Bit	Symbol	Access	Value	Description
2	DIAG_MODE	R/W	0	Diagnostic loopback mode. Used to diagnose the SPDIF Input module. 0 = The SPDIF input source is set to the SPDIF Input input pin (default). 1 = The SPDIF input source is set to the SPDIF OUT pin.
1	CAP_ENABLE	R/W	0	Writing a ‘1’ to this bit enables capture per the selected mode. Writing a ’0’ here stops any ongoing capture after completing any actions related to the current audio sample.
0	RESET	R/W	0	Writing a ‘1’ to this bit resets the SPDI block. The registers of the SPDI will all be reset to ‘0s’. This should be used with caution. Any ongoing capture will be interrupted.
Offset 0x10 A004	***SPDI_BASE1***			
31:6	BASE1	R/W	0	Selects the main memory buffer starting addresses used for DMA of audio data samples. Note: Any change to the SPDI_BASE1 register should only be done while a memory buffer is not being used by the hardware DMA. If changed it must be set before BUF1_FULL_CLR.
5:0	Reserved	R	0	
Offset 0x10 A008	***SPDI_BASE2***			
31:6	BASE2	R/W	0	Selects the main memory buffer starting addresses used for DMA of audio data samples. Note: Any change to the SPDI_BASE2 register should only be done while a memory buffer is not being used by the hardware DMA. If changed it must be set before BUF2_FULL_CLR.
5:0	Reserved	R	0	Hardwired to logic ‘0’
Offset 0x10 A00C	***SPDI_SIZE***			
31:6	SIZE (in bytes)	R/W	0	The size of the DMA buffers is specified in the SPDI_SIZE register. Note hardware limits the buffer size and starting address to be aligned to 64-byte addresses. Assignment to SPDI_BASE1, SPDI_BASE2 and SPDI_SIZE have no effect on the state of the SPDI_STATUS flags.
5:0	Reserved	R	0	Hardwired to logic ‘0’
Offset 0x10 A010	***SPDI_BPTR***			
31:0	ADDRESS	R	0	To aid software with finding the start of a block in memory, the SPDI_BPTR contains the address of the first occurrence of a frame 0 (indicating the starting boundary of a complete 192-frame block) within the currently filling memory buffer: BUF1 or BUF2. This is useful during capture of non-PCM coded data found in IEC61937 data streams.
Offset 0x10 A014	***SPDI_SMPMASK***			
31:8	Unused		-	

Table 6: SPDIF Input Registers *...Continued*

Bit	Symbol	Access	Value	Description
7:0	SMASK	R/W	0x00	Allows per bitmasking the least significant 8 bits of the incoming samples (corresponding to subframe bits [11:4]). The SMASK setting only applies to 32-bit capture mode (i.e., SAMP_MODE = 01). The 8 bits of SMASK will determine which subframe bits [11:4] will be captured and stored in memory. Note: Setting SMASK[7:0] bits to logic '1' will zero the corresponding subframe bit [11:4]. Others will pass unchanged.
SPDI_CBITSx Registers				
The SPDI_CBITSx registers contain the current block channel status bits. The meaning of each of the channel status fields can change depending upon whether the input source is a consumer or professional bitstream.				
Offset 0x10 A018	***SPDI_CBITS1***			
31:0	CBITS [31:0]	R	0	Channel Status register 1 contains bytes 0, 1, 2 and 3 of the current Channel Status block according to SPDI_CTL.UCBITS_SEL. It will always reflect the condition of the current decoded block of 192 frames and will always start at the block boundary. Register bit meaning will depend upon the source transmission (i.e., consumer vs. professional). See Table 3 for more details.
Offset 0x10 A01C	***SPDI_CBITS2***			
31:0	CBITS [31:0]	R	0	Channel Status register 2 contains bytes 4, 5, 6 and 7 of the current Channel Status block according to SPDI_CTL.UCBITS_SEL. See Table 4 for more details.
Offset 0x10 A020	***SPDI_CBITS3***			
31:0	CBITS [31:0]	R	0	Channel Status register 3 contains bytes 8, 9,10 and 11 of the current Channel Status block according to SPDI_CTL.UCBITS_SEL.
Offset 0x10 A024	***SPDI_CBITS4***			
31:0	CBITS [31:0]	R	0	Channel Status register 4 contains bytes 12, 13, 14 and 15 of the current Channel Status block according to SPDI_CTL.UCBITS_SEL.
Offset 0x10 A028	***SPDI_CBITS5***			
31:0	CBITS [31:0]	R	0	Channel Status register 5 contains bytes 16,17,18 and 19 of the current Channel Status block according to SPDI_CTL.UCBITS_SEL.
Offset 0x10 A02C	***SPDI_CBITS6***			
31:0	CBITS [191:159]	R	0	Channel Status register 6 contains bytes 20, 21, 22 and 23 of the current Channel Status block according to SPDI_CTL.UCBITS_SEL.
SPDI_UBITSx Registers				
The SPDI_UBITSx registers contain the current block user data channel bits. The meaning of each of the user data fields is dependent upon the application.				
Offset 0x10 A030	***SPDI_UBITS1***			
31:0	UBITS [31:0]	R	0	User bit 1 contains the state of user bytes 0,1, 2 and 3 of the block according to SPDI_CTL.UCBITS_SEL. The SPDI_UBITS register will always reflect the condition of the current decoded block of 192 frames. Register bit meaning will depend upon the source transmission.

Table 6: SPDIF Input Registers *...Continued*

Bit	Symbol	Access	Value	Description
Offset 0x10 A034 SPDI_UBITS2				
31:0	UBITS [31:0]	R	0	User bit 2 contains the state of user bytes 4, 5, 6 and 7 of the block according to SPDI_CTL.UCBITS_SEL. The SPDI_UBITS register will always reflect the condition of the current decoded block of 192 frames. Register bit meaning will depend upon the source transmission.
Offset 0x10 A038 SPDI_UBITS3				
31:0	UBITS [31:0]	R	0	User bit 3 contains the state of user bytes 8, 9, 10 and 11 of the block according to SPDI_CTL.UCBITS_SEL. The SPDI_UBITS register will always reflect the condition of the current decoded block of 192 frames. Register bit meaning will depend upon the source transmission.
Offset 0x10 A03C SPDI_UBITS4				
31:0	UBITS [31:0]	R	0	User bit 4 contains the state of user bytes 12, 13, 14 and 15 of the block according to SPDI_CTL.UCBITS_SEL. The SPDI_UBITS register will always reflect the condition of the current decoded block of 192 frames. Register bit meaning will depend upon the source transmission.
Offset 0x10 A040 SPDI_UBITS5				
31:0	UBITS [31:0]	R	0	User bit 5 contains the state of user bytes 16, 17, 18 and 19 of the block according to SPDI_CTL.UCBITS_SEL. The SPDI_UBITS register will always reflect the condition of the current decoded block of 192 frames. Register bit meaning will depend upon the source transmission.
Offset 0x10 A044 SPDI_UBITS6				
31:0	UBITS [191:159]	R	0	User bit 6 contains the state of user bytes 20, 21, 22 and 23 of the block according to SPDI_CTL.UCBITS_SEL. The SPDI_UBITS register will always reflect the condition of the current decoded block of 192 frames. Register bit meaning will depend upon the source transmission.
Offset 0x10 A048—AFDC Reserved				
Offset 0x10 AFE0 SPDI_STATUS				
31:10	Unused		-	
9	UNLOCK	R	0	UNLOCK active. This flag gets set to logic '1' if the SPDIF Input receiver is NOT locked onto an incoming stream. Programmers can use this UNLOCK indication, in conjunction with the LOCK bit, to determine the state of the receiver or to make a decision to adjust the oversampling frequency. See the definition of the LOCK bit. Possible causes of an out-of-lock state are: i) The oversampling frequency is too high or too low with respect to the applied input SPDIF sample rate. ii) Too much jitter in SPDIF input stream. iii) Absent, invalid or corrupted SPDIF stream applied to the interface/receiver. The flag can be cleared by a software write to UNLOCK_CLR.
8	UCBITS	R	0	User/Channel bits available. This flag is set if a new set of user data bits and channel status bits have been written to the SPDI_UBITSx and SPDI_CBITSx registers. Updated on a block basis.

Table 6: SPDIF Input Registers *…Continued*

Bit	Symbol	Access	Value	Description
7	LOCK	R	0	LOCK active 1 = The SPDIF Input receiver achieved lock onto the incoming stream. Use this LOCK flag, in conjunction with the UNLOCK flag, to determine the state of the receiver or to make a decision to adjust the oversampling frequency. The flag can be cleared by a software write to LOCK_CLR. LOCK means that the internal PLL is locked. A valid sequence of preambles is not required for LOCK.
6	VERR	R	0	Validity Error 1 = The hardware encounters a subframe that has the validity flag set to 1, indicating that the payload portion of the subframe is not reliable. The flag can be cleared by a software write to VERR_CLR.
5	PERR	R	0	Parity Error 1 = The hardware encounters a subframe that has a parity error. Parity is even for the subframe and applies to subframe bits [31:4] inclusive. Normally, the external SPDIF Input transmitter will set the subframe P bit to logic ‘1’ or logic ‘0’ so that bits [31:4] have an even number of logic “1s” and “0s”. The flag can be cleared by a software write to PERR_CLR.
4	OVERRUN	R	0	1 = Both external main memory DMA buffers are filled before a new empty buffer is assigned by the system control CPU. Hardware has performed a normal buffer switch over and is overwriting fresh, unconsumed data. This flag can be cleared by software write to OVR_CLR.
3	HBE (Bandwidth error)	R	0	Bandwidth Error 1 = The internal hardware DMA buffers in SPDI are full and at least one of them was not emptied before new input data arrived on the SPDI interface, indicating that DMA service latency is too long. This flag can be cleared by a software write to HBE_CLR.
2	BUF1_ACTIVE	R	0	This flag is set to logic ‘1’ if the hardware is currently filling memory DMA buffer 1. Otherwise, it is reset to logic ‘0’. This flag can be cleared by a software write to BUF1_ACTIVE_CLR.
1	BUF2_FULL	R	0	This flag is set to logic ‘1’ if memory DMA buffer 2 has been filled by the SPDI hardware. It can be cleared by a software write to BUF2_FULL_CLR.
0	BUF1_FULL	R	0	This flag is set to logic ‘1’ if memory DMA buffer 1 has been filled by the SPDI hardware. It can be cleared by a software write to BUF1_FULL_CLR.
Offset 0x10 AFE4	***SPDI_INTEN***			
31:10	Unused		-	
9	UNLOCK_ENBL	R/W	0	1 = UNLOCK bit in SPDI_STATUS is enabled for interrupts. 0 = UNLOCK bit in SPDI_STATUS is disabled for interrupts.
8	UCBITS_ENBL	R/W	0	1 = UCBITS bit in SPDI_STATUS is enabled for interrupts. 0 = UCBITS bit in SPDI_STATUS is disabled for interrupts.
7	LOCK_ENBL	R/W	0	1 = LOCK bit in SPDI_STATUS is enabled for interrupts. 0 = LOCK bit in SPDI_STATUS is disabled for interrupts.
6	VERR_ENBL	R/W	0	1 = VERR bit in SPDI_STATUS is enabled for interrupts. 0 = VERR bit in SPDI_STATUS is disabled for interrupts.

Table 6: SPDIF Input Registers *...Continued*

Bit	Symbol	Access	Value	Description
5	PERR_ENBL	R/W	0	1 = PERR bit in SPDI_STATUS is enabled for interrupts. 0 = PERR bit in SPDI_STATUS is disabled for interrupts.
4	OVR_ENBL	R/W	0	1 = OVERRUN bit in SPDI_STATUS is enabled for interrupts. 0 = OVERRUN bit in SPDI_STATUS is disabled for interrupts.
3	HBE_ENBL	R/W	0	1 = HBE bit in SPDI_STATUS is enabled for interrupts. 0 = HBE bit in SPDI_STATUS is disabled for interrupts.
2	BUF1_ACTIVE_ENBL	R/W	0	1 = BUF1_ACTIVE bit in SPDI_STATUS is enabled for interrupts. 0 = BUF1_ACTIVE bit in SPDI_STATUS is disabled for interrupts.
1	BUF2_FULL_ENBL	R/W	0	1 = BUF2_FULL bit in SPDI_STATUS is enabled for interrupts. 0 = BUF2_FULL bit in SPDI_STATUS is disabled for interrupts.
0	BUF1_FULL_ENBL	R/W	0	1 = BUF1_FULL bit in SPDI_STATUS is enabled for interrupts. 0 = BUF1_FULL bit in SPDI_STATUS is disabled for interrupts.
Offset 0x10 AFE8	***SPDI_INTCLR***			
31:10	Unused		-	
9	UNLOCK_CLR	R/W	0	1 = Clear UNLOCK bit in SPDI_STATUS. 0 = No effect
8	UCBITS_CLR	W	0	1 = Clear UCBITS in SPDI_STATUS. 0 = No effect.
7	LOCK_CLR	W	0	1 = Clears LOCK in SPDI_STATUS. 0 = No effect.
6	VERR_CLR	W	0	1 = Clear VERR in SPDI_STATUS. 0 = No effect
5	PERR_CLR	W	0	1 = Clear PERR in SPDI_STATUS. 0 = No effect.
4	OVR_CLR	W	0	1 = Clear OVERRUN in SPDI_STATUS. 0 = No effect.
3	HBE_CLR	W	0	1 = Clear HBE in SPDI_STATUS. 0 = No effect.
2	BUF1_ACTIVE_CLR	W	0	1 = Clear BUF1_ACTIVE in SPDI_STATUS. 0 = No effect.
1	BUF2_FULL_CLR	W	0	1 = Clear BUF2_FULL in SPDI_STATUS. 0 = No effect. SPDI_BASE1 must be valid before setting BUF2_FULL_CLR.
0	BUF1_FULL_CLR	W	0	1 = Clear BUF1_FULL in SPDI_STATUS. 0 = No effect. SPDI_BASE1 must be valid before setting BUF1_FULL_CLR.
Offset 0x10 AFEC	***SPDI_INTSET***			
31:10	Unused		-	
9	UNLOCK_SET	W	0	1 = UNLOCK bit in SPDI_STATUS is to be set to logic '1'. Level trigger interrupt will be raised to the external interrupt controller if the corresponding enable bit is set to logic '1'. 0 = No effect

Table 6: SPDIF Input Registers *...Continued*

Bit	Symbol	Access	Value	Description
8	UCBITS_SET	W	0	1 = UCBITS bit in SPDI_STATUS is to be set to logic '1'. Level trigger interrupt will be raised to the external interrupt controller if the corresponding enable bit is set to logic '1'. 0 = No effect
7	LOCK_SET	W	0	1 = LOCK bit in SPDI_STATUS is to be set to logic '1'. Level trigger interrupt will be raised to the external interrupt controller if the corresponding enable bit is set to logic '1'. 0 = No effect
6	VERR_SET	W	0	1 = VERR bit in SPDI_STATUS is to be set to logic '1'. Level trigger interrupt will be raised to the external interrupt controller if the corresponding enable bit is set to logic '1'. 0 = No effect
5	PERR_SET	W	0	1 = PERR bit in SPDI_STATUS is to be set to logic '1'. Level trigger interrupt will be raised to the external interrupt controller if the corresponding enable bit is set to logic '1'. 0 = No effect
4	OVR_SET	W	0	1 = OVERRUN bit in SPDI_STATUS is to be set to logic '1'. Level trigger interrupt will be raised to the external interrupt controller if the corresponding enable bit is set to logic '1'. 0 = No effect
3	HBE_SET	W	0	1 = HBE bit in SPDI_STATUS is to be set to logic '1'. Level trigger interrupt will be raised to the external interrupt controller if the corresponding enable bit is set to logic '1'. 0 = No effect
2	BUF1_ACTIVE_SET	W	0	1 = BUF1_ACTIVE bit in SPDI_STATUS is to be set to logic '1'. Level trigger interrupt will be raised to the external interrupt controller if the corresponding enable bit is set to logic '1'. 0 = No effect
1	BUF2_FULL_SET	W	0	1 = BUF2_FULL bit in SPDI_STATUS is to be set to logic '1'. Level trigger interrupt will be raised to the external interrupt controller if the corresponding enable bit is set to logic '1'. 0 = No effect
0	BUF1_FULL_SET	W	0	1 = BUF1_FULL bit in SPDI_STATUS is to be set to logic '1'. Level trigger interrupt will be raised to the external interrupt controller if the corresponding enable bit is set to logic '1'. 0 = No effect
Offset 0x10 AFF4	***SPDI_PWR_DWN***			
31:1	Unused		-	
0	PWR_DWN	R/W	0	The bit is used to provide power control status for system software block power management.
The PWR_DWN register is provided for software use only. The PWR_DWN bit has no functionality internal to the SPDI Input module. The bit is used to provide power control status for system software block power management. The default power on state of this bit is logic '0'.				
Offset 0x10 AFFC	***SPDI_MODULE_ID***			
31:16	Module ID	R	0x0110	This field identifies the block as type SPDIF Input. SPDIF Input ID = 0x0110.
15:12	MAJ_REV	R	0	Major Revision ID
11:8	MIN_REV	R	0x1	Minor Revision ID
7:0	APERTURE	R	0	Aperture size
The MODULE_ID register allows software identification of the SPDIF Input module. The values found in the MODULE_ID register will change with each version of the SPDIF Input module.				

Chapter 19: Memory Based Scaler

PNX15xx Series Data Book – Volume 1 of 1

Rev. 2 — 1 December 2004 Product data sheet

1. Introduction

Memory-based scaling is done independent of any video clocks by reading the video data from memory and writing the scaled pictures back to the memory. A single scaler can therefore be used to scale more than one video stream.

The memory-based scaler can perform the following operations:

- Vertical and horizontal scaling
  - Linear and non-linear aspect-ratio conversion
- De-interlacing
  - Simple median
  - Majority-selection (median filtering with previous field, spatial temporal average of 2 fields, same position from next or previous field depending on whether three or two field majority selection) [Note: majority selection is done on luma only]
  - Field insertion[1] and line doubling (*i.e.*, repeating the same line twice).... *not straightforward but doable with programming tricks*
  - Edge-Dependent De-interlacing (EDDI) --- a post-processing step done only on luma
- Anti-flicker filtering
- Conversions between 4:2:0, 4:2:2 and 4:4:4
- Indexed to true color conversion
- Color expansion/compression (different quantizations for color components, e.g., RGB565 -> RGB32)
- Deplanarization/planarization
- Variable color space conversion with programmable matrix coefficients (mutually exclusive with horizontal scaling)
- Color-key and alpha processing
  - Conversion between color-key and alpha
  - Alpha scaling

1. The current MBS implementation can not perform film-mode detection.

- Measurements
  - Histogram measurement
  - Noise estimation
  - Blackbar detection
  - Blacklevel measurement
  - UV bandwidth measurement
- Task-list based programming

Most of the above functions can be performed during a single pass, though the filter quality (length) may vary depending on the performed operations.

Special mode and features:

- Color key to alpha and alpha to color key conversion (color re-keying)
- Non-linear phase interpolation (phase LUT)

2. Functional Description

2.1 MBS Block Level Diagram

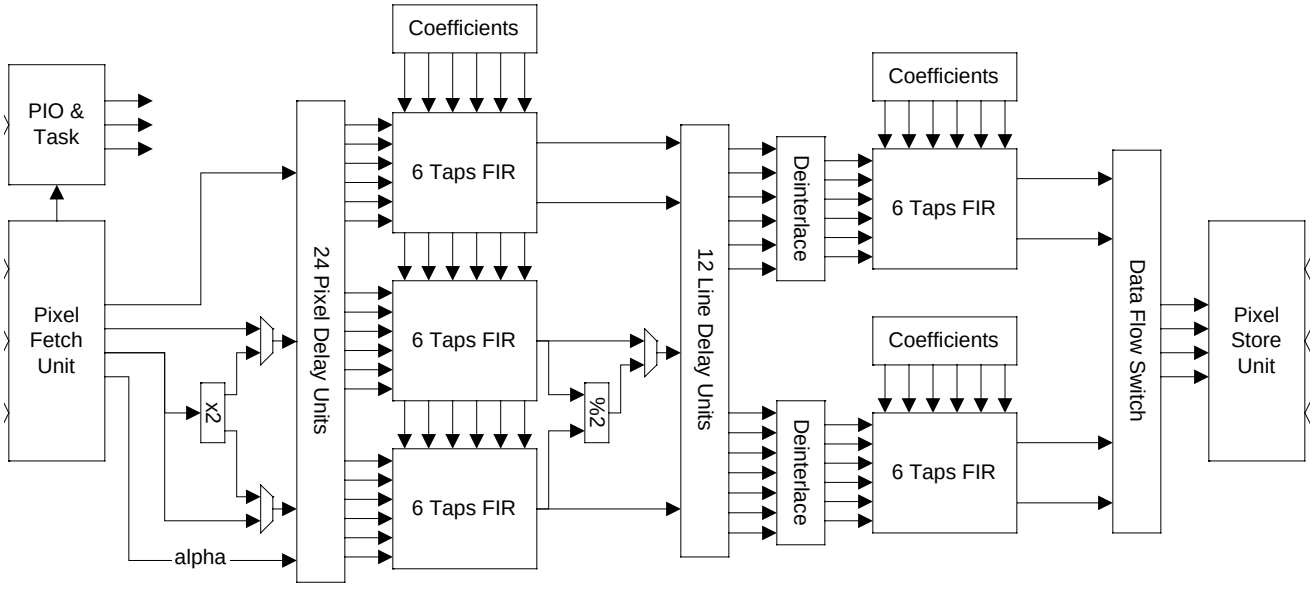


Figure 1: MBS Block Diagram

2.2 Data Flow

This section gives an idea of the processing functions in the MBS.

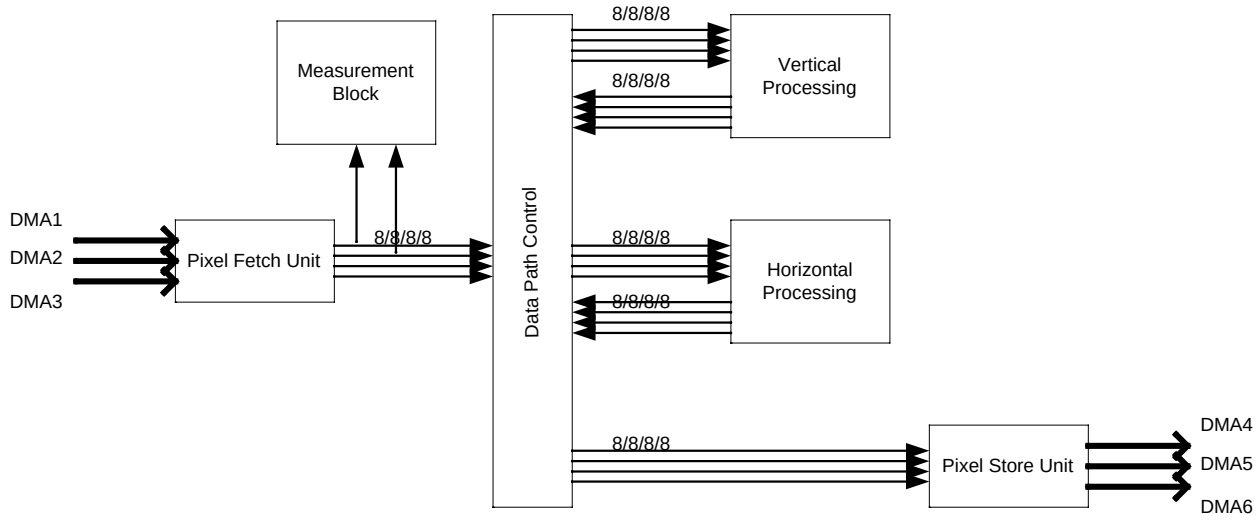


Figure 2: MBS Top Level

Figure 2 shows the top-level structure of MBS. There are two independent processing pipelines — Vertical processing Pipeline and Horizontal Processing Pipeline — that, for most applications, can be used in any order; for 3-field majority selection, however, vertical processing has to be done first.

2.2.1 Horizontal Processing Pipeline

Figure 3 elaborates on the Horizontal Processing Pipeline (HPP). HPP is responsible for chroma up-sampling, horizontal scaling, color-space conversion, and chroma down-sampling. Note that only one of either horizontal scaling or color-space conversion can be done at any one time., i.e., these operations are mutually exclusive. The scaler coefficients are the same for both chroma and luma.

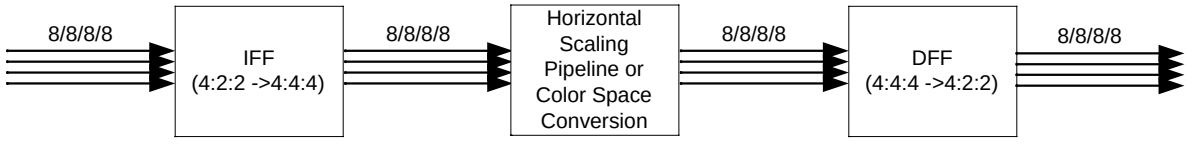


Figure 3: MBS Horizontal Processing Pipeline

2.2.2 Vertical Processing Pipeline

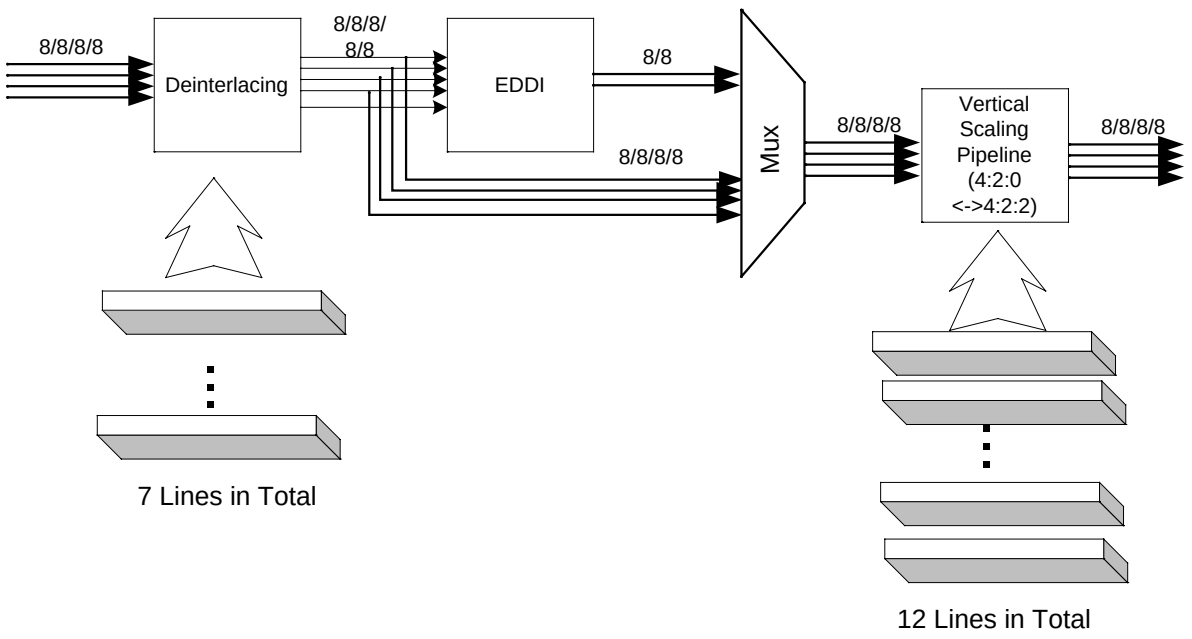


Figure 4: MBS Vertical Processing Pipeline

Figure 4 shows the Vertical Processing Pipeline (VPP). VPP comprises de-interlacing, EDDI-based post-processing, and vertical scaling (that also allows 4:2:0 <-> 4:2:2 sampling conversions for chroma). De-interlacing for luma samples uses majority-select or median filtering; de-interlacing for chroma always uses median filtering. For vertical scaling, there are 2 separate coefficient tables: one for luma and the other for chroma, thereby allowing the use of different filter coefficients for chroma and luma.

2.3 Data Processing in MBS

Table 1, Table 2, and Table 3 tabulate the various processing modes and orders in the MBS.

Table 1: Pipeline Processing (Horizontal First Mode)

Input	IFF[1]	Horizontal	DFF[2]	Vertical	Output
- 4:2:0 input (semi) planar - 4:2:2 input	up-sample to 4:4:4	- color space conversion (full matrix plus offset) or - scaling only (6 tap direct or transposed polyphase)	down-sample to 4:2:x	-de-interlacing[3] and/or - scaling (6 tap direct polyphase)	- 4:2:0 output (semi) planar - 4:2:2 output (packed, semi-planar, planar)
- 4:4:4 input (8-32 BPP) - LUT mode (8/4/2/1 BPP)[4]	bypass		bypass	- scaling only (4 tap direct polyphase) and/or - Anti Flicker (static 4 tap filter)	- 4:4:4 output (16-32 BPP) - 4:4:4:4 output (16-32 BPP)
- 4:4:4:4 input (16-32 BPP) - LUT mode (8/4/2/1 BPP)[5]	bypass	- scaling only (3 tap direct polyphase)	bypass	- scaling only (3 tap direct polyphase) and/or - Anti Flicker (static 3 tap filter)	

[1] Interpolation FIR Filter
[2] Decimation FIR Filter
[3] VTM (luma and chroma) or 2 field MSA (luma) with or without EDDI (see Table 3 for de-interlacing).
[4] Alpha component from LUT is not used.
[5] Alpha component from LUT is used.

Table 2: Pipeline Processing (Vertical First Mode)

Input	Vertical	IFF	Horizontal	DFF	Output
- 4:2:0 input - 4:2:2 input	- de-interlacing (see Table 3) and/or - scaling (6 tap direct polyphase)	up-sample to 4:4:4	- color space conversion (full matrix plus offset) or - scaling (6 tap direct or transposed polyphase)	down-sample to 4:2:x	- 4:2:0 output (2-3 planes) - 4:2:2 output (packed, semi-planar, planar)
- 4:4:4 input (8-32 BPP) - LUT mode (8/4/2/1 BPP)	- scaling (4 tap direct polyphase) and/or - Anti Flicker (static 4 tap filter)	bypass		bypass	- 4:4:4 output (16-32 BPP) - 4:4:4:4 output (16-32 BPP)
- 4:4:4:4 input (16-32 BPP) - LUT mode (8/4/2/1 BPP)	- scaling (3 tap direct polyphase) and/or - Anti Flicker (static 3 tap filter)	bypass	- scaling (3 tap direct polyphase)	bypass	

Table 3: De-Interlacing Mode Maximum Filter Lengths

Input Format	EDDI	MSA 2 Field (Y:UV Taps)	MSA 3 Field (Y:UV Taps)	Median (Y:UV Taps)
4:2:0 or 4:2:2 planar	Yes	6:6	Not supported	6:6
4:2:0 or 4:2:2 semi-planar	Yes	6:6	6:6[1]	6:6
4:2:2 single plane	No	Not supported	Not supported	6:6

[1]Only supported in Vertical-First mode

2.4 General Operations

This section provides the details on how the MBS functions. A description of each functional group is provided.

2.4.1 Task Control

Since the MBS is capable of processing several video streams in sequence, a pipelining mechanism is implemented for scaling a sequence of tasks. A task is described by a data structure stored in memory. Writing the base address of this task into the Task FIFO schedules the task to be executed after the completion (processing) of the previously-scheduled tasks.

In addition to the task list in the FIFO, each Task structure in memory can consist of a linked list of sub tasks that will be executed in sequence (*e.g.*, HD scaling task via partitioning). The software scheduling algorithm is responsible for preventing the Task FIFO from overflowing. An interrupt can be generated, once the last task in the FIFO gets executed, in order to request new tasks from the scheduler. Other interrupt events also exist and they aid in the task of keeping the task FIFO filled and avoiding overflow in the four available FIFO slots.

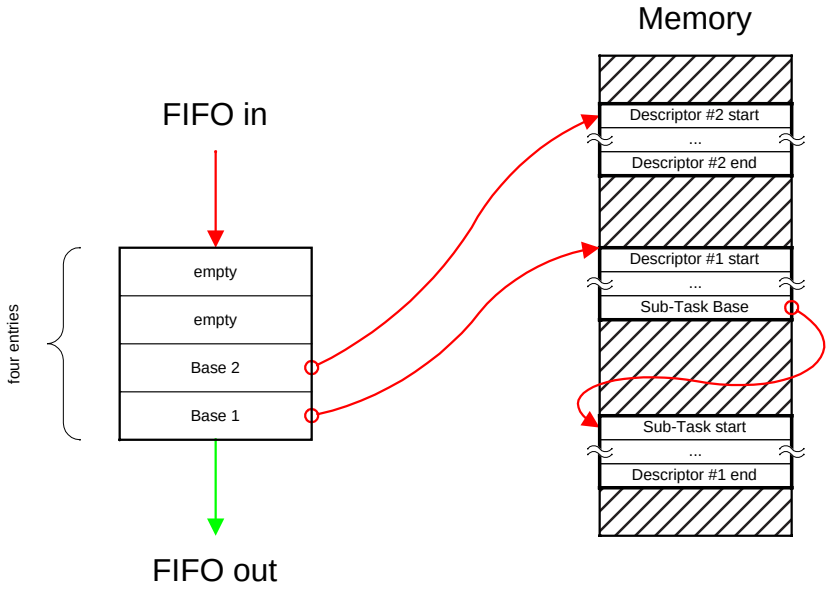


Figure 5: Task FIFO and Linked List

Table 4 shows the opcodes allowed in a descriptor list. The data structure consists of 32 bit words; therefore, endian mode rules do apply. The command type is defined in the lower two bits of the 32-bit word. The remaining bits are decoded depending on the command type.

Table 4: Task Descriptor Opcode Table

Command	Bits	Function	Description
Jump	Link to address		
	25:3	address	Defines location where processing of commands continues.
	1:0	opcode	= 00 (binary)
Exec/Stop	End of task list		
	2	stop flag	If this flag is set, processing of the MBS task is stopped.
	1:0	opcode	= 01 (binary)
Queue	Start processing and queue next task		
	25:3	address	Task at this location is started after processing of current task finished.
	1:0	opcode	= 10 (binary)
Load	Load register		
	31:24	count	Define number of registers to be loaded minus one.
	19:16	mask	Write mask, if bit is zero according byte is written, if bit is set byte is not written.
	11:2	Index	Load registers starting at offset.
	1:0	opcode	= 11 (binary)

2.4.2 Video Source Controls

Source Video data for the MBS can be fetched via three dedicated DMA engines. It can be fetched from memory in several packed or planar formats. The source window is defined by one set of width and height values which are automatically translated into the corresponding number of 64-bit words to be fetched per line for each plane. Video lines can be fetched in a reverse order as well, thereby allowing horizontal flipping of images for applications like video conferencing.

To allow fetching of interlaced video lines from different locations in memory, each plane can be assigned a second set of base address registers. Pitches are defined separately only for luma and chroma planes. The lower three bits of the first three base address registers are used as an intra-long-word offset for the leftmost pixel components of each line. The offset has to be a multiple of the number of bytes per component.

Fixed Input Formats

The fixed input formats, as shown in Table 5, consist of indexed, packed and planar modes.

Table 5: Input Pixel Formats

Format	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
1 bit indexed	I
2 bit indexed	2
4 bit indexed	4 bit
8 bit indexed	8 bit Index
YUV 4:4:4, planar YUV 4:2:x, planar RGB 4:4:4, planar	plane #1 plane #2 plane #3
	Y8 or R8
	U8 or G8
	V8 or B8
YUV 4:2:x, semi planar	plane #1 plane #2
	Y8
	V8 / U8
packed YUY2 4:2:2	U8 or V8 / Y8
packed UYVY 4:2:2	Y8 / U8 or V8
packed variable 16 bit	variable YUV or RGB 4:4:4
packed variable 32 bit	variable YUV 4:4:4 , 4:2:2 or RGB 4:4:4

For indexed formats, sizes of 1, 2, 4, or 8 bits per pixel can be selected. A 32-bit color look up table is used to expand indexed modes into true color, including alpha values.

Predefined packed modes are available for YUY2 and UYVY formats. Other packed modes can be selected by using one of the three variable input formats described below.

Planar formats can be used to fetch pixel components from different locations in memory. In semi-planar modes, a video field is defined by one plane for Y samples and one shared plane for U/V. In full planar formats, each component is fetched from a separate plane.

When processing a 4:2:2 or 4:2:0 input stream, the first sample fetched from memory has to be a Y/U pair!

Variable Input Formats

In addition to the predefined fixed input pixel formats above, the MBS also includes three variable input format modes for packed 4:4:4 and 4:2:2 pixel extraction. In these modes, the pixel components can be extracted from programmable locations within 32- or 16-bit units.

2.4.3 Horizontal Video Filters

Interpolation Filter

All horizontal video processing is based on equidistant sampled components. All 4:2:2 video streams, therefore, have to be up-sampled before being scaled horizontally. The interpolation FIR filter used can interpolate interspersed or co-sited chroma samples. Mirroring of samples at the field boundaries compensate for run in and run out conditions of the filter.

Decimation Filter

After horizontal processing, chrominance may be down sampled to reduce memory bandwidth or allow for higher quality vertical processing that is not available otherwise. Mirroring of samples at the field boundaries compensate for run in and run out conditions of the filter.

Normal Polyphase Filter Mode

The normal polyphase filter can be used to zoom out or downscale a video image. Depending on the number of components, the filter uses 6 taps (three-component mode) or 3 taps (four-component mode). For the normal polyphase filter, run-in and run-out behavior are compensated by appropriate mirroring of samples at the field boundaries. In order to align the first output pixel with the geometric location of the first input pixel, the start phase has to be chosen as:

$$\text{dto\_offset} = 0x200 \cdot \text{filter\_length} \quad (17)$$

Transposed Polyphase Filter Mode

The transposed polyphase filter can only be used to downscale. By reversing the flow through the normal polyphase filter, the transposed filter takes the output pixels as reference and accumulates input pixels. The advantage of the transposed polyphase filter is the longer effective filter length which allows for filtering out high frequencies that interfere with downscaling. A drawback of this approach is that the filter coefficients have to get optimized for the scaling ratio in order to avoid visible DC ripples. Compensation for run-in and run-out behavior of the filter is not available. Use of the transposed filter is limited to three components only.

The scaling ratio for the transposed polyphase *HSP_ZOOM_0* = 0x1 0000h * *zoom_x*, with *zoom_x* = *pixel_out* / *pixel_in*

Linear Phase Interpolation Mode (LPI)

In the linear phase interpolation mode, samples between two pixels are interpolated by using the linear equation px = (1-phase) * x_{left} + phase * x_{right}. Linear phase interpolation creates adequate results especially when used with computer generated graphics which, unlike motion video, is usually not bandwidth limited.

The LPI mode is enabled by setting HSP_PHASE_MODE = 7. Horizontal processing mode has to be set to normal polyphase (HSP_MODE = 2) and four-component mode has to be enabled (HSP_FIR_COMP = 1)

Color Space Matrix Mode

In addition to the normal and transposed polyphase filtering (scaling), the FIR filter structure can instead be programed to perform color-space-conversion functions. A dedicated set of registers holds the coefficients for the color space matrix.

2.4.4 Vertical Video Filters

The polyphase filter used for vertical processing can only be operated in the normal polyphase mode.This filter can be used as a two-component 6-tap filter, a three-component 4-tap filter or a four-component 3-tap filter (optionally, linear phase interpolation). In the two-component mode, each filter unit can be run independently, to allow 4:2:0 to 4:2:2 conversion or vice versa.

2.4.5 De-Interlacing in MBS

The current de-interlacing algorithms, implemented inside the MBS, are as follows:

- Simple Median Filtering (Vertical Temporal Median -- VTM)
- Majority Selection Algorithm (MSA)
- Field Insertion and Line Doubling

MBS supports both field insertion and line doubling based simple de-interlacing in the pixel-fetch unit.

If de-interlacing is enabled, the previous and current fields have to be fed into the scaler as one interleaved frame. For the MSA, the first line has to originate in the previous field, while in VTM, the first line has to be fetched from the current field. Base Address Mode = 1 can be selected to assist in weaving the two fields together into one frame.

Edge Dependent De-Interlacing (EDDI)

The purpose of EDDI is to improve the appearance of long diagonal edges, without degrading other parts of the image. EDDI is used as a post-processing unit for the de-interlacer.

2.4.6 Color-Key Processing

Color-key processing, to convert alpha values into color key values and/or vice versa, are only available for the 4:4:4 video formats.

- **Color Key to Alpha Conversion**

  To avoid color key values getting altered during video processing, color-key matching samples can be tagged as *transparent* by generating an alpha value of zero at the input of the MBS. Non-keyed samples get assigned a programmable alpha value. After video processing is over, the alpha value can be stored in memory together with the other components or can be used to re-key the keying color back into the video stream (see Alpha to Color Key Convert below**)**

- **Color Key Replace**

  Scaling of video streams containing color keys can lead to artifacts at the corners between keyed and non-keyed samples, due to the “smearing effect of the low pass filter. The effect is heightened by the highly saturated colors normally used for keying. To avoid these artifacts, color keyed samples are replaced with either gray, black, or the previous sample (gray if at the start of the line),.

- **Alpha to Color Key Convert**

If the alpha value, after horizontal and vertical processing, is below a fixed threshold (0x80), the sample is replaced by the color key.

- **Fixed Alpha Insert**

  The alpha value defined in the Color Key register is inserted, as the fourth component after horizontal and vertical processing, in the PSU unit (if CKEY_K2A=1).

2.4.7 Alpha Processing

Alpha processing is only available when the horizontal and vertical filter blocks are either bypassed or operated in the four-component mode. If the horizontal filter is used for color space conversion, the alpha information is kept time-aligned with the other three components.

2.4.8 Video Data Output

After processing the video stream, the MBS can split the video data into one or more (up to three) different streams. For each stream, there is a memory base address. There are two line-pitch registers further defining the DMA streams. The number of streams (planes) is defined by the output-format register. Depending on its value, the video components get packed into 64-bit words. These words then get buffered and transferred to the external memory in more effective clusters. A list of the supported output video formats is shown in Table 6. Packing of a pixel into 64-bit units is always done from right to left, while bytes within one pixel unit are ordered according to the Endian settings (according to the global Endian setting, the Endian bit in the output format register can invert the setting).

Table 6: Output Pixel Formats

Format	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
planar YUV or RGB (4:4:4, 4:2:2 or 4:2:0)	plane #1 plane #2 plane #3		Y8 or R8 U8 or G8 V8 or B8
semi planar YUV (4:2:2 or 4:2:0)	plane #1 plane #2		Y8 or R8 U8
packed 4/4/4 RGBa		alpha R4	G4 B4
packed 4/5/3 RGBa		alpha R4	G5 B3
packed 5/6/5 RGB		R5 G6	B5
packed YUY2 4:2:2		U8 or V8	Y8
packed UYVY 4:2:2		Y8	U8 or V8
packed 888 RGB(a)	(alpha) R8 or Y8	G8 or U8	B8 or V8
packed 4:4:4 VYU(a)	(alpha) V8	Y8	U8

Remark: Table 6 shows location of the first 'pixel unit' within a 64 bit word in the little endian mode. The selected endian mode will affect the position of the components within multi-byte pixel units!

2.4.9 Address Generation

Each of the three video planes is assigned a set of two base address registers and two line address pointers. Depending on the base address mode, video data corresponding to each plane is written using one pointer, or using both pointers alternating on each line. At the end of the line, the pitch value is added to the active line address pointer. The pitch defines the difference in the address of two vertically adjacent pixels. The pitch values are defined separately for chroma and luma components only. The lower three bits of the first three base address registers are used as an intra-long-word offset for the leftmost pixel components of each line. The offset has to be a multiple of the number of bytes per component.

2.4.10 Interrupt Generation

The following interrupt events are defined:

- **TASK_ERROR**

  Current processing task leads to a pipeline error, MBS has to be reset to resume with new scaling task(s).

- **TASK_END**

  Current task processing is done, but some data might still remain in the output FIFO - this Interrupt denotes the point when the MBS is ready to start the processing of a new task.

- **TASK_OVERFLOW**

  Task fifo overflow - task request written into Task- fifo-register got ignored.

- **TASK_IDLE**

  Task finished and the task fifo is empty.

- **TASK_EMPTY**

  Task fifo runs empty - generation of this interrupt requires that there was a pending task in the task fifo. This interrupt will not be generated if tasks are only submitted in the MBS idle state.

- **TASK_DONE**

  Current task finished and all writes complete - on reception of this interrupt, all data will be accessible in the main memory.

2.4.11 Measurement Functions

All measurements in the MBS are based on analyzing a video stream, within a programmable window, using specific algorithms. The measurement results are stored in registers once per field/frame. A special interrupt is generated to indicate that the measurement is done. Based on the measurement results, software sets the control parameters for the picture processing units.

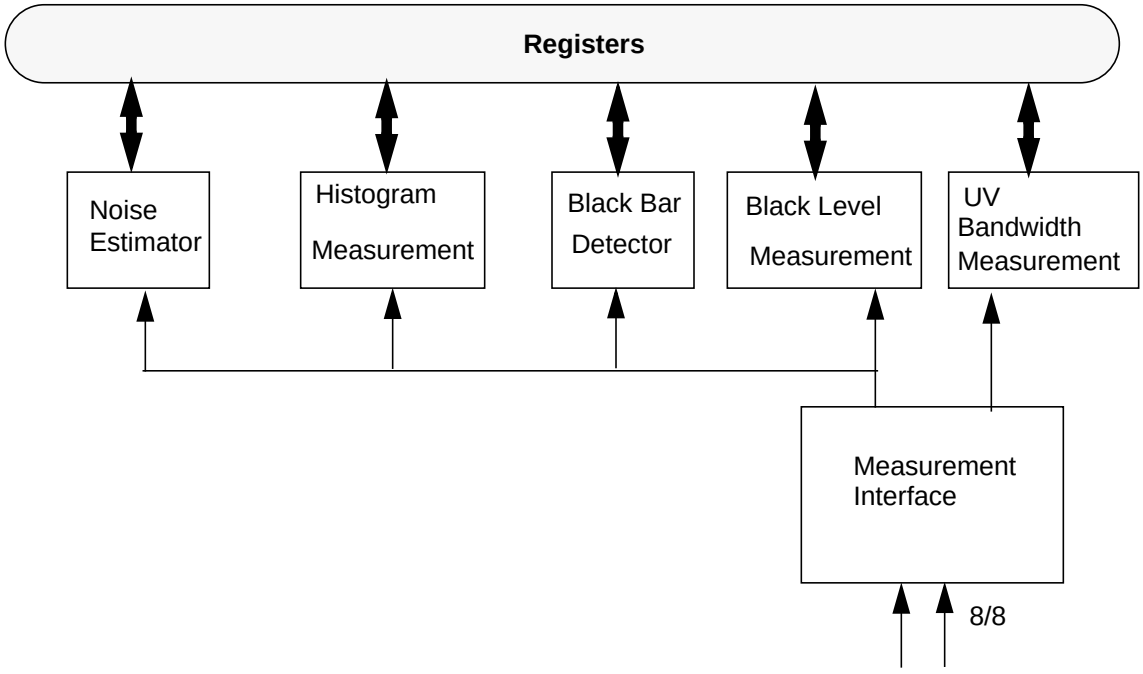


Figure 6: Measurement in the MBS

As shown in Figure 6, the MBS measurement block comprises:

- Measurement Block Interface

  Converts the inputs format (Y, U/V 8bits) into the format needed by the measurement units (Y, U/V 9bits).

- Histogram Measurement

  Analyses the Y component of the input data stream. Used for dynamic contrast improvement (i.e., histogram modification in QVCP).

- Noise Estimator

  Analyses the Y component. Controls the setting of LSHR pool ressource in QVCP.

- Black Bar Detector

  Analyses the Y component. Results are used to expand the picture to the display size in case of black bars at the top and/or bottom of the picture.

- Black Level Measurement

  Analyses the Y component. Used for black stretch.

- UV Bandwidth Measurement

  Analyses the UV component. Controls the DCTI setting in QVCP.

Remark: MBS and QTNR both use the same measurement functions/block.

3. Register Descriptions

3.1 Register Summary

Table 7: Register Summary

Offset	Name	Description
0x10 C000	MBS_MODE	MBS operation mode
0x10 C040	TASK_QUEUE	Task queue FIFO
0x10 C044	TASK_STATUS 1	Task queue status
0x10 C048	TASK_STATUS 2	Shows last command executed
0x10 C100	PFU_FORMAT	Input format and mode
0x10 C104	PFU_WINDOW	Source window size
0x10 C108	PFU_VARFORM	Variable source format
0x10 C140	PFU_BASE1	Source base address DMA #1
0x10 C144	PFU_PITCH1	Source line pitch component 1
0x10 C148	PFU_BASE2	Source base address DMA #2
0x10 C14C	PFU_PITCH2	Source line pitch component 2 and 3
0x10 C150	PFU_BASE3	Source base address DMA #3
0x10 C154	PFU_BASE4	Source base address DMA #4
0x10 C158	PFU_BASE5	Source base address DMA #5
0x10 C15C	PFU_BASE6	Source base address DMA #6
0x10 C200	HSP_ZOOM_0	Initial zoom for 1st pixel in line (unsigned)
0x10 C204	HSP_PHASE	Horizontal phase control
0x10 C208	HSP_DZOOM_0	Initial zoom delta for 1 pixel in line (signed)
0x10 C20C	HSP_DDZOOM	Zoom delta change (signed)
0x10 C220	CSM_COEFF0	Color space matrix coefficients C_{00} - C_{02}
0x10 C224	CSM_COEFF1	Color space matrix coefficients C_{10} - C_{12}
0x10 C228	CSM_COEFF2	Color space matrix coefficients C_{20} - C_{22}
0x10 C22C	CSM_OFFS1	Color space matrix offset coefficients D_0-D_2
0x10 C230	CSM_OFFS2	Color space matrix rounding coefficients E_0-E_2
0x10 C240	VSP_ZOOM_0	Initial zoom for 1st pixel in line (unsigned)
0x10 C244	VSP_PHASE	Vertical phase control
0x10 C248	VSP_DZOOM_0	Initial zoom delta for 1 pixel in line (signed)
0x10 C24C	VSP_DDZOOM	Zoom delta change (signed)
0x10 C260	EDDI_CTRL1	EDDI Control Register 1

Table 7: Register Summary *...Continued*

Offset	Name	Description
0x10 C264	EDDI_CTRL2	EDDI Control Register 2
0x10 C280	CKEY_MASK	Color key and alpha control
0x10 C284	CKEY_VALUE	Color key and alpha components
0x10 C300	PSU_FORMAT	Output format and mode
0x10 C304	PSU_WINDOW	Target window size
0x10 C340	PSU_BASE1	Target base address DMA #1
0x10 C344	PSU_PITCH1	Target line pitch component 1
0x10 C348	PSU_BASE2	Target base address DMA #2
0x10 C34C	PSU_PITCH23	Target line pitch component 2 and 3
0x10 C350	PSU_BASE3	Target base address DMA #3
0x10 C354	PSU_BASE4	Target base address DMA #4
0x10 C358	PSU_BASE5	Target base address DMA #5
0x10 C35C	PSU_BASE6	Target base address DMA #6
0x10 C400—C7FC	COLOR_TABLE	Color look-up table
0x10 C800—C9FC	COEFF_TABLE1	Coefficient table for horizontal filter (0-5)
0x10 CA00—CDFC	COEFF_TABLE2	Coefficient table for vertical luma filter (0-5)
0x10 CC00—CDFC	COEFF_TABLE3	Coefficient table for vertical chroma filter (0-5)
0x10 CE00	FORMAT_CTRL	Formatter control
0x10 CE0C	FLAG_CTRL	Measurement finish mask
0x10 CE10	HISTO_CTRL	Histogram control
0x10 CE18	HISTO_WIN_START	Histogram window start
0x10 CE1C	HISTO_WIN_END	Histogram window end
0x10 CE20	NEST_CTRL1	Noise estimation control 1
0x10 CE24	NEST_CTRL2	Noise estimation control 2
0x10 CE28	NEST_HWIN	Noise estimation window start
0x10 CE2C	NEST_VWIN	Noise estimation window end
0x10 CE30	BBD_CTRL	Black bar detection control
0x10 CE38	BBD_WIN_START	Black bar detection window start
0x10 CE3C	BBD_WIN_END	Black bar detection window end
0x10 CE40	BLD_CTRL	Black level detection control
0x10 CE48	BLD_WIN_START	Black level window start
0x10 CE4C	BLD_WIN_END	Black level window end
0x10 CE50	BWD_CTR /DATA_1	UV Bandwidth detection control
0x10 CE58	BWD_WIN_START	UV Bandwidth window start
0x10 CE5C	BWD_WIN_END	UV Bandwidth window end

Table 7: Register Summary *...Continued*

Offset	Name	Description
0x10 CE8F	HISTO_DATA_1	Histogram status
0x10 CE90	HISTO_DATA_2	Histogram data output
0x10 CE94	HISTO_DATA_3	Histogram data output
0x10 CE98	HISTO_DATA_4	Histogram data output
0x10 CE9C	HISTO_DATA_5	Histogram data output
0x10 CEA0	HISTO_DATA_6	Histogram data output
0x10 CEA4	HISTO_DATA_7	Histogram data output
0x10 CEA8	HISTO_DATA_8	Histogram data output
0x10 CEAC	HISTO_DATA_9	Histogram data output
0x10 CEB0	NEST_DATA_1	Noise estimation status 1
0x10 CEB4	NEST_DATA_2	Noise estimation status 2
0x10 CEB8	BBD_DATA_1	Black bar detection status 1
0x10 CEBC	BBD_DATA_2	Black bar detection status 2
0x10 CEC0	BLD_DATA	Black level detection status
0x10 CEC4	BWD_DATA_1	UV Bandwidth detection status 1
0x10 CEC8	BWD_DATA_2	UV Bandwidth detection status 2
0x10 CFE0	INT_STATUS	Interrupt status register
0x10 CFE4	INT_ENABLE	Interrupt enable register
0x10 CFE8	INT_CLEAR	Interrupt clear register
0x10 CFEC	INT_SET	Interrupt set register
0x10 CFFC	MODULE_ID	Module Identification and revision information

3.2 Register Table

Table 8: Memory Based Scaler (MBS) Registers

Bit	Symbol	Access	Value	Description
Operating Mode Control Registers				
Offset 0x10 C000	***MBS Mode Control***			
31:30	DPM_SEQ	R/W	0	Data path sequence 0x: bypass all filter stages (format conversion only) 10: HSP -> VSP 11: VSP -> HSP
29	ALT_MSA3	R/W	0	1: chose alternate MSA3 mode
28	PREVIOUS_FIRST_C	R/W	0	1: previous field first (chroma)
27	PREVIOUS_FIRST_L	R/W	0	1: previous field first (luma)
26	ENABLE_EDDI	R/W	0	1: enable EDDI

Table 8: Memory Based Scaler (MBS) Registers *...Continued*

Bit	Symbol	Access	Value	Description
25:24	VSP_DE-INTERLACE	R/W	0	De-interlacing mode 00: no de-interlacing 01: vertical temporal median 10: majority selection algorithm (2 field) 11: majority selection algorithm (3 field)
23:22	Reserved			
21:20	COEFF_LUT_MODE	R/W	0	Coefficient look-up table access mode 00: each table gets written separately 01: writes to coeff table 2 are copied into table 3 10: writes to coeff table 1 are copied into table 2 11: writes to coeff table 1 are copied into table 2 and 3
19	DISABLE_IRQ	R/W	0	1: disables the MBS interrupts- task done and task idle. Used for concatenated tasks.
18	NO_AUTO_SKIP	R/W	0	Disable Auto Skip When set to zero (default) the MBS will automatically skip all remaining operations within a task once the TASK_DONE interrupt was generated. If enable the MBS will continue until all pipeline stages are idle.
17	SKIP_TASK	W	0	Skip current task Writing a one into this bit will reset the current task and continue with previously scheduled tasks
16	SOFT_RESET	W	0	Soft reset Writing a one into this bit will reset the block and all outstanding scaling tasks
15	Reserved			
14	IFF_CLAMP	R/W	0	Clamp mode for IFF (affects U/V only) 0: clamp to 0-255 1: clamp to 16 - 240 (CCIR range)
13:12	IFF_MODE	R/W	0	Interpolation mode 00: bypass 01: reserved 10: co-sited 11: interspersed
11	Reserved			
10	DFF_CLAMP	R/W	0	Clamp mode for DFF (affects U/V only) 0: clamp to 0-255 1: clamp to 16 - 240 (CCIR range)
9: 8	DFF_MODE	R/W	0	Decimation mode 00: bypass 01: co-sited (sub sample) 10: co-sited (low pass) 11: interspersed
7	VSP_CLAMP	R/W	0	Clamp mode for VSP 0: clamp to 0-255 1: clamp to CCIR range defined by VSP_RGB (bit 6)

Table 8: Memory Based Scaler (MBS) Registers *...Continued*

Bit	Symbol	Access	Value	Description
6	VSP_RGB	R/W	0	Color space mode, defines CCIR clamping range for VSP 0: processing in YUV color space (CCIR range: 16 - 235(Y), 16 - 240(U/V)) 1: processing in RGB color space (CCIR range: 16 - 235)
5:4	VSP_MODE	R/W	0	Vertical processing mode 00: bypass mode 01: reserved 10: normal polyphase mode 11: reserved
3	HSP_CLAMP	R/W	0	Clamp mode for HSP 0: clamp to 0-255 1: clamp to CCIR range defined by HSP_RGB (bit 2)
2	HSP_RGB	R/W	0	Color space mode, defines CCIR clamping range for HSP 0: processing in YUV color space (CCIR range: 16 - 235(Y), 16 - 240(U/V)) 1: processing in RGB color space (CCIR range: 16 - 235)
1:0	HSP_MODE		0	Horizontal processing mode 00: bypass mode 01: color space matrix mode 10: normal polyphase mode 11: transposed polyphase mode
Video Informations Registers				
Offset 0x10 C040	***Task FIFO***			
31:3	TASK_BASE	W		Scale task FIFO Must be aligned to 8 byte boundary
2	Reserved		0	
1:0	TASK_CMD	W		Command mode 00: process task descriptor at given base 01: start scaling with current register settings 1x: reserved
Offset 0x10 C044	***Task Status 1***			
31:4	Reserved		0	
3:0	TASK_PENDING	R		Number of pending scaling tasks (including current)
Offset 0x10 C048	***Task Status 2***			
31:0	LAST_COMMAND	R		Last Command executed
Input Format Control Registers				
Offset 0x10 C100	***Input Format***			

Table 8: Memory Based Scaler (MBS) Registers *...Continued*

Bit	Symbol	Access	Value	Description
31:30	PFU_BAMODE	R/W	0	Base address mode 00 = single set (e.g. progressive video source) base 1-3 according to number of planes (plane 1-3) 01 = alternate sets each line (e.g. interlaced video source) base 1-3, first line in, etc. (plane 1-3) base 4-6, second line in, etc. (plane 1-3) 1x = reserved
29:16	Reserved			
15	PFU_HFLIP	R/W		Mirror input mode 0: normal 1: mirrored input
14	Reserved	R/W		
13	PFU_ENDIAN	R/W		Input format endian mode 0: same as system endian mode 1: opposite of system endian mode
12:8	Reserved			
7:0	PFU_IPFMT	R/W	0	Input formats 00 (hex) = YUV 4:2:0, semi-planar 03 (hex) = YUV 4:2:0, planar 08 (hex) = YUV 4:2:2, semi-planar 0B (hex) = YUV 4:2:2, planar 0F (hex) = RGB or YUV 4:4:4, planar 24 (hex) = 1-bit indexed 45 (hex) = 2-bit indexed 66 (hex) = 4-bit indexed 87 (hex) = 8-bit indexed A0 (hex) = packed YUY2 4:2:2 A1 (hex) = packed UYVY 4:2:2 AC (hex) = 16-bit variable contents 4:4:4 E8 (hex) = 32-bit variable contents 4:2:2 EC (hex) = 32-bit variable contents 4:4:4
Offset 0x10 C104	***Source Window Size***			
31:27	Reserved			
26:16	PFU_LSIZE	R/W	0	Line size Defines size of input window 1 = one pixel **Remark:** Internal buffer lines are only 1024 pixels. Horizontal and vertical scaling need to be ordered such that the intermediate result fits into 1024 pixels buffer lines.
15:11	Reserved			
10:0	PFU_LCOUNT	R/W	0	Line count Defines size of input window 1 = one line
Offset 0x10 C108	***Variable Format Register***			
31:29	PFU_SIZE_C4 [2:0]	R/W	0	Size component #4 (alpha or V) Number of bits minus 1 (e.g. 7 = 8 bits per component)

Table 8: Memory Based Scaler (MBS) Registers *...Continued*

Bit	Symbol	Access	Value	Description
28:24	PFU_OFFS_C4 [4:0]	R/W	0	Offset component #4 (alpha or V) Index of MSB position within 32 bit word (0-31)
23:21	PFU_SIZE_C3 [2:0]	R/W	0	Size component #3 (V or B or Y2) number of bits minus 1 (e.g. 7 = 8 bits per component)
20:16	PFU_OFFS_C3 [4:0]	R/W	0	Offset component #3 (V or B or Y2) index of MSB position within 32 bit word (0-31)
15:13	PFU_SIZE_C2[2:0]	R/W	0	Size component #2 (U or G) number of bits minus 1 (e.g. 7 = 8 bits per component)
12:8	PFU_OFFS_C2[4:0]	R/W	0	Offset component #2 (U or G) index of MSB position within 32 bit word (0-31)
7:5	PFU_SIZE_C1[2:0]	R/W	0	Size component #1 (Y or R) number of bits minus 1 (e.g. 7 = 8 bits per component)
4:0	PFU_OFFS_C1[4:0]	R/W	0	Offset component #1 (Y or R) index of MSB position within 32 bit word (0-31)
Video Input Address Generation Control Registers				
Offset 0x10 C140	***Source Base Address #1***			
31:28	Unused		-	
27: 3	PFU_BASE1	R/W		Base address DMA #1 used depending on PFU_BAMODE setting
2:0	PFU_OFFSET1	R/W		Base address byte offset DMA #1 bits define pixel offset within multi pixel 64 bit words (e.g. a 16bit pixel can be placed on any 16 bit boundary)
Offset 0x10 C144	***Source Line Pitch #1***			
31:15	Unused		-	
14: 3	PFU_PITCH1	R/W	0	Line pitch DMA #1, signed value (two's complement) Used for all packed formats and for plane 1.
2:0	Unused		-	
Offset 0x10 C148	***Source Base Address #2***			
31:28	Unused		-	
27: 3	PFU_BASE2	R/W		Base address DMA #2 Used depending on PFU_BAMODE setting.
2:0	PFU_OFFSET2	R/W		Base address byte offset DMA #2 Bits define pixel offset within multi pixel 64-bit words (e.g., a 16-bit pixel can be placed on any 16-bit boundary).
Offset 0x10 C14C	***Source Line Pitch #2***			
31:15	Unused		-	
14: 3	PFU_PITCH2	R/W		Line pitch DMA #2, signed value (two's complement) Used for planes 2 and 3.
2:0	Unused		-	
Offset 0x10 C150	***Source Base Address #3***			
31:28	Unused		-	
27: 3	PFU_BASE3	R/W		Base address DMA #3 Used depending on PFU_BAMODE setting.

Table 8: Memory Based Scaler (MBS) Registers *...Continued*

Bit	Symbol	Access	Value	Description
2:0	PFU_OFFSET3	R/W		Base address byte offset DMA #3 Bits define pixel offset within multi pixel 64-bit words (e.g., a 16-bit pixel can be placed on any 16-bit boundary).
Offset 0x10 C154	***Source Base Address #4***			
31:28	Unused		-	
27: 3	PFU_BASE4	R/W		Base address DMA #4 Used depending on PFU_BAMODE setting.
2:0	Unused		-	
Offset 0x10 C158	***Source Base Address #5***			
31:28	Unused		-	
27: 3	PFU_BASE5	R/W		Base address DMA #5 Used depending on PFU_BAMODE setting.
2:0	Unused		-	
Offset 0x10 C15C	***Source Base Address #6***			
31:28	Unused		-	
27: 3	PFU_BASE6	R/W		Base address DMA #6 Used depending on PFU_BAMODE setting.
2:0	Unused		-	
Horizontal Video Processing Control Registers				
Offset 0x10 C200	***Initial Zoom***			
31:29	HSP_PHASE_MODE[2:0]	R/W	0	Phase mode 0: 64 phases 1: 32 phases 2: 16 phases 3: 8 phases 4: 4 phases 5: 2 phases 6: fixed phase 7: linear phase interpolation (only valid for 4 component mode)
28	HSP_NO_CROP	R/W	0	Disable line length cropping 0: cropping enabled (default) 1: cropping disabled, used for striped scaling tasks
27	HSP_UV_SEQ	R/W	0	Chroma sample re-sequence 0: normal sequence (default) 1: skip first chroma sample (Used for striped scaling tasks in 4:2:x transposed mode if stripe would start on an odd pixel location.)
26	HSP_FIR_COMP[1:0]	R/W		Horizontal filter components 0: three components, 6 tap FIR each 1: four components, 3 tap FIR each In color space matrix mode this value has to remain zero.
25:20	Reserved			

Table 8: Memory Based Scaler (MBS) Registers *...Continued*

Bit	Symbol	Access	Value	Description
19: 0	HSP_ZOOM_0[19:0]	R/W	0	Initial zoom for 1st pixel in line (unsigned, LSB = 2^{-16}) 2 0000 (hex): downscale 50% 1 0000 (hex): no scaling = 100% 0 8000 (hex): zoom 2 x (transposed: downscale 50%)
Offset 0x10 C204	***Phase Control***			
31	Reserved		-	
30:28	HSP_QSHIFT[2:0]	R/W	0	Quantization shift control Used to change quantization before being multiplied with HSP_MULTIPLY. 100 (bin): divide by 16 101 (bin): divide by 8 110 (bin): divide by 4 111 (bin): divide by 2 000 (bin): multiply by 1 001 (bin): multiply by 2 010 (bin): multiply by 4 011 (bin): multiply by 8 Warning: A value range overflow caused by an improper quantization shift can not be compensated later by multiplying with a HSP_MULTIPLY value below 0.5!
27:26	Reserved		-	
25	HSP_QSIGN	R/W	0	Quantization sign bit
24:16	HSP_QMULTIPLY[8:0]	R/W	0	Quantization multiply control Used to compensate for different weight sums in transposed polyphase or color space matrix mode, remaining bits are fractions (largest number is 511/512) Value range: $0 \leq m < 1.0$. Instead of using values in the range of $m < 0.5$ the quantization shift HSP_QSHIFT should be modified to gain more precision in the truncated result.
15:13	Reserved		-	
12: 0	HSP_OFFSET_0	R/W	0	Initial start offset for DTO
Offset 0x10 C208	***Initial Zoom delta***			
31:26	Reserved			
25: 0	HSP_DZOOM_0[25:0]	R/W	0	Initial zoom delta for 1 pixel in line (signed, LSB = 2^{-27}) Used for non constant scaling ratios.
Offset 0x10 C20C	***Zoom delta change***			
31:29	Reserved		-	
28: 0	HSP_DDZOOM[28:0]	R/W	0	Zoom delta change (signed, LSB = 2^{-40}) used for non constant scaling ratios
Color Space Matrix Registers				
Offset 0x10 C220	***Color space matrix coefficients C_{00} - C_{02}***			
31:30	Unused		-	
29:20	CSM_C02[9:0]	R/W	0	Coefficient C02, two's complement
19:10	CSM_C01[9:0]	R/W	0	Coefficient C01, two's complement
9:0	CSM_C00[9:0]	R/W	0	Coefficient C00, two's complement

Table 8: Memory Based Scaler (MBS) Registers *...Continued*

Bit	Symbol	Access	Value	Description
Offset 0x10 C224	***Color space matrix coefficients C_{10} - C_{12}***			
31:30	Unused		-	
29:20	CSM_C12[9:0]	R/W	0	Coefficient C12, two's complement
19:10	CSM_C11[9:0]	R/W	0	Coefficient C11, two's complement
9:0	CSM_C10[9:0]	R/W	0	Coefficient C10, two's complement
Offset 0x10 C228	***Color space matrix coefficients C_{20} - C_{22}***			
31:30	Unused		-	
29:20	CSM_C22[9:0]	R/W	0	Coefficient C22, two's complement
19:10	CSM_C21[9:0]	R/W	0	Coefficient C21, two's complement
9:0	CSM_C20[9:0]	R/W	0	Coefficient C20, two's complement
Offset 0x10 C22C	***Color space matrix offset coefficients D_0 - D_2***			
31:29	Unused		-	
28	CSM_D2_TWOS	R/W	0	Offset coefficient D_2 type 0 = unsigned 1 = signed
27:20	CSM_D2[7:0]	R/W	0	Offset coefficient D_2
19	Unused		-	
18	CSM_D1_TWOS	R/W	0	Offset coefficient D_1 type 0 = unsigned 1 = signed
17:10	CSM_D1[7:0]	R/W	0	Offset coefficient D_1
9	Unused		-	
8	CSM_D0_TWOS	R/W	0	Offset coefficient D_0 type 0 = unsigned 1 = signed
7:0	CSM_D0[7:0]	R/W	0	Offset coefficient D_0
Offset 0x10 C230	***Color space matrix offset coefficients E_0 - E_2***			
31:30	Unused		-	
29:20	CSM_E2[9:0]	R/W	0	Offset coefficient E2, two's complement
19:10	CSM_E1[9:0]	R/W	0	Offset coefficient E1, two's complement
9:0	CSM_E0[9:0]	R/W	0	Offset coefficient E0, two's complement
Vertical Video Processing Control Registers				
Offset 0x10 C240	***Initial Zoom***			

Table 8: Memory Based Scaler (MBS) Registers *...Continued*

Bit	Symbol	Access	Value	Description
31:29	VSP_PHASE_MODE[2:0]	R/W	0	Phase mode 0: 64 phases 1: 32 phases 2: 16 phases 3: 8 phases 4: 4 phases 5: 2 phases 6: fixed phase 7: linear phase interpolation (only valid for4 component mode)
28	Reserved			
27:26	VSP_FIR_COMP[1:0]	R/W		Vertical filter components 00: two components, 6 tap FIR each[1] 01: reserved 10: three components, 4 tap FIR each 11: four components, 3 tap FIR each [1] Filter lengths differ when in de-interlacing mode.
25:20	Reserved			
19: 0	VSP_ZOOM_0[19:0]	R/W	0	Initial zoom for 1st pixel in line (unsigned, LSB = 2^{-16}) 2 0000 (hex): downscale 50% 1 0000 (hex): no scaling = 2^{0} 0 8000 (hex): zoom 2 x
Offset 0x10 C244	***Phase Control***			
31	Reserved		-	
30:28	VSP_QSHIFT[2:0]	R/W	0	Quantization shift control Used to change quantization before being multiplied with 0.5. 100 (bin): divide by 16 101 (bin): divide by 8 110 (bin): divide by 4 111 (bin): divide by 2 000 (bin): multiply by 1 001 (bin): multiply by 2 010 (bin): multiply by 4 011 (bin): multiply by 8
27:26	Reserved		-	
25	VSP_QSIGN	R/W	0	Quantization sign bit
24:21	VSP_LDIFF_C[3:0]	R/W	0	Line offset for chroma line count (signed, needed for slicing only)
20	Reserved		-	
19:14	VSP_OFFSET_C[12:8]		-	Initial start offset for chroma DTO (Used for 4:2:0 scaling and de-interlacing only.)
13: 0	VSP_OFFSET_0[12:0]	R/W	0	Initial start offset for DTO
Offset 0x10 C248	***Initial Zoom delta***			
31:26	Reserved			
25: 0	VSP_DZOOM_0[25:0]	R/W	0	Initial zoom delta for 1 pixel in line (signed, LSB = 2^{-27}) Used for non-constant scaling ratios.
Offset 0x10 C24C	***Zoom delta change***			

Table 8: Memory Based Scaler (MBS) Registers *...Continued*

Bit	Symbol	Access	Value	Description
31:29	Reserved		-	
28: 0	VSP_DDZOOM[28:0]	R/W	0	Zoom delta change (signed, LSB = 2^{-40}) Used for non-constant scaling ratios.
EDDI Registers				
Offset 0x10 C260	***EDDI Control Register 1***			
31:28	Reserved			
27:24	EWM_REL	R/W	0	Edge Width Scale factor for edge matching EWM_REL/16 * max_width < min_width EWM_REL = 0 to 15
23:22	EWM_ABS	R/W	0	Edge Width Difference for edge matching (max_width - min_width) < EWM_ABS
21:16	MIN_LUMDIFF	R/W	0	Minimum Luminance difference
15:9	FPX_THR	R/W	0	Filter threshold
8:4	MAX_NRP	R/W	0	Maximum possible width for an edge
3:1	MIN_NRP	R/W	0	Minimum required width for an edge
0	ENABLE_EDDI	R/W	0	1: EDDI Enable Note: The ENABLE_EDDI bit in mode control reg must be set to 1 for the EDDI to be functional.
Offset 0x10 C264	***EDDI Control Register 2***			
31:16	Reserved			
15:12	RCM_REL	R/W	0	RC scale factor for RC matching RCM_REL /16 * max_rc < min_rc RCM_REL = 0 to 15
11:10	RCM_ABS	R/W	0	RC difference for RC matching (max_rc - min_rc) < RCM_ABS
9	Reserved			
8:4	SEARCH_LIMIT	R/W	0	Search limit to find a matching edge SEARCH_LIMIT/16 * edge_width SEARCH_LIMIT = 0 to 31
3:1	Reserved			
0	RC_CHECK	R/W	0	1: Enable rc check to calculate AIR
Color Keying Control Registers				
Offset 0x10 C280	***Color Key Control***			
31:30	CKEY_K2A	R/W	0	Color key to alpha convert 00 = no alpha manipulation 01 = fixed alpha at output 10 = reserved 11 = color key to alpha convert Alpha value for non-key sample is taken from CKEY_ALPHA register, alpha value for key sample is set to zero.

Table 8: Memory Based Scaler (MBS) Registers *...Continued*

Bit	Symbol	Access	Value	Description
29:28	CKEY_A2K	R/W	0	Alpha mode to color key convert 00 = no alpha manipulation 01 = reserved 10 = reserved 11 = alpha to color key convert Samples with alpha component below 80 (hex) are replaced with values defined in CKEY_COMP 1-3.
27:26	CKEY_REPLACE	R/W	0	Color keying replace mode 00 = no color component manipulation 01 = replace keyed color components with black (10, 10, 10) 10 = replace keyed color components with gray (80, 80, 80) 11 = replace keyed color components with last non-key value
25:24	Reserved			
23: 16	CKEY_MASK1	R/W	0	Color key mask component 1 Defines bits of color component that are compared against color key value setting to key sample.
15: 8	CKEY_MASK2	R/W	0	Color key mask component 2 Defines bits of color component that are compared against color key value setting to key sample.
7: 0	CKEY_MASK3	R/W	0	Color key mask component 3 Defines bits of color component that are compared against color key value setting to key sample.
Offset 0x10 C284	***Color Key Components***			
31: 24	CKEY_ALPHA	R/W	0	Alpha value Defines the alpha value to be used for keyed samples.
23: 16	CKEY_COMP1	R/W	0	Color key component 1 Defines value of color key for component 1 (red or Y).
15: 8	CKEY_COMP2	R/W	0	Color key component 2 Defines value of color key for component 2 (green or U).
7: 0	CKEY_COMP3	R/W	0	Color key component 3 Defines value of color key for component 3 (blue or V).
Video Output Format Control Registers				
Offset 0x10 C300	***Video Output Format***			
31:30	PSU_BAMODE	R/W	0	Base address mode 00 = single set (e.g. progressive video source) base 1-3 according to number of planes (plane 1-3) 01 = alternate sets each line (e.g. anti-flicker mode) base 1-3, first line out, etc. (plane 1-3) base 4-6, second line out, etc. (plane 1-3) 1x = reserved
29:14	Reserved			
13	PSU_ENDIAN	R/W	0	Output format endian mode 0: same as system endian mode 1: opposite of system endian mode
12	Reserved			

Table 8: Memory Based Scaler (MBS) Registers *...Continued*

Bit	Symbol	Access	Value	Description
11:10	PSU_DITHER	R/W	0	Output format dither mode 00: no dithering 01: error dispersion (never reset pattern) 10: error dispersion (reset pattern at startup) 11: error dispersion (reset pattern every field)
9:8	PSU_ALPHA	R/W	0	Output format alpha mode 00: no alpha (alpha byte not written to memory) 01: alpha byte written (see Color Keying Control) 10: reserved 11: reserved
7:0	PSU_OPFMT	R/W	0	Output formats 00 (hex) = YUV 4:2:0, semi-planar 03 (hex) = YUV 4:2:0, planar 08 (hex) = YUV 4:2:2, semi-planar 0B (hex) = YUV 4:2:2, planar 0F (hex) = RGB or YUV 4:4:4, planar A9 (hex) = compressed 4/4/4 + (4 bit alpha) AA (hex) = compressed 4/5/3 + (4 bit alpha) AD (hex) = compressed 5/6/5 A0 (hex) = packed YUY2 4:2:2 A1 (hex) = packed UYVY 4:2:2 E2 (hex) = YUV or RGB 4:4:4 + (8 bit alpha) E3 (hex) = VYU 4:4:4 + (8 bit alpha)
Offset 0x10 C304	***Target Window Size***			
31:27	Reserved			
26:16	PSU_LSIZE	R/W	0	Line size Used for horizontal cropping after scaling. 0 = cropping disabled 1 = one pixel **Remark:** Internal buffer lines are only 1024 pixels. Horizontal and vertical scaling need to be ordered such that the intermediate result fits into 1024 pixels buffer lines.
15:11	Reserved			
10:0	PSU_LCOUNT	R/W	0	Line count Used for vertical cropping after scaling. 0 = cropping disabled 1 = one line
Video Output Address Generation Control Registers				
Offset 0x10 C340	***Target Base Address #1***			
31:28	Unused		-	
27: 3	PSU_BASE1	R/W		Base address DMA #1 Used depending on PSU_BAMODE setting.
2:0	PSU_OFFSET1	R/W		Base address byte offset DMA #1 Bits define pixel offset within multi pixel 64-bit words (e.g., a 16-bit pixel can be placed on any 16-bit boundary).
Offset 0x10 C344	***Target Line Pitch #1***			

Table 8: Memory Based Scaler (MBS) Registers *...Continued*

Bit	Symbol	Access	Value	Description
31:15	Unused		-	
14: 3	PSU_PITCH1	R/W		Line pitch DMA #1, signed value (two's complement) Used for all packed formats and for plane 1.
2:0	Unused		-	
Offset 0x10 C348	***Target Base Address #2***			
31:28	Unused		-	
27: 3	PSU_BASE2	R/W		Base address DMA #2 Used depending on PSU_BAMODE setting.
2:0	PSU_OFFSET2	R/W		Base address byte offset DMA #2 Bits define pixel offset within multi pixel 64-bit words (e.g., a 16-bit pixel can be placed on any 16-bit boundary).
Offset 0x10 C34C	***Target Line Pitch #2***			
31:15	Unused		-	
14: 3	PSU_PITCH2	R/W		Line pitch DMA #2, signed value (two's complement) Used for planes 2 and 3.
2:0	Unused		-	
Offset 0x10 C350	***Target Base Address #3***			
31:28	Unused		-	
27: 3	PSU_BASE3	R/W		Base address DMA #3 Used depending on PSU_BAMODE setting.
2:0	PSU_OFFSET3	R/W		Base address byte offset DMA #3 Bits define pixel offset within multi pixel 64-bit words (e.g., a 16-bit pixel can be placed on any 16-bit boundary).
Offset 0x10 C354	***Target Base Address #4***			
31:28	Unused		-	
27: 3	PSU_BASE4	R/W		Base address DMA #4 Used depending on PSU_BAMODE setting.
2: 0	Unused		-	
Offset 0x10 C358	***Target Base Address #5***			
31:28	Unused		-	
27: 3	PSU_BASE5	R/W		Base address DMA #5 Used depending on PSU_BAMODE setting.
2: 0	Unused		-	
Offset 0x10 C35C	***Target Base Address #6***			
31:28	Unused		-	
27: 3	PSU_BASE6	R/W		Base address DMA #6 Used depending on PSU_BAMODE setting
2: 0	Unused		-	
Miscellaneous Registers				
Offset 0x10 C400—C7FC Color Look Up Table				
31:24	LUT_ALPHA[x]	W	-	Alpha

Table 8: Memory Based Scaler (MBS) Registers *...Continued*

Bit	Symbol	Access	Value	Description
23:16	LUT_RED[X][7:0]	W	-	Red or Y
15:8	LUT_GREEN[X][7:0]	W	-	Green or U
7:0	LUT_BLUE[X][7:0]	W	-	Blue or V
Offset 0x10 C800—C9FC Coefficient Table #1 Taps 0-5 (Horizontal)				
63:62	Unused		-	
61:52	TAP_5[X][9:0]	W	-	Inverted coefficient, tap #5, two's complement
51:42	TAP_4[X][9:0]	W	-	Inverted coefficient, tap #4, two's complement
41:32	TAP_3[X][9:0]	W	-	Inverted coefficient, tap #3, two's complement
31:30	Unused		-	
29:20	TAP_2[X][9:0]	W	-	Inverted coefficient, tap #2, two's complement
19:10	TAP_1[X][9:0]	W	-	Inverted coefficient, tap #1, two's complement
9:0	TAP_0[X][9:0]	W	-	Inverted coefficient, tap #0, two's complement
Offset 0x10 CA00—CBFCCoefficient Table #2 Taps 0-5 (Vertical - Luma)				
63:62	Unused		-	
61:52	TAP_5[X][9:0]	W	-	Inverted coefficient, tap #5, two's complement
51:42	TAP_4[X][9:0]	W	-	Inverted coefficient, tap #4, two's complement
41:32	TAP_3[X][9:0]	W	-	Inverted coefficient, tap #3, two's complement
31:30	Unused		-	
29:20	TAP_2[X][9:0]	W	-	Inverted coefficient, tap #2, two's complement
19:10	TAP_1[X][9:0]	W	-	Inverted coefficient, tap #1, two's complement
9:0	TAP_0[X][9:0]	W	-	Inverted coefficient, tap #0, two's complement
Offset 0x10 CC00—CDFCCoefficient Table #3 Taps 0-5 (Vertical - Chroma)				
63:62	Unused		-	
61:52	TAP_5[X][9:0]	W	-	Inverted coefficient, tap #5, two's complement
51:42	TAP_4[X][9:0]	W	-	Inverted coefficient, tap #4, two's complement
41:32	TAP_3[X][9:0]	W	-	Inverted coefficient, tap #3, two's complement
31:30	Unused		-	
29:20	TAP_2[X][9:0]	W	-	Inverted coefficient, tap #2, two's complement
19:10	TAP_1[X][9:0]	W	-	Inverted coefficient, tap #1, two's complement
9:0	TAP_0[X][9:0]	W	-	Inverted coefficient, tap #0, two's complement
Measurement Finish Flags Mask				
Offset 0x10 CE0C Flaggen Control Registers				
31:6	Reserved	-	-	
5	eaf_bbar_enable	R/W	0	'1' : meas_finish is only generated if eaf_bbar has occurred '0' : meas_finish is independent from eaf_bbar
4	eaf_blklvl_enable	R/W	0	'1' : meas_finish is only generated if eaf_blklvl has occurred '0' : meas_finish is independent from eaf_blklvl
3	eaf_histo_enable	R/W	0	'1' : meas_finish is only generated if eaf_histo has occurred '0' : meas_finish is independent from eaf_histo

Table 8: Memory Based Scaler (MBS) Registers *...Continued*

Bit	Symbol	Access	Value	Description
2	eaf_noise_enable	R/W	0	'1' : meas_finish is only generated if eaf_noise has occurred '0' : meas_finish is independent from eaf_noise
1	eaf_uvbw_enable	R/W	0	'1' : meas_finish is only generated if eaf_uvbw has occurred '0' : meas_finish is independent from eaf_uvbw
0	eaf_film_enable	R/W	0	'1' : meas_finish is only generated if eaf_film has occurred '0' : meas_finish is independent from eaf_film
Formatter Registers				
Offset 0x10 CE00	***Formatter Control***			
31:9	Reserved		-	
8:7	LINE_VALID [1:0]	W	3	Determines the lines to be measured 00 = reserved 01 = only passes even occurrences of lines for measurements 10 = only passes odd occurrences of lines for measurements 11 = passes all occurrences of lines for measurements
6	INPUT_FORMAT_UV	W	0	UV input range type selector: 0 (unsigned) = 0 (minimum) 32 (negative, 100% saturation) 256 (uncolored) 480 (positive, 100% saturation) 511 (maximum) 1 (signed) = -256 (minimum) -224 (negative, 100% saturation) 0 (uncolored) 224 (positive, 100% saturation) 255 (maximum)
5	INPUT_FORMAT_Y	W	0	Y input range type selector: 0 (unsigned) = 0 (minimum) 32 (black) 470 (white, 100%) 511 (maximum) 1 (signed) = -256 (minimum) -224 (black) 214 (white, 100%) 255 (maximum)
4:3	OUTPUT_FORMAT_UV [1:0]	W	0	UV output type selector 00 = Output is 9 bit, with LSB fixed to 0 (true 8 bit) 01 = Output is 9 bit, with LSB copied from LSB + 1 10 = Output is 9bit, from undithered 8 bit 11 = Output is true 9 bit The output range is always interpreted as signed
2:1	OUTPUT_FORMAT_Y [1:0]	W	0	Y output type selector 00 = Output is 9 bit, with LSB fixed to 0 (true 8 bit) 01 = Output is 9 bit, with LSB copied from LSB + 1 10 = Output is 9bit, from undithered 8 bit 11 = Output is true 9 bit The output range is always interpreted as unsigned

Table 8: Memory Based Scaler (MBS) Registers *...Continued*

Bit	Symbol	Access	Value	Description
0	Reserved		-	
Histogram Measurement Registers				
Offset 0x10 CE10	***Histogram Control Register***			
31:27	Reserved		-	
26	SUBSAMPLE_8x	R/W	0	8x subsample or 4x subsample of input data 0 = 4x subsample (for standard definition input) 1 = 8x subsample (for high definition input)
25	REPAIR_AC_ERROR	R/W	0	Enable measurement on gradual slope 0 = disable (default) 1 = enable
24:17	Reserved		-	
16:8	HISTO_THRES [8:0]	R/W	0	Threshold value for the measurement algorithm (0...511)
7:4	HISTO_GAIN [3:0]	R/W	F	0 to 10 = Gain for selection of the 8 output bits 11 to 14 = not used 15 = Calculate HISTO_GAIN in hardware
3	Reserved		-	
2	HISTO_NOISE_RED	R/W	1	Enable / disable noise reduction feature 0 = disable 1 = enable
1	FILT2_ENABLE	R/W	1	Enable / disable the use of filter 2 0 = disable 1 = enable
0	FILT1_ENABLE	R/W	1	Enable / Disable the use of filter1 0 = disable 1 = enable
Offset 0x10 CE18	***Histogram Window Start***			
31:27	Reserved		-	
26:16	HISTO_XWS [10:0]	R/W	1	Horizontal histogram measurement window start
15:11	Reserved		-	
10:0	HISTO_YWS [10:0]	R/W	1	Vertical histogram measurement window start
Offset 0x10 CE1C	***Histogram Window End***			
31:27	Reserved		-	
26:16	HISTO_XWE [10:0]	R/W	2D0	Horizontal histogram measurement window end
15:11	Reserved		-	
10:0	HISTO_YWE [10:0]	R/W	120	Vertical histogram measurement window end
Offset 0x10 CE8C	***Histogram Data Output 1***			
31:29	Reserved		-	
28:20	Y_MAX [8:0]	R		Maximum luminance value
19	Reserved		-	
18:10	Y_MIN [8:0]	R		Minimum luminance value
9:8	Reserved		-	

Table 8: Memory Based Scaler (MBS) Registers *...Continued*

Bit	Symbol	Access	Value	Description
7:0	HISTO_MAX_VALUE [7:0]	R		Level of the maximum histogram data
Offset 0x10 CE90	***Histogram Data Output 2***			
31:24	HISTO_DATA_3 [7:0]	R		Measured histogram data (bin 3, range 46...63)
23:16	HISTO_DATA_2 [7:0]	R		Measured histogram data (bin 3, range 32...47)
15:8	HISTO_DATA_1 [7:0]	R		Measured histogram data (bin 1, range 16...31)
7:0	HISTO_DATA_0 [7:0]	R		Measured histogram data (bin 0, range 0...15)
Offset 0x10 CE94	***Histogram Data Output 3***			
31:24	HISTO_DATA_7 [7:0]	R		Measured histogram data (bin 7, range 112...127)
23:16	HISTO_DATA_6 [7:0]	R		Measured histogram data (bin 6, range 96...111)
15:8	HISTO_DATA_5 [7:0]	R		Measured histogram data (bin 5, range 80...95)
7:0	HISTO_DATA_4 [7:0]	R		Measured histogram data (bin 4, range 64...79)
Offset 0x10 CE98	***Histogram Data Output 4***			
31:24	HISTO_DATA_11 [7:0]	R		Measured histogram data (bin 11, range 176...191)
23:16	HISTO_DATA_10 [7:0]	R		Measured histogram data (bin 10, range 160...175)
15:8	HISTO_DATA_9 [7:0]	R		Measured histogram data (bin 9, range 144...159)
7:0	HISTO_DATA_8 [7:0]	R		Measured histogram data (bin 8, range 128...143)
Offset 0x10 CE9C	***Histogram Data Output 5***			
31:24	HISTO_DATA_15 [7:0]	R		Measured histogram data (bin 15, range 240...255)
23:16	HISTO_DATA_14 [7:0]	R		Measured histogram data (bin 14, range 224...239)
15:8	HISTO_DATA_13 [7:0]	R		Measured histogram data (bin 13, range 208...223)
7:0	HISTO_DATA_12 [7:0]	R		Measured histogram data (bin 12, range 192...207)
Offset 0x10 CEA0	***Histogram Data Output 6***			
31:24	HISTO_DATA_19 [7:0]	R		Measured histogram data (bin 19, range 304...319)
23:16	HISTO_DATA_18 [7:0]	R		Measured histogram data (bin 18, range 288...303)
15:8	HISTO_DATA_17 [7:0]	R		Measured histogram data (bin 17, range 272...287)
7:0	HISTO_DATA_16 [7:0]	R		Measured histogram data (bin 16, range 256...271)
Offset 0x10 CEA4	***Histogram Data Output 7***			
31:24	HISTO_DATA_23 [7:0]	R		Measured histogram data (bin 23, range 368...383)
23:16	HISTO_DATA_22 [7:0]	R		Measured histogram data (bin 22, range 352...367)
15:8	HISTO_DATA_21 [7:0]	R		Measured histogram data (bin 21, range 336...351)
7:0	HISTO_DATA_20 [7:0]	R		Measured histogram data (bin 20, range 320...335)
Offset 0x10 CEA8	***Histogram Data Output 8***			
31:24	HISTO_DATA_27 [7:0]	R		Measured histogram data (bin 27, range 432...447)
23:16	HISTO_DATA_26 [7:0]	R		Measured histogram data (bin 26, range 416...431)
15:8	HISTO_DATA_25 [7:0]	R		Measured histogram data (bin 25, range 400...415
7:0	HISTO_DATA_24 [7:0]	R		Measured histogram data (bin 24, range 384...399)
Offset 0x10 CEAC	***Histogram Data Output 9***			

Table 8: Memory Based Scaler (MBS) Registers *...Continued*

Bit	Symbol	Access	Value	Description
31:24	HISTO_DATA_31 [7:0]	R		Measured histogram data (bin 31, range 496...511)
23:16	HISTO_DATA_30 [7:0]	R		Measured histogram data (bin 30, range 480...495)
15:8	HISTO_DATA_29 [7:0]	R		Measured histogram data (bin 29, range 464...479)
7:0	HISTO_DATA_28 [7:0]	R		Measured histogram data (bin 28, range 448...463)
Noise Estimation Registers				
Offset 0x10 CE20	***Noise Estimator Control Register 1***			
31:22	Reserved		-	
21	NEST_STALL	W	0	Stall of measurement unit 0 = measurement unit active (default) 1 = measurement unit stalled
20:19	CLIP_OFFSET [1:0]	W	0	Selection of clipping levels
18	SEL_SOB_NEGLECT_EXT	W	0	Selection of external SOB_NEGLECT_FLAG
17	SOB_NEGLECT_EXT	W	0	External SOB_NEGLECT flag
16:13	COMPENSATE [3:0]	W	0	Signed value added to NEST before low pass filtering
12:10	GAIN_UPBND [2:0]	W	0	Selection of coupling to low boundary
9:8	YPSCALE [1:0]	W	0	Scaling factor of the prefilter unit
7:0	WANTED_VALUE [7:0]	W	46	Controls the updown counting of images. When the number of times ce_SOB and ce_SAD are in agreement (equal '1') in an image, is smaller than WANTED_VALUE, unit are counting down; otherwise counting up.
Offset 0x10 CE24	***Noise Estimator Control Register 2***			
31:24	Reserved		-	
23:16	LB_DETAIL [7:0]	W	32	Lower limit for absolute difference between two adjacent pixels
15:8	Reserved		-	
7:0	UPB_DETAIL [7:0}	W	BE	Upper limit for absolute difference between two adjacent pixels
Offset 0x10 CE28	***Noise Estimator Window Start***			
31:27	Reserved		-	
26:16	NEST_XWS [10:0]	W	1	Horizontal Noise Estimator window start
15:11	Reserved		-	
10:0	NEST_YWS [10:0]	W	1	Vertical Noise Estimator window start
Offset 0x10 CE2C	***Noise Estimator Window End***			
31:27	Reserved		-	
26:16	NEST_XWE [10:0]	W	2D0	Horizontal NOise Estimator window end
15:11	Reserved		-	
10:0	NEST_YWE [10:0]	W	120	Vertical Noise Estimator window end
Offset 0x10 CEB0	***Noise Estimator Data Output 1***			
31:26	Reserved		-	
27:24	NEST [3:0]	R		The number of times in an image that the number of events (ce_SOB = ce_SAD) is lower than WANTED_VALUE.

Table 8: Memory Based Scaler (MBS) Registers *...Continued*

Bit	Symbol	Access	Value	Description
23:16	NEST_FILT[7:0]	R		Recursive filtered NEST value
15:8	DETAIL_CNT_L [7:0]	R		LSBs of the number of times in an image that the difference between two adjacent pixels falls within the range of 2*LB_DETAIL and 2*UPB_DETAIL
7:0	DETAIL_CNT_H [7:0]	R		MSBs of the number of times in an image that the difference between two adjacent pixels falls within the range of 2*LB_DETAIL and 2*UPB_DETAIL
Offset 0x10 CEB4	***Noise Estimator Data Output 2***			
31:9	Reserved		-	
7:0	GREY_COUNT [7:0]	R		MSBs of the number of pixels with values within the gray range
Black Bar Detection Registers				
Offset 0x10 CE30	***Black Bar Detector Control Register***			
31:24	BBD_EVENT_VALUE2 [7:0]	W	10	If number of black pixels > BBD_EVENT_VALUE2 the line is considered as black.
23:16	BBD_SLICE_LEVEL2 [7:0]	W	20	If luminance level < BBD_SLICE_LEVEL2 (multiplied by 2) the pixel is considered as black.
15:8	BBD_EVENT_VALUE1 [7:0]	W	15	If number of black pixels > BBD_EVENT_VALUE1 the line is considered as black.
7:0	BBD_SLICE_LEVEL1 [7:0]	W	55	If luminance level < BBD_SLICE_LEVEL1 (multiplied by 2) the pixel is considered as black.
Offset 0x10 CE38	***Black Bar Detection Window Start***			
31:27	Reserved		-	
26:16	BBD_XWS [10:0]	W	1	Horizontal Black Bar Detection window start
15:11	Reserved		-	
10:0	BBD_YWS [10:0]	W	1	Vertical Black Bar Detection window start
Offset 0x10 CE3C	***Black Bar Detection Window End***			
31:27				
26:16	BBD_XWE [10:0]	W	2D0	Horizontal Black Bar Detection window end
15:11				
10:0	BBD_YWE [10:0]	W	120	Vertical Black Bar detection window end
Offset 0x10 CEB8	***Black Bar Detection Data Output 1***			
31:27	Reserved		-	
26:16	BBD_LAST_VID_LINE1 [10:0]	R		Number of last video line detected (first detector)
15:11	Reserved		-	
10:0	BBD_FIRST_VID_LINE 1[10:0]	R		Number of first video line detected (first detector)
Offset 0x10 CEBC	***Black Bar Detection Data output 2***			
31:27	Reserved		-	
26:16	BBD_LAST_VID_LINE2 [10:0]	R		Number of last video line detected (second detector)

Table 8: Memory Based Scaler (MBS) Registers *...Continued*

Bit	Symbol	Access	Value	Description
15:11	Reserved		-	
10:0	BBD_FIRST_VID_LINE 2 [10:0]	R		Number of first video line detected (second detector)
Black Level Detection Registers				
Offset 0x10 CE40	***Black Level Detection Control***			
31:1	Reserved		-	
0	FILT1_ENABLE	W	1	Enable / Disable the use of filter1 0 = disable 1 = enable
Offset 0x10 CE48	***Black Level Detection Window Start***			
31:27	Reserved		-	
26:16	BLD_XWS [10:0]	W	1	Horizontal black level detection window start
15:11	Reserved		-	
10:0	BLD_YWS [10:0]	W	1	Vertical black level detection window start
Offset 0x10 CE4C	***Black Level Detection Window End***			
31:27	Reserved		-	
26:16	BLD_XWE [10:0]	W	2D0	Horizontal black level detection window end. Pixels from XWS up to and including XWE are processed.
15:11	Reserved		-	
10:0	BLD_YWE [10:0]	W	120	Vertical black level detection window end. Lines from YWS up to and including YWE are processed.
Offset 0x10 CEC0	***Black Level Detection Control / Output***			
8:0	SMARTBLACK [8:0]	R		Minimum luminance level
UV Bandwidth Detection Registers				
Offset 0x10 CE50	***Bandwidth Detection Control***			
31:9	Reserved		-	
8:0	MAX_DELTA_BW [8:0]	W	1FF	Slope of the rectifier: 0: no slope 511: maximum slope (default)
Offset 0x10 CE58	***Bandwidth Detection Window Start***			
31:27	Reserved		-	
26:16	BWD_XWS [10:0]	W	1	Horizontal bandwidth detection window start
15:11	Reserved		-	
10:0	BWD_YWS [10:0]	W	1	Vertical bandwidth detection window start
Offset 0x10 CE5C	***Bandwidth Detection Window End***			
31:27	Reserved		-	
26:16	BWD_XWE [10:0]	W	2D0	Horizontal bandwidth detection window end
15:11	Reserved		-	
10:0	BWD_YWE [10:0]	W	120	Vertical bandwidth detection window end

Table 8: Memory Based Scaler (MBS) Registers *...Continued*

Bit	Symbol	Access	Value	Description
Offset 0x10 CEC4	***Bandwidth Detection Output 1***			
31:29	Reserved		-	
28:20	V_MAX [8:0]	R		Signed maximum chrominance V value
19	Reserved		-	
18:10	V_MIN [8:0]	R		Signed minimum chrominance V value
9	Reserved		-	
8:0	UV_BANDWIDTH [8:0]	R		Estimated value of bandwidth detection
Offset 0x10 CEC8	***Bandwidth Detection Data Output 2***			
31:29	Reserved		-	
28:20	U_MAX [8:0]	R		Signed maximum chrominance U value
19	Reserved		-	
18:10	U_MIN [8:0]	R		Signed minimum chrominance U value
9:0	Reserved		-	
Interrupt and Status Control Registers				
Offset 0x10 CFE0	***Interrupt Status***			
31:30	STAT_PSU[1:0]	R	0	Status of pixel store unit (test signal)
29:27	Reserved			
26:16	STAT_PSU_LINE	R	0	Status of pixel store unit, line currently written
15:14	STAT_VSP[1:0]	R	0	Status of vertical scale pipe (test signal)
13:12	STAT_DFF[1:0]	R	0	Status of decimation FIR filter (test signal)
11:10	STAT_HSP[1:0]	R	0	Status of horizontal scale pipe (test signal)
9:8	STAT_IFF[1:0]	R	0	Status of interpolation FIR filter (test signal)
7	STAT_PFU[0]	R	0	Status of pixel fetch unit (test signal)
6	STAT_MEAS_DONE[	R	0	Status of measurement progress
5	STAT_TASK_ERROR	R	0	Processing error
4	STAT_TASK_END	R	0	Current task processing done
3	STAT_TASK_OVERFLOW	R	0	Task FIFO overflow
2	STAT_TASK_IDLE	R	0	Task finished and the task FIFO is empty.
1	STAT_TASK_EMPTY	R	0	Task FIFO runs empty .
0	STAT_TASK_DONE	R	0	Current task finished and all write complete.
Offset 0x10 CFE4	***Interrupt Enable***			
31:7	Reserved			
6	IEN_MEAS_DONE	R/W		Measurement processing complete.
5	IEN_TASK_ERROR	R/W	0	Processing error
4	IEN_TASK_END	R/W	0	Current task processing done.
3	IEN_TASK_OVERFLOW	R/W	0	Task FIFO overflow
2	IEN_TASK_IDLE	R/W	0	Task finished and the task FIFO is empty.

Table 8: Memory Based Scaler (MBS) Registers *...Continued*

Bit	Symbol	Access	Value	Description
1	IEN_TASK_EMPTY	R/W	0	Task FIFO runs empty.
0	IEN_TASK_DONE	R/W	0	Current task finished and all write complete.
Offset 0x10 CFE8	***Interrupt Clear***			
31:7	Reserved			
6	CLR_MEAS_DONE	W		Measurement processing complete.
5	CLR_TASK_ERROR	W	0	Processing error
4	CLR_TASK_END	W	0	Current task processing done.
3	CLR_TASK_OVERFLOW	W	0	Task FIFO overflow
2	CLR_TASK_IDLE	W	0	Task finished and the task FIFO is empty.
1	CLR_TASK_EMPTY	W	0	Task FIFO runs empty.
0	CLR_TASK_DONE	W	0	Current task finished and all write complete.
Offset 0x10 CFEC	***Interrupt Set***			
31:7	Reserved			
6	SET_MEAS_DONE	W		Measurement processing complete.
5	SET_TASK_ERROR	W	0	Processing error
4	SET_TASK_END	W	0	Current task processing done.
3	SET_TASK_OVERFLOW	W	0	Task FIFO overflow
2	SET_TASK_IDLE	W	0	Task finished and the task FIFO is empty.
1	SET_TASK_EMPTY	W	0	Task FIFO runs empty.
0	SET_TASK_DONE	W	0	Current task finished and all write complete.
Offset 0x10 CFF4	***Powerdown***			
31	Power_Down	RW	0	0 = Normal operation 1 = Powerdown mode
30:0	Reserved			
Offset 0x10 CFFC	***Module ID***			
31: 16	MOD_ID	R	0119	Module ID; unique 16-bit code
15: 12	REV_MAJOR	R	0x2	Major revision counter
11: 8	REV_MINOR	R	0x8	Minor revision counter
7: 0	APP_SIZE	R	0	Aperture Size: 0 = 4KB

Chapter 20: 2D Drawing Engine

PNX15xx Series Data Book – Volume 1 of 1

Rev. 2 — 1 December 2004 Product data sheet

1. Introduction

The purpose of this module is to accelerate the 2D drawing operations that are most heavily used in a graphics environment. The 2D drawing engine (2DDE or DE in the following text or data book) interfaces with both the internal MMIO-DTL bus and the internal memory bus. DE operation may be synchronous to the memory interface, with a small portion of the module operating at the MMIO bus clock frequency.

1.1 Features

The major features of the 2D drawing engine are:

- High Speed Operation
- 3 operand bit BLT (256 Raster operations)
- Alpha Blending
- Transparent bit BLT
- Fast host monochrome to full color expansion for text, mono bitmaps, and patterns
- Lines

2. Functional Description

A block diagram of the 2D drawing engine is shown in Figure 1.

2.1 2D Drawing Engine Block Level Diagram

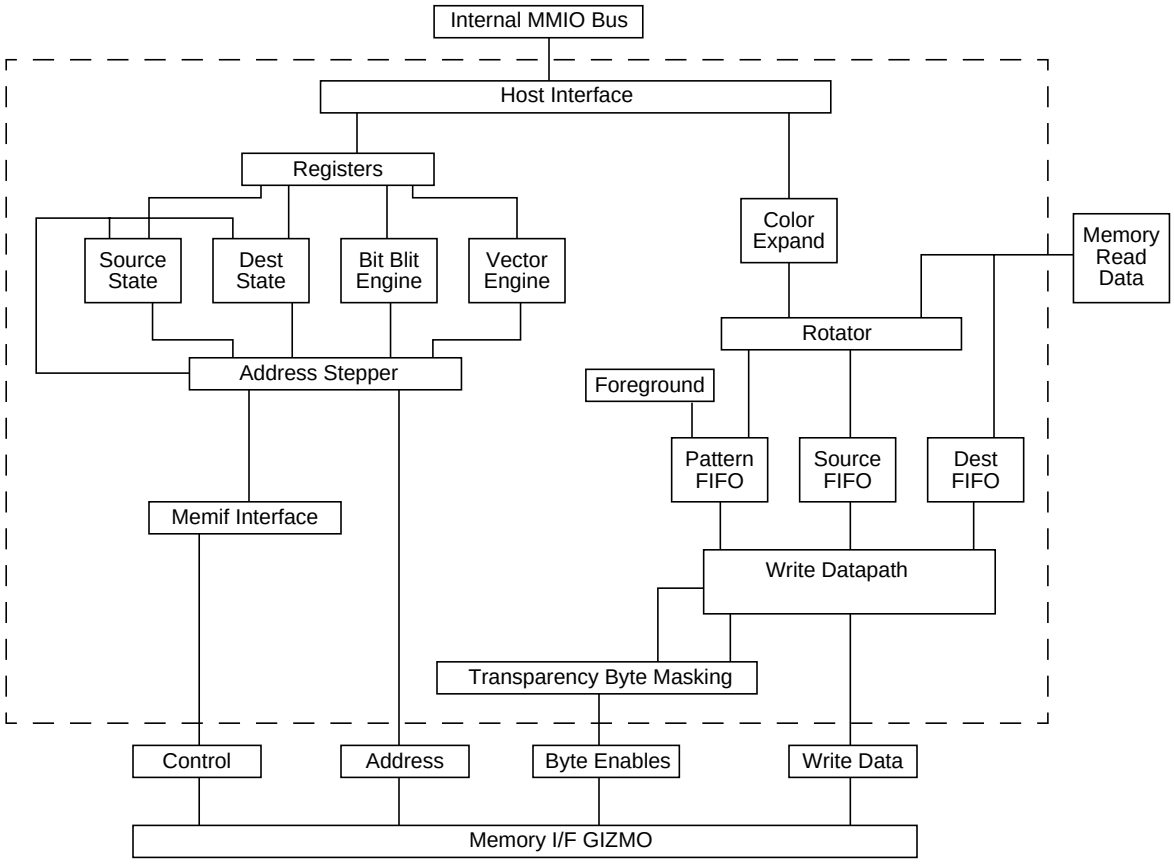


Figure 1: 2D Drawing Engine Block Diagram

2.2 Architecture

2.2.1 Registers

This block contains the MMIO registers for the drawing engine.

2.2.2 Host Interface

This block synchronizes and FIFOs data from the internal MMIO bus. Host data and patterns pass through this block as well as register operations.

2.2.3 Color Expand

Monochrome bitmaps, fonts, and patterns pass through this block and are expanded to the appropriate full color depth. The color expanded data will then be sent to the rotator block for alignment or loaded into the Pattern RAM. Full color data also passes through this block and is combined from DWORDS into QWORDS. Alpha data from the host is converted to the appropriate format by this block.

2.2.4 Rotator

This module aligns source data to the destination.

2.2.5 Source FIFO

This structure can hold up to 256 bytes of source data. Source data may be provided from either frame buffer memory or the host processor via the MMIO bus. Data in this FIFO will always be aligned to the destination. The output of this FIFO is used as the source operand for ROPs and is also used when transparency is dependent on the source.

2.2.6 Pattern FIFO

This structure can hold up to 256 bytes of pattern data. Pattern data is always locked to screen position and therefore never needs to be aligned. This FIFO can hold one 8 X 8 pattern in true color, 2 patterns in high color, and 4 patterns in pseudo-color mode. This FIFO will contain the foreground color value if solid patterns are enabled. The output of this FIFO is used as the pattern operand for ROPs. Transparency on patterns is not allowed.

2.2.7 Destination FIFO

This structure can hold up to 256 bytes of destination data. Destination data is required when a ROP requires the contents of the frame buffer to be combined with source data, pattern data, or both. Destination data is by definition always aligned and therefore does not need to be rotated. The output of this FIFO is used as the destination operand for ROPs and is also used when transparency is dependent on the destination.

2.2.8 Write Datapath

The write datapath includes the ROP, Alpha Blending, and color compare functions. The output of this block is the 64-bit data to be sent to the memory.

2.2.9 Source State

This block maintains the current state of the source address while the address stepper is processing the destination address.

2.2.10 Destination State

This block maintains the current state of the destination address while the address stepper is processing the source address. It also may maintain DST reads while DST writes are occurring, or vice versa.

2.2.11 Address Stepper

This block is responsible for the calculation of the addresses required for a given operation. The output of this block provides the address to the MTC and provides byte masking information to the byte mask block.

2.2.12 Bit BLT Engine

This is the main control logic for bit BLT commands. This block breaks BLTs down into appropriate subcommands such as SRC read, DST write, etc. and then sequences through those subcommands.

2.2.13 Vector Engine

This is the main control for line commands. This block contains the Bresenham engine.

2.2.14 Memory Interface

This block handles the interface to the DVP memory highway gizmo.

2.2.15 Byte Masking

This block combines bit BLT or vector byte mask information and transparency information to generate the byte enables for the MTC. This block also generates the block write masks.

2.3 General Operations

The Drawing Engine supports two types of operations: lines and bit BLTs. Line capability is limited to solid lines, with the software responsible for the calculation of the initial error terms of the Bresenham algorithm. Lines are provided as a compatibility check mark and are not highly optimized.

Bit BLTs are the primary function of the drawing engine and are highly optimized for performance. Bit BLT is the generic term used to indicate the transfer and processing of a block of visual data from one location to another. To specify the type of bit BLT, the following information is required:

- Raster Operation (ROP): of 256 possible ROPs
- Alpha Blend Mode: Source or Surface
- Source data location and type: system memory or CPU, mono, color, or alpha
- Transparency: On Source, On Destination, or none
- Pattern: 8 X 8 mono, 8 X 8 color, or solid

2.3.1 Raster Operations

The Drawing Engine supports a three operand bit BLT. The three operands are source, destination, and pattern. There are eight logical operations that can be performed with any combination of the three operands: one, zero, and, or, nand, nor, xor, nxor. This yields a total of 256 combinations of ROP that can be performed. Although all 256 ROPs are possible, only a handful of these ROPs are typically used.

Examples of a single operand BLT would be a pattern fill or a basic screen-to-screen copy where the source overwrites the destination. An example of a two operand BLT is the source is “xor’d” with the destination. A three operand BLT could “and” the source, pattern, and destination. The raster operation is defined by an 8-bit field specifying one of the possible 256 operations.

Raster-ops are turned off if alpha blending is enabled.

2.3.2 Alpha Blending

The Drawing Engine supports alpha blending of source and destination data. The destination data can be either 32-bit αRGB:8888, 32-bit αYUV8888, 16-bit RGB:565, 16-bit aRGB 4444, or 16-bit RGBa 4534. Source data can come either from the memory or from the host data port. The alpha calculations are based on the source and "surface" alpha values.

Raster-ops are disabled if alpha blending is selected.

The following combinations of source and destination data are supported.

Table 1: Source and Destination Data

Source	Source Format	Destination Format	Notes
Memory	αRGB:8888	αRGB:8888	Std alpha calculations
Host	αRGB:8888	αRGB:8888	Std alpha calculations
Host	α:4	αRGB:8888	MonoHostBColor reg provides src color
Host	α:8	αRGB:8888	MonoHostBColor reg provides src color
Memory	RGB:565	RGB:565	Surface reg specifies alpha
Host	RGB:565	RGB:565	Surface reg specifies alpha
Host	α:4	RGB:565	MonoHostBColor reg provides src color
Host	α:8	RGB:565	MonoHostBColor reg provides src color
Memory	RGBα:4534	RGBα:4534	Std alpha calculations
Host	RGBα:4534	RGBα:4534	Std alpha calculations
Host	α:4	RGBα:4534	MonoHostBColor reg provides src color
Host	α:8	RGBα:4534	MonoHostBColor reg provides src color
Memory	αRGB:4444	αRGB:4444	Std alpha calculations
Host	αRGB:4444	αRGB:4444	Std alpha calculations
Host	α:4	αRGB:4444	MonoHostBColor reg provides src color
Host	α:8	αRGB:4444	MonoHostBColor reg provides src color

2.3.3 Source Data Location and Type

The data for the source operand can reside in either the frame buffer memory or be provided by the host processor via the MMIO bus. There are five source selection options currently defined:

1. Source data is held in frame buffer memory (always full color data)
2. Source data is provided by the processor via the MMIO bus and is full color
3. Source data is provided by the processor via the MMIO bus and is a monochrome bitmap

4. Source data is provided by the processor via the MMIO bus and is a monochrome text font.
5. Source data is provided by the processor via the MMIO bus and is Alpha Blending data.

Option 1 is used for normal screen-to-screen operations such as a source-to-destination copy. Option 2 is used for a host-to-screen copy when the source data is a full color bitmap. Option 3 is used for color expanding a monochrome bitmap. The monochrome bitmap will be expanded to full color using the foreground and background color registers within the drawing engine. Option 4 is similar to Option 3 except that it is designed to handle tightly packed fonts, which are used to render text. A three-bit field is used to select the appropriate source data option. Option 5 is similar to Option 2 except that the data provided on the MMIO bus is either packed 4 bit alpha or packed 8-bit alpha information.

2.3.4 Patterns

The drawing engine provides an 8-by-8 pattern or “brush.” The pattern is always locked to the screen. This pattern can be solid, monochrome, or full color. If the pattern is solid, the color value will be taken from the foreground color register. A monochrome pattern is stored in system memory as two DWORDS (64 bits) of monochrome data. This monochrome data is written to the drawing engine where it is color expanded to the appropriate color depth and stored in the pattern RAM. A full color pattern will be directly transferred from system memory into the pattern RAM. Only one pattern may be stored in the pattern RAM at a given time.

2.3.5 Transparency

The drawing engine supports transparency on either source or destination data. Transparent patterns are not supported. Transparency works by comparing either source or destination to a value stored in a color compare register and then allowing (or disallowing) a write to occur based on the result of the compare. Transparency is implemented on a per-pixel basis: at 8 bpp one byte is used for the compare value. At 16 bpp one word is used for the compare, and at 32 bpp the entire color compare register is used for the compare. A transparency mask is provided in order to exclude certain bits within a pixel from being used in the color compare.

2.3.6 Block Writes

Block writes are not supported by the drawing engine.

3. Operation

3.1 Register Programming Guidelines

3.1.1 Alpha Blending

The drawing engine supports alpha blending of Source, Destination, and a global SurfaceAlpha. The following destination formats are useful with alpha blending:

- RGB:565(16 bit per pixel)
- αRGB:4444(16 bit per pixel)

- RGBα:4534(16 bit per pixel)
- αRGB:8888(32 bit per pixel)
- αYUV:8888(32 bit per pixel, full range)
- αYUV:8888(32 bit per pixel, D1 range)

Destination pixels always reside in memory. Source pixels may either come from memory (if they are in the same color format as the destination) or from the host data port. In addition to accepting pixels in the destination format, the host data port also accepts 4 or 8-bit alpha values. In this case, the color values for each source pixel come from the MonoHostBColor register (address 1424h).

The source pixels may contain either ‘normal’ or ‘pre-multiplied’ color values. If the source pixel is pre-multiplied, the alpha value in the source pixel is not applied to the source pixel. ‘Normal’ color values will have the source alpha applied.

In addition to blending source and destination data using the source’s alpha, the drawing engine also has a global “surface alpha.” The surface alpha is applies to all pixels in a alpha BLT. Surface alpha is stored in the MonoHostFColor register (address 1420h).

Two aYUV:8888 formats are supported. One of the formats (PFormat=1) assumes full range YUV data i.e., 0xff..0x00. The other format, (PFormat=2) assumes the normal D1 limited range of values: Y:235:16, U/V:240:16.

The following algorithm describes the alpha blending data flow. We assume that all values have been normalized to a range of 1..0. In reality, this will be represented by actual values of 0xFF..0x00.

```
// fetch a source pixel into Src
if (BltCtl.SRC == 0)
// fetch src pixel from system memory
....
else if (BltCtl.SRC == 1)
// src data is color from CPU
Src= HostData[31:0]
else if  (BltCtl.SRC == 4)
begin
// fetch 4 bits of alpha from host, expand to 8 bits
Src.alpha = (HostData[3:0] << 4) | HostData[3:0] ;
Src.red  = HAlphaColor.red
Src.green = HAlphaColor.green
Src.blue  = HAlphaColor.blue
end
else if  (BltCtl.SRC == 5)
begin
// fetch 8 bits of alpha from host
Src.alpha = HostData[7:0] ;
Src.red  = HAlphaColor.red
Src.green = HAlphaColor.green
Src.blue  = HAlphaColor.blue
end
else if (BltCtl.Src == 6)
// set src to 1 if no src data.
begin
Src.red  = HAlphaColor.red
Src.green = HAlphaColor.green
Src.blue  = HAlphaColor.blue
Src.alpha  = HAlphaColor.alpha
End

// copy mono foreground info into the SurfAlpha
SurfAlpha = MonoHostFColor

// handle src bitmaps without alpha values
if (BltCtl.A[1:0] == 3 || (PixelSize==16 && PFormat == 0) )
Src.alpha = 1;

// handle non-premultiplied source pixels
if (BltCtl.A[1:0] == 1)
begin
Src.red *= Src.alpha ;
Src.green *= Src.alpha ;
Src.blue *= Src.alpha ;
end

// scale the src with the surface alpha values
Src.red *= SurfAlpha.red;
Src.green *= SurfAlpha.green;
Src.blue *= SurfAlpha.blue;
Src.alpha *= SurfAlpha.alpha
```

```
// now update the destination
if (BltCtl.A[3] == 0)
begin
Dst.red = clamp(Src.red + (1 - Src.alpha) * Dst.red) ;
Dst.green = clamp(Src.green + (1 - Src.alpha) * Dst.green) ;
Dst.blue = clamp(Src.blue + (1 - Src.alpha) * Dst.blue) ;
// optionally update the dst alpha
if (BltCtrl.A[2])
Dst.alpha = clamp(Src.alpha + (1 - Src.alpha) * Dst.alpha);
end
else
begin
Dst.red = clamp(Src.red);
Dst.green = clamp(Src.green);
Dst.blue = clamp(Src.blue);
// optionally update the dst alpha
if (BltCtrl.A[2])
Dst.alpha = clamp(Src.alpha)
end
```

Alpha values in 32-bit pixels are always in the upper byte of the DWORD.

3.1.2 Mono Expand

The Engine needs to deal with two types of monochrome data from the host: fonts and bitmaps. The primary difference between fonts and bitmaps is how data is padded to byte boundaries.

First, it is worthwhile to note the bit ordering of monochrome data in a DWORD. Bit 7 is the left most pixel, bit 24 is the right most pixel. It is (unfortunately) a mish-mash of data formats since pixels in a byte are big-endian, but bytes are little-endian ordered. Thus, in a DWORD, bits are processed in the following order:

bit7, bit6, bit5,... bit0, bit15, bit4F4... bit8, bit 23... bit16, bit31... bit24

Fonts can be transferred to the drawing engine in a highly packed format with no pad data between bits on adjacent scanlines. Pad is added after the last bits in the last data byte. Since the font data in system memory always begins on a byte boundary, the host processor can easily arrange to deliver a series of 32-bit aligned DWORDs to the Engine. This font format has been widely used by Microsoft since Windows 95.

In the following example, the data stream for a 5*6 font requires 1 DWORD to be sent to the Engine.

```
                    31                                                0
Data Stream:00110001001001111010000110000110

Rendered Font:  10000
                11010
                10000
                10010
                01110
                01100
```

Note the 2 pad bits (25 & 24) in the last byte.

Since font data starts on a byte boundary, a source shift parameter is not required. The Width field of the BltSize register determines how many host bits will be processed before advancing to the next scanline. There are no pad bits between data bits of adjacent scanlines - the data is highly packed. Excess bits in the very last DWORD of host data will be discarded. The Engine will require

(BltSize.Width*BltSize.Height +31)/32

DWORDS for each glyph drawn. Mono Bitmaps sent to the Engine from Windows are not necessarily byte aligned on the left or right edges. The starting and ending bits of each scanline may be in the middle of a byte.

For mono bitmaps, the five LSBs of the SrcLinear register determine which bit in the first DWORD has the first valid data bit. The BltSize.Width register determines how many bits will be expanded/drawn for each scanline. The host will send

(BltSize.Width + (SrcLinear&31) + 31)/32

DWORDS for each scanline. The Engine will discard unused bits in the last DWORD of each scanline as pad bits. The driver must always send the correct number of DWORDS for each scanline in the bitmap.

3.1.3 Mono BLT Register Setup

To deal with both host-to-screen mono bitmap and text data in a general manner, the Engine uses up to eight parameters as shown in Table 2.

Table 2: Mono Bitmap & Text Data Parameters

Parameter	Description
DstXY or DstLinear	Specifies the drawing destination on the screen
SrcLinear	The three LSBs specify the leading pad bits in the first byte of data on each scanline.
DstStride	Specifies the number of pixels between scanlines in the destination
MonoHostFColor	Foreground color
MonoHostBColor	Background color

Table 2: Mono Bitmap & Text Data Parameters *…Continued*

Parameter	Description
CCColor	Color compare color for transparency
BltCtl	Drawing function
BltSize	Destination width and height

When the BltCtl.SRC register is set for font rendering (3), the Engine automatically accommodates the predefined padding and alignment requirements for fonts. This means that the host data will be tightly packed with no padding between scanlines or before the very first pixel. After the first font glyph has been sent from the host, only two registers will need to be re-loaded to initiate the next text BLT: DstXY/DstLinear and BltSize.

3.1.4 Solid Fill Setup

Solid fill is a common graphics operation. To achieve low programming overhead, the drawing engine only uses five parameters to set up a solid color fill.

Table 3: Solid Color Fill Parameters

Parameter	Description
DstXY or DstLinear	Specifies the drawing destination on the screen
DstStride	Specifies the number of pixels between scanlines in the destination
MonoPatFColor	Specifies the drawing color.
BltCtl	This register indicates that a solid fill operation is desired by setting the SRC field to 5, CC field to 0. The ROP field is a don’t care.
BltSize	Specifies destination width and height, initiates drawing function.

3.1.5 Color BLT Setup

The Engine uses up to 10 parameters to set up a color BLT.

Table 4: Color BLT Parameters

Parameters	Description
DstXY or DstLinear	Specifies the drawing destination on the screen.
SrcXY or SrcLinear	Specifies the location of source data for screen-to-screen BLTs. Specifies start of line alignment for host-to-screen operations.
SrcStride	Specifies the number of pixels between scanlines in the source for screen-to-screen BLTs. Unused for host-to-screen BLTs.
DstStride	Specifies the number of pixels between scanlines in the destination.
MonoPatFColor	Specifies the drawing color for solid fill BLTs using block write. Also may be used to load the PatRam quickly.
MonoPatBColor	The background color register may be used to help load the PatRam quickly.
CCColor	The Color Compare Color may be loaded if transparency is enabled in the BltCtl register.

Table 4: Color BLT Parameters *...Continued*

Parameters	Description
TransMask	The transparency mask is used to mask out bits for color compare operations.
BltCtl	drawing function: ROP type, color compare enables, source data path
BltSize	Specifies destination width and height, initiates drawing function.

Simple Screen-to-Screen BLTs use four registers to initiate the operation: SrcXY/SrcLinear, DstXY/DstLinear, BltCtl, and BltSize. This assumes that the stride registers have already been initialized with the current screen pitch.

Simple Color Host-to-Screen BLTs use four registers to initiate the operation: DstXY/DstLinear, SrcXY/SrcLinear, BltCtl, and BltSize. This assumes that the DstStride register has already been initialized with the current screen pitch.

The SrcXY/SrcLinear register specifies the data alignment at the start of each scanline. The first DWORD of data will have (SrcLinear & 3) pixels of pad data in the least significant bytes. Each scanline of host data is padded to end on a DWORD boundary. The Engine will draw BltSize.Width pixels for each scanline of the BLT. The number of DWORDS of host data for each scanline is

((BltSize.Width + (SrcLinear & 3))*(PSize/8) + 3)/4

Advanced BLTs will initialize a small group of additional registers. If transparency is desired, the Color Compare Color and Transparency Mask registers are loaded.

If patterns are used, the DstXY register and the PatRam (see below) must be loaded. The pattern is usually tiled to the screen assuming the upper left corner of the pattern is anchored to the upper left corner of the screen. To easily do this, the three low bits of the X and Y fields of the DstXY register are used to derive the initial alignment of the pattern data. The pattern register can be “un-anchored” by first writing to the DstXY register to specify the pattern alignment, then writing the DstLinear with the actual destination screen address. The last write to the DstXY register will set the pattern alignment. This means that you must always load the DstXY register before starting a BLT that uses the pattern.

3.1.6 PatRam

The PatRam is an 8*8 pixel pattern cache that provides pattern data for the ROP ALU. The PatRam operates at either 8, 16, or 32 bits per pixel as specified in the PSize register:

In 8 bit per pixel mode, only the first 64 bytes of the PatRam are used. The host must initialize bytes 0 to 63 prior to initiating a BLT that uses a pattern. In this mode, byte 0 of the PatRam is the upper left pixel in the ram.

In 16 bit per pixel mode, only the first 128 bytes of the PatRam are used. The host must initialize bytes 0 to 127 prior to initiating a BLT that uses a pattern. In this mode, bytes 0 and 1 of the PatRam are the upper left pixel in the ram. Byte 0 is the LSB of the pixel and contains five bits of the blue component and three bits of the green component. Byte 1 contains the remaining bits of the green component and five bits of the red component.

In 32-bit per pixel mode, all 256 bytes of each scanline of the PatRam are used. The host must load the entire PatRam prior to initiating a BLT that uses a pattern. In this mode, bytes 0..3 of the PatRam are the upper left pixel in the ram. Byte 0 is the LSB of the pixel and contains the blue component, byte 1 contains the green component, byte two contains the red component, and byte three is the alpha component (usually unused).

The PatRam can be loaded in two different ways to load either color data or monochrome data.

The 256 byte PatRamColor section of the register space allows direct byte/word/DWORD access to the PatRam. This allows arbitrary pattern data to be loaded. 64 pixels of data must be loaded into the PatRam prior to use. This ranges from 64 to 256 bytes of data depending on color depth. Byte 0 of this space is the first byte of PatRam scanline 0.

To assist in loading mono patterns, the 8 byte PatRamMono section of the register space provides automatic color expansion of mono data while loading the PatRam. Thus, the host only sends 16 bytes (four DWORDS) of data to initialize the entire pattern regardless of color depth: MonPatFColor, MonoPatBColor, and 8 bytes of mono data (64 bits). Each bit in the mono data stream loads the appropriate byte(s) of the PatRam with either the foreground color if the bit is a 1, or else the background color if the bit is a 0. Byte0/Bit7 loads the left pixel of scanline 0, byte1/Bit7 loads the left pixel of scanline 1, etc.

4. Register Descriptions

The drawing engine uses five areas of memory space:

1. The first memory space area is for the drawing engine command registers. These registers are used to set up and execute drawing engine commands. There is a block of unused memory space that has been reserved for future drawing engine functions.
2. The next area decoded is for monochrome pattern data. Although only 8 bytes of monochrome pattern data are required, a 256-byte decode is implemented to allow color expansion to occur to different areas of the pattern RAM.
3. The third decoded area is for full color pattern data. 64, 128, or 256 bytes of data may be written to the pattern RAM.
4. The fourth area decoded is for “real time” drawing registers. Unlike command registers which are pipelined, real time registers can be accessed immediately.
5. The last memory area decoded is a 64-KB area that is used to transfer host data to the drawing engine.

The addresses in the following table are offsets from the MMIO aperture base.

Table 5: 2DE Memory Space Addresses

Offset Range	Decoded for
0x04,F400 - 0x04,F47F 0x04,F5F8 – 0x04,F5FF	Drawing Engine Command registers
0x04,F480 – 0x04,F5F7	Reserved for future use
0x04,F600 - 0x04,F6FF	Monochrome Pattern Data
0x04,F700 - 0x04,F7FF	Color Pattern Data
0x04,F800 - 0x04,F80B	Real Time Drawing Engine registers
0x05,0000 - 0x05,FFFF	Drawing Engine Host Data

4.1 Register Summary

Table 6: 2D Command Registers

Offset	Symbol	Description
0x04 F400	SrcAddrBase	SRC Base address for XY->linear
0x04 F404	DstAddrBase	DST Base address for XY->linear
0x04 F408	Psize	Color depth for drawing operations
0x04 F40C	SrcLinear	BLT src address (linear)
0x04 F410	DstLinear	Vector/BLT dst address (linear)
0x04 F414	SrcStride	Scanline src pitch for BLTs
0x04 F418	DstStride	Scanline dst pitch for BLTs/vectors
0x04 F41C	CCColor	Color compare target
0x04 F420	MonoHostFColor and SurfAlpha	Foreground color for mono host expansion, SurfAlpha register for alpha blending
0x04 F424	MonoHostBColor and HAlphaColor	Background color for mono host expansion, color value for host alpha data for alpha blending
0x04 F428	BltCtl	Raster Op, etc. selection
0x04 F42C	SrcXY	BLT src address (XY)
0x04 F430	DstXY	Vector/BLT dst address (XY)
0x04 F434	BltSize	BLT Size (width and height) *Initiates BLT Drawing*
0x04 F438	DstXY2	Vector/BLT dst address (XY)
0x04 F43C	VecConst	Vector Bresenham constants
0x04 F440	VecCount	Vector length, octant, error term. Initiates Line Drawing
0x04 F444	TransMask	Transparency Mask

Table 6: 2D Command Registers *…Continued*

Offset	Symbol	Description
0x04 F448 – 0x04F5F4	Reserved	Reserved for future use
0x04 F5F8	MonoPatFColor	Foreground color for mono pattern expansion, lines, and solid fills
0x04 F5FC	MonoPatBColor	Background color for mono pattern expansion, lines, and solid fills

Table 7: 2D Real Time Drawing Registers

Offset	Symbol	Description
0x4F600 – 0x4F6FFC	PatRamMono	Monochrome Pattern cache RAM
0x4F700 – 0x4F7FFC	PatRamColor	Full Color Pattern cache RAM
0x4F800	EngineStatus	Status control of the engine
0x4F804	Panic Control	Reset
0x4F808	EngineConfig	Interrupt configuration
0x4F80C	HostFIFOStatus	Number of entries in the host FIFO
0x4 F810 – 0x4FFF0	Reserved	Reserved for future use
0x4FFF4	PowerDown	Powerdown activation
0x4FFFC	DeviceID	Module ID and aperture size
0x50000 – 0x5FFFC	HostData	64KB Memory space for the host

4.2 Register Tables

Table 8: Registers Description

Bit	Symbol	Access	Value	Description
2D Command Registers				
Offset 0x04 F400	***Source Address Base***			
31:29	Swap[2:0]	R/W	-	Specifies endian-swapping on reads from memory. When Swap[2] is 0, swapping is determined by the global endian setting, possibly modified by BSI[0] in EngineConfig. When Swap[2] is 1, swapping is determined by Swap[1:0] as shown: 00=No swapping. Memory is little-endian. 01=Bytes are swapped within each 16-bit word. 10=Words are swapped within each 32-bit double word. 11=Bytes are swapped within each 32-bit double word.
28:24	Reserved			

Table 8: Registers Description

Bit	Symbol	Access	Value	Description
23:16	Off[22:16]	R/W	-	Specifies the offset for pixel (0,0) of a source bitmap for XY to Linear conversion of addresses. Bits 2:0 must be set to 0.
15:8	Off[15:8]	R/W	-	
7:0	Off[7:0]	R/W	-	

This register specifies the offset for pixel (0,0) of a source bitmap for XY to Linear conversion of addresses.

This register is interpreted as a byte address. The lower three bits of this register always read 0, regardless of the value written. This implies that the source base address must be aligned to a 64-bit boundary.

Under many circumstances, this register will be initialized to the proper offset and then changed only for special effects.

Table 9: Destination Address Base

Bit	Symbol	Access	Value	Description
Offset 0x04 F404	***Destination Address Base***			
31:29	Swap[2:0]	R/W	-	Specifies endian-swapping on reads from memory. When Swap[2] is 0, swapping is determined by the global endian setting, possibly modified by BSI[0] in EngineConfig. When Swap[2] is 1, swapping is determined by Swap[1:0] as shown: 00=No swapping. Memory is little-endian. 01=Bytes are swapped within each 16-bit word. 10=Words are swapped within each 32-bit double word. 11=Bytes are swapped within each 32-bit double word.
28:24	Reserved			
23:16	Off[22:16]	R/W	-	Specify the offset for pixel (0,0) of a destination bitmap for XY to Linear conversion of addresses. Bits 2:0 must be set to 0.
15:8	Off[15:8]	R/W	-	
7:0	Off[7:0]	R/W	-	

This register specifies the offset for pixel (0,0) of a destination bitmap for XY to Linear conversion of addresses.

This register is interpreted as a byte address. The lower three bits of this register are always interpreted as 0, regardless of the value written. When reading the contents of this register, the lower three bits will be read back as 0. This implies that the source base address must be aligned to a 64-bit boundary.

Under many circumstances, this register will be initialized to the proper offset and then changed only for special effects.

Table 10: Pixel Size

Bit	Symbol	Access	Value	Description
Offset 0x04 F408	*Pixel Size*			
31:16	Reserved			
15:8	PFormat[7:0]	R/W	3	Currently used only during Alpha-Blending operations. The format is dependent on the color depth. Refer to Table 11 below for bit assignments. 16 bpp: 0000_0000RGB 565 0000_0001αRGB 4444 0000_0010RGBα 4534 0000_1000RGB 565 dithered 0000_1001αRGB 4444 dithered 0000_1010RGBα 4534 dithered 32 bpp: 0000_0000αRGB 8888 0000_0001αVYU 8888 0000_0010αYUV 8888 Note: Specifying any other combination of bits will result in an invalid command being executed.
7:0	Depth[5:0]	R/W	08	Specifies the number of bits per pixel. 00100000b32 bpp 00010000b16 bpp 00001000b8 bpp All other values are reserved.

This register specifies the pixel size and format for the drawing engine for BLTs and vectors. This register is unchanged by any drawing operations.

The Table 11 shows the pixel bit assignments for each of the six color formats above. Note that in the 4534 case, the format name is not indicative of the actual order of the bits.

Table 11: Pixel Format Bit Assignments

Format	31 24	23 16	15 8	7 0
RGB 565			RRRRRGGG	GGGBBBBB
αRGB 4444			aaaaRRRR	GGGGBBBB
RGBα 4534			aaaaRRRR	GGGGGBBB
αRGB 8888	aaaaaaaa	RRRRRRRR	GGGGGGGG	BBBBBBBB
αVYU 8888	aaaaaaaa	VVVVVVVV	YYYYYYYY	UUUUUUUU
αYUV 8888	aaaaaaaa	YYYYYYYY	UUUUUUUU	VVVVVVVV

The YUV values have ranges which match the D1 format in the CCIR 656. Y is an unsigned value ranging from 16 to 235. U and V are signed offset values ranging from 16 to 240, where 128 represents the zero point.

When Depth is 16, PFormat[3] enables dithering the results of an alpha blend operation. All alpha blend operations are done with 8 bits of precision for each component. The components of the source pixels are expanded to 8 bits by replicating the high-order bits into the low-order bits. The results of the computations can be thought of as being in fixed point, with 3, 4, 5 or 6 bits of precision to the left of the decimal point, depending on the component and color format. When dithering is enabled, a 4x4 dither is applied by adding a constant fraction to each component and truncating the results to fit in the resulting pixel. The constant fraction is taken from the following table, based on the X and Y coordinates of the destination pixel.

Table 12: Dithering

Format	X mod 4 = 0	X mod 4 = 1	X mod 4 = 2	X mod 4 = 3
Y mod 4 = 0	7/8	3/8	1/8	5/8
Y mod 4 = 1	1/8	5/8	7/8	3/8
Y mod 4 = 2	5/8	7/8	3/8	1/8
Y mod 4 = 3	3/8	1/8	5/8	7/8

Table 13: Source Linear

Bit	Symbol	Access	Value	Description
Offset 0x04 F40C	***Source Linear***			
31:24	Reserved			
23:16	Adr[22:16]	R/W	-	Used to load the linear source address for a BLT operation.
15:8	Adr[15:8]	R/W	-	
7:0	Adr[7:0]	R/W	-	

This register is used to load the linear source address for a BLT operation.

It must be loaded with a pixel aligned address. Note that loading the SrcXY register actually causes this register to be loaded with the proper linear pixel address.

This register is interpreted as a byte address, except during a monochrome host to screen bit BLT, or a 4-bit or 8-bit expand alpha BLT. In the monochrome BLT case, Adr[4:0] specifies the first valid bit within the first DWORD transferred by the host. Adr[4:3] specifies the correct byte and Adr[2:0] specify the correct bit. In the 4-bit expand case, Adr[3:0] specifies the first valid nibble within the alpha data transferred by the host. In the 8-bit expand case, Adr[2:0] specifies the first valid byte within the alpha data transferred by the host. This register is unchanged by drawing operations.

Table 14: Destination Linear

Bit	Symbol	Access	Value	Description
Offset 0x04 F410	***Destination Linear***			
31:24	Reserved			
23:16	Adr[22:16]	R/W	0	Used to load the starting linear pixel address for a vector or the destination linear address for a BLT operation.
15:8	Adr[15:8]	R/W	0	
7:0	Adr[7:0]	R/W	0	

This register is used to load the starting linear pixel address for a vector or the destination linear address for a BLT operation.

It is interpreted as a byte address and must be loaded with a pixel aligned address. Note that loading the DstXY register actually causes this register to be loaded with the proper linear pixel address.

This register may not be used to specify the destination address for a command that utilizes patterns.

Table 15: Source Stride

Bit	Symbol	Access	Value	Description
Offset 0x04 F414	***Source Stride***			
31:14	Reserved			
13:0	SrcStr	R/W	0x3FF8	Used to load the starting linear pixel address for a vector or the destination linear address for a BLT operation. Bits 2:0 must be set to 0.

This register holds the unsigned offset between adjacent source scanlines for screen-to-screen BLTs. Under many circumstances, this register will be initialized to the screen pitch and then changed only for special effects.

This 14-bit register is interpreted as an unsigned byte value. It is used during a BLT to step from scanline to scanline. It is also used to convert a SrcXY address to a SrcLinear address according to the following formula:

SrcLinear = SrcXY.Y * SrcStride + SrcXY.X (1) + SrcAddrBase

(1) This value is adjusted for pixel color depth.

There are no restrictions on this register except that the lower three bits are always interpreted as 0, regardless of the value written. When reading the contents of this register, the lower three bits will be read back as 0. This implies the source stride must be a multiple of 8 bytes.

As this is an unsigned register, it is always interpreted as a positive value. The direction in which a source is traversed is controlled by the BLT direction field in the BltCtl register.

This register may be useful for BLTing bitmaps stored in off screen memory in a 1D format to the screen. It is unchanged by any drawing operations.

Table 16: Destination Stride

Bit	Symbol	Access	Value	Description
Offset 0x04 F418	***Destination Stride***			
31:14	Reserved			
13:8	DstStr[13:8]	R/W	0	Hold the offset between adjacent scanlines for BLTs and vectors. Bits 2:0 must be set to 0.
7:0	DstStr[7:0]	R/W	0	

This register holds the offset between adjacent scanlines for BLTs and vectors. Under many circumstances, this register will be initialized to the screen pitch and then changed only for special effects. It is interpreted as an unsigned byte value.

This 14-bit signed register is used during BLTs and vectors to step from scanline to scanline. It is also used to convert a DstXY address to a DstLinear address according to the following formula:

DstLinear = DstXY.Y * DstStride + DstXY.X (1) + DstAddrBase

(1) This value is adjusted for pixel color depth

There are no restrictions on this register except that the lower three bits are always interpreted as 0, regardless of the value written. When reading the contents of this register, the lower three bits will be read back as 0. This implies that the destination stride must be a multiple of 8 bytes.

As an unsigned register, it is always interpreted as a positive value. The direction in which the destination is traversed is controlled by the BLT direction field in the BltCtl register.

This register may be useful for BLTing bitmaps stored in offscreen memory in a 1D format to the screen. It is unchanged by drawing operations.

Table 17: Color Compare

Bit	Symbol	Access	Value	Description
Offset 0x04 F41C Color Compare				
31:24	CCCol[31:24]	R/W	0	Holds the color compare target color.
23:16	CCCol[23:16]	R/W	0	
15:8	CCCol[15:8]	R/W	0	
7:0	CCCol[7:0]	R/W	0	

This register holds the color compare target color. This register should be initialized prior to any BLT that enables color compare. The appropriate number of bytes needs to be loaded in accordance with the current color depth. Thus, if the current depth is 8 bits, only the lowest byte need be written. If the depth is 16 bits, the lowest two bytes need to be written.

When reading the value of this register, the lower byte will be replicated in all four byte lanes in 8 bpp mode. In 16 bpp mode, the lower word will be replicated into the upper word. In 32 bpp mode, all bits are unique and will read back the 32-bit data that was written. This register is unchanged by drawing operations.

Table 18: Mono Host F Color or SurfAlpha

Bit	Symbol	Access	Value	Description
Offset 0x04 F420	*Mono Host F Color or SurfAlpha*			
31:24	FCol[31:24]alpha[7:0]	R/W	0	Specifies the foreground color for host bitmap monochrome expansion. For alpha-blending, this register specifies the global surface alpha values.
23:16	FCol[23:16]R/V/Y[7:0]	R/W	0	
15:8	FCol[15:8]G/Y/U[7:0]	R/W	0	
7:0	FCol[7:0]B/U/V[7:0]	R/W	0	

This register holds the foreground color for host bitmap monochrome expansion. The appropriate number of bytes need to be loaded in accordance with the current color depth. Thus, if the current depth is 8 bits, only the lowest byte need be written. If the depth is 16 bits, the lowest two bytes need to be written.

When used as SurfAlpha, this register always contains four unsigned 8-bit alpha values, regardless of color depth.

Note: When reading the value of this register, the lower byte will be replicated in all four byte lanes in 8 bpp mode. In 16 bpp mode, the lower word will be replicated in the upper word. In 32 bpp mode, all bits are unique and will read back the 32-bit data that was written. This register is unchanged by drawing operations.

Table 19: Mono Host B Color or HAlpha Color

Bit	Symbol	Access	Value	Description
Offset 0x04 F424	*Mono Host B Color or HAlpha Color*			
31:24	BCol[31:24]	R/W	0	Hold the background color for host bitmap monochrome expansion. For alpha-blending this register may provide color data.
23:16	BCol[23:16]R/V/Y[7:0]	R/W	0	
15:8	BCol[15:8]G/Y/U[7:0]	R/W	0	
7:0	BCol[7:0]B/U/V[7:0]	R/W	0	

This register holds the background color for host bitmap monochrome expansion. The appropriate number of bytes need to be loaded in accordance with the current color depth. Thus, if the current depth is 8 bits, only the lowest byte need be written. If the depth is 16 bits, the lowest two bytes need to be written.

When used as HostAlphaColor, the least significant three bytes of this register always contain 8-bit color values, regardless of the current depth. The most significant byte is supplied by the expanded host data during a 4-bit or 8-bit alpha expand BLT. In the YUV formats, the order of the components in the least significant three bytes matches that of the current PFormat.

When reading the value of this register, the lower byte will be replicated into all four byte lanes in 8 bpp mode. In 16 bpp mode, the lower word will be replicated into the upper word. In 32-bpp mode, or when A[3:0] in BltCtl are non-zero, all bits are unique and will read back the 32-bit data that was written. This register is unchanged by drawing operations.

Table 20: Blt Control

Bit	Symbol	Access	Value	Description
Offset 0x04 F428	*Blt Control*			
31:27	Reserved			
26:24	BD[2:0]	R/W	0	The BD[2:0] field specifies the bitblt direction. BD[0] specifies the horizontal BLT direction. It is bit 24 of this register: 0 = Blt direction is left to right. 1 = Blt direction is right to left. BD[1] specifies the vertical BLT direction. It is bit 25 of this register: 0 = Blt direction is top to bottom. 1 = Blt direction is bottom to top. BD[2] enables vertical flipping during the BLT by allowing the source data to be read in the opposite vertical direction from the destination data. Bd[2] is bit 26 of this register: 0 = Src is read as specified in BD[1.] 1 = Src is read in reverse of BD[1].
23:20	A[3:0]	R/W	0	The A[3:0] field controls alpha-blending operations and is in bits 23:20 of this register. A[1:0] controls enabling of alpha-blending and the format of the source data. The valid values are: 0 = Alpha-blending is disabled. 1 = Src data contains normal alpha data. 2 = Src data contains pre-multiplied alpha data. 3 = Src data does not have alpha data, surface alpha is the only alpha value. Bit 2 of this field controls the updating of the alpha field in the destination: 0 = Preserve destination alpha field. 1 = Update destination alpha field. Bit 3 of this field controls whether destination data participates in the alpha blend operation. It is used to implement “unary” blends: 0 = Blend source with destination data 1 = Blend source with black IN RGB mode, “black” means RGB = (0, 0, 0). In YUV mode, “black” means YUV = (16, 128, 128).
19	Reserved			
18:16	CC[2:0]	R/W	0	The CC[2:0] specifies the operation of the color compare hardware. CC[2:0] are bits 19:16 of this register. Legal values are: 0 = Color compare disabled. 1 = SRC data is used for color compare, perform write operation if colors match. 2 = DST data is used for color compare, perform write operation if colors match. 3 = Reserved 4 = Reserved 5 = SRC data is used for color compare, perform write operation if colors don’t match. 6 = DST data is used for color compare, perform write operation if colors don’t match. 7 = Reserved

Table 20: Blt Control

Bit	Symbol	Access	Value	Description
15:13	Reserved			
12	SP	R/W	0	The SP field indicates if the pattern is a solid color. SP is bit 12 of this register. Legal values are: 0 = Patterns are handled normally. 1 = The value held in the MonoPatFColor foreground color register will be used as pattern data.
11	Reserved			
10:8	SRC[2:0]	R/W	0	The SRC[2:0] field is a 3-bit parameter specifying how the source data are created. SRC[2:0] are in bits 11:8 of this register: 0 = SRC data is color data from SGRAM. This is used for screen-to-screen BLTs with either ROPs or alpha blends. 1 = SRC data is color bitmap data from the host processor. This is used for host-to-screen BLTs with either ROPs or alpha blends. 2 = SRC data is PC mono bitmap data from the host processor. This is used for color expanding host-to-screen BLTs. This encoding implies that host data is padded to a DWORD boundary at the end of scanlines. 3 = SRC data is PC mono font data from the host processor. This is used for text rendering. This encoding implies highly packed host data and forces the Drawing Engine to use SrcStride=BltSize. Width and SrcLinear=0. 4 = SRC data is 4-bit alpha values from the host. This option is used with alpha-blending. 5 = SRC data is 8-bit alpha values from the host. This option is used with alpha-blending. 6 = Use only surface values for alpha-blending, no SRC data present. 7 = Reserved
7:0	ROP[7:0]	R/W	0	This field is an 8-bit parameter that specifies the rasterop. It is the same format used by GDI. ROP[7:0] are in bits 7:0 of this register.

This register specifies the current rasterop, alpha, and other parameters for the drawing engine data path. This register must be properly initialized for BLTs, Alpha Blends, and vectors. This register is unchanged by any drawing operations.

Table 21: Source Address, XY Coordinates

Bit	Symbol	Access	Value	Description
Offset 0x04 F42C	***Source Address, XY Coordinates***			
31:27	Reserved			
26:24	Y[10:8]	R/W	0	Unsigned 11-bit Y source address
23:16	Y[7:0]	R/W	0	
15:11	Reserved			
10:8	X[10:8]	R/W	0	Unsigned 11-bit X source address
7:0	X[7:0]	R/W	0	

This register is used to load the source XY address for a BLT operation.

The X and Y fields are unsigned 11 bit numbers allowing a 2K by 2K address space. Since the drawing engine uses linear addresses internally, the X and Y coordinates in this register will be converted to a linear address. It is byte accessible; a write to the high byte of this register begins the conversion process from XY to linear. It is not used for vectors.

Note that SrcLinear should be utilized when using monochrome bitmaps or text. The lower six bits of SrcLinear specify the starting bit position within a DWORD. Also loads the SrcLinear register with the converted XY address.

Table 22: Destination Address, XY Coordinates

Bit	Symbol	Access	Value	Description
Offset 0x04 F430	***Destination Address, XY Coordinates***			
31:27	Reserved			
26:24	Y[10:8]	R/W	0	Unsigned 11-bit Y destination address
23:16	Y[7:0]	R/W	0	
15:11	Reserved			
10:8	X[10:8]	R/W	0	Unsigned 11-bit X destination address
7:0	X[7:0]	R/W	0	

This register is used to load the starting XY pixel destination coordinate for a drawing operation.

The X and Y fields are unsigned 11-bit numbers allowing a 2K by 2K address space. Since the drawing engine uses linear addresses internally, the X and Y coordinates in this register will be converted to a linear address. It is byte accessible; a write to the high byte of this register begins the conversion process from XY to linear.

This register causes the same behavior as writing to DstXY2, which is provided to allow for command register bursting during vector commands.

Drawing commands that require patterns must use this register to specify the destination coordinate. Using the DstLinear register to specify destination during a pattern command will cause errant results.

Table 23: BLT Size

Bit	Symbol	Access	Value	Description
Offset 0x04 F434	***BLT Size***			
31:28	Reserved			
27:16	H	R/W	FFF	Height of the BLT in scanlines
15:12	Reserved			
11:0	W	R/W	FFF	Width of the BLT in pixels

Also loads the DstLinear register with the converted XY address.This register specifies the height and width of a BLT operation in pixels/scanlines.

W[11:0] specifies the width of a BLT in pixels. The minimum allowed value is 1 and the maximum value is 4k-1. Zero is not a valid value for this field. W[11:0] corresponds to bits 11:0 of this register. H[11:0] specifies the height of a BLT in lines. The minimum allowed value is 1 and the maximum value is 4k-1. Zero is not a valid value for this field. H[11:0] corresponds to bits 27:16 of this register.

Note that loading the high byte of this register starts a BLT or Alpha Blending operation. This register is unchanged by any drawing operations.

Table 24: Destination Address, XY2 Coordinates

Bit	Symbol	Access	Value	Description
Offset 0x04 F438	***Destination Address, XY2 Coordinates***			
31:27	Reserved			
26:24	Y[10:8]	R/W	0	Unsigned 11-bit Y destination address
23:16	Y[7:0]	R/W	0	
15:11	Reserved			
10:8	X[10:8]	R/W	0	Unsigned 11-bit X destination address
7:0	X[7:0]	R/W	0	

This register is used to load the starting XY pixel destination coordinate for a drawing operation.

The X and Y fields are unsigned 11-bit numbers allowing a 2K by 2K address space. Since the drawing engine uses linear addresses internally, the X and Y coordinates in this register will be converted to a linear address. It is byte accessible; a write to the high byte of this register begins the conversion process from XY to linear.

This register causes the same behavior as writing to DstXY, which is provided to allow for command register bursting during BitBlt commands.

Drawing commands that require the use of patterns must use this register to specify the destination coordinate. Using the DstLinear register to specify the destination during a pattern command will cause errant results. Also loads the DstLinear register with the converted XY address.

Table 25: Vector Constant

Bit	Symbol	Access	Value	Description
Offset 0x04 F43C	***Vector Constant***			
31:24	Const1 [15:8]	R/W	0	Const1 = 2 x dmin
23:16	Const1 [7:0]	R/W	0	
15:8	Const0 [15:8]	R/W	0	Const0 = 2 x dmin - 2 x dmax
7:0	Const0 [7:0]	R/W	0	

This register holds the two error term values for the Bresenham line algorithm. These values are computed by the host processor as follows:

```
Const0 = 2 x dmin - 2 x dmax
```

```
Const1 = 2 x dmin
```

This register is unchanged after the vector is completed.

Table 26: Vector Count Control

Bit	Symbol	Access	Value	Description
Offset 0x04 F440	***Vector Count Control***			
31:24	InitErr[15:8]	R/W	0	Initial error value for the Bresenham line algorithm
23:16	InitErr[7:0]	R/W	0	
15:8	Oct, Len[11:8]	R/W	0	Bits 15:13 specify the drawing octant as follows: Bit 15 1 = Y is major axis 0 = X is major axis Bit 14 1 = negative X step 0 = positive X step Bit 13 1 = negative Y step 0 = positive Y step
7:0	Len[7:0]	R/W	0	Length of the major axis in pixels

This register holds the vector length, drawing octant, and the initial error value for the Bresenham line algorithm. The initial value of the error term is held in bits 31:16 and is computed by the host processor:

```
InitErr = 2 x dmin - dmax
Len = dmax + 1
```

where dmin = min (dx,dy), dmax = max(dx,dy)

The last pixel can be left undrawn by simply decrementing the Len parameter prior to loading.

Note: Loading the high byte of this register starts the vector drawing engine.

A Len value of 0 is illegal and should be avoided.

This register is unchanged after the vector is completed, but you need to reload it to start the next vector anyway.

Table 27: TransMask

Bit	Symbol	Access	Value	Description
Offset 0x04 F444	***TransMask***			
31:24	TMask[31:24]	R/W	NI	Transparency mask used in the color compare
23:16	TMask[23:16]	R/W	NI	
15:8	TMask[15:8]	R/W	NI	
7:0	TMask[7:0]	R/W	NI	

This register holds the transparency mask used in the color compare. It should be initialized prior to any BLT that enables color compare. The appropriate number of bytes need to be loaded in accordance with the current color depth. Thus, if the current depth is 8 bits, only the lowest byte need be written. If the depth is 15 or 16 bits, the lowest two bytes need to be written. In 32-bit mode, all bytes should be written.

Setting a bit to 1 in this register enables the corresponding bit within a pixel to be used in the color compare. Clearing a bit to 0 in this register excludes the corresponding bit within a pixel from being used in the color compare. This effectively causes that bit(s) to be considered a match in the color compare. Correct programming values are shown below:

08bpp: 0x000000FF // All 8 bits included in Color Compare

15bpp: 0x00007FFF // 15 lower bits included in Color Compare

16bpp: 0x0000FFFF // All 16 bits included in Color Compare

32bpp: 0x00FFFFFF // 24 lower bits included in Color Compare

When reading the value of this register, the lower byte will be replicated into all four byte lanes in 8-bpp mode. In 15 or 16-bpp mode, the lower word will be replicated into the upper word. In 32-bit mode, all bits are unique and will read back the 32-bit data that was written. This register is unchanged by drawing operations.

Table 28: MonoPatFColor

Bit	Symbol	Access	Value	Description
Offset 0x04 F5F8 MonoPatFColor				
31:0	MonoPatFColor[31:0]	R/W	NI	Specifies the foreground color for monochrome pattern expansion, lines, and solid fills.

This register specifies the foreground color for monochrome pattern expansion, lines, and solid fills. The appropriate number of bytes need to be loaded in accordance with the current color depth. Thus, if the current depth is 8 bits, only the lowest byte need be written. If the depth is 16 bits, the lowest two bytes need to be written.

When reading the value of this register, the lower byte will be replicated into all four byte lanes in 8-bpp mode. In 16-bpp mode, the lower word will be replicated into the upper word. In 32-bit mode, all bits are unique and will read back the 32-bit data that was written. This register is unchanged by drawing operations.

Table 29: MonoPatBColor

Bit	Symbol	Access	Value	Description
Offset 0x04 F5FC MonoPatBColor				
31:0	MonoPatBColor[31:0]	R/W	NI	Holds the background color monochrome pattern expansion, lines, and solid fills.

This register holds the background color for monochrome pattern expansion, lines, and solid fills. The appropriate number of bytes need to be loaded in accordance with the current color depth. Thus, if the current depth is 8 bits, only the lowest byte need be written. If the depth is 16 bits, the lowest two bytes need to be written.

When reading the value of this register, the lower byte will be replicated into all four byte lanes in 8-bpp mode. In 16-bpp mode, the lower word will be replicated into the upper word. In 32-bit mode, all bits are unique and will read back the 32-bit data that was written. This register is unchanged by drawing operations.

Table 30: EngineStatus

Bit	Symbol	Access	Value	Description
Drawing Engine Real Time Registers				
Offset 0x04 F800 EngineStatus				
31:11	Reserved			
10	IRQ	R	0	Draw Engine interrupt request status 1 = A 2D interrupt is being requested. This reflects the actual state of the IRQ signal leaving the Drawing Engine.
9	DEDone	R	1	DeDone and DEBusy are the primary Drawing Engine activity indicators.They are the complement of each other. When DEBusy is logic 1 (DEDone a 0), the Drawing Engine is active. If EngineConfig bit 9 is a 1, “active” is defined as: processing a register access emptying commands or data from the Host FIFO performing an operation such as a bitblt or bitblt line waiting for a memory transaction to complete If EngineConfig bit 9 is a 0, the engine is active when a blt/vector starts and becomes inactive when the blt/vector finishes. All memory writes are complete, AND the host FIFO is empty. DEBusy is read as logic 0, the Drawing Engine is guaranteed to be idle.
8	DEBusy	R	0	
7:4	Reserved			
3	HFIFO_not empty	R	0	Host FIFO is not empty.
2	HFIFO_full	R	0	Host FIFO is full. BLT is busy; another BLT is pending in shadow registers.
1	Vector active	R	0	Vector is in process.
0	BLT active	R	0	BLT is in process.

This read-only register returns the drawing engine status. Writes to this register have no effect but do not hang the bus.

Table 31: PanicControl

Bit	Symbol	Access	Value	Description
Offset 0x04 F804	*PanicControl*			
31:8	Reserved			
7:0	RST	W	-	Used to reset the state machines in the Drawing Engine. Writing 0x0000_0001 to this register will halt the Drawing Engine and place it in an idle state. Writing 0x0000_0000 will allow the DE to be reprogrammed and resume normal operation. It may be necessary for software to implement a delay between setting the RST signal to 1 and resetting it to 0.

This register is currently used to reset the state machines in the drawing engine. It does not reset any other registers in the Engine.

Table 32: EngineConfig

Bit	Symbol	Access	Value	Description
Offset 0x04 F808	*EngineConfig*			
31:11	Reserved			
10	IRQ_CLR	R/W	0	IRQ_CLR (bit 10) is a self-clearing bit used to reset Drawing Engine IRQ flip-flop. The IRQ flip-flop is set when the DEBusy bit in the EngineStatus register transitions from 1 to 0. It is cleared under software control by setting IRQ_CLR to 1. See BMODE for a description of DEBusy.
9	BMODE	R/W	0	BMODE selects between two slightly different behaviors of DEBusy and DEDone (EngineStatus register). 0=the DEBusy bit is set to 1 when the BLT/vector state machine becomes active i.e., a drawing operation is starting. The bit is cleared only when the state machine is idle, all memory writes are completed, and the command FIFO is empty. 1=the DEBusy bit is set whenever the BLT/vector state machine is active OR the command FIFO is not empty. The DEBusy bit will be cleared when the engine is idle AND the command FIFO is empty. Note that in this mode, the Draw Engine will go busy whenever a register is loaded. Using interrupts can be tricky in this mode.
8	IRQ_EN	R/W	0	IRQ_EN (bit 8) is used to enable the interrupt signal leaving the Drawing Engine module. When IRQ_EN is set to a 1, the interrupt signal is enabled. When set to 0, the interrupt signal leaving the Drawing Engine is masked. The IRQ_EN does not affect the actual IRQ flip-flop. It merely masks the IRQ bit leaving the module.
7:3	Reserved			

Table 32: EngineConfig

Bit	Symbol	Access	Value	Description
2	BSM	R/W	0	BSI (bit 1) and BSM (bit 2) toggle byte and word swapping. Setting BSI to 1 reverses the sense of the global endian bit for data read from the host data input port, so that data are swapped when written when they normally would not be, and vice versa. This bit affects host BLT data, pattern loading. It is intended for debugging and should be set to 0 for normal operation.
1	BSI	R/W	0	BSI (bit 1) and BSM (bit 2) toggle byte and word swapping. Setting BSM to 1 reverses the sense of the global endian bit for data being read and written to the memory port. It is intended for debugging and should be set to 0 for normal operation.
0	Reserved			

This register is used to configure specific drawing engine features and to enable the interrupt request signal.

BSI (bit 1) and BSM (bit 2) toggle byte and word swapping. Certain registers which contain byte- or word-oriented data but which must be manipulated as DWORDS need to be byte- or word-swapped when written on big-endian architectures. These include HostData, PatRamMono and PatRamColor. The drawing engine automatically determines whether swapping is necessary based on the global “endian” flag.

The bottom byte of this register should not be altered while drawing commands are in progress.

Table 33: HostFIFOStatus

Bit	Symbol	Access	Value	Description
Offset 0x04 F80C	***HostFIFOStatus***			
31:6	Reserved			
5:0	Level[5:0]	R	0x20	Used to provide software with a method of determining the number of available entries in the Host FIFO.

This register is used to provide software with a method of determining the number of available entries in the Host FIFO.

There are 32 FIFO locations in the Host FIFO. If this register is read as 0, it indicates there are no available FIFO locations available for writing. A write to the Host FIFO while the FIFO is full (or almost full) will result in wait states on the PI bus.

If this register is read as 0x20, it indicates that all 32 entries are available for writing.

A software driver can read this register and subsequently write the corresponding number of DWORDS of data to the Host FIFO. This will prevent the PNX15xx Series from generating PCI retries since a write will not occur when the Host FIFO is full.

Table 34: POWERDOWN

Bit	Symbol	Access	Value	Description
Offset 0x04 FFF4	*POWERDOWN*			
31	POWER_DOWN	R/W	0	Powerdown register for the module 0 = Normal operation of the peripheral. This is the reset value. 1 = Module is powered down and module clock can be removed. At powerdown, module responds to all reads with DEADABBA (except for reads of powerdown bit) and all writes with ERR ACK (except for writes to powerdown bit).
30:0	Unused		-	Ignore during writes and read as zeroes.

This register is used to provide software with the module type, revision, and aperture size.

Table 35: Module ID

Bit	Symbol	Access	Value	Description
Offset 0x04 FFFC	*Module ID*			
31:16	ModuleID	R	0x0117	ModuleID
15:12	MajRev	R	0x2	Major revision
11:8	MinRev	R	0x0	Minor revision
7:0	Size	R	0x10	MMIO Aperture size is 68 KB.

Drawing Engine Data Registers

Table 36: Drawing Engine Data Registers

Bit	Symbol	Access	Value	Description
Offset 0x04 F600—F6FF	*PatRamMono*			
This address range allows write-only access to the pattern RAM. It is used to load monochrome patterns into the pattern cache with automatic color expansion. Each bit in this address range represents one pixel in the pattern cache. A ‘1’ written here is converted to the foreground color and written to the Pattern RAM. A ‘0’ written here is converted to the background color and written to the Pattern RAM. A monochrome pattern always consists of 64 bits of data regardless of the pixel color depth. The amount of color expanded data, however, is dependent on color depth. The monochrome data should be written to the address range 0x1600 - 0x1607. Note that patterns must be loaded sequentially because the lower order address bits are ignored as a pattern is loaded.				

Table 36: Drawing Engine Data Registers

Bit	Symbol	Access	Value	Description
Offset 0x04 F700—F7FF PatRamColor (256 bytes)				
This address range allows write access to the pattern cache ram. It is only for full color patterns. A full color pattern consists of 64 bytes of data at 8 bpp, 128 bytes at 16 bpp, and 256 bytes of data at 32 bpp. Full color patterns should be loaded starting at address 0x1700. Patterns must be loaded sequentially because the lower order address bits are ignored as a pattern is loaded.				
Offset 0x05 0000—FFFF Host Data (64 kB - Memory Space)				
This address space receives host data for BLTs that require host data. All addresses in this range map to the host write buffer. It allows the host to use REP MOVSD instructions to efficiently copy data from system memory to the BLT engine. When doing a BLT that requires host data (host to screen), it is necessary to program SRC[1:0] in the BltCtl register to select host data. This register is byte accessible, but... Host-to-Screen BLTs always require an integer number of DWORDs to be transferred from the host. Although the HostData port will accept byte writes to load individual bytes of data, writing the high byte actually transfers the host data into the Engine. Reads from this register return unknown data but do not hang the bus. Writing excess data to this register space is not recommended, but will not cause the bus to hang.				

Chapter 21: MPEG-1 and MPEG-2 Variable Length Decoder

PNX15xx Series Data Book – Volume 1 of 1

Rev. 2 — 1 December 2004 Product data sheet

1. Introduction

The MPEG-2 Variable Length Decoder (VLD) consists of two Write DMA channels that write back data to main memory; one for Run Length (RL) data and one for Macro Block Header (MBH) data.

The VLD decodes the Huffman encoded MPEG-1 and MPEG-2 (main profile and main level) video elementary bitstreams. The VLD unit, enabled by the CPU, operates independently during the slice-level decoding. The remaining bitstream decoding is carried out by the TriMedia (TM3260) CPUs and appropriate software. The VLD also assists the CPU and outputs a stream of macroblock headers and a stream of run-level pairs. Run-level pair expansion, produced by the VLD into actual quantized DCT values and their subsequent rearrangement into a natural order, is carried out by the TM3260.

The TM3260 CPUs are also responsible for restoring or “dequantizing” the quantized DCT values by multiplying them by the corresponding values in the 8x8 DCT quantization matrix. The TM3260 optionally performs the frequency domain filtering in association with the half-resolution mode, and perform the inverse discrete cosine transformation on each of the 8x8 dequantized blocks.

The TM3260 CPU reconstructs the final video samples from the macroblock header data decoded by the VLD, the reference frame data stored in main memory, and the differential data previously computed.

1.1 Features

After initialization, the TM3260 CPU controls the VLD through its command register. There are currently nine commands supported by the VLD:

- Shift the bitstream by some number of bits.
- Parse a given number of macroblocks, one row, or parse continuously without stopping at the slice header.
- Search for the next start code.
- Reset the Variable Length Decoder.
- Initialize the VLD.
- Search for the given start code.

- Flush VLD output buffers.

2. Functional Description

2.1 VLD Block Level Diagram

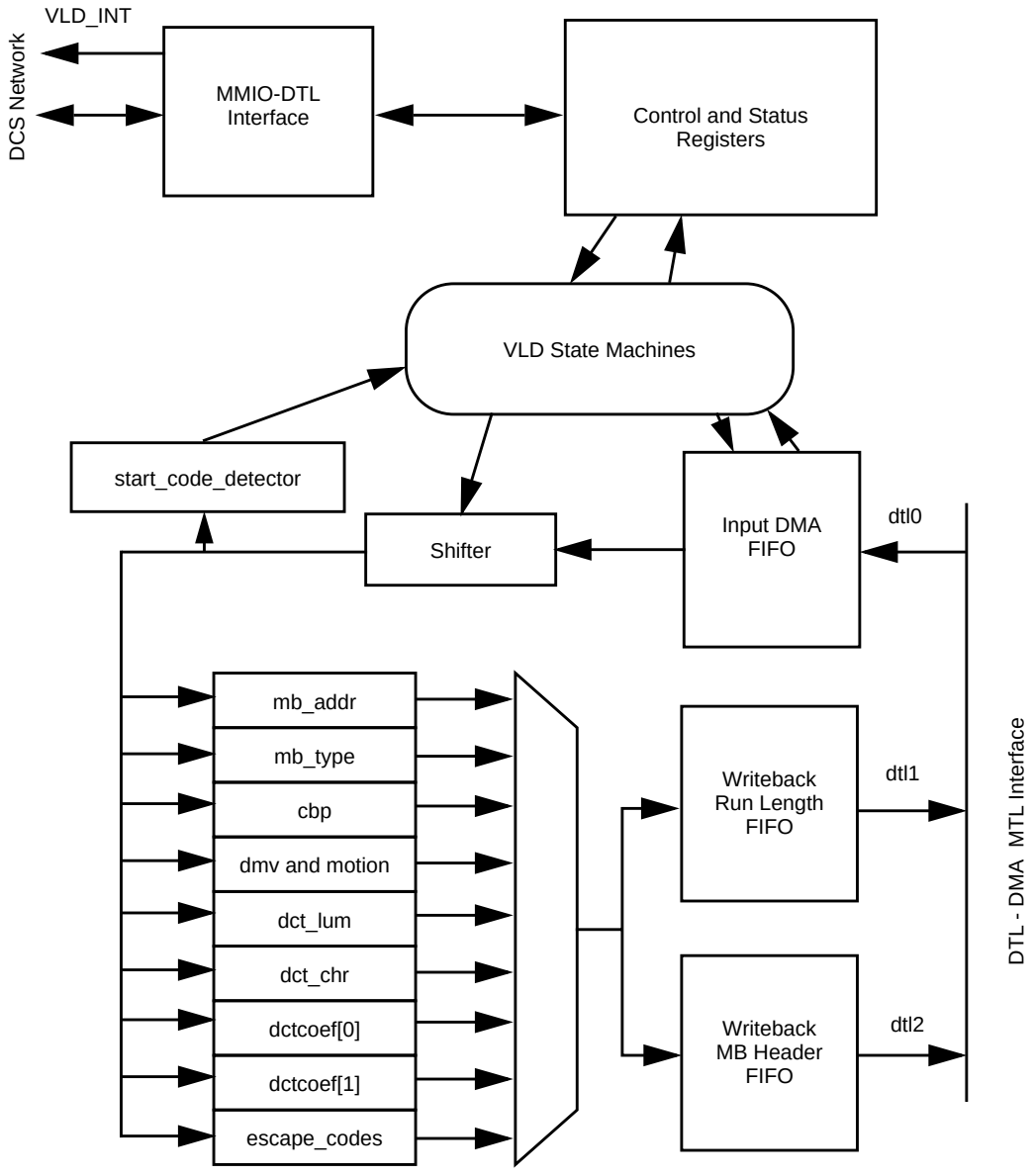


Figure 1: VLD Block Diagram

3. Operation

3.1 Reset-Related Issues

A system reset will cause the Variable Length Decoder to clear all registers to default values and will force all state machines to the IDLE state. Any DMA activity in

progress will be aborted.

The “Reset the Variable Length Decoder” command Section 3.2.9 on page 21-7 or Soft Reset is controlled by the hardware so that any DMA activity that is in progress will complete before the soft reset has a complete effect. Software should wait for the “VLD Command Done” status bit to be set before proceeding with the next command.

Remark: The VLD_INP_CNT is cleared after reset. However, this is not treated as a DMA_INPUT_DONE condition in the VLD and the CPU *will not* be interrupted by the VLD after reset with a DMA_INPUT_DONE condition. The MPEG video bitstream buffer should be refilled as needed and the VLD_INP_CNT rewritten with a proper value before issuing new commands to the VLD after reset.

Table 1: Software Reset Procedure

Cycle No.	Action	Remarks
i	The CPU issues the ‘Reset the Variable Length Decoder’ command by writing the corresponding command code into the VLD_COMMAND register.’	
i to j	The VLD will complete any DMA transactions that are already in progress. Any new DMA transactions will be aborted. The VLD then raises the vld_ready_to_reset signal.	Any DMA transactions, once started, will not be aborted in the middle.
k	If (vld_ready_to_reset) then the VLD interrupts the CPU.	Assumes k>j; otherwise it is jth cycle. The full reset clears all internal buffers, state machines, and leaves the following registers with the value of 0x0: VLD_COMMAND VLD_CTL VLD_BIT_CNT VLD_INP_CNT VLD_MBH_CNT VLD_RL_CNT The VLD_STATUS register contains the value 0x0000_0001.

3.2 VLD MMIO Registers

3.2.1 VLD Status (VLD_MC_STATUS)

The VLD_MC_STATUS register contains current status information which is most pertinent to the normal operation of an MPEG video decode application. Writing a logic ‘1’ to any of the status bits other than bit-0 clears the corresponding bit. Writing a logic ‘0’ has no effect. *Exception: Bit 0 (Command Done) is cleared only by issuing a new command. Writing a logic ‘1’ to bit zero of the status register will result in undefined behavior of the VLD*. Note that several status bits may be asserted simultaneously.

Table 2 lists the function of each status field.

Table 2: VLD STATUS

Name	Size (Bits)	Description
VLD Command Done	1	Logic ‘1’ indicates successful completion of current command. This bit is cleared by issuing a new command.
Start Code Detected	1	Logic ‘1’ indicates VLD encountered 0x000001 while executing current command. This bit is cleared by writing a logic ‘1’ to it.
Bitstream Error	1	Logic ‘1’ indicates VLD encountered an illegal Huffman code or an unexpected start code. Refer to Section 3.4 Error Handling for details on the error handling procedure. This bit is cleared by writing a logic ‘1’ to it.
DMA Input Done	1	Conditions for setting this bit depends on the value of the DMA_Input_Done field in the VLD_CTL register. Refer to Section 3 VLD Control and Section 3.3 VLD Operation for details. This bit is cleared by writing a logic ‘1’ to it.
DMA Macroblock Header Output Done	1	Logic ‘1’ indicates that the macroblock header DMA write transfer has completed.
DMA Run/Level Output Done	1	Logic ‘1’ indicates that the Run/Level DMA write transfer has completed.
RL overflow Error	1	Logic ‘1’ indicates Overflow of run/level values within a block. Refer to Section 3.4 for details on the error handling procedure. This bit is cleared by writing a logic ‘1’ to it.

When an error occurs in the VLD, the corresponding error flag (Bitstream error or RL overflow error) is set and an interrupt is generated if corresponding bits in the VLD_IE register is set. Refer to section Section 3.4 for details on the error handling mechanism.

3.2.2 VLD Interrupt Enable (VLD_IE)

This VLD_IE read/write register allows the CPU to control the initiation of the interrupt for the corresponding bits in the VLD_MC_STATUS register. Writing a one to any of these bits in the VLD_IE register enables the interrupt for the corresponding bit in the status register.

3.2.3 VLD Control (VLD_CTL)

When VLD detects a new slice start code in the bit-stream, it writes the lower 8-bits of the start code into the slice_start_code field of the VLD_CTL register before interrupting the CPU. When re-started, the VLD reads the slice_start_code from the VLD_CTL register and writes this value into bits 16-23 of the last word in the first mb_header and sets the mb_first bit to 1. To allow the CPU to switch bitstream at the slice level, CPU can write the desired slice start code and slice_start_code_strobe in the VLD control register. The value of ‘1’ in the slice_start_code_strobe will cause the VLD to update the slice_start_code field with the given slice_start_code value. The other fields in the VLD_CTL register are not updated when the input data contains a value of ‘1’ in the slice_start_code_strobe field. The slice_start_code_strobe bit is always read as 0. The CPU must write the slice_start_code only when the VLD is not active. In order to update the DMA_INPUT_DONE_MODE fields, the slice_start_code_strobe bit value must be set to ‘0’.

The use of the DMA_INPUT_DONE_MODE bit is described in Table 3.

Table 3: VLD Control

Name	Size (Bits)	Description
DMA_input_done_mode	1	When this bit is '0', VLD sets the DMA_INPUT_DONE flag (in VLD_MC_STATUS register) when the DMA_INP_CNT transitions from non-zero to zero. When this bit is '1', the same flag is set only with the additional condition that both DMA input buffers are empty. The slice_start_code_strobe bit field must be set to '0' in order to update this field.
slice_start_code	8	Slice start code when the VLD is restarted; the slice_start_code_strobe bit field must be set to '1' in order to update this field.
slice_start_code_strobe	1	When CPU writes 1 into this field, VLD copies the value of slice_start_code into its internal register. CPU should do this only when the VLD is stopped. This bit is always read as 0.

3.2.4 VLD DMA Current Read Address (VLD_INP_ADR) and Read Count (VLD_INP_CNT)

The CPU writes the main memory buffer address from which bitstream to be read by VLD in VLD_INP_ADR register. The number of bytes to be read by the VLD is updated by the CPU in the VLD_INP_CNT register.

The VLD unit uses two 64-byte buffers to store the input bitstream. The VLD reads the bitstream data from the main memory and updates the VLD_INP_ADR and the VLD_INP_CNT register. The content of the VLD_INP_ADR register reflects the next or the current fetch address of the bitstream data.

The VLD interrupts the CPU when it has consumed all the given bitstream data in the main memory (the DMA_INPUT_DONE condition). The value of the DMA_INPUT_DONE_MODE bit in the VLD_CTL register is used to select the condition for raising the DMA_INPUT_DONE flag. Refer to Table 3 for more details.

The VLD input address is word (32-bit) aligned and the count value in number of bytes is also word aligned.

3.2.5 VLD DMA Macroblock Header Current Write Address (VLD_MBH_ADR)

The CPU writes the main memory macroblock header buffer address in the VLD_MBH_ADR register in order to output the macroblock header data in main memory. The VLD updates this address whenever data is transferred to main memory via the DMA logic. The address always represents the next write address of the macroblock header data. This register must be 32-bit aligned.

3.2.6 VLD DMA Macroblock Header Current Write Count

The CPU writes the main memory macroblock header buffer size formatted as the number of 8-byte words into the VLD_MBH_CNT register in order to output the macroblock header data in main memory. The VLD updates the buffer size whenever data is transferred to main memory via the DMA logic. The buffer size always represents the remaining empty buffer space.

Note that in MPEG-2 when Macroblock Headers are written to main memory, they are written in groups of six 4-byte vectors (24 bytes).

3.2.7 VLD DMA Run-Level Current Write Address (VLD_RL_ADR)

The CPU writes the main memory run-level pair buffer address in the VLD_RL_ADR register in order to output the run-level pairs in main memory. The VLD updates this address whenever data is transferred to main memory via the DMA logic. The address always represents the next write address of the macroblock header data. The buffer address must be 32-bit aligned.

3.2.8 VLD DMA Run-Level Current Write Count

The CPU writes the main memory run-level pairs buffer size formatted as the number of words into the VLD_RL_CNT register in order to output the run-level pairs data in main memory. The VLD updates the buffer size whenever data is transferred to main memory via the DMA logic. The buffer size always represents the remaining empty buffer space.

3.2.9 VLD Command (VLD_COMMAND)

This read/write register indicates the next action to be taken by the VLD. A command is sent to the VLD by writing the corresponding 4-bit command code in the COMMAND field of the VLD_COMMAND register. Some commands require an associated count value which resides in the least significant 8 bits of this register. The following nine VLD commands are currently available:

- Shift the video bitstream by “count” bits (where “count” is given in the COUNT field of the register and must be between 0 and 15 inclusive).
- Parse “count” macroblocks (where “count” is given in the COUNT field of the register and must be between 0 and 255 inclusive).
- Search for the next start code.
- Reset the Variable Length Decoder.
- Initialize the VLD (to clear the VLD_BIT_CNT register).
- Search for the specific 8-bit start code pattern given in the COUNT field of the register.
- Flush the VLD output buffers to main memory.
- Parse one row of macroblocks.
- Parse macroblocks continuously. (COUNT field is unused, but cannot be programmed to 0.)

Table 4: VLD Commands

Code	Command	Flags Set (after completion of the command)	Description
0x1	Shift the bitstream by “count” bits	Command Done	VLD shifts the number of bits in its internal shift register. The shift register value is available in the VLD_SR register. The flag is reset by issuing the new command.
0x2	Parse for a given number of macroblocks	Command Done and/or Start Code Detected	VLD parses for a given number of macroblocks; however, if VLD encounters a start code, the parsing action will be terminated and VLD sets only the start code detected flag. If VLD parses the given number of macroblocks without encountering a start code, VLD will set the command done flag. The start code detected flag is reset by writing a ‘1’ value to the flag. The command done flag is reset by issuing the new command
0x3	Search for the next start code	Start Code Detected and Command Done	VLD search for a start code. The search code has 0x000001 prefix and additional 8-bit value; a 32-bit value with 0x000001 prefix. the start code detected flag is reset by writing a ‘1’ value to the flag. The command done flag is reset by issuing the new command
0x4	Reset the Variable Length Decoder	Command Done	Refer to Section 3.4 Error Handling.
0x5	Initialize the VLD	None	The bit count register is initialized to zero. The initialization action is immediate without any delay.
0x6	Search for the given start code	Start Code Detected and Command Done	VLD search for a start code with a given 8-bit lsb of the 32-bit start code. The search code has 0x000001 prefix and the additional 8-bit value is given in the ‘count’ field of the VLD_COMMAND register. The start code detected flag is reset by writing a ‘1’ value to the flag. The command done flag is reset by issuing the new command
0x7	Parse one row of macroblocks	Start Code Detected	This command instructs the VLD to parse one complete row of macroblocks. If the row contains more than one slice, VLD parses the intermediate slice headers without CPU intervention, provided these slice headers have a 0 bit after the 5-bit quantizer_scale_code. If the VLD encounters a start code different from the start code of the current slice, or if the slice header has a 1 bit after the quantizer_scale_code, it sets the Start-Code-Detected flag and ends the operation. *WARNING:* The ‘Count’ field of the VLD_COMMAND register is still in effect as in the ‘Parse A Number Of Macroblocks’ Command: VLD stops and sets the Command-Done flag after ‘Count’ macroblock headers are parsed. ‘Count’ must be set to at least mb_width (number of macroblocks per row in the picture) to guarantee the entire row is parsed before the VLD stops.
0x8	Flush Write FIFOs	Command Done	The VLD flushed the remaining macroblock header data and the remaining run-level data to the main memory.
0x9	Parse Long	Command Done	This command instructs the VLD to parse and to continue as long as the next start code ID is for the next slice. Any other start code ID will cause the VLD to stop and interrupt with the ‘Command Done’ status bit set. Note the count field cannot be programmed to 0.

The CPU must wait for the VLD to halt before the next command can be issued. Note that there are several ways in which a command may complete. Only a successful completion is indicated by the command done bit in the status register. A command may complete unsuccessfully if a start code or an error is encountered before the

requested number of items have been processed. Note also that the expiration of a DMA count does not constitute the completion of a command. When a DMA count expires the VLD is stalled as it waits for a new DMA to be initiated. It is not halted.

The ‘Initialize VLD’ command initializes the VLD_BIT_CNT register (the bit counter) to zero.

The ‘Search for the given start code’ command searches for the start code pattern given in the COUNT field (in the least significant 8 bits) of the VLD_COMMAND register. A valid MPEG-1/-2 start code is a 32-bit pattern with the upper 24-bits equal to 0x000001. For each start code encountered in the bitstream, the 8-bits following the 0x000001 pattern is compared against the given 8-bit start code pattern until a perfect match is obtained. Once the given start-code is found, the VLD sets the COMMAND_DONE bit in the VLD_MC_STATUS register. At that point, the VLD will interrupt the CPU if the corresponding bit in the VLD_IE register is also set.

Table 5: VLD Command Register

Name	Size (Bits)	Description
Count	8	For the ‘Shift Bitstream’ command, only the lower 4 bits are used; the upper 4 bits should be set to 0. All 8 bits are used for the ‘Parse macroblocks’ and ‘Search for given start code’ commands.
Command	4	Command code of the VLD command to be executed

3.2.10 VLD Shift Register (VLD_SR)

This read only register is a shadow of the VLD’s operational shift register and it allows the CPU to access the bitstream through the VLD. Bits 0 through 15 are the current contents of the VLD shift register. Bit 16 to 31 are RESERVED and should be treated as undefined by the programmer.

3.2.11 VLD Quantizer Scale (VLD_QS)

This 5-bit register read/write register contains the quantization scale code to be output by the VLD until it is overridden by a macroblock quantizer scale code. The quantizer scale code is part of the macroblock header output.

3.2.12 VLD Picture Info (VLD_PI)

This 32-bit read/write register contains the picture layer information necessary for the VLD (and MC) to parse the macroblocks within that picture. Again, the values of each of these fields are determined by the appropriate standard (MPEG-1 or MPEG-2)

3.2.13 VLD Bit Count (VLD_BIT_CNT)

The number of bits consumed by the VLD is updated in the VLD_BIT_CNT register. VLD_BIT_CNT can be initialized to zero by issuing the ‘Initialize VLD’ command. It counts upward when bits are shifted out and consumed by the VLD. This counter wraps around after reaching the maximum value.

3.3 VLD Operation

The normal mode of operation will be for the CPU to request the VLD to parse some number of macroblocks. Once the VLD has begun parsing macroblocks it may stop for any one of the following reasons:

- The command was completed with no exceptions.
- A start code was detected.
- An error was encountered in the bitstream.
- The VLD input DMA completed and the VLD is stalled waiting for more data.
- One of the output DMA for RL or HDR processing has been completed.

Under normal circumstances, the CPU is interrupted whenever the VLD halts.

Consider the case in which the VLD has encountered a start code. At this point, the VLD will halt and set the status flag which indicates that a start code has been detected. This flag will generate an interrupt to the CPU. Upon entering the interrupt service routine, the CPU will read the VLD status register to determine the source of the interrupt. Once it has been determined that a start code has been encountered, the CPU will read 8 bits from the VLD shift register to determine the type of start code that has been encountered. If a slice start code has been encountered, the CPU will read from the shift register the slice quantization scale code and any extra slice information (from the slice header). The slice quantization scale code will then be written back to the VLD_QS register. Before exiting the interrupt service routine, the CPU will clear the start code detected status bit in the status register and issue a new command to process the remaining macroblocks.

3.3.1 VLD Input

The VLD reads the video bitstream from the main memory and performs the variable length decoding process. The CPU writes the main memory buffer address from which bitstream to be read by VLD in VLD_INP_ADR register. The number of bytes to be read by the VLD is updated by the CPU in the VLD_INP_CNT register.

The VLD unit uses two 64-byte buffers to store the input bitstream. The VLD reads the bitstream data from the main memory and updates the VLD_INP_ADR and the VLD_INP_CNT register. The content of the VLD_INP_ADR register reflects the next fetch address of the bitstream data. The content of the VLD_INP_CNT register reflects the number of bytes to be read from the main memory. When the number of bytes to be read from the main memory transitions from non-zero to zero, the DMA_INPUT_DONE flag in the VLD_MC_STATUS is set. An interrupt will be sent to the CPU also if the corresponding interrupt enable bit in the VLD_IE register is set. The CPU should then provide the new bitstream buffer address and the number of bytes in the bitstream buffer to the VLD.

The VLD unit also updates a bit counter in the VLD_BIT_CNT register to keep track of the number of bits consumed in the decoding process. The bit counter is updated only after a successful decoding of a symbol. The CPU can read this bit counter from the VLD_BIT_CNT register. This register can be initialized to zero by sending the ‘Initialize VLD’ command.

3.3.2 VLD Output

The output of the VLD are always transferred back to main memory for down stream software MPEG blocks. All VLD output are transferred via DMA to separate main memory areas for MB headers and run-level encoded DCT coefficients.

MPEG-2 Parsing

For each MPEG-2 macroblock parsed by the VLD, six 32 bit words of macroblock header information will be output from the VLD. See Figure 2.

The fields described in Figure 2 may or may not be valid depending upon the MPEG-2 video standard. See Table 6 for more details on the macroblock header data.

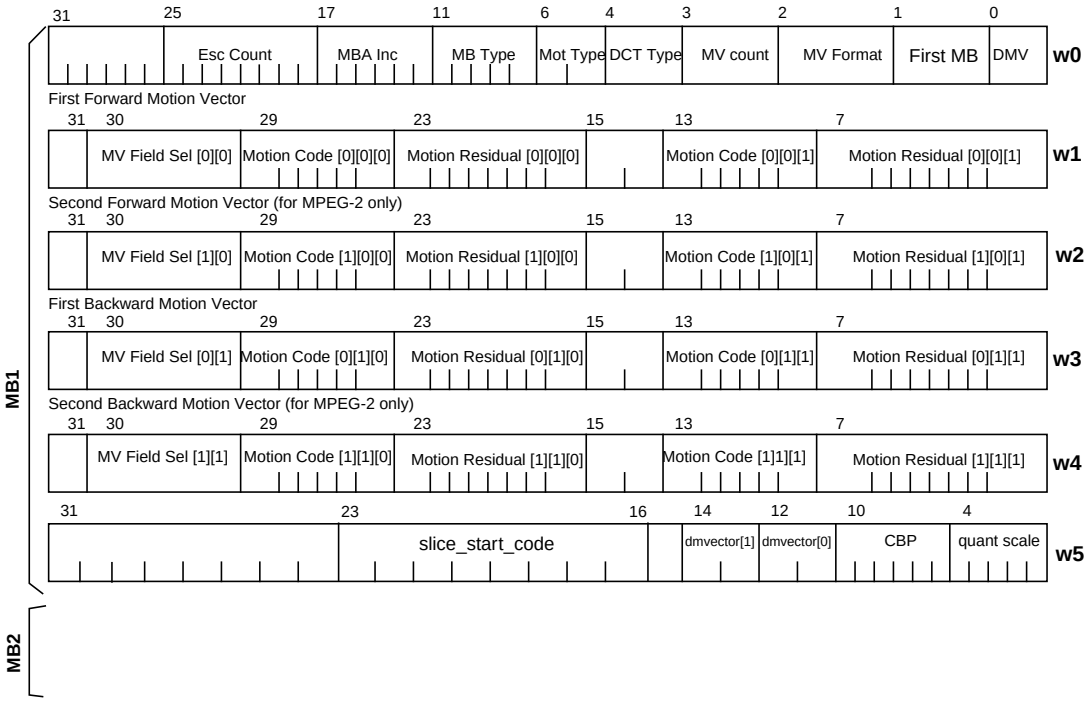


Figure 2: MPEG-2 Macro Block Header Output Format

Table 6: References for the MPEG-2 Macroblock Header Data

Item	Default Value	References from MPEG-2 Video Standard, IS 13818-2 Document
Esc Count	0	Section 6.2.5
MBA Inc	-	Section 6.2.5 and Table B-1
MB Type	Undefined	Section 6.2.5.1 and Tables B-2, B-3, and B-4; Only 5 Msb bits from the tables are used
MotType	Undefined	Section 6.2.5.1; Field or Frame motion type will be decided by the user
DCT Type	Undefined	Section 6.2.5.1
MV Count	Undefined	Tables 6-17 and 6-18. The MV Count value is one less than the value from the tables.
MV Format	Undefined	Tables 6-17 and 6-18
First MB	0	Set to ‘1’ for the first macroblock of a slice
DMV	Undefined	Tables 6-17 and 6-17

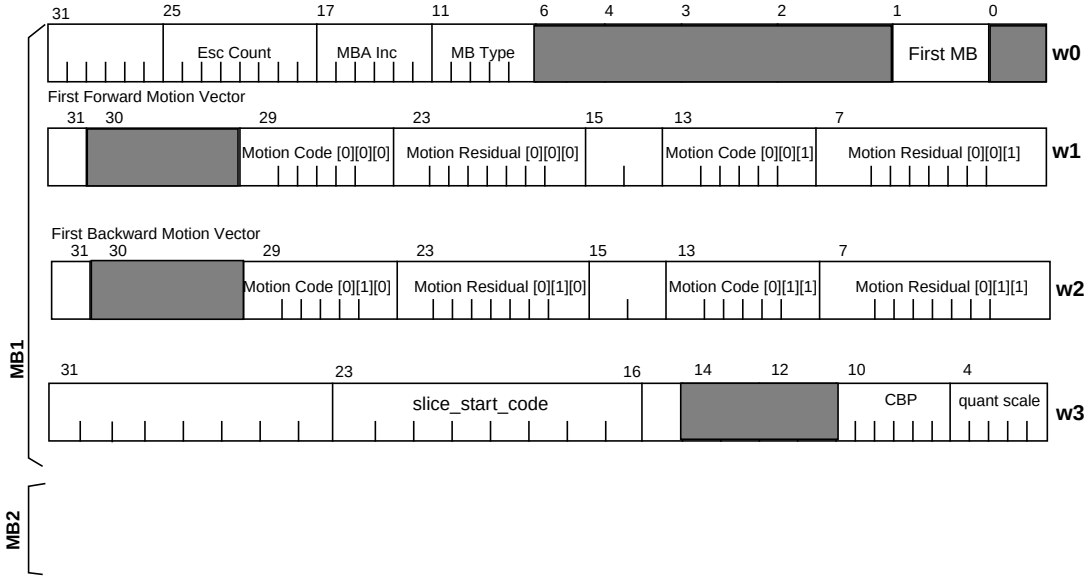
Table 6: References for the MPEG-2 Macroblock Header Data *...Continued*

Item	Default Value	References from MPEG-2 Video Standard, IS 13818-2 Document
MV Field Sel[0]0] to MV Field Sel[1][1]	Undefined	Section 6.2.5 and 6.2.5.2
Motion Code[0][0][0] to Motion Code[1][1][1]	Undefined	Section 6.2.5.2.1 and Table B-10
Motion Residual[0][0][0] to Motion Residual[1][1][1]	Undefined	Section 6.2.5.2.1; the corresponding rsize bits are extracted from the bitstream and stored as left justified; to get the final value shift the given number by (8 - corresponding rsize). The rsize values are stored in MP_VLD_PI register
Start Code	-	Copy of the internal start-code register; see Section 5. Register Descriptions.
dmvector[1] and dmvector[0]	Undefined	Section 6.2.5.2.1 and Table B-11; signed 2-bit integer from Table B11.
CBP	-	Section 6.2.5, 6.2.5.3 and Table B-9
Quant Scale	-	Section 6.2.5; 5 bits from bitstream; use Table 7-6 to compute the quant scale value.

MPEG-1 Parsing

For each MPEG-1 macroblock parsed by the VLD, four 32-bit words of macroblock header information will be output from the VLD. Refer to Figure 1.

The fields described in Figure 3 may or may not be valid depending upon the MPEG-1 video standard. See Table 7 for more details on the macroblock header data



Shaded areas represent bits not implemented in the MPEG-1 specification. These bits are written as zeros.

Figure 3: MPEG-1 Macro Block Header Output Format

Table 7: References for the MPEG-1 Macroblock Header Data

Item	Default value	References from IS 11172-2 document
Esc Count	0	Section 2.4.3.6
MBA Inc	-	Section 2.4.3.6
MB Type	Undefined	Section 2.4.3.6 and Tables B-2a to B2d
First MB	0	Set to '1' for the first macroblock of a slice
Motion Code[0][0][0] to Motion Code[0][1][1]	Undefined	Section 2.4.2.7 and Table B-4
Motion Residual[0][0][0] to Motion Residual[0][1][1]	Undefined	Section 2.4.2.7;the corresponding rsize bits are extracted from the bitstream and stored as left justified; to get the final value shift the given number by (8 - corresponding rsize). The rsize values are stored in MP_VLD_PI register.
Start Code	-	Copy of the internal start-code register; see Section 5. Register Descriptions.
CBP	-	Section 2.4.3.6 and Table B-3
Quant Scale	-	Section 2.4.2.7

The DCT coefficients associated with the macroblock are output to a separate memory area. Each DCT coefficient is represented as one 32 bit quantity (16 bits of run and 16 bits of level). For intra blocks, the DC term is expressed as 16 bits of DC size and a 16 bit value whose most significant bits (the number of bits used for DC level is determined by DC size) represent the DC level. Each block of DCT coefficients is terminated by a run value of 0xff.

Run-Level Output Data

The DCT coefficients associated with the macroblock are output to a separate memory area and each DCT coefficient is represented as one 32-bit quantity (16 bits of run and 16 bits of level). For intra blocks, the DC term is expressed as 16 bits of size and a 16-bit value whose most significant bits represent the DC level. The number of bits used for DC level is determined by the DC size)'. Each block of DCT coefficients is terminated by a run value of 0xFF.

3.3.3 Restart the VLD Parsing

The restart of the parsing command at the last decoded symbol position is done through the use of the VLD_BIT_CNT register to count the number of bits from the main memory VLD buffer that have been consumed. The VLD_BIT_CNT register can be initialized to zero by issuing the VLD_INIT command. It counts up when bits are shifted out of the VLD. This counter wraps around after reaching its maximum value.

3.4 Error Handling

The VLD can generate two types of errors. The VLD_MC_STATUS register has two VLD error flags.The VLD errors are 1) bitstream parsing error and 2) run-level overflow error. See Table 8 for details on the VLD error handling procedure.

3.4.1 Unexpected Start Code

When VLD encounters an unexpected start code, VLD sets the ‘Start Code Detected’ and ‘Bitstream Error’ flags (in VLD_MC_STATUS register). The start code value is left in the shift register (VLD_SR). VLD interrupts the CPU, if one of the corresponding interrupt bits in the VLD_IE register is enabled

3.4.2 RL Overflow

If the VLD encounters a situation where the last data for a macroblock is being emitted and the Run-Length code is not 0xFF, then the RL Overflow error flag is asserted and an interrupt is generated if the corresponding interrupt enable bit is set.

.

Table 8: VLD Error Handling

Cycle No.	Action	Remarks
i	VLD sets the appropriate error bit in the VLD_MC_STAUS register	The *vld_mc_error* signal is formed by OR’ing together all of the error bits in the VLD_MC_STATUS register is the Hence any MC error also drives the vld_mc_error signal high and the following error handling steps still apply
i to j	When the *vld_mc_error* signal is high, VLD completes any pending control or memory hwy. transactions. The valid data in the DMA output buffers will be flushed to the main memory. Then VLD asserts the *vld_ready_to_reset* signal and waits for the CPU to reset.	Any DMA transactions, once started, will not be aborted in the middle
k	If (vld_ready_to_reset) then the VLD interrupts the CPU.	Assumes k>j; otherwise it is jth cycle. The corresponding interrupt enable (IE) bit in the VLD_IE register is ‘1’ for the VLD to raise the interrupt.
l	CPU will perform the software reset.	Refer to Section 3.1 for the software reset procedure

3.4.3 Flush

The *flush_cmd* will be issued in two cases:

1. The bitstream contains error bits for a known amount of bytes and would like to terminate the decoding at a particular byte-offset of the bitstream buffer, and
2. when CPU decides to switch the bitstream at the start of a new slice-start-code in a new row.

The CPU will set the DMA_input_done_mode bit to ‘1’ in the VLD_CTL register for the first case.

The *flush_cmd* will be issued when the VLD stops after interrupting the CPU for the dma_input_done reason in the first case, and when the VLD stops after interrupting the CPU for the start_code_detected reason in the second case.

4. Application Notes

4.0.1 PNX1300 Series versus PNX15xx Series VLD

- The MPEG-2 Macroblock Header Output Format now differs in two ways: the First Forward Motion Vector bit[1] which was unused is now the “First Macro Block” bit, and the sixth word bits[23:16] now contain the Slice Start Code.
- The PNX1300 Series does not implement the “Parse Long” command.
- In the VLD_STATUS register, the PNX1300 Series does not implement the following bits:
  - Bit[6] RL Overflow
- In the VLD_CONTROL register, the PNX1300 Series does not implement the following bits:
  - Bit[16] Slice_strobe
  - Bit[15:8] Slice_start_code
  - Bit[2] DMA_input_done_mode
- In addition, the Little_Endian mode bit is on bit[1] in the PNX1300 Series; but it is bit[0] in this module.

5. Register Descriptions

5.1 PNX1300 Series and PNX15xx Series Register Differences

The current VLD is a compatible superset of the VLD that was implemented in the PNX1300 Series chip. Differences in the register definitions are noted in magenta text.

The PNX15xx Series implementation removed the RL/MBH write-back DMA channels. Differences from the current VLD implementation are noted in blue text.

The base address for the PNX15xx Series VLD module is 0x07 5000.

5.2 VLD Register Summary

Table 9: Register Summary

Offset	Symbol	Description
0x07 5000	VLD_COMMAND	Variable Length Decoder Command
0x07 5004	VLD_SR	VLD Shift Register (shadow)
0x07 5008	VLD_QS	Quantization Scale Code to be output by the VLD
0x07 500C	VPD_PI	VLD Picture Information
0x07 5010	VLD_MC_STATUS	VLD and MC Status register
0x07 5014	VLD_IE	VLD Interrupt Enable
0x07 5018	VLD_CTL	VLD Control register

Table 9: Register Summary *...Continued*

Offset	Symbol	Description
0x07 501C	VLD_INP_ADR	VLD Input Memory Address
0x07 5020	VLD_INP_CNT	VLD Input Count gives the number of bytes to be read from main memory
0x07 5024	VLD_MBH_ADR	VLD Macroblock Header Writeback Address
0x07 5028	VLD_MBH_CNT	VLD Macroblock Header Writeback Count
0x07 502C	VLD_RL_ADR	VLD Run-level Writeback Address
0x07 5030	VLD_RL_CNT	VLD Run-level Writeback Count
0x07 5034	VLD_BIT_CNT	VLD Bit Count gives the number of bits consumed by the VLD
0x07 5200	MC_PICINFO0	Macro-block height
0x07 5FF4	PD	Power Down Register
0x07 5FFC	MODULE ID	Module Identification Register

5.3 Register Table

Table 10: VLD Registers

Bit	Symbol	Access	Value	Description
Offset 0x07 5000	***VLD_COMMAND***			
31:12	Reserved	R	0	
11:8	Command	R/W	0	Command code of the VLD command to be executed 0x1 = Shift Bitstream by “Shift Count” bits 0x2 = Parse Macroblock 0x3 = Search for next Start Code 0x4 = Reset Variable Length Decoder 0x5 = Initialize VLD 0x6 = Search for the Start Code given in bits [7:0]. 0x7 = Parse Macroblock Row 0x8 = Flush Write FIFOs 0x9 = Parse Long
7:0	Mblock/Shift Count or start code	R/W	0	For the ‘Shift Bitstream’ command, only the lower 4-bit are used; the upper 4-bit should be set to 0. All 8 bits are used for the ‘Parse macroblocks’ and ‘Search for given start code’ commands. For Parse Long command these bits cannot be programmed to 0.
Offset 0x07 5004	***VLD_SR***			
31:15	Reserved	R	0	
15:0	Shift Register	R	NI	This register is a shadow of the VLD’s operational shift register and it allows the DSPCPU to access the bitstream through the VLD. Bits 15 through 0 are the current contents of the VLD shift register.
Offset 0x07 5008	***VLD_QS***			
31:5	Reserved	R	0	
4:0	Quant scale	R/W	NI	This register contains the quantization scale code to be output by the VLD until it is overridden by a macroblock quantizer scale code. The quantizer scale code is part of the macroblock header output.
Offset 0x07 500C	***VLD_PI***			

Table 10: VLD Registers *…Continued*

Bit	Symbol	Access	Value	Description
31:28	Vertical back. rsize	R/W	NI	Number of bits per backward vertical motion vector that are residual in the picture.
27:24	Horizontal back. rsize	R/W	NI	Number of bits per backward horizontal motion vector that are residual in the picture.
23:20	Vertical for. rsize	R/W	NI	Number of bits per forward vertical motion vector that are residual in the picture.
19:16	Horizontal for. rsize	R/W	NI	Number of bits per forward horizontal motion vector that are residual in the picture.
15:14	Reserved	R	0	
13	mpeg2mode	R/W	NI	1 = indicates if the current sequence is MPEG-2 0 = indicates if the current sequence is MPEG-1 (for error checking only)
12:7	Reserved	R	0	
6	mv_concealment	R/W	NI	1 = indicates forward motion vectors are coded in all intra macroblock headers of a picture. 0 = indicates forward motion vectors are not coded in all intra macroblock headers of a picture.
5	intra_vlc	R/W	NI	Use DCT table zero (intra_vlc = “0”)or one (intra_vlc = “1”)
4	frame_prediction_frame_dct	R/W	NI	If 1, motion_type = FRAME, and dct_type = 0. If 0, motion_type and dct_type follow the decoded values in the mb_header from the VLD. CPU should set it to 0 for Field Pictures and 1 for MPEG-1.
3:2	picture_structure	R/W	NI	1=top-field 2=bottom-field 3=frame picture 0=reserved.
1:0	picture_type	R/W	NI	1=I 2=P 3=B 0=D (MPEG-1 only)
Offset 0x07 5010	***VLD_MC_STATUS***			
31:16	Reserved	R	0	
15:7	Reserved	R/W	0	
6	RL overflow	R/W	0	Logic ‘1’ indicates Overflow of run/level values with in a block. Refer to Section 3.4 Error Handling for details on the error handling procedure. This bit is cleared by writing a logic ‘1’ to it.
5	DMA RL output done	R/W	0	Logic ‘1’ indicates that the Run Length data FIFO has been written to main memory. This bit is cleared by writing a logic ‘1’ to it.
4	DMA Header output done	R/W	0	Logic ‘1’ indicates that the Run Length data FIFO has been written to main memory. This bit is cleared by writing a logic ‘1’ to it.
3	DMA input done	R/W	0	Conditions for setting this bit depends on the value of the DMA_Input_Done field in the VLD_CTL register. Refer to Table 3 VLD Control and Section 3.3.1 for details. This bit is cleared by writing a logic ‘1’ to it.

Table 10: VLD Registers *…Continued*

Bit	Symbol	Access	Value	Description
2	Bitstream error	R/W	0	Logic '1' indicates VLD encountered an illegal Huffman code or an unexpected start code. Refer to Section 3.4 Error Handling for details on the error handling procedure. This bit is cleared by writing a logic '1' to it.
1	Start code detected	R/W	0	Logic '1' indicates VLD encountered 0x000001 while executing current command. This bit is cleared by writing a logic '1' to it.
0	VLD Command done	R/W	0	Logic '1' indicates successful completion of current command. This bit is cleared by issuing a new command.
Offset 0x07 5014	***VLD_IE***			
31:16	Reserved		0	
15:8	Reserved		0	-
7:0	VLD Int. Enables	R/W	0	Each of these bits enables the matching bit from the VLD_STATUS reg 0x010 to issue an IR to the CPU.
Offset 0x07 518	***VLD_CTL***			
31:17	Reserved		0	
16	slice_start_code_strobe	R/W	0	When CPU writes 1 into this field, VLD copies the value of slice_start_code into its internal register. CPU should do this only when the VLD is stopped. This bit is always read as 0.
15:8	slice_start_code	R/W	0	Slice start code when the VLD is restarted; the slice_start_code_strobe bit field must be set to '1' in order to update this field.
7:3	Reserved		0	
2	DMA-input-done-mode	R/W	0	When this bit is '0', VLD sets the DMA_INPUT_DONE flag (in VLD_MC_STATUS register) when the DMA_INP_CNT transitions from non-zero to zero. When this bit is '1', the same flag is set only with the additional condition that both highway input buffers are empty. The slice_start_code_strobe bit field must be set to '0' in order to update this field)
1	Reserved	R/W	1	
0	Little_Endian	R/W	1	Force the VLD to operate in Little Endian mode when '1'. When set to '0' the VLD operates in Big Endian mode. The slice_start_code_strobe bit must be set to '0' in order to update this bit.
Offset 0x07 501C	***VLD_INP_ADR***			
31:0	VLD Input Memory Address	R/W	0	Memory address from which VLD is reading (updated when DMA read transfer is completed). Must be 32-bit word aligned.
Offset 0x07 5020	***VLD_INP_CNT***			
31:15	Reserved		0	
14:0	VLD Input Count	R/W	0	Number of bytes to be read from main memory
Offset 0x07 5024	***VLD_MBH_ADR***			
31:0	MB Header Memory Address	R/W	0	Memory address to which the VLD writes macroblock headers when VLD_CTL[1] is set. Must be 32-bit word aligned.

Table 10: VLD Registers *…Continued*

Bit	Symbol	Access	Value	Description
Offset 0x07 5028	***VLD_MBH_CNT***			
31:15	Reserved			
14:0	MB Header Count	R/W		Number of MB Header bytes to be written to main memory when VLD_CTL[1] is set.
Offset 0x07 502C	***VLD_RL_ADR***			
31:0	VLD RL Writeback Address	R/W	0	Memory address to which the VLD writes Run Length data when VLD_CTL[1] is set. Must be 32-bit word aligned.
Offset 0x07 5030	***VLD_RL_CNT***			
31:15	Reserved		0	
14:0	VLD RL Count	R/W	0	Number of RL bytes to be written to main memory when VLD_CTL[1] is set.
Offset 0x07 50034	***VLD_BIT_CNT***			
31:18	Reserved		0	
17:0	Actual Bit Count	R	0	Bits consumed
Offset 0x07 5FF4	***POWER_DOWN***			
31	Power_Down	R/W	0	Power Down indicator 1 = Powerdown 0 = Power up
30:0	Reserved		0	
Offset 0x07 5FFC	***MODULE_ID***			
31:16	Module ID	R	0x014D	Variable Length Decoder Module ID register
15:12	Major Rev	R	0	Major Revision ID
11:8	Minor Rev	R	0	Minor Revision ID
7:0	Aperture Size	R	0	Aperture = 4 kB

Chapter 22: Digital Video Disc Descrambler

PNX15xx Series Data Book – Volume 1 of 1

Rev. 2 — 1 December 2004 Product data sheet

1. Introduction

The Digital Video Disc (DVD) Descrambler allows three major processes:

1. Authentication process.
2. Key Conversion process.
3. Main data descrambling process.

A DVD player system consists then of a DVD-ROM drive which accepts the DVD-ROM Disc and reads the information from the disc to the drive controller. The drive controller connects the disc transport with an interconnecting bus (PCI-XIO) and hence to the host system (PNX15xx Series). The host system interacts with the DVDD module to send authorization requests, receive authorization, replies, and transfer data between the DVD-ROM Drive and the DVDD module.

1.1 Functional Description

Only customers that have signed an NDA may be allowed to receive implementation and programming details. Please contact Philips Semiconductors, Inc. Nexperia Marketing group to obtain the DVD Descrambler Programming Supplement Document. A Non-Disclosure Agreement is required.

PHILIPS

Chapter 23: LAN100 — Ethernet Media Access Controller

PNX15xx Series Data Book – Volume 1 of 1

Rev. 2 — 1 December 2004 Product data sheet

1. Introduction

The Ethernet Media Access Controller (LAN100) of the PNX15xx Series enables applications to receive and transmit data over an Ethernet link.

1.1 Features

The LAN100 is a DMA-capable, 10/100 Mb/s Ethernet Media Access (MAC) Controller. It connects to an external Ethernet Physical Layer (PHY) chip using a standard Media Independent Interface (MII) or standard Reduced MII interface (RMII).

The LAN100 provides the following functions:

- Receives and transmits Ethernet packets via an external PHY chip
- Connects to an external PHY chip via standard MII or standard Reduced MII (RMII) interface
- Implements the MAC sublayer of IEEE standard 802.3
- Provides a flexible choice of FIFO buffers and on-chip host buses
- DMA and FIFO managers with scatter/gather DMA and FIFO of frame descriptors
- Memory traffic is optimized by buffering and prefetching
- Receive-packet filtering includes perfect address matching, hash table, imperfect filtering, and four pattern-match filters.
- Wake-on-LAN power management support allows system wake-up, using the receive filters or a magic frame detection filter
- Supports real-time traffic using time stamps
- Contains dual transmit descriptor buffers, one for real-time traffic, one for non-real-time traffic
- Supports Quality-of-Service (QoS) using a low-priority and a high-priority transmit queue
- Provides VLAN support
- Implements IEEE 802.3/clause 31 flow control for both receive and transmit.

2. Functional Description

2.1 Chip I/O and System Interconnections

Figure 1 presents a simplified view of the I/O interfaces and system interconnection.

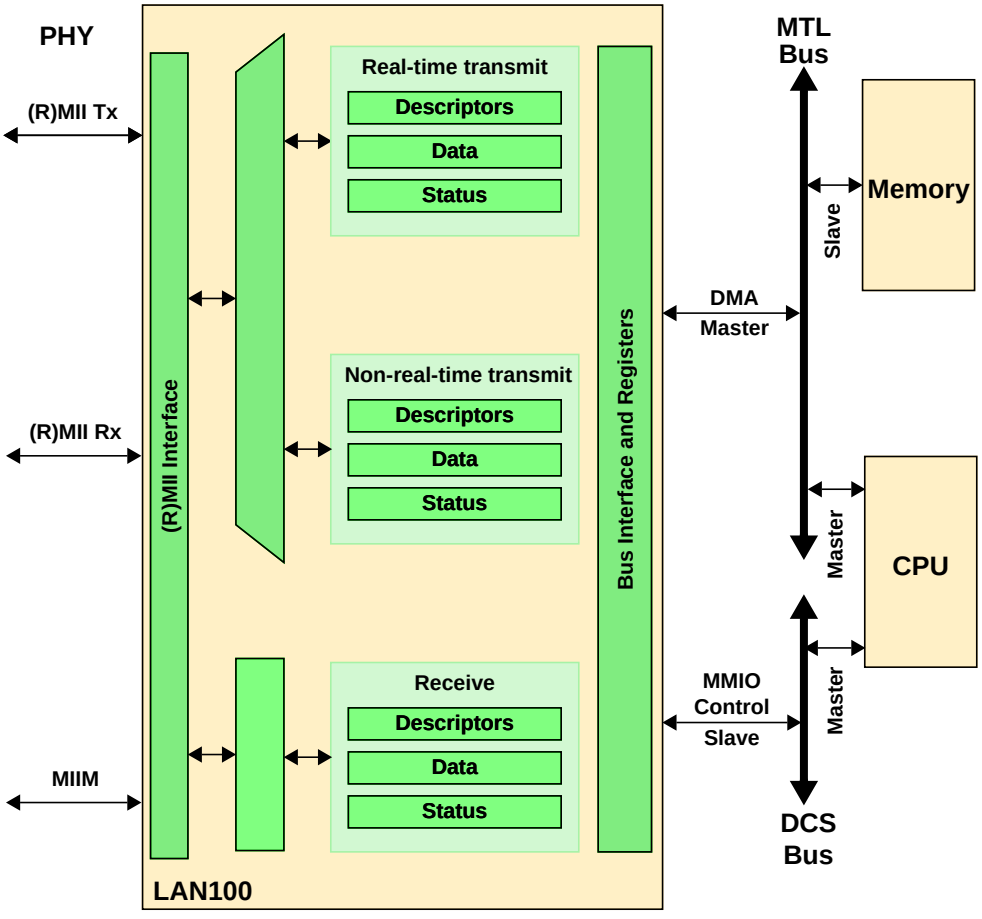


Figure 1: Simplified LAN100 I/O Block Diagram

On the left-hand side, the LAN100 is connected to the off-chip Ethernet PHY using the Media Independent Interface (MII) or Reduced Media Independent Interface (RMII), which includes transmit, receive, and management connections. On the right-hand side, the LAN100 is connected to the on-chip system buses:

- The MMIO control port of the LAN100 allows CPU access to the LAN100's internal registers via the internal DCS bus of the PNX15xx Series. The LAN100 MMIO port is a slave on the DCS bus.
- The Direct Memory Access (DMA) port of the LAN100 performs DMA via the internal MTL bus of the PNX15xx Series. The LAN100 can initiate transactions while it is a master on the MTL bus. The LAN100 has multiple DMA interfaces to allow both non-real-time and real-time transmit modes, and receive mode.

2.2 Functional Block Diagram

Figure 2 shows a more detailed block diagram of the main internal units of the LAN100. Primary components of the LAN100, shown as blocks outlined in black, are described below. The Transmit Datapath and Receive Datapath are shown with unoutlined background colors.

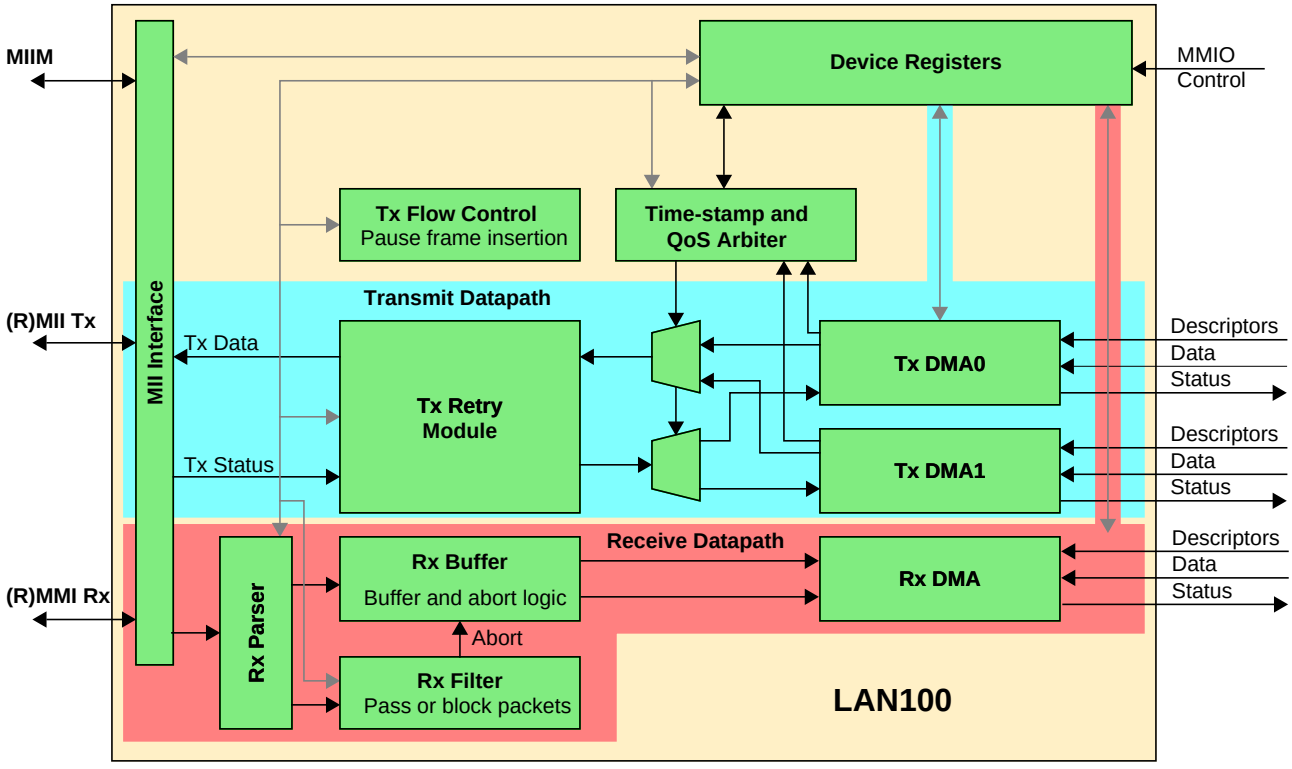


Figure 2: LAN100 Functional Block Diagram

The registers provide MMIO acces to the LAN100. They interface to the transmit and receive data-paths, and to the MII Interface.

The MII Interface connects the LAN100 to the off-chip PHY.

The Transmit Datapath has two transmit DMA managers, DMA0 and DMA1, which read descriptors and data from memory and write status to memory. In real-time mode, DMA0 can be used to handle real-time transmissions and DMA1 can be used to handle non-real-time transmissions. In Quality-of-Service (QoS) mode, DMA0 has the lowest priority and DMA1 has the highest priority.

Transmission from both of the transmit DMA managers is mediated by arbitration logic, including:

- Multiplexers to select one of the transmit DMA managers
- Transmit Retry module to handle Ethernet retry and abort conditions
- Transmit Flow Control module to insert Ethernet pause frames where needed.

The Receive Datapath includes:

- Receive parser, which detects packet types by parsing part of the packet header
- Receive filter, which can filter out certain Ethernet packets, applying different filtering schemes
- Receive buffer, which implements a delay for receive packets to allow the filter to filter out certain packets before storing them to memory
- Receive DMA manager, which reads descriptors from memory and writes data and status to memory.

2.3 Description

The Ethernet Media Access Controller (LAN100) and associated device driver software offer the functionality of the Media Access Control (MAC) sublayer of the data link layer in the OSI reference model (see IEEE Standard 802.3 [1]).

The MAC sublayer transmits and receives frames to and from the next higher protocol level, the MAC client layer, typically the Logical Link Control sublayer. The device driver software implements the interface to the MAC client layer. It sets up MMIO registers in the LAN100, maintains FIFOs of packets in memory, and receives results back from the LAN100, typically via interrupts.

When packets are transmitted, packet fields can be concatenated using the scatter/gather DMA functionality of the LAN100 to avoid unnecessary data copying. The hardware can add the preamble and start frame delimiter fields, and can optionally add the CRC under program control. Or, software can partially set up the Ethernet frames by concatenating the destination address field, source address field, the length/type field, the MAC client data field, and optionally, the CRC in the frame check sequence field of the Ethernet frame.

When a packet is received, the LAN100 strips the preamble and start frame delimiter and passes the rest of the packet - the rest is the Ethernet frame - to the device driver, including destination address, source address, length/type field, MAC client data, and frame check sequence.

Apart from its basic packet management functions, the LAN100 contains receive and transmit DMA managers that control receive- and transmit-data streams. Frames are passed via descriptor FIFOs (arrays) located in host memory, so that the hardware can process many packets without software or CPU support. Packets can consist of multiple fragments that are accessed with scatter/gather DMA. The DMA managers optimize memory bandwidth by prefetching and buffering.

The LAN100 is connected to the MTL bus using multiple DMA interfaces that perform data buffering to adapt the data burst rate of the MTL bus to the data rate required for the Ethernet protocol.

The LAN100 provides MMIO registers via a slave interface to the DCS bus to allow software to access the internal registers of the LAN100.

Real-time traffic is supported using two transmit queues:

- A real-time queue sends frames at the time specified in transmit time-stamps

- A non-real-time queue sends packets immediately.

The two transmit queues can also be configured to support a generic Quality-of-Service (QoS) system: a high-priority queue provides high quality of service for packets; a low priority queue runs when possible.

A receive time-stamp indicates the exact moment in time a packet has been received.

Receive blocking filters are used to identify received packets that are not addressed to this Ethernet station, so that they can be discarded. The Rx filters include a perfect address filter, a hash filter, and four pattern-matching filters.

Wake-on-LAN power management support makes it possible to wake the system up from a power-down state (a state in which some of the clocks are switched off) when wake-up frames are received over the LAN. Wake-up frames are recognized by the receive filtering modules or by a Magic Frame detection technology. System wake-up occurs by triggering an interrupt.

An interrupt logic block raises and masks interrupts and keeps track of the cause of interrupts. The interrupt block sends interrupt request signals to the CPU. Interrupts can be enabled, cleared, and set by software.

Support for IEEE 802.3/clause 31 flow control is implemented in the Flow Control block. Receive flow-control frames are automatically handled by the LAN100. Transmit flow-control frames can be initiated by software. In half-duplex mode, the flow-control module will generate back pressure by sending out continuous preamble only interrupted by pauses to prevent the jabber limit.

The LAN100 has both a standard IEEE 802.3/clause 22 Media Independent Interface (MII) bus and a Reduced Media Independent Interface (RMII) to connect to an external Ethernet PHY chip [3]. MII or RMII mode can be selected by a bit in the LAN100 Command register. The standard nibble-wide MII interface allows a low-speed data connection to the PHY chip at speeds of 2.5 MHz at 10 Mbit/s or 25 MHz at 100 Mbit/s. The RMII interface allows connection to the PHY with low pin-count and double-speed data clock. Registers in the PHY chip are accessed via the MMIO interface through the serial management connection of the MII bus operating at 2.5 MHz.

3. Register Descriptions

3.1 Register Summary

The base address for LAN100 MMIO registers begins at offset 0x07,2000 with respect to MMIO_BASE.

After a hard or soft reset via the RegReset bit of the Command register, all bits in all registers are reset to 0, unless shown otherwise in Table 2.

Reading write-only registers will return a read error. Writing read-only registers will return a write error. Unused or resrved bits must be ignored on reads and written as 0.

The register map of the LAN100 includes MII Interface registers and registers for controlling DMA transfers, flow control and filtering. It also includes interrupt control and module ID registers. Table 1 provides a summary of all LAN100 registers.

Table 1: LAN100 MMIO Register Map

Offset	Name	R/W	Function
MII Interface			
0x07 2000	MAC1	R/W	MAC Configuration register 1
0x07 2004	MAC2	R/W	MAC Configuration register 2
0x07 2008	IPGT	R/W	Back-to-back Inter-packet Gap register
0x07 200C	IPGR	R/W	Non B2B Inter-packet Gap register
0x07 2010	CLRT	R/W	Collision Window / Retry register
0x07 2014	MAXF	R/W	Maximum frame register
0x07 2018	SUPP	R/W	PHY Support register
0x07 201C	TEST	R/W	Test register
0x07 2020	MCFG	R/W	MII Management Configuration register
0x07 2024	MCMD	R/W	MII Management Command register
0x07 2028	MADR	R/W	MII Management Address register
0x07 202C	MWTD	WO	MII Management Write Data register
0x07 2030	MRDD	RO	MII Management Read Data register
0x07 2034	MIND	RO	MII Management Indicators register
0x07 2038			Reserved
0x07 203C			Reserved
0x07 2040	SA0	R/W	Station Address 0 register
0x07 2044	SA1	R/W	Station Address 1 register
0x07 2048	SA2	R/W	Station Address 2 register
0x07 204C to 0x07 20FC			Reserved
Control Registers			
0x07 2100	Command	R/W	Command register
0x07 2104	Status	RO	Status register
0x07 2108	RxDescriptor	R/W	Receive Descriptor Base Address register
0x07 210C	RxStatus	R/W	Receive Status Base Address register
0x07 2110	RxDescriptorNumber	R/W	Number of Receive Descriptors register
0x07 2114	RxProduceIndex	RO	Receive Produce Index register
0x07 2118	RxConsumeIndex	R/W	Receive Consume Index register
0x07 211C	TxDescriptor	R/W	Non Real-Time Transmit Descriptor Base Address register
0x07 2120	TxStatus	R/W	Non Real-Time Transmit Status Base Address register

Table 1: LAN100 MMIO Register Map

Offset	Name	R/W	Function
0x07 2124	TxDescriptorNumber	R/W	Non Real-Time Number of Transmit Descriptors (minus one encoded) register
0x07 2128	TxProduceIndex	R/W	Non Real-Time Transmit Produce Index register
0x07 212C	TxConsumeIndex	RO	Non Real-Time Transmit Consume Index register
0x07 2130	TxRtDescriptor	R/W	Real-time Transmit Descriptor Base Address register
0x07 2134	TxRtStatus	R/W	Real-time Transmit Status Base Address register
0x07 2138	TxRtDescriptorNumber	R/W	Number of Real-time Transmit Descriptors register (minus-one encoded)
0x07 213C	TxRtProduceIndex	R/W	Real-time Transmit Produce Index register
0x07 2140	TxRtConsumeIndex	RO	Real-time Transmit Consume Index register
0x07 2144	BlockZone	R/W	Block Zone for Real-time Mode register
0x07 2148	QoSTimeout	R/W	Time-out for QoS Mode register
0x07 214C			Reserved
0x07 2150			Reserved
0x07 2154			Reserved
0x07 2158	TSV0	RO	First Part of MII Interface Transmit Status register
0x07 215C	TSV1	RO	Second Part of MII Interface Transmit Status register
0x07 2160	RSV	RO	Receive Status from MII Interface register
0x07 2164 to 0x07 216C			Reserved
0x07 2170	FlowControlCounter	R/W	Flow Control Mirror and Pause Timer register
0x07 2174	FlowControlStatus	RO	Flow Control Internal Mirror Counter
0x07 2178 to 0x07 21F8			Reserved
0x07 21FC	GlobalTimeStamp	RO	Global Time-stamp Counter
Rx Filter Registers			
0x07 2200	RxFilterCtrl	R/W	Receive Filter Control
0x07 2204	RxFilterWoLStatus	RO	Receive Filter Wake-on-LAN Status
0x07 2208	RxFilterWoLClear	WO	Receive Filter Wake-on-LAN Status Clear
0x07 220C	PatternMatchJoin	R/W	Join method for Pattern Matching Filters
0x07 2210	HashFilterL	R/W	Bit 31:0 of Hash Table
0x07 2214	HashFilterH	R/W	Bit 63:32 of Hash Table
0x07 2218 to 0x07 222C			Reserved for Hash Table Extension

Table 1: LAN100 MMIO Register Map

Offset	Name	R/W	Function
0x07 2230	PatternMatchMask0L	R/W	Bit 31:0 of Pattern Match 0
0x07 2234	PatternMatchMask0H	R/W	Bit 63:32 of Pattern Match 0
0x07 2238	PatternMatchCRC0	R/W	CRC Value for Pattern Match 0
0x07 223C	PatternMatchSkip0	R/W	Skip Bytes for Pattern Match 0
0x07 2240	PatternMatchMask1L	R/W	Bit 31:0 of Pattern Match 1
0x07 2244	PatternMatchMask1H	R/W	Bit 63:32 of Pattern Match 1
0x07 2248	PatternMatchCRC1	R/W	CRC Value for Pattern Match 1
0x07 224C	PatternMatchSkip1	R/W	Skip Bytes for Pattern Match 1
0x07 2250	PatternMatchMask2L	R/W	Bit 31:0 of Pattern Match 2
0x07 2254	PatternMatchMask2H	R/W	Bit 63:32 of Pattern Match 2
0x07 2258	PatternMatchCRC2	R/W	CRC Value for Pattern Match 2
0x07 225C	PatternMatchSkip2	R/W	Skip Bytes for Pattern Match 2
0x07 2260	PatternMatchMask3L	R/W	Bit 31:0 of Pattern Match 3
0x07 2264	PatternMatchMask3H	R/W	Bit 63:32 of Pattern Match 3
0x07 2268	PatternMatchCRC3	R/W	CRC Value for Pattern Match 3
0x07 226C	PatternMatchSkip3	R/W	Skip Bytes for Pattern Match 3
0x07 2270 to 0x07 2FDC			Reserved
Standard Registers			
0x07 2FE0	IntStatus	RO	Interrupt Status register
0x07 2FE4	IntEnable	R/W	Interrupt Enable register
0x07 2FE8	IntClear	WO	Interrupt Clear register
0x07 2FEC	IntSet	WO	Interrupt Set register
0x07 2FF0			Reserved
0x07 2FF4	PowerDown	R/W	Power-down register
0x07 2FF8			Reserved
0x07 2FFC	ModuleID	RO	Module ID

In QoS mode, that is, when the EnableQoS bit of the Command register is set, the real-time transmit registers correspond to the registers of the low-priority transmit channel and the non-real-time transmit registers correspond to the registers of the high priority transmit channel.

3.2 Register Definitions

This section defines the bits of the individual LAN100 registers. The MII Interface registers are mapped to addresses in the range 0x07 2000 – 0x07 20FC. For more information about the MII Interface registers than is provided here, please refer to [2].

Table 2: LAN100 Registers

Bit	Symbol	Access	Value	Description
MII Interface Registers				
Offset 0x07 2000	*MAC Configurationuration Register 1 (MAC1)*			
31:16	-	-	0	Unused
15	SOFT_RESET	R/W	1	Setting this bit puts all modules within the MII Interface into the reset state. MII Interface registers are not reset. It has no effect upon LAN100 components other than the MII Interface. Clearing this bit restores the MII Interface to operation. Its default value is 1 (reset).
14	SIMULATION_RESET	R/W	0	Setting this bit will reset the random number generator within the Transmit function of the MII Interface.
13:12	-	-	0	Unused
11	RESET_PEMCS/Rx	R/W	0	Setting this bit puts the MAC Control Sublayer / Rx domain logic into the reset state.
10	RESET_PERFUN	R/W	0	Setting this bit puts the Receive Function logic in the reset state.
9	RESET_PEMCS_Tx	R/W	0	Setting this bit puts the MAC Control Sublayer / Tx domain logic in the reset state.
8	RESET_PETFUN	R/W	0	Setting this bit puts the MII Interface Transmit function logic in the reset state.
7:5	-	-	0	Unused
4	LOOPBACK	R/W	0	Setting this bit causes the MAC Transmit interface to be looped backed to the MAC Receive interface. Clearing this bit results in normal operation.
3	TX_FLOW_CONTROL	R/W	0	When set, PAUSE Flow Control frames are allowed to be transmitted. When cleared, Flow Control frames are blocked.
2	RX_FLOW_CONTROL	R/W	0	When set, the MAC acts upon received PAUSE Flow Control frames. When cleared, received PAUSE Flow Control frames are ignored.
1	PASS_ALL_RECEIVE_FRAMES	R/W	0	When set, the MAC will indicate PASS CURRENT RECEIVE FRAME for all frames regardless of type (normal vs. Control). When cleared, the MAC deasserts PASS CURRENT RECEIVE FRAME for valid Control frames.
0	RECEIVE_ENABLE	R/W	0	Set this bit to enable receiving frames. Internally, the MAC synchronizes this control bit to the incoming receive stream and outputs SYNCHRONIZED RECEIVE ENABLE, to be used by the host system to qualify receive frames.
Offset 0x07 2004	*MAC Configuration Register 2 (MAC2)*			
31:16	-	-	0	Unused
14	EXCESS_DEFER	R/W	0	When set, the MAC will defer to carrier indefinitely as per the Standard [1]. When cleared, the MAC will abort when the excessive deferral limit is reached, and will provide feedback to the host system.
13	BACK_PRESSURE	R/W	0	When set, the MAC after incidentally causing a collision during back pressure will immediately retransmit without backoff, reducing the chance of further collisions and ensuring transmit packets get sent.

Table 2: LAN100 Registers *…Continued*

Bit	Symbol	Access	Value	Description
12	NO_BACKOFF	R/W	0	When set, the MAC will immediately retransmit following a collision rather than using the Binary Exponential Backoff algorithm as specified in the Standard [1].
11:10	-	-	0	Unused
9	LONG_PREAMBLE_ ENFORCEMENT	R/W	0	When set, the MAC only allows receive packets which contain preamble fields less than 12 bytes in length. When cleared, the MAC allows any length preamble as per the Standard [1].
8	PURE_PREAMBLE_ ENFORCEMENT	R/W	0	When set, the MAC will verify the content of the preamble to ensure it contains 0x55 and is error-free. Packets with errors in the preamble are discarded.
7	AUTO_DETECT_PAD_ ENABLE	R/W	0	Set this bit to cause the MAC to automatically detect the type of frame, either tagged or un-tagged, by comparing the two octets following the source address with 0x8100 (VLAN Protocol ID) and pad accordingly. For more information refer to the MII Interface documentation. [2]
6	VLAN PAD ENABLE	R/W	0	Set this bit to cause the MAC to pad all short frames to 644 bytes and append a valid CRC. For more information, refer to the MII Interface documentation. [2]
5	PAD_CRC_ENABLE	R/W	0	Set this bit to have the MAC pad all short frames. Clear this bit if frames presented to the MAC have a valid length. This bit is used in conjunction with AUTO_DETECT_PAD_ENABLE and VLAN_PAD_ ENABLE.
4	CRC_ENABLE	R/W	0	Set this bit to append a cyclic redundancy check (CRC) to every frame, whether padding was required or not. This bit must be set if PAD_CRC_ENABLE is set. Clear this bit if frames presented to the MAC contain a CRC.
3	DELAYED_CRC	R/W	0	This bit determines the number of bytes, if any, of proprietary header information that exist on the front of IEEE 802.3 frames. For more information refer to the MII Interface documentation. [2]
2	HUGE_FRAME_ ENABLE	R/W	0	When set, frames of any length can be transmitted and received.
1	FRAME_LENGTH_ CHECKING	R/W	0	When set, both transmit and receive frame lengths are compared to the length/type field. If the length/type field represents a length, then the check is performed. Mismatches are reported on the Transmit/Receive Statistics Vector.
0	FULL_DUPLEX	R/W	0	When set, the MAC operates in full-duplex mode. Disable for half-duplex operation.
Offset 0x07 2008	*Back-to-Back Inter-Packet-Gap Register (IPGT)*			
31:7	-	-	0	Unused

Table 2: LAN100 Registers *…Continued*

Bit	Symbol	Access	Value	Description
6:0	BACK_TO_BACK_ INTER_PACKET_GAP	R/W	0	This is a programmable field representing the nibble time offset of the minimum possible period between the end of any transmitted packet, to the beginning of the next. In full-duplex mode, the register value should be the desired period in nibble times minus 3. In half-duplex mode, the register value should be the desired period in nibble times minus 6. In full-duplex, the recommended setting is 0x15 (21 decimal), which represents the minimum IPG of 0.96 ms (in 100 Mb/s) or 9.6 ms (in 10 Mb/s). In half-duplex the recommended setting is 0x12 (18 decimal), which also represents the minimum IPG of 0.96 ms (in 100 Mb/s) or 9.6 ms (in 10 Mb/s).
Offset 0x07 200C	*Non Back-to-Back Inter-Packet-Gap Register (IPGR)*			
31:15	-	0	-	Unused
14:8	NON_BACK_TO_ BACK_INTER_ PACKET_GAP_PART_1	0	R/W	This is a programmable field representing the optional carrierSense window referenced in IEEE 802.3 section 4.2.3.2.1 titled “Carrier Deference”. If carrier is detected during the timing of IPGR1, the MAC defers to carrier. If, however, carrier becomes active after IPGR1, the MAC continues timing IPGR2 and transmits, knowingly causing a collision, thus ensuring fair access to the medium. Its range of values is 0x0 to IPGR2.
7	-	0	-	Unused
6:0	NON_BACK_TO_ BACK_INTER_ PACKET_GAP_PART_2	0x12	R/W	This is a programmable field representing the Non-Back-to-Back Inter-Packet-Gap. The default is 0x12 (18 decimal), which represents the minimum IPG of 0.96 ms (in 100 Mb/s) or 9.6 ms (in 10 Mb/s).
Offset 0x07 2010	*Collision Window / Retry Register (CLRT)*			
31:14	-	-	0	Unused
13:8	COLLISION_WINDOW	R/W	0x37	This is a programmable field representing the slot time or collision window during which collisions occur in properly configured networks. Since the collision window starts at the beginning of transmission, the preamble and SFD is included. Its default of 0x37 (55 decimal) corresponds to the count of frame bytes at the end of the window.
7:4		-	0	Unused
3:0	RETRANSMISSION_ MAXIMUM	R/W	0xF	This is a programmable field specifying the number of retransmission attempts following a collision before aborting the packet because of excessive collisions. The Standard specifies the attemptLimit to be 0xF (15 decimal).
Offset 0x07 2014	*Maximum Frame Register (MAXF)*			
31:16	-	-	0	Unused
15:0	MAXIMUM_FRAME_ LENGTH	R/W	0x0600	This field’s reset value is 0x0600, which represents a maximum receive frame of 1536 octets. An untagged maximum size Ethernet frame is 1518 octets. A tagged frame adds four octets for a total of 1522 octets. If a shorter maximum length restriction is desired, program this 16-bit field.
Offset 0x07 2018	*PHY Support Register (SUPP)*			
The SUPP register is only relevant if an SMII, RMII, PMD or ENDEC interface is provided to the PHY.				
31:16	-	-	0	Unused

Table 2: LAN100 Registers *…Continued*

Bit	Symbol	Access	Value	Description
15	RESET_PESMII	R/W	0	This bit resets the Serial MII logic.
14:13	-	-	0	Unused
12	PHY_MODE	R/W	1	This bit configures the Serial MII logic with the connecting SMII device type. Set this bit when connecting to a SMII PHY. Clear this bit when connecting to a SMII MAC. When MAC is selected, the SMII will operate at 100 Mb/s, full duplex.
11	RESET_PERMII	R/W	0	This bit resets the Reduced MII logic.
10:9	-	-	0	Unused
8	SPEED	R/W	0	This bit configures the Reduced MII logic for the current operating speed. When set, 100 Mb/s mode is selected. When cleared, 10 Mb/s mode is selected.
7	RESET_PE100X	R/W	0	This bit resets the PE100X module which contains the 4B/5B symbol encipher/decipher logic. This effects the PE100X module only.
6	FORCE_QUIET	R/W	0	When set, transmit data is quieted, which allows the contents of the cipher to be output. When cleared, normal operation is enabled. Effects PE100X module only.
5	NO_CIPHER	R/W	0	When set, the raw transmit 5B symbols are transmitted without ciphering. When cleared, normal ciphering occurs. Effects PE100X module only.
4	DISABLE_LINK_FAIL	R/W	0	When set, the 330ms Link Fail timer is disabled allowing for shorter simulations. Removes the 330 mS link-up time before reception of streams is allowed. When cleared, normal operation occurs. Effects PE100X module only.
3	RESET_PE10T	R/W	0	This bit resets the PE10T module which converts MII nibble streams to the serial bit stream of 10T transceivers. Effects PE10T module only.
2	-	-	0	Unused
1	ENABLE_JABBER_PROTECTION	R/W	0	This bit enables the Jabber Protection logic within the PE10T in ENDEC mode. *Jabber* is the condition where a transmitter is stuck on for longer than 50ms to prevent other stations from transmitting. Effects PE10T module only.
0	BIT_MODE	R/W	0	When set, the MAC is in 10BASE-T ENDEC mode, which changes decodes (such as EXCESS_DEFER) to be based on the bit clock rather than the nibble clock.
Offset 0x07 201C	*Test Register (TEST)*			
31:3	-	-	0	Unused
2	TEST_BACKPRESSURE	R/W	0	Setting this bit will cause the MAC to assert back pressure on the link. Back pressure causes the preamble to be transmitted, raising carrier sense. A transmit packet from the system will be sent during back pressure.
1	TEST_PAUSE	R/W	0	This bit causes the MAC Control sublayer to inhibit transmissions, just as if a PAUSE Receive Control frame with a non-zero pause time parameter was received.
0	SHORTCUT_PAUSE_QUANTA	R/W	0	This bit reduces the effective PAUSE Quanta from 64 byte-times to 1 byte-time.

Table 2: LAN100 Registers *…Continued*

Bit	Symbol	Access	Value	Description
Offset 0x07 2020	*MII Mgmt Configuration (MCFG)*			
31:16	-	-	0	Unused
15	RESET_MII_MGMT	R/W	0	For more information refer to the MII Interface documentation [2].
14:5	-	-	0	Unused
4:2	CLOCK_SELECT	R/W	0	This field is used by the clock divide logic to create the MII Management Clock (MDC) which IEEE 802.3u defines to be no faster than 2.5 MHz. Some PHYs support clock rates up to 12.5 MHz, however. For more information refer to the MII Interface documentation [2].
1	SUPPRESS_ PREAMBLE	R/W	0	For more information refer to the MII Interface documentation [2].
0	SCAN_INCREMENT	R/W	0	For more information refer to the MII Interface documentation [2].
Offset 0x07 2024	*MII Mgmt Command (MCMD)*			
31:2	-	-	0	Unused
1	SCAN	R/W	0	This bit causes the MII Management module to perform read cycles continuously. This is useful for monitoring the Link Fail timer, for example.
0	READ	R/W	0	This bit causes the MII Management module to perform a single read cycle. The read data is returned in Register MRDD (MII Mgmt Read Data).
Offset 0x07 2028	*MII Mgmt Address (MADR)*			
31:13	-	-	0	Unused
12:8	PHY_ADDRESS	R/W	0	This field represents the 5-bit PHY address field of Management cycles. Up to 31 PHYs can be addressed (0 is reserved).
7:5	-	-	0	Unused
4:0	REGISTER_ADDRESS	R/W	0	This field represents the 5-bit Register Address field of Management cycles. Up to 32 registers can be accessed.
Offset 0x07 202C	*MII Mgmt Write Data (MWTD)*			
31:16	-	-	0	Unused
15:0	WRITE_DATA	WO	0	When written, an MII Management write cycle is performed using the 16-bit data and the pre-configured PHY and Register addresses from Register (0x0A).
Offset 0x07 2030	*MII Mgmt Read Data (MRDD)*			
31:16	-	-	0	Unused
15:0	READ_DATA	RO	0	Following a MII Management Read Cycle, the 16-bit data can be read from this location.
Offset 0x07 2030	*MII Mgmt Read Data (MRDD)*			
31:3	-	-	0	Unused
2	NOT_VALID	RO	0	When set, this bit indicates the MII Management Read cycle has not completed, and the Read Data is not yet valid.
1	SCANNING	RO	0	When set, this bit indicates a scan operation (continuous MII Management Read cycles) is in progress.

Table 2: LAN100 Registers *…Continued*

Bit	Symbol	Access	Value	Description
0	BUSY	RO	0	When set, this bit indicates the MII Management module is currently performing an MII Management read or write cycle.
Offset 0x07 2040	*Station Address (SA0)*			
31:16	-	-	0	Unused
15:8	STATION_ADDRESS_1ST_OCTET	R/W	0	This field holds the first octet of the station address.
7:0	STATION_ADDRESS__2ND_OCTET	R/W	0	This field holds the second octet of the station address.
Offset 0x07 2044	*Station Address (SA1)*			
31:16	-	-	0	Unused
15:8	STATION_ADDRESS_3RD_OCTET	R/W	0	This field holds the third octet of the station address.
7:0	STATION_ADDRESS_4TH_OCTET	R/W	0	This field holds the fourth octet of the station address.
Offset 0x07 2048	*Station Address (SA2)*			
31:16	-	-	0	Unused
15:8	STATION_ADDRESS_5TH_OCTET	R/W	0	This field holds the fifth octet of the station address.
7:0	STATION_ADDRESS_6TH_OCTET	R/W	0	This field holds the sixth octet of the station address.
Offset 0x07 2100	*Command Register (Command)*			
31:12	-	-		Unused
11	EnableQoS	R/W	0	When set, the arbiter operates in QoS mode, in which the real-time transmit channel has low priority and the non-real-time channel has high priority
10	FullDuplex	R/W	0	When set, indicates full-duplex operation.
9	RMII	R/W	0	When set, indicates RMII mode; if clear, then MII mode.
8	TxFlowControl	R/W	0	Enable IEEE 802.3 / clause 31 flow control sending pause frames in full duplex and continuous preamble in half duplex.
7	PassRxFilter	R/W	0	When set to ‘1’, disables receive filtering, i.e., all packets received are written to memory.
6	PassRuntFrame	R/W	0	When set to ‘1’, runt frame packets smaller than 64 bytes are passed to memory unless they have a CRC error. If set to ‘0’, runt frames are filtered out.
5	RxReset	WO	0	If set, reset the Receive Datapath.
4	TX RESET	WO	0	If set, reset the Transmit Datapath.
3	RegReset	WO	0	If set, reset all datapaths and the host registers. The MII Interface must be reset separately.
2	TxRtEnable	R/W	0	Enable the real-time Transmit Datapath.
1	TxEnable	R/W	0	Enable the non-real-time Transmit Datapath.
0	RxEnable	R/W	0	Enable the Receive Datapath.
Offset 0x07 2104	*Status Register (Status)*			

Table 2: LAN100 Registers *…Continued*

Bit	Symbol	Access	Value	Description
The values represent the status of the three channels/datapaths. In case the status is 1 the channel is active meaning it: • is enabled and the Rx/TxRt/TxEnable bit is set in the Command register or it just got disabled while still transmitting or receiving a frame • and for transmit channels the transmit queue is not empty i.e. ProduceIndex != ConsumeIndex • and for the receive channel the receive queue is not full i.e. ProduceIndex != ConsumeIndex - 1 The status transitions from active to inactive if the channel is disabled by software resetting the Tx/Rt/TxRtEnable bit in the Command register and if the channel has committed the status and data of the current frame to memory. The status also transition to inactive if the transmit queue is empty or if the receive queue is full and status and data have been committed to memory.				
31:3	-	-		Unused
2	TxRtStatus	RO		If ‘1’, the real-time transmit datapath is active. If ‘0’, the channel is inactive.
1	TxStatus	RO		If ‘1’, non-real-time transmit datapath is active. If ‘0’, the channel is inactive.
0	RxStatus	RO		If ‘1’, the receive datapath is active, if ‘0’, the channel is inactive.
Offset 0x07 2108	*Receive Descriptor Base Address Register (RxDescriptor)*			
The receive descriptor base address is a byte address aligned to a word boundary, (i.e. the two LSBs of the address are fixed to 0). The register contains the lowest address in the array of descriptors.				
31:2	RxDescriptor	R/W	0	MSBs of receive descriptor base address.
1:0	-	RO	0	Fixed to 0.
Offset 0x07 210c	*Receive Status Base Address Register (RxStatus)*			
The receive status base address is a byte address aligned to a double word boundary i.e. LSB 2:0 are fixed to 3’b000.				
31:3	RxStatus	R/W	0	MSBs of receive status base address.
2:0	-	RO	0	Fixed to 0.
Offset 0x07 2110	*Receive Number of Descriptors Register (RxDescriptorNumber)*			
This register defines the number of descriptors in the descriptor array for which RxDescriptor is the base address. The number of descriptors should match the number of states. The register uses minus-one encoding, i.e. if the array has 8 status elements, the value in the register should be 7.				
31:16	-	-		Unused
15:0	RxDescriptorNumber	R/W	0	Number of descriptors in the descriptor array for which RxDescriptor is the base address. The number of descriptors is minus-one encoded.
Offset 0x07 2114	*Receive Produce Index (RxProduceIndex)*			
This register indexes the descriptor that is going to be filled next by the Receive Datapath. After a packet has been received, hardware increments the index and wrapps to 0 when the value of RxDescriptorNumber has been reached. If the RxProduceIndex equals RxConsumeIndex – 1, the array is full, and any further packets being received will cause a buffer overrun error.				
31:16	-	-		Unused
15:0	Rx Produce Index	RO		Index of the descriptor that is going to be filled next by the Receive Datapath.
Offset 0x07 2118	*Receive Consume Index (RxConsumeIndex)*			

Table 2: LAN100 Registers *…Continued*

Bit	Symbol	Access	Value	Description
This register indexes the descriptor that is to be processed next by the software receive driver. The receive array is empty as long as RxProduceIndex equals RxConsumeIndex. As soon as the array is not empty, software can process the packet pointed to by RxConsumeIndex. After a packet has been processed by software, software should increment the RxConsumeIndex register, wrapping to 0 once the RxDescriptorNumber has been reached. If the RxProduceIndex equals RxConsumeIndex – 1, the array is full, and any further packets being received will cause a buffer overrun error.				
31:16	-	-		Unused
15:0	RxConsumeIndex	R/W	0	Index of the descriptor that is going to be processed next by the receive software.
Offset 0x07 211C	*Non-real-time Transmit Descriptor Base Address Register (TxDescriptor)*			
This register is a byte address aligned to a word boundary (i.e. the two LSBs are fixed to 0). The register contains the lowest address in the array of descriptors.				
31:2	TxDescriptor	R/W	0	MSBs of non-real-time descriptor base address
1:0	-	RO	0	Fixed to 2'b00
Offset 0x07 2120	*Non-real-time Transmit Status Base Address Register (TxStatus)*			
This register is a byte address aligned to a double word boundary (i.e., the three LSBs are fixed to 0). The register contains the lowest address in the array of statuses.				
31:3	TxStatus	R/W	0	MSBs of non-real-time transmit status base address
2:0	-	RO	0	Fixed to 0
Offset 0x07 2124	*Non-real-time Transmit Number Of Descriptors Register (TxDescriptorNumber)*			
This register defines the number of descriptors in the descriptor array for which TxDescriptor is the base address. The number of descriptors should match the number of statuses. The register uses minus-one encoding, i.e., if the array has 8 status elements, the value in the register should be 7.				
31:16	-	-		Unused
15:0	TxDescriptorNumber	R/W	0	Number of descriptors in the descriptor array for which TxDescriptor is the base address. The register is minus-one encoded
Offset 0x07 2128	*Non-real-time Transmit Produce Index (TxProduceIndex)*			
This register defines the descriptor that is going to be filled next by the software transmit driver. The transmit descriptor array is empty as long as TxProduceIndex equals TxConsumeIndex. As soon as the array is not empty, the non-real-time transmit hardware will start transmitting packets, if enabled. After a packet has been processed by software, software should increment the TxProduceIndex, wrapping to 0 once the TxDescriptorNumber has been reached. If the TxProduceIndex equals TxConsumeIndex – 1, the descriptor array is full and software should stop producing new descriptors until hardware has transmitted some packets and updated the TxConsumeIndex.				
31:16	-	-		Unused
15:0	TxProduceIndex	R/W	0	Index of the descriptor that is going to be filled next by the non-real-time transmit software driver.
Offset 0x07 212C	*Non-real-time Transmit Consume Index (TxConsumeIndex)*			
This register defines the descriptor that is going to be transmitted next by the hardware non-real-time transmit process. After a packet has been transmitted, hardware increments the index, wrapping to 0 once TxDescriptorNumber has been reached. If the TxConsumeIndex equals TxProduceIndex, the descriptor array is empty, and the transmit channel will stop transmitting until software produces new descriptors.				
31:16	-	-		Unused
15:0	TxConsumeIndex	RO		Index of the descriptor that is going to be transmitted next by the non-real-time Transmit Datapath.
Offset 0x07 2130	*Real-time Transmit Descriptor Base Address Register (TxRtDescriptor)*			

Table 2: LAN100 Registers *…Continued*

Bit	Symbol	Access	Value	Description
This register is a byte address aligned to a word boundary (i.e., the two LSBs are fixed to 0). The register contains the lowest address in the array of descriptors.				
31:2	TxRtDescriptor	R/W		MSBs of real-time descriptor base address.
1:0	-		0	Fixed to 0.
Offset 0x07 2134	*Real-time Transmit Status Base Address Register (TxRtStatus)*			
This register is a byte address aligned to a double word boundary (i.e., the three LSBs are fixed to 0). The register contains the lowest address in the array of status words.				
31:3	TxRtStatus	R/W		MSBs of real-time transmit status base address.
2:0	-		0	Fixed to 0.
Offset 0x07 2138	*Real-time Transmit Number Of Descriptors Register (TxRtDescriptorNumber)*			
This register defines the number of descriptors in the descriptor array for which TxRtDescriptor is the base address. The number of descriptors should match the number of statuse words. The register uses minus-one encoding, i.e., if the array has 8 status elements, the value in the register should be 7.				
31:16	-	-		Unused
15:0	TxRtDescriptorNumber	R/W	0	Number of descriptors in the descriptor array for which TxRtDescriptor is the base address. The number of descriptors is minus-one encoded.
Offset 0x07 213C	*Real-time Transmit Produce Index Register (TxRtProduceIndex)*			
This register defines the descriptor that is going to be filled next by the software transmit driver. The transmit descriptor array is empty as long as TxRtProduceIndex equals TxRtConsumeIndex. As soon as the array is not empty, the real-time transmit hardware will start transmitting packets, if enabled. After a packet has been processed by software, software should increment the TxRtProduceIndex, wrapping to 0 once TxRtDescriptorNumber has been reached. If the TxRtProduceIndex equals TxRtConsumeIndex – 1, the descriptor array is full, and software should stop producing new descriptors until hardware has transmitted some packets and updated the TxRtConsumeIndex.				
31:16	-	-		Unused
15:0	TxRtProduceIndex	R/W	0	Index of the descriptor that is going to be filled next by the real-time transmit software driver.
Offset 0x07 2140	*Real-time Transmit Consume Index Register (TxRtConsumeIndex)*			
This register defines the descriptor that is going to be transmitted next by the hardware real-time transmit process. After a packet has been transmitted, hardware increments the index, wrapping to 0 once TxRtDescriptorNumber has been reached. If the TxRtConsumeIndex equals TxRtProduceIndex, the descriptor array is empty, and the transmit channel will stop transmitting until software produces new descriptors.				
31:16	-	-		Unused
15:0	TxRtConsumeIndex	RO		Index of the descriptor that is going to be transmitted next by the real-time Transmit Datapath.
Offset 0x07 2144	*Transmit Block Zone Register (BlockZone)*			
The BlockZone Register is only used in real-time/non-real-time arbitration mode, i.e., when the EnableQoS bit in the Command register is deasserted. The BlockZone register defines a window before a real-time transmission time-stamp in which no new non-real-time transmissions can be started, so as to free up the Ethernet for a pending real-time transmission. The size of the BlockZone window in seconds is: BlockZone * $T_{\text{Time-stamp Clock}}$ No new non-real-time transfers will be started if: DescriptorTimeStamp < GlobalTimeStamp + BlockZone The real-time transmission will start as soon as: DescriptorTimeStamp < GlobalTimeStamp.				
31:16	-	-		Unused

Table 2: LAN100 Registers *…Continued*

Bit	Symbol	Access	Value	Description
15:0	BlockZone	R/W	0	Time margin for a transmit time-stamp during which no new non-real-time packets will be transmitted to free up the Ethernet for upcoming real-time transmissions.
Offset 0x07 2148	*Transmit Quality Of Service Time-out Register (QoSTimeout)*			
The QoSTimeout Register is only used in QoS mode, i.e., when the QoSEnable bit of the Command register is set.				
31:16	-	-		Unused
15:0	QoSTimeout	R/W		Specifies the maximum number of clock cycles a low-priority transmission must wait for transmission. If the time-out counter expires, the low-priority transmission will get the highest priority. QoSTimeout is specified in units of 64 times the transmit clock cycle.
Offset 0x07 2158	*Transmit Status Vector 0 Register (TSV0)*			
The Transmit Status Vector registers TSV0 and TSV1 store the most recent transmit status returned by the MII Interface. Since the status vector consists of more than 4 bytes, status is distributed over two registers: TSV0 and TSV1.				
31	VLAN	RO		Set if the frame's length/type field contained 0x8100, which is the VLAN protocol identifier.
30	Backpressure	RO		Set if carrier-sense method back pressure was previously applied.
29	PAUSE	RO		Set if the frame was a control frame with a valid PAUSE opcode.
28	Control frame	RO		Set if the frame was a control frame.
27-12	Total bytes	RO		The total number of bytes transferred, including collided attempts.
11	Underrun	RO		Set if the host side caused a buffer underrun condition.
10	Giant	RO		Set if the byte count in the frame was greater than [15:0].
9	LateCollision	RO		Set if a collision occurred beyond the collision window (512 bit times).
8	MaximumCollision	RO		If set, the packet was aborted because it exceeded the maximum allowed number of collisions.
7	ExcessiveDefer	RO		If set, the packet was deferred in excess of 6071 nibble times in 100Mb/s, or 24287 bit times in 10Mb/s mode.
6	PacketDefer	RO		If set, the packet was deferred for at least one attempt, but less then an excessive defer.
5	Broadcast	RO		Set if the packet's destination was a broadcast address.
4	Multicast	RO		Set if the packet's destination was a multicast address.
3	Done	RO		Indicates that the transmission of the packet was completed.
2	LengthOutOfRange	RO		Indicates that the frame type/length field was larger than 1500 bytes.
1	LengthCheckError	RO		Indicates that the frame length field does not match the actual number of data items, and is not a type field.
0	CRCError	RO		Set if the attached CRC in the packet did not match the CRC generated internally.
Offset 0x07 215C	*Transmit Status Vector 0 Register (TSV1)*			
The Transmit Status Vector registers TSV0 and TSV1 store the most recent transmit status returned by the MII Interface. Since the status vector consists of more than 4 bytes, status is distributed over two registers: TSV0 and TSV1.				
31:20	-	-		Unused

Table 2: LAN100 Registers *…Continued*

Bit	Symbol	Access	Value	Description
19:16	TransmitCollisionCount	RO		The number of collisions the current packet incurred during transmission attempts. The maximum number of collisions (16) cannot be represented.
15:0	TransmitByteCount	RO		The total number of bytes in the frame not counting the collided bytes.
Offset 0x07 2160	*Receive Status Vector Register (RSV)*			
The Receive Status Vector register stores the most recent receive status returned by the MII Interface.				
31	-	-		Unused
30	VLAN	RO		The frame's length/type field contained 0x8100, which is the VLAN protocol identifier.
29	UnsupportedOpcode	RO		The current frame was recognized as a Control Frame, but it contains an unknown opcode.
28	Pause	RO		The frame was a control frame with a valid PAUSE opcode.
27	ControlFrame	RO		The frame was a control frame.
26	DribbleNibble	RO		Indicates that after the end of packet, another 1-7 bits were received. A single nibble, called the "dribble nibble," is formed but not sent out.
25	Broadcast	RO		The packet's destination was a broadcast address.
24	Multicast	RO		The packet's destination was a multicast address.
23	ReceiveOK	RO		The packet had valid CRC and no symbol errors.
22	LengthOutOfRange	RO		Indicates that the frame type/length field was larger than 1518 bytes.
21	LengthCheckError	RO		Indicates that the frame length field does not match the actual number of data items, and is not a type field.
20	CRCError	RO		The attached CRC in the packet did not match the CRC generated internally.
19	ReceiveCodeViolation	RO		Indicates that MII data does not represent a valid receive code when LAN_RX_ER is asserted during the data phase of a frame.
18	CarrierEventPreviouslySeen	RO		Indicates that at some time since the last receive statistics, a carrier event was detected.
17	RXDVEventPreviouslySeen	RO		Indicates that the last receive event seen was not long enough to be a valid packet.
16	PacketPreviouslyIgnored	RO		Indicates that a packet since the last RSV was dropped.
15:0	ReceivedByteCount	RO		Indicates length of received frame.
Offset 0x07 2170	*Flow Control Counter Register (FlowControlCounter)*			
31:16	PauseTimer	R/W		In full-duplex mode, the PauseTimer field specifies the value that is inserted into the pause timer field of a pause flow control frame. In half-duplex mode, the PauseTimer field specifies the number of backpressure cycles.
15:0	MirrorCounter	R/W		In full-duplex mode, the MirrorCounter specifies the number of cycles to wait before reissuing the Pause control frame. In half-duplex mode, the MirrorCounter allows to keep on sending out the preamble until TxFlowControl is de-asserted.
Offset 0x07 2174	*Flow Control Status Register (FlowControlStatus)*			

Table 2: LAN100 Registers *…Continued*

Bit	Symbol	Access	Value	Description
31:16	-	-		Unused
15:0	MirrorCounterCurrent	RO		In full-duplex mode, this register represents the current value of the datapath's mirror counter which counts up to the value of the MirrorCounter bits from the FlowControlCounter register. In half-duplex mode, the datapath's mirror counter counts until it reaches the value of the PauseTimer bits in the FlowControlCounter register.
Offset 0x07 21FC	*Global Time-stamp Register (GlobalTimeStamp)*			
The global time-stamp register records the 32-bit running value of the global Time-stamp Clock. Internally, the counter is used to generate the time-stamp field in the transmit/receive status fields. In the real-time transmit mode, it is used as a reference for the transmit arbiter.				
31:0	GlobalTimeStamp	RO		Binary (not grey-coded) value of the global Time-stamp Counter.
Offset 0x07 2200	*Receive Filter Control Register (RxFilterCtrl)*			
31:14	-	-		Unused
13	RxFilterEnWoL	R/W		When set, the result of the perfect address matching filter, the imperfect hash filter, and the pattern match filter will generate a WoL interrupt in case of a match.
12	MagicPacketEnWoL	R/W		When set, the result of the magic packet filter will generate a WoL interrupt in case of a match.
11:8	PatternMatchEn	R/W		Each of the four bits enables one of the pattern-matching filter units. The lowest order bit corresponds to filter unit 0.
7	AndOr	R/W		The AND/OR relation between the pattern-matching filter and the accepting group of bits below. If set, the result of the pattern match filter is ANDed with the ORed results of the accepting group below. When set to 0, the result of the pattern-matching filter is ORed with the ORed results of the accepting group below. See Section 5.2.
6	-	-		Unused
5	AcceptPerfectEn	R/W		When set to '1', the packets with an address identical to the station address are accepted.
4	AcceptMulticastHashEn	R/W		When set to '1', multicast packets that pass the imperfect hash filter are accepted.
3	AcceptUnicastHashEn	R/W		When set to '1', unicast packets that pass the imperfect hash filter are accepted.
2	AcceptMulticastEn	R/W		When set to '1', all multicast packets are accepted.
1	AcceptBroadcastEn	R/W		When set to '1', all broadcast packets are accepted.
0	AcceptUnicastEn	R/W		When set to '1', all unicast packets are accepted.
Offset 0x07 2204	*Receive Filter WoL Status Register (RxFilterWoLStatus)*			
The bits in this register store the cause for a WoL. Status can be cleared by writing the RxFilterWoLClear register.				
31:9	-	-		Unused
8	MagicPacketWoL	RO		When set to '1', a magic packet filter caused WoL.
7	RxFilterWoL	RO		When set to '1', the receive filter caused WoL.
6	PatternMatchWoL	RO		When set to '1', the pattern-matching filter caused WoL.
5	AcceptPerfectWoL	RO		When set to '1', the perfect address-matching filter caused WoL.

Table 2: LAN100 Registers *…Continued*

Bit	Symbol	Access	Value	Description
4	AcceptMulticastHashWoL	RO		When set to '1', a multicast packet that passed the imperfect hash filter caused WoL.
3	AcceptUnicastHashWoL	RO		When set to '1', a unicast packet that passed the imperfect hash filter caused WoL.
2	AcceptMulticastWoL	RO		When set to '1', a multicast packet caused WoL.
1	AcceptBroadcastWoL	RO		When set to '1', a broadcast packet caused WoL.
0	AcceptUnicastWoL	RO		When set to '1', a unicast packet caused WoL.
Offset 0x07 2208	*Receive Filter WoL Clear Register (RxFilterWoLClear)*			
The bits in this register are write-only; writing resets the corresponding bits in the RxFilterWoLStatus register.				
31:9	-	-		Unused
8	MagicPacketWoLClr	WO		When set, the corresponding status bit in the RxFilterWoLStatus register is cleared.
7	RxFilterWoLClr	WO		When set, the corresponding status bit in the RxFilterWoLStatus register is cleared.
6	PatternMatchWoLClr	WO		When set, the corresponding status bit in the RxFilterWoLStatus register is cleared.
5	AcceptPerfectWoLClr	WO		When set, the corresponding status bit in the RxFilterWoLStatus register is cleared.
4	AcceptMulticastHashWoLClr	WO		When set, the corresponding status bit in the RxFilterWoLStatus register is cleared.
3	AcceptUnicastHashWoLClr	WO		When set, the corresponding status bit in the RxFilterWoLStatus register is cleared.
2	AcceptMulticastWoLClr	WO		When set, the corresponding status bit in the RxFilterWoLStatus register is cleared.
1	AcceptBroadcastWoLClr	WO		When set, the corresponding status bit in the RxFilterWoLStatus register is cleared.
0	AcceptUnicastWoLClr	WO		When set, the corresponding status bit in the RxFilterWoLStatus register is cleared.
Offset 0x07 220C	*Pattern Matching Join Register (PatternMatchJoin)*			
See Section 3.3 on page 23-25 for a description of the functions of this register. See Table 3 for a description of the Join operation code fields.				
31:24	-	-		Unused
23:20	Join0123	R/W		Control bits for joining the result of Join012 and Join123.
19:16	Join123	R/W		Control bits for joining the result of Join12 and Join23.
15:12	Join012	R/W		Control bits for joining the result of Join01 and Join12.
11:8	Join23	R/W		Control bits for joining the result of pattern-matching filter units 2 and 3.
7:4	Join12	R/W		Control bits for joining the result of pattern-matching filter units 1 and 2.
3:0	Join01	R/W		Control bits for joining the result of pattern-matching filter units 0 and 1.
Offset 0x07 2210	*Hash filter table LSBs register (HashFilterL)*			

Table 2: LAN100 Registers *…Continued*

Bit	Symbol	Access	Value	Description
31:0	HashFilterL	R/W		Bit 31:0 of the imperfect filter hash table for receive filtering.
Offset 0x07 2214 Hash filter table MSBs register (HashFilterH)				
31:0	HashFilterH	R/W		Bit 63:32 of the imperfect filter hash table for receive filtering.
Offset 0x07 2230/40/50/60 PatternMatch Unit 0/1/2/3 Mask LSBs Register (PatternMatchMask0/1/2/3L)				
The PatternMatchMask registers specify a mask for the pattern matching windows so that some bytes can be masked out in the CRC calculation. The PatternMatchMask consists of 64 byte-enable signals, one for each byte in the pattern-matching window. The pattern-matching mask is distributed over two 32-bit registers. The LAN100 has four pattern-matching units.				
31:0	PatternMatchMask0/1/2/3L	R/W		Bits 31:0 of the pattern-matching mask for filter unit 0/1/2/3. Each bit represents a byte-enable in the pattern-matching window.
Offset 0x07 2234/44/54/64 PatternMatch Unit 0/1/2/3 Mask MSBs Register (PatternMatchMask0/1/2/3H)				
The PatternMatchMask registers specify a mask for the pattern matching windows so that some bytes can be masked out in the CRC calculation. The PatternMatchMask consists of 64 byte-enable signals, one for each byte in the pattern-matching window. The pattern matching mask is distributed over two 32-bit registers. The LAN100 has four pattern-matching units.				
31:0	PatternMatchMask0/1/2/3H	R/W		Bits 63:32 of the pattern-matching mask for filter unit 0/1/2/3. Each bit represents a byte-enable in the pattern-matching window.
Offset 0x07 2238/48/58/68 PatternMatch Unit 0/1/2/3 CRC Register (PatternMatchCRC0/1/2/3)				
Each of the four pattern-matching filters calculates a 32-bit CRC on a 64-byte window. If the CRC matches the 32-bit golden CRC value in the filter unit's CRC register, a match is found.				
31:0	PatternMatchCRC0/1/2/3	R/W		The golden CRC for pattern-matching filter unit 0/1/2/3.
Offset 0x07 223C/4C/5C/6C PatternMatch Unit 0/1/2/3 Skip Bytes (PatternMatchSkip0/1/2/3)				
Each of the four pattern-matching filters calculates a 32-bit CRC on a 64-byte window. The window can have an offset with respect to the start of the frame. The Pattern Match Unit 0/1/2/3 Skip Bytes register specifies the number of bytes that must be skipped before starting the window.				
31:0	PatternMatchSkip0/1/2/3	R/W		The number of bytes in a frame that need to be skipped before starting pattern-matching filtering in unit 0/1/2/3.
Offset 0x07 2FE0 Interrupt Status Register (IntStatus)				
The interrupt status register is read-only. Bits can be set via the IntSet register. Bits can be cleared via the IntClear register.				
31:14	-	-		Unused
13	WakeupInt	RO		Interrupt was triggered by a Wakeup event detected by the receive filter.
12	SoftInt	RO		Interrupt was triggered by software writing a 1 in the IntSet register.
11	TxRtDoneInt	RO		Interrupt was triggered because a real-time descriptor was transmitted and the Interrupt bit in its descriptor was set.
10	TxRtFinishedInt	RO		Interrupt was triggered because all real-time descriptors have been processed, so that now ProduceIndex == ConsumeIndex.
9	TxRtErrorInt	RO		Interrupt was triggered on real-time transmit errors: LateCollision, ExcessiveCollision, ExcessiveDefer, and NoDescriptor or Underrun.
8	TxRtUnderrunInt	RO		Interrupt set on a fatal underrun error in the real-time transmit queue. The fatal interrupt should be resolved by a Tx soft-reset. The bit is not set in case of a non fatal underrun error.

Table 2: LAN100 Registers *…Continued*

Bit	Symbol	Access	Value	Description
7	TxDoneInt	RO		Interrupt was triggered because a non-real-time descriptor was transmitted and the Interrupt bit in its descriptor was set.
6	TxFinishedInt	RO		Interrupt was triggered because all non-real-time descriptors have been processed, so that now ProduceIndex == ConsumeIndex.
5	TxErrorInt	RO		Interrupt was triggered on non-real-time transmit errors: LateCollision, ExcessiveCollision, ExcessiveDefer, and NoDescriptor or Underrun.
4	TxUnderrunInt	RO		Interrupt set on a fatal underrun error in the non real-time transmit queue. The fatal interrupt should be resolved by a Tx soft-reset. The bit is not set in case of a non fatal underrun error.
3	RxDoneInt	RO		Interrupt was triggered because a receive descriptor has been processed and the Interrupt bit in its descriptor was set.
2	RxFinishedInt	RO		Interrupt was triggered because all receive descriptors have been processed, so that now ProduceIndex == ConsumeIndex.
1	RxErrorInt	RO		Interrupt was triggered on receive errors: AlignmentError, RangeError, LengthError, SymbolError, CRCError, or NoDescriptor or Overrun.
0	RxOverrunInt	RO		Interrupt was triggered on fatal overrun error in the receive queue. The fatal interrupt should be resolved by a Rx soft-reset. The bit is not set in case of a non fatal underrun error.
Offset 0x07 2FE4	*Interrupt Enable Register (IntEnable)*			
31:14	-	-		Unused
13	WakeupIntEn	R/W		Enable interrupts triggered by a Wakeup event detected by the receive filter.
12	SoftIntEn	R/W		Enable interrupts triggered when software writes a 1 to the int_set Softinterrupt register.
11	TxRtDoneIntEn	R/W		Enable interrupts triggered when a real-time descriptor has been transmitted while the Control.Interrupt bit in the descriptor was set.
10	TxRtFinishedIntEn	R/W		Enable triggering interrupts when all real-time descriptors have been processed, when ProduceIndex == ConsumeIndex.
9	TxRtErrorIntEn	R/W		Enable interrupts on real-time transmit errors.
8	TxRtUnderrunIntEn	R/W		Enable interrupts on real-time transmit buffer or descriptor underrun conditions.
7	TxDoneIntEn	R/W		Enable interrupts when a non-real-time descriptor has been transmitted and the Interrupt bit in its descriptor was set.
6	TxFinishedIntEn	R/W		Enable interrupts when all non-real-time descriptors have been processed, when ProduceIndex == ConsumeIndex.
5	TxErrorIntEn	R/W		Enable interrupts on non-real-time transmit errors.
4	TxUnderrunIntEn	R/W		Enable interrupts on non-real-time transmit buffer or descriptor underrun conditions.
3	RxDoneIntEn	R/W		Enable interrupts when a receive descriptor has been processed and the Interrupt bit in its descriptor was set.
2	RxFinishedIntEn	R/W		Enable interrupts when all receive descriptors have been processed, when ProduceIndex == ConsumeIndex.

Table 2: LAN100 Registers *…Continued*

Bit	Symbol	Access	Value	Description
1	RxErrorIntEn	R/W		Enable interrupts on receive errors.
0	RxOverrunIntEn	R/W		Enable interrupts on receive buffer overrun or descriptor underrun conditions.
Offset 0x07 2FE8	*Interrupt Clear Register (IntClear)*			
The Interrupt Clear register is write-only. Writing a 1 to a bit of the register clears the corresponding bit in the Status register. Writing a 0 to a bit of the register does not affect the corresponding interrupt status.				
31:14	-	-		Unused
13	WakeupIntSet	WO		Writing a 1 clears the corresponding status bit in IntStatus.
12	SoftIntSet	WO		Writing a 1 clears the corresponding status bit in IntStatus.
11	TxRtDoneIntSet	WO		Writing a 1 clears the corresponding status bit in IntStatus.
10	TxRtFinishedIntSet	WO		Writing a 1 clears the corresponding status bit in IntStatus.
9	TxRtErrorIntSet	WO		Writing a 1 clears the corresponding status bit in IntStatus.
8	TxRtUnderrunIntSet	WO		Writing a 1 clears the corresponding status bit in IntStatus.
7	TxDoneIntSet	WO		Writing a 1 clears the corresponding status bit in IntStatus.
6	TxFinishedIntSet	WO		Writing a 1 clears the corresponding status bit in IntStatus.
5	TxErrorIntSet	WO		Writing a 1 clears the corresponding status bit in IntStatus.
4	TxUnderrunIntSet	WO		Writing a 1 clears the corresponding status bit in IntStatus.
3	RxDoneIntSet	WO		Writing a 1 clears the corresponding status bit in IntStatus.
2	RxFinishedIntSet	WO		Writing a 1 clears the corresponding status bit in IntStatus.
1	RxErrorIntSet	WO		Writing a 1 clears the corresponding status bit in IntStatus.
0	RxOverrunIntSet	WO		Writing a 1 clears the corresponding status bit in IntStatus.
Offset 0x07 2FEC	*Interrupt Set Register (IntSet)*			
The interrupt set register is write-only. Writing a 1 to a bit of the register sets the corresponding bit in the Status register. Writing a 0 to a bit of the register does not affect the interrupt status.				
31:14	-	-		Unused
13	WakeupIntSet	WO		Writing a 1 sets the corresponding status bit in IntStatus.
12	SoftIntSet	WO		Writing a 1 sets the corresponding status bit in IntStatus.
11	TxRtDoneIntSet	WO		Writing a 1 sets the corresponding status bit in IntStatus.
10	TxRtFinishedIntSet	WO		Writing a 1 sets the corresponding status bit in IntStatus.
9	TxRtErrorIntSet	WO		Writing a 1 sets the corresponding status bit in IntStatus.
8	TxRtUnderrunIntSet	WO		Writing a 1 sets the corresponding status bit in IntStatus.
7	TxDoneIntSet	WO		Writing a 1 sets the corresponding status bit in IntStatus.
6	TxFinishedIntSet	WO		Writing a 1 sets the corresponding status bit in IntStatus.
5	TxErrorIntSet	WO		Writing a 1 sets the corresponding status bit in IntStatus.
4	TxUnderrunIntSet	WO		Writing a 1 sets the corresponding status bit in IntStatus.
3	RxDoneIntSet	WO		Writing a 1 sets the corresponding status bit in IntStatus.
2	RxFinishedIntSet	WO		Writing a 1 sets the corresponding status bit in IntStatus.
1	RxErrorIntSet	WO		Writing a 1 sets the corresponding status bit in IntStatus.
0	RxOverrunIntSet	WO		Writing a 1 sets the corresponding status bit in IntStatus.

Table 2: LAN100 Registers *…Continued*

Bit	Symbol	Access	Value	Description
Offset 0x07 2FF4	*Power-down Register (PowerDown)*			
The PowerDown register is used to block all MMIO access except access to the PowerDown register. Setting the bit will return an error on all register access except for access to the PowerDown register. In case the LAN100 is in a power-down state because, for example, no PHY is connected, then software should set the PowerDown bit in the PowerDown register in order to prevent deadlock (for example, due to a speculative read operation of the CPU).				
31	PowerDown	R/W		If set, all register access will return a read/write error except accesses to the PowerDown register.
30:0	-	-		Unused
Offset 0x07 2FFC	*Module ID Register (ModuleID)*			
The ModuleID register is used to store standard Philips module ID information.				
31:16	ModuleID	RO	0x3902	Unique 16-bit module identification.
15:12	MajRev	RO	0x1	Major design revision number.
11:8	MinRev	RO	0x1	Minor design revision number.
7:0	ApertureSize	RO	0	Represents the aperture of the software registers in the MMIO register space. Aperture size is 4 Kb, corresponding to a value 0 in this field.

3.3 Pattern Matching Join Register

The Pattern Matching Join register (PatternMatchJoin) defines the way the outputs of the four pattern-matching sections are joined together into a single pattern-matching result. Joining is done by a pyramid architecture in three stages. The first stage

consists of three join modules, the second stage consists of two modules and the final stage consists of one module, as depicted in Figure 3. Each join module can produce a single result from two inputs while doing a logic function on the two inputs.

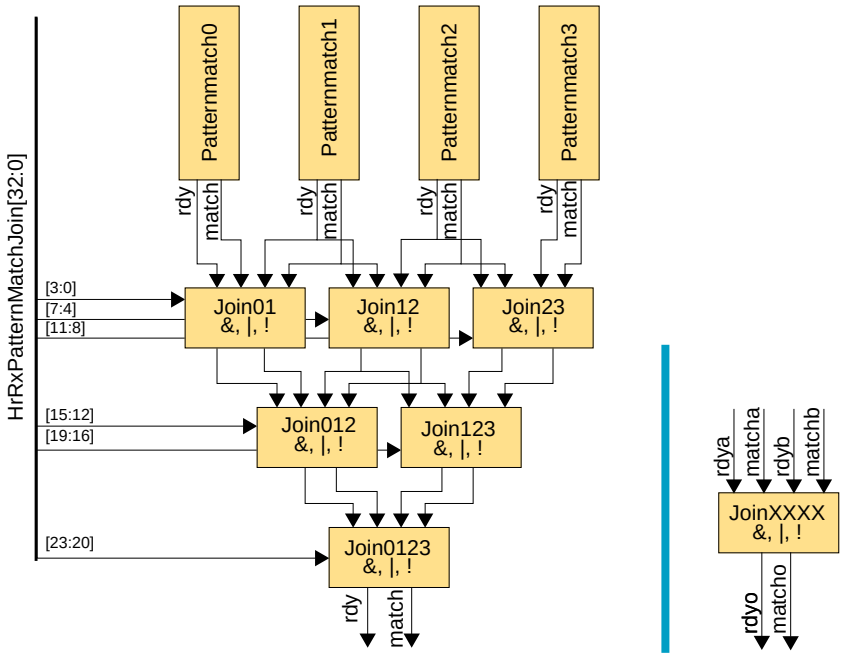


Figure 3: Pattern matching join function

Each of the join modules has four inputs *matcha*, *matchb* and *readya*, *readyb* and two outputs *matcho* and *readyo*. The ready output is the logic AND of the ready inputs (*readyo* = *readya* & *readyb*). The match output of a join module depends on the two match inputs and a logic function that can be programmed via four bits in the PatternMatchJoin register. The pattern in each nibble of that register defines the logic function the join module performs on the inputs to produce the output. Table 3 lists the bit definitions of the PatternMatchJoin register.

Table 3: PatternMatchJoin Register Nibble Functions

Nibble (binary)	Name	Function
0000_b	a & b	Logic AND
0001_b	a & !b	Logic AND NOT
0010_b	!a & b	Logic NOT AND
0011_b	!a & !b	Logic NOT AND NOT
0100_b	!(a & b)	Logic NOT OR NOT
0101_b	!(a & !b)	Logic NOT OR
0110_b	!(!a & b)	Logic OR NOT
0111_b	!(!a & !b)	Logic OR
1000_b	a ^ b	Logic XOR
1001_b	!(a ^ b)	Logic XNOR
1010_b	a	Feedthrough a

Table 3: PatternMatchJoin Register Nibble Functions *...Continued*

Nibble (binary)	Name	Function
1011_b	b	Feedthrough b
1100_b	!a	Invert a
1101_b	!b	Invert b
1110_b	0	Fix match output to zero
1111_b	1	Fix match output to one

4. Descriptor and Status Formats

This section defines the descriptor format for the transmit and receive scatter/gather DMA engines.

Each Ethernet packet can be broken into a set of *fragments*. Each fragment will have a single corresponding descriptor. The DMA managers in the LAN100 can scatter (for receive) and gather (for transmit) multiple fragments into a single Ethernet packet.

4.1 Receive Descriptors and Status

Figure 4 depicts the layout of the receive descriptors in memory.

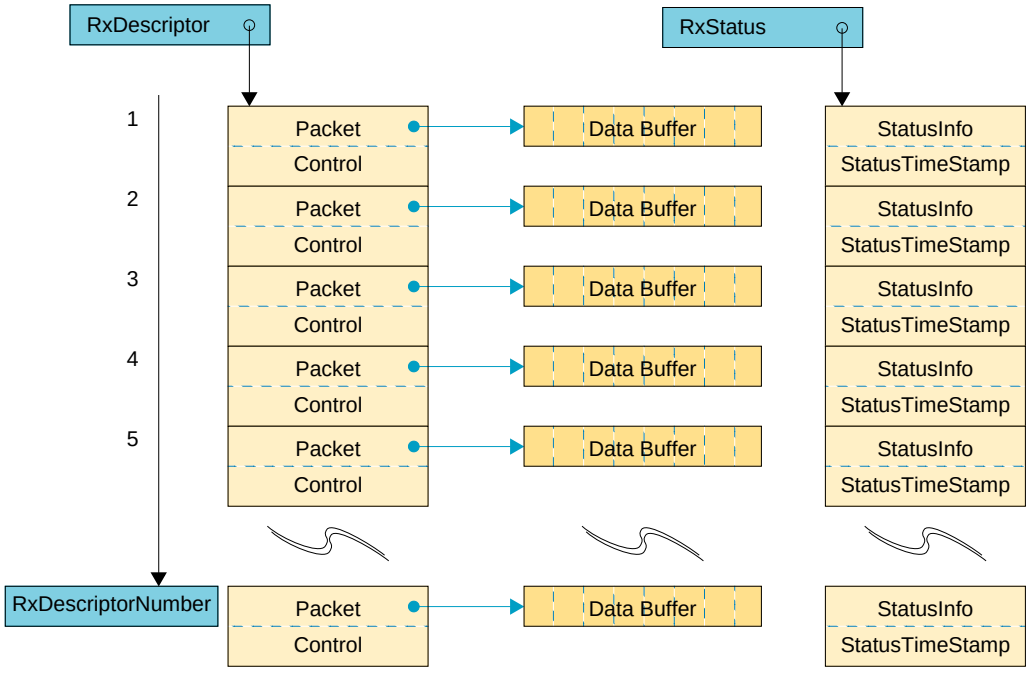


Figure 4: Receive descriptor memory layout

The receive descriptors are stored in an array in memory. The base address of the array is stored in the RxDescriptor register. The number of descriptors in the array is stored in the RxDescriptorNumber register using a minus-one encoding format. For example, if the array has 8 elements, the RxDescriptorNumber register value should

be 7. For each element of the descriptor array, there is an associated status field in the status array. The base address of the status array is stored in the RxStatus register. During operation, that is, when the Receive Datapath is enabled, the RxDescriptor, RxStatus and RxDescriptorNumber registers should not be modified. The base address of the descriptor array as stored in the RxDescriptor register should be aligned on an 8-byte address boundary. The status array base address in the RxStatus register must be aligned on an 8-byte address boundary.

The RxConsumeIndex and RxProduceIndex registers contain counters that start at 0 and wrap back around to 0 when they equal RxDescriptorNumber. The RxProduceIndex indexes the descriptor that will be filled by the next packet received by the LAN100. It is incremented by the hardware. The RxConsumeIndex is programmed by software, and is the index of the next descriptor that the software receive driver is going to process. If RxProduceIndex == RxConsumeIndex, then the receive buffer is empty. If RxProduceIndex == RxConsumeIndex – 1 then the receive buffer is full, and newly received data would generate an overflow unless the software driver frees up some descriptors.

Each Receive Descriptor Structure requires two words (8 bytes) of memory. Likewise each Receive Status Structure requires two words (8 bytes) of memory. Receive Descriptor Structures consist of a Packet word containing a pointer to a data buffer for storing receive data and a Control word. The address offset of the Packet word in the receive descriptor structure is 0, and the address offset of the Control word is offset by 4 bytes with respect to the Receive Descriptor Structure address, as defined in Table 4.

Table 4: Receive Descriptor Structure

Name	Address Offset	Size	Function
Packet	0x0	31:0	Base address of the data buffer for storing receive data.
Control	0x4	31:0	Control information, see Table 5.

The Packet word is a 32-bit byte-aligned address value containing the base address of the data buffer. The definition of the Control word bits are given in Table 5.

Table 5: Receive Descriptor Control Word

Bit	Name	Function
31	Interrupt	If set, generate an RxDone interrupt when the data in this packet or packet fragment and the associated status information has been committed to memory.
30:11		Unused
10:0	Size	Size in bytes of the data buffer. This is the size of the buffer reserved by the device driver for a packet or packet fragment, i.e., the byte size of the buffer pointed to by the Packet word. The size is –1 encoded, e.g., if the buffer is 8 bytes, the size field should be equal to 7.

Table 6 lists the components of the receive status structure.

Table 6: Receive Status Structure

Name	Address Offset	Size	Function
StatusInfo	0x0	31:0	Receive status return flags, see table 7
StatusTimeStamp	0x4	31:0	time-stamp of the receive completion

Each Receive Status Structure consists of two words. The StatusTimeStamp word contains a copy of the Time-stamp Counter value at the time the reception completed. If the fragment is not the last fragment of the packet, the status reflects the value of the time-stamp when the fragment buffer was filled completely. If the fragment is the last in a multi-packet reception, the time-stamp will be a copy of the Time-stamp Counter at the moment the MII Interface generates the status. The StatusInfo word contains flags returned by the MII Interface and flags generated by the Receive Datapath reflecting the status of the reception. Table 7 lists the bit definitions in the StatusInfo word.

Table 7: Receive Status Information Word

Bit	Name	Function
31	Error	An error occurred during reception of this packet. This is a logic OR of AlignmentError, RangeError, LengthError, SymbolError and CRCError.
30	LastFrag	When set, this bit indicates this descriptor is the last fragment of a packet. If the packet consists of a single fragment, this bit is also set.
29	NoDescriptor	No new Rx descriptor is available, and the frame is too long for the buffer size in the current receive descriptor.
28	Overrun	Receive overrun. The adapter can not accept the data stream or the status stream.
27	AlignmentError	An alignment error is flagged when dribble bits are detected or a CRC error is detected. This is in accordance with IEEE std. 802.3/clause 4.3.2. (See RSV[26] & RSV[20].)
26	RangeError	The received packet exceeds maximum packet size. (See RSV[22].)
25	LengthError	The frame length field value in the packet does not match the actual data byte-length, and specifies an invalid length. (See RSV[21].)
24	SymbolError	The PHY reports a bit error over the MII Interface during reception. (See RSV[19].)
23	CRCError	CRC error. (See RSV[20].)
22	Broadcast	The received packet is of type broadcast. (See RSV[25].)
21	Multicast	A multicast packet has been received. (See RSV[24].)
20	FailFilter	Indicates this packet has failed the Rx filter. These packets are not supposed to pass to memory. But because of the limitation of buffer FIFO size, part of this packet may already have been passed to memory. Once the packet was found to have failed the Rx filter, the remainder of the packet will be discarded without passing to memory. However, if Command.PassRxFilter is set, the whole packet will be passed to memory.
19	VLAN	Indicates a VLAN frame. (See RSV[30].)

Table 7: Receive Status Information Word

Bit	Name	Function
18	ControlFrame	Indicates this is a control frame for flow control, either a pause frame or a frame with an unsupported opcode.
17:11	-	Unused
10:0	EntryLevel	Size in bytes of the actual data transferred into one fragment buffer. In other words, this is the size of the packet or fragment as actually written by the DMA manager for one descriptor. This may be different from the Control.Size bits that indicate the size of the buffer allocated by the device driver. Size is –1 encoded, e.g., if the buffer has 8 bytes the EntryLevel value should be 7.

For multi-fragment packets, the value of the AlignmentError, RangeError, LengthError, SymbolError and CRCError bits in all but the last fragment in the packet will be 0; likewise the value of the FailFilter, Multicast, Broadcast, VLAN and ControlFrame bits is undefined. The status of the last fragment in the packet will copy the value for these bits from the MII Interface. All fragment statuses will have a valid LastFrag, EntryLevel, Error, Overrun and NoDescriptor bits.

4.2 Transmit Descriptors and Status

Figure 5 shows the layout of transmit descriptors in memory. The layout and format of real-time and non-real-time descriptors is identical.

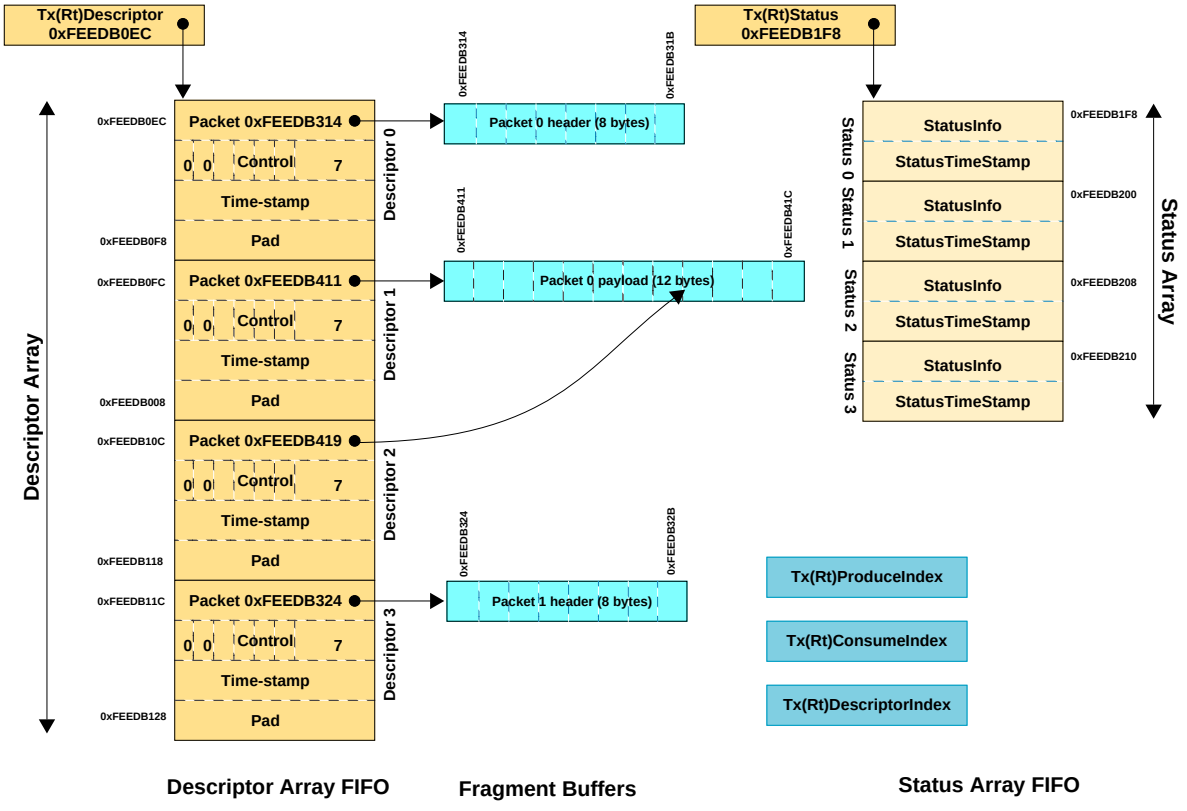


Figure 5: Transmit Descriptor Memory Layout

For each of the two transmit channels, the transmit descriptors are stored in an array in memory. The base address of the non-real-time transmit descriptor array is stored in the TxDescriptor register. Likewise for real-time transmissions, the descriptor array base address is stored in the TxRtDescriptor register. The number of descriptors in the array is stored in the Tx(Rt)DescriptorNumber register using minus-one encoding, for example, if the array has 8 elements the register value should be 7.

Parallel to the descriptors, there is an array for status. For each element of the descriptor array, there is an associated status field in the status array. The base address of the status array is stored in the Tx(Rt)Status register. During operation, that is, when the Transmit Datapath is enabled, the Tx(Rt)Descriptor, Tx(Rt)Status and Tx(Rt)DescriptorNumber registers should not be modified. The base address of the descriptor array, as stored in the Tx(Rt)Descriptor register, must be aligned on a 16-byte address boundary. The status array base address, as stored in the Tx(Rt)Status register, must be aligned on an 8-byte address boundary.

The TxConsumeIndex and TxProduceIndex registers are counters that start at 0 and wrap back to 0 when they equal TxDescriptorNumber, which indicates the number of descriptors that have been processed in the non-real-time transmit channel. The TxRtConsumeIndex and TxRtProduceIndex perform the same function for the real-time channel. The Tx(Rt)ProduceIndex contains the index of the next descriptor that is going to be filled by the software driver. The Tx(Rt)ConsumeIndex contains the index of the next descriptor that is going to be transmitted by the hardware. If Tx(Rt)ProduceIndex == Tx(Rt)ConsumeIndex, then the transmit buffer is empty. If Tx(Rt)ProduceIndex == Tx(Rt)ConsumeIndex – 1, then the transmit buffer is full, and the software driver cannot add new descriptors unless the hardware has transmitted some packets and freed up some descriptors.

As shown in Table 8, each Transmit Descriptor Structure takes four word locations (16 bytes) of memory. Likewise each Transmit Status Structure takes two words (8 bytes) in memory. Each Transmit Descriptor Structure consists of a Packet word that points to the data buffer containing transmit data, a Control word and a TimeStamp word used for real-time transmission. For non-real-time transmission, the TimeStamp word is ignored. The Pad word is always ignored and is just used to align the descriptors on a 16-byte boundary. The Packet field has a zero address offset, the control field has a 4-byte address offset, the time-stamp field has an 8 byte offset with respect to the descriptor address.

Table 8: Transmit Descriptor Fields

Name	Address Offset	Size	Function
Packet	0x0	31:0	Base address of the data buffer containing transmit data.
Control	0x4	31:0	Control information, see Table 9.
TimeStamp	0x8	31:0	Time stamp for real-time transmission. This time stamp indicates the moment when this packet is to be transmitted.
Pad	0xC	31:0	Unused word to pad to 8 byte boundary.

The Packet word is a 32-bit byte-aligned address value containing the base address of the data buffer. The time-stamp field is a 32-bit value which is compared against the internal Time-stamp Counter for real-time transmission. The definition of the Control word bits is listed in Table 9.

Table 9: Transmit Descriptor Control Word

Bit	Name	Function
31	Interrupt	If set, generate an Tx(Rt)Done interrupt when the data in this packet or packet fragment has been sent and the associated status information has been committed to memory.
30	Last	If set, this bit indicates that this is the descriptor for the last fragment in the receive packet. If not set, the fragment from the next descriptor should be appended.
29	CRC	If set, append a hardware CRC to the packet.
28	Pad	If set, pad short packets to 64 bytes.
27	Huge	If set, this bit enables huge frames. When not set, this bit prevents transmission of more than MAXF[15:0]. When set, it allows unlimited frame sizes.
26	Override	Per-packet override. If set, bits [30:27] override the defaults from the MII Interface internal registers. If not set, Control bits [30:27] will be ignored and the default values from the MII Interface will be used.
25:11		Unused
10:0	Size	Size in bytes of the data buffer. This is the size of the packet or fragment as it must be fetched by the DMA manager. In most cases, it will be equal to the byte size of the data buffer pointed to by the Packet field of the descriptor. Size is –1 encoded, e.g., a buffer of 8 bytes is encoded as the Size value 7.

Table 10 lists the fields in the Transmit Status Structure.

Table 10: Transmit Status Structure

Name	Address Offset	Size	Function
StatusInfo	0x0	31:0	Transmit status return flags, see Table 11.
StatusTimeStamp	0x4	31:0	Time-stamp of transmit completion.

The Transmit Status Structure consists of two words. The StatusTimeStamp word contains a copy of the Time-stamp Counter value at the time the Transmit Status Structure was received from the MII Interface. If the fragment is not the last fragment in the packet, the time-stamp value will be a copy of the Time-stamp Counter at the time all data in the fragment was accepted by the Tx retry module. The StatusInfo

word contains flags returned by the LAN100 and flags generated by the Transmit Datapath reflecting the status of the transmission. Table 11 lists the bit definitions in the StatusInfo word.

Table 11: Transmit Status Information Word

Bit	Name	Function
31	Error	An error occurred during transmission. This is a logic OR of LateCollision, ExcessiveCollision and ExcessiveDefer.
30	NoDescriptor	The Tx stream is interrupted, because a descriptor is not available.
29	Underrun	A Tx underrun occurred because the adapter did not produce transmit data or the adapter is not accepting statuses.
28	LateCollision	An Out-of-window-collision was seen, causing packet abort. (See TSV[29].)
27	ExcessiveCollision	Indicates this packet exceeded the maximum collision limit and was aborted. (See TSV[28].)
26	ExcessiveDefer	This packet incurred deferral beyond the maximum deferral limit and was aborted. (See TSV[27].)
25	Defer	This packet incurred deferral, because the medium was occupied. This is not an error unless excessive deferral occurs. (See TSV[26].)
24:21	CollisionCount	The number of collisions this packet incurred, up to the Retransmission Maximum. (See TSV[19:16].)
20:0	-	Unused

For multi-fragment packets, the value of the LateCollision, ExcessiveCollision, ExcessiveDefer, Defer and CollissionCount bits in all but the last fragment in the packet will be 0. The status of the last fragment in the packet will copy the value for these bits from the MII Interface. All fragment statuses will have valid Error, NoDescriptor and Underrun bits.

5. LAN100 Functions

This section defines the functions of the LAN100. After introducing the DMA concepts and giving a description of the basic transmit and receive functions, this section covers such advanced features as flow control, receive filtering, and the like.

5.1 MMIO Interface

5.1.1 Overview

The LAN100 MMIO interface is connected to the DCS system control bus so that the host registers are visible from the CPU. The MMIO interface allows device driver software on the CPU to interact with the LAN100.

The MMIO interface has a 32-bit datapath and an address aperture of 4KB. It only supports word accesses. Table 1 on page 23-6 lists the LAN100 registers.

The MMIO interface will return a read error if an MMIO read operation accesses a write-only register; likewise a write error is returned if an MMIO write operation accesses to the read-only register. An MMIO read or write error will be returned on MMIO read or write accesses to reserved registers.

If the PowerDown bit of the PowerDown register is set, all MMIO read and write accesses will return a read or write error except for accesses to the PowerDown register.

5.2 Direct Memory Access

5.2.1 Descriptor FIFOs

The LAN100 includes three high-performance DMA managers. The DMA managers make it possible to transfer packets directly to and from memory with little support from the processor, and without the need to trigger an interrupt for each packet.

The DMA managers work with FIFOs of packet descriptor structures and status structures that are stored in host memory. The descriptor structures and status structures act as an interface between the Ethernet hardware module and the device driver software. There is one descriptor FIFO for receive packets and there are two descriptor FIFOs for transmit packets, one for real-time transmit traffic, and one for non-real-time transmit traffic. By separating the areas in memory where the device driver and Ethernet module each carry out write operations, it is easy to maintain memory coherency and to make the descriptors *cache safe*, so that cache memory can be used to store descriptors. Using caching and buffering for packet descriptors, the memory traffic and memory bandwidth utilization of descriptors can be kept small. This makes the descriptor format scalable to high speeds, including gigabit ethernet.

Each packet descriptor structure contains a pointer to a data buffer containing a packet or packet fragment, a control word, a status word, and a time stamp for real-time transmit.

The software driver determines the memory locations of the descriptor and status arrays and writes their base addresses in the TxDescriptor, TxRtDescriptor, RxDescriptor and TxStatus, TxRtStatus, RxStatus registers. The number of descriptor structures and status structures in each array should be written in the TxDescriptorNumber, TxRtDescriptorNumber and RxDescriptorNumber registers. The number of descriptor structures in an array should correspond to the number of status structures in the associated status array.

Descriptor structure arrays must be aligned on a 4-byte (32-bit) address boundary; status structure arrays must be aligned on an 8-byte (64-bit) address boundary.

5.2.2 Ownership of Descriptors

Both device driver software and Ethernet hardware can read and write the descriptor FIFOs simultaneously to produce and consume descriptors. A descriptor is either owned by the device driver or it is owned by the Ethernet hardware. Only the owner of a descriptor reads or writes its value. Typically, the sequence of use and ownership of descriptor structures and status structures is as follows:

- A descriptor structure is owned and set up by the device driver

- Ownership of the descriptor structure and corresponding status structure is passed by the device driver to the Ethernet module, which reads the descriptor structure and writes information to the status structure.
- The Ethernet module passes ownership of the descriptor structure back to the device driver, which uses the information in the status structure and then recycles both to be used for another packet.

Software must pre-allocate the arrays used to implement the FIFOs.

Software can hand over ownership of descriptor structures and status structures to the hardware by incrementing the TxProduceIndex, TxRtProduceIndex, and RxConsumeIndex registers, wrapping around to 0 if the array boundary is crossed. Hardware hands over descriptor structures and status structures to hardware by updating the TxConsumeIndex, TxRtConsumeIndex, and RxProduceIndex registers.

After handing over a descriptor to the receive and transmit DMA hardware, device driver software should not modify the descriptor or reclaim the descriptor by decrementing the TxProduceIndex, TxRtProduceIndex, and RxConsumeIndex registers, because descriptors may have been prefetched by the hardware. In this case the device driver software will have to wait until the packet has been transmitted. Or, the device driver can perform soft-reset of the transmit and/or Receive Datapaths, which will also reset the descriptor FIFOs.

5.2.3 Sequential Order with Wrap-around

Descriptors are read from the arrays, and statuses are written to the arrays, in sequential order with wrap-around. Sequential order means that when the Ethernet module has finished reading or writing a descriptor or status, the next descriptor or status it reads or writes is the one at the next higher, adjacent memory address. Wrap-around means that when the Ethernet module has finished reading or writing the last descriptor or status of the array (with the highest memory address), the next descriptor or status it reads or writes is the first descriptor or status of the array at the base address of the array.

5.2.4 Full and Empty State of FIFOs

The descriptor FIFOs can be *empty*, *partially full* or *full*. A FIFO is *empty* when all descriptors are owned by the producer. A FIFO is *partially full* if both producer and consumer own part of the descriptors and both are busy processing those descriptors. A FIFO is *full* when all descriptors (except one) are owned by the consumer, so that the producer has no more room to process packets.

Ownership of descriptors is indicated with the use of a *consume index* and a *produce index*. The *produce index* is the first element of the array owned by the producer. It is also the index of the array element that is next going to be used by the producer of packets (which may already be busy using it and subsequent elements). The *consume index* is the first element of the array that is owned by the consumer. It is also the number of the array element next to be consumed by the software (which may already be busy consuming it and subsequent elements).

If the consume index and the produce index are equal, the FIFO is empty, and all array elements are owned by the producer. If the consume index equals the produce index plus one, then the array is full and all array elements (except the one at the

produce index) are owned by the consumer. One array element is kept empty even with a full FIFO, so that it is easy to distinguish the full or empty state by looking at the value of the produce index and consume index.

An array must have at least two elements to be able to indicate a full FIFO with a produce index of value 0 and a consume index of value 1. The wrap-around of the arrays is taken into account when determining if a FIFO is full, so a produce index that indicates the last element in the array and a consume index that indicates the first element in the array also means the FIFO is full. When the produce index and the consume index are unequal and the consume index is not the produce index plus one (with wrap around taken into account), then the FIFO is partially full and both the consumer and producer own enough descriptors to be able to operate actively on the FIFO.

5.2.5 Interrupt Bit

The descriptor structures have an Interrupt bit, which if enabled, can be programmed by software to interrupt the CPU when a packet with this bit set is processed. When the Ethernet module is processing a descriptor and finds this bit set, it will cause an interrupt to be triggered (after committing status to memory) by setting the RxDoneInt, TxDoneInt or TxRTDoneInt bits in the IntStatus register and driving an interrupt to the CPU. If the Interrupt bit is not set in the descriptor, then the RxDoneInt, TxDoneInt or TxRTDoneInt are not set, and no interrupt is triggered. Note: the corresponding bits in IntEnable must also be set to trigger interrupts.

This offers flexible ways of managing the descriptor FIFOs. For instance, the device driver could add 10 packets to the Tx descriptor FIFO and set the Interrupt bit in descriptor number 5 in the FIFO. This would invoke the interrupt service routine before the transmit FIFO is completely exhausted. The device driver could add another batch of packets to the descriptor array without interrupting continuous transmission of packets.

5.2.6 Packet Fragments

For maximum flexibility in packet storage, packets can be split up into multiple packet fragments with fragments located in different places in memory. In this case, one descriptor is used for each packet fragment. Thus, a descriptor can point to a single packet or to a fragment of a packet. Fragments allow for scatter/gather DMA operations:

- Transmit packets are gathered from multiple fragments in memory
- Receive packets can be scattered to multiple fragments in memory.

By stringing together fragments, it is possible to create large packets from small memory areas. Another use of fragments is to be able to locate a packet header and packet body in different places and to concatenate them without copy operations in the device driver.

For transmission, the Last bit in the descriptor Control field indicates if the fragment is the last in a packet; for receive packets the Last bit in the StatusInfo field of the status words indicates if the fragment is the last in the packet. If the Last bit is 0, the next descriptor belongs to the same Ethernet packet, If the Last bit is 1 the next descriptor is a new Ethernet packet.

5.3 Initialization

After reset, the LAN100 software driver must initialize the LAN100 hardware. During initialization the software must:

- Configure the PHY via the MII Management Interface (MIIM)
- Select RMII or MII mode
- Configure the transmit and receive DMA engines
- Configure the host registers (MAC1,MAC2, etc.) in the MII Interface
- Remove the soft reset condition from the MII Interface
- Enable the receive and Transmit Datapaths

Depending on the PHY connected to the LAN100, the software must initialize registers in the PHY via the MII Management Interface. The software can read and write PHY registers by programming the MCFG, MCMD, and MADR registers of the LAN100. Write data should be written to the MWTD register; read data and status information can be read from the MRDD and MIND registers.

The LAN100 supports RMII and MII PHYs. During initialization, software must select MII or RMII mode by programming the Command register. After initialization, the RMII or MII mode should not be modified.

Transmit and receive DMA engines should be initialized by the device driver, allocating the descriptor and status arrays in memory. Real-time transmit, non-real-time transmit, and receive each have their own dedicated descriptor and status arrays. The base addresses of these arrays must be programmed in the TxDescriptor/TxStatus, TxRtDescriptor/TxRtStatus and RxDescriptor/RxStatus registers. The number of descriptor structures in an array should match the number of status structures.

Please note that the Transmit Descriptor Structures are 16 bytes each while the Receive Descriptor Structures and status structures of both receive and transmit are 8 bytes each. All descriptor arrays must be aligned on 4-byte boundaries; status arrays must be aligned on 16-byte boundaries. The number of descriptors in the descriptor arrays must be written to the TxDescriptorNumber, TxRtDescriptorNumber, and RxDescriptorNumber registers using a –1 encoding, that is, the value in the registers is the number of descriptors minus one. For example, if the descriptor array has 4 descriptors, the value of the number of descriptors register should be 3.

After setting up the descriptor arrays, packet buffers must be allocated for the receive descriptors before enabling the Receive Datapath. The Packet field of the receive descriptors must be filled with the base address of the packet buffer of that descriptor. Among others, the Control field in the receive descriptor must contain the size of the data buffer using –1 encoding.

The Receive Datapath has a configurable filtering function for discarding or ignoring specific Ethernet packets. The filtering function should also be configured during initialization.

After a hard reset, the soft reset bit in the MII Interface will remain asserted. Before enabling the LAN100, the soft reset condition must be removed.

The receive function is enabled in two steps. The receive DMA manager must be enabled and the Receive Datapath of MII Interface must be enabled. To prevent overflow in the receive DMA engine, it should be enabled by setting the RxEnable bit in the Command register before enabling the Receive Datapath in the MII Interface by setting the RECEIVE_ENABLE bit in the MAC1 register.

The non-real-time and real-time transmit DMA engine can be enabled any time by setting the TxEnable and TxRtEnable bits in the Command register.

Before enabling the datapaths, several options can be programmed, such as automatic flow control, transmit-to-receive loop-back for verification, full- or half-duplex modes, and so forth.

Base addresses of FIFOs and FIFO sizes cannot be modified without soft reset of the receive and Transmit Datapaths.

5.4 Transmit process

5.4.1 Overview

This section outlines the transmission process. The LAN100 has two Transmit Datapaths which can be configured as real-time, non-real-time, high- or low priority. For more information on non-real-time and low- or high-priority transmission, please refer to Section 5.8. In the following subsections the prefix *TxRt* refers to the real-time/low-priority Transmit Datapath and the prefix *Tx* refers to the non-real-time/high-priority Transmit Datapath.

5.4.2 Device Driver Sets Up Descriptors and Data

Before setting up one or more descriptors for transmission, the device driver should select if the packet should go to the real-time or non-real-time FIFO. Real-time traffic or low-priority QoS traffic should go to the real-time Tx descriptor FIFO while non-real-time or high-priority QoS traffic should go to the non-real-time Tx descriptor FIFO. If the selected descriptor FIFO is full, the device driver should wait for the FIFO to become not full before writing the descriptor in the FIFO. If the selected FIFO is not full, the device driver should use the descriptor indexed by TxProduceIndex from the array pointed to by TxDescriptor (or the descriptor indexed by TxRtProduceIndex from the TxRtDescriptor array for real-time/low priority QoS).

The Packet pointer in the descriptor is set to point to a data packet or packet fragment to be transmitted. The Size field in the Command field of the descriptor should be set to the number of bytes in the fragment buffer, –1 encoded. Additional control information can be indicated in the Control field in the descriptor (including bits for Interrupt, Last, CRC, and Pad). The time-stamp field in the descriptor must be initialized for real-time transmissions.

After writing the descriptor, it must be handed over to the hardware by incrementing (and possibly wrapping) the TxProduceIndex or TxRTProduceIndex registers.

If the Transmit Datapath is disabled, the device driver should not forget to enable the (non-) real-time Transmit Datapath by setting the TxEnable or TxRtEnable bit in the Command register.

If transmitting other than the last fragment of a multi-fragment packet, the Last bit in the descriptor must be set to 0; for the last fragment the Last bit must be set to 1. To trigger an interrupt when the packet has been transmitted and transmission status has been committed to memory, set the Interrupt bit in the descriptor Control field to 1. To have the hardware add a CRC in the frame sequence control field of this Ethernet frame, set the CRC bit in the descriptor. This should be done if the CRC has not already been added by software. To enable automatic padding of small packets to the minimum required packet size, set the Pad bit in the Control field of the descriptor to 1. In typical applications bits CRC and Pad are both set to 1.

The device driver can set up interrupts using the IntEnable register to wait for a completion signal from the hardware, or it can periodically inspect (poll) the progress of transmission. It can also add new packets at the end of the descriptor FIFO, while hardware consumes descriptors at the start of the FIFO.

The device driver can stop the transmit process by resetting Command.TxEnable and Command.TxRTEnable to 0. The transmission will not stop immediately; packets already being transmitted will be transmitted completely and the status will be committed to memory before deactivating the datapath. The status of the Transmit Datapath can be monitored by the device driver reading the TxRtStatus/TxStatus bits in the Status register.

As soon as the (non-) real-time Transmit Datapath is enabled and the corresponding Tx(Rt)ConsumeIndex and Tx(Rt)ProduceIndex are not equal (i.e. the hardware still must process packets from the descriptor FIFO), the Tx(Rt)Status bit in the Status register will return to 1 (active).

5.4.3 Tx(Rt) DMA Manager Reads Tx(Rt) Descriptor Arrays

When the TxEnable bit (TxRtEnable bit for real-time traffic) is set, the Tx DMA manager reads the descriptors from memory using block transfers at the address determined by TxDescriptor and TxConsumeIndex, or, for real-time traffic, at the address determined by TxRtDescriptor and TxRtConsumeIndex. The block size of the block transfer is determined by the total number of descriptors owned by the hardware, which equals Tx(Rt)ProduceIndex – Tx(Rt)ConsumeIndex.

5.4.4 Tx(Rt) DMA manager transmits data

After reading the descriptor, the transmit DMA engine reads the associated packet data from memory and transmits the packet. After the transfer is complete, the Tx DMA manager writes status information back to the StatusInfo and StatusTimeStamp words of the status. The value of the Tx(Rt)ConsumeIndex is only updated after status information has been committed to memory. The Tx DMA manager continues to transmit packets until the descriptor FIFO is empty. If the transmit FIFO is empty, the Tx(Rt)Status bit in the Status register will return to 0 (inactive). If the descriptor FIFO is empty, the Ethernet hardware will set the Tx(Rt)FinishedInt bit of the IntStatus register. The Transmit Datapath will still be enabled.

The Tx DMA manager inspects the Last bit of the descriptor Control field when loading the descriptor. If the Last bit is 0, this indicates that the packet consists of multiple fragments. The Tx DMA manager gathers all the fragments from the host memory visiting a string of packet descriptors. It appends the fragments, and sends

them out as one Ethernet frame on the Ethernet connection. When the Tx DMA manager finds a descriptor with the Last bit in the Control field set to 1, this indicates the last fragment of the frame, and thus the end of the Ethernet frame.

5.4.5 Update ConsumeIndex

Each time the Tx(Rt) DMA manager commits a status word to memory, it completes the transmission of a descriptor. It increments the Tx(Rt)ConsumeIndex (taking wrap around into account) to hand the descriptor back to the device driver software. Software can re-use the descriptor for new transmissions after the hardware has handed it back.

The device driver software can keep track of the progress of the DMA manager by reading the Tx(Rt)ConsumeIndex register to see how far along the transmit process is. When the Tx descriptor FIFOs get emptied completely, the TxConsumeIndex and TxRTConsumeIndex retain their last value.

5.4.6 Write Transmission Status

After the packet has been transmitted over the (R)MII bus and the status has been committed to memory, the StatusInfo and StatusTimeStamp words of the packet descriptor are updated by the DMA manager.

If the descriptor is for the last fragment of a packet (or for the whole packet if there are no fragments), then error flags are set (on failure) or cleared (on success) depending on the success or failure of the packet transmission. Error flags are Error, LateCollision, ExcessiveCollision, Underrun, ExcessiveDefer, and Defer. The CollisionCount field is set to the number of collisions the packet incurred, up to the Retransmission Maximum programmed in the Collision Window/Retry register. The current time-stamp time is written to the StatusTimeStamp field.

Statuses for all but the last fragment in the packet will be written as soon as the data in the packet has been accepted by the Tx(Rt) DMA manager. Even if the descriptor is for a packet fragment other than the last fragment, the error flags and time-stamp are returned. If the MII Interface detects a transmission error during transmission of a (multi-fragment) packet, the rest of the transmit data and all remaining fragments of the packet are still read. After an error, the remaining transmit data is discarded by the MII Interface. In case of errors during transmission of a multi fragment packet, the error statuses will be repeated until the last fragment of the packet. Statuses for all but the last fragment in the packet will be written as soon as the data in the packet has been accepted by the Tx(Rt) DMA manager. The status for the last fragment in the packet will only be written after the transmission has completed on the Ethernet connection.

The status of the last packet transmission can also be inspected by reading the TSV0 and TSV1 registers. These registers do not report statuses on a fragment basis and do not store information of previously sent packets.

5.4.7 Transmission Error Handling

When an error occurs during the transmit process, the Tx(Rt) DMA manager will report the error via the transmission Status written in the Status FIFO and the IntStatus interrupt status register.

The transmission can generate several types of errors: LateCollision, ExcessiveCollision, ExcessiveDefer, Underrun, and NoDescriptor. All have corresponding bits in the transmission Status. On top of the separate bits in the Status, bits for LateCollision, ExcessiveCollision, ExcessiveDefer are ORed together into the Error bit of the Status. Errors are also propagated to the IntStatus register: the Tx(Rt)Error bit in the IntStatus register is set in case of a LateCollision, ExcessiveCollision, ExcessiveDefer, or NoDescriptor error, while underrun errors are reported in the Tx(Rt)Underrun bit of the IntStatus register.

Underrun errors can have three causes:

- the next fragment in a multi fragment transmission is not available. This is a non fatal error. A NoDescriptor status will be returned on the previous fragment and the IntStatus.Tx(Rt)Error bit will be set.
- the transmission fragment data is not available while the LAN100 has already started sending the frame. This is a non fatal error. An Underrun status will be returned on trans-fer and IntStatus.Tx(Rt)Error bit will be set.
- the flow of transmission statuses stalls and a new status has to be written while a previous status still waits to be transferred. This is a fatal error which can only be resolved by soft resetting the HW.

The first and second situations are non fatal and the device driver has to resend the frame or have upper SW layers resend the frame. In the third case the HW is in an undefined state and needs to be soft reset by setting the Command.TxReset bit.

After reporting a LateCollision, ExcessiveCollision, ExcessiveDefer or Underrun error the transmission of the erroneous frame will be aborted, remaining transmission data and frame fragments will be dis-carded and transmission will continue with the next frame in the descriptor array.

Device drivers should catch the transmission errors and take action.

5.4.8 Transmit Triggers Interrupts

The Transmit Datapath can generate four different interrupt types:

- If the Interrupt bit in the descriptor Control field is set, the Tx DMA will set the Tx(Rt)DoneInt bit in the IntStatus register after sending the fragment and committing the associated transmission status to memory. Even if a descriptor (fragment) is not the last in a multi-fragment packet, the Interrupt bit in the descriptor can be used to generate an interrupt.
- If the descriptor FIFO is empty while the Ethernet hardware is enabled, the hardware will set the Tx(Rt)FinishedInt bit of the IntStatus register.
- In case memory does not provide transmission data at a sufficiently high bandwidth, the transmission may underrun, in which case the Tx(Rt)Underrun bit will be set in the IntStatus register. Another cause for underrun is if the transmission status interface stalls. This is a fatal error which requires a softreset of the transmission queue.
- In the event of a transmission error (such as LateCollision, ExcessiveCollision or ExcessiveDefer) or if the device driver provided initial fragments but did not provide the rest of the fragments (NoDescriptor) or in case of a non fatal overrun the hardware will set the Tx(Rt)ErrorInt bit of the IntStatus register.

All of the above interrupts can be enabled and disabled by setting or resetting the corresponding bits in the IntEnable register. Enabling or disabling interrupts does not affect the IntStatus register contents, only the propagation of the interrupt status to the CPU.

The interrupts, either of individual packets or of the whole list, are a good means of communication between the DMA manager and the device driver, triggering the device driver to inspect the status words of descriptors that have been processed.

5.4.9 Transmit example

Figure 6 illustrates the transmit process with a packet header of 8 bytes and a packet payload of 12 bytes.

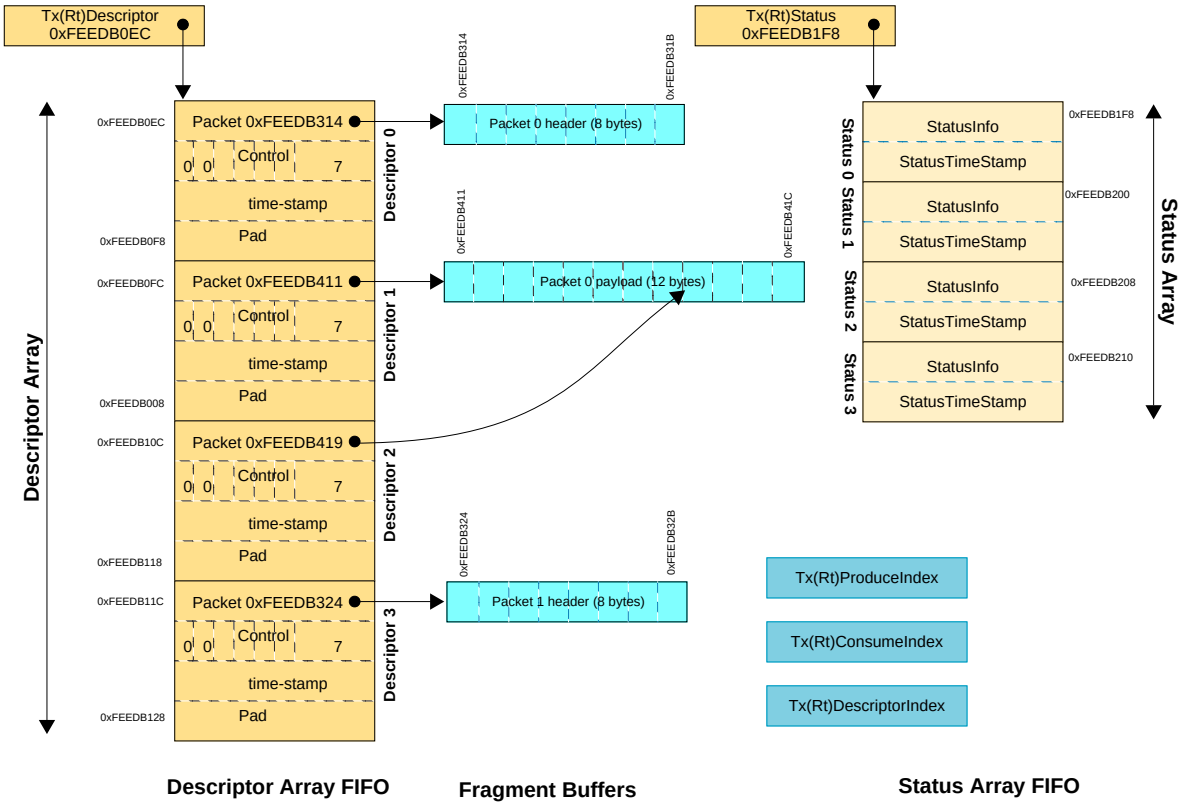


Figure 6: Transmit example memory and registers

After reset, the values of the DMA registers will be zero. During initialization, the device driver will allocate the descriptor and status array in memory. In this example, an array of four descriptors is allocated; the array is 4x4x4 bytes and aligned on a 4-byte address boundary. Since the number of descriptors should match the number of statuses, the status array consists of four elements; the array is 4x2x4 bytes and aligned on an 8-byte address boundary. The device driver writes the base address of the descriptor array (0xFEEDB0EC) in the Tx(Rt)Descriptor register and the base address of the status array (0xFEEDB1F8) in the Tx(Rt)Status register. The device

driver writes the number of descriptors and statuses (4) in the Tx(Rt)DescriptorNumber register. The descriptors and statuses in the arrays need not be initialized, yet.

Initialization may already enable the Transmit Datapath by setting the Tx(Rt)Enable bit in the Command register. In case the Transmit Datapath is enabled while there are no further packets to send, the Tx(Rt)FinishedInt interrupt flag will be set. To reduce the processor interrupt load, some interrupts can be disabled by setting the relevant bits in the IntEnable register.

Now suppose application software wants to transmit a packet of 12 bytes using a TCP/IP protocol (though in realistic applications, packets will be larger than 12 bytes). The TCP/IP stack will add a header to the packet. Because the LAN100 can perform scatter/gather DMA, the packet header need not be located in memory at the beginning of the payload data. The device driver can program a Tx gather DMA operation to collect header and payload data. To do so, the device driver will program the first descriptor to point at the packet header; the Last flag in the descriptor will be set to 0 to indicate a multi-fragment transmission. The device driver will program the next descriptor to point at the actual payload data. The maximum size of a payload buffer is 2KB, so a single descriptor suffices to describe the payload buffer. For the sake of the example though, the payload is distributed across two descriptors. After the first descriptor in the array describing the header, the second descriptor in the descriptor array describes the initial 8 bytes of the payload; the third descriptor in the array describes the remaining 4 bytes of the packet. In the third descriptor the Last bit in the Control word is set to 1 to indicate it is the last descriptor in the packet. In this example the Interrupt bit in the descriptor Control field is set in the last fragment of the packet to trigger an interrupt after the transmission completed. The Size field in the descriptor's Control word is set to the number of bytes in the fragment buffer, –1 encoded.

Note that in more realistic applications, the payload would be split across multiple descriptors only if it is more than 2KB. Also note that transmission payload data is forwarded to the hardware without the device driver having to copy it (zero copy device driver).

After setting up the descriptors for the transaction, the device driver increments the Tx(Rt)ProduceIndex register by 3, since three descriptors have been programmed. If the Transmit Datapath was not enabled during initialization the device driver must enable the datapath now.

If the Transmit Datapath is enabled, the LAN100 will start transmitting the packet as soon as it detects the Tx(Rt)ProduceIndex is not equal to Tx(Rt)ConsumeIndex. The Tx(Rt) DMA will start reading the descriptors from memory with the base address from the Tx(Rt)Descriptor register and a block size of *3 descriptors * 4 words per descriptor = 12*. The memory system will return the descriptors and the LAN100 will accept them one by one while issuing read commands for reading the transmit data fragments. The commands will have the address from the Packet field in the descriptor and a block size equal to the Size field in the descriptor.

As soon as transmission read data is returned from memory, the LAN100 will try to start transmission on the Ethernet connection via the (R)MII Interface.

While issuing the descriptor read commands, the Tx(Rt) DMA manager also begins to write the transmission status. The status write command address will be taken from the Tx(Rt)Status register; the block size will be *3 statuses * 1 double words = 3*. After transmitting each fragment of the packet, the Tx(Rt) DMA will write the status of the fragment's transmission. Statuses for all but the last fragment in the packet will be written as soon as the data in the packet has been accepted by the Tx(Rt) DMA manager. The status for the last fragment in the packet will only be written after the transmission has completed on the Ethernet connection.

The Tx(Rt) DMA manager checks if status write data has been committed to memory. Only then are the Tx(Rt)ConsumeIndex updated and the interrupt flags forwarded to the IntStatus register. The LAN100 tags transmit statuses continuously but does not necessarily tag every status individually.

Since the Interrupt bit in the descriptor of the last fragment is set, after committing status of the last fragment to memory, the LAN100 will trigger a Tx(Rt)DoneInt interrupt which triggers the device driver to inspect the status information.

In this example, the device driver cannot add new descriptors as long as the LAN100 has incremented the Tx(Rt)ConsumeIndex because the descriptor array is full (even though one descriptor is not programmed yet (see Section 5.2.4 on page 23-35). Only after committing the status for the first fragment to memory and updating the Tx(Rt)ConsumeIndex to 1 can the device driver program the next (the fourth) descriptor. The fourth descriptor can be programmed before completely transmitting the first packet.

In this example the hardware adds the CRC. If the device driver software adds the CRC, the CRC trailer can be considered another packet fragment which can be added by doing another gather DMA operation.

Figure 7 depicts the memory transactions and the transactions on the MII interface for this example.

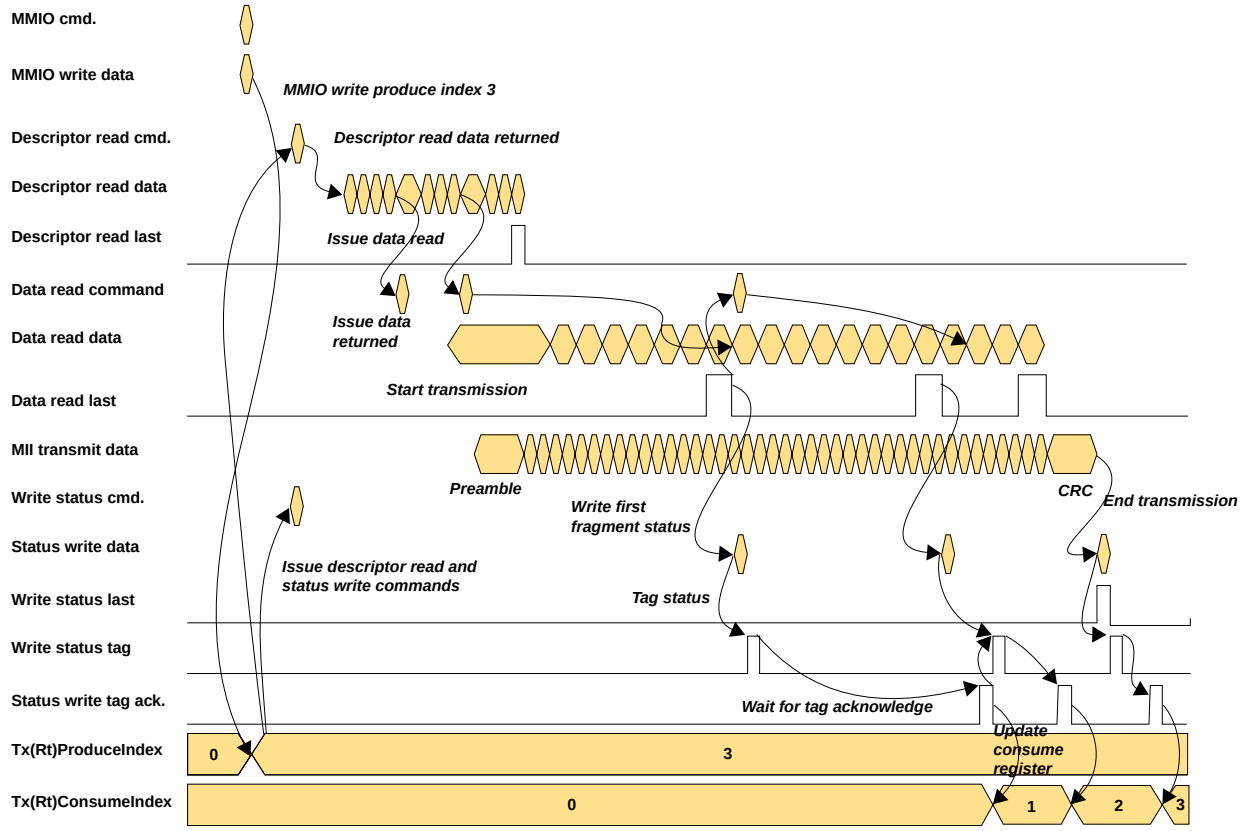


Figure 7: Transmit example waves

Each byte transferred from memory is transmitted across the MII Interface as a byte, and the MII interface hardware adds the preamble, frame delimiter leader, and the CRC trailer, if hardware CRC is enabled. Once transmission on the MII Interface commences, the transmission cannot be interrupted without generating an underrun error, which is why descriptors and data read commands are issued as soon as possible and are pipelined. In 10Mb/s and 100Mb/s mode, the output signals look similar, but in 10Mb/s mode, the transmit clock is scaled down by a factor 10.

In case an RMII PHY is connected, the data communication between the MII Interface and the PHY is communicated at half the data-width and twice the clock frequency (50 MHz). In Figure 7, the frequency of the *MII transmit data* signal will be doubled in the 100Mb/s mode. In 10Mb/s mode, data will only be transmitted once every 10 clock cycles (an clock gate disables the 50MHz clock for the intervening nine cycles).

5.5 Receive process

This section outlines the receive process including the activities in the device driver software.

5.5.1 Device Driver Sets Up Descriptors

After initializing the receive descriptor and status arrays to receive packets from the Ethernet connection (as defined in Section 5.3), the Receive Datapath should be enabled via the MAC1 register and the Control register.

During initialization, each Packet pointer in the descriptors is set to point to a data fragment buffer. The size of the buffer is stored in the Size bits of the Control field of the descriptor. Additionally the Control field in the descriptor has an Interrupt bit which allows generation of an interrupt after a fragment buffer has been filled and its status has been committed to memory.

After the Receive Datapath is initialized and enabled, all descriptors are owned by the receive hardware and should not be modified by the software unless hardware hands over the descriptor by incrementing the RxProduceIndex indicating a packet has been received. The device driver is allowed to modify the descriptors after a (soft) reset of the Receive Datapath.

5.5.2 Rx DMA Manager Reads Rx Descriptor Arrays

When the RxEnable bit in the Command register is set, the Rx DMA manager reads descriptors from memory with block transfers at the address determined by RxDescriptor and RxProduceIndex. The LAN100 will start reading descriptors even before actual receive data arrives on the MII interface (called *descriptor prefetching*). The block size of the descriptor read block transfer is determined by the total number of descriptors owned by the hardware: RxConsumeIndex – RxProduceIndex – 1. Transferring blocks of descriptors maximizes prefetching and minimizes memory loading. Read data returned from the descriptor read operation is consumed per descriptor, and only if needed.

5.5.3 Rx DMA Manager Receives Data

After reading the descriptor, the receive DMA engine waits for the MII Interface to return receive data that pass the receive filtering process. Receive packets that do not match the filtering criteria are not passed to memory. For more information on filtering refer to Section 5.12. Once a packet passes the receive filter, the data is written in the descriptors fragment buffer in memory. The Rx DMA manager does not write beyond the size of the buffer. In case a packet is received that is larger than a descriptor's fragment buffer, the packet will be written to multiple fragment buffers of consecutive descriptors. If a multi-fragment packet is received, all but the last fragment in the packet will return a status word with the Last bit set to 0. Only on the last fragment of a packet is the Last bit set in the status word. If a fragment buffer is the last of a packet, the buffer may not be filled completely. The first receive data of the next packet will be written to the next descriptor's fragment buffer.

After receiving a fragment, the Rx DMA manager writes status information back to the StatusInfo and StatusTimeStamp fields of the status word.The LAN100 writes the fill level of a descriptor's fragment buffer in the EntryLevel field of the Status word. The value of the RxProduceIndex is only updated after the fragment data and the fragment status information has been committed to memory. This is checked by sensing the write acknowledge signal returned from memory. The Rx DMA manager continues to receive packets until the descriptor FIFO is full. If it becomes full, the

Ethernet hardware will set the RxFinishedInt bit of the IntStatus register. The Receive Datapath will still be enabled. If the receive FIFO is full, any new receive data will generate an overflow error and interrupt the CPU.

5.5.4 Update ProduceIndex

Each time the Rx DMA manager commits fragment data and the associated status word to memory, it completes the reception of a descriptor and it increments the RxProduceIndex (taking wrap-around into account) and hands the descriptor back to the device driver software. Software can re-use the descriptor for new reception by handing it back to hardware when the receive data has been processed.

The device driver software can keep track of the progress of the DMA manager by reading the RxProduceIndex register to see how far along the receive process is. When the Rx descriptor FIFO is emptied completely, the RxProduceIndex retains its last value.

5.5.5 Write Reception Status

After the packet has been received from the MII Interface, the StatusInfo and StatusTimeStamp words of the packet descriptor are updated by the DMA manager.

If the descriptor is for the last fragment of a packet (or for the whole packet if there are no fragments), then, depending on the success or failure of packet reception, error flags (Error, NoDescriptor, Overrun, AlignmentError, RangeError, LengthError, SymbolError, and CRCError) are set in the status word. The EntryLevel field is set to the number of bytes actually written to the fragment buffer, –1-encoded. For fragments that are not the last in the packet, the EntryLevel will match the size of the buffer. The current time-stamp time is written to the StatusTimeStamp field. If the reception reports an error, any remaining data in the receive packet is discarded and the Last bit will be set in the receive status field so the error flags in all but the last fragment of a packet will always be 0.

The status of the last receive packet can also be inspected by reading the RSV register. The register does not report statuses on a fragment basis and does not store information about previously received packets.

5.5.6 Reception Error Handling

When an error occurs during the receive process, the Rx DMA manager will report the error via the receive status word written in the Status FIFO and the IntStatus interrupt status register.

The receive process can generate several types of errors, including: AlignmentError, RangeError, LengthError, SymbolError, CRCError, Overrun, and NoDescriptor. All have corresponding bits in the receive status word. In addition to this, AlignmentError, RangeError, LengthError, SymbolError, CRCError are ORed together into the Error bit of the status word. Errors are also propagated to the IntStatus register. The RxError bit in the IntStatus register is set in case of a AlignmentError, RangeError, LengthError, SymbolError, CRCError, and NoDescriptor error; fatal Overrun errors are report in the RxOverrun bit of the IntStatus register. On fatal overrun errors the Rx datapath needs to be soft rest by setting the Command.RxReset bit.

Overrun errors can have three causes:

- in case of a multi-fragment reception the next descriptor may be missing. In this case the NoDescriptor field is set in the status word of the previous descriptor and the RxError in the IntStatus register is set. This error is non fatal.
- the data flow on the receiver data interface stalls corrupting the packet. In this case the overrun bit in the status word is set and the RxError bit in the IntStatus register is set. This error is non fatal.
- the flow of transmission statuses stalls and a new status has to be written while a previous status still waits to be transferred. This error will corrupt the HW state and requires the HW to be soft reset. The error is detected and sets the Overrun bit in the IntStatus register.

The first overrun situation will result in an incomplete frame with a NoDescriptor status and the IntStatus. RxError bit set. SW should discard the partially received frame. In the second overrun situation the frame data will be corrupt which results in the Overrun status bit being set in the Status word while the IntError interrupt bit is set. In the third case receive errors cannot be reported in the receiver Status arrays which corrupts the HW state; the errors will still be reported in the IntStatus register s Overun bit and the Command.RxReset bit should be used to soft reset the HW.

Device drivers should catch the receive errors and take action.

5.5.7 Receive Triggers Interrupts

The Receive Datapath can generate four different interrupt types:

- If the Interrupt bit in the descriptor Control field is set, the Rx DMA will set the RxDoneInt bit in the IntStatus register after receiving a fragment and committing the associated data and status to memory. Even if a descriptor (fragment) is not the last in a multi-fragment packet, the Interrupt bit in the descriptor can be used to generate an interrupt.
- If the descriptor FIFO is full while the Ethernet hardware is enabled, the hardware will set the RxFinishedInt bit of the IntStatus register.
- If memory does not consume receive data at a sufficiently high bandwidth, the receive process may overrun, in which case the RxOverrun bit will be set in the IntStatus register.
- In case of a receive error (AlignmentError, RangeError, LengthError, SymbolError, CRCError) or a multi fragment frame where the device driver did provide descriptors for the initial fragments but did not provide the descriptors for the rest of the fragments or if a non fatal data Overrun occured the hardware will set the RxErrorInt bit of the IntStatus register.

All of the above interrupts can be enabled and disabled by setting or resetting the corresponding bits in the IntEnable register. Enabling or disabling does not affect the IntStatus register contents, only the propagation of the interrupt status to the CPU.

The interrupts, either of individual packets or of the whole list, are a good means of communication between the DMA manager and the device driver, triggering the device driver to inspect the status words of descriptors that have been processed.

5.5.8 Device Driver Processes Receive Data

The device driver can be signaled to read the descriptors that have been handed over to it by the hardware by observing status word flags (for example, RxDoneInt), receiving interrupts, or polling for the case where RxProduceIndex – RxConsumeIndex is not 0. The device driver should inspect the status words in the status FIFO to check for multi-fragment packets and receive errors.

The device driver can forward receive data and status to higher-level software layers. After data and status are processed, the descriptors, status words and data buffers may be recycled and handed back to hardware by incrementing RxConsumeIndex.

5.5.9 Receive example

Figure Figure 8 illustrates the receive process in an example receiving a packet of 16 bytes.

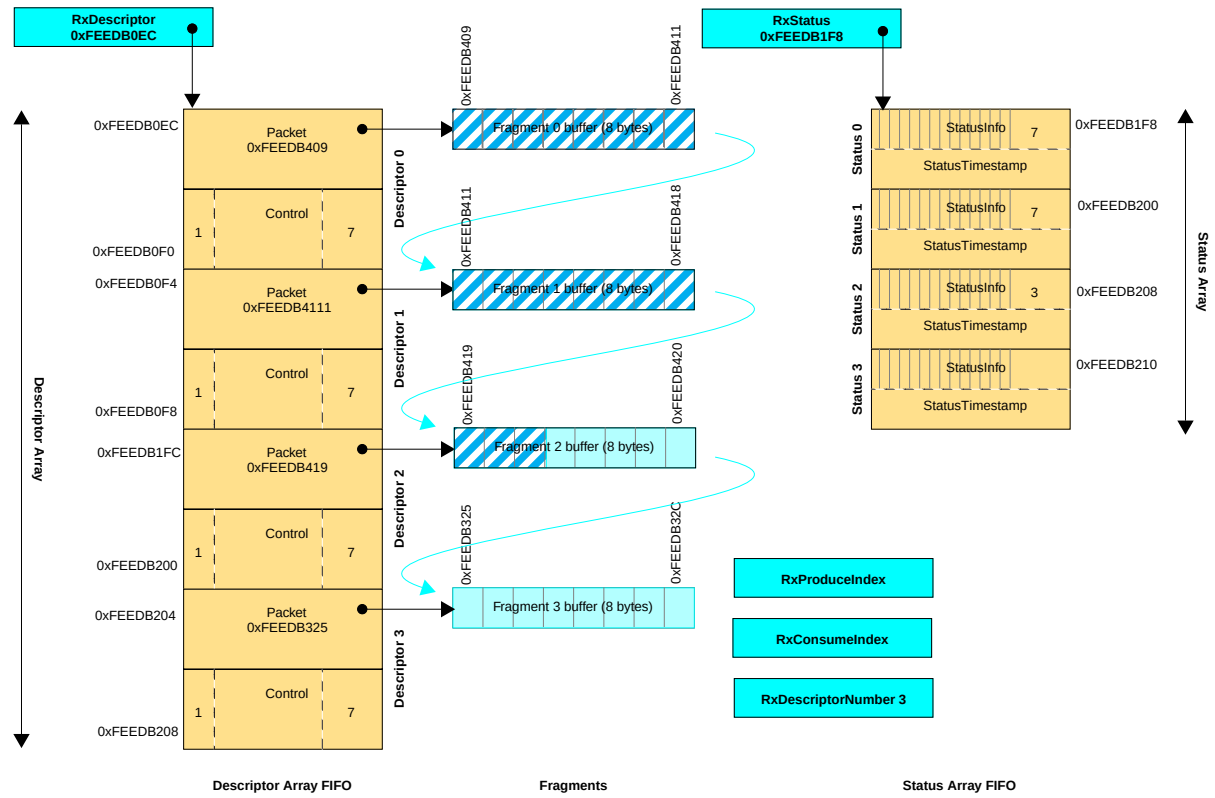


Figure 8: Receive example memory and registers

After reset, the values of the LAN100 DMA registers will be zero. During initialization, the device driver must allocate the descriptor and status array in memory. In this example, an array of four descriptors is allocated. The array is 4x2x4 bytes, and is aligned on a 4-byte address boundary. Since the number of descriptors should match the number of status words, the status array consists of four elements. The status array is 4x2x4 bytes, and is aligned on an 8-byte address boundary. The device driver writes the base address of the descriptor array (0xFEEDB0EC) to the RxDescriptor register, and the base address of the status array (0xFEEDB1F8) to the RxStatus

register. The device driver writes the number of descriptors and status words (4) in the RxDescriptorNumber register. The descriptors and status words in the arrays need not be initialized, yet.

After allocating the descriptors, a fragment buffer must be allocated for each of the descriptors. Each fragment buffer can be between 0 and 2K bytes. The base address of the fragment buffer is stored in the Packet field of the descriptors. The number of bytes in the fragment buffer is stored in the Size field of the descriptor Control word. The Interrupt field in the Control word of the descriptor can be set to generate an interrupt as soon as the descriptor has been filled by the receive process. In this example the fragment buffers are 8 bytes, so the value of the Size field in the Control word of the descriptor is set to 7. Note that in this example the fragment buffers are actually a continuous memory space; even when a packet is distributed over multiple fragments, most of the time it will be in a linear, continuous memory space; only when the descriptors wrap at the end of the descriptor array will the packet not be in a continuous memory space.

The device driver should enable the receive process by writing a 1 to the RxEnable bit of the Command register after which the MAC must be enabled by writing a 1 to the RECEIVE_ENABLE bit of the MAC1 configuration register. The LAN100 will now start receiving Ethernet packets. To reduce the processor interrupt load, some interrupts can be disabled by setting the relevant bits in the IntEnable register.

After the Rx DMA manager is enabled, it will start reading descriptors from memory. In this example, the number of descriptors is 4. Initially the RxProduceIndex and RxConsumeIndex are 0. Since the descriptor array is considered full if RxProduceIndex == RxConsumeIndex – 1, the Rx DMA manager can only read (RxConsumeIndex – RxProduceIndex – 1) = 3 descriptors (note the index wrapping). The Rx DMA manager reads the descriptors from memory; the start address will be 0xFEEDBDEC (RxDescriptor) and the block size will be *3 descriptors * 2 words per descriptor –1 = 5* (because the block size is –1 encoded).

While descriptor read commands the Rx DMA manager also sets itself up to write the transmission status. The status write command address will be taken from the RxStatus register; the block size value will be *3 status words * 1 double word –1 = 2* (because block size is –1 encoded).

After enabling the receive function in the LAN100, it will start receiving data from the MII Interface starting at the next packet. That is, if the receive function is enabled while the MII Interface is in the middle of a packet, that packet will be discarded and reception will start with the next packet. The LAN100 will strip the preamble and start-of-frame delimiter from the packet. If the packet passes the receive filtering, the Rx DMA manager will start writing the packet to the first fragment buffer.

For example, if the incomming packet is 19 bytes, then it will be distributed over three fragment buffers. After writing the initial 8 bytes in the first fragment buffer, the status for the first fragment buffer will be written and the Rx DMA will continue filling the second fragment buffer. Since this is a multi-fragment receive, the status word of the first fragment will have a 0 for the Last bit in the StatusInfo word; the EntryLevel field will be set to 7 (for a value of 8, because of –1 encoding). After writing the 8 bytes in the second fragment, the Rx DMA will continue writing the third fragment. The status of the second fragment will be like the status of the first fragment: Last = 0, EntryLevel = 7. After writing the three bytes in the third fragment buffer, the end of the packet has

been reached and the status of the third fragment is written. The third fragment's status word will have the Last bit set to 1 and the EntryLevel equal to 2 (for a value of 3, because of –1 encoding).

The next packet received from the MII interface will be written to the fourth fragment buffer, therefore five bytes of the third buffer will be unused.

The Rx DMA manager checks to make sure receive data and status data have been committed to memory. Only if memory acknowledges that the data has been committed to memory will the RxProduceIndex be updated and the interrupt flags be forwarded to the IntStatus register. The LAN100 tags receive data and statuses continuously but does not necessarily tag every data and every status individually.

After committing status of the fragments to memory the LAN100 will trigger a RxDoneInt interrupt, which triggers the device driver to inspect the status information. In this example all descriptors have the Interrupt bit set in the Control word, so all descriptors will generate an interrupt after committing data and status to memory.

In this example, the receive function of the LAN100 cannot read new descriptors as long as the device driver does not increment the RxConsumeIndex because the descriptor array is full (even though one descriptor is not programmed yet, see Section 5.2.4 on page 23-35). Only after the device driver has forwarded the receive data to application software and after the device driver has updated the RxConsumeIndex by incrementing it by the number of received fragments (3 in this case) can the LAN100 continue reading descriptors and receive data.

Figure 9 illustrates what the memory transactions and the MII Interface transactions for this example could look like.

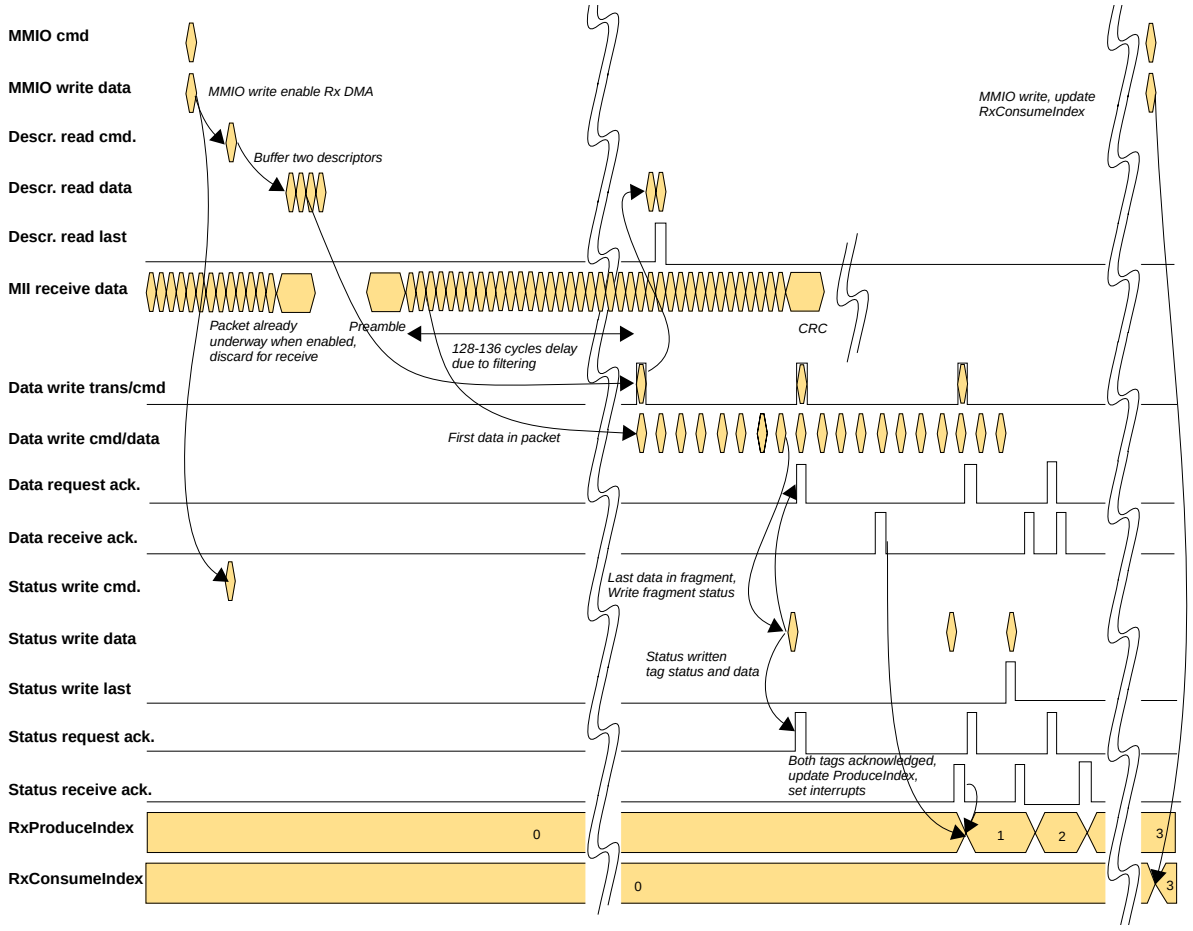


Figure 9: Receive example waves

Each pair of nibbles on the MII Interface is transferred to memory as a byte after being delayed by 128 or 136 cycles for filtering by the receive filter and buffer modules. The LAN100 removes the preamble, frame start delimiter, and CRC from the MII data and checks the CRC. To limit the probability of NoDescriptor errors, the LAN100 buffers two descriptors. After the write to memory is acknowledged for data and status, the RxProduceIndex is updated. The software device driver should now process the receive data, after which the it should update the RxConsumeIndex. For 100 Mb/s and 10 Mb/s the waveforms look identical except for frequency: in 10 Mb/s mode the MII receive input clock is 2.5 MHz and in 100 Mb/s mode the input clock is 25 MHz.

In case an RMII PHY is connected to the MII Interface, the data communication between the LAN100 and the PHY takes place at half the data-width and twice the clock frequency (50 MHz). In Figure 9, the signal marked “MII receive data” will have doubled frequency for the 100Mb/s mode. In 10Mb/s mode, data will only be transmitted once every 10 clock cycles; an external clock gate disables the 50MHz clock for the 9 cycles.

5.6 Transmission Retry

If a packet collision occurs on the Ethernet, it usually takes place during the collision window spanning the first 64 bytes of a packet. If collision is detected, the LAN100 will retry the transmission. For this reason, the first 64 bytes of a packet are buffered by the LAN100 so that it can be used during the retry. A transmission retry within the first 64 bytes in a packet is fully transparent to the application and device driver software.

When a collision occurs outside of the 64-byte collision window, a LateCollision error is triggered and the transmission is aborted. After a LateCollision error, the remaining data in the transmit packet is discarded. The LAN100 sets the Error and LateCollision bits in the packet's status fields. The Tx(Rt)Error bit in the IntStatus register is set. If the corresponding bit in the IntEnable register is set, the Tx(Rt)Error bit in the IntStatus register will be propagated to the CPU. The device driver software should catch the interrupt and take appropriate actions.

The RETRANSMISSION_MAXIMUM field of the CLRT register can be used to configure the maximum number of retries before aborting the transmission.

5.7 time-stamps

The LAN100 has an internal Time-stamp Counter register, readable by software via the GlobalTimeStamp register. After reset, the Time-stamp Counter is 0. Every clock tick of the Time-stamp Clock, the value of the Time-stamp Counter is incremented by 1. After 2^{32}–1 clock ticks, the counter wraps back to 0.

The Time-stamp Counter is only reset by asserting a hard reset.

Since the time-stamp is 32 bits in length, the maximum time that can be counted is $(2^{32}–1) * T_{clk}$ where T_{clk} is the period of the Time-stamp Clock. For a 100 MHz Time-stamp Clock this corresponds to a 42 second period. The actual frequency of the Time-stamp Clock may depend upon the software stack. The maximum frequency supported by the hardware is 200 MHz.

The value of the Time-stamp Counter is copied in the time-stamp field of the status word returned with transmit and receive fragments and packets. The device driver is able to determine the exact moment of transmission, reception and latencies using the time-stamp from the status word and the actual time-stamp value in the GlobalTimeStamp register.

5.8 Transmission modes

5.8.1 Overview

The LAN100 hardware has two transmission datapaths: Tx and TxRt. These transmission datapaths can be switched in two modes:

- Real-time/non-real-time mode: In this mode, the TxRt transmission datapath handles real-time transmissions, and the Tx datapath handles non-real-time transmissions.

- Quality of Service (QoS) mode: In this mode, the TxRt transmission datapath handles low priority transmission, while the Tx datapath handles high priority transmissions.

Each transmit data-path has it's own associated descriptor and status array in memory and it's own DMA manager for transferring data to and from memory. An arbiter internal to the LAN100 decides which of the transmit DMA managers should be allowed to transmit a packet, based on the mode of operation, the time-stamp and the priority.

The modes can be selected by the device driver by programming the QoSEnable bit in the Command register. Setting the bit to 0 selects real-time/non-real-time mode; setting the bit to 1 selects QoS mode. The device driver should configure one of the modes during initialization of the LAN100.

On a simple network stack, QoS mode may be used by creating devices named "eth0" and "eth1", one of which maps to the high priority Transmit Datapath while the other maps to the low priority Transmit Datapath.

In order to use the real-time/non-real-time mode, the network stack should have some support for time-stamps.

5.8.2 Real-time/non-real-time transmission mode

In real-time/non-real-time mode, that is, when the QoSEnable bit of the Command register is set to 0, the Tx Transmit Datapath transmits non-real-time packets while the TxRt datapath transmits real-time packets.

Transmit time-stamps allow software to specify the time when the packet should be transmitted for each packet in the TxRt descriptor array. A transmit time-stamp value is included by software in the time-stamp field of real-time transmit packet descriptors. These fields are ignored in the non-real-time descriptors or in QoS mode.

There are two transmit descriptors arrays, one for real-time traffic with transmit time-stamps and one for non-real-time traffic without transmit time-stamps. Each descriptor array is handled in sequential order by the associated DMA manager (Tx and TxRt). Packets in the real-time descriptor array have priority over packets in the non-real-time array as soon as the time-stamp has supplied. Packets in the real-time array must be ordered by software in ascending time-stamp value order. The transmit arbiter will inspect the time-stamp of a real-time packet and pass it for transmission as soon as the GlobalTimeStamp counter register exceeds the time-stamp located in the time-stamp field in the descriptor of the packet's first fragment. While the packet in the TxRt DMA is waiting for the GlobalTimeStamp to exceed the time-stamp in the current packet, the arbiter will permit the Tx DMA to transmit packets.

Care must be taken that the scheduling of real-time packets does not completely lock out non-real-time packets from gaining access to the Ethernet connection. Also, when multiple real-time packets are scheduled, there must be enough difference between their time-stamp values to have time to actually complete the packet transmissions.

When a packet misses its time-stamp moment, that is, when the current value of the time-stamp generator is larger than the time-stamp value at the moment when a descriptor is read by the Tx DMA manager, then the packet will be sent *as soon as possible* after that moment. However, since time-stamps will eventually be reused

(because the GlobalTimeStamp counter wraps around 0), packets with time-stamps that are late are only sent out when the difference between the time-stamp of the packet and the local time-stamp of the time-stamp generator is less than *half the total time-stamp range*, that is, less than 2^{31}. If the difference is more than 2^{31}, then the packet will not be transmitted, and the packet and all other packets in the real-time descriptor list must wait until the value of the time-stamp generator wraps around to its time-stamp value, or until software removes such a “stuck” packet by soft resetting the Transmit Datapath.

The register BlockZone can be used to specify a time period before a transmit time-stamp value during which no new transmission of non-real-time packets can be started. This can help to free up the Ethernet wire for the impending real-time packet. The unit of time of the BlockZone register is equal to the unit of time of the time-stamp generator.

In pseudo-code, the real-time/non-real-time arbitration proceeds as follows:

```
diff[31:0] = GlobalTimeStamp[31:0]
     - TxRtDescriptor.timeStamp[31:0];
bs[31:0] = GlobalTimeStamp[31:0] + BlockZone[31:0]
     - TxRtDescriptor.timeStamp[31:0]
if (!diff[31]) // is GlobalTimeStamp > descriptor time-stamp?
     // True
     // If possible, issue TxRt packet
else if (!bs)
     // We are in the BlockZone, so do not issue
else
     // If possible, issue Tx packet
```

If a non-real-time packet is still occupying the transmit logic when the time-stamp moment of a real-time packet is reached, then the packet will be transmitted as soon as the non-real-time packet has finished. When the value in BlockZone is 0, the BlockZone mechanism is disabled.

Apart from using time-stamps for real-time transmission, the LAN100 also reports back to software the actual moment that packets are received or transmitted, as specified in Section 5.7.

Real-time/non-real-time example

Figure 10 shows an example of real-time and non-real-time traffic.

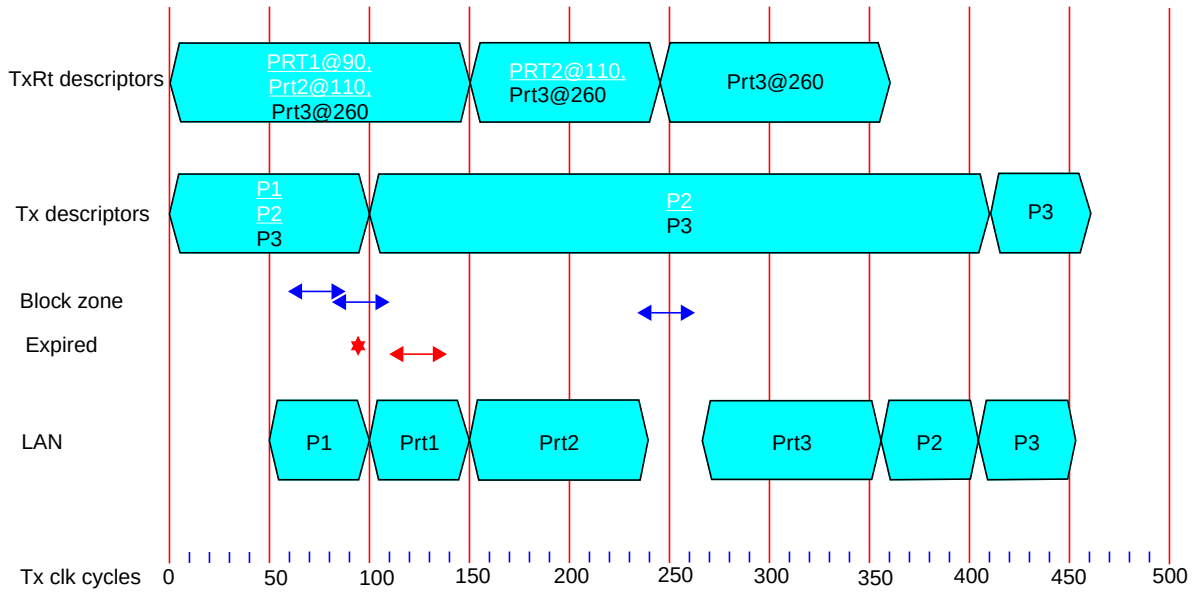


Figure 10: Real-time/non-real-time transmit example

In this example initially both transmission queues have three descriptors; the first packet in the real-time queue must be transmitted at time-stamp 90, the second packet must be transmitted at time-stamp 110, and the third packet must be transmitted at time-stamp 260. The packets in the non-real-time queue will be transmitted as soon as possible.

In this example the BlockZone register is set to 30 time-stamp cycles.

After the device driver has set up the descriptors at time-stamp 50, the LAN100 starts transmission. The initial real-time packet need not be transmitted yet, so a packet from the non-real-time queue is transmitted first. During transmission of the non-real-time packet, the time-stamp of the first real-time packet expires, and the real-time packet will be transmitted as soon as transmission of the non-real-time packet completes.

After transmitting the first real-time packet, the time-stamp of the second real-time packet expires, and the transmission of the second real-time packet commences as soon as the first real-time packet completes transmission.

After sending the second real-time packet, no new packets are sent out, since the time-stamp is within the range of the block zone of the third real-time packet. As soon as the Time-stamp Counter passes the packet's time-stamp, the third real-time packet is transmitted.

After completing transmission of the third real-time packet, the real-time queue is empty, and the remaining non-real-time packets are transmitted as soon as possible.

5.8.3 Quality-of-service Transmission Mode

The two transmit descriptor/status arrays and datapaths can also be used to implement a generic quality-of-service (QoS) mechanism for transmit packets.

The QoS mechanism distinguishes two priority queues: a queue with low priority and a queue with high priority. The Tx descriptor FIFO (using TxDescriptor and TxStatus) and Tx DMA manager implement the high-priority queue and the TxRt descriptor FIFO (using TxRtDescriptor and TxRtStatus) and TxRt DMA manager implement the low-priority queue.

To implement QoS, software should enter transmit packets that have a high quality of service requirements into the Tx (high-priority) descriptor array, while other packets should be entered into the TxRt (low-priority) descriptor array. If there are any packets in the high priority queue, they are sent out first before any packets that are waiting in the low-priority queue. Packets in the low-priority queue are only sent if the high-priority queue is empty.

To prevent starvation of packets in the low priority queue, the QoSTimeout register defines the maximum number of transmit clock cycles that low-priority packet must wait at the head of the low-priority queue. An internal time-out counter starts counting transmit clock cycles, starting from 0, as soon as a low-priority packet reaches the transmission arbiter. If the low-priority packet is still waiting to be transmitted when the counter reaches the QoSTimeout register's value, then the arbiter will promote the priority of only that one low-priority packet over the high priority queue. After sending the low-priority packet, the low-priority queue will be set back to the low priority. The waiting time of a packet in the low-priority queue can be larger than the QoSTimeout register's value if it has to wait to reach the head of the queue.

To enable the QoS mechanism, set bit EnableQoS bit in the Command register to 1. When the QoS mechanism is active, the time-stamp word in the Tx and TxRt descriptors is ignored, and the BlockZone register is disabled. The descriptor format is still the same.

QoS example

Figure 11 shows an example of QoS transmissions.

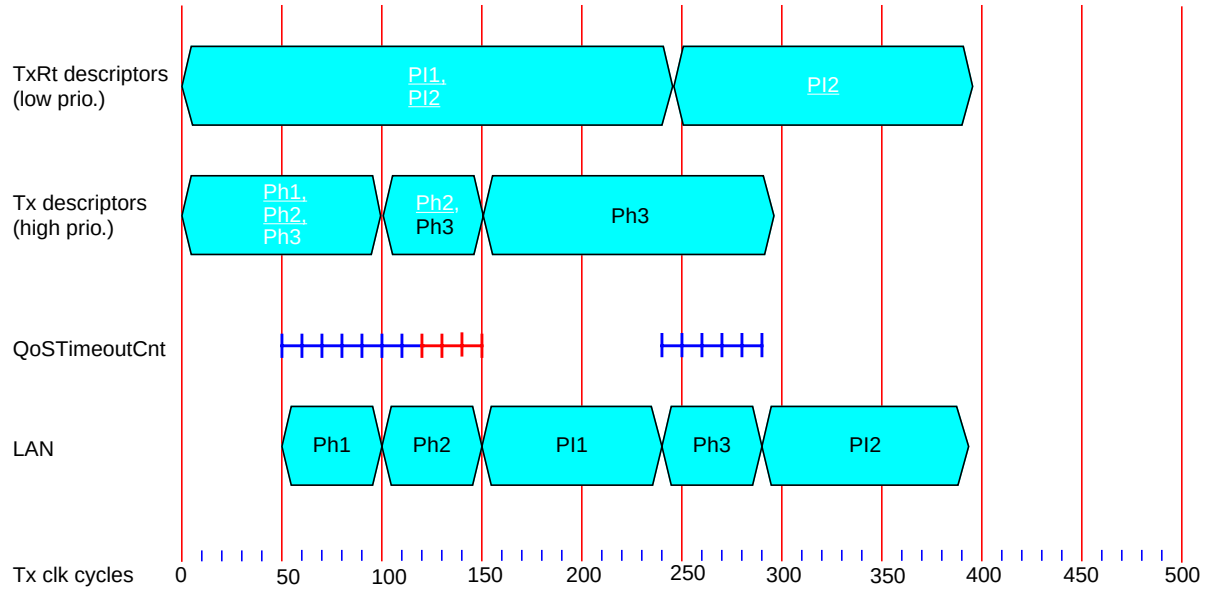


Figure 11: QoS transmission example

In this example, the low-priority queue has two packets and the high priority queue contains three packets. The QoSTimeout register is set to 70 transmit clock cycles. Initially the LAN100 will start transmitting packets from the high-priority queue. As soon as the QoSTimeout expires, the LAN100 will send out a single packet from the low-priority queue, after which it continues transmitting packets from the high-priority queue. As soon as the remaining packet in the high-priority queue has been transmitted and the high-priority queue is empty, the LAN100 will continue transmitting packets from the low-priority queue.

5.9 Duplex Modes

The LAN100 can operate in full-duplex and half-duplex mode. Half- or full-duplex mode must be configured by the device driver software during initialization of the LAN100.

For full duplex, the FullDuplex bit of the Command register must be set to 1 and the FULL_DUPLEX bit of the MAC2 configuration register must be set to 1. For half duplex, the same bits must be set to 0.

5.10 IEEE 802.3/Clause 31 Flow Control

5.10.1 Overview

For full-duplex connections, the LAN100 supports IEEE 802.3/clause 31 flow control using pause frames. This type of flow control may be used in full-duplex point-to-point connections. Flow control allows a receiver to stall a transmitter, for example, when the receive buffers are (almost) full. For this purpose the receiving side sends a pause frame to the transmitting side.

Pause frames use units of 512 bit-slot times, corresponding to 128 times the ratio of receive clock to transmit clock cycles.

5.10.2 Receive Flow Control

In full-duplex mode the LAN100 will suspend its transmissions when it receives a pause control frame. Receive flow control is initiated by the receiving side of the LAN100. It is enabled using the MAC1 configuration register by setting the RX_FLOW_CONTROL bit to 1. If the RX_FLOW_CONTROL bit is zero, then the LAN100 ignores received pause control frames. When a pause frame is received on the Rx side of the LAN100, transmission on the Tx side will be interrupted after the currently transmitting frame has completed for an amount of time as indicated in the received pause frame. The Transmit Datapath of the LAN100 will stop transmitting data for the number of 512-bit slot times encoded in the pause-timer field of the received pause control frame.

By default, the received pause control frames are not forwarded to the device driver. To forward the received flow-control frames to the device driver, set the PASS_ALL_RECEIVE_FRAMES bit in the MAC1 configuration register.

5.10.3 Transmit Flow Control

If device drivers need to stall the receive data, for example, because software buffers are full, the LAN100 can transmit pause control frames. Transmit flow control must be initiated by the device driver software; there is no IEEE 802.3/31 flow control initiated by the hardware, such as the DMA managers.

With software flow control, the device driver can detect when the process of receiving packets must be interrupted by sending out Tx pause frames. Note that due to Ethernet delays, a few packets can still be received before the flow control takes effect and the receive stream stops.

Transmit flow control is activated by writing 1 to the TxFlowControl bit of the Command register. When the Ethernet module operates in full-duplex mode, this will result in transmission of IEEE 802.3/31 pause frames. The flow control continues until a 0 is written to TxFlowControl bit of the Command register.

If the MAC is operating in full-duplex mode, then setting the TxFlowControl bit of the Command register will start a pause-frame transmission. The value inserted into the pause-timer value field of transmitted pause frames is programmed via the PauseTimer[15:0] bits in the FlowControlCounter register. When the TxFlowControl bit is deasserted, another pause frame having a pause-timer value of 0x0000 is automatically sent to abort flow control and resume transmission.

When the flow control must last a long time, a sequence of pause frames must be transmitted. This is supported with a mirror counter mechanism. To enable mirror counting, write a non-zero value to the MirrorCounter[15:0] bits in the FlowControlCounter register. When the TxFlowControl bit is asserted, a pause frame is transmitted. After sending the pause frame, an internal mirror counter is initialized to zero. The internal mirror counter starts incrementing once every 512 bit-slot times. When the internal mirror counter reaches the MirrorCounter value, another pause frame is transmitted with a pause-timer value equal to the PauseTimer field from the FlowControlCounter register. The internal mirror counter is reset to zero and the counter restarts.

The register MirrorCounter[15:0] is usually set to a smaller value than the PauseTimer[15:0] register to ensure an early expiration of the mirror counter so as to be able to send a new pause frame before the transmission on the other side can resume. By continuing to send pause frames before the transmitting side finishes counting the pause timer, the pause can be extended as long as TxFlowControl is asserted. This continues until TxFlowControl is deasserted, when a final pause frame with a pause-timer value of 0x0000 is automatically sent to abort flow control and resume transmission.

To disable the mirror counter mechanism, write the value 0 to MirrorCounter field in the FlowControlCounter register. When using the mirror counter mechanism to account for time-of-flight delays, frame transmission time, queuing delays, crystal frequency tolerances, and response time delays, the MirrorCounter should be programmed conservatively, typically at about 80% of the PauseTimer value.

If the software device driver sets the MirrorCounter field of the FlowControlCounter register to zero, the LAN100 will only send one pause control frame. After sending the pause frame, an internal pause counter is initialized to zero; the internal pause counter is incremented by one every 512 bit-slot times. Once the internal pause counter reaches the value of the PauseTimer register, the LAN100 will reset the TxFlowControl bit in the Command register. The software device driver can poll the TxFlowControl bit to detect when the pause completes.

The value of the internal counter in the flow-control module can be read out via the FlowControlStatus register. If the MirrorCounter is non-zero, the FlowControlStatus register will return the value of the internal mirror counter; if the MirrorCounter is zero, the FlowControlStatus register will return the value of the internal pause counter value.

The device driver may dynamically modify the MirrorCounter register value and switch between zero MirrorCounter and non-zero-valued MirrorCounter modes.

Transmit flow control is enabled via the TX_FLOW_CONTROL bit in the MAC1 configuration register. If the TX_FLOW_CONTROL bit is zero, then the MAC will not transmit pause control frames, and software must not initiate pause frame transmissions, and the TxFlowControl bit in the Command register should be zero.

Transmit Flow Control Example

Figure 12 illustrates the transmit flow control.

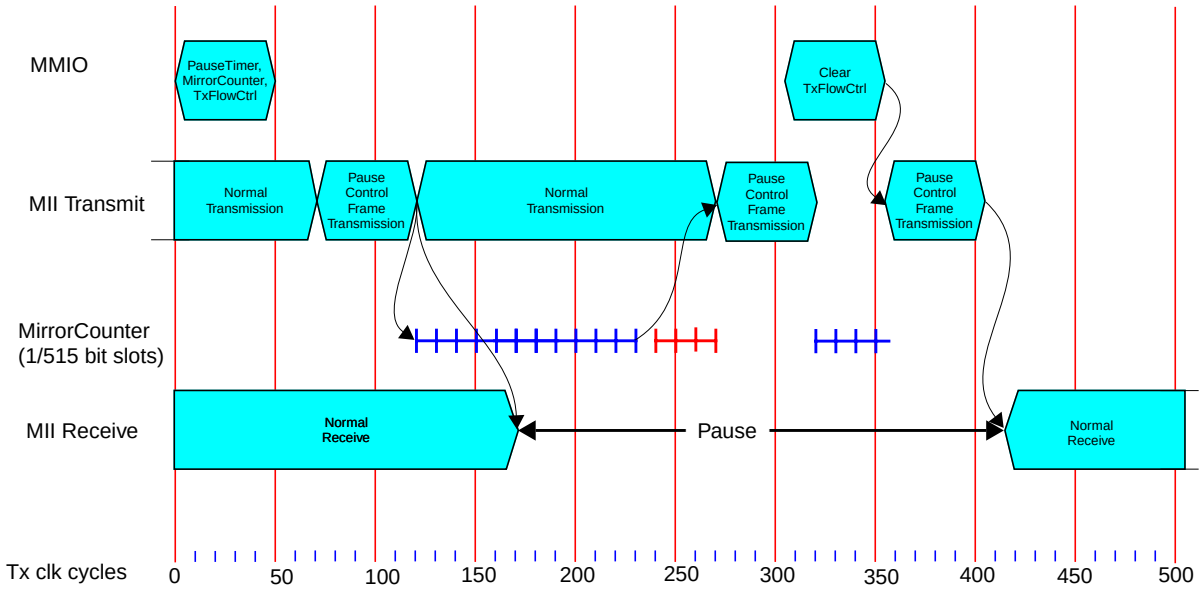


Figure 12: Transmit flow control

In this example, the LAN100 receives a packet while transmitting another packet (operating in full duplex.) The device driver detects some buffer might overrun, and enables the transmit flow control by programming the PauseTimer and MirrorCounter fields of the FlowControlCounter register after which it enables the transmit flow control by setting the TxFlowControl bit in the Command register.

In response to enabling flow control, the LAN100 will send out a pause control frame on the LAN after the packet currently being transmitted has finished. When the pause frame transmission completes, the internal mirror counter will start counting bit slots. As soon as the counter reaches the value in the MirrorCounter field, another pause frame is transmitted. While counting, the Transmit Datapath will continue normal transmissions.

As soon as software disables transmit flow control, a zero-pause control frame is transmitted to resume the receive process.

5.11 Half-duplex Mode Back Pressure

In half-duplex mode, the LAN100 can generate back pressure to stall receive packets by sending a continuous preamble that basically jams any other transmissions on the Ethernet medium. When the Ethernet module operates in half-duplex mode, asserting the TxFlowControl bit in the Command register will cause continuous preamble to be applied on the Ethernet wire, effectively blocking traffic from any other Ethernet station on the same segment.

In half-duplex mode, when the TxFlowControl bit goes high, continuous preamble is sent until TxFlowControl is deasserted. If the medium is idle, the LAN100 begins transmitting its preamble, which raises carrier sense, causing all other stations to defer. In the event that transmitting the preamble causes a collision, the back pressure “rides through” the collision. The colliding station backs off, and then defers to the back pressure. During back pressure, if the user wishes to send a frame, the back pressure is interrupted, the frame is sent, and then the back pressure is resumed. If TxFlowControl is asserted for longer than 3.3 ms in 10 Mbps mode or 0.33 ms in 100 Mbps mode, back pressure will cease sending preamble for several byte times to avoid the jabber limit.

5.12 Receive filtering

5.12.1 Overview

The Ethernet module can filter out receive packets, analyzing the Ethernet destination address in the packet. This capability greatly reduces the load on the host system, because the many Ethernet packets that are typically addressed to other stations would otherwise have to be inspected and rejected by the device driver software, while using up bandwidth, memory space and host CPU time. Address filtering can be implemented using the perfect address filter or the (imperfect) hash filter. The latter produces a 6-bit hash code which can be used as an index into a 64-entry programmable hash table. In addition to the destination address, other portions of a packet can be included in the filter decision by using a pattern match filter. The result of the pattern-matching filter can be combined with the address filtering to create the final filtering result.

The receive filter has two modes:

- AND mode: The AndOr bit of the RxFilterCtrl register is set to 1. The result of the perfect address filter and the hash filter is ORed into a partial result which is ANDed with the result of the pattern-matching filter. If the pattern-matching filter is not enabled, the filter will not pass any packets.
- OR mode: The AndOr bit of the RxFilterCtrl register is set to 0. The result of the perfect address filter and the hash filter is ORed into a partial result which is ORed with the result of the pattern-matching filter.

Figure 13 depicts a functional view of the receive filter.

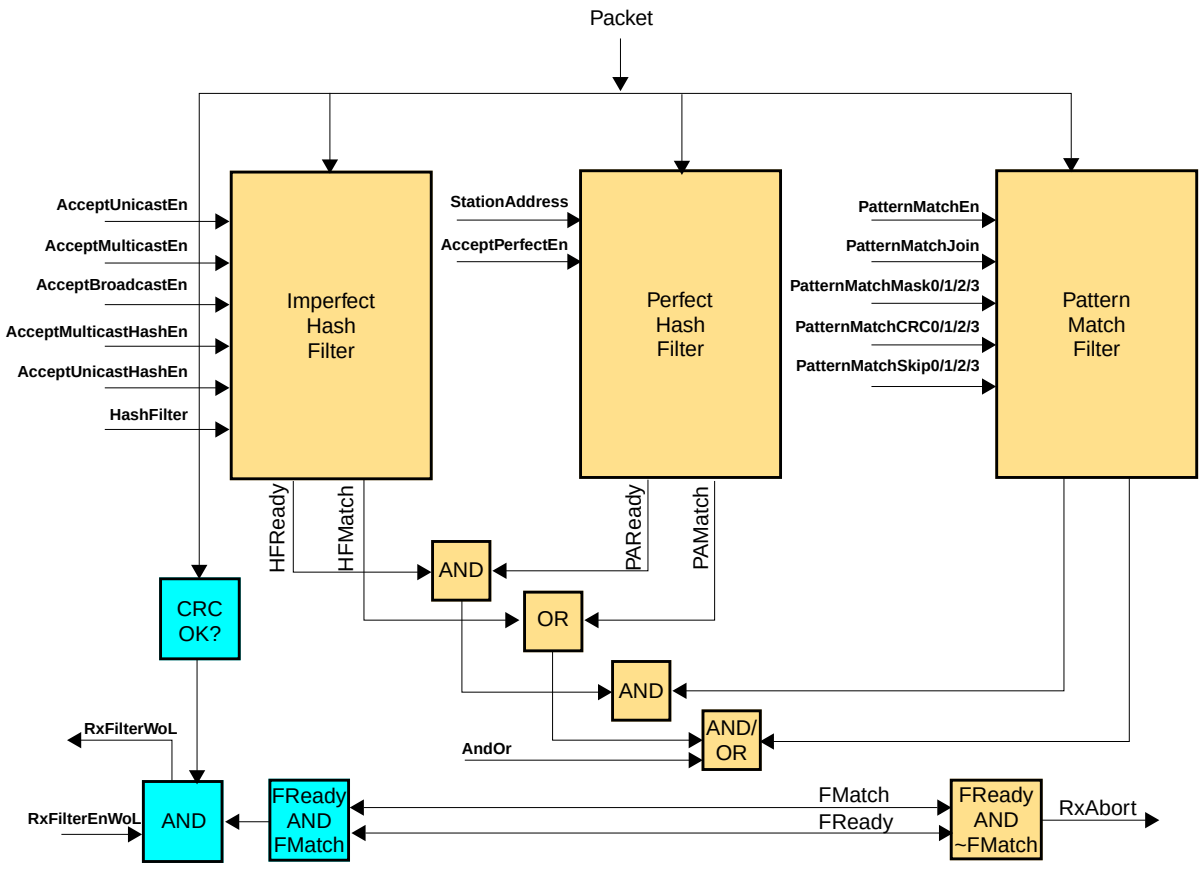


Figure 13: Receive filter block diagram

On the top of the diagram, the Ethernet receive packet enters the filters. Each filter is controlled by signals from the software view. Each filter produces a “Ready” output and a “Match” output. If Ready is 0, then the Match value is “don’t care”; a filter will assert its Ready output when it is finished. If the filter finds a packet that matches, it will assert its Match output along with its Ready output. The results of the filters are combined by logic functions into a single RxAbort output. If the RxAbort output is asserted, the packet need not be received.

The blocks in the left-hand side of the diagram are used for WoL and are discussed in Section 5.13.

In order to reduce memory traffic, the Receive Datapath of the LAN100 has a FIFO buffer that stores 68 bytes of receive data, so the LAN100 will only start writing the received packet to memory after a 68-byte delay. If the RxAbort signal is asserted during the initial 68 bytes of the packet, the packet can be discarded. The packet will be removed from the buffer and not stored to memory at all, thereby also not using any receive descriptors. If the RxAbort signal is asserted *after* the initial 68 bytes in a packet, part of the packet is already written to memory, and the LAN100 will stop writing further data in the packet to memory; the FailFilter bit in the status word of the packet will be set to indicate the software device driver can discard the packet immediately.

5.12.2 Unicast, Broadcast and Multicast

Generic filtering based on the type of packet (where the types are unicast, multicast or broadcast) can be programmed using the AcceptUnicastEn, AcceptMulticastEn and AcceptBroadcastEn bits from the RxFilterCtrl register. Setting the AcceptUnicast, AcceptMulticast and AcceptBroadcast bits causes all packets of types unicast, multicast and broadcast, respectively, to be accepted, ignoring the Ethernet destination address in the packet. To program “promiscuous mode”, that is, to accept all packets, set all 3 bits to 1.

5.12.3 Perfect Address Match

When a packet with a unicast destination address is received, a perfect filter compares the destination address with the 6-byte station address (programmed in the Station Address registers SA0, SA1, and SA2). If the AcceptPerfectEn bit in the RxFilterCtrl register is set to 1, and the address matches, the packet is accepted.

5.12.4 Imperfect Hash Filtering

An imperfect filter is available, based on a hash mechanism. This filter applies a hash function to the destination address and uses the hash to access a table that indicates if the packet should be accepted. The advantage of this type of filter is that a small table can cover any possible address. The disadvantage is that the filtering is imperfect, meaning that sometimes packets are accepted that should have been discarded.

Hash Function

The standard Ethernet cyclic redundancy check (CRC) function is calculated from the 6-byte destination address in the Ethernet packet (this CRC is calculated anyway as part of calculating the CRC of the whole packet), then bits [28:23] out of the 32 bits CRC result are taken to form the hash. The 6-bit hash is used to access the hash table: it is used as an index in the 64-bit HashFilter register that has been programmed with values that are to be accepted. If the selected value is set to 1, the packet is accepted.

The device driver can initialize the hash filter table by writing to the registers HashFilterL and HashfilterH. HashFilterL contains bits 0 through 31 of the table and HashFilterH contains bit 32 through 63 of the table. So, hash value 0 corresponds to bit 0 of the HashfilterL register and hash value 63 corresponds to bit 31 of the HashFilterH register.

Multicast and Unicast

The imperfect hash filter can be applied to multicast addresses by setting the AcceptMulticastHashEn bit in the RxFilter register to 1.

The same imperfect hash filter that is available for multicast addresses can also be used for unicast addresses. This is useful in order to respond to a multitude of unicast addresses without enabling all unicast addresses. The hash filter can be applied to unicast addresses by setting the AcceptUnicastHashEn bit in the RxFilter register to 1.

5.12.5 Pattern Match Filtering and Logic Functions

The LAN100 has four pattern-matching filters. Each filter is capable of covering 64-byte window in any location specified by PatternMatchSkip0/1/2/3[10:0] registers. The pattern match filters analyze received packets for certain patterns. If the filter finds a match, the filter asserts its match signal. Depending on the Join function and the “AndOr” relation with other filters, the packet is then accepted or rejected.

The pattern-matching filters are imperfect filters. They calculate a 32-bit CRC on a 64-byte window. The window can have an offset as specified by the associated PatternMatchSkip register. The Skip register is 11 bits wide, allowing up to 2047 bytes at the start of the packet to be skipped. The PatternMatchMask registers specify a mask for the pattern matching windows so that some bytes can be masked out in the CRC calculation. A byte is only input in the CRC calculation if the corresponding byte enable bit in the PatternMatchMask register is set. A pattern match filter produces a match if the calculated CRC matches the CRC in the associated PatternMatchCRC register. Each of the four pattern match filters is enabled by setting the corresponding PatternMatchEn[3:0] bit in the RxFilter register to 1.

The same 32-bit CRC and polynomial are used as with the standard Ethernet CRC.

The PatternMatchMask register is 64 bits long, allowing up to 64 consecutive bytes of the received packet to be included in the CRC calculation. When bit *b* in the PatternMatch register is 1 and the value of the Skip register is *s*, then receive packet byte number (*s*+*b*) is included in the CRC calculation. Counting starts at the destination address field of the Ethernet MAC frame. The PatternMatchMask register is split into low-order and high-order parts, and registers PatternMatchMaskL and PatternMatchMaskH are each 32 bits wide.

There are four separate pattern-matching filters supported, and there are 4 sets of related registers (PatternMatchMaskL0/1/2/3, PatternMatchMaskH0/1/2/3, PatternMatchCRC0/1/2/3, PatternMatchSkip0/1/2/3), one set for each filter.

The PatternMatchJoin register bits allow several of the pattern matches to be combined together to implement a more complex combined check. This can be used to create a pattern match of more than 64 bits, and the multiple pattern matches do not need to check adjacent portions of the receive packet.

Section 3.3 defines the PatternMatchJoin register.

For example, if the PatternMatchJoin register is set to 0xAFAFF0, then the results of the pattern-match filter 0 and pattern-match filter 1 will be ANDed together, while the results of filter 2 and 3 will be ignored. If the PatternMatchJoin register is set to 0x7A1BB1, then the join logic will perform the complex function:

$$(a \,\&\, !b) \mid c \mid !d$$

where *a*, *b*, *c*, *d* are the results from the 0/1/2/3 filters.

A pattern match is enabled by setting the corresponding PatternMatchEn bit in the RxFilterWOLControl register to 1. This can be done separately for each of the four filters. If the system is in power-down mode, if a pattern match detects a wake-up event, the corresponding PatternMatch bit in the RxFilterStatus register is set. If the

RxFilterWOLEn is also enabled, a wake up event will cause an interrupt to the system, and the RxFilterWOL in the RxFilterStatus register will be also be set. All the status bits in RxFilterStatus must be cleared by software after the system is powered up. However when the system is powered up and is receiving packets normally, the corresponding PatternMatch bit in the RxFilterStatus register for each one of the four pattern-match filters is put in status display mode for the incoming packet. They are read-only, and there is no need to clear them. When multiple pattern matches are joined, their PatternMatch bits are all set together based on the outcome of the joined pattern match.

5.12.6 Enabling and Disabling Filtering

The pattern-matching filters described in the previous sections can be bypassed by setting the PassRxFilter bit in the Command register. When the PassRxFilter bit is set, all receive packets will be passed to memory. In this case the device driver software must implement all filtering functionality in software.

Setting the PassRxFilter bit does not affect the runt frame filtering as defined in the next section.

5.12.7 Runt Frames

A packet with less than 64 bytes (or 68 bytes for VLAN frames) is shorter than the minimum Ethernet frame size, and therefore is considered erroneous. For example, they might be collision fragments. The Receive Datapath automatically filters and discards runt frames without writing them to memory and without using a receive descriptor.

When a runt frame has a correct CRC, there is a possibility that it is intended to be useful. The device driver can receive runt frames with correct CRC by setting the PassRuntFrame bit of the Command register to 1.

5.13 Wake-up on LAN

5.13.1 Overview

The Ethernet module supports power management with remote wake-up over a local area network. The host system of the Ethernet module can be powered down, even including part of the LAN100 Ethernet module itself, while the Ethernet module continues to listen to packets on the LAN. Packets that are appropriately formed can be received and recognized by the Ethernet module, and can be used to trigger the host system to wake up from its power-down state.

System wake-up takes effect through an interrupt. When a wake-up event is detected, the WakeupInt bit in the IntStatus register is set. The interrupt status will trigger an interrupt if the corresponding WakeupIntEn bit in the IntEnable register is set.

While in a power-down state, the packet that generates a Wake-up-on-LAN event is lost.

There are two ways in which Ethernet packets can trigger wake-up events: generic Wake-up on LAN and Magic Packet. The generic Wake-up-on-LAN functionality is based on the filtering as defined in Section 5.12; magic packet filtering uses an

additional filter for magic packet detection. In both cases, a WoL event is only triggered if the triggering packet has a valid CRC. Figure 13 illustrates the wake-up functionality.

The following subsections describe the two WoL mechanisms.

The **RxFilterWoLStatus** register can be read by the software to inspect the reason for a Wake-up event. Before going into a power-down condition, the power management software should clear the register by writing the RxFilterWolClear register.

5.13.2 Filtering for WoL

The receive filter functionality as described in Section 5.12 can be used to generate WoL events. If the RxFilterEnWoL bit of the RxFilterCtrl register is set, the receive filter will set the WakeupInt bit of the IntStatus register if a packet is received that passes the filter. The interrupt will only be generated if the CRC of the packet is correct.

In case a wake-up is generated by the receive filter, the RxFilterWoL bit in the RxFilterWoLStatus register will be set along with the origin of the filter match, which will be set in the AcceptPerfectWoL, AcceptMulticastHashWoL, AcceptUnicastHashWoL, AcceptMulticastWoL, AcceptBroadcastWoL, AcceptUnicastWoL bits of the same register. Software can reset these bits by writing a 1 to the corresponding bits of the RxFilterWoLClear register.

5.13.3 Magic Packet WoL

The LAN100 supports wake-up using AMD's Magic Packet technology (see “Magic Packet technology”, Advanced Micro Devices). A Magic Packet is a specially formed packet solely intended for wake-up purposes. This packet can be received, analyzed and recognized by the Ethernet module and used to trigger a wake-up event.

A packet is a Magic Packet if it contains the station address repeated 16 times in its data portion with no breaks or interruptions, preceded by six synchronization bytes with the value 0xFF. Other data may surround the Magic Packet pattern in the data portion of the packet. The whole packet must be a well-formed Ethernet frame.

The magic packet detection unit analyzes the Ethernet packets, extracts the packet address and checks the payload for the Magic Packet pattern. The address from the packet is used for matching the pattern (not the address in the SA0/1/2 registers). A magic packet only sets the wake-up interrupt status bit if the packet passes the receive filter, as illustrated in figure Figure 13: the result of the receive filter is ANDed with the magic packet filter result to produce the final result.

Magic Packet filtering is enabled by setting the MagicPacketEnWoL bit of the RxFilterCtrl register. Note that when doing Magic Packet WoL, the RxFilterEnWoL bit in the RxFilterCtrl register should be 0. Setting the RxFilterEnWoL bit to 1 would accept all packets for a matching address, not just the Magic Packets, that is, WoL using Magic Packets is more strict.

When a magic packet is detected, apart from the WakeupInt bit in the IntStatus register, the MagicPacketWoL bit is set in the RxFilterWoLStatus register. Software can reset the bit by writing a 1 to the corresponding bit of the RxFilterWoLClear register.

Magic Packet WoL Example

An example of a Magic Packet with station address 11h 22h 33h 44h 55h 66h is as follows. MISC indicates miscellaneous additional data bytes in the packet.

DESTINATION SOURCE MISC FF FF FF FF FF

FF 11 22 33 44 55 66 11 22 33 44 55 66 11 22 33 44

55 66 11 22 33 44 55 66 11 22 33 44 55 66 11 22 33

44 55 66 11 22 33 44 55 66 11 22 33 44 55 66 11 22

33 44 55 66 11 22 33 44 55 66 11 22 33 44 55 66 11

22 33 44 55 66 11 22 33 44 55 66 11 22 33 44 55 66

11 22 33 44 55 66 11 22 33 44 55 66 MISC CRC

5.14 Enabling and Disabling Receive and Transmit

5.14.1 Enabling and Disabling Reception

After reset, the receive function of the LAN100 is disabled. The receive function can be enabled by the device driver by setting the RxEnable bit in the Command register and the RECEIVE_ENABLE bit in the MAC1 configuration register of the LAN100.

The status of the Receive Datapath can be monitored by the device driver by reading the RxStatus bit of the Status register. Figure 14 illustrates the finite state machine (FSM) for generating the RxStatus bit.

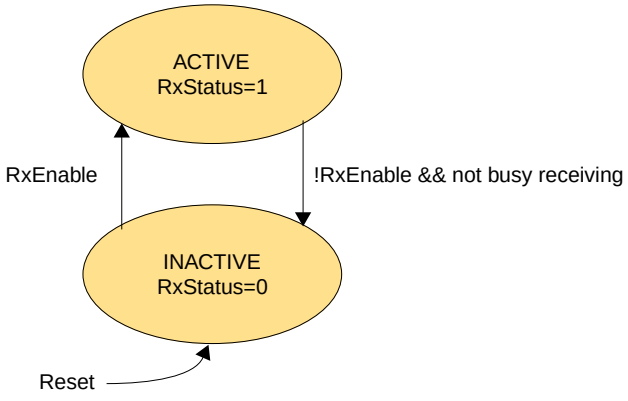


Figure 14: Receive Active/Inactive state machine

After a reset, the FSM is in the INACTIVE state. As soon as the RxEnable bit is set in the Command register, the FSM transitions to the ACTIVE state. As soon as the RxEnable bit is cleared, the FSM returns to the INACTIVE state. If the Receive Datapath is busy receiving a packet when it is put in the disabled state, packet reception will continue and the packet will be received completely and stored to memory along with its status before it returns to the INACTIVE state.

As shown in Figure 14, after a soft reset (see Section 5.19.2), the Receive Datapath is inactive until the datapath is re-enabled.

5.14.2 Enabling and Disabling Transmission

After reset, the transmit function of the LAN100 is disabled. The Transmit Datapaths (Tx and TxRt) must be enabled separately. The device driver enables the Tx datapath by setting the TxEnable bit in the Command register to 1. The device driver enables the TxRt datapath by setting the TxRt bit in the Command register to 1.

The status of the Transmit Datapaths can be monitored by the device driver by reading the TxStatus and TxRtStatus bits of the Status register. The FSM for both Tx and TxRt are the same, as shown in Figure 15. .

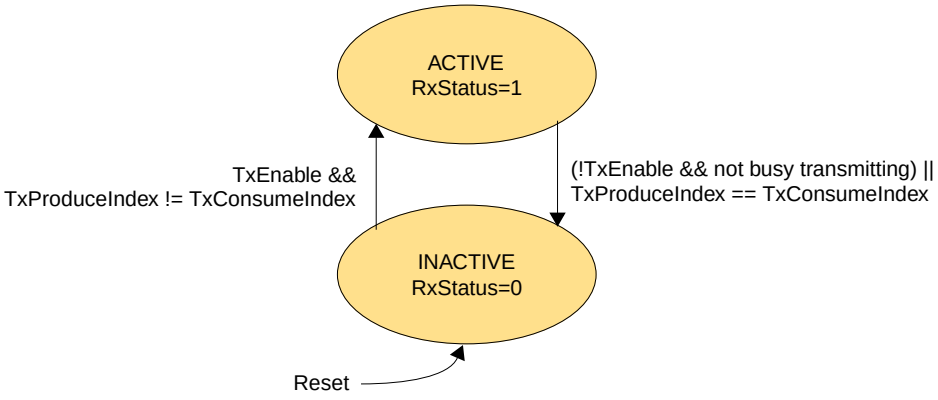


Figure 15: Transmit Active/Inactive state machine

After reset, the FSMs are in the INACTIVE state. As soon as the Tx(Rt)Enable bit is set in the Command register and the Produce and Consume indices are not equal, the FSM transitions to the ACTIVE state. As soon as the Tx(Rt)Enable bit is cleared and the Transmit Datapath has completed all pending transmissions including committing the transmission status to memory, the FSM returns to the INACTIVE state. The FSM will also return to the INACTIVE state if the Produce and Consume indices are equal again, that is, when all packets have been transmitted.

As shown in Figure 15, the Transmit Datapath is inactive after a soft reset (see Section 5.19.2) until the datapath is re-enabled.

5.15 Transmission Padding and CRC

In the event that a packet is received with a length of less than 60 bytes (or 64 bytes for VLAN frames) the LAN100 can pad the packet to 64 or 68 bytes, including a 4-byte CRC Frame Check Sequence (FCS). Padding is affected by the value of the

AUTO_DETECT_PAD_ENABLE (ADPEN), VLAN_PAD_ENABLE (VLPEN) and PAD_CRC_ENABLE (PADEN) bits of the MAC2 configuration register as well as the Override and Pad bits from the transmit descriptor Control word. CRC generation is affected by the CRC_ENABLE (CRCE) and DELAYED_CRC (DCRC) bits of the MAC2 configuration register and the Override and CRC bits from the transmit descriptor Control word.

The effective pad enable (EPADEN) is equal to the PAD_CRC_ENABLE bit from the MAC2 register if the Override bit in the descriptor is 0. If the Override bit is 1, then EPADEN will be taken from the descriptor Pad bit. Likewise, the effective CRC enable (ECRCE) equals CRCE if the Override bit is 0, otherwise it equal the CRC bit from the descriptor.

If padding is required and enabled, a CRC will always be appended to the padded frames. A CRC will only be appended to the non-padded frames if ECRCE is set.

If EPADEN is 0, the packet will not be padded, and no CRC will be added unless ECRCE is set.

If EPADEN is 1, then small packets will be padded, and a CRC will always be added to the padded frames. In this case, if ADPEN and VLPEN are both 0, then the packets will be padded to 60 bytes, and a CRC will be added, creating 64-byte packets. If VLPEN is 1, the packets will be padded to 64 bytes and a CRC will be added, creating 68 bytes packets. If ADPEN is 1 while VLPEN is 0, VLAN packets will be padded to 64 bytes, non-VLAN packets will be padded to 60 bytes, and a CRC will be added to padded packets creating 64- or 68-byte padded packets.

In case CRC generation is enabled, CRC generation can be delayed by four bytes to skip proprietary header information.

5.16 Huge Frames and Frame Length Checking

The HUGE_FRAME_ENABLE bit in the MAC2 configuration register can be set to 1 to enable transmission and reception of packets of any length. Huge frame transmission can be enabled on a per-packet basis by setting the Override and Huge bits in the transmit descriptor Control word.

When enabling huge frames, the LAN100 will not check frame lengths or report frame length errors (RangeError and LengthError). If huge frames are enabled, the received byte count in the RSV register may be invalid because the frame may exceed the maximum size. However, the EntryLevel fields from the receive status arrays will be valid.

The LAN100 will check frame lengths by comparing the length/type field of the packet to the actual number of bytes in the packet, and report a LengthError by setting the corresponding bit in the receive StatusInfo word.

The MAXF register in the LAN100 allows the device driver to specify the maximum number of bytes in a frame. The LAN100 will compare the actual receive frame to the MAXF value and report a RangeError in the receive StatusInfo word if the packet is larger.

5.17 Statistics Counters

In Ethernet applications generally, many counters that keep Ethernet traffic statistics must be maintained. There are a number of standards specifying such counters, such as IEEE Std 802.3 / Clause 30. Other standards are RFC 2665 and RFC 2233.

The approach taken here is as follows: by default, all counters are implemented in software. With the help of the StatusInfo field in the packet status word, many of the important statistics events listed in the standards can be counted by software. Currently, no counters are added in hardware.

5.18 Status Vectors

A transmit status vector and a receive status vector are available in registers TSV0, TSV1, and RSV. These registers can be polled with software. Normally, they are of limited use, because the communication between driver software and Ethernet module takes place primarily through the packet descriptors. Statistical events are counted by software in the device driver. However, these transmit and receive status vectors are made visible for debug purposes. The values in these registers are simple copies of the transmit and receive status vectors as produced by the MII Interface. They are valid as long as the status vectors are valid, and should typically only be read when the transmit and receive processes are halted.

The TSV0, TSV1 and RSV registers are defined in Section 3.2.

5.19 Reset

The LAN100 has a hard reset input and several soft resets which can be activated by setting the appropriate bits in its registers. All registers in the LAN100 have a reset value of 0 unless specified differently in Section 3.2.

5.19.1 Hard Reset

The LAN100 module experiences a hard reset when the PNX15xx Series is reset. After a hard reset, all register values in the software view will be set to their default value as specified in Section 3.2.

5.19.2 Soft Reset

Parts of the LAN100 can be reset by software by setting bits in the Command register and the MAC1 configuration register.

The MAC1 register has six different reset bits:

- SOFT_RESET: Setting this bit will put all components in the MII Interface in the reset state except for the MII Interface registers (in address locations 72 0000 to 72 00FC). The soft reset bit defaults to 1, and must be cleared after a hardware reset to enable the MII Interface.
- SIMULATION_RESET: Setting this bit resets the random number generator in the Transmit Function. The value after a hard reset is 0.

- RESET_PEMCS_Rx: Setting this bit will reset the MAC Control Sublayer (the pause frame logic) and the Receive function of the MII Interface. The value after a hard reset is 0.
- RESET_PERFUN: Setting this bit will reset the receive function in the MII Interface. The value after a hard reset is 0.
- RESET_PEMCS_Tx: Setting this bit will reset the MAC Control Sublayer (pause frame logic) and the Transmit function of the MII Interface. The value after a hard reset is 0.
- RESET_PETFUN: Setting this bit will reset the transmit function of the MII Interface. The value after a hard reset is 0.

All the above reset bits must be cleared by software.

The RESET_PERMII bit in the SUPP register allows soft resetting of the RMII logic. The reset must be cleared by software.

The Command register (see Table 2) has three different reset bits:

- TxReset: Writing a 1 to the TxReset bit will reset the Transmit Datapath, excluding the MII Interface portions, including all (read-only) registers in the Transmit Datapath, and also the TxProduceIndex register in the host registers module. Soft resetting the Transmit Datapath will abort all memory operations of the Transmit Datapath. The reset bit will be cleared automatically by the LAN100. Soft resetting the Tx datapath will clear the TxStatus and TxRtStatus bits in the Status register.
- RxReset: Writing a 1 to the RxReset bit will reset the Receive Datapath, excluding the MII Interface portions, including all (read-only) registers in the Receive Datapath, and also the RxConsumeIndex register in the host registers module. Soft resetting the Receive Datapath will abort all memory operations of the Receive Datapath. The reset bit will be cleared automatically by the LAN100. Soft resetting the Rx datapath will clear the RxStatus bit in the Status register.
- RegReset: Writing a 1 to the RegReset bit resets all of the datapaths and LAN100 registers, but not the registers in the MII Interface. Soft resetting the registers aborts all memory operations of the Transmit and Receive datapaths. The reset bit will be cleared automatically by the LAN100.

To do a full soft reset of the LAN100 device driver, software must:

- Set the SOFT_RESET bit in the MAC1 register to 1
- Set the RegReset bit in the Command register (this bit clears automatically)
- Reinitialize the MII Interface registers (0x000-0x0FC)
- Clear the SOFT_RESET bit in the MAC1 register to 0.

To reset just the Transmit Datapath, the device driver software must:

- Set the RESET_PEMCS_Tx bit in the MAC1 register to 1

- Disable the Tx DMA managers by setting the Tx(Rt)Enable bits in the Command register to 0
- Set the TxReset bit in the Command register (this bit clears automatically)
- Reset the RESET_PEMCS_Tx bit in the MAC1 register to 0.

To reset just the Receive Datapath the device driver software must:

- Disable the receive function by resetting the RECEIVE_ENABLE bit in the MAC1 configuration register and resetting of the RxEnable bit of the Command register.
- Set the RESET_PEMCS_Rx bit in the MAC1 register to 1
- Set the RxReset bit in the Command register (this bit clears automatically)
- Reset the RESET_PEMCS_Rx bit in the MAC1 register to 0.

A soft reset of the Transmit Datapaths will abort any transmit-descriptor read, status-write, and data-read operations.

A soft reset of the Receive Datapath will abort any receive-descriptor read, status-write and data-write operations.

6. System Integration

6.1 MII Interface I/O

Table 12 summarizes the pin interface of the LAN100.

Table 12: LAN100 Pin Interface to external PHY

Pin	Direction	Description
LAN_CLK	Out	Clock to feed the external PHY, usually 50 MHz
LAN_TX_CLK/LAN_REF_CLK	In	MII Transmit Clock or RMII Reference Clock
LAN_TX_EN	Out	MII or RMII Transmit Enable
LAN_TXD3	Out	MII Transmit Data
LAN_TXD2		MII Transmit Data
LAN_TXD1		MII or RMII Transmit Data
LAN_TXD0		MII or RMII Transmit Data
LAN_TX_ER	Out	MII Transmit Error
LAN_CRS/LAN_CRS_DV	In	MII Carrier Sense or RMII Carrier Sense and Receive Data Valid. **This pin is 5 V input tolerant.**
LAN_COL	In	Collision Detect. **This pin is 5 V input tolerant.**
LAN_RX_CLK	In	MII Receive Clock
LAN_RXD3	In	MII Receive Data
LAN_RXD2		MII Receive Data
LAN_RXD1		MII or RMII Receive Data
LAN_RXD0		MII or RMII Receive Data
LAN_RX_DV	In	MII Receive Data Valid

Table 12: LAN100 Pin Interface to external PHY *…Continued*

Pin	Direction	Description
LAN_RX_ER	In	MII or RMII Receive Error
LAN_MDIO	In/Out	MII Management data I/O
LAN_MDC	Out	MII Management Data clock

6.2 Power Management

The LAN100 supports power management by means of external clock switching. Basically, all clocks in the LAN100 module can be switched off. If WoL is needed, the receive clock should not be switched off.

The LAN100 supports two power management modes:

- *Sleep mode*: the PNX15xx Series is active while most clocks are switched off. The CPU is active and can still communicate with the LAN100 via MMIO registers.
- *Coma mode*: most of the clocks in the PNX15xx Series are switched off, including the LAN100, CPU, and MMIO clock.

6.2.1 Sleep Mode

The LAN100 can be put in sleep mode by setting the PowerDown bit in the PowerDown register if the receive and transmit DMA managers are disabled and inactive. Clocks should only be disabled after software has set the PowerDown bit. Software should prevent access to components of the LAN100 that have been switched off. If an external PHY is connected, a WoL can still trigger an interrupt.

To enter sleep mode, software should:

- Disable both transmit DMA managers and the receive DMA manager by writing the Command register
- Wait for the transmit and Receive Datapaths to be inactive by waiting for a Finished interrupt or by polling the Status register.
- Set the PowerDown bit by writing to the PowerDown register.

If no PHY is connected software can directly set the PowerDown bit and disable the clocks.

To exit sleep mode sofware should:

- Reset the PowerDown bit
- Reenable the receive and Transmit Datapaths.

6.2.2 Coma Mode

In coma mode, most of the clocks in the PNX15xx Series can be switched off including the CPU and MMIO clocks. If an external PHY is connected, the receive filter and WoL detection will still be active and capable of generating a WoL interrupt to the system's power-management controller.

To enter coma mode, software should first enter sleep mode. The system's power-management controller can then disable the MMIO clock. To exit coma mode, software should re-enable the MMIO clock before following the wake-up step from Section 6.2.1.

6.3 Disabling the LAN100

In implementations that do require Ethernet functionality, the external PHY can be omitted, and the I/O pins from the PHY, including the clock inputs, can be tied to 0 or 1. In this case, software should switch the LAN100 to sleep mode by setting the PowerDown bit in the PowerDown register.

6.4 Little/big Endian

The LAN100 memory interfaces take care of the conversion to the endian mode of the PNX15xx Series system buses. When packing multiple Ethernet bytes into a word in order to optimize bus traffic, the LAN100 orders the bytes as required by the system bus. Internally, all components of the LAN100 use bytes as the native width for transmission and reception of data. Therefore, the LAN100 itself is endian insensitive.

6.5 Interrupts

The LAN100 has a single interrupt request output directed to the CPU.

After taking the interrupt, the CPU's interrupt service routine must read the IntStatus register to locate the origin of the interrupt. Software interrupt conditions can be set by writing to the IntSet register; interrupt conditions can be cleared by writing to the IntClear register.

The transmit and Receive Datapaths can only set interrupt status, they cannot clear status. The SoftInt interrupt cannot be set by hardware and can be used by software for test purposes.

For more information on the source of an interrupt refer to Section 5.4.8 and Section 5.5.7.

6.6 Errors and Aborts

Several conditions cause the LAN100 to generate errors:

- An illegal MMIO access can cause an MMIO error response as defined in Section 5.1. These errors are handled by the MMIO adapter, which may be propagated back to the CPU across the system bus.
  If the PowerDown bit is set, any access to MMIO registers will result in an error response except for access to the PowerDown register.
- Packet reception can cause errors, including AlignmentError, RangeError, LengthError, SymbolError, CRCError, NoDescriptor, and Overrun. These errors are reported in the receive status word and in the interrupt status register. For more information refer to Section 4.1 and Section 5.5.

- Packet transmission can cause errors including LateCollision, ExcessiveCollision, ExcessiveDefer, NoDescriptor, and Underrun. These errors are reported back in the transmission status word and in the interrupt status register. For more information refer to Section 4.2 and Section 5.4.

6.7 Cache coherency

Because the device driver can produce descriptors with write-only operations and consume the status fields with read-only operations, transfers can be made “cache safe,” and descriptors can be packed together in cache blocks and cached. Status words can also be packed together. The device driver must take care of cache coherency if cache coherency is not enforced by a snooping cache in the host processor.

If the device driver doesn’t own all of the status words in a cache block that contains multiple status words, then the Ethernet hardware may be writing to a status field in memory while that status is also included in a cache block loaded in the host processor’s cache, causing the values for that status in the host cache to be stale. This can be solved by invalidating these cache blocks and causing them to be read again from the host memory whenever the device driver receives ownership of these status words, that is, when the Tx(Rt)ConsumeIndex or RxProduceIndex are updated.

Before updating the Tx(Rt)ProduceIndex or RxConsumeIndex, the device driver must make sure the associated descriptors are written back from the cache to the memory so that the new descriptor values become visible to the Ethernet hardware. When flushing these cache blocks, care must be taken not to modify the fields of descriptors that are already owned by the Ethernet hardware.

Alternatively, the device driver can use non-cached memory traffic for the descriptors and statuses. In that case, there is no need to worry about cache coherency, however, a higher amount of memory traffic may be required for the descriptors and status words.

REFERENCES

[1] IEEE Std 802.3, 2000 Edition, (Incorporating IEEE Std 802.3, 1998 Edition, IEEE Std 802.3ac-1998, IEEE Std 802.3ab-1999, and 802.3ad-2000)IEEE standard 802.3

[2] MII Interface Packet Engines 10/100 Mbps Ethernet Media Access Controller with MII Management Interface, Release 3.5, 2001

[3] PE-RMII Reduced Media Independent Interface for the Alcatel 10/100 Mb/s Ethernet Media Acces Controller, release 3.2, 2002.

Chapter 24: TM3260 Debug

PNX15xx Series Data Book – Volume 1 of 1

Rev. 2 — 1 December 2004 Product data sheet

1. Introduction

The TM3260 Debug (TM_DBG) interface consists of the Test Access Port (TAP), the TAP Controller, a JTAG Instruction register and internal debug registers. The TAP controller from which the TM_DBG module receives its commands resides in the test control block, which also facilitates boundary scanning and other DFT features.

1.1 Features

The TM_DBG has registers that can be programmed for control and communication with an on-chip TriMedia TM3260 CPU.

2. Functional Description

2.1 General Operations

2.1.1 Test Access Port (TAP)

The Test Access Port (TAP) includes four dedicated input pins and one output pin:

- TCK (Test Clock)
- TMS (Test Mode Select)
- TDI (Test Data In)
- TDO (Test Data Out)

TCK provides the clock for test logic required by the JTAG standard. TCK is asynchronous to any system clock. Stored state devices in the JTAG controller will retain their state indefinitely when TCK is stopped at 0 or 1.

The signal received at TMS is decoded by the TAP controller to control test functions. The test logic is required to sample TMS at the rising edge of TCK.

Serial test instructions and test data are received at TDI. The TDI signal is required to be sampled at the rising edge of TCK. When test data is shifted from TDI to TDO, the data must appear without inversion at TDO after a number of rising and falling edges of TCK determined by the length of the instruction or test data register selected.

TDO is the serial output for test instructions and data from the TAP controller. Changes in the state of TDO must occur at the falling edge of TCK. This is because devices connected to TDO are required to sample TDO at the rising edge of TCK. The TDO driver must be in an inactive state (i.e., TDO line HIghZ) except when the scanning of data is in progress.

2.1.2 TAP Controller

The TAP controller is a finite state machine that synchronously responds to changes in TCK and TMS signals. The TAP instructions and data are serially scanned into the TAP controller's instruction and data registers via the common input line TDI. The TMS signal tells the TAP controller to select either the TAP instruction register or a TAP data registers as the destination for serial input from the common line TDI. An instruction scanned into the instruction register selects a data register to be connected between TDI and TDO to become the destination for serial data input.

The TAP controller's state changes are determined by the TMS signal. The states are used for scanning in/out TAP instruction and data, updating instruction, and data registers, and for executing instructions.

The controller's state diagram (Figure 1) shows separate states for "Capture," "Shift" and "Update" of data and instructions in order to leave the contents of a data register or an instruction register undisturbed until serial scan in is finished and the Update state is entered. By separating the Shift and Update states, the contents of a register (the parallel stage) is not affected during scan in/out.

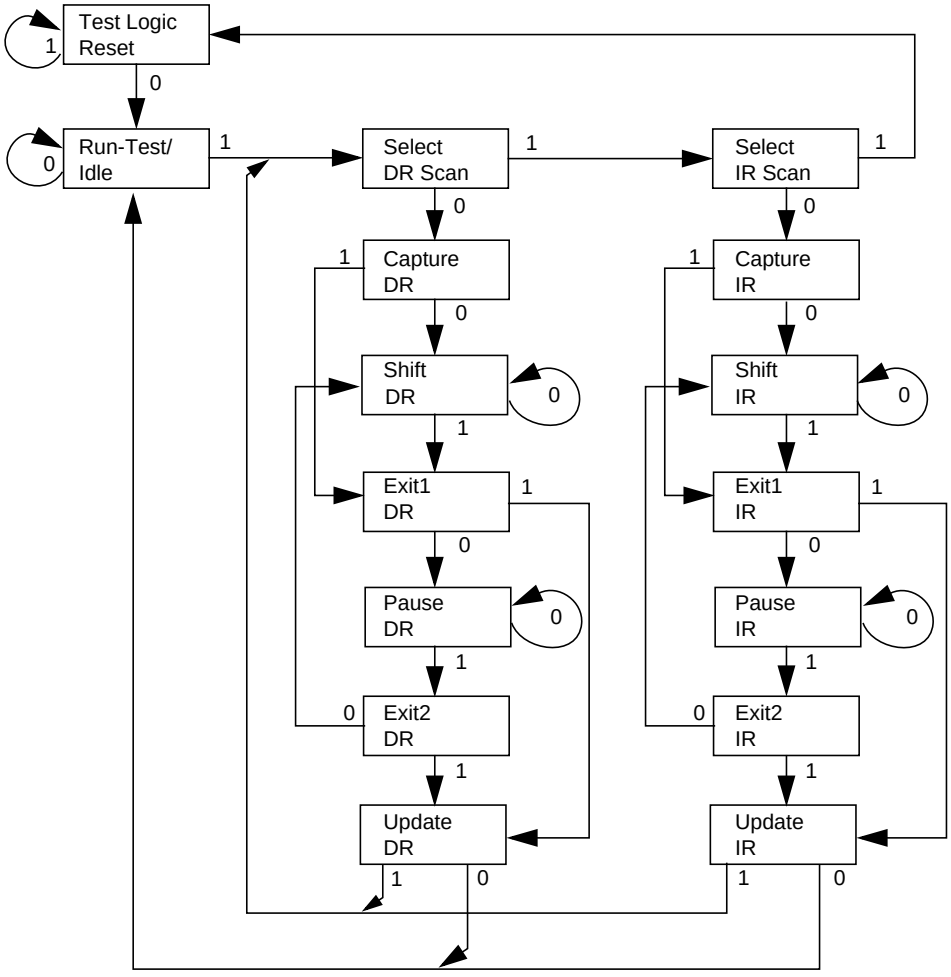


Figure 1: State Diagram of TAP Controller

The TAP controller must be in Test Logic Reset state after power up. It remains in that state as long as TMS is held at 1. The controller transitions to Run-Test/Idle state from Test Logic Reset state when TMS = 0. The Run-Test/Idle state is an idle state of the controller in between scanning in/out an instruction/data register. The "Run-Test" part of the name refers to start of built-in tests. The "Idle" part of the name refers to all other cases. Note that there are two similar substructures in the state diagram, one for scanning in an instruction and another for scanning in data. To scan in/out a data register, one has to scan in an instruction first.

An instruction or data register must have at least two stages, the shift register stage and the parallel input/output stage. When an n-bit data register is to be "read," the register is selected by an instruction. The register's contents are "captured" first

(loaded in parallel into the shift register stage), n bits are shifted in and simultaneously, n bits are shifted out. Finally the register is “updated” with the new n bits shifted in.

Note that when a register is scanned, its old value is shifted out of TDO and the new value shifted in via TDI is written to the register at the Update state. Hence, scan in/out involve the same steps. This also means that reading a register via JTAG destroys its contents unless otherwise stated.

Some registers are specified as read-only via JTAG so that when the controller transitions to the Update state for the read-only register, the update has no effect. Some times, read/write registers are needed (e.g., control registers used for handshake) which must be read non-destructively. In such cases, the value shifted in determines whether the old value is “remembered” or something else happens.

2.1.3 PNX15xx Series JTAG Instruction Set

PNX15xx Series uses a 5-bit JTAG instruction register. The unspecified opcodes are private and their effects are undefined. Table 1 lists the JTAG instructions related to TM_DBG module.

Table 1: JTAG TM3260 Instruction Encoding

Encoding	Instruction name	Action
10001	TM_DBG_SEL_DATA_IN	Select TM_DBG_DATA_IN register
10010	TM_DBG_SEL_DATA_OUT	Select TM_DBG_DATA_OUT register
10011	TM_DBG_SEL_IFULL_IN	Select TM_DBG_IFULL_IN register
10100	TM_DBG_SEL_OFULL_OUT	Select TM_DBG_OFULL_OUT register
10101	TM_DBG_SEL_TM_DBG_CTL1	Select TM_DBG_CTRL register 1
10110	TM_DBG_SEL_TM_DBG_CTL2	Select TM_DBG_CTRL register 2

The standard JTAG instructions such as EXTEST, SAMPLE/PRELOAD, BYPASS and IDCODE are implemented and listed in Table 2.

Table 2: JTAG Instruction Encoding

Encoding	Instruction Name	Action
00000	EXTEST	Select boundary scan register
00001	SAMPLE	Select boundary scan register
00010	ICODE	Select Identification register
11111	BYPASS	Select bypass register

3. Operation

3.1 Register Programming Guidelines

Because the JTAG data registers live in MMIO space and are accessible by both the TM3260 CPU and the JTAG controller at the same time, race conditions must not exist either in hardware or in software. The communication protocol uses a handshake mechanism to avoid software race conditions.

3.1.1 Handshaking and Communication Protocol

The following describes the mechanism for transferring data via JTAG.

Transfer from Debug Front-End to Debug Monitor

The debugger front-end running on a host transfers data to a debug monitor via the TM_DBG_DATA_IN register. It must poll the TM_DBG_CTRL2.ifull bit to check if the TM_DBG_DATA_IN register can be written to. If the TM_DBG_CTRL2.ifull bit is clear, the front-end may scan data into the TM_DBG_DATA_IFULL_IN register.

Note that data and control bits may be shifted in with SEL_IFULL_IN instruction and the bit shifted into TM_DBG_CTRL2.ifull register must be 1. This action triggers an interrupt. The debug monitor must copy the data from TM_DBG_DATA_IN register into its private area when servicing the interrupt and then clear the TM_DBG_CTRL2.ifull bit. This allows the JTAG interface module to write the next piece of data to the TM_DBG_DATA_IN register.

Transfer from Monitor to Front-End

The monitor running on TM3260 must check if TM_DBG_CTRL1.ofull is clear and if so, it can write data to TM_DBG_DATA_OUT. After that, the monitor must set the TM_DBG_CTRL1.ofull bit. The debugger front-end polls the TM_DBG_CTRL1.ofull bit. When set, it can scan out the TM_DBG_DATA_OUT register and clear the TM_DBG_CTRL1.ofull bit. Since TM_DBG_DATA_OUT is read-only via JTAG, the update action at the end of scan out has no effect on TM_DBG_DATA_ OUT. The TM_DBG_CTRL1.ofull bit however, must be cleared by shifting in the value 1.

Controller States

In the power on reset state, TM_DBG_CTRL2.ifull, TM_DBG_CTRL1.ofull and TM_DBG_CTRL1.sleepless bits are cleared.

Example of Data Transfer via JTAG

Scanning in a 5-bit instruction will take 12 TCK cycles from the Run-Test/Idle state:

- 4 cycles to reach Shift-IR state
- 5 cycles for actual shifting in
- 1 cycle to exit1-IR state
- 1 cycle to update-IR state,
- 1 cycle back to Run-Test/Idle state.

Likewise, scanning in a 32-bit data register will take 38 TCK cycles, and transferring an 8-bit TM_DBG_CTRL data register will take 14 TCK cycles from Idle state. However, if a data transfer follows instruction transfer, then the transition to DR scan stage can be done without going through Idle state, thereby saving 1 cycle.

Transfer of Data to TM3260 via JTAG

Poll control register to check if input buffer is empty or not and scan in data when it is empty and set the ifull control bit to 1 triggering an interrupt. Note that scanning in any instruction automatically scans out the 3 least significant bits (including the ifull or ofull bit) of the selected TM_DBG_CTRL register.

Table 3: Transfer of Data In via JTAG

Action	Number of TCK Cycles
IR shift in SEL_IFULL_IN instruction	12
While TM_DBG_CTRL2.ifull = 1, scan in SEL_IFULL_IN instruction	11+
DR scan 33 bits of register TM_DBG_IFULL_IN	38
TOTAL	61+ cycles

3.2 Debug Settings

Figure 2 shows an overview of the JTAG access path from a host machine to a target PNX15xx Series system and a simplified block diagram of the PNX15xx Series processor. The JTAG Interface Module, shown separately in the diagram, may be a PC add-on card such as PC-1149.1/100F Boundary Scan Controller Board or a similar module connected to a PC serial or parallel port. The JTAG interface module is necessary for PNX15xx Series systems that are not plugged into a PC. For PC-hosted PNX15xx Series systems, the host based TM3260 debugger front-end can communicate with the target resident debug monitor via the PCI bus.

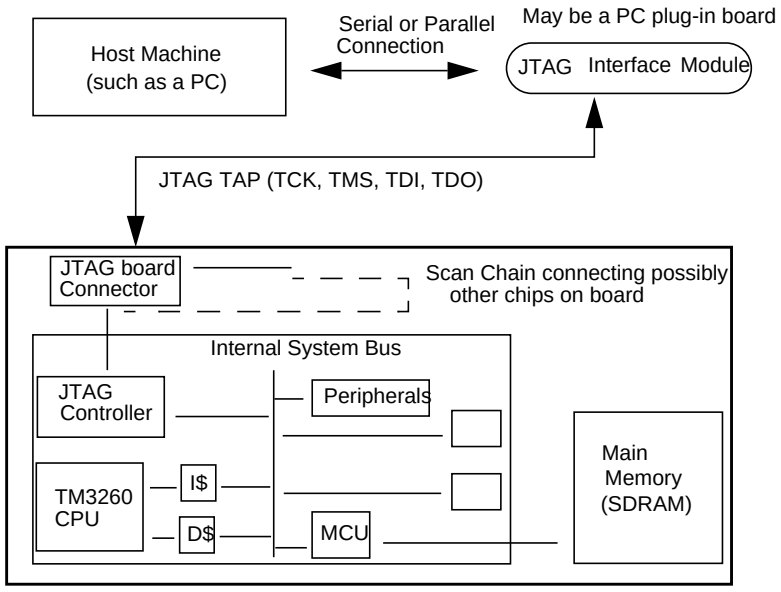


Figure 2: System with JTAG Access

Enhancements to the standard JTAG functionality include a handshake mechanism for transferring data to and from a PNX15xx Series processor's MMIO registers, support for posting an interrupt, and resetting processor state.

The actual interpretation of the contents of the MMIO registers is determined by a software protocol used by the debug monitor running on the internal TM3260 CPU and the debug front-end running on a host machine.

The communication between a host computer and a target system via JTAG requires the following major components:

1. A Host computer with a serial or parallel interface.

   The host computer transfers data to and from the JTAG interface module, preferably in word-parallel fashion. Also needed is JTAG interface device driver software to access and modify the registers of the JTAG interface module.

2. A JTAG interface module (hardware) that asynchronously transfers data to and from the host computer.

   The interface module synchronously transfers data to and from the JTAG TAP on a PNX15xx Series processor, supplies the test clock TCK and other signals to the JTAG controller on PNX15xx Series. The interface module may be a PC plug-in board.

   This module may transfer data from and to the host computer in bit-serial or word-parallel fashion. It transfers data from and to the JTAG registers on the PNX15xx Series processor in bit-serial fashion in accordance with the IEEE 1149.1 Standard. The JTAG interface module connects to a 4 or 5-pin JTAG connector on the PNX15xx Series board which provides a path to the JTAG pins on the PNX15xx Series processor. It is the responsibility of the interface module to scan data in and out of the PNX15xx Series processor into its internal buffers and make them available to the host computer.

3. A JTAG controller on the PNX15xx Series processor which provides a bridge between the external JTAG TAP and the internal system.

   The controller transfers data from/to the TAP to/from its scannable registers asynchronous to the internal system clock. A monitor running on the internal TM3260 CPU and the debugger front-end running on a host computer exchange data via JTAG by reading/writing the MMIO registers reserved for this purpose, including two control registers used for handshaking.

4. Register Descriptions

The PNX15xx Series has two JTAG data registers and two JTAG control registers (see Figure 3) in MMIO space and a number of JTAG instructions to manipulate those registers. Table 4 lists the MMIO addresses of the JTAG data and control registers. The addresses are offsets from the MMIO_BASE.

Remark: The sleepless bit is not used in PNX15xx Series.

Remark: All references to instruction and data registers refer to JTAG instructions and data registers only (not TM3260 instruction or data registers).

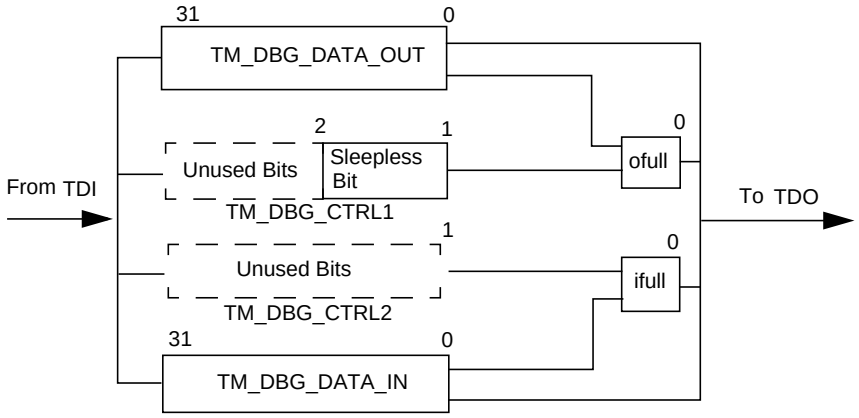


Figure 3: Additional JTAG Data and Control Registers

Data Registers

There are two 32-bit data registers, TM_DBG_DATA_IN and TM_DBG_DATA_OUT in the MMIO space. Both can be connected in between TDI and TDO like the standard Bypass and Boundary-Scan registers of JTAG.

- The TM_DBG_DATA_IN register can be read or written to via the JTAG port.
- The TM_DBG_DATA_OUT register is read-only via the JTAG port, so that scanning out TM_DBG_DATA_OUT is non-destructive.
- The TM_DBG_DATA_IN and TM_DBG_DATA_OUT are readable/writable from the TM3260 processor via the usual load/store operations.

Control Registers

There are two control registers, TM_DBG_CTRL1 and TM_DBG_CTRL2, in MMIO space.

- The TM_DBG_CTRL registers are used for handshake between a debug monitor running on a TM3260 CPU and a debugger front-end running on a host.
- TM_DBG_CTRL1.ofull = 1 means that TM_DBG_DATA_OUT has valid data to be scanned out. On the power-on reset of the PNX15xx Series, TM_DBG_CTRL1.ofull = 0. TM_DBG_CTRL1.ofull is both readable and writable via the JTAG tap. Writing 0 to TM_DBG_CTRL1.ofull via JTAG is a “remember” operation i.e., TM_DBG_CTRL1.ofull retains its previous state. Writing 1 to TM_DBG_CTRL1.ofull via JTAG is a ‘clear’ operation i.e., TM_DBG_CTRL1.ofull becomes 0.
- TM_DBG_CTRL2.ifull = 0 means that the TM_DBG_DATA_IN register is empty. TM_DBG_CTRL2.ifull = 1 means that TM_DBG_DATA_IN has valid data and the debug monitor has not yet copied it to its private area. Upon power on reset of the TM3260 processor, TM_DBG_CTRL2.ifull = 0. TM_DBG_CTRL2.ifull is readable

and writable via JTAG. Writing 0 to TM_DBG_CTRL2.ifull via JTAG is a ‘remember’ operation i.e., TM_DBG_CTRL2.ifull retains it previous state. Writing 1 to TM_DBG_CTRL2.ifull posts a TM3260 interrupt on hardware line 49.

- The TM_DBG_CTRL1.sleepless bit has no longer a function. However it still can be read and written to by the PNX15xx Series CPU via load/store operations and by the debugger front-end running on a host by scan in/out.

JTAG Virtual Registers

There are two virtual registers, TM_DBG_IFULL_IN and TM_DBG_OFULL_OUT:

- The first virtual register TM_DBG_IFULL_IN connects the registers TM_DBG_CTRL2.ifull and TM_DBG_DATA_IN in series. Likewise, the virtual register TM_DBG_OFULL_OUT connects TM_DBG_CTRL1.ofull and TM_DBG_DATA_OUT in series.
- The reason for the virtual registers is to shorten the time for scanning the TM_DBG_DATA_IN and TM_DBG_DATA_OUT registers. Without virtual registers, one must scan in an instruction to select TM_DBG_DATA_IN, scan in data, scan an instruction to select TM_DBG_CTRL register and finally scan in the control register. With virtual register, one can scan in an instruction to select TM_DBG_IFULL_IN and then scan in both control and data bits. Similar savings can be achieved for scan out using virtual registers.

4.1 Register Summary

Table 4: Register Summary

Offset	Symbol	Description
0x06 1000	TM_DBG_N_DATA_IN	Input register for data coming from the JTAG
0x06 1004	TM_DBG_N_DATA_OUT	Output register for data going to the JTAG
0x06 1008	TM_DBG_N_CTRL1	Control register 1 for output data
0x06 100C	TM_DBG_N_CTRL2	Control register 2 for input data
0x06 1FE0	TM_DBG_N_INT_ST	Interrupt Status Register
0x06 1FE4	TM_DBG_N_INT_EN	Interrupt Enable Register
0x06 1FE8	TM_DBG_N_INT_CLR	Interrupt Clear Register
0x06 1FEC	TM_DBG_N_INT_SET	Interrupt Set Register
0x06 1FF0	Reserved	
0x06 1FF4	TM_DBG_N_POWER_DOWN	Powerdown Register
0x06 1FF8	Reserved	
0x06 1FFC	Module ID	Module Identification Register

Table 5: TM_DBG 1 Registers

Bit	Symbol	Access	Value	Description
TM3260 Debug Registers				
Offset 0x06 1000	***TM_DBG_DATA_IN***			
31:0	TM_DBG_DATA_IN[31:0]	R/W	0	TM_DBG debugger input data
Offset 0x06 1004	***TM_DBG_DATA_OUT***			
31:0	TM_DBG_DATA_OUT[31:0]	R/W	0	TM_DBG debugger output data
Offset 0x06 1008	***TM_DBG_CTRL1***			
31:2	Unused		-	
1	sleepless	R/W	0	Set bit to prevent TM_DBG debug module from going into powerdown.
0	ofull	R/W	0	TM_DBG output data available handshake bit
Offset 0x06 100C	***TM_DBG_CTRL2***			
31:1	Unused		-	
0	ifull	R/W	0	TM_DBG input data available handshake bit
Offset 0x06 1FE0	***TM_DBG_INT_ST***			
31:1	Unused		-	
0	INTR_STATUS	R	0	Interrupt Status register: A logic “1” indicates JTAG interrupt detected.
Offset 0x06 1FE4	***TM_DBG_INT_EN***			
31:1	Unused		-	
0	INTR_EN	R/W	0	Interrupt Enable register: A logic “1” written to this bit enables the corresponding interrupt in the Interrupt Status register.
Offset 0x06 1FE8	***TM_DBG_INT_CLR***			
31:1	Unused		-	
0	INTR_CLR	W	0	Interrupt Clear register: A logic “1” written to this bit clears the corresponding interrupt in the Interrupt Status register. This bit is self-clearing.
Offset 0x06 1FEC	***TM_DBG_INT_SET***			
31:1	Unused		-	
0	INTR_SET	W	0	Interrupt Set register: A logic “1” written to this bit sets the corresponding interrupt in the Interrupt Status register.
Offset 0x06 1FF4	***TM_DBG_POWER_DOWN***			
31	POWER_DOWN	R/W	0	TM_DBG Powerdown indicator 1 = Powerdown 0 = Power up When this bit equals 1, no other registers are accessible.
30:0	Unused		-	
Offset 0x06 1FFC	***MODULE ID***			
31:16	MOD_ID	R	0x0127	TM_DBG Module ID Number

Table 5: TM_DBG 1 Registers *...Continued*

Bit	Symbol	Access	Value	Description
15:12	REV_MAJOR	R	0	Major revision
11:8	REV_MINOR	R	0	Minor revision
7:0	APP_SIZE	R	0	Aperture size is 0 = 4 KB.

Chapter 25: I²C Interface

PNX15xx Series Data Book – Volume 1 of 1

Rev. 2 — 1 December 2004 Product data sheet

1. Introduction

The PNX15xx Series chip contains an I²C module which interfaces with a variety of peripherals including consumer electronic appliances, video image processing equipment, and miniature cards. The IIC module supports bi-directional data transfer between master and slave in slow and fast speeds as well as multiple master and slave modes. Arbitration between simultaneously transmitting masters can be maintained without corruption of serial data on the bus. Serial clock synchronization allows devices with different bit rates to communicate via one serial bus, and it can be used as a handshake mechanism to suspend and resume serial transfer. The IIC hardware architecture and software protocol is simple and versatile.

The on-chip IIC module provides a serial interface that meets the I²C bus specification and supports all transfer modes to and from the I²C bus. It supports the following functionality:

- Both normal and fast modes up to 400 kHz.
- 32-bit word access from the CPU; no buffering is supported.
- Generation of interrupts on I²C state change
- Four modes of operation: master transmitter and receiver, slave transmitter and receiver.
- STO bit and the actual addresses of the Special Function Registers (SFRs).

The I²C bus is a multi-master bus. This means that more than one device capable of controlling the bus can be connected to it. Because more than one master could try to initiate a data transfer at the same time, a collision detection scheme is used to arbitrate between the multiple masters. If two or more masters attempt to transfer information onto the bus, the first to produce a ‘one’ when the other produces a ‘zero’ will detect the collision and back off transferring information.

The clock signals during arbitration are a synchronized combination of the clocks generated by the masters using the wired-AND connection to the SCL line. Two wires, SDA serial data) and SCL (serial clock), carry information between the devices connected to the I²C bus.

Each device can operate as either a transmitter or receiver, depending on the function of the device. In addition to transmitters and receivers, devices can also be considered as masters or slaves when performing data transfers. A master is the device which initiates a data transfer on the bus and generates the clock signals to permit that transfer. Any device addressed by a master is considered a slave.

Generation of clock signals on the I2C bus is always the responsibility of the master device. Each master generates its own clock signals when transferring data on the bus. Bus clock signals from a master can only be altered when they are stretched by a slow-slave device holding down the clock line or by another master when arbitration occurs.

1.1 Features

The main features of the I2C bus are as follows:

- Bi-directional data transfer between masters and slaves
- Multi-master bus (no central master)
- Arbitration between simultaneously transmitting masters without corruption of serial data on the bus
- Serial clock synchronization, which allows communication between devices with different bit rates
- Using serial clock synchronization as a handshake mechanism to suspend and resume serial transfer
- May be used for test and diagnostic purposes.

2. Functional Description

2.1 General Operations

The IIC module supports a master/slave I2C bus interface with an integrated shift register, shift timing generation and slave address recognition. It is compliant with the I2C Bus Specification. Both the I2C standard mode (100 kHz SCL) and fast mode (up to 400 kHz SCL) are supported.

2.1.1 IIC Arbitration and Control Logic

In the master transmitter mode, the arbitration logic checks that every transmitted logic ‘1’ actually appears as a logic 1 on the I2C bus. If another device on the bus overrules a logic ‘1’ and pulls the SDA line low, arbitration is lost and the IIC module immediately changes from master transmitter to slave receiver. The IIC module will continue to output clock pulses (on SCL) until transmission of the current serial byte is complete. Arbitration may also be lost in the master receiver mode. Loss of arbitration in this mode can only occur while the IIC module is returning a “not acknowledge” (logic ‘1’) to the bus. Arbitration is lost when another device on the bus pulls this signal LOW. Since this can occur only at the end of a serial byte, the IIC module generates no further clock pulses.

The synchronization logic will synchronize the serial clock generator with the clock pulses on the SCL line from another device. If two or more master devices generate clock pulses, the “mark” (high level) duration is determined by the device that generates the shortest marks, and the “space” (low level) duration is determined by the device that generates the longest spaces.

A slave may stretch the space duration to slow down the bus master. The space duration may also be stretched for handshaking purposes. This can be done after each bit or after a complete byte transfer. The IIC module will stretch the SCL space duration after a byte has been transmitted or received and the acknowledge bit has been transferred. This block also controls all of the signals for serial byte handling. It provides the shift pulses for DAT, enables the comparator, generates and detects START and STOP conditions, receives and transmits acknowledge bits, controls the master and slave modes, contains interrupt request logic and monitors the I^2C bus status.

2.1.2 Serial Clock Generator

This programmable clock pulse generator provides the SCL clock pulses when the IIC module is in master transmitter or master receiver mode. It is switched off when the IIC module is in a slave mode. The output frequency is dependent on the CR bits in the control register. The output clock pulses have a 50% duty cycle unless the clock generator is synchronized with other SCL clock sources, as described above.

2.1.3 Bit Counter

The bit counter tracks the number of bits that have been received during the byte transfer. The output from this counter is used to trigger events, such as address recognition and acknowledge generation, which occur at specific points during the byte transfer.

2.1.4 Control Register

This register is used by the micro controller to control the generation of START and STOP conditions, enable the interface, control the generation of ACKs, and to select the clock frequency.

2.1.5 Status Decoder and Register

There are 26 possible bus states if all four modes of the IIC module are used. The status decoder takes all of the internal status bits and compresses them into a 5-bit code. This code is unique for each I^2C bus status. The 5-bit code may be used for processing the various service routines. Each service routine processes a particular bus status. The 5-bit status code is stored in bits 7-3 of the status register. Bits 2-0 and 31-8 of the status register are always zero.

2.1.6 Input Filter

Input signals SDA and SCL from IO pad cells are synchronized with the internal clock. Spikes shorter than three clock periods are filtered out.

2.1.7 Address Register and Comparator

This SFR may be loaded with the 7-bit slave address to which IIC module will respond when programmed as a slave. The least significant bit is used to enable the general call address recognition. The comparator compares the received 7-bit slave address with its own slave address. It also compares the first received 8-bit byte with the general call address. If an equality is found, the appropriate status bits are set and an interrupt is requested.

2.1.8 Data Shift Register

This register contains a byte of serial data to be transmitted or a byte which has just been received. Like all the registers in this module, only bits 7-0 are used. Data in DAT is always shifted from right to left; the first bit to be transmitted is the MSB (bit 7) and, after a byte has been received, the first bit of received data is located at the MSB (bit 7) of DAT. While data is being shifted out, data on the bus is simultaneously being shifted in; DAT always contains the last byte present on the bus. Thus, in the event of lost arbitration, the transition from master transmitter to slave receiver is made with the correct data in DAT.

2.1.9 Related Interrupts

The serial interrupt signal (iic_intrn) issues an interrupt when any one of the 26 possible IIC module states are entered. The only state that never causes an interrupt is state 0xF8, which indicates that no relevant state information is available.

2.1.10 Modes of Operation

The IIC module hardware may operate in any of the following four modes:

- Master Transmitter
- Master Receiver
- Slave Receiver
- Slave Transmitter

As a master, the IIC module will generate all the serial clock pulses and the START and STOP conditions. A transfer ends with a STOP condition or with a repeated START condition. Since a repeated START condition is also the beginning of the next serial transfer, the I^2C bus will not be released.

Two types of data transfers are possible on the I^2C bus:

- Data transfer from a master transmitter to a slave receiver. The first byte transmitted by the master is the slave address. Next follows a number of data bytes. The slave returns an acknowledge bit after each received byte.
- Data transfer from a slave transmitter to a master receiver. The first byte (the slave address) is transmitted by the master. The slave then returns an acknowledge bit. Next follows the data bytes transmitted by the slave to the master. The master returns an acknowledge bit after each received byte except the last byte. At the end of the last received byte, a “not acknowledge” is returned.

In a given application, the IIC module may operate as a master and as a slave. In the slave mode, the IIC module hardware looks for its own slave address and the general call address. If one of these addresses is detected, an interrupt is requested. When the micro controller wishes to become the bus master, the hardware waits until the bus is free before the master mode is entered so that a possible slave action is not interrupted. If bus arbitration is lost in the master mode, the IIC module switches to the slave mode immediately and can detect its own slave address in the same serial transfer.

Master Transmitter Mode

Serial data is output through SDA while SCL outputs the serial clock. The first byte transmitted contains the slave address of the receiving device (7-bit SLA) and the data direction bit as in Figure 1. In this case the data direction bit (R/W) will be a logic ‘0’ (W). Serial data is transmitted 8 bits at a time. After each byte is transmitted, an acknowledge bit is received. START and STOP conditions are output to indicate the beginning and end of a serial transfer.

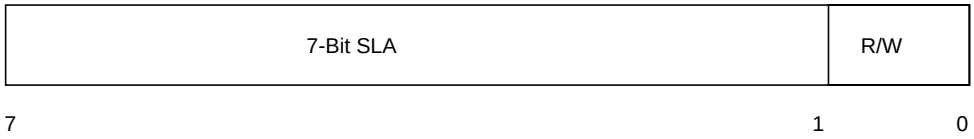


Figure 1: SDA First Transmitted Byte

In the master transmitter mode, a number of data bytes can be transmitted to the slave receiver. Before the master transmitter mode can be entered, the IIC_CONTROL register must be initialized with the EN bit set and the STA and STO bits reset. EN must be set to enable the IIC module interface. If the AA bit is reset, the IIC module will not acknowledge its own slave address or the general call address if they are present on the bus. This will prevent the IIC module interface from entering a slave mode.

The master transmitter mode may now be entered by setting the STA bit. The IIC module will then test the I^2C bus and generate a start condition as soon as the bus becomes free. When a START condition is transmitted, the status code in the status register (STA) will be 0x08. This status code must be used to vector to an interrupt service routine that loads IIC_DAT with the slave address and the data direction bit (SLA+W).

When the slave address and direction bit have been transmitted and an acknowledgment bit has been received, a number of status codes in STA are possible. The appropriate action to be taken for any of the status codes is detailed in Table 5 on page 25-11. After a repeated start condition (state 0x10), the IIC module may switch to the master receiver mode by loading IIC_DAT with SLA+R.

Master Receiver Mode

The first byte transmitted contains the slave address of the transmitting device (7-bit SLA) and the data direction bit. In this case, the data direction bit (R/W) will be logic 1 (R). Serial data is received via SDA while SCL outputs the serial clock. Serial data is received 8 bits at a time. After each byte is received, an acknowledge bit is transmitted. START and STOP conditions are output to indicate the beginning and end of a serial transfer.

In the master receiver mode, a number of data bytes are received from a slave transmitter. The transfer is initialized as in the master transmitter mode. When the START condition has been transmitted, the interrupt service routine must load IIC_DAT with the 7-bit slave address and the data direction bit (SLA+R).

When the slave address and data direction bit have been transmitted and an acknowledgment bit has been received, a number of status codes are possible in STA. The appropriate action to be taken for each of the status codes is detailed in Table 5. After a repeated start condition (state 0x10), the IIC module may switch to the master transmitter mode by loading IIC_DAT with SLA+W.

Slave Receiver Mode

Serial data and the serial clock are received through SDA and SCL. After each byte is received, an acknowledge bit is transmitted. START and STOP conditions are recognized as the beginning and end of a serial transfer. Address recognition is performed by hardware after reception of the slave address and direction bit.

In the slave receiver mode, a number of data bytes are received from a master transmitter. To initiate the slave receiver mode, IIC_ADDRESS must be loaded with the 7-bit slave address to which the IIC module will respond when addressed by a master. Also the least significant bit of IIC_ADDRESS should be set if the interface should respond to the general call address (0x00). The IIC_CONTROL register, should be initialized with EN and AA set and STA and STO reset in order to enter the slave receiver mode. Setting the AA bit will enable the logic to acknowledge its own slave address or the general call address and EN will enable the interface.

When IIC_ADDRESS and IIC_CONTROL have been initialized, the IIC module waits until it is addressed by its own slave address, followed by the data direction bit which must be ‘0’ (W) for the IIC module to operate in the slave receiver mode. After its own slave address and the W bit have been received, a valid status code can be read from IIC_DAT. This status code should be used to vector to an interrupt service routine. The appropriate action to be taken for each of the status codes is detailed in Table 5. The slave receiver mode may also be entered if arbitration is lost while the IIC module is in the master mode.

If the AA bit is reset during a transfer, the IIC module will return a not acknowledge (logic ‘1’) to SDA after the next received data byte. While AA is reset, the IIC module does not respond to its own slave address or a general call address. However, the I^2C bus is still monitored and address recognition may be resumed at any time by setting AA. This means that the AA bit may be used to isolate the IIC module from the I^2C bus temporarily.

Slave Transmitter Mode

The first byte is received and handled as in the slave receiver mode. However, in this mode, the direction bit will indicate that the transfer direction is reversed. Serial data is transmitted via SDA while the serial clock is input through SCL. START and STOP conditions are recognized as the beginning and end of a serial transfer.

In the slave transmitter mode, a number of data bytes are transmitted to a master receiver. Data transfer is initialized as in the slave receiver mode. When IIC_ADDRESS and IIC_CONTROL have been initialized, the IIC module waits until it is addressed by its own slave address followed by the data direction bit, which must be ‘1’ (R) for the IIC module to operate in the slave transmitter mode. After its own slave address and the R bit have been received, a valid status code can be read from STA. This status code is used to vector to an interrupt service routine. The

appropriate action to be taken for each of these status codes is detailed in the Table 5. The slave transmitter mode may also be entered if arbitration is lost while the IIC module is in the master mode.

If the AA bit is reset during a transfer, the IIC module will transmit the last byte of the transfer and enter state 0xC0 or 0xC8. The IIC module is switched to the “not addressed” slave mode and will ignore the master receiver if it continues the transfer. Thus the master receiver receives all 1s as serial data. While AA is reset, the IIC module does not respond to its own slave address or a general call address. However, the I^2C bus is still monitored and address recognition may be resumed at any time by setting AA. This means that the AA bit may be used to temporarily isolate the IIC module from the I^2C bus.

3. Register Descriptions

Table 1: Register Summary

Offset	Symbol	Description
0x04 5000	I2C_CONTROL	Controls the operation mode of the IIC module
0x04 5004	I2C_DAT	Byte of data to be transmitted or received
0x04 5008	I2C_STATUS	Indicates the status of the IIC module
0x04 500C	I2C_ADDRESS	Slave address of the IIC module
0x04 5010	I2C_STOP	Set STO flag
0x04 5014	I2C_PD	Powerdown, reset IIC sampling clock registers except for MMIO registers
0x04 5018	I2C_SET_PINS	Set I^2C bus SDA and/or SCL signals low
0x04 501C	I2C_OBS_PINS	Observe I^2C bus SDA and SCL signals
0x04 5020—9FDC	Reserved	
0x04 5FE0	I2C_INT_STATUS	Interrupt Status register
0x04 5FE4	I2C_INT_EN	Interrupt Enable register
0x04 5FE8	I2C_INT_CLR	Interrupt Clear register
0x04 5FEC	I2C_INT_SET	Interrupt software set register
0x04 5FF0	Reserved	
0x04 5FF4	I2C Powerdown	Powerdown mode, switch module clock off
0x04 5FF8	Reserved	
0x04 5FFC	Module ID	Module Identification and revision information

3.1 Register Tables

Table 2: IIC Registers

Bit	Symbol	Access	Value	Description
Offset 0x04 5000	*I2C CONTROL*			
31:8	Unused		-	Ignore upon read. Write as zeroes.
7	AA	R/W	0	IIC acknowledge bit 0 = Acknowledge not returned during acknowledge clock pulse 1 = Acknowledge returned during acknowledge clock pulse
6	EN	R/W	0	IIC enable bit 0 = Disable IIC module 1 = Enable IIC module
5	STA	R/W	0	IIC start bit 0 = Slave mode, accept transactions 1 = Master mode, generate start condition if bus is free
4	STO	R	0	IIC stop bit 0 = Slave mode, accept transactions 1 = Generate stop condition on I2C bus when IIC module is master.
3	Unused		-	Ignore upon read. Write as zeroes.
2:0	CR	R/W	100	These three bits determine the serial clock frequency when IIC module is in master mode. This field shall be changed only when EN bit is 0. The IIC Clock is divided as follows to achieve the desired frequency. The table assumes the IIC module receives a 24 MHz clock from the Clock module. 0: 60 -> 400 KHz 1: 80 -> 300 KHz 2: 120 -> 200 KHz 3: 160 -> 150 KHz 4: 240 -> 100 KHz 5: 320 -> 75 KHz 6: 480 -> 50 KHz 7: 960 -> 25 KHz

Bit 7: AA Address Acknowledge

If the AA flag is set, an acknowledge (low level to SDA) will be returned during the acknowledge clock pulse on the SCL line when:

- The “own slave address” has been received.
- The general call address has been received while the general call bit (GC) in the ADR register is set.
- A data byte has been received while IIC module is in the master receiver mode.
- A data byte has been received while IIC module is in the addressed slave receiver mode.

If the AA flag is cleared and the IIC-bus module is in the addressed slave transmitter mode, state 0xC8 will be entered after the last serial bit is transmitted. The IIC module leaves state 0xC8, enters the “not addressed” slave receiver mode, and the SDA line remains at a high level. In state 0xC8, the AA flag can be set again for future address recognition.

If the AA flag is cleared and the IIC module is in the “not addressed” slave mode, its own slave address and the general call address are ignored. Consequently, no acknowledge is returned, and a serial interrupt is not requested.

Thus, IIC module can be temporarily released from the I^2C bus while the bus status is monitored. While the IIC module is released from the bus, START and STOP conditions are detected, and serial data is shifted in.

Address recognition can be resumed at any time by setting the AA flag. If the AA flag is set when the part’s own slave address or the general call address has been partly received, the address will be recognized at the end of the byte transmission.

Bit 6: EN Enable

- When EN is ‘0’, the SDA and SCL outputs are constantly at 1 level leading to a high impedance state at the associated port lines SDA and SCL. The state of the SDA and SCL input lines are ignored; IIC module is in the “not addressed” slave state, and the STO bit in IIC_CONTROL is forced to ‘0’. No other bits are affected. SDA and SCL port lines may be used as open drain I/O ports.
- When EN is ‘1’, IIC module is enabled. The port latches associated to SDA and SCL must be set to logic 1.
- EN should not be used to temporarily release IIC module from the I^2C bus because when EN is reset, the I^2C bus status is lost. The AA flag should be used to temporarily release the IIC module.

Bit 5: STA Start bit

When the STA bit is set to enter a master mode, the IIC module hardware checks the status of the I^2C bus and generates a START condition if the bus is free. If it is not free, the IIC module waits for a STOP condition (which will free the bus) and generates a START condition after a delay of a low clock period in the internal serial clock generator.

If STA is set while IIC module is already in master mode and one or more bytes are transmitted or received, the IIC module transmits a repeated START condition. STA may be set at any time. It may also be set when the IIC module is an addressed slave.

When the STA bit is reset, no START condition or repeated START condition will be generated.

Bit 4: STO Stop bit

A write to IIC_CONTROL Register will not affect this bit. It must be written via IIC_STOP Register. When the STO bit is set while IIC module is in a master mode, a STOP condition is transmitted to the I^2C bus.

When the STOP condition is detected on the bus, the IIC module hardware clears the STO flag. In a slave mode, the STO flag may be set to recover from an error condition. In this case, no STOP condition is transmitted to the I^2C bus. However, the IIC module hardware behaves as if a STOP condition has been received and switches to the defined “not addressed” slave receiver mode. The STO flag is automatically cleared by the hardware.

If the STA and STO bits are both set, the STOP condition is transmitted to the I^2C bus if IIC module is in master mode. In slave mode, the IIC module generates an internal STOP condition, which is not transmitted. It then transmits a START condition.

When the STO bit is reset, no STOP condition will be generated.

There is also no interrupt generated for detection of a STOP condition which was created by the IIC. (The IIC needs to be in master mode to do this). And there is no status code for this condition in the IIC STATUS register.

Bits [2:0]: CR

For details, see Bits[2:0] in Offset 0x04 5000 I2C CONTROL of the register table.

Table 3: IIC Registers

Bit	Symbol	Access	Value	Description
Offset 0x04 5004	***I2C DATA REGISTER***			
31:8	Unused		-	Ignore upon read. Write as zeroes.
7:0	DAT	R/W	0	Write byte data to be transmitted on the I2C bus. Read byte data has been received from the I2C bus.

Bit [7:0]: DAT

DAT contains a byte of serial data to be transmitted or a byte which has just been received. This special function register can be read from or written to while it is not in the process of shifting a byte. This occurs when IIC module is in a defined state. Data in DAT is always shifted from right to left: the first bit to be transmitted is the MSB (bit 7), and after a byte has been received, the first bit of received data is located at the MSB of DAT. While data is being shifted out, data on the bus is simultaneously being shifted in; DAT always contains the last data byte present on the bus. Thus, in the event of lost arbitration, the transition from master transmitter to slave receiver is made with the correct data in DAT. Reset initializes DAT to 0x00.

A logic ‘1’ in DAT corresponds to a high level on the I^2C bus, and a logic ‘0’ corresponds to a low level on the bus. Serial data shifts through DAT from right to left.

DAT and the ACK flag form a 9-bit shift register which shifts in or shifts out 8 bits, followed by an acknowledge bit. The ACK flag is controlled by the IIC module hardware and cannot be accessed by the CPU. Serial data is shifted through the ACK flag into DAT on the rising edges of clock pulses on the SCL line. When a byte has been shifted into DAT, the serial data is available in DAT, and the acknowledge bit is returned by the control logic during the 9th clock pulse. Serial data is shifted out from DAT via a buffer on the falling edges of clock pulses on the SCL line.

When the CPU writes to DAT, the buffer is loaded with the contents of DAT(7) which is the first bit to be transmitted to the SDA line. After nine serial clock pulses, the eight bits in DAT will have been transmitted to the SDA line, and the acknowledge bit will be present in ACK. Note that the eight transmitted bits are shifted back into DAT.

Table 4: IIC Registers

Bit	Symbol	Access	Value	Description
Offset 0x04 5008	*I2C STATUS REGISTER*			
31:8	Unused		-	Ignore upon read. Write as zeroes.
7:3	STA	R	0	Indicates the status of the IIC module.
2:0	Unused		-	Ignore upon read. Write as zeroes.

Bit [7:3]: STA Status register

STA is a read-only special function register. Writing to this register has no affect. The three least significant bits are always zero. The bits 7-3 contain the status code. There are 26 possible status codes. When STA contains 0xF8, no relevant state information is available and no serial interrupt is requested. Reset initializes STA to 0xF8. All other STA values correspond to defined IIC module states.

See the last row of Table 5 for an explanation of the terms used.

Table 5: Status Codes

Status Code STA	Bus/Module Status	Application Software Response					
			To CON				
		To/From DAT	STA	STO	SI	AA	Next Action Taken
0x00	Bus Error	No DAT Action	0	1	0	X	HW will enter the “not addressed” slave mode.
0x08	A Start condition has been transmitted	Load SLA+W	X	0	0	X	SLA+W will be transmitted, ACK bit will be received (I2C bus module will be switched to MST/TRX mode).
		Load SLA+R	X	0	0	X	SLA+R will be transmitted, ACK bit will be received (I2C bus module will be switched to MST/REC mode).
0x10	A repeated Start condition has been transmitted	Load SLA+W	X	0	0	X	SLA+W will be transmitted, ACK bit will be received (I2C bus module will be switched to MST/TRX mode).
		Load SLA+R	X	0	0	X	SLA+R will be transmitted, ACK bit will be received (I2C bus module will be switched to MST/REC mode).
0x18	SLA+W has been transmitted; ACK has been received	Load data byte	0	0	0	X	Data byte will be transmitted; ACK bit will be received.
		no DAT action	1	0	0	X	Repeat START will be transmitted
		no DAT action	0	1	0	X	STOP condition will be transmitted; STOP flag will be reset.
		no DAT action	1	1	0	X	STOP condition followed by a START condition will be transmitted; STO flag will be reset.

Table 5: Status Codes *...Continued*

Status Code STA	Bus/Module Status	Application Software Response					
			To CON				
		To/From DAT	STA	STO	SI	AA	Next Action Taken
0x20	SLA+W has been transmitted; NOT ACK has been received	Load data byte	0	0	0	X	Data byte will be transmitted; ACK bit will be received.
		no DAT action	1	0	0	X	Repeat START will be transmitted.
		no DAT action	0	1	0	X	STOP condition will be transmitted; STOP flag will be reset.
		no DAT action	1	1	0	X	STOP condition followed by a START condition will be transmitted; STO flag will be reset.
0x28	Data byte in DAT has been transmitted; ACK has been received	Load data byte	0	0	0	X	Data byte will be transmitted; ACK bit will be received
		no DAT action	1	0	0	X	Repeat START will be transmitted.
		no DAT action	0	1	0	X	STOP condition will be transmitted; STOP flag will be reset.
		no DAT action	1	1	0	X	STOP condition followed by a START condition will be transmitted; STO flag will be reset.
0x30	Data byte in DAT has been transmitted; NOT ACK has been received	Load data byte	0	0	0	X	Data byte will be transmitted; ACK bit will be received.
		no DAT action	1	0	0	X	Repeat START will be transmitted.
		no DAT action	0	1	0	X	STOP condition will be transmitted; STOP flag will be reset.
		no DAT action	1	1	0	X	STOP condition followed by a START condition will be transmitted; STO flag will be reset.
0x38	Arbitration lost in the SLA+R/W or Data bytes	No DAT action	0	0	0	X	I2C bus will be released; the “not addressed” slave mode will be entered.
		No DAT action	1	0	0	X	A START condition will be transmitted when the bus becomes free.
0x40	SLA+R has been transmitted; ACK has been received	No DAT action	0	0	0	0	Data byte will be received; NOT ACK bit will be returned.
		No DAT action	0	0	0	1	Data byte will be received; ACK bit will be returned
0x48	SLA+R has been transmitted; NOT ACK has been received	No DAT action	1	0	0	X	Repeated START condition will be transmitted
		No DAT action	0	1	0	X	STOP condition will be transmitted; STO flag will be reset.
		No DAT action	1	1	0	X	STOP condition followed by a START condition will be transmitted; STO flag will be reset.
0x50	Data byte has been received; ACK has been returned	Read data byte	0	0	0	0	Data byte will be received; NOT ACK bit will also been returned.
		Read data byte	0	0	0	1	Data byte will be received; ACK bit will be returned.

Table 5: Status Codes *...Continued*

Status Code STA	Bus/Module Status	Application Software Response					
			To CON				
		To/From DAT	STA	STO	SI	AA	Next Action Taken
0x58	Data byte has been received; NOT ACK has been returned	Read data byte	1	0	0	X	Repeated START condition will be transmitted.
		Read data byte	0	1	0	X	STOP condition will be transmitted; STO flag will be reset.
		Read data byte	1	1	0	X	STOP condition followed by a START condition will be transmitted; STO bit will be reset.
0x60	Own SLA+W has been received; ACK has been returned	No STA action	X	0	0	0	Data byte will be received and NOT ACK will be returned.
		No STA action	X	0	0	1	Data byte will be received and ACK will be returned.
0x68	Arbitration lost in SLA+R/W as master; Own SLA+W has been received; ACK has been returned	No STA action	X	0	0	0	Data byte will be received and NOT ACK will be returned.
		No STA action	X	0	0	1	Data byte will be received and ACK will be returned.
0x70	General call address (00H) has been received; ACK has been returned	No STA action	X	0	0	0	Data byte will be received and NOT ACK will be returned.
		No STA action	X	0	0	1	Data byte will be received and ACK will be returned.
0x78	Arbitration lost in SLA+R/W as master; General call address has been received; ACK has been returned	No STA action	X	0	0	0	Data byte will be received and NOT ACK will be returned.
		No STA action	X	0	0	1	Data byte will be received and ACK will be returned.
0x80	Previously addressed with own SLA; data byte has been received; ACK has been returned	No STA action	X	0	0	0	Data byte will be received and NOT ACK will be returned.
		No STA action	X	0	0	1	Data byte will be received and ACK will be returned.

Table 5: Status Codes *…Continued*

Status Code STA	Bus/Module Status	Application Software Response					
			To CON				
		To/From DAT	STA	STO	SI	AA	Next Action Taken
0x88	Previously addressed with own SLA; data byte has been received; NOT ACK has been returned	Read data byte	0	0	0	0	Switched to “not addressed” SLV mode; no recognition of own SLA or general call address.
		Read data byte	0	0	0	1	Switched to “not addressed” SLV mode; own SLA will be recognized; general call address will be recognized if ADR(0) =’1’.
		Read data byte	1	0	0	0	Switched to “not addressed” SLV mode; no recognition of own SLA or general call address. A START condition will be transmitted when bus becomes free.
		Read data byte	1	0	0	1	Switched to “not addressed” SLV mode; own SLA will be recognized; general call address will be recognized if ADR(0) =’1’; A START condition will be transmitted when the bus becomes free.
0x90	Previously addressed with general call; data byte has been received; ACK has been returned	Read data byte	X	0	0	0	Data byte will be received and NOT ACK will be returned.
		Read data byte	X	0	0	1	Data byte will be received and ACK will be returned.
0x98	Previously addressed with general call; data byte has been received; NOT ACK has been returned	Read data byte	0	0	0	0	Switched to “not addressed” SLV mode; no recognition of own SLA or general call address.
		Read data byte	0	0	0	1	Switched to “not addressed” SLV mode; Own SLA will be recognized; general call address will be recognized if ADR(0) =’1’.
		Read data byte	1	0	0	0	Switched to “not addressed” SLV mode; no recognition of own SLA or general call address. A START condition will be transmitted when the bus becomes free.
		Read data byte	1	0	0	1	Switched to “not addressed” SLV mode; own SLA will be recognized; general call address will be recognized if ADR(0) =’1’; A START condition will be transmitted when the bus becomes free.

Table 5: Status Codes *...Continued*

Status Code STA	Bus/Module Status	Application Software Response					
			To CON				
		To/From DAT	STA	STO	SI	AA	Next Action Taken
0xA0	A STOP condition or repeated START condition has been received while still addressed as SLV/ (REC or TRX) (But for SLV/TRX a violation of I2C bus format)	No STA action	0	0	0	0	Switched to “not addressed” SLV mode; no recognition of own SLA or general call address.
		No STA action	0	0	0	1	Switched to “not addressed” SLV mode; Own SLA will be recognized; general call address will be recognized if ADR(0) =’1’.
		No STA action	1	0	0	0	Switched to “not addressed” SLV mode; no recognition of own SLA or general call address. A START condition will be transmitted when the bus becomes free.
		No STA action	1	0	0	1	Switched to “not addressed” SLV mode; own SLA will be recognized; general call address will be recognized if ADR(0) =’1’. A START condition will be transmitted when the bus becomes free.
0xA8	Own SLA+R has been received; ACK has been returned	Load data byte	X	0	0	0	Last data byte will be transmitted and ACK bit will be received.
		Load data byte	X	0	0	1	Data byte will be transmitted; ACK bit will be received.
0xB0	Arbitration lost in SLA+R/W as master; Own SLA+R has been received; ACK has been received	Load data byte	X	0	0	0	Last data byte will be transmitted and ACK bit will be received.
		Load data byte	X	0	0	1	Data byte will be transmitted; ACK bit will be received.
0xB8	Data byte in DAT has been transmitted; ACK has been received	Load data byte	X	0	0	0	Last data byte will be transmitted and ACK bit will be received.
		Load data byte	X	0	0	1	Data byte will be transmitted; ACK bit will be received.
0xC0	Data byte in DAT has been transmitted; NOT ACK has been received (AA = X)	No STA action	0	0	0	0	Switched to “not addressed” SLV mode; no recognition of own SLA or general call address.
		No STA action	0	0	0	1	Switched to “not addressed” SLV mode; own SLA will be recognized; general call address will be recognized if ADR(0) =’1’.
		No STA action	1	0	0	0	Switched to “not addressed” SLV mode; no recognition of own SLA or general call address. A START condition will be transmitted when the bus becomes free.
		No STA action	1	0	0	1	Switched to “not addressed” SLV mode; own SLA will be recognized; general call address will be recognized if ADR(0) =’1’. A START condition will be transmitted when the bus becomes free.

Table 5: Status Codes *...Continued*

Status Code STA	Bus/Module Status	Application Software Response					Next Action Taken
		To/From DAT	To CON STA	STO	SI	AA	
0xC8	Last data byte in DAT has been transmitted (AA = 0); ACK has been received	No STA action	0	0	0	0	Switched to “not addressed” SLV mode; no recognition of own SLA or general call address.
		No STA action	0	0	0	1	Switched to “not addressed” SLV mode; own SLA will be recognized; general call address will be recognized if ADR(0) =’1’.
		No STA action	1	0	0	0	Switched to “not addressed” SLV mode; no recognition of own SLA or general call address. A START condition will be transmitted when the bus becomes free.
		No STA action	1	0	0	1	Switched to “not addressed” SLV mode; Own SLA will be recognized; general call address will be recognized if ADR(0) =’1’. A START condition will be transmitted when the bus becomes free.
0xF8	No information available	No DAT action	No IIC_CONTROL action				Wait or proceed with current transfer.
MST—Master SLV—Slave REC—Receiver TRX—Transmitter			SLA—Slave address W—Write bit R—Read bit X—Don’t care bit				

Table 6: IIC Registers

Bit	Symbol	Access	Value	Description
Offset 0x04 500C	*I2C ADDRESS REGISTER*			
31:8	Unused		-	Ignore upon read. Write as zeroes.
7:1	SLAVE_ADDR	R/W	0x00	Slave address when in slave mode
0	GEN_CALL_ADDR	R/W	0x0	General call address 0 = Does not generate interrupt if general call address is detected on the I2C bus. 1= Generates interrupt if general call address is detected.

Bit [7:1]: SLAVE_ADDR Slave Address

SLAVE_ADDR is not affected by the IIC module hardware. The contents of this register are irrelevant when IIC module is in a master mode. In the slave mode, the bits 7:1 must be loaded with the micro controller’s own slave address. These bits correspond to the 7-bit slave address which will be recognized on the incoming data stream from the I^2C bus. When the slave address is detected and the interface is enabled, a serial interrupt will be generated.

Bit 0: GEN_CALL_ADDR General Call Address

When this bit is set, the general call address (Slave Address[7:1] on I^2C bus = 0x00, R/W bit on I^2C bus = 0) is recognized. If not set, this bit is ignored.

Table 7: IIC Registers

Bit	Access	Value	Symbol	Description
Offset 0x04 5010			***I2C STOP REGISTER***	
31:1		-	Unused	Ignore upon read. Write as zeroes.
0	R/W	0	STO	Set and read STO flag Writes: In master mode, set STO flag to indicate a STOP has been requested. Then generate a STOP condition on I^2C bus. When the STOP condition is detected on the bus, the IIC module hardware clears the STO flag. In slave mode, set STO flag to indicate a STOP has been requested. No STOP condition is transmitted to the I^2C bus. However, the IIC module hardware behaves as if a STOP condition has been received and switches to the defined “not addressed” slave receiver mode. The STO flag is immediately cleared by the hardware so that software can never see it set. Reads: View the STO flag to see if a STOP has been requested. STO flag can also be viewed by reading the STO bit of IIC_CONTROL. See STO bit of IIC_CONTROL register for more information
Offset 0x04 5014			***I2C PD REGISTER***	
31:3		-	Unused	Ignore upon read. Write as zeroes.
2	R/W	0	PD	This bit synchronously resets the IIC clock domain except for the MMIO registers. 0 = Don’t reset IIC clock domain. 1 = Reset IIC clock domain. Note: Do not reset the IIC clock domain until the IIC module is disabled using bit 6 of the IIC Control register.
1:0		-	Unused	Ignore upon read. Write as zeroes.
Offset 0x04 5018			***I2C BUS SET REGISTER***	
31:2		-	Unused	Ignore upon read. Write as zeroes.
1	W	0	SET_SCL_LOW	Pull I2C SCL bus signal to logic one or zero: 1 = Pulls SCL signal to logic zero. 0 = SCL signal is not pulled to logic zero.
0	W	0	SET_SDA_LOW	Pull I2C SDA bus signal to logic one or zero: 1 = Pulls SDA signal to logic zero. 0 = SDA signal is not pulled to logic zero.
Offset 0x04 501C			***I2C BUS OBSERVATION REGISTER***	
31:2		-	Unused	Ignore upon read. Write as zeroes.
1	R	0	OBSERVE_SCL	Observe I2C SCL bus signal: 1 = SCL signal is at logic one. 0 = SCL signal is at logic zero.
0	R	0	OBSERVE_SDA	Observe I2C SDA bus signal 1 = SDA signal is at logic one. 0 = SDA signal is at logic zero.
Offset 0x04 5020—9FDC			***Reserved***	

Table 7: IIC Registers *...Continued*

Bit	Access	Value	Symbol	Description
Offset 0x04 5FE0			***I2C INTERRUPT STATUS REGISTER***	
31:1		-	Unused	Ignore upon read. Write as zeroes.
0	R	0	INT_STATUS	Interrupt status register. It reports any pending interrupts: 1 = Interrupt i pending. 0 = Interrupt i not pending.
Offset 0x04 5FE4			***I2C INTERRUPT ENABLE REGISTER***	
31:1		-	Unused	Ignore upon read. Write as zeroes.
0	R/W	0	INT_ENABLE	Interrupt enable register 1 = Interrupt i is enabled. 0 = Interrupt is disabled.
Offset 0x04 5FE8			***I2C INTERRUPT CLEAR REGISTER***	
31:1		-	Unused	Ignore upon read. Write as zeroes.
0	W	0	INT_CLEAR	Interrupt clear register. 1 = Interrupt is cleared. 0 = Interrupt is not cleared.
Note: The IIC module will look at a new transaction on the I2C bus as soon as the previous interrupt has been cleared. Therefore, software must make sure that “interrupt clear” is the last transaction that is sent to the IIC module before starting a new transaction.				
Offset 0x04 5FEC			***I2C INTERRUPT SET REGISTER***	
31:1		-	Unused	Ignore upon read. Write as zeroes.
0	W	0	INT_SET	Interrupt set register. Allows software to set interrupts. 1 = Interrupt is set 0 = Interrupt is not set.
Offset 0x04 5FF0			***Reserved***	
Offset 0x04 5FF4			***I2C POWERDOWN REGISTER***	
31	R/W	0	POWER_DOWN	0 = Normal operation of peripheral. This is the reset value. 1 = Module is powerdown and module clock can be removed. Module returns DEADABBA on all reads except for reads of the powerdown bit. Module generates ERR ACK on all writes except for writes to the powerdown bit.
30:0		-	Unused	Ignore upon read. Write as zeroes.
Offset 0x04 5FF8			***Reserved***	
Offset 0x04 5FFC			***I2C MODULE ID REGISTER***	
31:16	R	0x0105	MODULE ID	Unique 16-bit code. Module ID
15:12	R	0x0	MAJREV	Major Revision
11:8	R	0x3	MINREV	Minor Revision
7:0	R	0x00	MODULE APERTURE SIZE	Aperture size = 4 kB*(bit_value+1), so 0 means 4 kB (the default).

Chapter 26: Memory Arbiter

PNX15xx Series Data Book – Volume 1 of 1

Rev. 2 — 1 December 2004 Product data sheet

1. Introduction

All the memory traffic of PNX15xx Series modules is centralized into an internal Hub, through the MTL bus, before it gets to the main memory interface module. In addition to this network function, the HUB includes a generic arbiter for memory bandwidth allocation.

Remark: The arbiter only deals with module memory traffic and not with CPU memory traffic which is handled by the Main Memory Interface module, see Chapter 9 DDR Controller. This is different approach than the PNX1300 Series arbiter.

1.1 Features

The key features of the HUB are:

- Provides a hierarchical memory access network that connects module DMA ports to a single access port of the Main Memory interface. DMA agents, i.e. the PNX15xx Series modules are organized in clusters.
- Includes simple round-robin sub-arbitration for lower levels of hierarchy
- Provides sophisticated intermediate arbitration for upper levels of the network hierarchy
- Default settings allow each module to have access to the memory but may not fit latency requirement as soon as many modules are turned on simultaneously

2. Functional Description

The Arbiter module is used as an arbiter between different DMA channel clusters. Inside these clusters traffic from related DMA channels of Peripherals are combined by applying round-robin arbitration. (see Table 1 for a list of clusters and sub-arbitrated DMA channels).

The arbitration engine combines Time-Division Multiple Access (TDMA), priority, and round-robin methods; resulting in a guaranteed and high-level quality of service. The arbitration engine ensures programmable maximum latency and programmable minimal bandwidth to the unified resource. It also makes sure that best effort agents are fairly granted when higher priority agents do not request the channel.

The priority table can be dynamically altered by software. Two priority tables are implemented from which the inactive table can be changed on-the-fly. The Arbiter hardware takes care of smooth switching between the two tables.

After reset, the Arbiter is in “boot” mode and guarantees that each requesting agent is given a “grant” to main memory (Round Robin is the default arbitration method).

2.1 Arbiter Features

- Time-Division Multiple Access (TDMA) arbitration
  - guarantees maximum allowed latency
  - 128 TDMA slots
- Priority arbitration
  - guarantees minimum required bandwidth
  - 16 Priority slots
- Two level Round Robin arbitration
  - provides equal opportunities to the lower priority “best effort” or DMA write agents
  - 16 round robin slots in the first level
  - 8 round robin slots in the second level
- Dynamic arbitration scheme
  - Two sets of arbitration parameters can be defined. Selection can be made dynamically via software based on system needs.

2.2 ID Mapping

The Table 1 shows the mapping of each module to an unique identification numbers. The first column shows the IDs when programing the TDMA wheel. The second column indicates which ID number to use when programing the priority and roundrobin list. Table 1 also shows the amount of sub-arbitration for the given modules. If not otherwise noted the amount of buffering per DMA channel is 256 bytes.

Table 1: Peripheral ID and Sub-Arbitration

TDMA ID	ID	Modules	DMA Channels	Buffer size	Transaction size
0x8	0x0	2DDE	1 x R, 1 x W	2 x 256-byte buffer	128 Bytes
0x9	0x1	PCI	2 x R, 2 x W	4 x 256-byte buffer	128 Bytes
0xA	0x1	QVCP	4 x R	4 x 512-byte buffer	128 Bytes
0xB	0x3	VIP	3 x W	3 x 256-byte buffer	128 Bytes
0xC	0x4	VLD	1 x R, 2 x W	3 x 256-byte buffer	128 Bytes
0xD	0x5	FGPI	2 x W	2 x 512-byte buffer	128 Bytes
0xE	0x6	Reserved			
0xF	0x7	MBS (r)	3 x R	3 x 256-byte buffer	128 Bytes
0x0	0x8	MBS (w)	3 x W	3 x 256-byte buffer	128 Bytes
0x1	0x9	10/100 MAC	2 R x 3 W	10 x 32-byte buffer	32 Bytes
0x2	0xA	FGPO	2 x R	4 x 256-byte buffer 1 x 16-byte buffer	128 Bytes

Table 1: Peripheral ID and Sub-Arbitration *...Continued*

TDMA ID	ID	Modules	DMA Channels	Buffer size	Transaction size
0x3	0xB	SPDI/O, AI/O, GPIO (r,w)	3 x R, 3 x W	2 x 128-byte buffer	64 Bytes
0x4	0xC	DVDD	1 x R, 1 x W	2 x 256-byte buffer	128 Bytes
0x5	0xD	DCS Gate	1 x W	2 x 32-bit buffer	8 Bytes
0x6:0x7	0xE:0xF	Reserved			

2.2.1 DCS Gate

The DCS gate is a simple connection between the DCS bus and the Hub. Any 32-bit write transaction that is in the range of DCS_DRAM_LO to DCS_DRAM_HI causes a write transaction to main memory via the DCS Gate. This is provided for booting PNX15xx Series from EEPROM. The DCS_DRAM_LO and DCS_DRAM_HI registers are located in the Global Registers; see Chapter 3 System On Chip Resources and Figure 3 on page 3-30 for a simplified block diagram of PNX15xx Series.

2.3 Arbitration Algorithm

One of the most important purposes of the arbiter is to guarantee a high level of quality of service to the DMA agents (PNX15xx Series modules). In technical terms this means:

- the ability to guarantee a programmable maximum latency to DMA agents
- the ability to guarantee a programmable amount of bandwidth to DMA agents
- the ability to provide equal opportunity to DMA agents
- any (complex) combination of the three mechanisms mentioned above

The arbiter is not optimized to process requests for memory access from CPUs. Typically the performance of CPUs depends directly on the access latency to memory and for this reason they require the lowest possible memory latency. To realize this CPUs can best get their performance requirements via a private port on a multi-port memory controller. Therefore, the CPUs are not connected to the arbiter and do not route memory requests via the Hub.

To support the quality of service features as mentioned above the arbiter algorithm consists of a combination of three basic arbitration mechanisms. These are:

- Time-Division Multiple Access (TDMA) arbitration to guarantee maximum latency
- priority arbitration to guarantee bandwidth to reading Soft Real Time DMA (SRT DMA) agents
- round-robin arbitration to guarantee bandwidth to writing SRT DMA agents
- round-robin arbitration to provide equal opportunity for Best Effort (BE) DMA agents

The combination of these three basic algorithms operate together in the arbiter as shown in Figure 1.

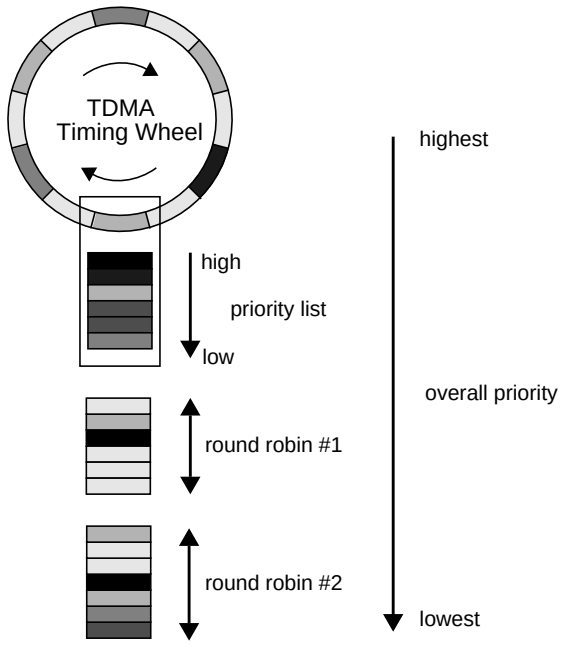


Figure 1: Arbitration Scheme

The TDMA timing wheel is implemented with 128 entries, numbered 1 to 128. The TDMA_entries field in the NR_entries_A[1] register will determine the actual number of entries that are used. TDMA entries higher than this value will be ignored. If the TDMA_entries is greater than 128 then all 128 entries are used, but no more. If TDMA_entries is set to zero then the TDMA timing wheel is not used for arbitration.

The priority list is implemented with 16 entries, numbered 1 to 16. The Priority_entries field in the NR_entries_A register will determine the actual number of entries that are used. If a value greater than 16 is written all 16 entries are used, but no more. If the Priority_entries is set to zero then the priority list is not used for arbitration.

The round robin #1 list is implemented with 16 entries, numbered 1 to 16. The round_robin1_entries field in the NR_entries_A register will determine the actual number of entries that are used. If a value greater than 16 is written all 16 entries are used, but no more. If the round_robin1_entries is set to zero then the round robin #1 list is not used for arbitration.

The round robin #2 list is implemented with 8 entries, numbered 1 to 8. The round_robin2_entries field in the NR_entries_A register will determine the actual number of entries that are used. If a value greater than 8 is written all 8 entries are used, but no more. If the round_robin2_entries is set to zero then the round robin #2 list is not used for arbitration.

1. All references to “Set A” registers also apply equally to “Set B”.

Assuming the arbiter has been configured to include the priority list and both round-robin lists, any arbiter decision is made through the following four steps:

1. First the DMA requests are compared against the current entry in the TDMA timing wheel. If the agent in the current entry is requesting this agent will be granted.
2. If the agent in the current entry is not requesting the DMA requests will be compared against the agents in the priority list and if one or more of the agents in the priority list is requesting the one that has the highest priority will be granted.
3. If none of the DMA requests matches the current entry in the TDMA timing wheel or one or more entries in the priority list, the arbiter will grant the DMA agent that has not been served for the longest time by choosing from the round robin #1 list. Every time the arbiter provides a grant to any DMA agent, the round robin #1 arbiter checks if this agent is in it's list and makes that agent the lowest priority entry in the round robin #1 list. If a certain agent is granted because of its entry in the TDMA timing wheel or priority list and the same agent has also an entry in the round-robin #1 list, then in the next clock cycle this agent will have the lowest priority in the round-robin #1 list. Also, in case there are multiple entries of the same agent in the round-robin #1 list, the highest entry in the list gets the lowest priority during the next cycle. The other entries of the same agents do not get the lowest priority.
4. If none of the DMA requests matches: the current entry in the TDMA timing wheel, or one or more entries in the priority list or one or more entries in the first round-robin list, the arbiter will grant the DMA agent that has not been served for the longest time from the round robin #2 list of entries. The round-robin #2 list operates the same way as the round-robin #1 list but all entries in this list have a lower priority than the entries in the round-robin #1 list.

The TDMA wheel will proceed to the next entry if and only if one of the two following situations apply:

- when there is a grant at the level of the TDMA wheel
- when there is no match in the complete list (TDMA, priority and both round-robin lists)

All entries in the TDMA wheel, priority list and both round-robin lists are fully programmable via the DTL MMIO interface of the arbiter. The same is true for the number of entries in any of these four. It is also possible to set the number of entries in the TDMA wheel, priority list and/or round-robin lists to zero. This allows the user to use only one of the four mechanisms or any combination of them. In case all four are set to zero for the active set of entries, the arbiter defaults to a round-robin arbitration over all agents.

The arbitration algorithm only starts after the arbiter has been properly initialized via the programming registers. Following the de-assertion of a hard reset, the arbiter uses a simple counting algorithm to arbitrate between all request inputs. In this boot mode agents are granted in the order that they are internally wired.

2.3.1 Arbiter Startup Behavior

After reset is de-asserted, the arbiter is placed in boot mode. In this mode, the arbiter sequentially grants each agent access to the memory if the agent has asserted its request. After de-assertion of rst_an starting with req[0], then req[1], etc. Four agents are checked in each clock cycle. This means that in the situation that only req[15] is asserted, it will take four clock cycles before the arbiter will grant this agent. In the first clock cycle it will check req[0] up to req[3], in the second clock cycle req[4] up to req[7], in the third clock cycle req[8] up to req[11] and the fourth clock cycle req[12] up to req[15]. The boot counter increments to next value when all agents corresponding to that count value have been serviced or when there is no request from the agents corresponding to that count value.

This mode is not intended to intelligently allocate memory bandwidth. Its goal is to simply make sure that all agents that request get granted. While in boot mode, it is expected that the system software will set up the arbiter via the DTL MMIO port and switch to the normal operation mode. As there are two sets of configuration registers (A and B), software should initialize one of the sets and then select the normal operation mode that corresponds to that set via a write to the Arbiter Control register. If necessary, the alternate set may be configured differently and the new configuration may be engaged by simply writing the new mode in the Arbiter Control register.

3. Operation

3.1 Clock Programming

The Hub operates with the Memory Controller clock, as well as the clocks of all the peripheral modules that connect to the Hub. There is no separate clock for the Hub.

3.2 Register Programming Guidelines

The default configuration of the Arbiter is to provide Round Robin access to all peripheral devices. This can be altered by software by programming the Arbiter. Once the Arbiter configuration is completed, the system should be able to operate without further change to the Arbiter; however it is possible for software to change the Arbiter configuration on-the-fly in order to change the minimum latency or the minimum memory bandwidth that is available to each peripheral device.

Remark: The active set of configuration registers (set A or set B) cannot be read by software once that set is activated. The inactive set may be safely written or read. If software needs to have access to the values within the active set, then a copy of these values should be maintained in main memory as a reference.

4. Register Descriptions

Table 2: Register Summary

Offset	Symbol	Description
0x06 4000—41FC	TDMA A	128 entries of TDMA timing wheel for set A
0x06 4200—423C	PRIORITY A	16 entries of priority list for set A
0x06 4240—427C	Reserved	
0x06 4280—42BC	FIRST Round Robin A	16 entries of first round robin list for set A
0x06 42C0—42FC	Reserved	
0x06 4300—431C	LAST Round Robin A	8 entries of last round robin list for set A
0x06 4320—43FC	Reserved	
0x06 4400—45FC	TDMA B	128 entries of TDMA timing wheel for set B
0x06 4600—463C	PRIORITY A	16 entries of priority list for set B
0x06 4640—467C	Reserved	
0x06 4680—46BC	FIRST Round Robin B	16 entries of first round robin list for set B
0x06 46C0—46FC	Reserved	
0x06 4700—471C	LAST Round Robin B	8 entries of last round robin list for set B
0x06 4720—47FC	Reserved	
0x06 4800	NR Entries A	Number of valid entries in arbitration lists for set A
0x06 4804	NR Entries B	Number of valid entries in arbitration lists for set B
0x06 4808—48FC	Reserved	
0x06 4900	Control	Register to control operation mode of arbiter
0x06 4904	Status	Register to monitor operation mode of arbiter
0x06 4908—4FF8	Reserved	
0x06 4FFC	MODULE_ID	Module ID and revision information

4.1 Register Table

Table 3: PMAN (Hub) Arbiter Registers

Bit	Symbol	Access	Value	Description
Arbiter Registers (Set A)				
Offset 0x06 4000—41FC Entries of TDMA Timing Wheel (Set A)				
31:10	Reserved	R	0	To ensure software backward compatibility, writes to unused or reserved bits should be zero and reads must be ignored.
9:8	R/W Grant	R/W	0	Grant on read, write or both 0x0 = grant independent whether it is a read or write 0x1, 0x2, 0x3 = reserved
7:5	Reserved	R	0	To ensure software backward compatibility, writes to unused or reserved bits should be zero and reads must be ignored.
4:0	Agent_ID	R/W	0	ID of the agent that is identified by this entry (see Table 1 on page 26-2 for IDs).

Table 3: PMAN (Hub) Arbiter Registers *...Continued*

Bit	Symbol	Access	Value	Description
Offset 0x06 4200—423C Entries of Priority List (Set A)				
Note: Offset 0x200 has the highest priority. This register is identical to Offset 0x06 4000—41FC Entries of TDMA Timing Wheel (Set A).				
Offset 0x06 4280—42BC Entries of Round Robin List #1 (Set A)				
This register is identical to Offset 0x06 4000—41FC Entries of TDMA Timing Wheel (Set A).				
Offset 0x06 4300—431C Entries of Round Robin List #2 (Set A)				
This register is identical to Offset 0x06 4000—41FC Entries of TDMA Timing Wheel (Set A).				
Arbiter Registers (Set B)				
Offset 0x06 4400—45FC Entries of TDMA Timing Wheel (Set B)				
31:10	Reserved	R	0	To ensure software backward compatibility, writes to unused or reserved bits should be zero and reads must be ignored.
9:8	R/W Grant	R/W	0	Grant on read, write or both. 0x0 = grant independent whether it is a read or write 0x1, 0x2, 0x3 = reserved
7:5	Reserved	R	0	To ensure software backward compatibility, writes to unused or reserved bits should be zero and reads must be ignored.
4:0	Agent_ID	R/W	0	ID of the agent that is identified by this entry.
Offset 0x06 4600—463F Entries of Priority List (Set B)				
Note: Offset 0x200 has the highest priority. This register is identical to Offset 0x06 4400—45FC Entries of TDMA Timing Wheel (Set B).				
Offset 0x06 4680—46BC Entries of Round Robin List #1 (Set B)				
This register is identical to Offset 0x06 4400—45FC Entries of TDMA Timing Wheel (Set B).				
Offset 0x06 4700—471C Entries of Round Robin List #2 (Set B)				
This register is identical to Offset 0x06 4400—45FC Entries of TDMA Timing Wheel (Set B).				
Offset 0x06 4800 NR_ENTRIES_A (Set A)				
31:28	Reserved	R/W	0	To ensure software backward compatibility, writes to unused or reserved bits should be zero and reads must be ignored.
27:24	round_robin2_entries	R/W	0	Number of valid entries in last round robin list #2 Programming any value > 8 will result in use of the full round-robin list.
23:21	Reserved	R/W	0	To ensure software backward compatibility, writes to unused or reserved bits should be zero and reads must be ignored.
20:16	round_robin1_entries	R/W	0	Number of valid entries in first round robin list #1 Programming any value > 16 will result in use of the full round-robin list.
15:13	Reserved	R/W	0	To ensure software backward compatibility, writes to unused or reserved bits should be zero and reads must be ignored.
12:8	priority_entries	R/W	0	Number of valid entries in priority list Programming any value > 16 will result in use of full priority list.
7:0	TDMA_entries	R/W	0	Number of valid entries in TDMA wheel Programming any value > 128 will result in use of all 128 entries.
Offset 0x06 4804 NR_ENTRIES_B (Set B)				
This register is identical to Offset 0x06 4800 NR_ENTRIES_A (Set A).				

Table 3: PMAN (Hub) Arbiter Registers *...Continued*

Bit	Symbol	Access	Value	Description
Offset 0x06 4900	***Arbiter Control***			
31:2	Reserved	R/W	0	To ensure software backward compatibility, writes to unused or reserved bits should be zero and reads must be ignored.
1:0	Arbiter_mode	R/W	0	Operational mode of the Arbiter 00 = Boot mode 01 = Use register set A. 10 = Use register set B. 11 = Reserved
Offset 0x06 4FFC	***Arbiter Module_ID***			
31:16	Module_ID	R	0x1010	Arbiter module ID number
15:12	Major_revision	R	0	
11:8	Minor_revision	R	0	
7:0	Aperture	R	0	4 KB aperture size

Chapter 27: Power Management

PNX15xx Series Data Book – Volume 1 of 1

Rev. 2 — 1 December 2004 Product data sheet

1. Power Management Mechanisms

This chapter describes a standard programming model used to facilitate power management of peripheral modules.

In the PNX15xx Series, the primary method for reducing the power consumption of certain modules is to reduce the frequency or completely stop the clock signal to those modules. However, modules such as the CPU and the memory system have their own built-in power management mechanisms.

These mechanisms are described individually in this chapter due to their uniqueness and interconnection with system-level operations and global resources.

1.1 Clock Management

The on-chip clock module contains registers which connect/disconnect different clock sources to peripheral modules. The clock signals of most modules on the PNX15xx Series can be stopped if the appropriate procedure is followed. Due to wake-up logistics, however, it is not possible to completely stop the clock signals to some of the modules.

1.1.1 Essential Operating Infrastructure During Powerdown

The DCS bus (also called MMIO bus) clock should not be stopped directly but only using the procedure defined in Chapter 5 The Clock Module Section 27 on page 27-1.

Remark: Once all the clocks have been turned off, the PCI module cannot reply to any request on the PCI bus. Therefore it must be ensured that this condition does not arise.

1.1.2 Module Powerdown Sequence

The following sequence of events must take place for powerdown to occur:

- The TM3260 or an external host prepares the module for powerdown. This is achieved by disabling the module if it was active. Applying a software reset is recommended.
- At this point, any pending device interrupts should have been handled by the CPU to ensure the device does not generate new interrupts or bus transactions when disabled. Note that module registers are still fully functional. Any device register can be read/written and the device can be re-enabled if desired.

- The CPU writes/sets the device's POWERDOWN control bit (usually bit 31 of offset 0xFF4). Setting this bit causes the device to power down elements such as memories and register files.

Remark: This bit does NOT gate the internal module clock or other clocks as clock gating is not allowed inside a module.

- After setting the powerdown bit, none of the device's registers is accessible, except the one containing the POWERDOWN bit, which is 100% operational.
  - Reads from any other register do not hang.
  - Writes to any register (except the POWERDOWN bit register) are completed fully or result in an error.
- The CPU programs the Clock module to stop/slow down the clock signals. This assures the Clock module is stopped in a controlled way (no glitches/illegal periods).
- At this point, the register with the POWERDOWN bit is still the only accessible register and the block is fully powered down.

1.1.3 Peripheral Module Wakeup Sequence

Waking up a module is also under CPU control and is the reverse sequence to powerdown:

- Program the Clock module so that all related clock sources are set to their normal operational frequencies.
- Reset the POWERDOWN bit in the module.
- Set up the module's configuration registers if needed.
- Enable the module.

1.1.4 TM3260 Powerdown Modes

The TM3260 CPU has two modes: Partial Powerdown and Full Powerdown.

Partial Powerdown Mode

The TM3260 CPU enters partial powerdown mode by performing a 'store' to a specific MMIO address (the POWERDOWN register). The TM3260s then finish any pending transactions and go into a partial powerdown. In partial powerdown mode, cycle counters, timers and interrupt logic in the TM3260s are still active. The TM3260 CPU wakes up from partial powerdown when an interrupt occurs or there is an access to its MMIO space. This commonly used by the idle task in an operating system.

Full Powerdown Mode

The TM3260 also have an externally-initiated full power shutdown mode i.e., no wakeups when an interrupt occurs. Entering this mode is requested by asserting an input signal to the TM3260. When this signal is asserted, theTM3260 finishes pending transactions and gates-off its core clock.

This powerdown state can be initiated by an external host processor by writing to bit TM32_CONTROL.TM_PWRDWN_REQ, see Chapter 3 System On Chip Resources. The TM3260 only exits this mode when this bit is de-asserted. At this point in time the TM3260 clock may be removed by the host.

The second method to shutdown the TM3260 clock as well as the MMIO clock is to follow the procedure defined in Chapter 5 The Clock Module Section 27 on page 27-1. This is solution for standalone systems where PNX15xx Series is the master of the system.

1.1.5 SDRAM Controller

Power consumption of the MMI is lowest when it is halted. There are two different ways to achieve halting the MMI:

- Writing the halt register field of a software programmable MMIO register.
- Programming the MMI to go into halt mode automatically after a certain period of inactivity.

Remark: Before halting the MMI, make sure that there are no pending memory transactions.

MMIO Directed Halt

The HALT bit of MMIO register IP_2031_CTL can be written with a ‘1’ to indicate a request for halting. Write a ‘0’ to this bit to indicate a request for taking the DDR controller out of halt mode.

Remark: It is recommended that putting the MMI in MMIO direct-halt mode (with MMIO registers) before reprogramming the configuration and timing registers in MMI so that the on-going transactions are not effected. When MMIO registers DDR_MR and DDR_EMR are reprogrammed, a start action has to be performed (after the MMI is unhalted), for the new DDR values to take effect.

Auto Halt

The MMI can be programmed such that it goes into halt mode when it has observed a certain period of inactivity. This is accomplished by programming the MMIO registers AUTO_HAL_LIMIT and IP_2031_CTL. The MMI will exit the halt mode automatically when a new MTL memory request is presented to one of its input ports. The MTL clock and DCS clock cannot be turned off to operate in this mode.

Remark: This modes introduces extra latency on memory transactions and it is not a recommended operating mode.

Chapter 28: Pixel Formats

PNX15xx Series Data Book – Volume 1 of 1

Rev. 2 — 1 December 2004 Product data sheet

1. Introduction

Several hardware subsystems in the PNX15xx Series deal directly with images. Such hardware subsystems must follow the same memory representation and interpretation of images in order to successfully pass images between subsystems.

Software modules also constitute subsystems that can produce or consume images. Although most modules are designed with an abstraction layer from the underlying format, some legacy modules may exist that assume a particular pixel representation in memory.

In order to run a wide variety of software modules the PNX15xx Series uses the following pixel format strategy:

- A limited number of native pixel formats are supported by all image subsystems, as appropriate.
- The Memory Based Scaler supports conversion from arbitrary pixel formats to any native format during the anti-flicker filtering operation. (This operation is usually required on graphics images anyway, so no extra passes are introduced.)
- Hardware subsystems support all native pixel formats in both little-endian and big-endian system operation.
- Software always sees the same component layout for a native pixel format unit, whether it is running in little-endian or big-endian mode— i.e., for a given native format, RGB (or YUV) and alpha are always in the same place.
- Software dealing with multiple units at a time in Single Instruction Multiple Data (SIMD) style must be aware of system endian mode.
- The native formats of the PNX15xx Series include the most common indexed, packed RGB, packed YUV and planar YUV formats used by Microsoft® DirectX and Apple® QuickTime, with 100% bit layout compatibility in both little and big-endian modes.

2. Summary of Native Pixel Formats

Table 1 and Figure 1 summarize the native pixel formats and image hardware subsystems that support them.

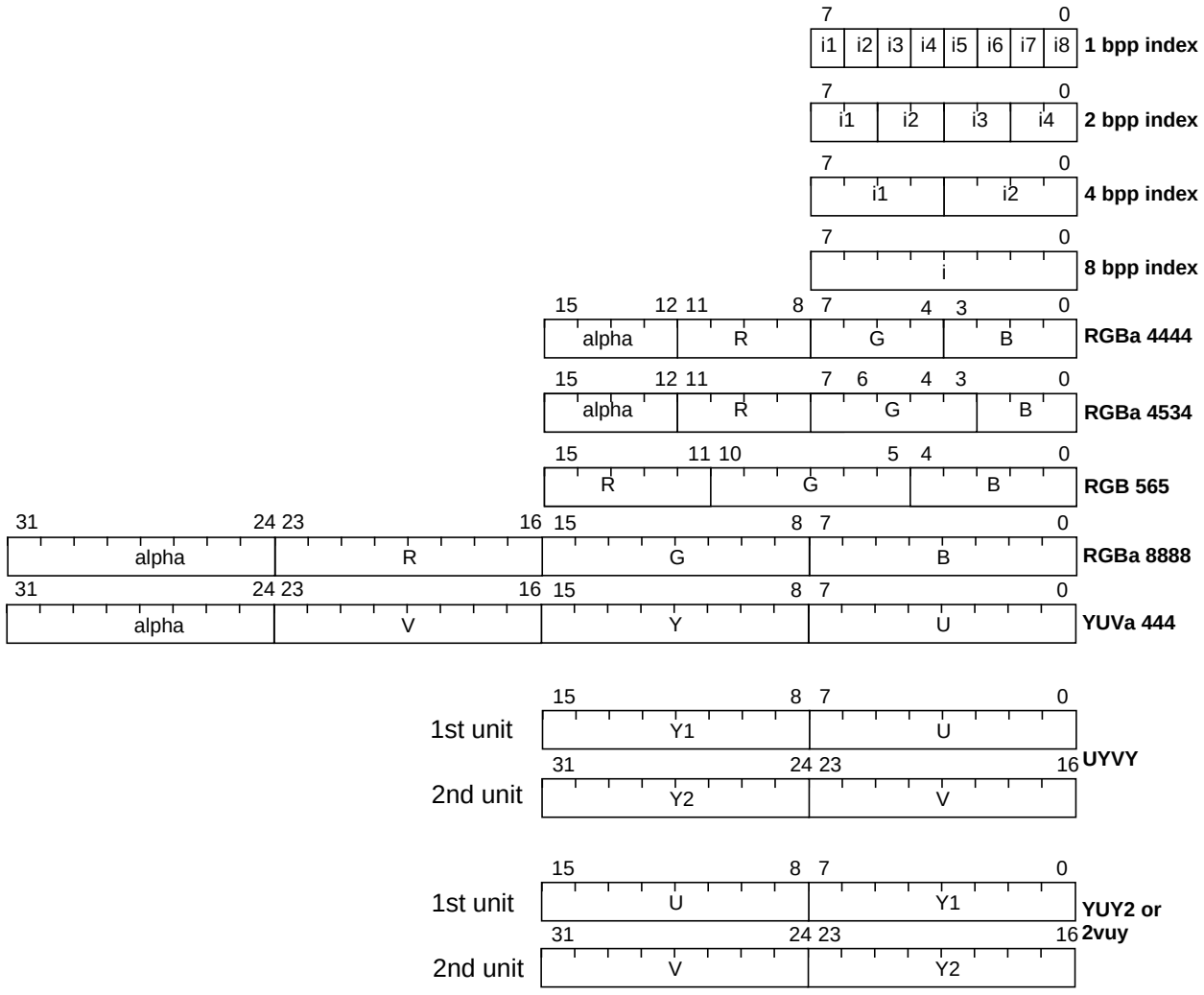
Table 1: Native Pixel Format Summary

Name	Note	VIP Out	MPG Out	MBS In	MBS Out	2D Draw Eng (2)	QVCP In
1 bpp indexed	CLUT entry = 24-bit color + 8-bit alpha			x			x
2 bpp indexed				x			x
4 bpp indexed				x			x
8 bpp indexed				x		x	x
RGBa 4444	16-bit unit, containing 1 pixel with alpha	(1)		x	x	x	x
RGBa 4534		(1)		x	x	x	x
RGB 565	16-bit unit, containing 1 pixel, no alpha	(1)		x	x	x	x
RGBa 8888	32-bit unit, containing 1 pixel with alpha	(1)		x	x	x	x
packed YUVa 4:4:4	32-bit unit containing 1 pixel with alpha	x		x	x	x	x
packed YUV 4:2:2 (UYVY)	16-bit unit, 2 successive units contain 2 horizontally adjacent pixels, no alpha	x		x	x		x
packed YUV 4:2:2 (YUY2, 2vuy)		x		x	x		x
planar YUV 4:2:2	3 arrays, 1 for each component	x		x	x		
semi-planar YUV 4:2:2	2 arrays, 1 with all Ys, 1 with U and Vs	x		x	x		x
planar YUV 4:2:0	3 arrays, 1 for each component			x	x		
semi-planar YUV 4:2:0	2 arrays, 1 with all Ys, 1 with U and Vs		x	x	x		x
semi-planar 10-bit YUV 4:2:2	2 arrays, 1 with all Ys, 1 with U and Vs 3Ys are packed in 4 Bytes and 3 sets of UV pixels are packed in 8 Bytes						x
semi-planar 10-bit YUV 4:2:0	2 arrays, 1 with all Ys, 1 with U and Vs 3Ys are packed in 4 Bytes and 3 sets of UV pixels are packed in 8 Bytes						x
Packed 10-bit YUV 4:2:2(UYVY)	6Ys and 3UVs are packed in 16 Bytes.						x

(1) The VIP is capable of producing RGB formats, but not when performing horizontal scaling.

(2) Shown are the 2D Drawing Engine frame buffer formats where drawing, rasterops and alpha-blending of surfaces can be accelerated. The 2D Drawing Engine host port also supports 1 bpp monochrome font/pattern data, and 4 and 8-bit alpha only data for host initiated anti-aliased drawings.

The layout shown in Figure 1 is the way that a unit ends up in a CPU register given a unit size (8, 16 or 32-bit) load operation, regardless of the PNX15xx Series endian mode of operation.



(For planar YUV 4:2:0 formats, refer to Figure 9 and Figure 10.)

Figure 1: Native Pixel Format Unit Layout

3. Native Pixel Format Representation

3.1 Indexed Formats

The indexed formats support a 1, 2, 4 or 8-bit pixel format. For each of the respective 2, 4, 16 or 256 code values, a full look-up for a 24-bit color and 8-bit alpha is performed, using a subsystem-specific, programmable color look-up table.

Figure 2 shows the “software view” of the four indexed formats. Packing of pixels within the byte always uses the “first, left-most pixel in most significant bit(s)” packing convention. Pixel groups from left to right have increasing memory byte addresses.

Indexed formats are accepted by the QVCP and MBS. The 2D Drawing Engine supports 8 bpp indexed as a frame buffer format, but supports no other indexed variants.

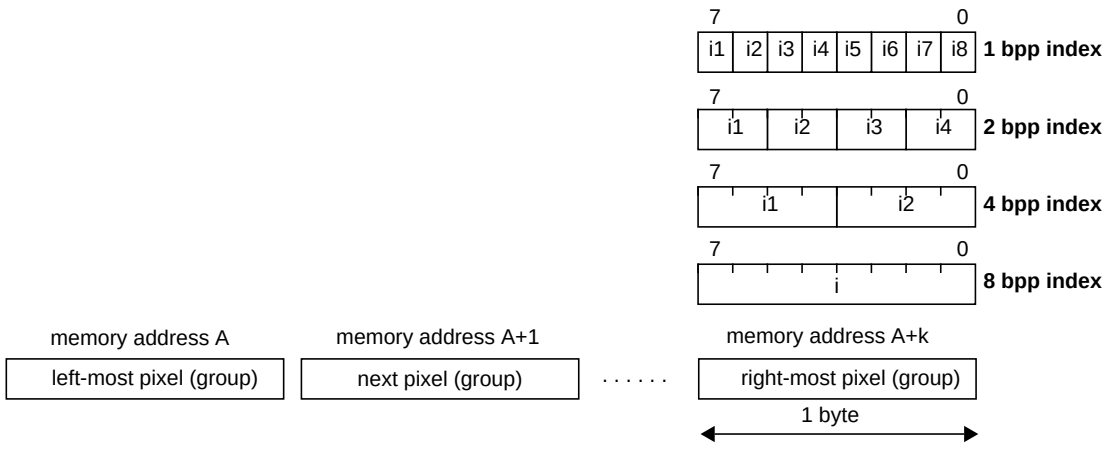


Figure 2: Indexed Formats

3.2 16-Bit Pixel-Packed Formats

The 16-bit native formats are RGBa 4444, RGBa 4534 and RGB 565.

Figure 3 shows the “software view” of these formats. The CPU register layout is always the same when performing 16-bit load/store instructions, regardless of system endian mode. Adjacent pixels have left-to-right increasing memory addresses.

16-bit formats are accepted and produced by the QVCP, VIP, MBS and the 2D Drawing Engine.

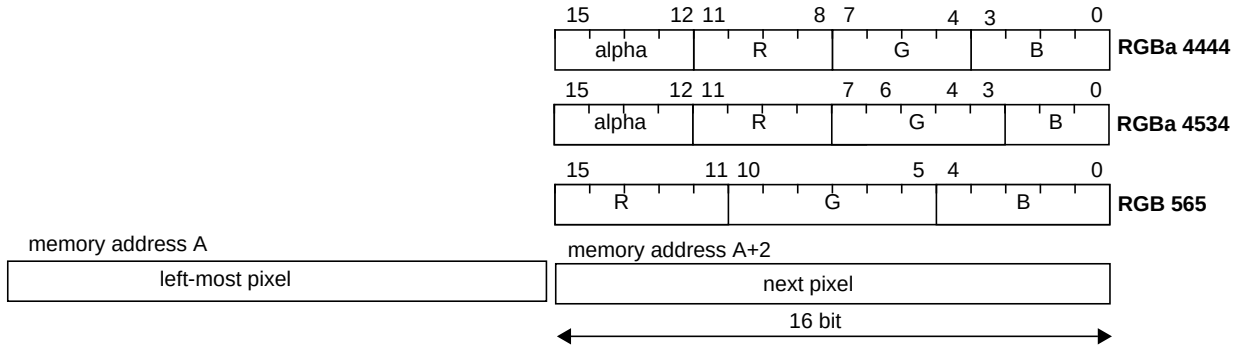


Figure 3: 16-Bit Pixel-Packed Formats

3.3 32-Bit Pixel-Packed Formats

The 32-bit formats include RGBa 8888 and YUVa 4:4:4 with an 8-bit per pixel alpha.

Figure 4 shows the “software view,” resulting from a 32-bit load into a CPU register. This view is independent of system endian mode. Left-to-right pixels have increasing memory addresses.

32-bit formats are accepted and produced by all units except the MPEG-2 decoder.

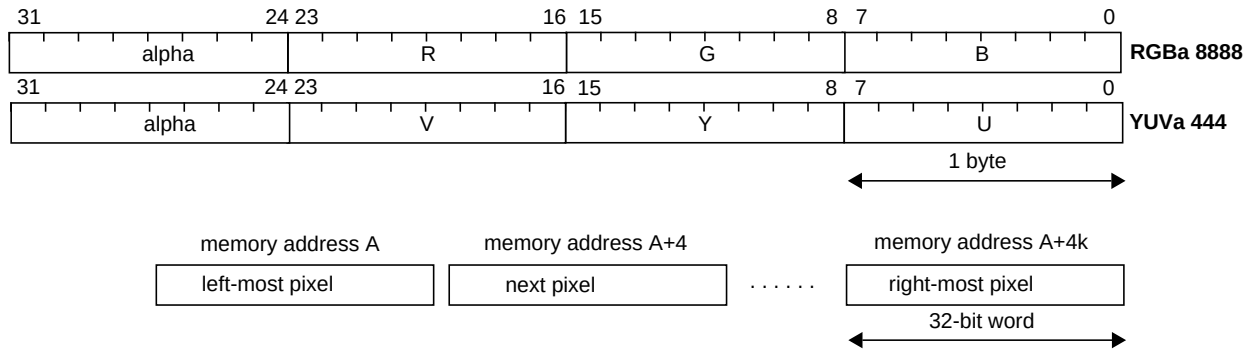


Figure 4: 32-Bit/Pixel Packed Formats

3.4 Packed YUV 4:2:2 Formats

Packed YUV 4:2:2 formats store two horizontally adjacent pixels into two 16-bit units. Each pixel has an individual luminance (Y1 for the left of the pair, Y2 for the right of the pair). There is a single U and V value associated with the pair. The U and V values are taken from the same spatial position as the Y1 sample (Figure 7).

There are two variants of packed YUV 4:2:2: the Microsoft ‘UYVY’ format and the Microsoft ‘YUY2’ format. The big-endian view of the YUY2 format is identical to the Power MacIntosh ‘2vuy’ format.

Figure 5 and Figure 6 show the software view of UYVY and YUY2/2vuy. Two successive 16-bit units contain a pair of pixels. This view is independent of system endian mode.

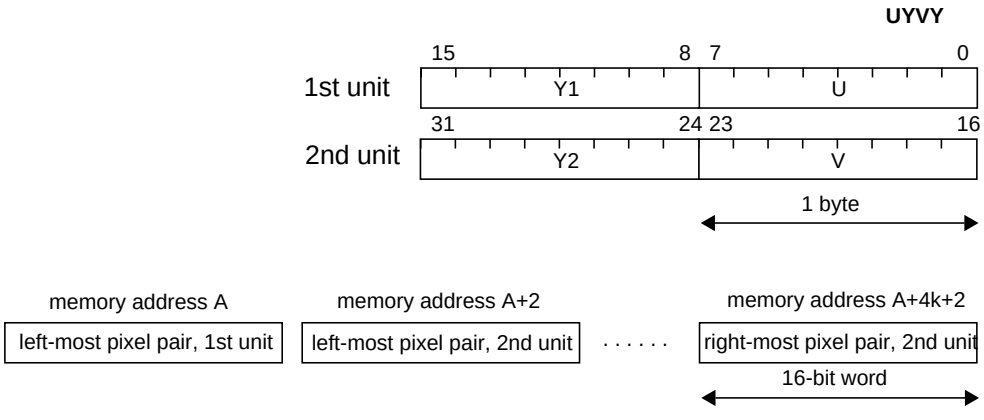


Figure 5: UYVY Packed YUV 4:2:2 Format

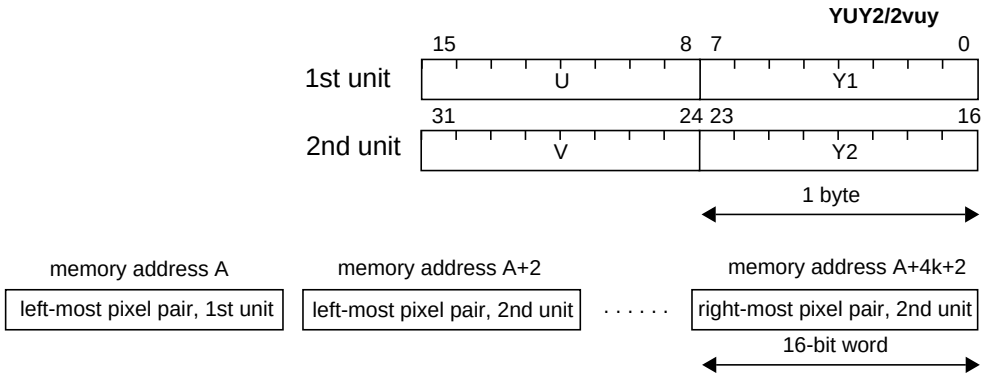


Figure 6: YUY2/2vuy Packed YUV 4:2:2 Format

3.5 Planar YUV 4:2:0 and YUV 4:2:2 Formats

The spatial sampling structure of planar YUV 4:2:0 data is shown in Figure 8. The planar YUV 4:2:2 data has the spatial sampling structure as shown in Figure 7.

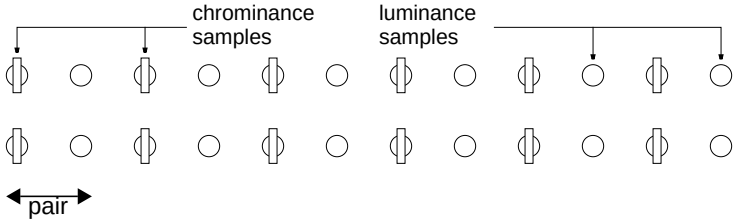


Figure 7: Spatial Sampling Structure of Packed and Planar YUV 4:2:2 Data

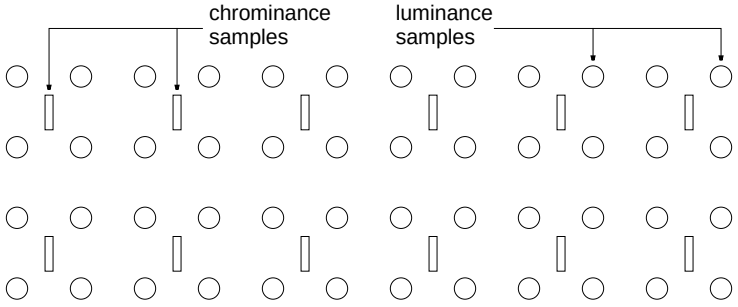


Figure 8: Spatial Sampling Structure of YUV 4:2:0 Data

3.5.1 Planar Variants

There are two variants of planar YUV 4:2:0 and YUV 4:2:2 formats.

Planar or 3-Plane Format

An image is described by 3 pointer values (Py, Pu, Pv). Each pointer points to a 2D array of Y, U and V values as shown in Figure 9.

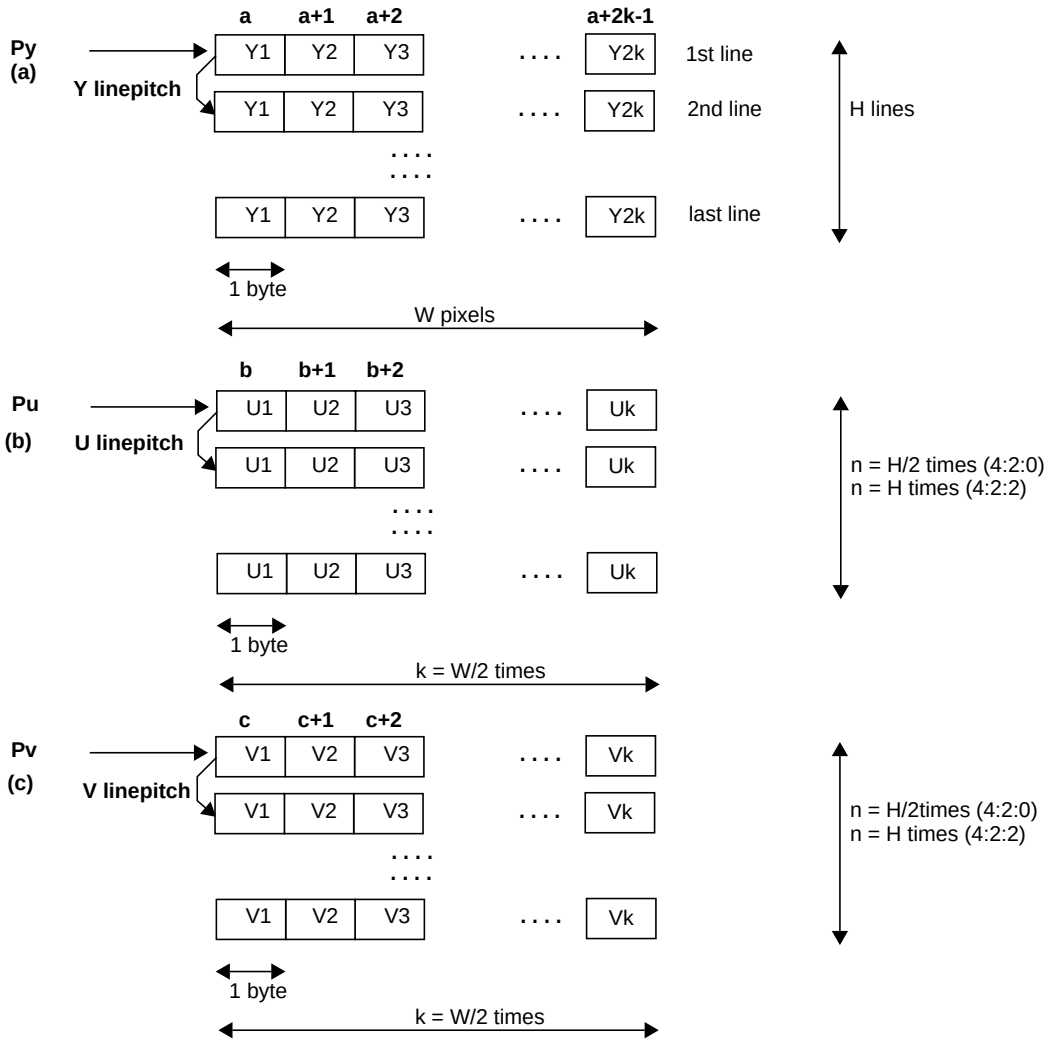


Figure 9: Planar YUV 4:2:0 and 4:2:2 Formats

Semi-Planar or 2-Plane Format

An image is described by 2 pointer values (Py, Puv). The Y pointer points to a 2D array of Y values. The Puv pointer points to a 2D array of UV pair values. Note that the U value of a UV pair always has the lower byte address. See Figure 10.

The MBS supports all planar formats on input and output. The VIP can produce the planar and semi-planar YUV 4:2:2 planar formats. The semi-planar YUV 4:2:0 format is the only format produced by the MPEG video decoder hardware.

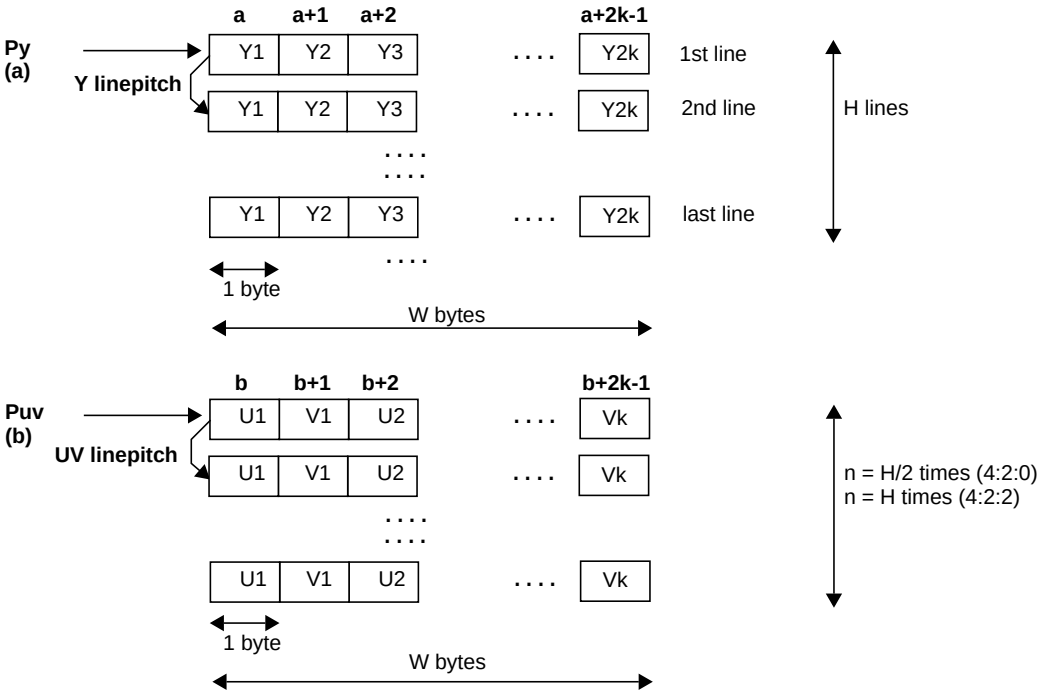


Figure 10: Semi-Planar YUV 4:2:0 and YUV 4:2:2 Formats

3.5.2 Semi-Planar 10-Bit YUV 4:2:2 and 4:2:0 Formats

The semi-planar 10-bit YUV 4:2:2 and 4:2:0 formats, shown in Figure 11, are generated by an external device and stored in the PNX15xx Series memory through the Tunnel interface. QVCP supports the semi-planar 10-bit YUV 4:2:2 and 4:2:0 formats.

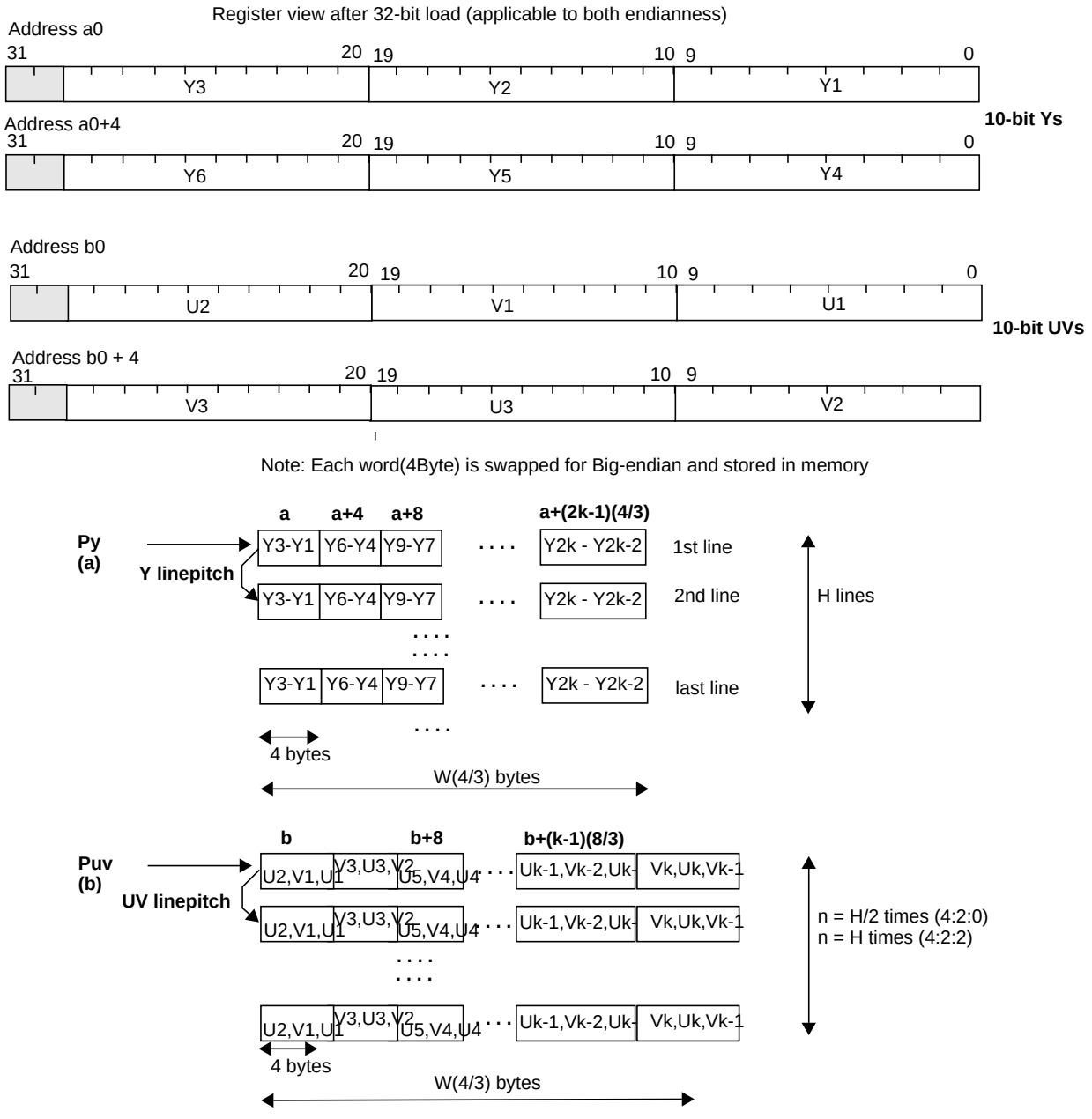


Figure 11: Semi-Planar 10-bit YUV 4:2:0 and YUV 4:2:2 Formats

3.5.3 Packed 10-bit YUV 4:2:2 format

The packed 10-bit YUV 4:2:2 format is generated by an external device and stored in the PNX15xx Series memory through the Tunnel interface. QVCP supports the packed 10 bit YUV 4:2:2 format, as shown in Figure 12.

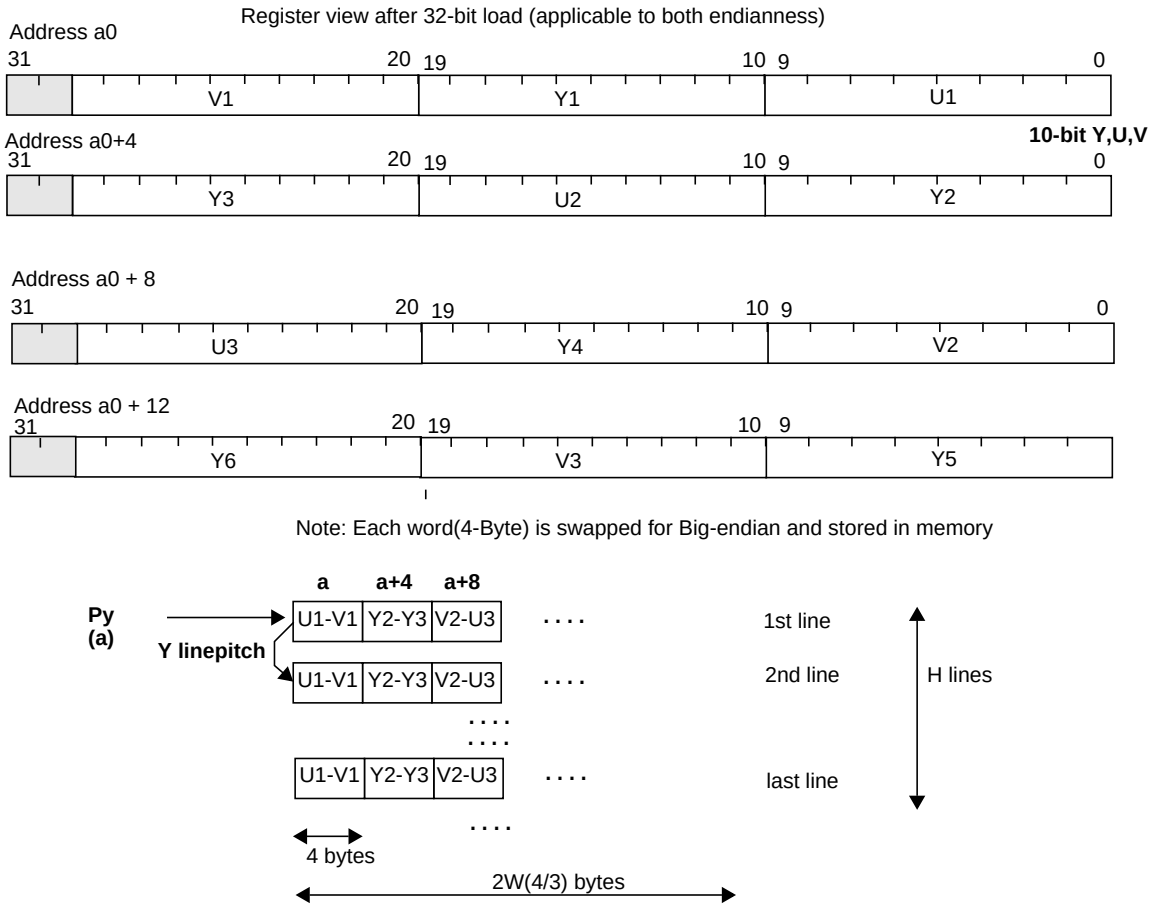


Figure 12: Packed 10-bit YUV 4:2:2 Format

4. Universal Converter

The MBS input stage contains a universal pixel format converter that can convert any packed 16 or 32-bit pixel RGB format to an 8-bit alpha, R, G and B value for internal processing. This conversion can be done in combination with any MBS operation, particularly anti-flicker filtering.

The conversion is done by specifying the following:

- the width (16 or 32 bits) of a unit (this designates endian mode handling)
- the position (bit 31..0) within the unit for each of the alpha, R,G and B fields
- the width (1..8 bit) of each of the alpha, R,G and B fields

5. Alpha Value and Pixel Transparency

Many of the native pixel formats include a per-pixel alpha value. This alpha value is an inverse measure of the ‘transparency’ of a pixel. The QVCP and 2D Drawing Engine are the only subsystems that interpret per-pixel alpha values when compositing surfaces. The native pixel format convention of a per-pixel alpha is shown in Table 2.

Table 2: Alpha Code Value and Pixel Transparency

Alpha Code	Transparency	Value
0	Fully transparent	0/256
1	Almost fully transparent	1/256
...		
254	Almost fully opaque	254/256
255	Fully opaque	256/256

The QVCP input stage allows full alpha value table look-up using a 256 entry, 8-bit wide table. This can be used to translate a non-native format alpha convention to the native convention.

6. RGB and YUV Values

8-Bit Data

For 8-bit data, the full range of values is allowed i.e., [0~255]. As an option, the data range could be clipped in the MBS or VIP according to the ITU-K BT. 601-4 specification.

- Y range [16~235]
- U range [16~240]
- V range [16~240]

or

- R range [16~235]
- G range [16~235]
- B range [16~235]

10-Bit Data

For 10-bit data, the full range of values is allowed i.e., [0~1023]. The blank level is programmable through the register.

7. Image Storage Format

With the exception of the planar formats, described in Section 3.5, the layout of an image in memory is determined by the following elements:

- pixel format, which implies the unit size(s)
- origin pointer—the (byte) address of the first unit of the image
- line pitch—the address difference between a pixel on a line and a pixel directly below it
- width W, in number of pixels
- height H, in number of lines

Note that for indexed formats, each unit contains one or more pixels. For the packed formats, a unit is a pixel, with the exception of packed YUV 4:2:2 where two units are needed to describe a pixel pair.

Figure 13 shows how images are stored in memory

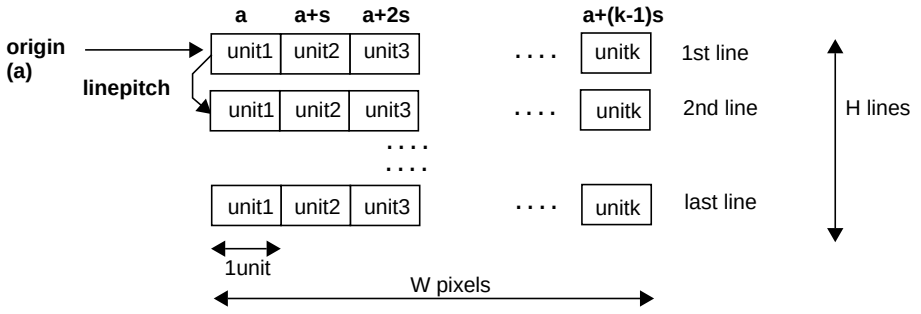


Figure 13: Image Storage Format

8. System Endian Mode

The PNX15xx Series is designed to run either little-endian or big-endian software. The entire system always operates in a single endian mode—i.e. the CPU and all hardware subsystems run either little or big-endian. This is determined by a global endian mode flag.

The endian mode determines how a multi-byte value is stored to/loaded from memory byte addresses.

For the native pixel formats, Section 3. always shows two elements in the figures: the layout of a ‘unit’, which is always 8, 16 or 32 bits, and the mapping of adjacent units to memory byte addresses. These two elements are always maintained, independent of system endian mode.

What this implies is that each hardware subsystem needs to map a unit to memory byte addresses in an endian mode-dependent manner. The rules are as follows:

- Storing a 16-bit unit to address ‘A’ results in modifying memory bytes ‘A’ and ‘A+1’
- Storing a 32-bit unit to memory address ‘A’ results in modifying memory bytes ‘A’ to ‘A+3’

- In little-endian mode, the least significant bits of a unit go to the lowest byte address
- In big-endian mode, the most significant bits of a unit go to the lowest byte address
- Going from left to right, adjacent units go to increasing memory addresses

Chapter 29: Endian Mode

PNX15xx Series Data Book – Volume 1 of 1

Rev. 2 — 1 December 2004 Product data sheet

1. Introduction

Two addressing conventions exist in the computer industry: little-endian and big-endian.

- In the little-endian convention, a multi-byte number is stored in memory with the least significant byte at the lowest memory address, and subsequent bytes at increasing addresses.
- In the big-endian convention, the most significant byte is stored at the lowest memory address, and subsequent bytes are stored at increasing addresses.

The PNX15xx Series supports both big-endian mode and little-endian mode, allowing it to run either little-endian or big-endian software, as required by the particular application.

This chapter explains the concepts and programmer's view of data structures required for module control and module DMA. The chapter also contains a section that shows how hardware modules, buses and bridges implement the programmer's view.

1.1 Features

- The PNX15xx Series supports big-endian and little-endian operation.
- The system as a whole (all CPUs and all on-chip DMA modules) must operate in the same endian mode.
- When used with an external CPU, the PNX15xx Series must operate in the same endian mode as the external CPU. If the endian modes between the external CPU and the PNX15xx Series are different, then the external CPU must be aware of this difference and appropriately handle all data swapping.
- The endian-mode choice is made at boot time. It can be changed by software, but this requires a partial system reset.

2. Functional Description

2.1 Endian Mode System Block Diagram

Figure 1 shows a system block diagram with two example of DMA modules. Each DMA module deals with a 16-bit unit size. Module 1 transfers data via a 32-bit DTL bus with DTL Data ordering rules, while Module 2 transfers data via a 32-bit DTL bus with address-invariance rules. For the following, assume that both modules are input

modules that move data to system memory via a DTL-DMA interface. For DMA output modules that read data from memory, the operation is similar but in the opposite direction.

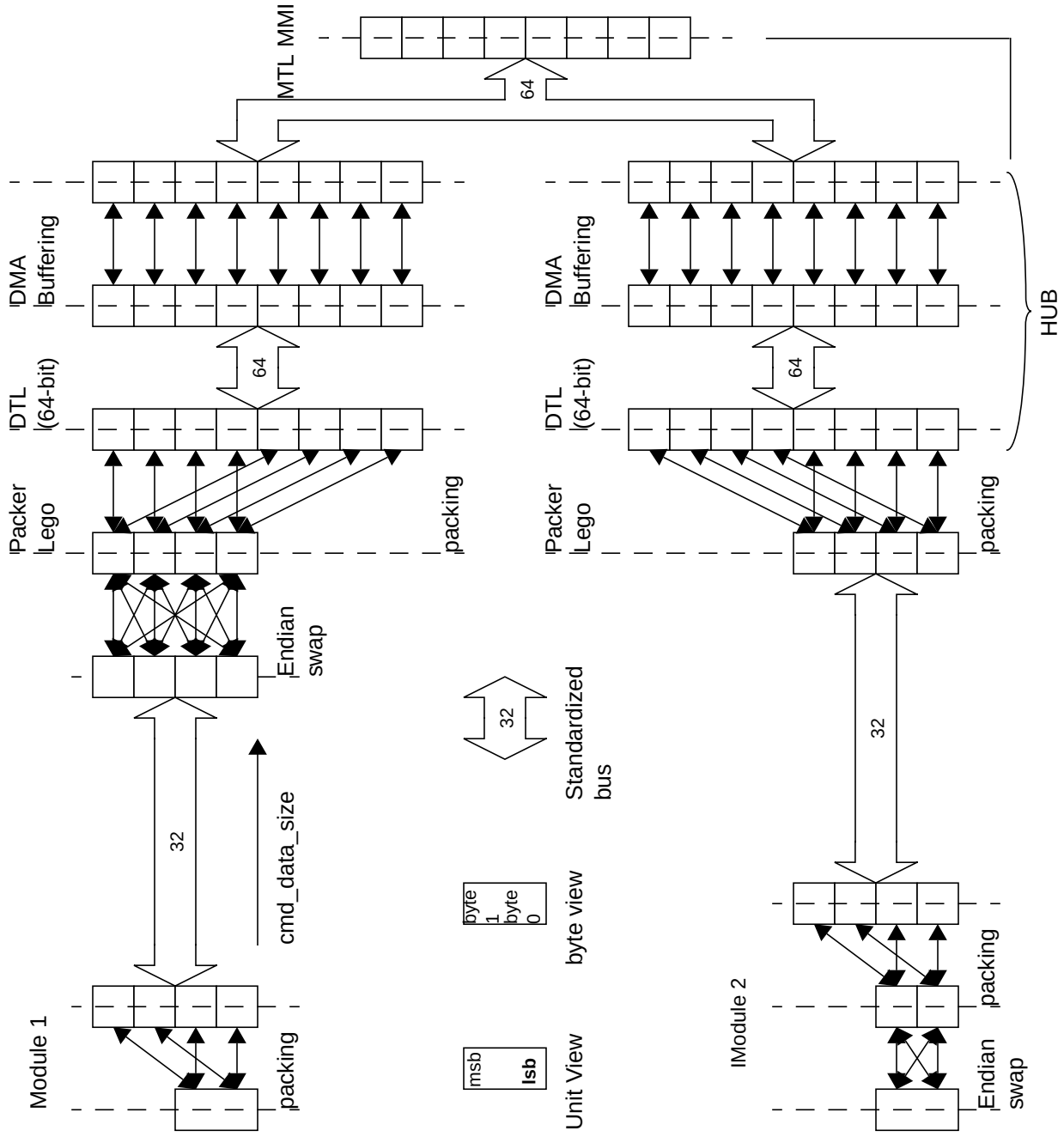


Figure 1: System Block Diagram: Endian-Related Blocks

3. Endian Mode Theory

There are two basic laws of endian mode: one imposed by CPU history, and one by convention. Both must be met by any system architecture that implements dual-endian operation capability. In addition, there are some implementation choices for a system architecture. Section 6. explains the choices that were made for the PNX15xx Series on-chip buses. These choices are somewhat arbitrary, but they must be followed to ensure future compatibility.

3.1 Law 1: The “CPU Rule”

This section is intended to explain CPU endian modes in detail. For those familiar with CPUs and endian modes, it is optional reading.

The following summarizes how CPUs and byte-addressable memory operate:

- When storing an “n byte” size item from a CPU register to memory at address “A,” the bytes modified are always the bytes with byte address “A”..”A+n-1.”
- In little-endian mode, the byte at address “A” receives the least significant bits of the multi-byte item.
- In big-endian mode, the byte at address “A” receives the most significant bits.

Consider the following example a (hypothetical) C struct:

```
    struct {
    UInt8C;//"command" byte
    UInt8F;//"flags" byte
    UInt16L;//"length" 16 bit value
    UInt32A;//"address" 32 bit value
} DMA_Descriptor;
```

Remark: This is based on an example in the Apple® publication, ”Designing PCI Cards and Drivers for Power Macintosh Computers,” Appendix A.

A compiler would assign byte offsets as follows: C:0, F:1, L:2, A:4. This assignment is independent of system endian mode.

Figure 2 and Figure 3 show the two layout views.

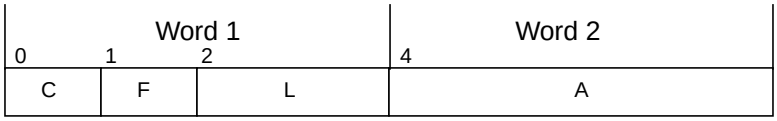


Figure 2: Big-Endian Layout of DMA_Descriptor

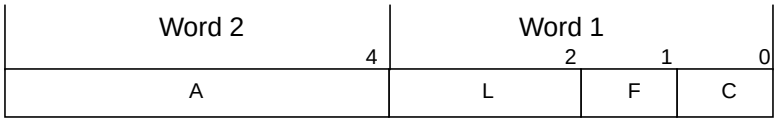


Figure 3: Little-Endian Layout of DMA_Descriptor

The TM3260 CPU on the PNX15xx Series both support 8, 16 and 32-bit data types and a memory system that is byte addressable. The CPU support a big-endian and little-endian mode of operation. The effect of a CPU store instruction on memory is defined in Table 1. As an example, a 16-bit store operation always stores the 16-bit quantity contained in the 16 lsbits of the CPU register. And the memory locations affected are “a” and “a+1.” But which byte goes where is dependent upon endian mode.

Table 1: Memory Result of a Store to Address ‘a’ Instruction

Endian Mode	R13 Content	Data Size	Result of $Store_{size}$(R13, Address a)
little	0x04050607	8 bits	m[a] = 0x07
little	0x04050607	16 bits	m[a] = 0x07; m[a+1] = 0x06
little	0x04050607	32 bits	m[a] = 0x07; m[a+1] = 0x06; m[a+2] = 0x05; m[a+3] = 0x04
big	0x04050607	8 bits	m[a] = 0x07
big	0x04050607	16 bits	m[a] = 0x06; m[a+1] = 0x07
big	0x04050607	32 bits	m[a] = 0x04; m[a+1] = 0x05; m[a+2] = 0x06; m[a+3] = 0x07

The effect of a CPU load instruction on a register is defined for unsigned and signed loads in Table 2 and Table 3. Note that a load always sets all bits of the CPU register. In the case of an unsigned load, higher order bits are filled with zeroes. In the case of a signed load, higher order bits are filled with the sign bit of the data item loaded.

Table 2: Register Result of an (Unsigned) Load Instruction

Memory Content	Endian Mode	Data Size	Register Value Result of $Load_{size}$(Address a)
m[a] = 0xAA; m[a+1] = 0xBB; m[a+2] = 0xCC; m[a+3] = 0xDD	little	8 bits	0x000000AA
m[a] = 0xAA; m[a+1] = 0xBB; m[a+2] = 0xCC; m[a+3] = 0xDD	little	16 bits	0x0000BBAA
m[a] = 0xAA; m[a+1] = 0xBB; m[a+2] = 0xCC; m[a+3] = 0xDD	little	32 bits	0xDDCCBBAA
m[a] = 0xAA; m[a+1] = 0xBB; m[a+2] = 0xCC; m[a+3] = 0xDD	big	8 bits	0x000000AA
m[a] = 0xAA; m[a+1] = 0xBB; m[a+2] = 0xCC; m[a+3] = 0xDD	big	16 bits	0x0000AABB
m[a] = 0xAA; m[a+1] = 0xBB; m[a+2] = 0xCC; m[a+3] = 0xDD	big	32 bits	0xAABBCCDD

Table 3: Register Result of a (Signed) Load Instruction

Memory Content	Endian Mode	Data Size	Register Value Result of $Load_{size}$(address a)
m[a] = 0xAA; m[a+1] = 0xBB; m[a+2] = 0xCC; m[a+3] = 0xDD	little	8 bits	0xFFFFFFAA
m[a] = 0xAA; m[a+1] = 0xBB; m[a+2] = 0xCC; m[a+3] = 0xDD	little	16 bits	0xFFFFBBAA
m[a] = 0xAA; m[a+1] = 0xBB; m[a+2] = 0xCC; m[a+3] = 0xDD	little	32 bits	0xDDCCBBAA
m[a] = 0xAA; m[a+1] = 0xBB; m[a+2] = 0xCC; m[a+3] = 0xDD	big	8 bits	0xFFFFFFAA
m[a] = 0xAA; m[a+1] = 0xBB; m[a+2] = 0xCC; m[a+3] = 0xDD	big	16 bits	0xFFFFAABB
m[a] = 0xAA; m[a+1] = 0xBB; m[a+2] = 0xCC; m[a+3] = 0xDD	big	32 bits	0xAABBCCDD

An interesting example is the small C program below that determines whether the program runs on a big-endian or little-endian mode machine.

```
int w = 0x04050607;
char *a = (char *) &w;
if (*a == 0x04) printf("big-endian"); else printf("little-endian");
```

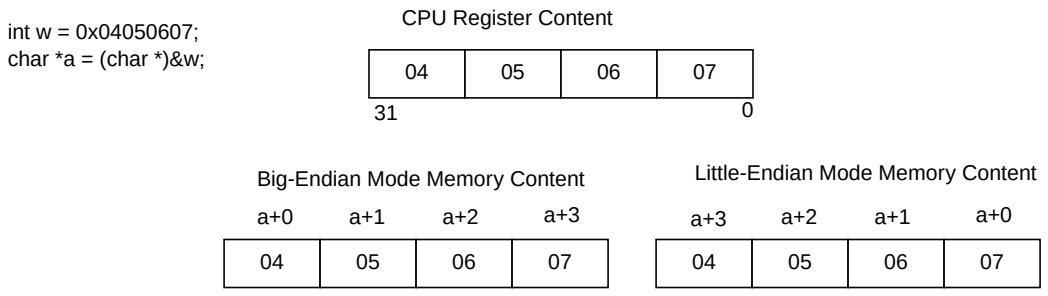


Figure 4: Memory Content Created by the C Program

3.2 Law 2: The “DMA Convention Rule”

The DMA convention rule says that “when a stream of items enters the system, items should be placed in memory such that an item that arrived later has a higher address value.” On output, a similar convention holds—items sent first are those with the lowest addresses.

A variant of this rule relates to the storage of images. Pixels from left to right have increasing addresses. Lines from top to bottom have increasing addresses.

This is a convention that keeps programmers sane. It may also be seen as arbitrary, but obviously the best choice between two alternatives.

A more precise version of this rule is:

If item ’0’ of a DMA item stream is placed at address “A,” item “i” of a DMA stream should be placed at byte address “A+i*s,” where “s” is the item size in bytes.

For an example of this rule, refer to Section 5.

4. PNX15xx Series Endian Mode Architecture Details

The programmer's view of the PNX15xx Series endian architecture is as follows:

- The CPU and the modules on the PNX15xx Series store and retrieve audio samples, image pixels and data observing both the CPU rule and DMA rule.
- The system as a whole runs in either little-endian or big-endian mode.
- The mode is determined by the "BIG_ENDIAN" bit in the SYS_ENDIANMODE register, see Chapter 3 System On Chip Resources Section 3.3 on page 3-8, which is "exported" to other modules.
- The value of this bit is set during system initialization.

4.1 Global Endian Mode

The CPU and all the modules always operate in a single endian mode. This endian mode is determined by the BIG_ENDIAN bit in the SYS_ENDIANMODE register of the PNX15xx Series Global register module. The value of this bit is set during system boot and normally not changed afterward.

Remark: The TM32 CPU core endian mode is determined by a bit in its PCSW. This is historically set by the "crt0.s" software module on the TM32 CPU core, which initializes the PCSW. The PNX15xx Series version of this software module is responsible for reading the SYS_ENDIANMODE.BIG_ENDIAN bit value and establishing the same TM32 CPU core endian mode as the rest of the system.

4.2 Module Control

All the modules have Control and Status registers, accessed by CPU Programmed I/O. In the PNX15xx Series, all programmed I/O happens through Memory Mapped I/O registers. A separate Device Control and Status Bus (DCS Bus) is used for all MMIO programming. A CPU can access Device Control and Status registers by using the correct MMIO address for a module register. In the PNX15xx Series, all module registers are 32 bits wide and may only be accessed through 32-bit load/store operations.

A control/status register load/store always copies the 32 bits verbatim between a CPU register and the module register. The module's left-most msb (bit 31) ends up in the CPU's left-most msb (bit 31), and the module's right-most lsb (bit 0) ends up in the right-most CPU register bit. This happens regardless of system endian mode settings.

MMIO load and store instructions always see the same bit layout of module MMIO registers, regardless of endian mode. The field and bit layout is precisely as specified in the module register table with bit 31 designating the msb and bit 0 the lsb.

Remark: The packing and ordering of packed bit structure fields in C compilers are not precisely defined. Typically, big-endian C compilers pack fields from left (msb) to right (lsb). Little-endian C compilers pack from right to left. Because of this and also because of inherent inefficient code when accessing structure fields, it is not recommended to use C structure declarations to access MMIO register fields.

4.3 Module DMA

Every DMA capable module in PNX15xx Series observes the system big-endian signal, and therefore the global SYS_ENDIANMODE.BIG_ENDIAN value, to determine how to write each data item or unit to memory.

An example of a unit would be:

- 16-bit audio sample
- 32-bit audio sample
- 32-bit unit containing a RGBa 8888 true color pixel with alpha value.

The module performs byte swapping within units as needed, and packs units as needed for transmission across on-chip buses. Byte swapping is done in such a fashion that 8, 16 and 32-bit units always end up in memory bytes in the form prescribed by the CPU rule. Successive item packing are placed in incrementing addresses, as designated by the DMA rule.

4.4 SIMD Programming Issues

The module DMA hardware architecture ensures that software dealing with loads and stores of unit size data can be written in a way that is oblivious to the endian mode.

With the current TM32 CPU core, this is not possible for software that performs Single Instruction Multiple Data (SIMD) style programming using multimedia operations. Consider the case of a FIR filter, operating on 16-bit sample units, but using 2-at-a-time load/store/multiply operations. Which of the two 16-bit halfwords is the earlier sample?

- For big-endian mode, the msb halfword contains the earlier sample.
- For little-endian mode, the lsb halfword contains the earlier sample.

The current TM32 CPU core on PNX15xx Series requires that SIMD software be written aware of endian mode.

4.5 Optional Endian Mode Override

Some PNX15xx Series DMA modules have bits in a control register that allow override of the global endian mode. Refer to each module for details. This method is used only in modules that deal with DMA of data of a single, fixed size (in a given mode). Such modules implement a field that allows selection of the following modes:

- Normal mode (reset default), obey global PNX15xx Series endian mode
- Explicit little-endian, unswapped
- Swap over 16 bits
- Swap over 32 bits

5. Example: Audio In—Programmer's View

The PNX15xx Series Audio In module receives mono or stereo, 8 or 16-bit/sample audio data and fills memory with an 8 or 16-bit data structure. The data structure is put in memory according to both endian mode laws.

A programmer's view of the Audio In function is sketched in Figure 5. The programmer sees the address of each item (according to the DMA rule), and expects that 16-bit values are correctly stored in memory according to the CPU law. The DMA logic of the Audio In module therefore needs to write data in memory (byte) locations *precisely,* per Table 4. Note the programmer also sees the Control and Status registers of the Audio In module per Figure 6. These registers are always seen with the same bit-layout in the CPU register, regardless of endian mode.

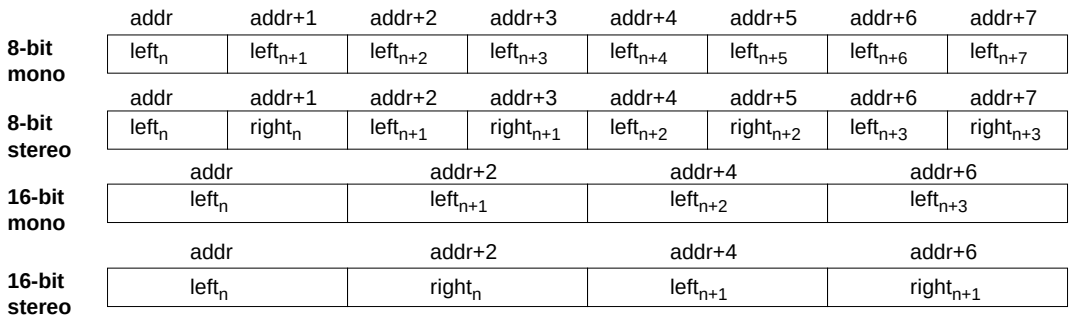


Figure 5: Audio In Memory Data Structure (Programmer's View)

Table 4: Precise Mapping Audio In Sample Time and Bits to Memory Bytes

Operating Mode	m[addr]	m[addr+1]	m[addr+2]	m[addr+3]
8-bit mono—little-endian or big-endian	left$_n$[7:0]	left$_{n+1}$[7:0]	left$_{n+2}$[7:0]	left$_{n+3}$[7:0]
8-bit stereo—little-endian or big-endian	left$_n$[7:0]	right$_n$[7:0]	left$_{n+1}$[7:0]	right$_{n+1}$[7:0]
16-bit mono—little-endian	left$_n$[7:0]	left$_n$[15:8]	left$_{n+1}$[7:0]	left$_{n+1}$[15:8]
16-bit mono—big-endian	left$_n$[15:8]	left$_n$[7:0]	left$_{n+1}$[15:8]	left$_{n+1}$[7:0]
16-bit stereo—little-endian	left$_n$[7:0]	left$_n$[15:8]	right$_n$[7:0]	right$_n$[15:8]
16-bit stereo—big-endian	left$_n$[15:8]	left$_n$[7:0]	right$_n$[15:8]	right$_n$[7:0]

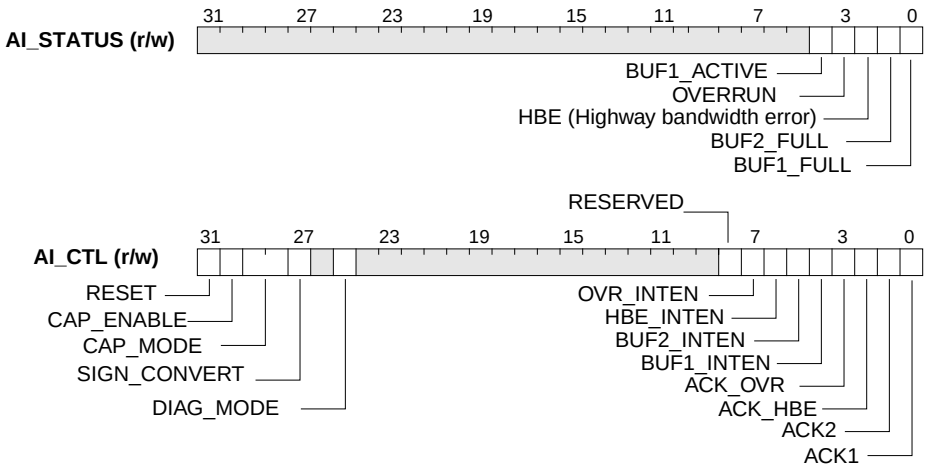


Figure 6: Audio In Control/Status MMIO Registers

6. Implementation Details

The PNX15xx Series system has two different bus structures:

- Device Control and Status Network, or DCS Network; and
- Pipelined Memory Access Network, or PMAN Network, or MTL bus.

6.1 PMAN Network Endian Block Diagram

The system endian mode of operation is designated to each component by the system big-endian signal - ’1’ for big-endian mode, ’0’ for little-endian mode.

Referring to Figure 1, note that both modules are identical. They perform all DMA data transfers across a standard 32-bit “DTL interface.” Module 1 uses the data ordering rules according to Table 5 while Module 2 uses the address invariance rules according to Table 6. The PMAN interface to Module 1 therefore has to convert the data format to address invariant format (this is done in the "Endian Swap" portion of the PMAN structure) while Module 2 does not require any such conversion (the module is already in the address invariant format). The rest of the PMAN and memory interface structure is address invariant.

The PMAN Endian Swap unit (as shown for Module 1) or the Module endian swap unit (as shown in Module 2), must deal with unit endian swapping and unit packing. Swapping is defined as “positioning each byte of a unit correctly with respect to the memory byte address that it is supposed to go to.” Swapping is what implements the CPU rule. Packing is defined as “the action that places consecutive units simultaneously on a wider bus in order to implement the DMA rule.”

The DMA module need not be aware of the details of either the DTL, or the MTL Bus. It just swaps and packs, based on its knowledge of unit size and system endian mode, and creates the valid DTL interface data.

Two standard solutions are provided to interface the DTL to the PMAN MTL network buses:

- The “PMAN Buffer” connects the 32-bit DTL interface to the MTL-Bus.
- The “packer Lego” and “DMA Buffering” map the 32-bit DTL interface to the 64-bit MTL Memory Bus.

Note that the connection from the 32-bit DTL interface to the MTL Bus uses swapping only if the 32-bit DTL interface is not address invariant.

The following subsections show how unit data of different lengths travels across the three key interfaces: the 32-bit DTL interface, the DCS Network and the MTL Memory Bus.

6.2 DMA Across a DTL Interface

Modules that interface to the DTL bus can deal with data on the bus in two ways:

- Data can be put on the bus in their natural form (without flipping them for endian mode but indicating the size of the data with *cmd_data_size*). In this case, the module is not aware of the SYS_ENDIAN bit in the system. The module follows DTL data ordering rules.
- Modules can put the data on the bus in the address invariant mode (Note: This is also the 8-bit mode). In this mode, the module uses the SYS_ENDIAN bit to flip the data appropriately depending on the size of the data and the system endian mode.

6.2.1 DTL Data Ordering Rules

Data is transferred across the DTL Interface by the following rules:

- The value of *cmd_data_size* indicates the width of the data item(s) as 8 * $2^{cmd\_data\_size}$ bits. For example, with 8-bit data, *cmd_data_size* is set to 0x0, for 16-bit data, *cmd_data_size* is 0x1, etc.
- In all cases, the data item (e.g. Byte) that corresponds to the lowest address is transferred on the low order data bits of the DTL rd_data or wr_data

Modules dealing with 8, 16 or 32-bit units must place bytes on the DTL interface given in Table 7. The Endian Swap Units then convert the data into true address invariant views based on the *cmd_data_size*.

Table 5: DTL Interface Rules

Module Item Unit Size	System Endian Mode	DTL_D[31:24]	DTL_D[23:16]	DTL_D[15:8]	DTL_D[7:0]
8 bits	either	item #4 with address a+3	item #3 with address a+2	item #2 with address a+1	item #1 with address a
16 bits	either	item #2 with address a+2		item #1 with address a	
		bits 15..8	bits 7..0	bits 15..8	bits 7..0
32 bits	either	item with address a			
		bits 31..24	bits 23..16	bits 15..8	bits 7..0

6.2.2 Address Invariant Data Ordering Rules

The address invariance rule of the DTL interface is given in Table 6. A given byte lane implies the address, regardless of endian mode of the system.

Table 6: 32 Bit DTL Interface Byte Address

DTL-D[31:24]	DTL-D[23:16]	DTL-D[15:8]	DTL-D[7:0]
4n+3	4n+2	4n+1	4n+0

Modules dealing with 8, 16 or 32-bit units must place bytes on the DTL interface given in Table 7.

Table 7: DTL Interface Rules

Module Item Unit Size	System Endian Mode	DTL_D[31:24]	DTL_D[23:16]	DTL_D[15:8]	DTL_D[7:0]
8 bits	either	item #4 with address a+3	item #3 with address a+2	item #2 with address a+1	item #1 with address a
16 bits	big	item #2 with address a+2		item #1 with address a	
		bits 7..0	bits 15..8	bits 7..0	bits 15..8
16 bits	little	item #2 with address a+2		item #1 with address a	
		bits 15..8	bits 7..0	bits 15..8	bits 7..0
32 bits	big	item with address a			
		bits 7..0	bits 15..8	bits 23..16	bits 31..24
32 bits	little	item with address a			
		bits 31..24	bits 23..16	bits 15..8	bits 7..0

6.3 Data Transfers Across the DCS Network

The DCS Network is said to be “endian neutral”. Although this bus is intended to transfer mainly 32-bits of control and status data, it can carry smaller units of data (8-bit, 16-bit for e.g., CPU accesses to PCI devices). Accesses on this bus are mainly MMIO accesses. The only memory accesses allowed are for booting purposes via the DCS Gate module. This is only accessed by the boot module. The DCS Network is not designed for DMA burst transfers.

Modules that transfer smaller data items (8- or 16-bit) must observe the packing rules in Table 8.

Table 8: DCS Network Data Transfer Rules (32 Bits at-a-time Transfer)

Module Item Unit Size	System Endian Mode	DCS_D[31:24]	DCS_D[23:16]	DCS_D[15:8]	DCS_D[7:0]
8 bits	big	item #1 with address a	item #2 with address a+1	item #3 with address a+2	item #4 with address a+3
8 bits	little	item #4 with address a+3	item #3 with address a+2	item #2 with address a+1	item #1 with address a

Table 8: DCS Network Data Transfer Rules (32 Bits at-a-time Transfer) *...Continued*

Module Item Unit Size	System Endian Mode	DCS_D[31:24]	DCS_D[23:16]	DCS_D[15:8]	DCS_D[7:0]
16 bits	big	item #1 with address a bit15.....bit0		item #2 with address a+2 bit15.....bit0	
16 bits	little	item #2 with address a+2 bit15.....bit0		item #1 with address a bit15.....bit0	
32 bits	either	item (address a) bit31.....bit0			

6.4 DMA Across the MTL Bus

The 64-bit MTL bus associates byte-addresses with byte lanes in a fixed manner, independent of system endian mode, according to Table 9.

Table 9: MTL Memory Bus Byte Address

Data[63:56]	Data[55:48]	Data[47:40]	Data[39:32]	Data[31:24]	Data[23:16]	Data[15:8]	Data[7:0]
8n+7	8n+6	8n+5	8n+4	8n+3	8n+2	8n+1	8n+0

Modules that directly place data in or retrieve data from memory are DMA modules. High bandwidth modules, or modules that require low latency access to memory, perform DMA over the MTL Bus.

DMA modules use large (64-byte or larger) block transfers using the 8-byte lanes of the MTL Bus within the Hub.

Modules that deal with 8, 16 or 32-bit item data sizes and connect directly to the MTL Bus must follow the rules in Table 10. Note in particular the bit numbers of the 16 and 32-bit data items in big-endian modes. Modules that use the DTL interface can rely on the standard packer and DMA Endian Swap Units to accomplish the MTL Bus rules.

Table 10: MTL Memory Bus Item DMA Rules

Module Item Unit Size	System Endian Mode	Data [63:56]	Data [55:48]	Data [47:40]	Data [39:32]	Data [31:24]	Data [23:16]	Data [15:8]	Data [7:0]
8 bits	either	item # 8 addr a+7	item #7 addr a+6	item #6 addr a+5	item #5 addr a+4	item #4 addr a+3	item #3 addr a+2	item #2 addr a+1	item #1 addr a
16 bits	big	item #4 address a+6 b7...b0 b15...b8		item #3 address a+4 b7...b0 b15...b8		item #2 address a+2 b7...b0 b15...b8		item #1 address a b7...b0 b15...b8	
16 bits	little	item #4 address a+6 b15...b8 b7...b0		item #3 address a+4 b15...b8 b...b0		item #2 address a+2 b15...b8 b7...b0		item #1 address a b15...b8 b7...b0	
32 bits	big	item #2 address a+4 b7...b0 b15...b8 b23...b16 b31...b24				item #1 address a b7..b0 b15...b8 b23...b16 b31...b24			
32 bits	little	item #2 address a+4 b31...b24 b23...b16 b15...b8 b7...b0				item #1 address a b31...b24 b23...b16 b15...b8 b7...b0			

6.5 DTL-to-MTL Adapters

The DTL-to-MTL adaptor translates DTL-Bus initiated read and write transactions to MTL Memory transactions. The DTL interface can either be an address invariant (See Section 6.2.2 on page 29-12) or follow DTL data ordering rules (See Section 6.2.1 on page 29-11).

For DTL interfaces that are address invariant, the translation is performed in a byte-address invariant way, i.e. the byte address associated with every 8-bit quantity must be equal on each side of the bridge. Note that this translation need not be aware of the unit size being transported. The module that was the originator of units has performed swapping and packing such that each byte has been given the correct byte address.

For DTL interfaces that follow DTL data ordering rules, the translation takes into account the data unit size on the DTL and the system endian mode, and converts the data into an address invariant view on the MTL interface. The Module that was the originator of units need not perform swapping and packing such that each byte has been given the correct byte address (The IP need not be aware of the System Endianmode signal).

The translation occurs in two steps. When writing data to memory, the first step flips the 32-bit DTL-Bus writes to a 32-bit address invariant view (depending on endian mode and unit data size). The second step performs packing from this 32-bit address-invariant data to the 64-bit MTL Memory Bus. For modules that read memory data, the two steps occur in the reverse direction.

The MTL-Bus associates addresses with each byte transferred, depending on the DTL-Bus unit data size and the setting of the big-endian signal. This byte address is given in Table 8, where the value “A” denotes the integer value on PI-Bus A[n:2] address wires.

The DTL interface can either follow the address invariance rules (as indicated in Table 7) or follow DTL Data ordering rules (as indicated in Table 5).

6.6 PCI Interface

The PCI interface on the PNX15xx Series connects to the off-chip PCI-bus, the DCS Network and the MTL Memory Bus. As with any bridge, the PCI interface must maintain the byte address of any byte of a transaction on all sides of the bridge.

The PCI interface bridges the following transactions in the PNX15xx Series:

- PCI master read/writes from/to PNX15xx Series MMIO registers using 32-bit transactions only
- PCI master read/writes from/to PNX15xx Series SDRAM
- DCS Network Master initiated read/writes from/to PCI targets
- PCI interface internal DMA transactions, where the source can be a PCI target or DRAM, and the destination is a PCI target or DRAM.

The 32-bit PCI bus uses byte address conventions identical to the DTL interface and MTL Memory Bus Interface: Refer to Table 11.

Table 11: 32 Bit PCI Interface Byte Address

PCI-AD[31:24]	PCI-AD[23:16]	PCI-AD[15:8]	PCI-AD[7:0]
4n+3	4n+2	4n+1	4n+0

The DCS Network uses byte address conventions given in Table 8.

The MTL Memory Bus uses the byte address conventions given in Table 9.

With the above byte address conventions on the three sides of the bridge and the byte address invariance rule for bridges, the swap modes can be derived. Since the convention on the PCI bus closely matches those on the MTL Bus.

7. Detailed Example

This section describes all steps involved in how a big-endian mode external CPU (e.g., a Power Macintosh), paints an RGB-565 pixel format frame buffer in the PNX15xx Series SDRAM and how this is displayed on the QVCP. This example illustrates the following:

- The Power Macintosh PCI bridge and its address invariance rule based swapper
- The BIG-endian PCI pixel transfer
- How data arrives correct in SDRAM in native RGB565 pixel format
- How the QVCP takes it and displays it
- How the TM32 CPU core sees the data

The Power Macintosh was the first platform that successfully demonstrated big-endian operations across the PCI bus. Details of how this works can be found in the Apple document “Designing PCI Cards and Drivers for Power MacIntosh Computers.”

Suppose that the big-endian CPU in the Power Macintosh uses a 32-bit store operation to create two RGB565 pixels. Pixel 1, the left-most pixel, has (byte) address “A” and pixel 2 has address “A+2.” Since these two pixels are transferred in a single 32-bit word, “A” is a multiple of 4.

The intermediate stages that the data goes through can be found in Figure 7

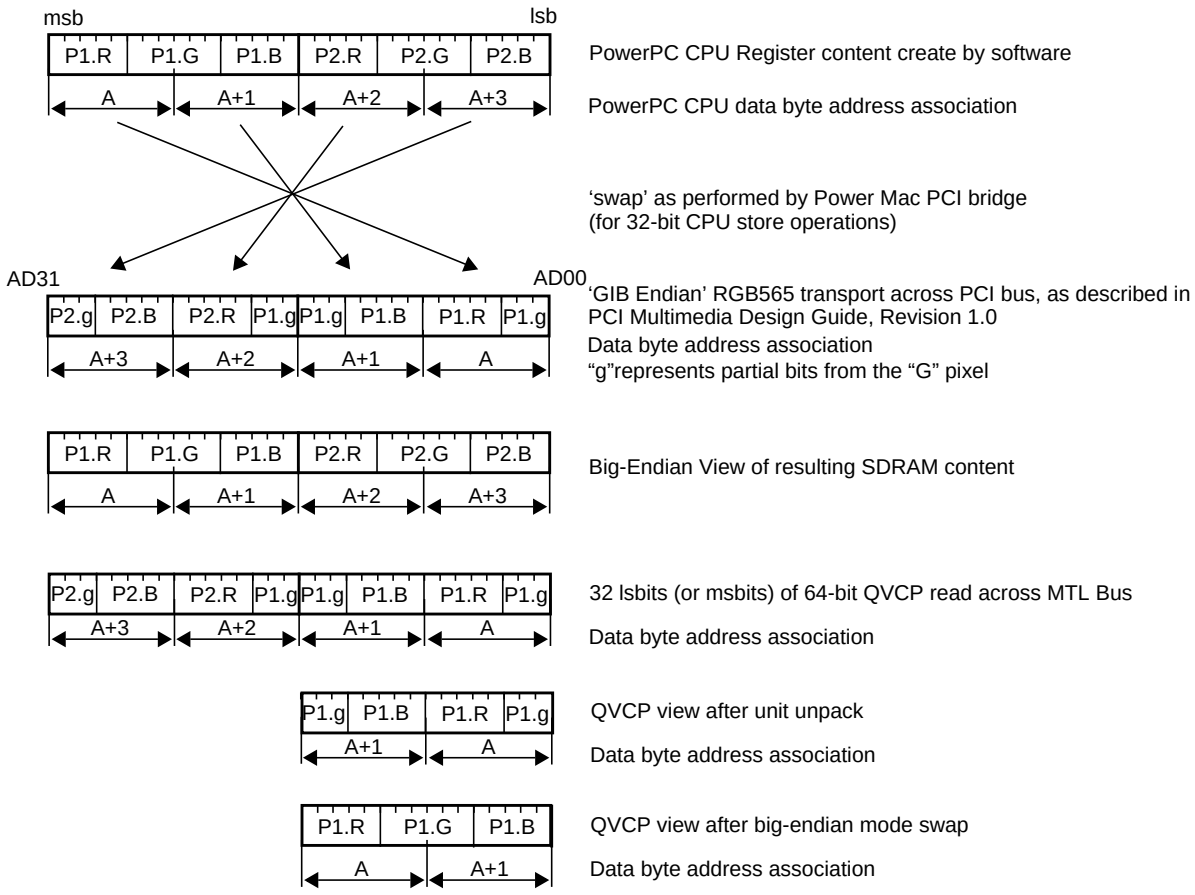


Figure 7: Big-Endian External CPU Drawing Two RGB-565 Pixels

Note that the Power Macintosh architecture contains a PCI bridge that maintains byte address invariance. Since all stages inside the PNX15xx Series maintain byte addresses, the end-to-end result of the complex sequence of actions is a successfully rendered pair of RGB565 pixels.

It is recommended to use only external big-endian CPU/PCI bridge combinations that implement the Power Macintosh style byte-invariant address model with the PNX15xx Series. Some external CPU PCI bridges may only contain a static, transaction-size CPU unaware swapper. The use of such external components is not recommended and will require special care in software.

Chapter 30: DCS Network

PNX15xx Series Data Book – Volume 1 of 1

Rev. 2 — 1 December 2004 Product data sheet

1. Introduction

The PNX15xx Series Device Control and Status (DCS) Network architecture is designed to support the following:

- Separates MMIO traffic from DMA traffic:
  - The TM3260 core has a high-performance, low-latency path to memory. TM3260 ICache and DCache traffic is separated.
- Provides low latency access to modules:
  - The TM3260 core has a low latency access to different modules of PNX15xx Series system.
- Supports timeout generation.

2. Functional Description

The DCS bus is intended for MMIO traffic to configure the various modules in the system. All modules can be accessed by the Boot module, and external PCI master or the TM3260 itself. Access between the DCS bus and the memory is provided by the DCS-Gate. This path is to be used by the boot module only! Figure 3 on page 3-30 in Chapter 3 System On Chip Resources pictures PNX15xx Series DCS bus.

A generic “Device Transaction Level” (DTL) point-to-point initiator-target communication protocol is used on the boundary between a module and the DCS bus. MMIO communication through the DTL protocol always consists of a single 32-bit data element.

Each module on the DCS bus has a unique ID. The bus controller provides programmable timeout generation with the following features:

- Captures error and timeout information:
  - Initiator ID of the currently granted DCS Initiator
  - 32-bit address of the currently granted DCS transaction
  - Encoded Target number for the currently selected DCS device
  - Additional information including read or write command information
- Allows interrupt generation for any non-masked timeouts and errors.

The DCS network controllers generate selects for all the DCS targets according to the address on the DCS network aperture map. See Chapter 3 System On Chip Resources, Section 11. on page 3-31 for addresses of DCS target apertures.

In the total aperture range there are holes, a.k.a. “Null” modules, between the different MMIO targets specified by noncontiguous offsets. Each hole is considered a null target. When an offset of 0xffc within each hole is addressed, the controller will respond with a module ID and the size of the region.

2.1 Error Generation

Error capture registers inside the network controller will capture the current address and operation that was in progress when an error is reported or when a timeout occurs. Once an error has been captured, the capture registers are no longer updated until the interrupt is cleared.

Errors caused by TM3260 32-bit read operations are *not captured*, and *not reported* to the TM3260. Therefore, when the currently selected initiator is the TM3260 (as determined by the arbiter), if the operation is a read, and the mask is all ones, any errors are blocked, *including timeout errors*. In this case, the capture registers are not updated and error signals are not asserted. This is to prevent errors on speculative loads.

2.2 Interrupt Generation

The DCS network controller generate an interrupt for:

- Error acknowledge detected on the DCS network
- DCS Network timeout

These interrupts can be enabled, cleared, software set and status seen by accessing registers on top of the DCS network controllers MMIO space.

2.3 Programmable Timeout

The timeout block uses a 17-bit counter to count clock cycles of an active transaction. The counter increments when the select signal (sel) is high. The counter synchronously resets to zero when sel is low.

When the timeout counter reaches a certain value determined by the control register BC_CTRL, the counter stops incrementing and the abort_all signal is sent to the currently selected target. Each timeout limit is a number equal to 2n-1, allowing the timeout detection circuit to be a simple mux which selects one bit from the timeout counter. Note that the three least significant bits of the counter are not monitored, because no transaction can complete in less than four clock cycles.

When a timeout occurs, the abort signal is asserted to the currently selected target. The timeout condition is also logically ORed with the DCS error input to force an error indication back to the target.

2.3.1 Arbitration

A “round robin” arbiter is used to selective grant access to each of the requesting initiators. The arbiter will default grant the last device that was granted. Therefore, when no initiator is requesting access, the default grant will be given to the initiator

that most recently performed a transaction. The initiator with a default grant can access a target one clock cycle faster than an initiator without the default grant. Assigning the default grant to the initiator that most recently used the “bus” is expected to yield the highest performance, since one initiator is likely to execute several transactions at once.

To achieve the round robin feature with a dynamic default grant, the arbiter uses an internal priority comparator. The comparator selects an initiator to grant by comparing the “priorities” of each device. The priority value consists of three bits. The most significant bit is high when there is a pending request from that initiator. The second most significant bit is generated by “last_grant”, which will be high for the most recently granted initiator *only if that initiator does not have a pending request*, and low for all other initiators. The third bit is a “uniform scheduling” value. This will be set to one for all bit locations higher than the last granted initiator and zero for all lower ones. The priority block will pick the highest value (3-bit) input. In the case of a tie, the lower numbered port will win.

2.4 Endian Mode

All DCS network ports use 32-bit data paths and the data values are viewed as 32-bit quantities. Even when an 8 or 16-bit read or write is performed, the transfer is considered to be a portion of a larger 32-bit quantity. The data transfers are never viewed as packed 8 or 16-bit values.

3. Register Descriptions

3.1 Register Summary

Table 1 summarizes the control and status registers visible inside the DCS Controller.

Table 1: DCS Controller_TriMedia Configuration Register Summary

Offset	Symbol	Description
0x10 3000	BC_CTRL	Timeout control register
0x10 300C	BC_ADDR	Error and timeout address register
0x10 3010	BC_STAT	Error and timeout status register
0x10 30D8	BC_INT_CLR_ENABLE	Clear bits in BC_INT_EN
0x10 30DC	BC_INT_SET_ENABLE	Set bits in BC_INT_EN
0x10 3FE0	BC_INT_STATUS	Interrupt Status register
0x10 3FE4	BC_INT_EN	Interrupt Enable register
0x10 3FE8	BC_INT_CLR	Interrupt Clear register
0x10 3FEC	BC_INT_SET	Interrupt software set register
0x10 3FFC	BC_MOD_ID	Module identification and revision information

Remark: The BC_INT_EN register is R/W, however newly written software drivers should consider BC_INT_EN as read only and should use the BC_INT_CLR_ENABLE and BC_INT_SET_ENABLE registers to update the value of BC_INT_EN.

3.2 Register Tables

Table 2 shows detailed bit locations for each register.

Table 2: DCS Controller_TriMedia Configuration Registers (Rev 0.32)

Bit	Symbol	Access	Value	Description
Offset 0x10 3000 BC_CTRL				
31:5	Reserved		-	Ignore upon read. Write as zeros
4:1	TOUT_SEL[3:0]	R/W	0x0	Timeout select 0x0 = Timeout generated after 7 consecutive wait cycles. 0x1 = Timeout generated after 7 consecutive wait cycles. 0x2 = Timeout generated after 7 consecutive wait cycles. 0x3 = Timeout generated after 15 consecutive wait cycles. 0x4 = Timeout generated after 31 consecutive wait cycles. 0x5 = Timeout generated after 63 consecutive wait cycles. 0x6 = Timeout generated after 127 consecutive wait cycles. 0x7 = Timeout generated after 255 consecutive wait cycles. 0x8 = Timeout generated after 511 consecutive wait cycles. 0x9 = Timeout generated after 1,023 consecutive wait cycles. 0xa = Timeout generated after 2,047 consecutive wait cycles. 0xb = Timeout generated after 4,095 consecutive wait cycles. 0xc = Timeout generated after 8,191 consecutive wait cycles. 0xd = Timeout generated after 16,383 consecutive wait cycles. 0xe = Timeout generated after 32,767 consecutive wait cycles. 0xf = Timeout generated after 65,535 consecutive wait cycles
0	TOUT_OFF	R/W	0x1	Timeout disable 0x0 = Timeout enabled. 0x1 = Timeout disabled.
Offset 0x10 300C BC_ADDR				
31:2	ERR_TOUT_ADDR[31:2]	R	0x00000000-	Full 30 bits of the address which causes an error or timeout.
1:0	Reserved		-	Ignore upon read. Write as zeroes.
Offset 0x10 3010 BC_STAT				
31:29	Reserved		-	Ignore upon read. Write as zeroes.
28:24	ERR_TOUT_GNT[4:0]	R	0x0	Active initiator causing error or timeout -- Initiator ID of the current pending transaction will be captured: 1: TM3260 2: BOOT 4: PCI
23:17	Reserved			Ignore upon read. Write as zeroes.

Table 2: DCS Controller_TriMedia Configuration Registers (Rev 0.32) *…Continued*

Bit	Symbol	Access	Value	Description
16:10	ERR_TOUT_SEL[6:0]	R	0x00	Selected agent during error or timeout 7'b 0000000 = PCI 7'b 0000001 = I^2C 7'b 0000010 = CLOCKs 7'b 0000011 = 2DDE 7'b 0000100 = RESET 7'b 0000101 = TMDBG 7'b 0000110 = SYSTEM REGISTERS 7'b 0000111 = MTL ARBITER, a.k.a. IP1010 7'b 0001000 = DDR CONTROLLER, a.k.a. IP2031 7'b 0001001 = FGPI 7'b 0001010 = FGPO 7'b 0001011 = LAN100 7'b 0001100 = LCD 7'b 0001101 = VLD 7'b 0001110 = TM3260 7'b 0001111 = GPIO 7'b 0010000 = VIP 7'b 0010001 = SPDO 7'b 0010010 = SPDI 7'b 0010011 = DVDD 7'b 0010100 = MBS 7'b 0010101 = QVCP 7'b 0010110 = AO 7'b 0010111 = AI 7'b 0011000 = PCI1 Aperture 7'b 0011001 = PCI2 Aperture 7'b 0011010 = XIO Aperture 7'b 0011011 = DMA (TM3260 to PCI space) 7'b 1011100 = Null/Error Target 7'b 1011101 = Network Controller Configuration Aperture
9	Reserved		-	Ignore upon read. Write as zeroes.
8	ERR_TOUT_READ	R	0x0	Value of cmd_read signal during error or timeout 1 = Read operation 0 = Write operation
7:4	ERR_TOUT_MASK[3:0]	R	0x0	Value of cmd_mask during error or timeout Indicates which bytes were to be read or written.
3:2	Reserved		-	Ignore upon read. Write as zeroes.
1	ERR_ACK	R	0	Error or Timeout 0 = Timeout 1 = Error
0	Reserved		-	Ignore upon read. Write as zeroes.
Offset 0x10 3FD8 BC_INT_CLR_ENABLE				
31:2	Reserved		-	Ignore upon read. Write as zeroes.
1	INT_CLR_ENABLE_TOUT	W	0	Timeout interrupt enable clear register. This is written by software to clear the interrupt enable (bit 1 of BC_INT_EN). 1 = Timeout interrupt enable is cleared 0 = Timeout interrupt enable is unchanged.

Table 2: DCS Controller_TriMedia Configuration Registers (Rev 0.32) *…Continued*

Bit	Symbol	Access	Value	Description
0	INT_CLR_ENABLE_ERROR	W	0	Error interrupt enable clear register. This is written by software to clear the interrupt enable (bit 0 of BC_INT_EN). 1 = Error interrupt enable is cleared 0 = Error interrupt enable is unchanged.
Offset 0x10 3FDC	***BC_INT_SET_ENABLE***			
31:2	Reserved		-	Ignore upon read. Write as zeroes.
1	INT_SET_ENABLE_TOUT	W	0	Timeout interrupt enable set register. This is written by software to set the interrupt enable (bit 1 of BC_INT_EN). 1 = Timeout interrupt enable is set 0 = Timeout interrupt enable is unchanged.
0	INT_SET_ENABLE_ERROR	W	0	Error interrupt enable set register. This is written by software to set the interrupt enable (bit 0 of BC_INT_EN). 1 = Error interrupt enable is set 0 = Error interrupt enable is unchanged.
Offset 0x10 3FE0	***BC_INT_STATUS***			
31:2	Reserved		-	Ignore upon read. Write as zeroes.
1	INT_STATUS_TOUT	R	0	Timeout interrupt status. Reports any pending timeout interrupts: 1 = Timeout interrupt pending: MMIO bus controller has generated a timeout because a target has violated the programmable limit for timeout (see BC_TOUT). 0 = Timeout interrupt is not pending.
0	INT_STATUS_ERROR	R	0	Error interrupt status. Reports any pending error interrupts: 1 = Error interrupt pending, meaning bus controller has detected an error acknowledge from a target. 0 = Error interrupt is not pending.
Offset 0x10 3FE4	***BC_INT_EN***			
31:2	Reserved		-	Ignore upon read. Write as zeroes.
1	INT_ENABLE_TOUT	R/W	0	Timeout interrupt enable register 1 = Timeout interrupt is enabled 0 = Timeout interrupt is disabled.
0	INT_ENABLE_ERROR	R/W	0	Timeout interrupt enable register 1 = Error interrupt is enabled 0 = Error interrupt is disabled.
Offset 0x10 3FE8	***BC_INT_CLR***			
31:2	Reserved		-	Ignore upon read. Write as zeroes.
1	INT_CLEAR_TOUT	W	0	Timeout interrupt clear register. This is written by software to clear the interrupt. 1 = Timeout interrupt is cleared 0 = Timeout interrupt is unchanged.
0	INT_CLEAR_ERROR	W	0	Error interrupt clear register. This is written by software to clear the interrupt. 1 = Error interrupt is cleared 0 = Error interrupt is unchanged.

Table 2: DCS Controller_TriMedia Configuration Registers (Rev 0.32) *…Continued*

Bit	Symbol	Access	Value	Description
Offset 0x10 3FEC	***BC_INT_SET***			
31:2	Reserved		-	Ignore upon read. Write as zeroes.
1	INT_SET_TOUT	W	0	Timeout interrupt set register. Allows software to set interrupts. 1 = Timeout interrupt is set 0 = Timeout interrupt is unchanged.
0	INT_SET_ERROR	W	0	Error interrupt set register. Allows software to set interrupts. 1 = Error interrupt is set 0 = Error interrupt is unchanged.
Offset 0x10 3FFC	***BC_MODULE_ID***			
31:16	MODULE_ID[15:0]	R	0xA049	Unique 16-bit code. This value varies depending on the value specified at compile time for each DCS Controller
15:12	MAJREV[3:0]	R	0x0	Major Revision ID
11:8	MINREV[3:0]	R	0x0	Minor Revision ID
7:0	MODULE_APERTURE_SIZE[7:0]	R	0x00	Aperture size = 4 kB*(bit_value+1), so 0 means 4 kB (the default).

Chapter 31: TM3260

PNX15xx Series Data Book – Volume 1 of 1

Rev. 2 — 1 December 2004 Product data sheet

1. Introduction

Please refer to “The TM3260 Architecture Databook”, Rev. 1.02, July 12 2004, or later revision, for a complete detailed description of the TM3260 architecture. Refer to Chapter 3 System On Chip Resources Section 6.3 on page 3-15 for details on how TM3260 is connected in PNX15xx Series System On Chip device.

PHILIPS

1. Data sheet status

Level	Data sheet status[1]	Product status[2][3]	Definition
I	Objective data	Development	This data sheet contains data from the objective specification for product development. Philips Semiconductors reserves the right to change the specification in any manner without notice.
II	Preliminary data	Qualification	This data sheet contains data from the preliminary specification. Supplementary data will be published at a later date. Philips Semiconductors reserves the right to change the specification without notice, in order to improve the design and supply the best possible product.
III	Product data	Production	This data sheet contains data from the product specification. Philips Semiconductors reserves the right to make changes at any time in order to improve the design, manufacturing and supply. Relevant changes will be communicated via a Customer Product/Process Change Notification (CPCN).

[1] Please consult the most recently issued data sheet before initiating or completing a design.

[2] The product status of the device(s) described in this data sheet may have changed since this data sheet was published. The latest information is available on the Internet at URL http://www.semiconductors.philips.com.

[3] For data sheets describing multiple type numbers, the highest-level product status determines the data sheet status.

2. Definitions

short-form specification — The data in a short-form specification is extracted from a full data sheet with the same type number and title. For detailed information see the relevant data sheet or data handbook.
Limiting values definition — Limiting values given are in accordance with the Absolute Maximum Rating System (IEC 60134). Stress above one or more of the limiting values may cause permanent damage to the device. These are stress ratings only and operation of the device at these or at any other conditions above those given in the Characteristics sections of the specification is not implied. Exposure to limiting values for extended periods may affect device reliability.
Application information — Applications that are described herein for any of these products are for illustrative purposes only. Philips Semiconductors make no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

3. Disclaimers

Life support — These products are not designed for use in life support appliances, devices, or systems where malfunction of these products can reasonably be expected to result in personal injury. Philips Semiconductors customers using or selling these products for use in such applications do so at their own risk and agree to fully indemnify Philips Semiconductors for any damages resulting from such application.
Right to make changes — Philips Semiconductors reserves the right to make changes in the products - including circuits, standard cells, and/or software - described or contained herein in order to improve design and/or performance. When the product is in full production (status 'Production'), relevant changes will be communicated via a Customer Product/Process Change Notification (CPCN). Philips Semiconductors assumes no responsibility or liability for the use of any of these products, conveys no license or title under any patent, copyright, or mask work right to these products, and makes no representations or warranties that these products are free from patent, copyright, or mask work right infringement, unless otherwise specified.

4. Licenses

Purchase of Philips I²C components conveys a license under the Philips' I²C patent to use the components in the I²C system provided the system conforms to the specification defined by Philips. This I²C specification can be ordered using the code 9398 393 40011.

Purchase of Philips RC5 components conveys a license under the Philips RC5 patent to use the components in RC5 system products conforming to the RC5 standard UATM-5000 for allocation of remote control commands defined by Philips.

5. Trademarks

TriMedia — is a trademark owned by Koninklijke Philips Electronics N.V.
Nexperia — is a trademark of Koninklijke Philips Electronics N.V.

6. Contact information

For additional information, please visit **http://www.semiconductors.philips.com**.

For sales office addresses, send e-mail to: **sales.addresses@www.semiconductors.philips.com**. **Fax: +31 40 27 24825**

PHILIPS

Let's make things better.