



Z90T366 ROM and Z90T361 OTP

eZVision 64 KWord Television Controller with OSD

Product Specification

PS005901-1100



This publication is subject to replacement by a later edition. To determine whether a later edition exists, or to request copies of publications, contact

ZiLOG Worldwide Headquarters

910 E. Hamilton Avenue
Campbell, CA 95008
Telephone: 408.558.8500
Fax: 408.558.8300
www.ZiLOG.com

Windows is a registered trademark of Microsoft Corporation.

Document Disclaimer

© 2000 by ZiLOG, Inc. All rights reserved. Information in this publication concerning the devices, applications, or technology described is intended to suggest possible uses and may be superseded. ZiLOG, INC. DOES NOT ASSUME LIABILITY FOR OR PROVIDE A REPRESENTATION OF ACCURACY OF THE INFORMATION, DEVICES, OR TECHNOLOGY DESCRIBED IN THIS DOCUMENT. ZiLOG ALSO DOES NOT ASSUME LIABILITY FOR INTELLECTUAL PROPERTY INFRINGEMENT RELATED IN ANY MANNER TO USE OF INFORMATION, DEVICES, OR TECHNOLOGY DESCRIBED HEREIN OR OTHERWISE. Except with the express written approval ZiLOG, use of information, devices, or technology as critical components of life support systems is not authorized. No licenses or other rights are conveyed, implicitly or otherwise, by this document under any intellectual property rights.



Table of Contents

| | | |
|------|--|----|
| 1 | Overview | 1 |
| 1.1 | Block Diagram | 3 |
| 1.2 | Pin Assignments | 4 |
| 1.3 | Pin Descriptions | 5 |
| 1.4 | Development Tools | 7 |
| 2 | Operation | 8 |
| 2.1 | CPU Description | 8 |
| 2.2 | Memory (ROM and RAM) | 14 |
| 2.3 | Clock Circuit Description | 16 |
| 2.4 | Reset Conditions | 18 |
| 2.5 | Power Management | 20 |
| 2.6 | I/O Port Configurations | 20 |
| 2.7 | Interrupts | 22 |
| 2.8 | Timers | 23 |
| 2.9 | ADC | 24 |
| 2.10 | Pulse Width Modulation | 26 |
| 2.11 | I2C Interface | 27 |
| 2.12 | On-Screen Display (OSD) | 32 |
| 2.13 | Cursor | 39 |
| 2.14 | Color Palette Assignment | 43 |
| 2.15 | Other Functions | 44 |
| 3 | Register Groups | 46 |
| 3.1 | Register Description | 47 |
| 3.2 | Bank0 (I/O Ports, I2C Interface, PLL Frequency, Cursor) Control Registers | 48 |
| 3.3 | Bank1 (Control Registers) | 53 |
| 3.4 | Bank2 (PWM Registers) | 67 |
| 3.5 | Bank3 (On Screen Display [OSD] registers) | 69 |

| | | |
|-----|------------------------------------|-----|
| 4 | Instruction Set | 81 |
| 4.1 | Instruction Summary | 81 |
| 4.2 | Instruction Operands | 84 |
| 4.3 | Instruction Format | 86 |
| 4.4 | Instruction Bit Codes | 87 |
| 4.5 | Instruction Format Examples | 91 |
| 4.6 | Instruction Timing | 97 |
| 4.7 | Instruction Op Codes | 98 |
| 5 | System Design Considerations | 169 |
| 6 | Electrical Characteristics | 171 |
| 6.1 | DC Peripherals | 172 |
| 6.2 | AC Characteristics | 174 |
| 6.3 | ANALOG RGB | 175 |
| 7 | Packaging | 176 |
| 8 | Ordering Information | 178 |

List of Figures

| | | |
|----|--|-----|
| 1 | Block Diagram..... | 3 |
| 2 | 52-Pin SDIP Pinout..... | 4 |
| 3 | Code Development Environment..... | 7 |
| 4 | AR Register Format | 12 |
| 5 | RAM, ROM, and Pointer Architecture | 14 |
| 6 | ROM Map | 15 |
| 7 | RAM Allocation | 16 |
| 8 | Clock Switching Block Diagram (8 -16 MHz) | 17 |
| 9 | 32 KHz Oscillator Recommended Circuit | 17 |
| 10 | Bidirectional Port Pins | 21 |
| 11 | Bidirectional Pins Multiplexed with I2C Port | 21 |
| 12 | Bidirectional Pins Multiplexed with ADC Inputs | 22 |
| 13 | IR Capture Register Block Diagram | 23 |
| 14 | Z90T361/6 ADC Block Diagram | 25 |
| 15 | ADC Data Packing | 26 |
| 16 | Master Mode | 30 |
| 17 | Slave Mode | 31 |
| 18 | Data Flow | 33 |
| 19 | Blank and R, G, B Outputs in Digital Mode | 33 |
| 20 | R, G, and B Outputs in Analog (Palette) Mode | 34 |
| 21 | Character Expansion | 35 |
| 22 | Table Settings | 37 |
| 23 | Cursor | 39 |
| 24 | Programmable Palette Control at AR Register | 43 |
| 25 | IR Capture Register Input | 44 |
| 26 | Loop Filter Pin Configuration | 44 |
| 27 | Pipeline Execution | 97 |
| 28 | System Block Diagram | 170 |
| 29 | Recommended Application Schematics | 175 |
| 30 | 52-Pin SDIP Package Dimensions | 176 |

List of Tables

| | | |
|----|---|----|
| 1 | Z90T366 or Z90T361 Pin Description | 5 |
| 2 | Internal Registers | 9 |
| 3 | Status Register | 9 |
| 4 | Ram Pointer Loop Description | 10 |
| 5 | Additional Control Registers | 13 |
| 6 | Reset Conditions | 18 |
| 7 | ADC Inputs Typical Range | 24 |
| 8 | Master I2C Bus Bit Rates | 27 |
| 9 | Master I2C Bus Interface Commands | 28 |
| 10 | Slave I2C Bus Interface Commands | 29 |
| 11 | Character Expansion Register | 36 |
| 12 | Attribute Assignment | 38 |
| 13 | Cursor Parameters | 40 |
| 14 | Memory Allocation for Cursor Bitmap | 41 |
| 15 | Fixed Palette Color Assignment | 43 |
| 16 | R4(1)<e:d> Settings | 45 |
| 17 | Register Summary | 46 |
| 18 | Bank Assignments | 47 |
| 19 | Register1, Bank0, Cursor Palette | 48 |
| 20 | Register2, Bank0, PLL Frequency Data Register | 48 |
| 21 | Register3, Bank0, I2C Interface Register | 50 |
| 22 | Register5, Bank0, Port 1 Data Register | 51 |
| 23 | Register4, Bank0, Port 0 Data Register | 51 |
| 24 | Register6, Bank0, Port 0 Direction Register | 52 |
| 25 | Register7, Bank0, Port 1 Direction Register | 52 |
| 26 | Register0, Bank1, Clamp Position Register | 53 |
| 27 | Register1, Bank1, Speed Control Register | 55 |
| 28 | Register2, Bank1, WDT/STOP (write only) and 9-bit Counter (read only) Control Register | 58 |
| 29 | Register3, Bank1, Standard Control Register | 58 |
| 30 | Register4, Bank1, ADC Control Register | 60 |
| 31 | Register5, Bank1, Timer Control Register | 61 |
| 32 | Register6, Bank1, Clock Switch Control Register | 63 |
| 33 | Register7, Bank1, Interrupts/WDT/SMR Control Register | 66 |
| 34 | Interrupt Priority | 67 |

| | | |
|----|--|----|
| 35 | Register0–Register5, Bank 2, PWM 1–6 Registers | 67 |
| 36 | Register6, Bank 2, Shadow Control Register | 68 |
| 37 | Register7, Bank 2, CGROM Offset Register | 69 |
| 38 | Register0–Register1, Bank 3, Write Operation, Shift Registers | 70 |
| 39 | Register0–Register2, Bank 3, Read Operation, Character Multiple Registers | 70 |
| 40 | Register2, Bank 3, Attributes Register, Write Operation | 71 |
| 41 | Register3, Bank 3, Write Operation, Attribute Data Register | 73 |
| 42 | Register3, Bank 3, Read Operation, Attributes Register | 73 |
| 43 | Register 3, Bank 3, Display Character Format for Attribute Data Register, OSD Mode Write Operation | 74 |
| 44 | Register 3, Bank 3, Control Character Format for Attribute Data Register, OSD Mode Write Operation | 75 |
| 45 | Register 3, Bank 3, Display Character Format for Attribute Data Register, Write Operation CCD Mode | 76 |
| 46 | Register 3, Bank 3, Control Character Format for Attribute Data Register, CCD Mode, Write Operation | 76 |
| 47 | Register4, Bank 3, OSD Control Register | 77 |
| 48 | Register5, Bank 3, Capture Register, Read Operation | 78 |
| 49 | Register6, Bank 3, Palette Control Register | 79 |
| 50 | Register7, Bank 3, Output Palette Control Register | 80 |
| 51 | Instruction Format Mnemonics | 81 |
| 52 | Accumulator Modification Instructions | 82 |
| 53 | Arithmetic Instructions | 82 |
| 54 | Bit Manipulation Instructions | 82 |
| 55 | Load Instructions | 83 |
| 56 | Logical Instructions | 83 |
| 57 | Program Control Instructions | 83 |
| 58 | Rotate and Shift Instructions | 83 |
| 59 | Instruction Operand Summary | 84 |
| 60 | Instruction Mnemonics/Operands | 85 |
| 61 | Condition Code Bits | 87 |
| 62 | Accumulator Modification Bits | 88 |
| 63 | Flag Modification Bits | 89 |
| 64 | Register Pointer/ Data Pointer Bits | 90 |
| 65 | Register Bits | 90 |
| 66 | General Instruction Format | 93 |
| 67 | Accumulator Modification Format | 93 |
| 68 | Flag Modification Format | 94 |



| | | |
|----|--|-----|
| 69 | Direct Internal Addressing Format | 95 |
| 70 | Short Immediate Addressing Format | 95 |
| 71 | Long Immediate Addressing Format | 96 |
| 72 | Jump and Call Instruction Formats | 96 |
| 73 | Instruction Op Codes | 98 |
| 74 | Instruction Descriptions | 103 |
| 75 | Instruction Format Mnemonics | 103 |
| 76 | Absolute Maximum and Minimum Ratings | 171 |
| 77 | DC Characteristics | 171 |
| 78 | R, G, and B Analog Output | 172 |
| 79 | ADC0/Small Range* | 173 |
| 80 | AC Characteristics | 174 |
| 81 | ADC1-ADC4/Full range | 174 |
| 82 | RGB Voltage Specification | 175 |
| 83 | RGB Time Specification | 175 |
| 84 | Controlling Dimensions | 177 |



Z90T366 ROM and Z90T361 OTP eZVision 64 KWord Television Controller with OSD

1 Overview

The eZVision Z90T366 and Z90T361 are the ROM and one-time programmable (OTP) versions of a Television Controller with On-Screen Display (OSD) that contains 64 KWords of program memory and 1 Kword of RAM.

ZiLOG's eZVision Z90T366 TV controller with On-Screen Display (OSD) is a highly-integrated solution for television design. The Z90T366 boasts a high-speed 16-bit, advanced Digital Signal Processor (DSP) and powerful OSD engine. Flexible and sophisticated, the OSD includes video display attributes, semi-transparency, programmable color palettes, and a hardware cursor for easy user interface. The Z90T366 supports parental control, closed captioning, and Extended Data System (XDS).

The Z90T366 is an ideal choice for mainstream television sets for both PAL and NTSC standards.

The Z90T36x family consists of the following two basic devices:

- the Z90T366 masked ROM
- the Z90T361 One Time Programmable (OTP) device

In addition, Zilog provides a comprehensive development suite for television system developers including an emulator capable of OTP programming, OSD evaluation board, C-compiler, Application Programmer Interface (API), ZiLOG's Developer Studio (ZDS) software, and Graphics User Interface (GUI) OSD screen design tools. These tools enable TV developers to work efficiently and effectively to bring new products to market.

- The eZVision **Z90T361** is the OTP controller used to develop code and prototypes for specific television applications or initial limited production. Program ROM and Character Generation ROM (CGROM) in the Z90T361 are both programmable.
- The eZVision **Z90T366** incorporates the ROM code developed by the customer with the Z90T361. Customer code is masked into both program ROM and CGROM.

The Z90T366 Television Controller with OSD is based on ZiLOG's Z89C00 processor core. The Z89C00 is a 16-bit, fractional, two's complement CMOS Digital Signal Processor (DSP). Most instructions are accomplished in a single clock cycle. This processor features a 24-bit Arithmetic Logic Unit (ALU) and a 24-bit Accumulator. The processor also contains a six-level stack and three vectored interrupts.

The Z90T366 contains 64 KWords of program ROM and 1 KWord of on-chip data RAM. Program ROM space can hold an unlimited number of characters with a 16x16, 16x18, and 16x20 programmable matrix in relocated Character Generation ROM (CGROM), which is only restricted by the available ROM. In addition, the Z90T366 contains four external register banks with eight registers each. Additional Control Registers (AR) are available to control new peripherals like palette banks and memory management.

An internal 24-MHz/2 system clock has a Phase Lock Loop (PLL) driven by an external 32.768-KHz crystal.

Six-channels of 4-bit Analog to Digital Converters (ADC) support the following:

- Analog control front panel buttons
- Audio level input
- Vertical Blank Interval (VBI) data capture

Six Pulse Width Modulator (PWM) outputs allow low-cost digital-to-analog conversion. The PWMs have 8-bit resolution to control video and audio attributes.

A Master/Slave I²C (Inter Integrated Circuit) bus interface provides serial system interconnect to common peripheral functions.

Twenty-four programmable I/O pins provide flexibility for other digital input/output functions.

An IR (InfraRed) remote capture register facilitates reliable remote data capture.

On-chip Horizontal Synchronization (H_{SYNC}) and Vertical Synchronization (V_{SYNC}) circuits generate a video time base (typically used for VCR and set-top applications) in the absence of an available video signal.

Micro-programmable OSD generation logic provides flexibility to tailor OSD features and functions. In addition to normal OSD functions, Closed Caption is supported in accordance with FCC Report and Order on GEN Docket No. 91-1, dated April 12, 1991. Expanded Data Service (XDS) capability is supported as well.

The Z90T366 is packaged in a 52-pin SDIP package.

Figure 1 is a block diagram of the internal structure of the chip. Figure 2 illustrates the pin locations, and Table 1 describes the function of each pin.

1.1 Block Diagram

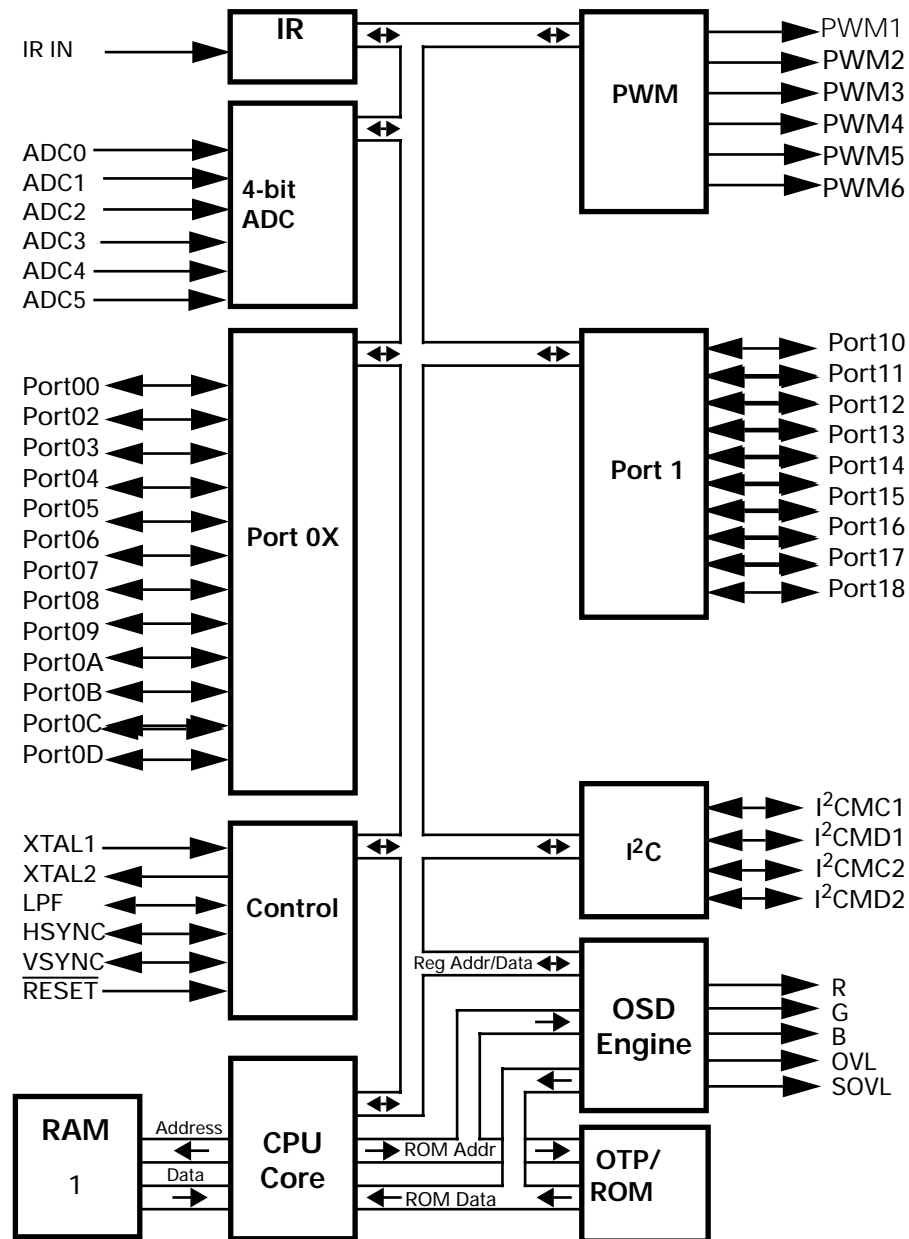


Figure 1 Block Diagram

1.2 Pin Assignments

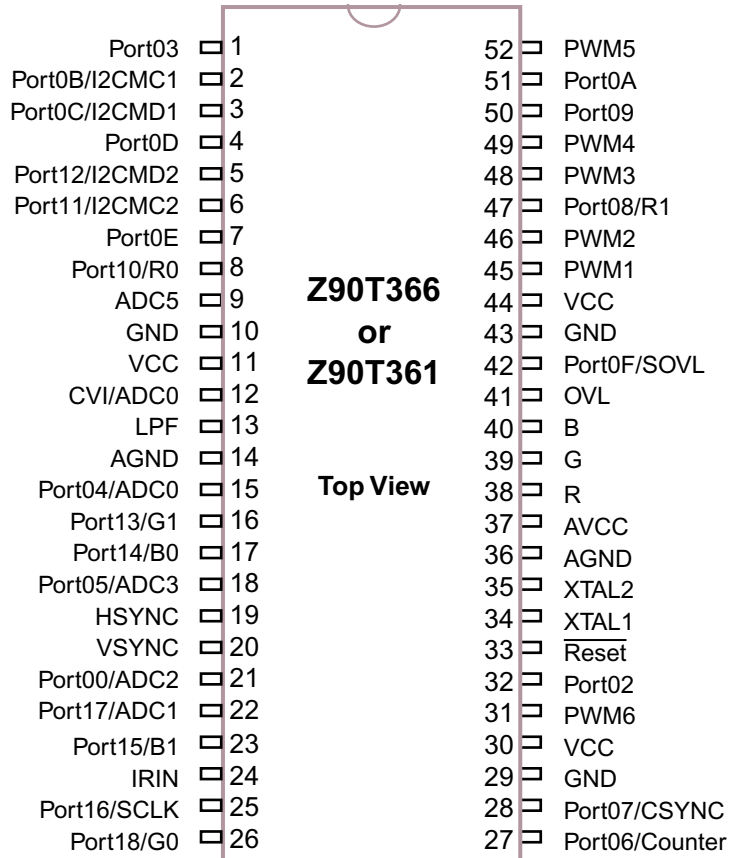


Figure 2 52-Pin SDIP Pinout

For simplicity, both the Z90T366 and Z90T361 will be referred to as the Z90T366.

1.3 Pin Descriptions

Table 1 Z90T366 or Z90T361 Pin Description

| Symbol | Pin # | Function | Direction |
|----------------------------|-------|---|-----------|
| Port03 | 1 | Port 03 | I/O |
| Port0B/I ² CMC1 | 2 | Port 0B or Master1 I ² C clock | I/O |
| Port0C/I ² CMD1 | 3 | Port 0C or Master1 I ² C data | I/O |
| Port0D | 4 | Port 0D | I/O |
| Port12/I ² CMD2 | 5 | Port 12 or Master2 I ² C data | I/O |
| Port11/I ² CMC2 | 6 | Port 11 or Master1 I ² C clock | I/O |
| Port0E | 7 | Port 0E | I/O |
| Port10/R0 | 8 | Port 10 or digital RGB output, red | I/O |
| ADC5 | 9 | ADC5 input | I |
| GND | 10 | Digital ground | Power |
| VCC | 11 | Digital Vcc | Power |
| CVI/ADC0 | 12 | ADC0 input or Composite Video Input | AI |
| LPF | 13 | Loop filter | AI/AO |
| AGND | 14 | Analog ground | Power |
| Port04/ADC0 | 15 | Port 04 or ADC0 input | I/O or AI |
| Port13/G1 | 16 | Port 13 or digital RGB output, green | I/O |
| Port14/B0 | 17 | Port 14 or digital RGB output, blue | I/O |
| Port05/ADC3 | 18 | Port 05 or ADC3 input | I/O or AI |
| HSYNC | 19 | Horizontal sync | I/O |
| VSNC | 20 | Vertical sync | I/O |
| Port00/ADC2 | 21 | Port 00 or ADC2 input | I/O or AI |
| Port17/ADC1 | 22 | Port 17 or ADC1 input | I/O or AI |
| Port15/B1 | 23 | Port 15 or digital RGB output, blue | I/O |
| IRIN | 24 | Infrared remote capture input | I |
| Port16/SCLK | 25 | Port 16 or internal process SCLK | I/O |
| Port18/G0 | 26 | Port 18 or digital RGB output, green | I/O |
| Port06/CNTR | 27 | Port 06 or counter input | I/O |

Table 1 Z90T366 or Z90T361 Pin Description (Continued)

| Symbol | Pin # | Function | Direction |
|---------------------------|--------------|---|------------------|
| Port07/CSYNC | 28 | Port 07 or composite sync output | I/O |
| GND | 29 | Digital ground | Power |
| VCC | 30 | Digital Vcc | Power |
| PWM6 | 31 | 8-bit PWM output | O |
| Port02 | 32 | Port 02 | I/O |
| $\overline{\text{RESET}}$ | 33 | Reset | I |
| XTAL1 | 34 | Crystal oscillator input | AI |
| XTAL2 | 35 | Crystal oscillator output | AO |
| AGND | 36 | Analog ground | Power |
| AVCC | 37 | Analog Vcc | Power |
| R | 38 | OSD video output to drive Red | O/AO |
| G | 39 | OSD video output to drive Green | O/ AO |
| B | 40 | OSD video output to drive Blue | O/AO |
| OVL | 41 | OSD overlay output | O |
| Port0F/SOVL | 42 | Port 0F or OSD semi-transparency overlay output | I/O |
| GND | 43 | Digital ground | Power |
| VCC | 44 | Digital Vcc | Power |
| PWM1 | 45 | 8-bit PWM output | O |
| PWM2 | 46 | 8-bit PWM output | O |
| Port08/R1 | 47 | Port 08 or digital RGB output, red | I/O |
| PWM3 | 48 | 8-bit PWM output | O |
| PWM4 | 49 | 8-bit PWM output | O |
| Port09 | 50 | Port 09 | I/O |
| Port0A | 51 | Port 0A | I/O |
| PWM5 | 52 | 8-bit PWM output | O |

1.4 Development Tools

The Z90T361 requires ZiLOG's Z90369ZEM Emulator with its proprietary ZiLOG Developer Studio (ZDS) software for programming. To view code effects, the emulator uses a ZiLOG On-Screen Display (ZOSD) board that connects directly to a television screen. Refer to Figure 3.

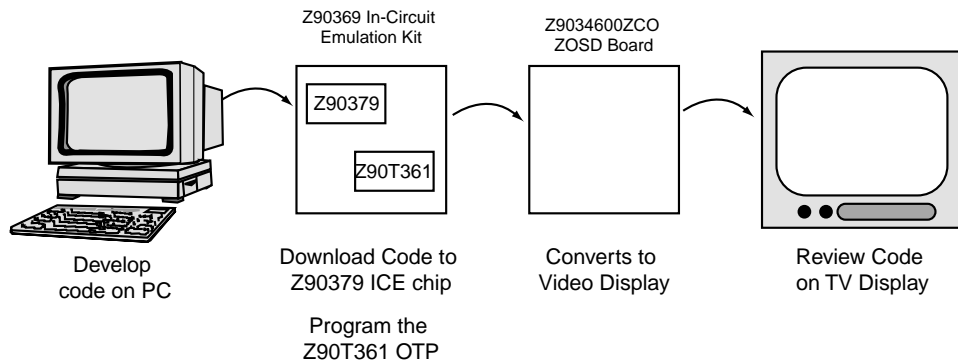


Figure 3 Code Development Environment

2 Operation

2.1 CPU Description

The Z89C00 core is a high-performance DSP that has a modified Harvard-type architecture with separate program and data memories. The design has been optimized for processing power.

The Z89C00 used in the Z90T366 device has been modified. The multiplier is disabled. However, the X and Y registers in the multiplier are still available and can be used as general-purpose registers. Refer to ZiLOG's Z89C00 documentation.

ALU

The 24-bit ALU has two input ports, one of which is connected to the output of the 24-bit Accumulator. The other input is connected to the 24-bit P-Bus; the upper 16 bits are connected to the 16-bit D-Bus.

Instruction Timing

Several instructions are executed in one machine cycle. Lengthy immediate instructions and Jump or Call instructions are executed in two machine cycles. When the program memory is referenced in internal RAM indirect mode, it requires three machine cycles. An additional machine cycle is required if the program counter (PC) is selected as the destination of a data transfer instruction. This only occurs with a register indirect branch instruction.

Hardware Stack

A six-level hardware stack is connected to the D-Bus to hold subroutine return addresses or data. The CALL instruction pushes PC+2 onto the stack. The RET instruction returns the contents of the stack to the program counter.

CPU Registers

The Z90T366 has 11 physical internal registers and four banks of eight external registers. In addition, it has nine virtual registers. The 11 internal registers are defined in Table 2, and the status register is defined in Table 3.

Internal Registers

Table 2 Internal Registers

| Register | Register Definition | Size |
|----------|--------------------------|-------------|
| X | General purpose register | 16 bits |
| Y | General purpose register | 16 bits |
| A | Accumulator | 24 bits |
| SR | Status Register | 16 bits |
| Pn:b | Six RAM Address Pointers | 8 bits each |
| PC | Program Counter | 16 bits |

Table 3 Status Register

| Bit/Field | Bit Position | R/W | Description |
|------------------------|--------------|-----|--|
| N | 15 | R | ALU Negative |
| OV | 14 | R | ALU Overflow |
| Z | 13 | R | ALU Zero |
| C | 12 | R | Carry |
| Reserved | 11 | R | Reserved |
| Reserved | 10 | R | Reserved |
| Reserved | 9 | R | Reserved |
| OP | 8 | R/W | Overflow Protection |
| IE | 7 | R/W | Interrupt Enable |
| Register Bank Selector | 6,5 | R/W | 00 Register Bank 0 01 Register Bank 1 10 Register Bank 2 11 Register Bank 3 |
| SFD | 4,3 | R/W | “Short Form Direct” Bits |
| RPL | 2-0 | R/W | RAM Pointer Loop Size |

X and **Y** are 16-bit general purpose registers.

A is a 24-bit Accumulator. The output of the ALU is sent to this register. When 16-bit data are transferred into this register, it goes into the 16 MSBs and the least

significant eight bits are set to zero. Only the upper 16 bits are transferred to the destination register when the Accumulator is selected as a source register in transfer instruction.

SR is the Status Register that contains the ALU status and the control bits listed in Table 3. The status register is always read in its entirety. S15-S12 are set/reset by the hardware and can only be read through software. They are set or reset by the ALU after an operation.

S8-S0 can be written by software. S8, if 0 (reset), allows the hardware to overflow. If S8 is set, the hardware clamps at maximum positive or negative values instead of overflowing. S7 enables interrupts. S6–S5 are used for “short form direct” addresses, which are described below. The definitions of S2-S0 are listed in Table 4.

Table 4 Ram Pointer Loop Description

| S2 | S1 | S0 | Loop Size |
|----|----|----|-----------|
| 0 | 0 | 0 | 256 |
| 0 | 0 | 1 | 2 |
| 0 | 1 | 0 | 4 |
| 0 | 1 | 1 | 8 |
| 1 | 0 | 0 | 16 |
| 1 | 0 | 1 | 32 |
| 1 | 1 | 0 | 64 |
| 1 | 1 | 1 | 128 |

Pn:b are the pointer registers for accessing data RAM.

(n= 0, 1, 2 refer to the pointer number)

(b = 0, 1 refers to RAM bank 0 or 1).

They can be read from or written to directly and can point directly to locations in data RAM or indirectly to Program Memory.

PC is the Program Counter. When this register is assigned as a destination register, one NOP machine cycle is automatically added to adjust the pipeline timing.

External Registers

The Z90T366 module is capable of accessing eight external registers directly using only the three external register address signals that are normally available. Two user bits (Status register S6-S5) are combined with the register address signals to provide the ability to address four banks of eight registers each. The registers most critical for speed are located together in Bank 3. In this specification, all external registers are referred to

$$RX(Y) <Z>$$

where:

X is a register number within a register bank;

Y is a bank number; and

Z is a bit field number

An external register bank can be selected by setting bits 6 and 5 in the status register to define the bank, then specifying the address of the register on the external register address bus.

External registers reside on the chip and are used to control the operation of all the peripheral modules in the device. By reading or writing to the fields in the external registers, the user can interact with the peripheral devices on the chip.

Virtual Registers

BUS is a read-only register that, when accessed, returns the contents of the D-Bus. It is a virtual register. (Physical RAM does not exist on the chip.)

Dn:b These eight data pointers refer to possible locations in RAM that can be used as pointers to locations in program memory. The programmer decides which location to choose two bits from in the status register and which two bits in the operand. This means only the lower 16 possible locations in RAM can be specified. At any one time there are eight usable pointers, four per bank, and the four pointers are in consecutive locations in RAM.

For example, if S3/S4 = 01 in the status register, then D0:0/D1:0/D2:0/D3:0 refers to locations 4/5/6/7 in RAM bank 0.

► **Note:** When the data pointers are being written to, a number is actually being loaded to Data RAM, so they can be used as a limited method for writing to RAM.

Additional Control Registers (AR)

Additional Control Registers (AR) control new peripheral blocks like palette banks and memory management. To activate ARs, R0(1) must be set to "1." ARs can

be disabled by setting R0(1) = 0, (POR) for software backward compatibility or if access to RAM location 1FFh is required.

The 128 eight-bit control registers (referred as AR or ARx<y:z>) use RAM-mapped I/O access. Location 1FFh in RAM is used to address up to 128 byte-width ARs. The AR number and written data are encoded into the data field as illustrated in Figure 4.

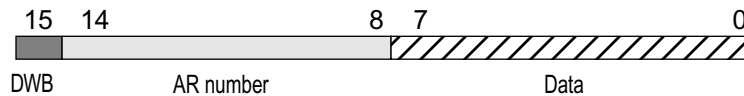


Figure 4 AR Register Format

When writing to address 1FFh, the Data Write Bit (DWB) and AR number are latched, depending on whether the DWB data field is either written to the selected port (latched) or discarded (not latched). The AR number and corresponding data are read after reading from the previously latched DWB address 1FFh.

To write to the AR, the data must be written to address 1FFh; DWB must be set to “1,” the port number must be specified in bits 8–14, and actual data must be specified in bits 0–7.

Example

```
LD    A, #(%8000 | 29 << 8 | %57); write 57 (hex) into the AR29
LD    %1FF, A;
```

The DWB and port number are latched for further reading if necessary.

To read from the AR, the address must be previously latched by writing it to address 1FFh with DWB set to “0.” Bits 0–7 have no meaning. Because the bits are not going to be written in this mode, only the port number is latched.

Example

```
LD    A, #(%0 | 30 << 8 | %0); DWB=0, latch AR30, data is not written
LD    %1FF, A;
LD    A, %1FF; read from AR30-%1EXX, where XX is current content
```

At least one cycle delay (NOP) is required between two consecutive accesses to the AR. If access is performed by a two-cycle instruction, no delay is necessary.

External memory must exhibit access times of less than 60 ns. Table 5 lists the additional control registers.

Table 5 Additional Control Registers

| AR # | Name | Bit position | Data | Function |
|--------|-------------|-------------------------------|-------------------------|--|
| 0 | Palette_8 | 76----- --543210 | Dh DDh | Reserved Palette8/Color0—R1R0G1G0B1B0 |
| 1 | Palette_8 | 76----- --543210 | Dh DDh | Reserved Palette8/Color1—R1R0G1G0B1B0 |
| 2 | Palette_8 | 76----- --543210 | Dh DDh | Reserved Palette8/Color2—R1R0G1G0B1B0 |
| 3 | Palette_8 | 76----- --543210 | Dh DDh | Reserved Palette8/Color3—R1R0G1G0B1B0 |
| 4 | Palette_8 | 76----- --543210 | Dh DDh | Reserved Palette8/Color4—R1R0G1G0B1B0 |
| 5 | Palette_8 | 76----- --543210 | Dh DDh | Reserved Palette8/Color5—R1R0G1G0B1B0 |
| 6 | Palette_8 | 76----- --543210 | Dh DDh | Reserved Palette8/Color6—R1R0G1G0B1B0 |
| 7 | Palette_8 | 76----- --543210 | Dh DDh | Reserved Palette8/Color7—R1R0G1G0B1B0 |
| 8–15 | Palette_9 | 76543210 | DDh | Same as AR 0–7 for Palette9 |
| 16–23 | Palette_10 | 76543210 | DDh | Same as AR 0–7 for Palette10 |
| 24–31 | Palette_11 | 76543210 | DDh | Same as AR 0–7 for Palette11 |
| 32–39 | Palette_12 | 76543210 | DDh | Same as AR 0–7 for Palette12 |
| 40–47 | Palette_13 | 76543210 | DDh | Same as AR 0–7 for Palette13 |
| 48–55 | Palette_14 | 76543210 | DDh | Same as AR 0–7 for Palette14 |
| 56–63 | Palette_15 | 76543210 | DDh | Same as AR 0–7 for Palette15 |
| 64–123 | | | | Reserved |
| 124 | PgLocation0 | 7----- -6----- --543210 | 0 1 0 1 DDh | Page0 is located internally—POR Page0 is located externally Internal ROM is enabled—POR Internal ROM is disabled (low power consumption) Page0—external (physical) page number |
| 125 | PgLocation1 | 7----- -6----- --543210 | 0 1 0 1 DDh | Page1 is located internally—POR Page1 is located externally Reserved Page1—external (physical) page number |
| 126 | PgLocation2 | 76543210 | DDh | Same as above for Page2 |

Table 5 Additional Control Registers (Continued)

| AR # | Name | Bit position | Data | Function |
|------|-------------|--------------|------|-------------------------|
| 127 | PgLocation3 | 76543210 | DDh | Same as above for Page3 |

RAM Addressing

The addresses in RAM can be specified in one of three ways: RAM, ROM, Pointers. Refer to Figure 5.

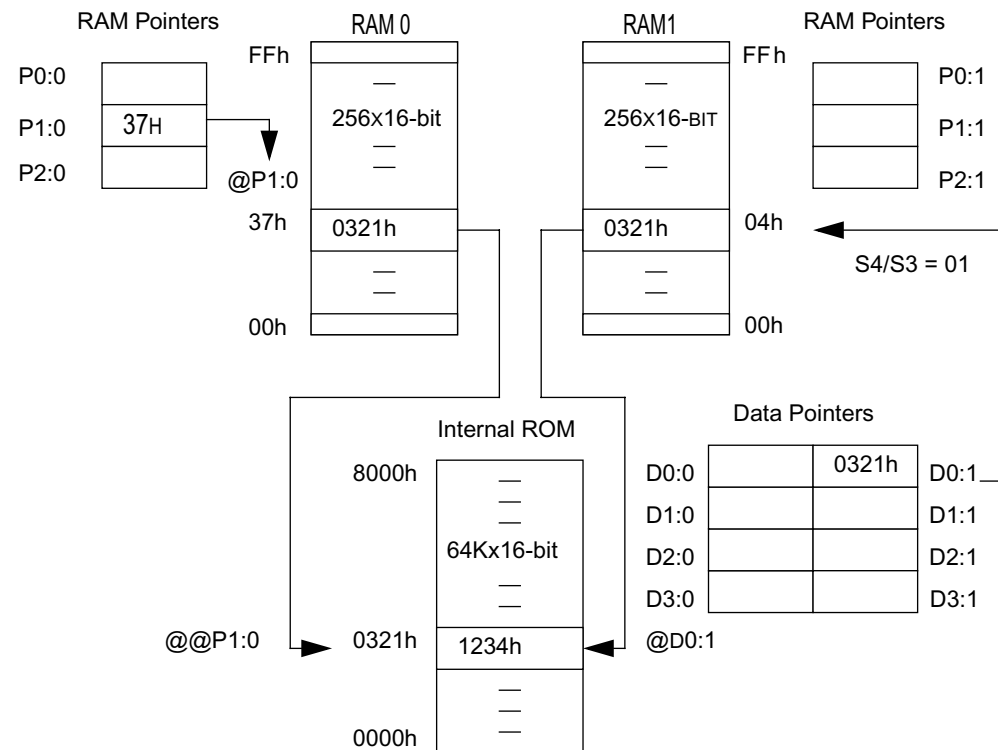


Figure 5 RAM, ROM, and Pointer Architecture

2.2 Memory (ROM and RAM)

The Z90T366 has 64K words of Read Only Memory (ROM) and 1K words of Random Access Memory (RAM).

ROM

The 64K words mask ROM is designed to provide storage for both program codes and Character set Graphic pixel matrices (CGROM). The address boundary between these applications is dependent on the storage required for character graphics.

The Program ROM (PGMROM) section can, in theory, be accessed anywhere in the addressable ROM space; however, because CGROM usually starts at location 0000h, program ROM resides in the higher address locations. The maximum available ROM space for program memory depends on the ROM reserved for CGROM (for an application).

CGROM can be placed anywhere in the 64K ROM address space by setting the CGROM address offset register R7(2). This offset is added to the character address before accessing ROM. By modifying the CGROM offset, several fonts can be accessed (limited by ROM size only). When reset, R7(2) = 0 (no offset) for backward compatibility with existing software. Refer to Figure 6.

| | | |
|------------|--|---------|
| 64K | Int0 vector | FFFFh |
| | Int1 vector | FFFEh |
| | Int2 vector | FFFDh |
| | Reset vector | FFFCCh |
| | | FFFBh |
| | Program ROM or CGROM | |
| Up to 5K | | 1400h |
| | CGROM—Bank 0, Scan lines 19, 20 or Bank1 or Program ROM | 13FFh |
| Up to 4.5K | | 1200h |
| | CGROM—Bank 0, Scan lines 17, 18 or Bank 1, or Program ROM | 11FFh |
| | | 1000h |
| 4K | Program ROM | 0FFFh |
| Up to 4K | | 10*nh |
| | CGROM-Bank 0 (n Characters) Scan lines 1-16 | 10*nh-1 |
| | | 0000h |

Figure 6 ROM Map

RAM

The 1K words RAM is organized in four banks of 256 words consisting of 16 bits each. Bank1.0 is always accessible. Bank0.0 is mapped to other bank(s); only one page from 0.X is active through bit selection. See Figure 7.

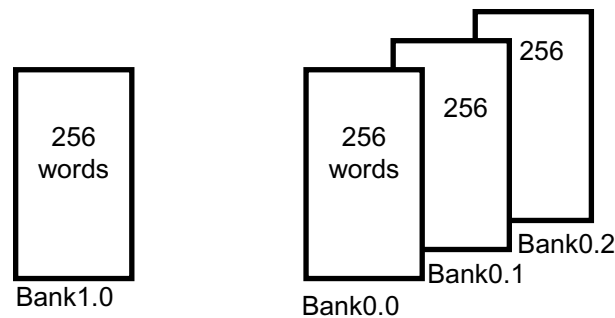


Figure 7 RAM Allocation

2.3 Clock Circuit Description

The processor is able to operate from several clock sources:

- Primary Phase Lock Loop VCO source (PVCO)
- Secondary Phase Lock Loop (SVCO)
- 32.768-KHz oscillator clock (OSC)

In addition, the processor clock can be halted temporarily to select the clock source or access ROM without disrupting normal operation of the processor.

An external crystal controls the internal 32.768-KHz oscillator. The crystal is used as the clock reference for the internal Phase Locked Loop (PLL). The PLL provides the internal PVCO clock for processor operation. The System Clock (SCLK) is generated internally by dividing the frequency of an appropriate oscillator (PVCO) by 2. The frequency of the SCLK after Power On Reset (POR) is 12.058 MHz.

The SCLK signal can be sent to the Port16 output pin under software control by setting bit 9 in register R3(1). The SVCO must be used as the system clock when the OSD is generated.

The clock switch control register R6(1) defines the source of the SCLK for the Z90T366 core. The block diagram in Figure 8 represents the clock switch circuit.

- **Note:** Clock switching is not recommended. This feature is only for advanced users.

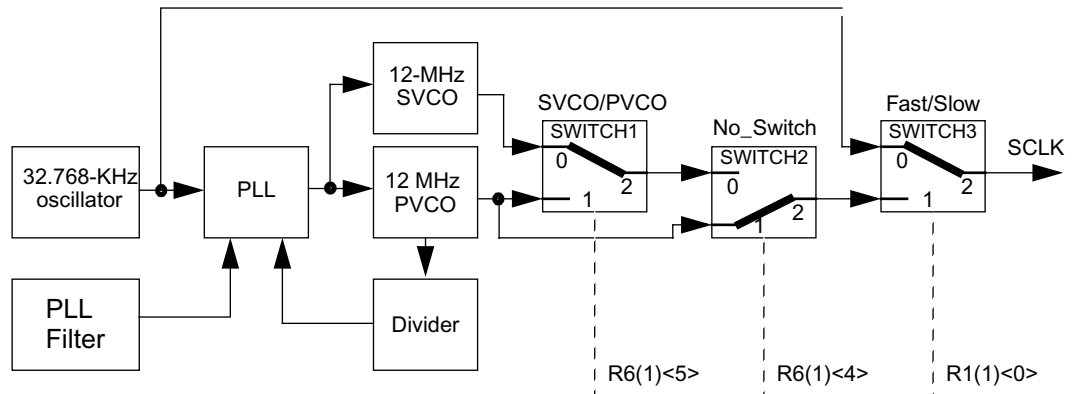


Figure 8 Clock Switching Block Diagram (8 -16 MHz)

- **Note:** Clock switching is discouraged. It is only for advanced users.

Input/Drive Circuits

The 32 KHz oscillator circuit in Figure 9 is suggested for proper clock operation.

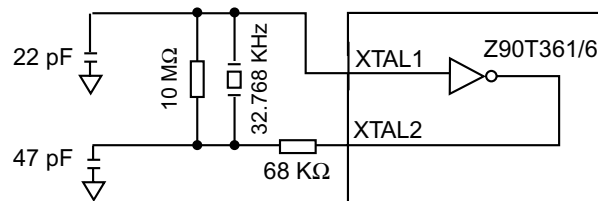


Figure 9 32 KHz Oscillator Recommended Circuit

2.4 Reset Conditions

Reset conditions including addresses and registers are listed in Table 6.

Table 6 Reset Conditions

| Addr | Register | Reset Condition | | | | | | | | | | | | | | | | Comments |
|-------|----------------------|-----------------|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|-------------------------------------|
| | | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
| R0(0) | reserved | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | Not available |
| R1(0) | Cursor Palette | x | x | 0 | 0 | x | x | x | x | x | x | x | x | x | x | x | x | Cursor palette gauge |
| R2(0) | pll_freq | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | PLL frequency control |
| R3(0) | I ² C_int | 0 | 0 | 0 | 0 | 0 | 1 | x | x | x | x | x | x | x | x | x | x | I ² C interface register |
| R4(0) | port0 | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | 16-bit I/O port 0 |
| R5(0) | port1 | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | 9-bit I/O port 1 |
| R6(0) | dir0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 16-bit port 0 direction |
| R7(0) | dir1 | x | x | x | x | x | x | x | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 9-bit port 1 direction |
| R0(1) | clamp_pos | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | x | x | x | x | x | x | x | position of video clamp pulse |
| R1(1) | sclk_freq | x | 0 | 0 | 0 | 0 | 0 | 0 | 0 | x | x | x | 0 | 0 | 0 | 0 | 0 | Stop/sleep/normal mode |
| R2(1) | 9-bit cntr | x | x | x | x | x | x | x | x | x | 0 | 0 | 0 | 0 | 0 | 0 | 0 | Stop and WDT, 9-bit counter |
| R3(1) | standard_ctl | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | x | x | x | 0 | 0 | Output H/VSYNC/ Blink Control |
| R4(1) | ADC_ctl | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | x | x | x | x | A/D converter control |
| R5(1) | cap_1s_ctl | x | x | x | x | 0 | 0 | x | x | x | x | x | x | x | x | x | x | Counter timers control |
| R6(1) | clock_ctl | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | x | x | x | x | Clock control (switch VCO/DOT) |
| R7(1) | wdt_smr_ctl | x | x | x | x | x | x | x | x | 0 | x | x | x | x | x | x | x | SMR and WDT control/interrupt |

Table 6 Reset Conditions (Continued)

| Addr | Register | Reset Condition | | | | | | | | | | | | | | | | Comments |
|-------|----------------|-----------------|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|---|
| | | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
| R0(2) | pwm_data1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | x | x | x | x | x | x | x | x | 8-bit PWM 1 data |
| R1(2) | pwm_data2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | x | x | x | x | x | x | x | x | 8-bit PWM 2 data |
| R2(2) | pwm_data3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | x | x | x | x | x | x | x | x | 8-bit PWM 3 data |
| R3(2) | pwm_data4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | x | x | x | x | x | x | x | x | 8-bit PWM 4 data |
| R4(2) | pwm_data5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | x | x | x | x | x | x | x | x | 8-bit PWM 5 data |
| R5(2) | pwm_data6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | x | x | x | x | x | x | x | x | 8-bit PWM 6 data |
| R6(2) | Shadow Ctrl | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | x | x | x | x | x | x | x | x | Shadow color ctrl |
| R7(2) | CGROM offset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | CGROM offset register |
| R0(3) | hi_x2_hi_x3 | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | Character multiple/ current data |
| R1(3) | lo_x2_mid_x3 | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | Character multiple/ next or previous data |
| R2(3) | Ch_x1_lo_x3 | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | Character multiple/ character graphics attribute |
| R3(3) | attr_data | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | Character attribute/ video RAM data |
| R4(3) | osd_cntl | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | On screen display control |
| R5(3) | cap_data | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | Capture register data |
| R6(3) | palette_color | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | Display palette color/ underline color |
| R7(3) | output palette | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | Output palette |

2.5 Power Management

There are two low-power operating modes for Z90T366: SLEEP mode and STOP mode.

SLEEP Mode

In SLEEP mode, the controller uses the 32.768-KHz clock for the SCLK to reduce power consumption.

STOP Mode

In STOP mode, the processor is suspended, and the power consumption is minimized.

2.6 I/O Port Configurations

User control can be monitored either through the front panel keypad scanning port or the 16-bit remote control capture register.

Two input/output port blocks are available for general-purpose digital I/O application. Each port bit can be programmed to be either an input or output port. To conserve the device pin count, some port pins are mapped to provide I/O to the ADC converter block and I²C interface block.

The 24 configurable I/O pins are general-purpose pins for functions such as serial data I/O, LED On/Off control, key scanning, power control and monitoring, and I²C serial data communications.

Port 0 and 1 directions are defined in R6(0) and R7(0), respectively. R4(0) and R5(0) are data registers for both Ports 0 and 1. Figure 10, Figure 11, and Figure 12 indicate I/O configuration and sharing with other functional units.

► **Note:** Port01 must be configured as an input in the user mode even though it is not called out in the device specification.

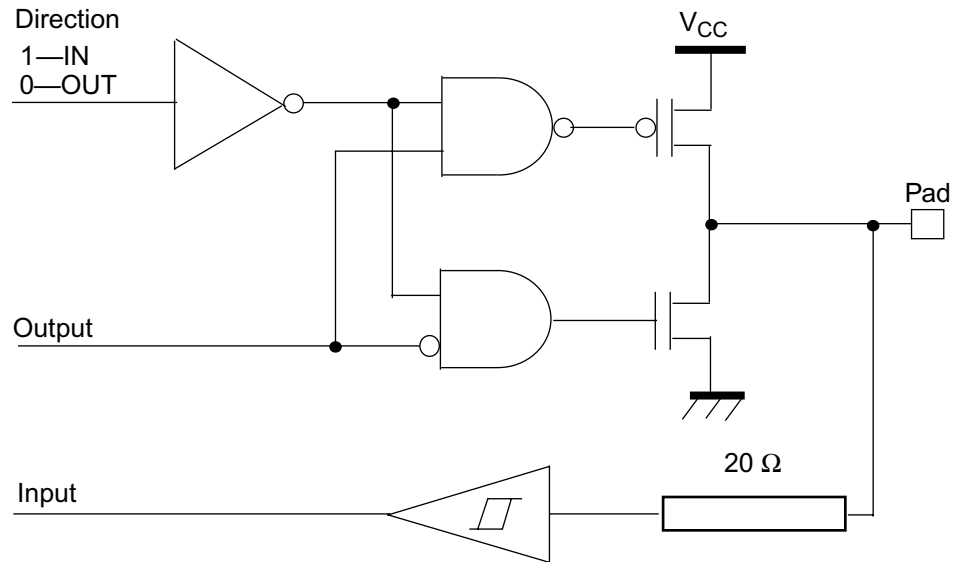


Figure 10 Bidirectional Port Pins

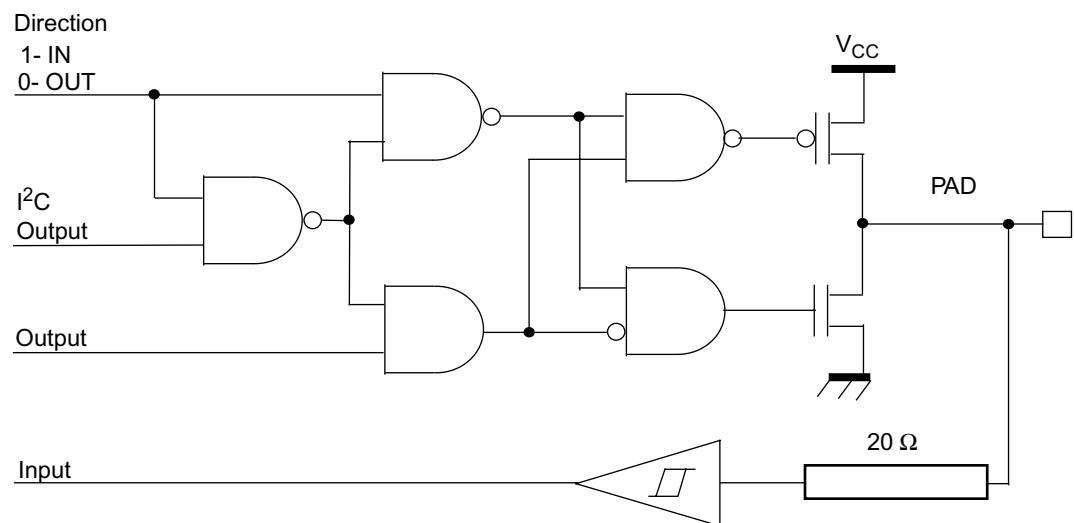


Figure 11 Bidirectional Pins Multiplexed with I²C Port

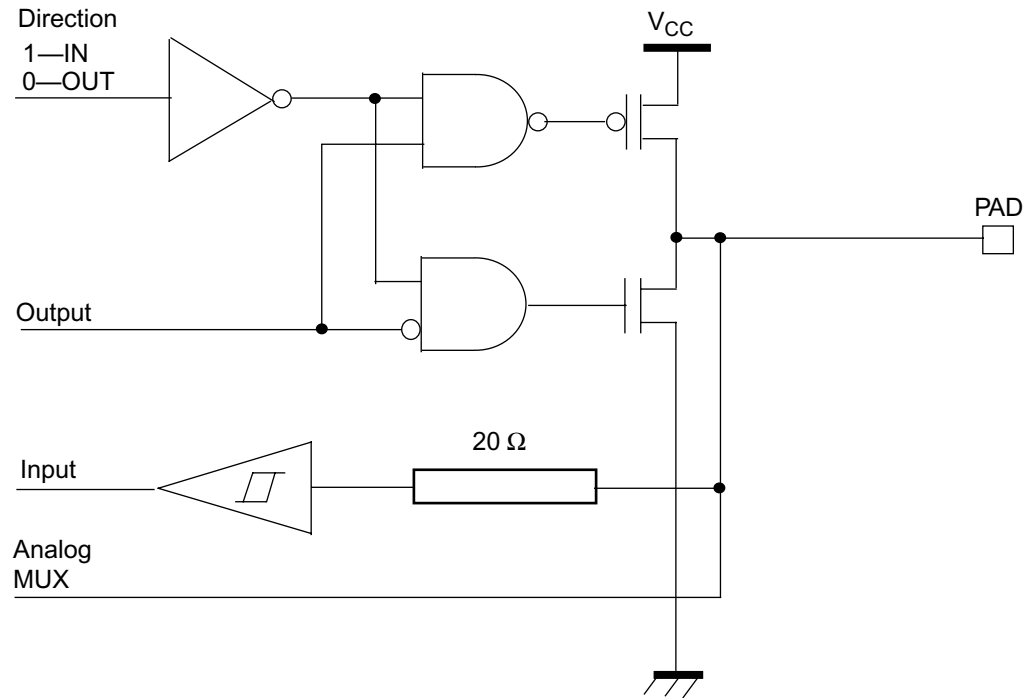


Figure 12 Bidirectional Pins Multiplexed with ADC Inputs

2.7 Interrupts

The Z90T366 has three external interrupt signals. There are four interrupt sources as follows:

- Horizontal sync (H_{SYNC})
- Vertical sync (V_{SYNC})
- IR Capture timer
- External event (Port09) } multiplexed

All interrupts are vectored. The capture timer and Port09 are multiplexed to the same interrupt.

Interrupt priorities are programmable. Each interrupt can be masked by setting fields in the external registers.

When the Z90T366 receives an interrupt request from one of the interrupt sources, it executes the interrupt service routine directly for that source.

External register R7 (1) controls interrupts.

2.8 Timers

Watch-Dog Timer

The watch-dog timer resets the CPU when it times out.

External register R7 (1) controls the watch-dog timer.

Real Time Clock

A clock timer, in real time, generates ticks every 1000, 250, 62.5 or 15.625 ms.

External register R5 (1) controls the real time clock.

IR Capture Timer

A capture timer measures time between edges of the IR signal. This timer can be programmed to measure timing from rising-to-rising, falling-to-rising, rising-to-falling, or falling-to-falling edges.

The IR capture timer is controlled by External register R5 (1) . Figure 13 is a block diagram of the IR capture register structure.

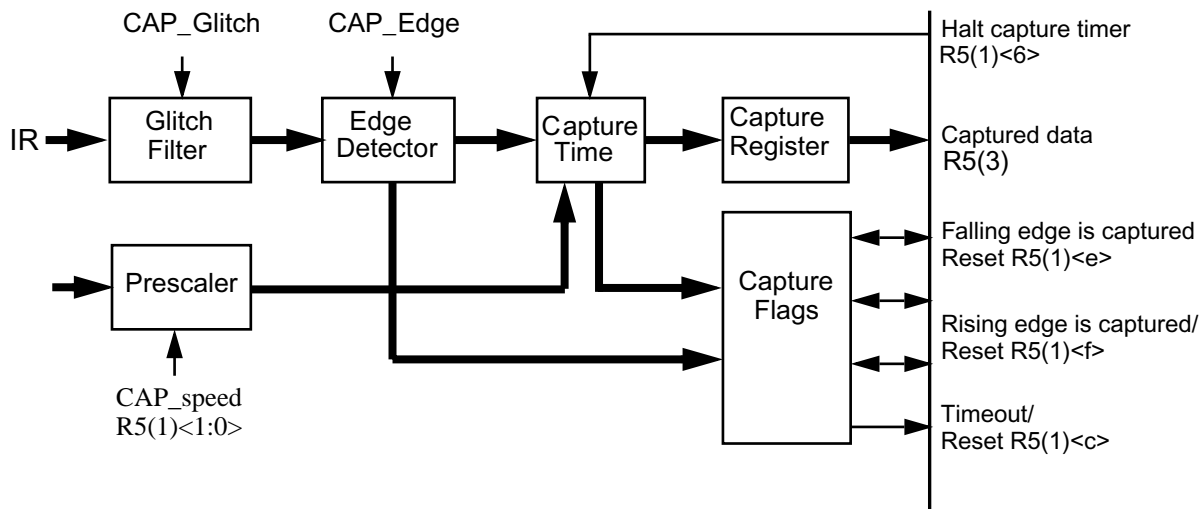


Figure 13 IR Capture Register Block Diagram

2.9 ADC

This function employs a 4-bit resolution, flash A-to-D converter. The six-to-one analog input multiplexor and conversion start circuits are controlled by the user program. The 4-bit conversion result is available to be read by the CPU at the end of each conversion.

One input channel (ADC0) is dedicated for quantizing VBI (vertical blanking interval) data for subsequent digital signal processing. Another channel, ADC5, is typically used for V_{SYNC} separation from the composite TV signal. These channels (ADC0 and ADC5) feature a special video clamp circuit that provides DC restoration of the composite video input signal. Typical VBI applications include Line 21 Closed Caption, Electronic Data Services, and StarSight Telecast. The range of ADC0 and ADC5 is from 1.5 to 2.0 V.

The four remaining channels of ADC (ADC1, ADC2, ADC3, and ADC4) are general purpose. They are typically used for tuner automatic frequency control and analog key entry. The range of ADC1–ADC4 is from 0 to 5.0 V.

The 4-bit ADC in the Z90T366 features six multiplexed inputs.

The allowed range for input signals differs for various ADC inputs according to Table 7.

Table 7 ADC Inputs Typical Range

| Input | Range (V) | Clamping | Typical application |
|-------------|-----------|------------|--|
| CVI/ADC0 | 1.5–2.0 | Yes (Ref–) | CCD sampling input |
| ADC1/Port17 | 0–5.0 | No | AFC input |
| ADC2/Port00 | 0–5.0 | No | Key scanning input |
| ADC3/Port05 | 0–5.0 | No | Key scanning input |
| ADC4/Port04 | 0–5.0 | No | Key scanning input |
| ADC5 | 1.5–2.0 | Yes (Ref+) | V_{SYNC} decoder sampling input |

Reference voltages that have been generated internally define the maximum range of the input signal for the ADC.

Nominal values are as follows:

$$\text{Ref+} = 2.0 \text{ V}$$

$$\text{Ref-} = 1.5 \text{ V @ } V_{\text{CC}} = 5.0 \text{ V}$$

For other V_{CC} values, the reference voltages must be prorated as follows:

$$\text{Ref+} = 0.4 * V_{CC}$$

$$\text{Ref-} = 0.3 * V_{CC}$$

The maximum sampling rate of the ADC converter is 3 MHz. It takes 4 SCLK cycles for valid output data from the ADC to become available. This is especially important if the application uses the single-shot mode.

The ADC exhibits monotonous conversion characteristics with a nonlinearity of less than 0.5 LSB. ADC0. The ADC has a range of 0.5V (from 1.5V to 2.0V) and is directly multiplexed to the input of the ADC. The remaining ADC inputs (ranging from 0V to 5V) use AGND and AV_{CC} voltage as a reference.

Figure 14 is a block diagram of the ADC inner structure, and Figure 15 illustrates ADC input circuits.

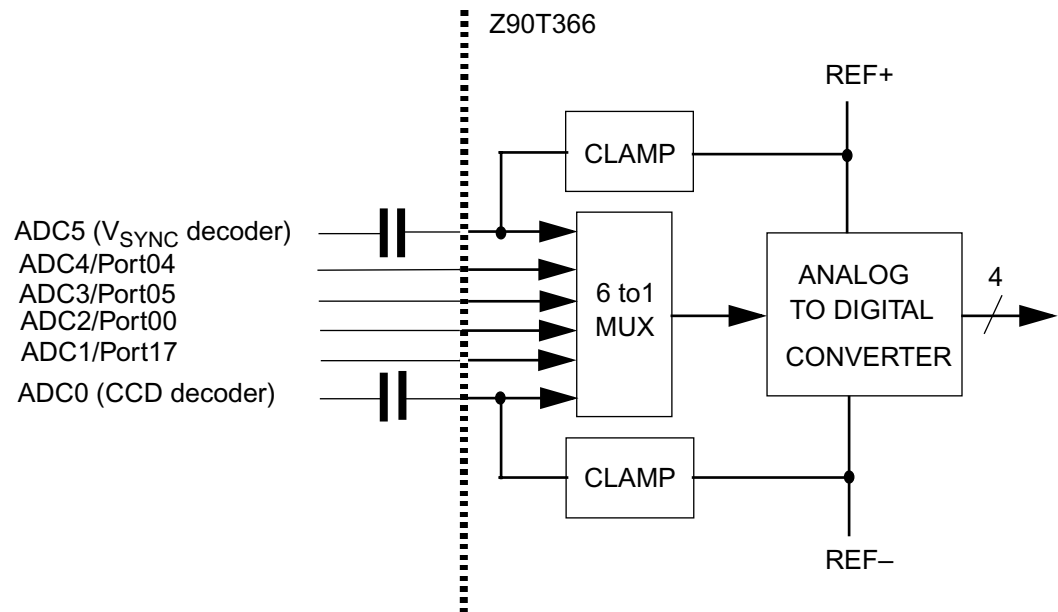


Figure 14 Z90T361/6 ADC Block Diagram

ADC Data Packing

Up to four 4-bit ADC data samples can be packed into one 16-bit word without software overhead. If $R4(1)<9> = 1$, every reading of $R4(1)$ returns the result, where the High 12 bits are the three previous ADC samples and the Low 4 bits are the current one, as illustrated in Figure 15.

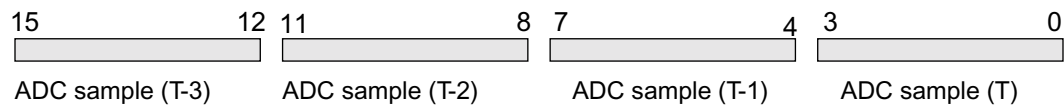


Figure 15 ADC Data Packing

The following routine shows ADC packing:

```
LD SR, %20; select RegBank1
LD A, EXT4; turn "ADC data packing" mode on
OR A, %0200; set R4(1)<9>
LD EXT4, A;
LD A, EXT4; read first ADC sample, A = %0005
LD A, EXT4; read second ADC sample, A = %005E
LD A, EXT4; read third ADC sample, A = %05E7
LD A, EXT4; read forth ADC sample, A= %5E74
LD A, EXT4; read fifth ADC sample, first sample is thrown away, A = %E741
```

The ADC is controlled by the external register $R4(1)$.

2.10 Pulse Width Modulation

Pulse Width Modulation (PWM) is used in conjunction with external low-pass filters to perform digital-to-analog conversion. Six PWMs (8-bit resolution each) generate signals for the control for video and sound attributes. In case of a chassis employing a frequency synthesis tuner, these PWMs can also control video or sound attributes. PWM's can also be used to acknowledge tones for remote or keyboard commands.

Each PWM circuit features a data register whose contents are set under program control. The data in the register determines the ratio of PWM High to PWM Low time. PWM data registers are not initialized when reset. In order to eliminate a potential glitch on a PWM output, it is recommended to initialize PWM data registers before enabling the VCOs.

External registers R0(2) to R5(2) are data registers for PWM1 to PWM6 accordingly.

2.11 I²C Interface

There are two hardware modules that support standard I²C bus protocol according to the I²C bus specification published by Philips in 1992, titled *I²C Peripherals for Microcontrollers Data Handbook*.

The first module, the Master, can be configured for fast (400 KHz) or slow (100 KHz) bit rates and can be used in applications with a single master.

The Z90T366 adds two additional non-standard bit rates (50 KHz and 10 KHz) and an additional multiplexed master port that is controlled by the I²CM_mux control bit.

Table 8 lists the bit rates for the Master I²C Bus.

Table 8 Master I²C Bus Bit Rates

| Mode | I ² C mode | Bit Rate | Actual Bit Rate |
|---------|-----------------------|-----------|-----------------|
| LO/Slow | — | 0–10 KHz | 10 KHz |
| HI/Slow | — | 0–50 KHz | 44 KHz |
| LO/Fast | Slow | 0–100 KHz | 91 KHz |
| HI/Fast | Fast | 0–400 KHz | 334 KHz |

To suppress possible problems on both data (SDA) and clock (SCL) lines, digital filters are available for all inputs of the I²C bus interface. These filters exhibit a time constant equal to $3T_{SCLK} = 250$ ns.

If the Master I²C interface is enabled, corresponding I/Os, Port11 and Port12, must be assigned as outputs.

Master and Slave modules cannot be used simultaneously because of the shared I²C data register (see the Register 3(0) data field). The software activates I²C modules by writing appropriate commands into the control register. To control the I²C bus interface, the control register R3(0) toggle bit <c> must point to an appropriate interface (Master or Slave).

M_disable or **S_disable** bits allow either the Master or Slave I²C interface to be disabled so as not to interfere with any activity associated with the Port pins. At Power-on Reset (POR), both I²C interfaces are enabled. To use the I²C interface,

the corresponding Port pin (multiplexed with the I²C Data and Clock) must be configured as an output, while M_disable or S_disable bits must be reset to 0.

External register R3(0) controls the I²C. Table 9 lists the Master I²C bus interface commands. Table 10 lists the Slave I²C bus interface commands. Figure 16 and Figure 17 are flow charts of the Master and Slave modes.

Table 9 Master I²C Bus Interface Commands

| Command | Notes/Function |
|----------------|---|
| 0 0 0 | This command sends a start bit, followed by an address byte specified in the “data” field (bits <7:0>), then fetches an acknowledgment in bit <0>. This command initializes communication and generates 9 SCL cycles. |
| 0 0 1 | This command sends one byte of data specified in the “data” field (bits <7:0>), then fetches an acknowledgment in bit <0>. This command is used in a WRITE frame and generates 9 SCL cycles. |
| 0 1 0 | This command sends bit <7> as an acknowledgment (ACK = 0, NAK = 1), then receives a data byte. This command is used in a READ frame when the next data byte is expected and generates 9 SCL cycles. Received data appears in the “data” field (bits <7:0>). |
| 0 1 1 | This command sends bit <7> as an acknowledgment (ACK = 0, NAK = 1). This command is used in a READ frame to terminate data transfer and generates one SCL cycle. |
| 1 0 0 1 0 1 | A NULL operation. This command must be used with a “RESET” bit and/or a “TOGGLE” bit <c>. Using the “RESET” and/or “TOGGLE” bits with any other command interferes with the logic of the I ² C interface. |
| 1 1 0 | This command receives one data byte. It is used in a READ frame to receive the first data byte after the address byte is transmitted. It generates 8 SCL cycles. |
| 1 1 1 | This command sends a stop bit and generates one SCL cycle. |

Table 10 Slave I²C Bus Interface Commands

| Command | Notes/Function |
|----------------|---|
| 0 0 0 | Reserved. Cannot be used. |
| 0 0 1 | This command sends bit <7> as an acknowledgment (ACK = 0 only), then receives one data byte. This command is used in a WRITE frame and requires 9 SCL cycles. Received data is read as a “data” field (bits <7:0>). |
| 0 1 0 | This command sends one byte of data specified in a “data” field (bits <7:0>), then fetches an acknowledgment in bit <0>. This command is used in a READ frame and requires 9 SCL cycles. |
| 0 1 1 | Reserved. Cannot be used. |
| 1 0 0 1 0 1 | A NULL operation. This command must be used with a “RESET” bit and/or “TOGGLE” bit <c>. Using the “RESET” and/or “TOGGLE” bits with any other command interferes with the logic of the I ² C interface. |
| 1 1 0 | This command sends a bit <7> as a not acknowledgment (NAK = 1 only) in a WRITE or READ frame. This command terminates I ² C communication and requires one SCL cycle. The “Sulfonamide” bit <a> is automatically reset when a “busy” bit <9> goes Low. This command sends a bit <7> as an acknowledgment (ACK = 0 only) in a READ frame and requires one SCL cycle. The Send data command (010) must be executed next. This command acknowledges an address byte in a READ frame. |
| 1 1 1 | Reserved. Cannot be used. |

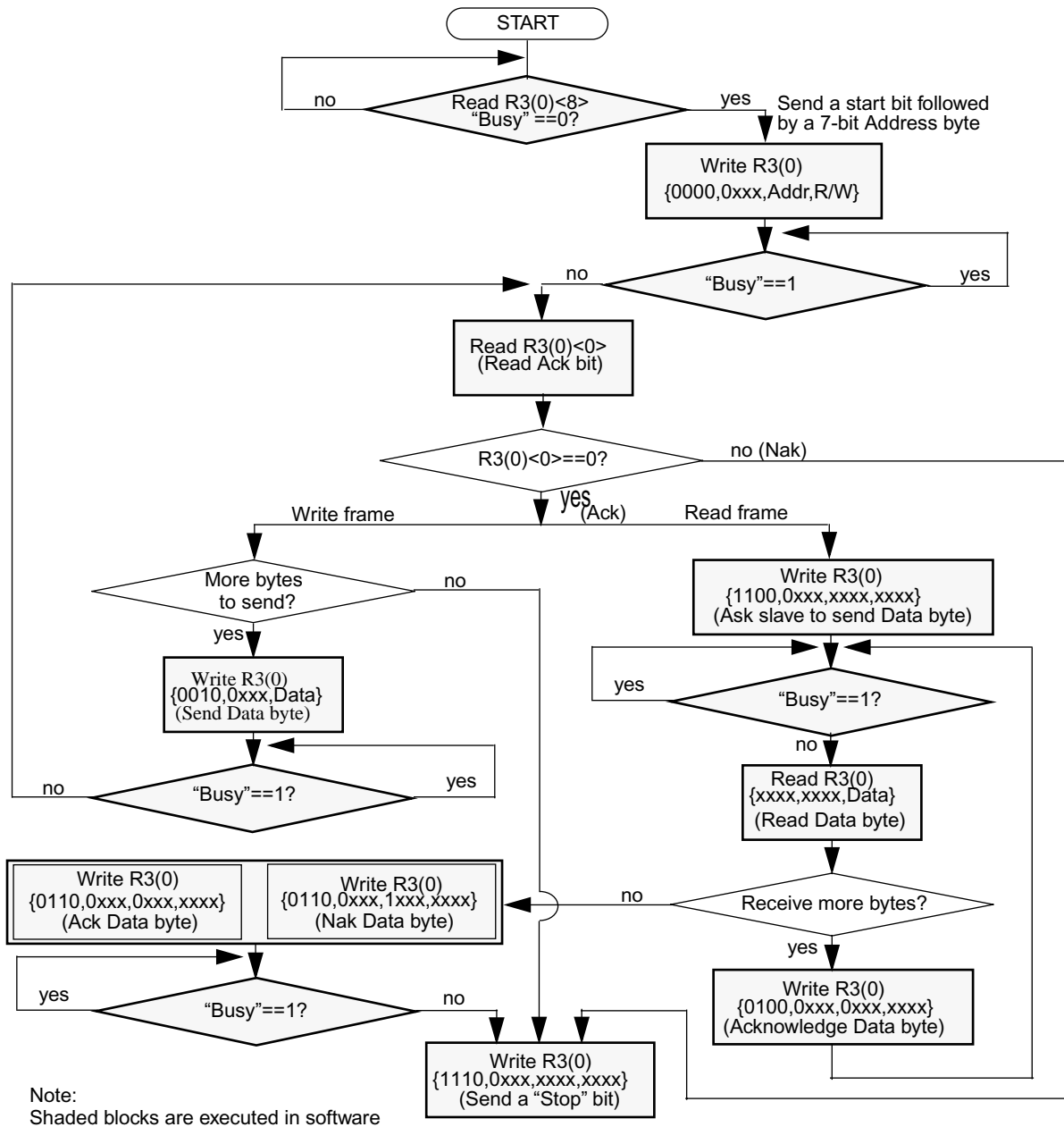
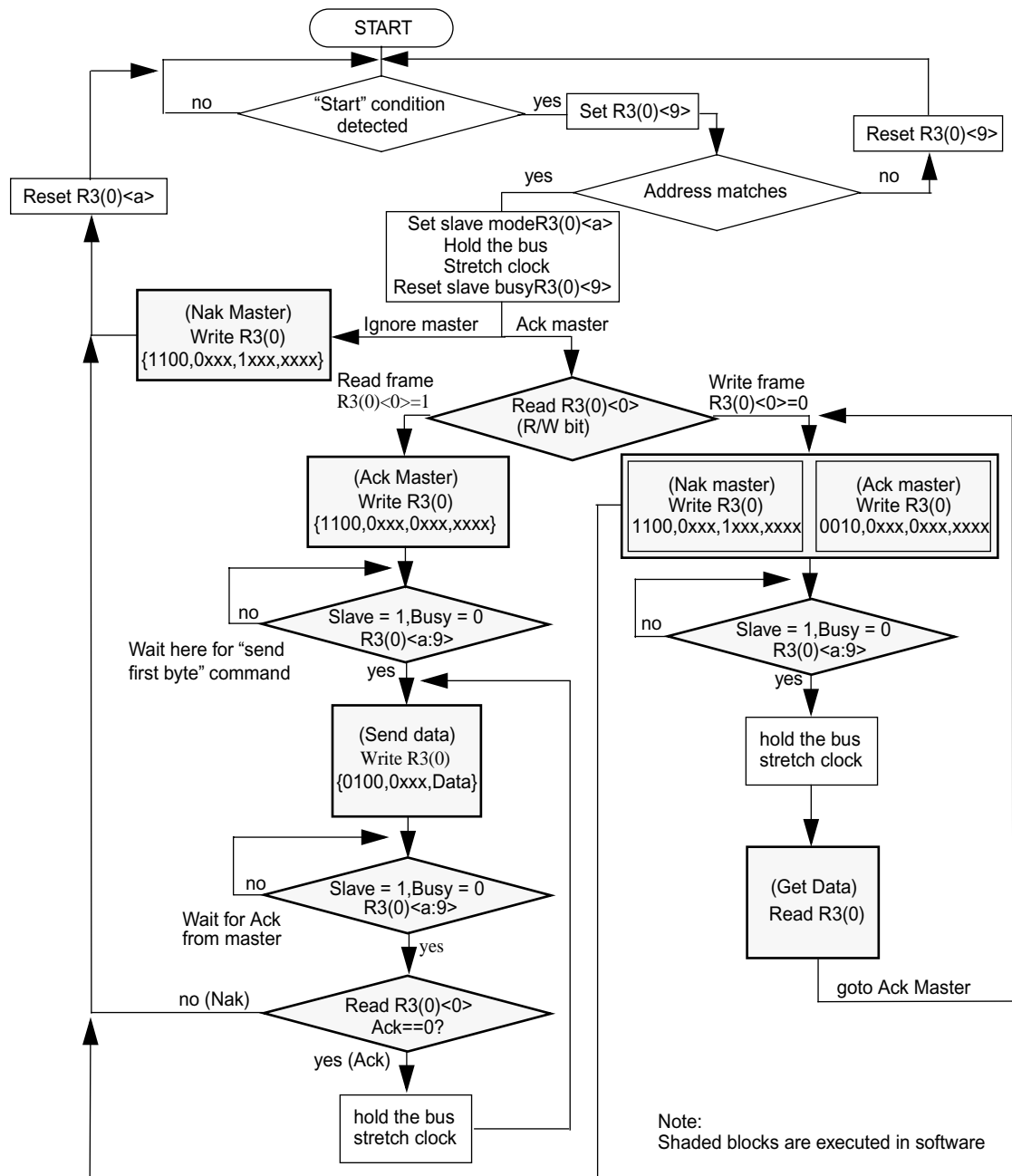


Figure 16 Master Mode



If a "Stop" condition is detected at any point, the hardware resets the "Slave" bit (R3(0)<a>) and releases the I²C bus.

Figure 17 Slave Mode

2.12 On-Screen Display (OSD)

The Z90T366 provides sophisticated on-screen display features. On-Screen Display has the following two modes:

- OSD Used to generate TV control OSD
- CCD Used to display Closed Caption information

OSD mode provides access to the full set of control attributes including latched and unlatched attributes. Unlatched attributes can be modified on a character-by-character basis. Control characters change latched attributes.

Any 256-character set can be displayed with many display attributes, including underlining, italics, blinking, eight foreground and background colors, character position offset delay, and background transparency. A 16-bit display character represents foreground color, background color, and underline attributes, which can be modified character by character. In addition, the Z90T361 supports eight fixed plus eight programmable color palettes out of 64 colors, independent left and right shadows with color control. Shadows are available on transparent and nontransparent backgrounds. Semi-transparency is supported on a character-by-character basis. A character's pixel array is stored as 16, 18, or 20 words in Character Generation ROM (CGROM).

Additional hardware provides the capability to display characters at two and three times normal size. The smoothing logic contained in the on-screen display improves the appearance of two and three times normal size characters. Shadows can be activated to improve the visibility of characters by adding a border (one pixel wide) on each side.

The Z90T366 provides RGB signals and a video blank signal. RGB outputs are available in two modes: digital and analog. In digital mode, the output RGB signals correspond to a primary colors palette. Analog mode supports 15 different palettes, which can be chosen under software control. In analog mode, each RGB output is generated by a 2-bit digital-to-analog converter. The user can switch the 2-bit digital inputs of the digital-to-analog converter to Port pins (Port10, Port13, Port14, Port15, Port18 and Port08) under software control by setting bit9 in register R3(1).

Video synchronization is normally obtained from H_FLYBACK and V_FLYBACK but can be generated by the Z90T366 and driven to the external deflection unit using the bidirectional H_SYNC and V_SYNC ports when external video synchronization signals are not present.

OSD is completely software controlled. Hardware supports the optimum generation of the character-based OSD; however, the CPU can bypass it and

generate pixels and attributes directly. The block diagram in Figure 18 illustrates the OSD data flow.

Figure 19 and Figure 20 indicate the R, G, B, and Blank output circuits.

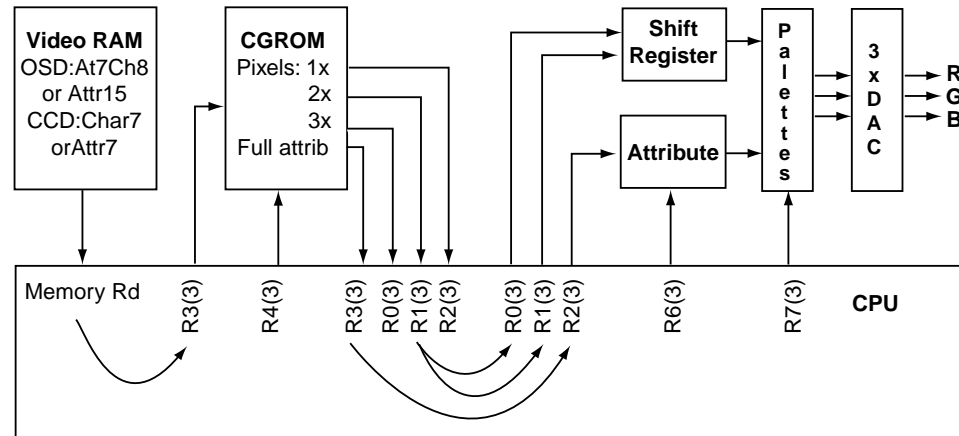


Figure 18 Data Flow

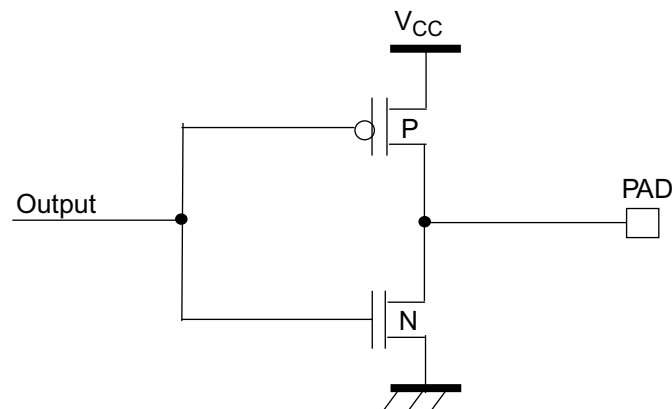


Figure 19 Blank and R, G, B Outputs in Digital Mode

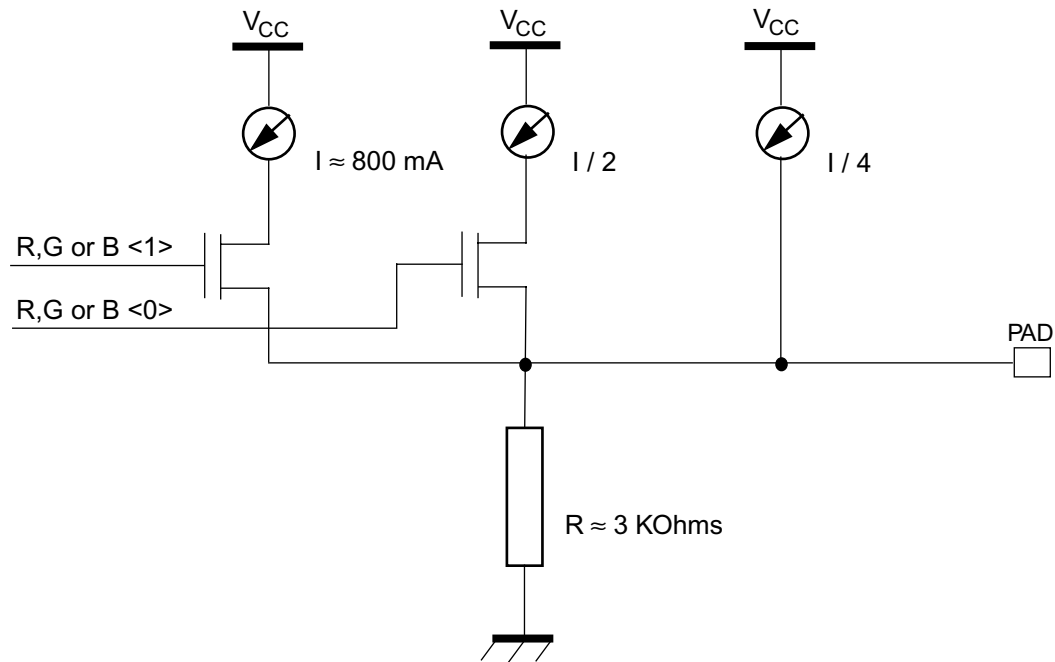


Figure 20 R, G, and B Outputs in Analog (Palette) Mode

Closed Caption Data Capture

Closed-caption text can be decoded directly from the composite video signal using the processor's digital signal processing capabilities and displayed on the screen. The character representation in this mode provides simple attribute control by inserting control characters. Each word of video RAM specifies two displayed characters.

The 4-bit flash A/D converter, with proper clamping, provides the ability to receive the composite video signal directly and process the closed-caption text embedded in the signal. Signal processing can be applied directly to the signal to improve decoder performance.

CGROM Relocation

CGROM can be placed anywhere in the 64K ROM address space by setting the CGROM address offset register R7(2). This offset is added to the CGROM address before accessing the ROM. By modifying the CGROM offset, several

fonts can be accessed (limited only by ROM size). When reset, $R7(2) = 0$ (no offset), making the Z90T366 backward-compatible with existing OSD control software.

The character scan line from CGROM addressed by the character register is fetched and stored into the CGROM capture register. If a pixel is set to 1, it displays the foreground color. If a pixel is set to 0, it displays the background color. The scan line can be stretched by the character multiplier to be two or three times normal character size by duplicating each bit in the word.

Controlling Character Expansion

The character size can be stretched to two or three times the size of the scan line. Hardware fetches data from CGROM and stretches the data to be read from registers $R0(3)$, $R1(3)$, and $R2(3)$. Figure 21 is a block diagram of the structure of the character expansion multiplexor, and Table 11 lists bit functions.

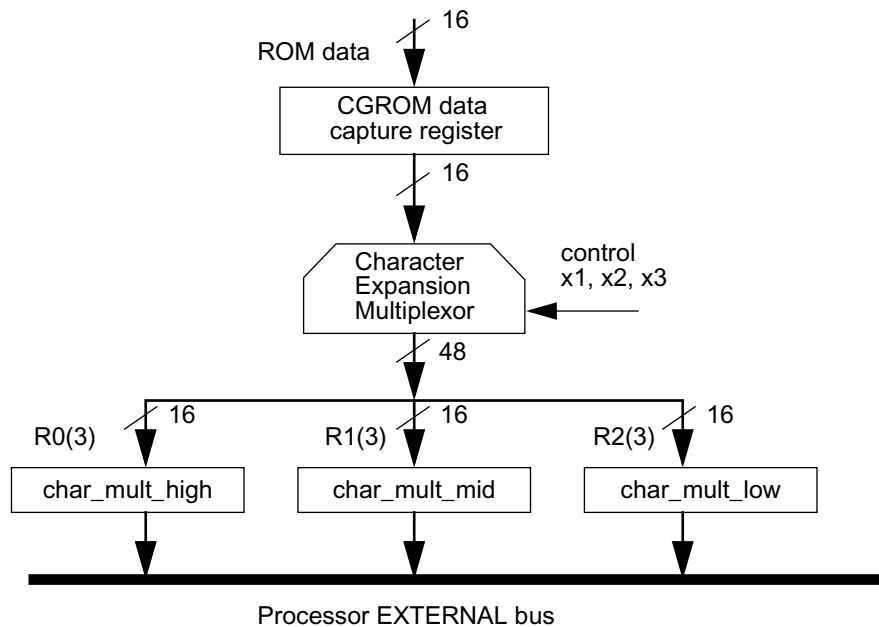


Figure 21 Character Expansion

Table 11 Character Expansion Register

| Capture Register Contents | Char_mult_high | Char_mult_mid | Char_mult_low |
|---------------------------|-------------------|-------------------|---------------|
| x1 operation | abcdefghijklmnop | | |
| x2 operation | aabbccddeeffgghh | iijjkkllmmnnooppp | |
| x3 operation | aaabbbcccdddeeeef | ffggghhhiiijjjkk | klmmnnnnnoopp |

Displayed Data Formats

The Z90T366 hardware supports the following two different data formats:

- **OSD mode, R4(3)<d> = 1** supports a standard OSD with full set of features.
- **CCD mode, R4(3)<d> = 0** supports reduced features which comply with the recommendations of the FCC on Closed Caption support.

In CCD mode, the background color of the characters *cannot* be changed and is always preset to BLACK.

OSD Mode

In OSD mode, each character occupies a 16-bit word in VRAM. There are two possible character formats defined: a “display” character and a “control” character. The code stored in “display” character format defines a character code and up to 7 attributes of the character.

The “control” character defines latched attributes and is presented on-screen as a space character. The combination of “display” and “control” characters provides versatile OSD generation.

Smoothing is supported for double-size (x2) and triple-size (x3) characters only.

CCD Mode

In CCD mode, each character occupies 8 bits (one byte) in VRAM. The CCD characters must be mapped into a 16-bit VRAM data field. The hardware supports compressed placement of characters in VRAM. Each word in VRAM is represented by a High byte and a Low byte. A currently active byte is selected by R4(3)<c>. The format and data representation in both bytes is exactly the same.

There are two possible character formats defined: a “display” character and a “control” character. The code stored in “display” character format defines a

character code. The “control” character defines up to five attributes (foreground color, italic, underline, blinking, and transparent). It is presented on screen as a space character. The combination of Display and Control characters provides the basis for a specified range of attributes defined by FCC specifications for CCD.

Shadows

Shadows, if enabled, are active on both transparent and nontransparent backgrounds. Two bits in the AttributeWR and AttributeRD registers, (R2(3)<1:0> and R3(3)<1:0>), control the type of shadow. Refer to Figure 22.

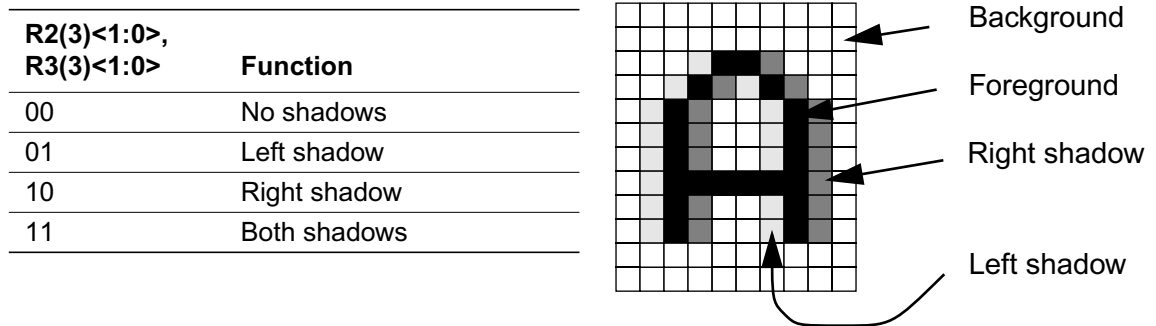


Figure 22 Table Settings

The smoothing attribute has been moved to R7(3)<5>.

The bit assignment in the Latched Attribute follows the bit assignment in R2(3).

The left and right shadow colors are independently controlled by R6(2)<d:b> and R6(2)<a:8>.

The smoothing control bit R7(3)<5> must be set to a “1” in order to activate fringing.

Semi-Transparent

Both semi-transparency (pin name SOVL) and transparency (pin name OVL) attributes are supported. The semi-transparency mode can be enabled through either latched or unlatched attributes. Latched attributes remain set until they are reset. Unlatched attributes remain set for only one character, which means the attribute must be constantly refreshed on a character-by-character basis.

Activation

To activate semi-transparency output, two bits must be set properly. Port 0F must be in output mode [R6(0)<f> = 0], and the SOVL/Port0F control bit must be in SOVL mode [R3(1)<6> = 1].

Latched Semi-Transparency

The latched semi-transparency attribute is controlled by bit [R2(3)<6>].

Unlatched Semi-Transparency

The unlatched semi-transparency attribute is controlled by bit [R3(3)<8>]. This bit has one of four possible assignments depending on how it is set up in [R7(3)<7:6>]. The four assignments are underline, semi-transparency, blinking, and CGROM bank select.

Notes:

1. The semi-transparency signal (SOVL), when active, is only valid with the background color. With the foreground color, SOVL is inactive. Therefore, characters do NOT take on a semi-transparent appearance (only the background does). This condition allows characters to be read without interference.
2. If both the transparency (background and foreground color are equal) and semi-transparency are activated, the OVL signal is High, and the SOVL signal is Low.

Attribute Assignment

Depending on R7(3)<7:6>, bit 8 of the Attribute_Data register, R3(3) (in character mode) can be assigned to control either "1st underline," "semi-transparency," "blinking," or "CGROM bank select," as indicated in Table 12.

Table 12 Attribute Assignment

| R7(3)<7:6> | R3(3)<8> |
|------------|---------------------|
| 00 | 1st underline (POR) |
| 01 | Semi-transparency |
| 10 | Blinking |
| 11 | CGROM bank1 select |

2.13 Cursor

The cursor buffer (a one-line pixel buffer) is loaded via the DMA on every line where the cursor is displayed (no software support is required). Horizontal size is programmable at 16, 32, or 48 pixels wide, and vertical size is programmable from 1 to 63 lines per field. The color depth is 2 bits per pixel, 3 programmable colors and the transparency. Depending on R1(0)<d>, the cursor's colors can be selected either from a current palette (R1(0)<d> = 0) or from Palette #6 (R1(0)<d> = 1). Refer to Table 15. The cursor image is stored in ROM as a bitmap. The number of cursors is limited by available ROM size.

The cursor is positioned by initializing cursor parameters in the beginning of every field.

Initialization occurs by setting the Cursor_Info_Load bit R7(3)<4> to 1, then writing sequentially to the R3(3) 16-bit parameters (COLOR, HPARAM, VPARAM and CADDR, respectively).

The cursor buffer is loaded from ROM at the leading edge of H_{SYNC} wherever the horizontal line requires a cursor. This process halts the CPU for 3/5/7 cycles depending on the cursor's horizontal size. The cursor bitmap address pointer (CADDR) is incremented automatically.

Though the cursor can be displayed anywhere on the screen, limiting the cursor to the OSD area is best. Outside of the OSD area, the cursor can jitter or become distorted.

The cursor bitmap is organized as pixel data placed sequentially in the ROM. The data format is described below.

For the interlaced mode, even and odd cursor bitmaps must be defined separately. Proper selection occurs during the cursor initialization at the beginning of every field. See Figure 23, Table 13, and Table 14.

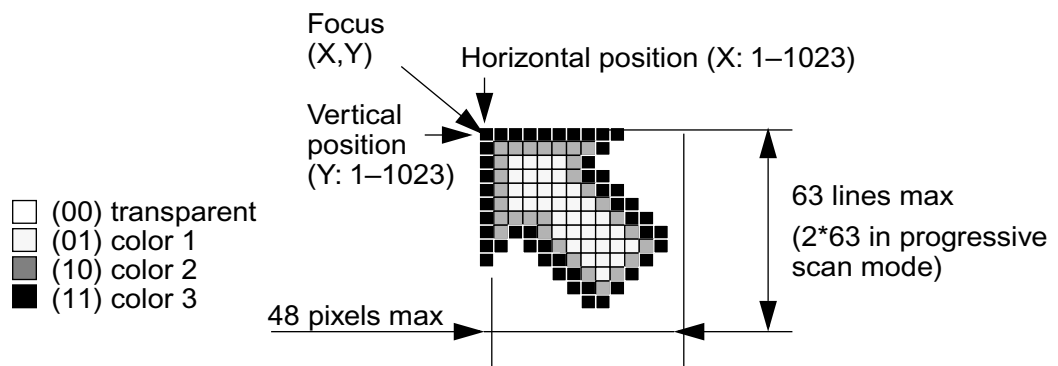


Figure 23 Cursor

Example

```
LD    SR,%#60 ; select RegBank3
OR    EXT7,%#0010; enable Cursor_Info_Load
LD    EXT3,#(3*64 + 7*8 + 2); load CCOLOR: color3 = 3, color2 = 7, color1 = 2
LD    EXT3,#(2*1024 + 120); load HPARAM: hsize = 2 (32 pixels), hpos = 120
LD    EXT3,#(28*1024 + 55); load VPARAM: vsize = 28, vpos = 55
LD    EXT3,%#6000; load CADDR—cursor bitmap address
AND   EXT7,%#FFEF; disable Cursor_Info_Load, ready for OSD
```

Table 13 Cursor Parameters

| Parameter | Reg field | Bit position | Data | Description |
|-----------|-----------|------------------|--------|---|
| CADDR | CADDR | fedcba9876543210 | D | Cursor bitmap address (pointer to cursor bitmap in ROM) |
| VPARAM | VSIZ | fedcba----- | 1–63 | Vertical size (lines in one field) |
| | VPOS | -----9876543210 | 1–1023 | Vertical position (lines in one field) |
| HPARAM | n/a | fedc----- | n/a | Reserved |
| | HSIZ | ----ba----- | 00 | No cursor |
| | | | 01 | 16 pixels wide |
| | | | 10 | 32 pixels wide |
| | | | 11 | 48 pixels wide |
| | HPOS | -----9876543210 | 1–1023 | Horizontal position (pixels from trailing edge of HSYNC) |
| CCOLOR | n/a | fedcba9----- | n/a | Reserved |
| | Color3 | -----876----- | 0–7 | Cursor Color 2 assignment |
| | Color2 | -----543--- | 0–7 | Cursor Color 1 assignment |
| | Color1 | -----210 | 0–7 | Cursor Color 0 assignment |

Table 14 Memory Allocation for Cursor Bitmap

| | |
|----------------------------|---|
| 16 Pixels Wide Mode | |
| AddrN: | L0_P15_B0, L0_P14_B0, L0_P13_B0, ... , L0_P1_B0, L0_P0_B0 |
| AddrN+1: | L0_P15_B1, L0_P14_B1, L0_P13_B1, ... , L0_P1_B1, L0_P0_B1 |
| AddrN+2: | L1_P15_B0, L1_P14_B0, L1_P13_B0, ... , L1_P1_B0, L1_P0_B0 |
| AddrN+3: | L1_P15_B1, L1_P14_B1, L1_P13_B1, ... , L1_P1_B1, L1_P0_B1 |
| | |
| AddrN+2n: | Ln_P15_B0, Ln_P14_B0, Ln_P13_B0, ... , Ln_P1_B0, Ln_P0_B0 |
| AddrN+2n+1: | Ln_P15_B1, Ln_P14_B1, Ln_P13_B1, ... , Ln_P1_B1, Ln_P0_B1 |
| 32 Pixels Wide Mode | |
| AddrN: | L0_P31_B0, L0_P30_B0, L0_P29_B0, ... , L0_P17_B0, L0_P16_B0 |
| AddrN+1: | L0_P31_B1, L0_P30_B1, L0_P29_B1, ... , L0_P17_B1, L0_P16_B1 |
| AddrN+2: | L0_P15_B0, L0_P14_B0, L0_P13_B0, ... , L0_P1_B0, L0_P0_B0 |
| AddrN+3: | L0_P15_B1, L0_P14_B1, L0_P13_B1, ... , L0_P1_B1, L0_P0_B1 |
| AddrN+4: | L0_P31_B0, L0_P30_B0, L0_P29_B0, ... , L0_P17_B0, L0_P16_B0 |
| AddrN+5: | L0_P31_B1, L0_P30_B1, L0_P29_B1, ... , L0_P17_B1, L0_P16_B1 |
| | |
| AddrN+4n: | Ln_P31_B0, Ln_P30_B0, Ln_P29_B0, ... , Ln_P17_B0, Ln_P16_B0 |
| AddrN+4n+1: | Ln_P31_B1, Ln_P30_B1, Ln_P29_B1, ... , Ln_P17_B1, Ln_P16_B1 |
| AddrN+4n+2: | Ln_P15_B0, Ln_P14_B0, Ln_P13_B0, ... , Ln_P1_B0, Ln_P0_B0 |
| AddrN+4n+3: | Ln_P15_B1, Ln_P14_B1, Ln_P13_B1, ... , Ln_P1_B1, Ln_P0_B1 |
| 48 Pixels Wide Mode | |
| AddrN: | L0_P47_B0, L0_P46_B0, L0_P45_B0, ... , L0_P13_B0, L0_P32_B0 |
| AddrN+1: | L0_P47_B1, L0_P46_B1, L0_P45_B1, ... , L0_P33_B1, L0_P32_B1 |
| AddrN+2: | L0_P31_B0, L0_P30_B0, L0_P29_B0, ... , L0_P17_B0, L0_P16_B0 |
| AddrN+3: | L0_P31_B1, L0_P30_B1, L0_P29_B1, ... , L0_P17_B1, L0_P16_B1 |
| AddrN+4: | L0_P15_B0, L0_P14_B0, L0_P13_B0, ... , L0_P1_B0, L0_P0_B0 |
| AddrN+5: | L0_P15_B1, L0_P14_B1, L0_P13_B1, ... , L0_P1_B1, L0_P0_B1 |
| AddrN+6: | L0_P47_B0, L0_P46_B0, L0_P45_B0, ... , L0_P33_B0, L0_P32_B0 |
| AddrN+7: | L0_P47_B1, L0_P46_B1, L0_P45_B1, ... , L0_P33_B1, L0_P32_B1 |
| | |

Table 14 Memory Allocation for Cursor Bitmap (Continued)

| | |
|-------------|---|
| AddrN+6n: | Ln_P47_B0, Ln_P46_B0, Ln_P45_B0, ... , Ln_P33_B0, Ln_P32_B0 |
| AddrN+6n+1: | Ln_P47_B1, Ln_P46_B1, Ln_P45_B1, ... , Ln_P33_B1, Ln_P32_B1 |
| AddrN+6n+2: | Ln_P31_B0, Ln_P30_B0, Ln_P29_B0, ... , Ln_P17_B0, Ln_P16_B0 |
| AddrN+6n+3: | Ln_P31_B1, Ln_P30_B1, Ln_P29_B1, ... , Ln_P17_B1, Ln_P16_B1 |
| AddrN+6n+4: | Ln_P15_B0, Ln_P14_B0, Ln_P13_B0, ... , Ln_P1_B0, Ln_P0_B0 |
| AddrN+6n+5: | Ln_P15_B1, Ln_P14_B1, Ln_P13_B1, ... , Ln_P1_B1, Ln_P0_B1 |

where

Lx_Py_Bz = line X, pixel Y, bit Z;
Line 0 = first (top) cursor's line;
Pixel 0 = first (left) cursor's pixel
Bit 1, Bit 0 = most and least significant bits of the cursor's color defined as
00 = transparent
01 = Color 1
10 = Color 2
11 = Color 3

2.14 Color Palette Assignment

The Z90T366 features a total of 16 color palettes, 8 of which are fixed and 8 of which are programmable. Palettes are selected by setting R7(3)<3:0>. Fixed palettes are defined in Table 15.

Table 15 Fixed Palette Color Assignment
(Color0 is Black; Color7 is White)

| Palette | Description | Color1 | | | Color2 | | | Color3 | | | Color4 | | | Color5 | | | Color6 | | |
|---------|-------------------|--------|----|----|--------|----|----|--------|----|----|--------|----|----|--------|----|----|--------|----|----|
| | | R | G | B | R | G | B | R | G | B | R | G | B | R | G | B | R | G | B |
| 0 | Digital RGB | 00 | 00 | 11 | 00 | 11 | 00 | 00 | 11 | 11 | 11 | 00 | 00 | 11 | 00 | 11 | 11 | 11 | 00 |
| 1 | Analog RGB | 00 | 00 | 11 | 00 | 11 | 00 | 00 | 11 | 11 | 11 | 00 | 00 | 11 | 00 | 11 | 11 | 11 | 00 |
| 2 | Greyscale_1 | 01 | 01 | 01 | 10 | 10 | 10 | 11 | 11 | 11 | 00 | 00 | 00 | 01 | 01 | 01 | 10 | 10 | 10 |
| 3 | Greyscale_2 | 00 | 00 | 00 | 01 | 01 | 01 | 01 | 01 | 01 | 10 | 10 | 10 | 10 | 10 | 10 | 11 | 11 | 11 |
| 4 | RGB_Cyan_2Grey | 00 | 00 | 11 | 00 | 11 | 00 | 00 | 11 | 11 | 11 | 00 | 00 | 01 | 01 | 01 | 10 | 10 | 10 |
| 5 | RGB_Magenta_2Grey | 00 | 00 | 11 | 00 | 11 | 00 | 01 | 01 | 01 | 11 | 00 | 00 | 11 | 00 | 11 | 10 | 10 | 10 |
| 6 | RGB_Yellow_2Grey | 00 | 00 | 11 | 00 | 11 | 00 | 01 | 01 | 01 | 11 | 00 | 00 | 10 | 10 | 10 | 11 | 11 | 00 |
| 7 | StarSight | 00 | 11 | 11 | 10 | 11 | 10 | 10 | 10 | 10 | 11 | 01 | 01 | 11 | 11 | 10 | 11 | 11 | 00 |

Programmable palettes (8–15) are mapped to AR0–AR63 (8 registers per palette). The register and bit assignments for Palette # 11 are listed in Figure 24.

Programmable palettes are grouped into 2 banks (palettes 8–11 and 12–15). Palettes in the bank cannot be modified if another palette from the same bank is displayed. An interleaving palette bank access must be created if on-the-fly palette modifications are required. One palette bank is used to display four colors, and the other bank is used for updates. See Figure 24.



Figure 24 Programmable Palette Control at AR Register

2.15 Other Functions

Video and Sound Attribute Control

Basic receiver functions such as color and volume can be controlled directly by six 6-bit pulse-width modulated ports.

InfraRed Capture Function

The Infrared Remote Control data capture feature uses a capture register to hold the time value from one transition of IR data to the next.

Software periodically checks and reads the capture status and the value if a new capture occurs. Subsequent decoding and command passing of the received IR signal is under software control. Figure 25 illustrates the IR input circuit.

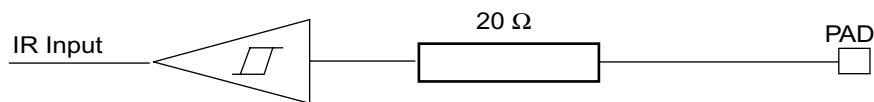


Figure 25 IR Capture Register Input

Loop Filter

The Loop Filter pin configuration is represented in Figure 26.

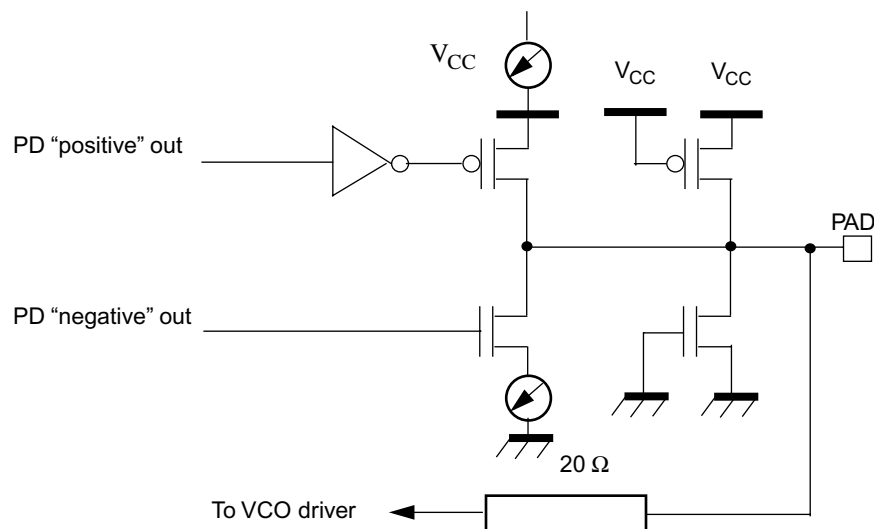


Figure 26 Loop Filter Pin Configuration

Hardware Accelerated 4-Bit and 8-Bit Shifts

Hardware-accelerated byte and nibble shifts significantly reduce software overhead. Shifts are created by assigning one particular RAM location (%1FE) a special meaning. Depending on the R4(1)<e:d> settings, data read from this address are either unmodified, rotated 4 bits left, 4 bits right, or byte swapped. See Table 16.

Table 16 R4(1)<e:d> Settings

| R4(1)<e:d> | Function |
|------------|-------------------------|
| 00 | Direct (unmodified)–POR |
| 01 | 4-bit left rotate |
| 10 | 4-bit right rotate |
| 11 | Byte swap |

Example

```
LD    SR,##%20      ; Select RegBank1
LD    A,EXT4         ; turn "hardware-supported shift" mode on
AND   A, ##%9FFF    ;
OR    A, ##%4000     ;
LD    EXT4, A        ; select "4-bit right rotate"
LD    A, ##%3ED7     ; load A = 3ED7h
LD    %1FE, A        ; write A to the RAM
LD    A, %1FE        ; A = 73EDh
LD    A,EXT4         ; turn "hardware-supported rotate" mode on
OR    A,##%6000      ;
LD    EXT4, A        ; select "byte swap"
LD    A, %1FE        ; A = D73Eh
```

3 Register Groups

Table 17 provides a summary of the registers in external banks.

Table 17 Register Summary

| Bank | Bank Sub Address | READ Register | WRITE Register | Description |
|-------|------------------|-------------------------|----------------|---|
| Bank0 | 7 | dir1 | | 9-bit I/O port 1 direction control |
| | 6 | dir0 | | 16-bit I/O port 0 direction control |
| | 5 | port1 | | 9-bit I/O port 1 |
| | 4 | port0 | | 16-bit I/O port 0 |
| | 3 | I ² C_int | | I ² C interface register |
| | 2 | pll_freq | | PLL frequency control |
| | 1 | write control register | | Cursor palette gauge Write control register |
| | 0 | Reserved | | Reserved |
| Bank1 | 7 | wdt_smr_ctl/Interrupt | | SMR and WDT control and interrupt |
| | 6 | clock_ctl | | Clock control (switch VCO/DOT) |
| | 5 | cap_1s_ctl | | Counter timers control |
| | 4 | ADC_ctl | | A/D converter control |
| | 3 | standard_ctl | | Output H/VSYNC/blink control |
| | 2 | 9-bit counter | STOP/WDT | Stop and WDT instructions, 9-bit counter |
| | 1 | sclk_freq | | Stop/sleep/normal mode |
| | 0 | clamp_pos | | Defines position of video clamp pulse |
| Bank2 | 7 | CGROM offset register | | CGROM offset register |
| | 6 | Shadow Control register | | Defines right and left shadow color |
| | 5 | pwm_data6 | | 8-bit PWM 6 data |
| | 4 | pwm_data5 | | 8-bit PWM 5 data |
| | 3 | pwm_data4 | | 8-bit PWM 4 data |
| | 2 | pwm_data3 | | 8-bit PWM 3 data |
| | 1 | pwm_data2 | | 8-bit PWM 2 data |
| | 0 | pwm_data1 | | 8-bit PWM 1 data |

Table 17 Register Summary (Continued)

| Bank | Bank Sub Address | READ Register | WRITE Register | Description |
|-------|------------------|----------------|------------------------------|---|
| Bank3 | 7 | output_palette | | Output palette |
| | 6 | palette_color | | Display palette color/underline color |
| | 5 | capture_data | I ² C slave addr. | Capture register data |
| | 4 | osd_control | | On screen display control |
| | 3 | attribute_data | vram_data | Character attribute/video RAM data |
| | 2 | ch_x1_lo_x3 | cg_attribute | Character multiple/character graphics attribute |
| | 1 | lo_x2_mid_x3 | cg_nxt_prv | Character multiple/next or previous data |
| | 0 | hi_x2_hi_x3 | cg_current | Character multiple/current data |

3.1 Register Description

The register file in the Z90T366 is organized into four banks that can be selected by writing to bits 5 and 6 (Register Bank Selector bits) in the Status Register of the Z90T366 core.

All registers are mapped into an external register space; each bank consists of 8 registers. The Status register is available to read or write at any time. The appropriate bank of registers must be selected before accessing the register. The software must keep track of which register bank is accessible at any time. Refer to Table 18 for register bank assignments.

Table 18 Bank Assignments

| Bank | Status Register | Bank Functions |
|-------|-----------------------|--|
| Bank0 | xxxx xxxx x00x xxxx b | I/O ports, I ² C interface, PLL frequency, cursor |
| Bank1 | xxxx xxxx x01x xxxx b | Control registers |
| Bank2 | xxxx xxxx x10x xxxx b | PWM1–PWM5 |
| Bank3 | xxxx xxxx x11x xxxx b | OSD, palette control |

3.2 Bank0 (I/O Ports, I²C Interface, PLL Frequency, Cursor) Control Registers

Table 19 defines the bits for Register1, R1(0) Cursor Palette Control Register.
Table 20 defines the bits for Register2, R2(0) PLL Frequency Data Register.

Table 19 Register1, Bank 0, Cursor Palette

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|-------|----------|----------|-----|-----|-----|-----|-----|-----|
| R/W | reserved | reserved | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | x | x | 0 | 0 | x | x | x | x |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | x | x | x | x | x | x | x | x |

Note: R = Read W = Write X = Indeterminate

| Reg Field | Bit Position | R | W | Data | Description |
|----------------|------------------|---|---|--------|---|
| Reserved | f e----- | | | | Reserved |
| Cursor Palette | --d----- | R | W | 1 0 | Palette #6 (recommended) Current palette—POR |
| PgWritenEn | ---c----- | R | W | 1 0 | Page Write Enable Page Write Disable—POR |
| PageWrite | ----ba9876543210 | R | W | xxxx | PageWrite# |

Table 20 Register2, Bank 0, PLL Frequency Data Register

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|-------|-----|-----|-----|-----|-----|----------|-----|-----|
| R/W | R/W | R/W | R/W | R/W | R/W | reserved | R/W | R/W |
| Reset | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |

Note: R = Read W = Write X = Indeterminate

| Reg Field | Bit Position | R | W | Data | Description |
|-------------------------------------|---------------|---|---|--------|--|
| M_disable | f----- | R | W | 1 0 | I ² C Master interface disabled I ² C Master interface enabled–POR |
| S_disable | -e----- | R | W | 1 0 | I ² C Slave interface disabled I ² C Slave interface enabled–POR |
| I ² CM_mux | --d----- | R | W | 1 0 | Select I ² MSD2, I ² MSC2–POR Select I ² MSD1, I ² MSC1 |
| I ² C_Out_ Resistance | ---c----- | R | W | 1 0 | 600Ω output resistance Normal CMOS port output resistance–POR |
| I ² C_speed_ _range | ----b----- | R | W | 1 0 | Low speed range (10 KHz, 50 KHz) High speed range (100 KHz, 400 KHz)–POR |
| Reserved | -----a----- | | | | Reserved |
| P46/ HSYNC2 | -----9----- | R | W | 10 | HSYNC logic takes input from HSYNC pin–POR. This bit must always be set to 0. |
| P07/ CSYNC | -----8----- | R | W | 1 0 | Composite Sync Output P07 I/O–POR |
| PLL_data | -----76543210 | R | W | D | PLL divider = 256 + D |

If the master I²C interface is enabled, the corresponding I/Os, Port11 and Port12 must be assigned as outputs. (The Z90T366 does not have pins to support a slave interface.)

The VCO, DOT, and SCLK frequency are defined as the following:

$$F_{VCO} = F_{DOT} = F_{SCLK} = XTAL * (256 + PLL_{DATA})$$

Therefore, XTAL = 32.768 KHz

At POR, the PLL frequency data register is preset to 70h, which corresponds to the VCO frequency of 12.058 MHz.

The PLL_data field can be loaded with any value from 00h. This value corresponds to an SCLK = 256*XTAL up to FFh, which corresponds to an SCLK = 511*XTAL.

Table 21 through Table 25 describe the bits in registers 3 through 7.

Table 21 Register3, BAnk 0, I²C Interface Register

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|-------|-----|-----|-----|-----|-----|-----|-----|-----|
| R/W | W | W | W | R/W | W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 1 | x | x |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | x | x | x | x | x | x | x | x |

Note: R = Read W = Write X = Indeterminate

| Reg Field | Bit Position | R | W | Data | Description |
|------------|---------------|---|---|------|---|
| Command | fed----- | | W | D | See Table 9 and Table 10 for a full description. |
| Toggle | ---c----- | R | | 1 | Slave interface |
| | | | | 0 | Master interface--POR condition |
| | | | W | 1 | Toggle active I ² C interface |
| | | | | 0 | No effect |
| Reset | ----b----- | W | | 1 | Reset Slave I ² C interface if bit <c> =1 |
| | | | | | Reset Master I ² C interface if bit <c> =0 |
| | | | | 0 | No effect |
| Slave_mode | -----a----- | R | | 1 | Slave mode is active (POR condition) |
| | | | | 0 | Slave mode is inactive |
| SlaveBusy | -----9----- | R | | 1 | Slave I ² C interface is busy |
| | | | | 0 | Slave I ² C interface is idle |
| MasterBusy | -----8----- | R | | 1 | Master I ² C interface is busy |
| | | | | 0 | Master I ² C interface is idle |
| Data | -----76543210 | R | W | xx | Received data |
| | | | | xx | Data to be sent |

Data written to R3(0)<cb> requires 4 cycles before being applied. Consecutive writings to these bits require at least a 6-cycle delay.

The received data is available for reading only when the “busy” bit is reset to a “0.” When POR, the speed of the I²C interface is set to “Low.”

Table 22 Register4, Bank 0, Port 0 Data Register

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|-------|-----|-----|-----|-----|-----|-----|-----|-----|
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | x | x | x | x | x | x | x | x |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | x | x | x | x | x | x | x | x |

Note: R = Read W = Write X = Indeterminate

| Reg Field | Bit Position | R | W | Data | Description |
|-----------|------------------|---|---|------|---|
| Port_data | fedcba9876543210 | R | | xxxx | If a port is configured in Input mode, enter the input data onto the port pins. |
| | | | W | xxxx | If a port is configured in Output mode, then the data is written directly to the port data. |

Table 23 Register5, Bank 0, Port 1 Data Register

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|-------|----------|----------|----------|----------|----------|----------|----------|-----|
| R/W | reserved | reserved | reserved | reserved | reserved | reserved | reserved | R/W |
| Reset | x | x | x | x | x | x | x | x |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | x | x | x | x | x | x | x | x |

Note: R = Read W = Write X = Indeterminate

| Reg Field | Bit Position | R | W | Data | Description |
|-----------|----------------|---|---|------|---|
| Reserved | fedcba9----- | | | | Reserved |
| Port_data | -----876543210 | R | | xxxx | If a port is configured in Input mode, enter the input data onto the port pins. |
| | | | W | xxxx | If a port is configured in Output mode, then the data is written directly to the port data. |

Table 24 Register6, Bank 0, Port 0 Direction Register

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|-------|-----|-----|-----|-----|-----|-----|-----|-----|
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

Note: R = Read W = Write X = Indeterminate

| Reg Field | Bit Position | R | W | Data | Description |
|----------------|------------------|---|---|------|---|
| Port_direction | fedcba9876543210 | R | W | xxxx | 1: Input mode for corresponding bit 0: Output mode for corresponding bit |

Table 25 Register7, Bank 0, Port 1 Direction Register

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|-------|----------|----------|----------|----------|----------|----------|----------|-----|
| R/W | reserved | reserved | reserved | reserved | reserved | reserved | reserved | R/W |
| Reset | x | x | x | x | x | x | x | 1 |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

Note: R = Read W = Write X = Indeterminate

| Reg Field | Bit Position | R | W | Data | Description |
|----------------|----------------|---|---|------|---|
| Reserved | fedcba9----- | | | | Reserved |
| Port_direction | -----876543210 | R | W | xxxx | 1: Input mode for corresponding bit 0: Output mode for corresponding bit |

3.3 Bank1 (Control Registers)

Table 26 through Table 33 provide bit functions for Bank 1 Control registers.

Table 26 Register0, Bank 1, Clamp Position Register

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|-------|----------|-----|-----|-----|-----|----------|----------|----------|
| R/W | R/W | R/W | R/W | R/W | R/W | reserved | reserved | reserved |
| Reset | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R/W | reserved | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | x | x | x | x | x | x | x |

Note: R = Read W = Write X = Indeterminate

| Reg Field | Bit Position | R | W | Data | Description |
|-------------------|----------------|---|---|--------|--|
| Disable_clamp_1 | f----- | R | W | 1 0 | ADC0 Clamp generation is disabled ADC0 Clamp generation is enabled |
| Disable_clamp_2 | -e----- | R | W | 1 0 | ADC5 Clamp generation is disabled ADC5 Clamp generation is enabled |
| Disable_tip_clamp | --d----- | R | W | 1 0 | ADC0 Tip clamp is disabled--POR ADC0 Tip clamp is enabled |
| Counter_input | ---c----- | R | W | 1 0 | Counter takes input from P06--POR Counter takes input from internal HSYNC Separator |
| ARenable | ----b----- | R | W | 1 0 | AR enabled AR disabled--POR |
| Reserved | -----a987----- | | | | Reserved |
| Position | -----6543210 | R | W | xx | Position of clamp pulse (from leading edge of the H-FLYBACK) |

At POR the disable_clamp bit is set to 1.

The clamp pulse is generated if Enabled (bit <f>) and the SCLK frequency is switched back to PVCO. The SVCO/PVCO flag in R6(1) must be reset to 0 before the current H_{SYNC}, regardless of whether the SVCO is enabled or disabled.

The clamp position is defined by the Position field. The width of the clamp pulse cannot be modified and is set to 1μs. The value that can be assigned to the "Position" field must be >10h and <7Fh. The time interval between the leading edge of the H-FLYBACK and the beginning of the clamp pulse can be calculated from the following equation:

$$T_{\text{DELAY}} = \text{Position} \left(\frac{1}{T_{\text{SCLK}}} \right) = \text{Position} \times 82\text{ns}$$

Table 27 Register1, Bank 1, Speed Control Register

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|-------|----------|-----|-----|-----|-----|-----|-----|-----|
| R/W | reserved | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R/W | W | W | W | R/W | R/W | R/W | R/W | R/W |
| Reset | x | x | x | 0 | 0 | 0 | 0 | 0 |

Note: R = Read W = Write X = Indeterminate

| Reg Field | Bit Position | R | W | Data | Description |
|------------------|---------------|---|---|--------|---|
| Reserved | f----- | | | | Reserved |
| Mode selection | -e----- | R | W | 1 0 | Standard interlace mode (single scan) Double scan -POR |
| H_SHIFT | --dcba98----- | R | W | D | Horizontal delay* POR = 0 |
| 1xHSYNC | -----7----- | | W | 1 0 | 1xHSYNC connected to Port03 1xHSYNC is 2xHSYNC/2-POR |
| Skip_HSYNC | -----6----- | | W | 1 0 | Skip next HSYNC Do not skip next HSYNC |
| Frame_start | -----5----- | | W | 1 0 | Field start initializaiton No effect |
| OSD_black | -----4---- | R | W | 1 0 | Next output line is OSD Next output line is black |
| Line_buffer_mode | -----3---- | R | W | 1 0 | Interlaced(OSD/black) Progressive (OSD/OSD)-POR |
| 2x_RGB | -----2-- | R | W | 1 0 | Double RGB output Normal RGB output-POR |
| Fast_enable | -----1- | R | W | 1 0 | PVCO/SVCO enabled PVCO/SVCO disabled-POR |
| Fast_slow | -----0 | R | W | 1 0 | SCLK is 12.058 MHz SCLK is 32.768 KHz-POR |

Note: * 1 step = 4 pixels

When a POR, SMR, or WDT reset occurs, both the Fast_enable and Fast/Slow are reset to 0. This event corresponds to an SCLK frequency of 32.768 KHz.

To switch from a 32.768 KHz SCLK to 12 MHz, use the following procedure:

1. Be sure the registerer bank is set to Bank 1 with the following settings.

```
CTL_BANK      EQU    %0020
              LD      SR, #CTL_BANK
```

2. Set the H_Position field R6(1)<3:0> to a nonzero value.

```
HSYNCH_DELAY_MIN EQU    %0001
              LD      CLK_CONTROL, #HSYNCH_DELAY_MIN
```

3. Enable the primary and secondary VCOs by setting by Fast_enable bit R1(1)<1> to 1.

```
VCO_ENABLE     EQU    %0002
              LD      SCLK_FREQ, #VCO_ENABLE
```

4. Wait one second for the 12 MHz PLL to stabilize (about 33,000 clock cycles at 32.768 KHz). The delay depends on the external Pll filter and can vary significantly.

```
              LD      A, #11000
VCO_DELAY:
              SUBA, #1
              JP      NZ, VCO_DELAY
```

5. Switch the SCLK to a fast clock by setting Fast/Slow bit R(1)<0> to 1.

```
VCO_ENABLE     EQU    %0002
FAST_CLK       EQU    %0001
              LD      SCLK_FREQ, #(VCO_ENABLE|FAST_CLK)
```

The following code is the complete code to switch from 32.768 KHz clock to 12 MHz clock:

```
CTL_BAN        EQU    %0020
VCO_ENABLE     EQU    %0002
FAST_CLK       EQU    %0001
HSYNCH_DELAY_MIN EQU    %0001
;
; Set register bank to Bank 1
```



```

;
LD      CLK_CONTROL, #HSYNCH_DELAY_MIN
;
; Change the system clock rate from 32 KHz up to 12 MHz
; before switching over
; 11000 * 3 * 32µSec ~= 1 second
;
LD      A, #11000
VCO_DELAY:
SUB     A, #1
JP      NZ, VCO_DELAY
LD      SCLK_FREQ, #(VCO_ENABLE | FAST_CLK)
;

```

To switch from the 12 MHz SCLK to 32.768 KHz, use the following procedure:

1. Switch the SCLK to a 32.768 KHz clock (set Fast/Slow bit R1(1)<0> to 0).
2. Wait for more than R2(0)<7:0> + 256 clock cycles (approximately 32 µS) for the SCLK to be switched.
3. Set the HSYNC_DELAY field R6(1)<3:0> to 0FH.
4. Disable the primary and secondary VCOs (set the Fast_enable bit R1(1)<1> to 0).

If R1(1)<3> is set to “1” (Interlaced OSD/Black), the interleaving of OSD and Black can be controlled by writing “1” or “0” into R1(1)<4>; this must be done once every V_{SYNC}.

Table 28 Register2, Bank1, WDT/STOP (write only) and 9-bit Counter (read only) Control Register

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|-------|----|----------|----------|----------|----------|----------|---|---|
| R/W | R | R | R | R | R | R | R | R |
| Reset | x | x | x | x | x | x | x | x |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R/W | R | reserved | reserved | reserved | reserved | reserved | W | W |
| Reset | x | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Note: R = Read W = Write X = Indeterminate

| Reg Field | Bit Position | R | W | Data | Description |
|---------------|----------------|---|---|--------|------------------------------------|
| Counter_value | fedcba987----- | R | | | Counter on Port06 value |
| Reserved | -----65432-- | | | | Reserved |
| WDT_instr | -----1- | | W | 1 0 | WDT enable, WDT reset No effect |
| STOP_instr | -----0 | | W | 1 0 | Stop No effect |

When a POR, SMR or a WDT reset occurs, the WDT is disabled. The WDT can be reenabled only after the PVCO and SVCO are enabled, and the part is switched into a Fast mode (SCLK = 12 MHz).

When switching the part into a SLOW mode (SCLK = 32.768 KHz), the WDT halts. To return to Fast mode, the WDT must be initialized again.

Table 29 Register3, Bank 1, Standard Control Register

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|-------|-----|-----|-----|-----|-----|-----|-----|-----|
| R/W | W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | x | x | x | 0 | 0 |

Note: R = Read W = Write X = Indeterminate

| Reg Field | Bit Position | R | W | Data | Description |
|------------------------------|--------------|---|---|------------------------------|--|
| Counter_reset | f----- | | W | 1 0 | Reset Counter on Port06 No effect |
| Counter_ON/OFF | -e----- | R | W | 1 0 | Counter on Port06 is ON Counter is OFF-POR condition |
| Mask_HVSYNC | --d----- | R | W | 1 0 | Disable HVSYNC output HVSYNC IN/OUT-POR condition |
| Char_size_16_18/20 | ---c----- | R | W | 1 0 | 16x20 character matrix 16x16 or 16x18 character matrix-POR |
| Bank0_sel | ----ba----- | R | W | 00 01 10 11 | RAM Bank 00-POR RAM Bank 01 RAM Bank 02 Reserved |
| RGBC/Port1 | -----9----- | R | W | 1 0 | SCLK, R1, R0, G1, G0, B1, B0 P16,P08,P10,P13,P18,P15,P14 |
| I ² C_HI/LO_speed | -----8----- | R | W | 1 0 | HI speed (400/50 KHz) LO speed (100/10 KHz)-POR |
| CGROM bank | -----7----- | R | W | 1 0 | Bank1 is selected (starts @1000h) Bank0 is selected (starts @0000h) |
| SOVL/Port0F | -----6----- | R | W | 1 0 | Semi-transparency P0f output |
| OSD_on/off | -----5----- | R | W | 1 0 | OSD is enabled OSD is disabled-POR |
| RGB_polarity | -----4----- | R | W | 1 0 | Negative Positive |
| Positive/Negative | -----3----- | R | W | 1 0 | Negative HVSYNC in output mode Positive HVSYNC in output mode |
| SYNC/OVL | -----2----- | R | W | 1 0 | HVOVL outputs HVSYNC outputs |
| 25/30_Hz and HV_polarity | -----10----- | R | W | 10 00 11 01 | Internal mode only (TV Standard) 50 Hz/625 lines support 60 Hz/525 lines support-POR External mode only (HV Polarity) Positive Negative |

Two bits define the polarity of the HVS SYNC signals. Bit <3> defines the polarity of the signals when they are configured as outputs (it does not affect internal HV–SYNC signals). Bit <1> defines the polarity of the external HV–SYNC signals which affect the device synchronization.

- **Note:**
1. The composite SYNC is active in internal mode only.
 2. When using the internally-generated COMPOSITE SYNC signal, be sure the SCLK is set to 12.09 MHz (R2(0)<7:0> = 71h). This action helps ensure the best HSYNC frequency approximation.

Table 30 Register 4, Bank 1, ADC Control Register

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|-------|----------|-----|-----|-----|-----|-----|-----|----------|
| R/W | reserved | R/W | R/W | R/W | R/W | R/W | R/W | reserved |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R/W | R/W | R/W | R/W | R/W | R | R | R | R |
| Reset | 0 | 0 | 0 | 0 | x | x | x | x |

Note: R = Read W = Write X = Indeterminate

| Reg Field | Bit Position | R | W | Data | Description |
|-------------|--------------|---|---|----------------------|--|
| Reserved | f----- | | | | Reserved |
| HW_shift | -ed----- | R | W | 00 01 10 11 | Direct (unmodified)–POR 4-bit shift left 4-bit shift right Byte swap |
| HSYNC_edge* | ---c----- | R | W | 1 0 | HSYNC is leading edge active HSYNC is trailing edge active–POR |
| ADC_ref | ----b----- | R | W | 1 0 | All ADCs use AV _{CC} and AGND as reference voltage All ADCs use 2.0V and 1.5V as reference voltage–POR |
| ADC_select | -----a----- | R | W | 1 0 | ADC4, ADC5 select ADC0, ADC1, ADC2, ADC3 select–POR condition |

| Reg Field | Bit Position | R | W | Data | Description |
|-----------------|--------------|---|---|----------------------|--|
| ADCdata-Packing | -----9----- | R | W | 1 0 | ADC data packing is on ADC data packing is off—POR |
| Reserved | -----8----- | | | | Reserved |
| ADCspeed | -----76----- | R | W | 00 01 10 11 | Single conversion—POR condition SCLK/4 SCLK/6 SCLK/8 |
| ADCsource | -----54----- | R | W | 00 01 10 11 | ADC0 (CVI)/ADC4 (P04)—POR ADC1 (P17)/ADC5 ADC2 (P00) ADC3 (P05) |
| ADCdata | -----3210 | R | | | ADC data |

Note: *If HSYNC is Leading Edge Active (R4(1)<c> = 1),the actual interrupt is delayed from the leading edge of HSYNC by 72 cycles (~6μS @12 MHz).

ADC0 has a signal range from 1.5 to 2.0 V. This field is always connected to the Composite Video Input pin and can be clamped to a Ref– voltage (1.5V).

ADC1, ADC2, ADC3, and ADC4 have a signal range from 0 to 5.0 V.

ADC5 features a signal range from 1.5 to 2.0 V. For this field, the input signal can be clamped to a Ref+ voltage (2.0V).

To use the I/O pin as an ADC input, the corresponding port must be set up as an input (refer to R4(0) and R6(0)).

Table 31 Register5, Bank 1,Timer Control Register

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|-------|-----|-----|-----|-----|----------|----------|-----|-----|
| R/W | R/W | R/W | R/W | R/W | reserved | reserved | R/W | R/W |
| Reset | x | x | x | x | 0 | 0 | x | x |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | x | x | x | x | x | x | x | x |

Note: R = Read W = Write X = Indeterminate

| Reg Field | Bit Position | R | W | Data | Description |
|---------------------|--------------|---|---|------|------------------------------|
| CAPint_r | f----- | R | | 1 | Rising edge is captured |
| | | | | 0 | No rising edge is captured |
| | | | W | 1 | Reset flag |
| | | | | 0 | No effect |
| CAPint_f | -e----- | R | | 1 | Falling edge is captured |
| | | | | 0 | No falling edge is captured |
| | | | W | 1 | Reset flag |
| | | | | 0 | No effect |
| Tout_1s | --d----- | R | | 1 | Timeout of 1s timer |
| | | | | 0 | No timeout of 1s timer |
| | | | W | 1 | Reset flag |
| | | | | 0 | No effect |
| Tout_CAP | ---c----- | R | | 1 | Timeout of Capture timer |
| | | | | 0 | No timeout of Capture timer |
| | | | W | 1 | Reset flag |
| | | | | 0 | No effect |
| Reserved | ----ba----- | | | | Reserved |
| Speed_1s | -----98----- | R | W | 00 | 1s |
| | | | | 01 | 250 ms |
| | | | | 10 | 62.5 ms |
| | | | | 11 | 15.625 ms |
| Port09/ CAP_int* | -----7----- | R | W | 1 | int2 source is Port09 |
| | | | | 0 | int2 source is Capture timer |
| CAP_halt* | -----6----- | R | W | 1 | Capture timer is halted |
| | | | | 0 | Capture timer is running |
| CAP_edge* | -----54----- | R | W | 00 | No Capture |
| | | | | 01 | Capture on rising edge only |
| | | | | 10 | Capture on falling edge only |
| | | | | 11 | Capture on both edges |
| CAP_glitch* | -----32--- | R | W | 00 | Glitch filter is disabled |
| | | | | 01 | <8TSCLK is filtered out |
| | | | | 10 | <32TSCLK is filtered out |
| | | | | 11 | <128TSCLK is filtered out |

| Reg Field | Bit Position | R | W | Data | Description |
|-------------|--------------|---|---|------|-------------|
| CAP_speed * | -----10 | R | W | 00 | SCLK/4 |
| | | | | 01 | SCLK/8 |
| | | | | 10 | SCLK/16 |
| | | | | 11 | SCLK/32 |

*Resetting a Capture Timer flag does not modify Capture Counter or Capture register data. When the glitch filter is enabled, the duration of the pulse is decreased by the CAP_glitch value.

Table 32 Register6, Bank 1, Clock Switch Control Register

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|-------|----------|----------|----------|----------|----------|----------|----------|----------|
| R/W | reserved | reserved | reserved | reserved | reserved | reserved | reserved | reserved |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R/W | reserved | reserved | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 1 | x | x | x | x |

Note: R = Read W = Write X = Indeterminate

| Reg Field | Bit Position | R | W | Data | Description |
|------------|-----------------|---|---|------|--|
| Reserved | fedcba9876----- | | | | Reserved |
| SVCO/PVCO | -----5----- | R | | 1 | SCLK = SVCO (flag) |
| | | | | 0 | SCLK = PVCO (flag) POR |
| | | | W | 1 | Switch SCLK to PVCO |
| | | | | 0 | No effect |
| No_Switch | -----4----- | R | W | 1 | SCLK = PVCO, no clock switching—POR |
| | | | | 0 | Clock switching is enabled |
| H_Position | -----3210 | R | W | D | Defines delay of H _{SYNC} interrupt by 4X SCLK cycles |

H_Position

The H_position field must be set to "F". The actual horizontal position adjustment is controlled by R1(1)<c:7>.

No_Switch

The No_switch bit determines if the system clock is permanently set to the Primary VCO (PVCO) or allowed to switch between PVCO and the Secondary VCO (SVCO). This bit is set to 1 (NO clock switching) at power up reset.



Caution: After the system has been switched to fast (12 MHz) clock both signals feeding into this switch **MUST** be PVCO **BEFORE** the switch setting is changed. Otherwise a short system clock can result which causes the processor to run at a higher frequency than specified. The instruction fetched from memory, at the location with the out-of-spec frequency, can be corrupted!

To ensure safe clocks, the following practices are recommended:

1. Set the No_switch to the required setting before switching from the 32.768 KHz to the fast (12 MHz) clock and leave it there permanently.
2. Use the following procedure when changing from Switching VCO to permanent PVCO while running from the fast clock:
 - Simultaneously set the H_position delay to 0x0, while leaving the No_switch enabled (0), and the SVCO/PVCO left as (0).
 - Wait a minimum of 80 clock cycles to flush any H_SYNC out of the system.
 - Simultaneously Switch SVCO/PVCO to PVCO (write 1 into R6(1)<5>), while leaving the No_switch enabled (0), and the H_position delay at 0x0.
 - Wait 3 system clock cycles to be sure that the clock has had time to switch to PVCO.
 - Switch No_switch to No clock switch (write 1 into R6(1)<6>). The H_position can be set to none-zero at this time as well.
3. Use the following procedure when changing from permanent PVCO to Switching VCO while running from the fast clock.
 - Simultaneously set the H_position delay to 0x0, while leaving the No_switch disabled (1), and the SVCO/PVCO setting as don't care (0).

- Wait a minimum of 80 clock cycles to flush residual H-sync out of the system.
- Simultaneously switch SVCO/PVCO to PVCO (write 1 into R6(1)<5>), while leaving the No_switch disabled (1), and the H_position delay at 0x0.
- Wait 3 system clock cycles to be sure that the clock has had time to switch to PVCO.
- Switch No_switch to Clock switching is Enabled (write 0 into R6(1)<6>). The H_position can be set to none-zero at this time as well.

SVCO/PVCO

The SVCO/PVCO bit when read back determines the current setting of the system clock. Writing a 1 to this bit switches the system clock from its current setting to PVCO. This switch has a glitch filter that removes random voltage spikes. It must be changed back to PVCO. The Z90T366 switches to the SVCO automatically when it receives an H-sync interrupt. This mechanism exists to synchronize the system clock exactly with the H-sync trailing edge. The result is a sharp start of the OSD, jitter free.

An example of a typical SVCO/PVCO switching follows:

1. System clock is set to PVCO.
2. H-sync interrupt occurs. (The system clock has automatically been set to the SVCO). OSD code is executed inside of the H-sync Interrupt Service Routine (ISR).
3. Before leaving the ISR, the user switches the clock back to PVCO.

The clamp pulse (defined in R0(1)) is generated only if the SVCO/PVCO switch is set to PVCO before receiving an H_{SYNC}. The software provides the correct switch setting before every H_{SYNC}.

Table 33 lists the interrupt/WDT for the WST/SMR control register.

Table 33 Register7, Bank 1, Interrupts/WDT/SMR Control Register

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|-------|-----|-----|-----|-----|-----|-----|-----|-----|
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | x | x | x | x | x | x | x | x |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | x | x | x | x | x | x | x |

Note: R = Read W = Write X = Indeterminate

| Reg Field | Bit Position | R | W | Data | Description |
|--------------|--------------|---|---|--|---|
| Int_priority | fed----- | R | W | x | See Table 34. |
| Int_mask | ---cba----- | R | W | 1xx 0xx x1x x0x xx1 xx0 | int2 is enabled int2 is disabled int1 is enabled int1 is disabled int0 is enabled int0 is disabled |
| WDTspeed | -----98----- | R | W | 00 01 10 11 | 1.83 ms 7.68 ms 31.12 ms 124.8 ms |
| SMRflag | -----7----- | R | | 0 1 | No Stop-Mode Recovery–POR Stop-Mode Recovery |
| SMR polarity | -----6----- | R | W | 0 1 | OR of all SMR sources NAND of all SMR sources |
| SMRsource | -----543210 | R | W | xx | Bit which corresponds to a “1” in xx binary representation is active |
| smr5 | -----5----- | | | | p09 |
| smr4 | -----4----- | | | | p14 |
| smr3 | -----3----- | | | | p13 |
| smr2 | -----2----- | | | | p12 |
| smr1 | -----1----- | | | | p11 |
| smr0 | -----0----- | | | | p10 |

The final result of the Stop-Mode Recovery (SMR) is RESET. Ports selected for SMR must be assigned as inputs, while the other SMR ports must be assigned as outputs exhibiting an inactive value. If any SMR source is active, and the Stop Mode is executed, the part resets immediately.

All core interrupts are set to interrupt0 > interrupt1 > interrupt2. These priorities *cannot* be changed and are embedded into the core. However, Z90T366 architecture provides flexibility to change the priority of the interrupts by switching the interrupt sources between interrupt inputs of the Z90T366 core. The correspondence between H_{SYNC}, V_{SYNC} and 1s/CAP interrupts sources, and int0, int1, and int2 interrupts inputs of the Z90T366 are listed in Table 34.

Table 34 Interrupt Priority

| Int_Priority Field | H _{SYNC} Is Switched To: | V _{SYNC} Is Switched To: | 1s/CAP Is Switched To: |
|--------------------|-----------------------------------|-----------------------------------|------------------------|
| 0 0 0 | int0 | int1 | int2 |
| 0 0 1 | int0 | int2 | int1 |
| 0 1 0 | int1 | int0 | int2 |
| 0 1 1 | int2 | int0 | int1 |
| 1 0 0 | int1 | int2 | int0 |
| 1 0 1 | int2 | int1 | int0 |

3.4 Bank2 (PWM Registers)

Table 35 lists the bits for the PWM registers.

Table 35 Register0–Register5, Bank 2, PWM 1–6 Registers

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|-------|----------|----------|----------|----------|----------|----------|----------|----------|
| R/W | reserved | reserved | reserved | reserved | reserved | reserved | reserved | reserved |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | x | x | x | x | x | x | x | x |

Note: R = Read W = Write X = Indeterminate

| Reg Field | Bit Position | R | W | Data | Description |
|-----------|---------------|---|---|------|----------------|
| Reserved | fedcba98----- | | | | Reserved |
| PWM_data | -----76543210 | R | W | xx | 8-bit PWM data |

All of the PWMs feature push-pull. Outputs from all PWMs are staged by one PVCO clock. The repetition frequency of the PWM output signals can be calculated from the following equation:

$$F_{\text{PWM}} = \frac{F_{\text{PVCO}}}{8 - 256} = \frac{12\text{MHz}}{2048} = 6\text{kHz}$$

When reset, PWM_data registers are not initialized; however, PWM output is set to 0. Because the PWM is clocked with PVCO, it is better to initialize the PWM_data before enabling PVCO.

Table 36 lists the bits for the shadow control register.

Table 37 lists the bits for the CGROM offset register.

Table 36 Register6, Bank 2, Shadow Control Register

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|-------|----------|----------|----------|----------|----------|----------|----------|----------|
| R/W | reserved | reserved | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R/W | reserved | reserved | reserved | reserved | reserved | reserved | reserved | reserved |
| Reset | x | x | x | x | x | x | x | x |

Note: R = Read W = Write X = Indeterminate

| Reg Field | Bit Position | R | W | Data | Description |
|---------------|---------------|---|---|------|----------------------------|
| Reserved | f e----- | | | | Reserved |
| Lshadow-Color | --dcb----- | R | W | xx | Left shadow color-POR = 0 |
| Rshadow-Color | -----a98----- | R | W | xx | Right shadow color-POR = 0 |
| Reserved | -----76543210 | | | | Reserved |

Table 37 Register7, Bank 2, CGROM Offset Register

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|-------|-----|-----|-----|-----|-----|-----|-----|-----|
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Note: R = Read W = Write X = Indeterminate

| Reg Field | Bit Position | R | W | Data | Description |
|-----------|------------------|---|---|------|---------------------|
| CGoffset | fedcba9876543210 | R | W | xxxx | CGROM offset, POR=0 |

3.5 Bank3 (On Screen Display [OSD] registers)

Table 38 lists the R0(3)- R2(3) character multiplier registers (Read operation).

Table 39 lists the R0(3)–R1(3) Shift Registers (Write operation).

Table 40 lists the R2(3) Attributes Register (Write operation).

Table 38 Register0–Register2, Bank 3, Read Operation, Character Multiple Registers

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|-------|-----|-----|-----|-----|-----|-----|-----|-----|
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | x | x | x | x | x | x | x | x |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | x | x | x | x | x | x | x | x |

Note: R = Read W = Write X = Indeterminate

| Reg Name | CGROM Data | Reg Add | Description |
|--------------|------------------|---------|--|
| cgrom_x2_hi | feedddccbbaa9988 | R0(3) | High word of double size character R4(3)<6> = 0 |
| cgrom_x3_hi | fffeeedddccbbba | | High word of triple size character R4(3)<6> = 1 |
| cgrom_x2_lo | 7766554433221100 | R1(3) | Low word of double size character R4(3)<6> = 0 |
| cgrom_x3_mid | aa99988877766655 | | Middle word of triple size character R4(3)<6> = 1 |
| cgrom_x1 | fedcba9876543210 | R2(3) | Single size character R4(3)<6> = 0 |
| cgrom_x3_lo | 5444333222111000 | | Low word of triple size character R4(3)<6> = 1 |

Table 39 Register0–Register1, Bank 3, Write Operation, Shift Registers

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|-------|-----|-----|-----|-----|-----|-----|-----|-----|
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | x | x | x | x | x | x | x | x |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | x | x | x | x | x | x | x | x |

Note: R = Read W = Write X = Indeterminate

| Reg Name | CGROM Data | Reg Address | Description |
|-------------------|------------------|-------------|-----------------------------------|
| current_reg | fedcba9876543210 | R0(3) | current line shift register |
| next/previous_reg | fedcba9876543210 | R1(3) | next/previous line shift register |

Registers R1(3) and R0(3) must be loaded with video data one time every 16 cycles. To support smoothing, register R1(3) must be updated every 16 cycles. The current line register is loaded first, followed by next/previous register during the next cycle. The next/previous register is loaded only if smoothing/fringing attributes are activated for the current character. If neither register is loaded, the space character is displayed. There is no difference between loading 0000h into either register or not loading at all.

Table 40 Register2, Bank 3, Attributes Register, Write Operation

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|-------|----------|-----|-----|-----|-----|-----|-----|-----|
| R/W | reserved | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | x | x | x | x | x | x | x | x |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | x | x | x | x | x | x | x | x |

Note: R = Read W = Write X = Indeterminate

| Reg Field | Bit Position | Data | Description |
|--------------------------------------|--------------|---|-------------------------|
| Reserved | f----- | | Reserved |
| Background color | -edc----- | 000 Black 001 Blue 010 Green 011 Cyan 100 Red 101 Magenta 110 Yellow 111 White | |
| Foreground color NOT Palette Mode | ----ba9----- | | Same as Background mode |
| Palette selection Palette Mode | ----ba----- | 00 Palette0 01 Palette1 10 Palette2 11 Palette3 | |
| 2nd_underline Palette Mode | -----9----- | 1 Second Underline is active 0 Second Underline is NOT active | |
| 1st_underline | -----8----- | 1 First Underline is active 0 First Underline is NOT active | |

| Reg Field | Bit Position | Data | Description |
|------------------|--------------|------|--|
| Shift_video | -----7----- | 1 | Video signal is delayed by 8 pixels |
| | | 0 | Standard character positioning |
| Semi-Transparent | -----6----- | 1 | Semi-Transparent background |
| | | 0 | Background color defined by "background color" field |
| Blinking | -----5----- | 1 | Blinking character |
| | | 0 | Not blinking character |
| Italic | -----4----- | 1 | Italic character |
| | | 0 | Not italic character |
| Color_delay | -----32-- | 00 | Character color changes instantly |
| | | 01 | Color changes with 4 pixels delay |
| | | 10 | Color changes with 8 pixels delay |
| | | 11 | Color changes with 12 pixels delay |
| Shadowing | -----10 | 00 | No shadowing |
| | | 01 | Left shadow |
| | | 10 | Right shadow |
| | | 11 | Both shadows |

► **Note:** If both the background and foreground colors of a character are set to be the same, the character's background is displayed as transparent.

The attributes register must be loaded 8 cycles after the current line register R0(3) is loaded. Loading the attributes register enables the OSD logic during the next 16 cycles. If the attributes register is not loaded, there is no active OSD, even if the current line register R0(3) is loaded. See Table 41.

Table 41 Register3, Bank 3, Read Operation, Attributes Register

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|-------|----------|-----|-----|-----|-----|-----|-----|-----|
| R/W | reserved | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | x | x | x | x | x | x | x | x |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | x | x | x | x | x | x | x | x |

Note: R = Read W = Write X = Indeterminate

| Reg Field | Bit Position | Data | Description |
|---------------|----------------|------|-------------|
| Same as R2(3) | Read operation | | |

The data read from the attribute register is a combination of attribute fields from the most recently displayed character and control character codes loaded into the attribute_data register. Character codes are fetched from Video RAM and must be loaded into the attribute_data register R3(3). Bit <f> of the attribute_data register (during a read) indicates whether the most recent character was a control or displayed character. The data read from the attribute_data register must be directly loaded into attribute register R2(3). Refer to Table 42.

Table 42 Register3, Bank 3, Write Operation, Attribute Data Register

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|-------|-----|-----|-----|-----|-----|-----|-----|-----|
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | x | x | x | x | x | x | x | x |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | x | x | x | x | x | x | x | x |

Note: R = Read W = Write X = Indeterminate

| Reg Field | Bit Position | Data | Description |
|-----------|------------------|------|----------------------------------|
| VRAM_data | fedcba9876543210 | xxxx | Character code fetched from VRAM |

Loading VRAM data into an attribute_data register initializes a CGROM access cycle. Four clock cycles after the LD instruction, the Z90T366 halts for three clock cycles to fetch the data from CGROM and latch it into a CGROM data capture register. After the CGROM data is latched, core operations are resumed. When a control character code is loaded into the attribute_data register, the CGROM data from address 0000 hex is fetched. Therefore, ZiLOG recommends placing a

space character at location 0000 hex in CGROM. Refer to Table 43 through Table 46 for the various VRAM data formats loaded in R3(3).

Table 43 Register 3, Bank 3, Display Character Format for Attribute Data Register, OSD Mode Write Operation

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|-------|-----|-----|-----|-----|-----|-----|-----|-----|
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | x | x | x | x | x | x | x | x |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | x | x | x | x | x | x | x | x |

Note: R = Read W = Write X = Indeterminate

| Reg Field | Bit Position | Data | Description |
|--|---------------|--|--------------------------------|
| Control bit | f----- | 0 | Display character |
| Background color* | -edc----- | 000 Black 001 Blue 010 Green 011 Cyan 100 Red 101 Magenta 110 Yellow 111 White | |
| Foreground color (Not Palette mode) | -----ba9----- | D | Same as Background_color |
| Foreground palette (Palette mode) | -----ba----- | 00 Palette 0 (defined in R6(3)<8-6> 01 Palette 1 (defined in R6(3)<b-9> 10 Palette 2 (defined in R6(3)<5-3> 11 Palette 3 (defined in R6(3)<2-0> | |
| Second underline (Palette mode) | -----9----- | 1 Second underline attribute is active 0 Second underline attribute is inactive | |
| Attribute8 | -----8----- | 1 Selected attribute (R7(3),<76>) is active 0 Selected attribute (R7(3),<76>) is inactive | |
| Character code | -----76543210 | D | Defines the character in CGROM |

Note: *If both the background and foreground colors of a character are set to be the same, the character's background is displayed as transparent.

Table 44 Register 3, Bank 3, Control Character Format for Attribute Data Register, OSD Mode Write Operation

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|-------|-----|----------|----------|----------|----------|----------|----------|----------|
| R/W | R/W | reserved | reserved | reserved | reserved | reserved | reserved | reserved |
| Reset | x | x | x | x | x | x | x | x |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | x | x | x | x | x | x | x | 1 |

Note: R = Read W = Write X = Indeterminate

| Reg Field | Bit Position | Data | Description |
|-------------|---------------|----------------------|---|
| Control bit | f----- | 1 | Control character |
| Reserved | -edcba98----- | | Reserved |
| Shift_video | -----7----- | 1 0 | Video signal is delayed by 8 pixels Standard character positioning |
| Transparent | -----6----- | 1 0 | Transparent background Background color defined by "background color" field |
| Blinking | -----5----- | 1 0 | Blinking character Not blinking character |
| Italic | -----4----- | 1 0 | Italic character Not italic character |
| Color_delay | -----32-- | 00 01 10 11 | Character color changes instantly Color changes with 4 pixels delay Color changes with 8 pixels delay Color changes with 12 pixels delay |
| Shadowing | -----10 | 00 01 10 11 | No shadowing Left shadow Right shadow Both shadows |

Smoothing is supported for double size (x2) and triple size (x3) characters only.

At reset, the background color in OSD mode is black. Foreground color, background color, blinking and italic attributes are delayed by 3/4 character. The smoothing attribute is enabled.

Table 45 Register 3, Bank 3, Display Character Format for Attribute Data Register, Write Operation CCD Mode

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|-------|-----|-----|-----|-----|-----|-----|-----|-----|
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | x | x | x | x | x | x | x | x |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | x | x | x | x | x | x | x | x |

Note: R = Read W = Write X = Indeterminate

| Reg Field | Bit Position | Data | Description |
|----------------|--------------|------|---------------------------------|
| Control bit | 7----- | 0 | Display character |
| Character code | -6543210 | | Defines the character in CGROM. |

Table 46 Register 3, Bank 3, Control Character Format for Attribute Data Register, CCD Mode, Write Operation

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|-------|-----|-----|-----|-----|-----|-----|-----|-----|
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | x | x | x | x | x | x | x | x |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | x | x | x | x | x | x | x | x |

Note: R = Read W = Write X = Indeterminate

| Reg Field | Bit Position | Data | Description |
|-------------|--------------|--------|--|
| Control bit | 7----- | 1 | Control character |
| Transparent | -6----- | 1 0 | Transparent background Background color defined by "background color" field |
| Blinking | --5----- | 1 0 | Blinking character Not blinking character |
| Italic | ---4---- | 1 0 | Italic character Not italic character |

| Reg Field | Bit Position | Data | Description |
|------------------|--------------|------|---------------------------------|
| Foreground color | ----321- | 000 | Black |
| | | 001 | Blue |
| | | 010 | Green |
| | | 011 | Cyan |
| | | 100 | Red |
| | | 101 | Magenta |
| | | 110 | Yellow |
| | | 111 | White |
| First underline | -----0 | 1 | Underline attribute is active |
| | | 0 | Underline attribute is inactive |

In CCD mode, each character occupies 8 bits (one byte) in VRAM. The CCD characters must be mapped into a 16-bit VRAM data field. The hardware supports compressed character placement in VRAM. Each word in VRAM is represented by HIGH byte and LOW byte. A currently active byte is selected by R4(3)<c>. The format and data representation for both bytes is the same.

There are two possible character formats defined: a “display” character and a “control” character. The code stored in “display” character format defines a character code. The “control” character defines up to seven attributes of the next character and is presented on screen as a space character.

Combining display and control characters generates of a CCD OSD according to FCC specification. Refer to Table 47.

Table 47 Register4, Bank 3, OSD Control Register

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|-------|-----|-----|-----|-----|-----|-----|-----|-----|
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | x | x | x | x | x | x | x | x |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | x | x | x | x | x | x | x | x |

Note: R = Read W = Write X = Indeterminate

| Reg Field | Bit Position | R | W | Data | Description |
|-----------|--------------|---|---|------|------------------------------|
| Underline | fe----- | R | W | 1x | Second underline is active |
| | | | | 0x | Second underline is inactive |
| | | | | x1 | First underline is active |
| | | | | x0 | First underline is inactive |

| Reg Field | Bit Position | R | W | Data | Description |
|--------------------|---------------|---|---|----------------------|---|
| OSD/CCD | ---d----- | R | W | 1 0 | OSD mode CCD mode |
| CCD_top/btm | ---c----- | R | W | 1 0 | The upper byte in VRAM is used The lower byte in VRAM is used |
| Italic_shift | ----ba98----- | R | W | x | Defines delay of the character |
| Blink_off/on | -----7----- | R | W | 0 1 | Blinking character is displayed Blinking character is NOT displayed (hidden) |
| MPX_bus | -----65---- | R | W | 00 01 10 11 | x1 character size x2 character size x3 character size Reserved |
| CGROM scan_line | -----43210 | R | W | | Defines CGROM addressing |

The Underline field must be set by firmware when scan lines that contain underline information are displayed. The underline bits are ANDed with the second and first underline active fields of data loaded into attribute register R2(3), causing the screen character to be underlined.

The Italic shift field defines a delay of video data. It is used to generate italic characters. The firmware decrements by 1 (the value of the Italic_shift field) for each consecutive line. The video signal is delayed only for characters that have the R2(3)<4> ("italic") bit set to 1.

Table 48 lists the bits for the capture register.

Table 48 Register5, Bank 3, Capture Register, Read Operation

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|-------|-----|-----|-----|-----|-----|-----|-----|---|
| R/W | R | R | R | R | R | R | R | R |
| Reset | x | x | x | x | x | x | x | x |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R |
| Reset | x | x | x | x | x | x | x | x |

Note: R = Read W = Write X = Indeterminate

| Reg Field | Bit Position | R | W | Data | Description |
|-----------|------------------|---|---|------|--|
| Cap_data | fedcba9876543210 | R | | | 16-bit captured data |
| I2C_saddr | -----7654321- | | W | | I ² C Slave interface address |

In Read mode, R5(3) returns the 16-bit captured data from the IRIN pin.
In Write mode, the 7-bit I²C slave interface address must be put in bit 7-1.

Table 49 lists the bits for the palette control register.

Table 49 Register6, Bank 3, Palette Control Register

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|-------|-----|-----|-----|-----|-----|-----|-----|-----|
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Note: R = Read W = Write X = Indeterminate

| Reg Field | Bit Position | R | W | Data | Description |
|-----------------|---------------|---|---|--|---|
| Palette | f----- | R | W | 1 0 | Palette mode is active Palette mode is INACTIVE |
| Underline color | -edc----- | R | W | 000 001 010 011 100 101 110 111 | Black Blue Green Cyan Red Magenta Yellow White |
| Palette1 | ----ba9----- | R | W | | Same as Underline color |
| Palette0 | -----876----- | R | W | | Same as Underline color |
| Palette3 | -----543--- | R | W | | Same as Underline color |
| Palette2 | -----210 | R | W | | Same as Underline color |

At POR the palette control register is reset to 0.

Table 50 lists the bits for the output palette control register.

Table 50 Register7, Bank 3, Output Palette Control Register

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|-------|-----|-----|-----|-----|-----|-----|-----|-----|
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Note: R = Read W = Write X = Indeterminate

| Reg Field | Bit Position | R | W | Data | Description |
|---------------------|---------------|---|---|----------------------|---|
| BLANK_delay | f e d c----- | R | W | D | VBlank and SVBLANK delay value—%00—POR condition |
| Background_on/off | ----b----- | R | W | 1 0 | Master background is on Master background is off—POR condition |
| Background_color | -----a98----- | R | W | D | Defines the color of the Master background (same as the palette) |
| AttrSelect | -----76----- | R | W | 00 01 10 11 | 1st underline—POR Semi-transparency Blinking CGROM bank select |
| Smoothing | -----5----- | R | W | 0 1 | Smoothing logic disabled—POR Smoothing logic enabled |
| Cursor Write Enable | -----4----- | R | W | 0 1 | Return “0” Cursor parameters write disabled—POR Cursor parameters write enabled |
| Palette # | -----3210 | R | W | | Palette number |

At POR the Output palette register is set to 0 for digital output.

Table 15 is the look-up table for the color palettes.

4 Instruction Set

The processor instruction set consists of 30 basic instructions. It has been optimized for high code density and reduced execution time. Single-cycle instruction execution is possible on most instructions.

The format for Op Codes and addressing modes is provided in the following tables but is normally not required. The assembler removes the burden of hand constructing the instruction format, by translating the mnemonics. System designers can access the instruction format when debugging.

4.1 Instruction Summary

The DSP instruction set can be broken down into the following types of instructions:

- Accumulator Modification
- Arithmetic
- Bit Manipulation
- Load
- Logical
- Program Control
- Rotate and Shift

Instruction format mnemonics are in Table 51. Table 52 through Table 58 list other instructions.

Table 51 Instruction Format Mnemonics

| Mnemonic | Description |
|-----------|--------------------------|
| A | Address |
| am | Accumulator Modification |
| b | RAM Bank |
| cc | Condition Code |
| const exp | Constant Expression |
| d | Destination Address |
| dest | Destination Value |
| fm | Flag Modification |

Table 51 Instruction Format Mnemonics (Continued)

| Mnemonic | Description |
|----------|------------------|
| op | Op Code |
| rp | Register Pointer |
| s | Source Address |
| src | Source Value |

Table 52 Accumulator Modification Instructions

| Mnemonic | Operands | Instruction |
|----------|----------|----------------|
| ABS | <cc>, A | Absolute Value |
| CP | A, <src> | Comparison |
| DEC | <cc>, A | Decrement |
| INC | <cc>, A | Increment |
| NEG | <cc>, A | Negate |

Table 53 Arithmetic Instructions

| Mnemonic | Operands | Instruction |
|----------|----------|-------------|
| ADD | A, <src> | Add |
| CP | A, <src> | Compare |
| SUB | A, <src> | Subtract |

Table 54 Bit Manipulation Instructions

| Mnemonic | Operands | Instruction |
|----------|----------|--------------------------------|
| CCF | None | Clear Carry Flag |
| CIEF | None | Clear Interrupt Enable Flag |
| COPF | None | Clear Overflow Protection Flag |
| SCF | None | Set Carry Flag |

Table 54 Bit Manipulation Instructions

| Mnemonic | Operands | Instruction |
|----------|----------|------------------------------|
| SIEF | None | Set Interrupt Enable Flag |
| SOPF | None | Set Overflow Protection Flag |

Table 55 Load Instructions

| Mnemonic | Operands | Instruction |
|----------|---------------|-------------|
| LD | <dest>, <src> | Load |
| POP | <dest> | Pop |
| PUSH | <src> | Push |

Table 56 Logical Instructions

| Mnemonic | Operands | Instruction |
|----------|----------|----------------------|
| AND | A, <src> | Logical AND |
| OR | A, <src> | Logical OR |
| XOR | A, <src> | Logical Exclusive OR |

Table 57 Program Control Instructions

| Mnemonic | Operands | Instruction |
|----------|----------|----------------|
| CALL | A | Call Procedure |
| JP | A | Jump |
| RET | None | Return |

Table 58 Rotate and Shift Instructions

| Mnemonic | Operands | Instruction |
|----------|----------|-------------|
| RL | <cc>, A | Rotate Left |

Table 58 Rotate and Shift Instructions

| Mnemonic | Operands | Instruction |
|----------|----------|------------------------|
| RR | <cc>, A | Rotate Right |
| SLL | <cc>, A | Shift Left Logical |
| SRA | <cc>, A | Shift Right Arithmetic |

4.2 Instruction Operands

To access the operands for the DSP, use the Register Pointers, Data Pointers, Hardware Registers, Direct Addressing, Immediate Data and Memory. There are nine distinct types of instruction operands. Table 59 and Table 60 describe these instructions.

Table 59 Instruction Operand Summary

| Symbolic Name | Syntax | Description |
|-------------------|--|-------------------------------|
| <pregs> | Pn:b | Register Pointer |
| <dregs> | Dn:b | Data Pointer |
| <hwregs> | X, Y, PC, SR, EXTn, A, BUS | Hardware Registers |
| <accind> | @A | Accumulator Indirect |
| <direct> | <const exp> | Direct Address Expression |
| <limm> | #<const exp> | Long (16-Bit) Immediate |
| <simm> | #<const exp> | Short (8-Bit) Immediate Value |
| <regind> | @Pn:b @Pn:b+ @Pn:b+Loop @Pn:b-Loop | Indirect Addressing of RAM |
| <memind> @Dn:b | @Dn:b @@Pn:b @@Pn:b+ @@Pn:b+Loop @@Pn:b-Loop | Indirect Addressing of ROM |

Table 60 Instruction Mnemonics/Operands

| # | Instruction | | Mnemonic/Operand Representation | |
|---|-------------|-----------|---------------------------------|-------------------|
| 1 | LD | P2:0, #F2 | LD | <pregs>, <simm> |
| 2 | LD | A, @P2:0 | LD | <hwreg>, <regind> |
| 3 | LD | X, @A | LD | <hwreg>, <accind> |
| 4 | LD | Y, #3CF5 | LD | <hwreg>, <limm> |
| 5 | SUB | A, @@P2:0 | SUB | A, <memind> |
| 6 | OR | A, @D2:0 | OR | A, <memind> |
| 7 | ADD | A, %F2 | ADD | A, <direct> |
| 8 | PUSH | D1:1 | PUSH | <dregs> |

<pregs> The register pointer mode is used for loading the pointer with the appropriate RAM address. This address references the RAM location that stores the requested data. The pointer can also be used to store 8-bit data when used as a temporary register. The pointers are connected to the lower 8 bits of the D-bus. Instruction 1 loads *Pointer 2*, RAM Bank0 with the value F2h.

<regind> The register indirect mode is used for indirect access to RAM. As noted in Instruction 2, the register indirect address method is used to get the operand to multiply it with the accumulator.

<dregs> The data-pointer mode is used as an indirect addressing method similar to @P2:0. The data pointers access the lower 16 bits of each RAM bank. Instruction 7 uses indirect addressing to *PUSH* information onto the stack.

<memind> Pointer or data registers can be used to access program memory. Both are commonly used to reference program memory. Instructions 5 and 6 display this addressing method. Either pointer is automatically incremented to assist in transferring sequential data.

<accind> Another method of indirect addressing is using the accumulator to store the address. Instruction 2 describes how to use this method.

<direct> The absolute RAM address is used in the direct mode. A range between 0 and 511 (000h to 1FFh) is allowed. The accumulator is used in conjunction with this method as a source or destination operand. Instruction 7 displays the accumulator as the destination.

<limm> This instruction indicates a long immediate load. A 16-bit word can be copied directly from the operand into the specified register or memory. Instruction 3 uses this method.

<simm> This instruction can only be used for immediate transfer of 8-bit data in the operand to the specified RAM pointer.

4.3 Instruction Format

The instruction format that specifies to the processor the action to be taken consists of the Op Code, destination, source, and other special bits. The assembler makes this operation transparent by providing mnemonics. Occasionally, the instruction format and development code can assist in debugging. Examples to clarify the various instruction formats and explain how specific bit patterns are developed and evaluated are provided below.

Most instructions require one 16-bit word containing the information necessary for the processor to execute the instruction correctly. This process requires one clock cycle for execution. Immediate addressing, immediate operands, **JUMP** and **CALL** instructions require two 16-bit words (two clock cycles). Each instruction type has a unique Op Code and format to differentiate various instructions. Different operations also have unique formats.

The variables *a*, *op*, *b*, *d*, *s*, *cc*, *am*, *fm*, *rp* are used in the instruction format to depict bits determined by the active instruction.

ADDITION and INC Formats

The Op Code and format for an instruction differ to allow the processor to differentiate between the instructions. For example, the **ADDITION** instruction requires that two operands be defined in the instruction.

ADD A, <regind>

| | | | | | | | | | | | | | | | |
|---------|----|----|----|----|----|---|-------------|----------------|---|---|---|-------------------|---|---|---|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | S | S | S | S |
| Op Code | | | | | | | RAM Bank | Condition Code | | | | Modification Code | | | |

The **INC** (increment) instruction requires that a condition and modification code be specified.

INC A

| | | | | | | | | | | | | | | | |
|---------|----|----|----|----|----|---|----------------|---|---|---|---|-------------------|---|---|---|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 |
| Op Code | | | | | | | Condition Code | | | | | Modification Code | | | |

INC and SLL Formats

The `INC` and `SLL` instructions have the same Op Code with an accumulator modification format. The least significant four bits, the modification code, determine the type of operation the accumulator performs.

`INC A`

| | | | | | | | | | | | | | | | |
|---------|----|----|----|----|----|---|----------------|---|---|---|---|-------------------|---|---|---|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| Op Code | | | | | | | Condition Code | | | | | Modification Code | | | |

`SLL A`

| | | | | | | | | | | | | | | | |
|---------|----|----|----|----|----|---|----------------|---|---|---|---|-------------------|---|---|---|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| Op Code | | | | | | | Condition Code | | | | | Modification Code | | | |

4.4 Instruction Bit Codes

The values in a series of bits in a register form patterns called `bit codes`. Types of bit codes include the following:

- Condition Codes
- Accumulator Modification Code
- Flag Modification Codes
- Source/Destination Field Designators
- Register Pointer/Data Pointer

The following tables list the options available and their corresponding instructions.

Condition Codes

Table 61 lists the condition codes that are used in accumulator modification, `CALL`, and `JUMP` instructions.

Table 61 Condition Code Bits

| Bit Code | Mnemonic Code Value | Condition Code Value | Condition Code |
|----------|---------------------|----------------------|----------------|
| 00000 | F | | False |

Table 61 Condition Code Bits (Continued)

| Bit Code | Mnemonic Code Value | Condition Code Value | Condition Code |
|----------|---------------------|----------------------|----------------------|
| 00001 | | | Unused |
| 00010 | NU0 | UI0 is set to 0 | Not User Zero |
| 00011 | NU1 | UI1 is set to 0 | Not User One |
| 00100 | NC | C is set to 0 | No Carry |
| 00101 | NZ | Z is set to 0 | Not Zero (Not Equal) |
| 00110 | NOV | OV is set to 0 | No Overflow |
| 00111 | PL | N is set to 0 | Plus (Not Negative) |
| 01xxx | | | Unused |
| 10000 | T | | True |
| 10001 | | | Unused |
| 10010 | U0 | UI0 is set to 1 | User Zero |
| 10011 | U1 | UI1 is set to 1 | User One |
| 10100 | C | C is set to 1 | Carry |
| 10101 | Z | Z is set to 1 | Zero (Equal) |
| 10110 | OV | OV is set to 1 | Overflow |
| 10111 | MI | N is set to 1 | Minus (Negative) |
| 11xxx | | | Unused |

Accumulator Modification Codes

Accumulator modification codes determine the type of modification made to the value in the accumulator. See Table 62. Condition codes are also used with CALL, and JUMP instructions.

Table 62 Accumulator Modification Bits

| Bit Code | Mnemonic | Operation |
|----------|----------|--------------|
| 0000 | RR | Rotate Right |
| 0001 | RL | Rotate Left |

Table 62 Accumulator Modification Bits

| Bit Code | Mnemonic | Operation |
|----------|----------|-------------|
| 0010 | SR | Shift Right |
| 0011 | SL | Shift Left |
| 0100 | INC | Increment |
| 0101 | DEC | Decrement |
| 0110 | NEG | Negate |
| 0111 | ABS | Absolute |

Flag Modification Codes

Flag modifications initialize or set/reset bits to accommodate interrupts, overflows, and carries. See Table 63.

Table 63 Flag Modification Bits

| Bit Code | Mnemonic | Operation | Flag | Value |
|----------|----------|---------------------------|------|-------|
| xx10 | CCF | Clear Carry | C | 0 |
| xx11 | SCF | Set Carry | C | 1 |
| x1x0 | CIEF | Clear Interrupt Enable | IE | 0 |
| x1x1 | SIEF | Set Interrupt Enable | IE | 1 |
| 1xx0 | COPF | Clear Overflow Protection | OP | 0 |
| 1xx1 | SOPF | Set Overflow Protection | OP | 1 |

Source/Destination Field Designators

Register pointers and data pointers provide convenient access to data. The pointers are a source or destination field in instructions. Specific bit codes are listed in Table 64. The register pointer offers optional incrementing or decrementing. This option is specified by the following instruction:

```
LD A, @P2:1+
```

Table 64 Register Pointer/ Data Pointer Bits

| Bit Code | Mnemonic |
|----------|--------------|
| 00xx | NOP |
| 01xx | +1 |
| 10xx | -1/loop |
| 11xx | +1/loop |
| xx00 | P0:0 or P0:1 |
| xx01 | P1:0 or P1:1 |
| xx10 | P2:0 or P2:1 |
| 0011 | D0:0 or D0:1 |
| 0111 | D1:0 or D1:1 |
| 1011 | D2:0 or D2:1 |
| 1111 | D3:0 or D3:1 |

Data pointers are automatically incremented when accessing program memory (for example, LD A, @D0:0) and do not require an incrementing option. Code in xx11 format is designated for a data pointer when source or destination format is used.

Additional source or destination designators include the other hardware registers provided by the processor. To determine if a data pointer, register pointer or a register is used as a source or destination is discussed in the next section.

Table 65 lists the bit code for mnemonic register names.

Table 65 Register Bits

| Bit Code | Mnemonic |
|----------|----------|
| 0000 | Bus |
| 0001 | X |
| 0010 | Y |
| 0011 | A |
| 0100 | SR |

Table 65 Register Bits (Continued)

| Bit Code | Mnemonic |
|----------|----------|
| 0101 | STACK |
| 0110 | PC |
| 0111 | Reserved |
| 1000 | EXT0 |
| 1001 | EXT1 |
| 1010 | EXT2 |
| 1011 | EXT3 |
| 1100 | EXT4 |
| 1101 | EXT5 |
| 1110 | EXT6 |
| 1111 | EXT7 |

4.5 Instruction Format Examples

Refer to the following examples indicating how bit codes are used in an instruction format.

Instruction Format

| | | | | | | | | | | | | | | | |
|---------|----|----|----|----|----|----|-------------|---|-------------|---|---|---|--------|---|---|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| op | op | op | op | op | op | op | b | d | d | d | d | s | s | s | s |
| Op Code | | | | | | | RAM bank | | Destination | | | | source | | |

Accumulator Modification Format

| | | | | | | | | | | | | | | | |
|---------|----|----|----|----|----|----|----------------|----|----|----|----|------------------|----|----|----|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| op | op | op | op | op | op | op | cc | cc | cc | cc | cc | am | am | am | am |
| Op Code | | | | | | | Condition Code | | | | | ACC Modification | | | |

Notes:

1. The Variables *a*, *op*, *b*, *d*, *s*, *cc*, *am*, *fm*, *rp* are used in the instruction format to depict bits determined by the instruction.
2. The General Instruction Format requires an Op Code, RAM bank bit, destination and source addresses. For example, LD A, @P2:1+

Load Instruction Format

| | | | | | | | | | | | | | | | |
|---------|----|----|----|----|----|---|-------------|-------------|---|---|---|--------|---|---|---|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 |
| Op Code | | | | | | | RAM Bank | Destination | | | | Source | | | |

The Op Code (0000001) provides a unique signature for the LD command. The processor uses this signature to determine the instruction format. The RAM bank bit is high (equal to 1) because of the instruction definition $b=1$ ($P_n:b$). The destination bit code is 0011 which corresponds to the accumulator. The source 0110 corresponds to the +1 option and P2:0 or P2:1. The RAM bank bit indicates that the processor loaded the accumulator with the operand designated by Pointer 2 Bank1 (P2:1).

Source and destination fields can be accessed from the register pointers, data pointers, or registers. The Op Code specifies the type of source and destination. An Op Code of 0000101 specifies that the source is an indirect address to program memory (@@P0:0 or @D0:0) and the destination is a register.

Instruction Format Listing

Instruction formats and applicable instructions are listed in Table 66 through Table 72.

Notes:

1. Several instructions provide various addressing modes to obtain operands; therefore, the same instruction can have several different formats depending on the addressing mode.
2. The variables *a*, *op*, *b*, *d*, *s*, *ce*, *am*, *fm*, *rp* are used in the instruction format to depict bits determined by the specific instruction used.

General Instruction Format

| | | | | | | | | | | | | | | | |
|---------|----|----|----|----|----|----|-------------|-------------|---|---|---|--------|---|---|---|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| op | op | op | op | op | op | op | b | d | d | d | d | s | s | s | s |
| Op Code | | | | | | | RAM Bank | Destination | | | | Source | | | |

Table 66 General Instruction Format

| Mnemonic | Operands | Bit Code | Hex |
|----------|----------|---------------------|------|
| ADD | A, @P0:0 | 1000001 0 0000 0000 | 8200 |
| AND | A, D1:1 | 1100110 1 0000 0111 | AB07 |
| CP | A,X | 0110000 0 0000 0001 | 6001 |
| LD | A,P | 0000000 0 0011 0111 | 0037 |
| POP | X | 0000000 0 0001 0101 | 0015 |
| PUSH | D0:0 | 0000001 0 0101 0011 | 0253 |
| RET | | 0000000 0 0110 0101 | 0065 |
| SUB | A,@D1:1 | 0010101 1 0000 0111 | 2B07 |
| XOR | A,P2:0 | 1111001 0 0000 0010 | F202 |

Accumulator Modification Format

| | | | | | | | | | | | | | | | |
|---------|----|----|----|----|----|----|----------------|----|----|----|----|------------------|----|----|----|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| op | op | op | op | op | op | op | cc | cc | cc | cc | cc | am | am | am | am |
| Op Code | | | | | | | Condition Code | | | | | ACC Modification | | | |

Table 67 Accumulator Modification Format

| Mnemonic | Operands | Bit Code | Hex Representation |
|----------|----------|--------------------|--------------------|
| ABS | Z, A | 1001000 10101 0111 | 9157 |
| DEC | A | 1001000 00000 0101 | 9005 |

Table 67 Accumulator Modification Format

| Mnemonic | Operands | Bit Code | Hex Representation |
|----------|----------|--------------------|--------------------|
| INC | NZ, A | 1001000 00101 0100 | 9054 |
| NEG | A | 1001000 00000 0110 | 9006 |
| RR | NU0, A | 1001000 00010 0000 | 9020 |
| RL | PL, A | 1001000 00111 0001 | 9071 |
| SLL | C, A | 1001000 10100 0011 | 9143 |
| SRA | U1, A | 1001000 10011 0010 | 9132 |

Flag Modification Format

| | | | | | | | | | | | | | | | |
|---------|----|----|----|----|----|----|----------------|----|----|----|----|-------------------|----|----|----|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| op | op | op | op | op | op | op | cc | cc | cc | cc | cc | fm | fm | fm | fm |
| Op Code | | | | | | | Condition Code | | | | | Flag Modification | | | |

Table 68 Flag Modification Format

| Mnemonic | Bit Code | Hex Representation |
|----------|--------------------|--------------------|
| CCF | 1001010 00000 0010 | 9402 |
| CIEF | 1001010 00000 0100 | 9404 |
| COPF | 1001010 00000 1000 | 9408 |
| SCF | 1001010 00000 0011 | 9403 |
| SIEF | 1001010 00000 0101 | 9405 |
| SOPF | 1001010 00000 1001 | 9409 |

Direct Internal Addressing Format

| | | | | | | | | | | | | | | | |
|---------|----|----|----|----|----|----|------------------------|---|---|---|---|---|---|---|---|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| op | op | op | op | op | op | op | a | a | a | a | a | a | a | a | a |
| Op Code | | | | | | | 9-Bit Internal Address | | | | | | | | |

Table 69 Direct Internal Addressing Format

| Mnemonic | Operands | Bit Code | Hex Representation |
|----------|----------|-------------------|--------------------|
| ADD | A, %FF | 1000011 011111111 | 86FF |
| AND | A, 255 | 1010011 011111111 | A6FF |
| CP | A, 255 | 0110011 011111111 | 66FF |
| LD | | 0000111 000010010 | 0E12 |

Short Immediate Addressing Format

| | | | | | | | | | | | | | | | |
|---------|----|----|----|----|---------------------|----|----|------------------------------|---|---|---|---|---|---|---|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| op | op | op | op | op | rp | rp | rp | a | a | a | a | a | a | a | a |
| Op Code | | | | | Register Pointer | | | 8-Bit Immediate Address/Data | | | | | | | |

Table 70 Short Immediate Addressing Format

| Mnemonic | Operands | Bit Code | Hex Representation |
|----------|------------|--------------------|--------------------|
| LD | P1:1, #%FA | 00011 101 11111010 | 1DFA |

Long Immediate Addressing Format

| | | | | | | | | | | | | | | | |
|---------------------|----|----|----|----|----|----|-------------|-------------|---|---|---|--------|---|---|---|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| op | op | op | op | op | op | op | b | d | d | d | d | s | s | s | s |
| Op Code | | | | | | | RAM Bank | Destination | | | | Source | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| a | a | a | a | a | a | a | a | a | a | a | a | a | a | a | a |
| 16-Bit Address/Data | | | | | | | | | | | | | | | |

Table 71 Long Immediate Addressing Format

| Mnemonic | Operands | Hex Representation |
|----------|------------|--------------------|
| ADD | A, #%1234 | 8800 1234 |
| AND | A, #% 26A4 | A800 26A4 |
| LD | X, #%6FFC | 0810 6FFC |
| PUSH | #%C32C | 0850 C32C |
| SUB | A, #%2444 | 2800 2444 |
| XOR | A, #%AFC2 | E800 AFC2 |

JUMP and CALL Instruction Formats

| | | | | | | | | | | | | | | | |
|---------|----|----|----|----|----|---|----------------|----|----|----|----|----------|---|---|---|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 | 1 | 0 | cc | cc | cc | cc | cc | s | s | s | s |
| Op Code | | | | | | | Condition Code | | | | | Not Used | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| a | a | a | a | a | a | a | a | a | a | a | a | a | a | a | a |

16-Bit Address

Table 72 Jump and Call Instruction Formats

| Mnemonic | Operands | Hex Representation |
|----------|----------|--------------------|
| CALL | END | 4800 0004 |
| JP | U1, END | 4D30 0004 |

4.6 Instruction Timing

The DSP can be executed with single cycle instructions using independent data memory and program memory buses in the system architecture and pipeline instructions. This method allows the instruction fetch and execution cycles to overlap. Figure 27 illustrates the execution sequence. The first instruction takes two clock cycles to execute; subsequent executions occur in a single cycle. All instruction fetch cycles have the same machine timing regardless of whether external or internal memory is used. Because the DSP contains a two-level pipeline, the `JUMP` and `CALL` instructions do not disrupt the execution process.

In two-word instructions, the second word is fetched while the first word is executing. Because the processor knows that the instruction is a `JUMP` or `CALL`, the second word is transferred to the program counter and the correct address is fetched into the pipeline. There is no disruption or pipeline flushing. The pipeline flow is affected when the program counter is the destination for a load. Because the load (`LD`) instruction is a single word instruction, the next instruction is fetched during load execution. To compensate for the instruction in the pipeline, that instruction is executed as a `NOP`.

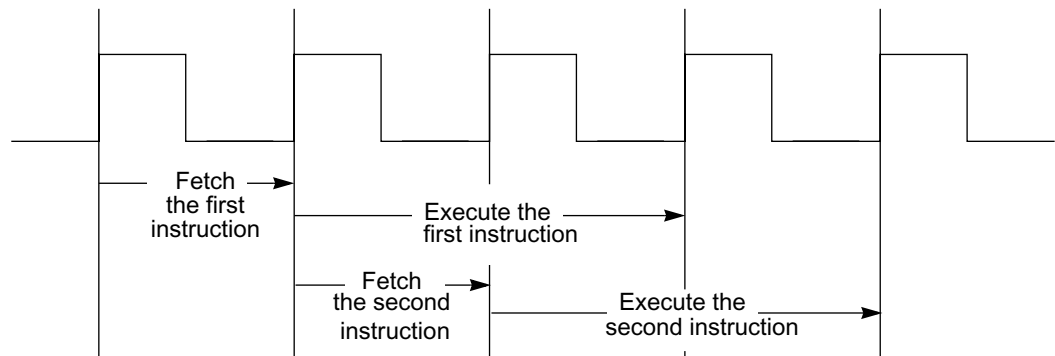


Figure 27 Pipeline Execution

4.7 Instruction Op Codes

Table 73 summarizes essential information about the instruction set.

Table 73 Instruction Op Codes

| Inst | Description | Op Code | Synopsis | Operands | Words | Cycles | Examples |
|------|------------------|---------|----------------------|---------------|-------|--------|-------------------|
| ABS | Absolute Value | 1001000 | ABS[<cc>,<src>] | <cc>,A | 1 | 1 | ABS NC, A |
| | | 1001000 | | A | 1 | 1 | ABS A |
| ADD | Addition | 1001001 | ADD<dest>, <src> | A,<pregs> | 1 | 1 | ADD A, P0:0 |
| | | 1000001 | | A,<dregs> | 1 | 1 | ADD A, D0:0 |
| | | 1000100 | | A,<limm> | 2 | 2 | ADD A, #%1234 |
| | | 1000101 | | A,<memind> | 1 | 3 | ADD A, @@P0:0 |
| | | 1000011 | | A,<direct> | 1 | 1 | ADD A, %F2 |
| | | 1000001 | | A,<regind> | 1 | 1 | ADD A, @P1:1 |
| | | 1000000 | | A,<hwregs> | 1 | 1 | ADD A, X |
| | | | | | | | |
| AND | Bitwise AND | 1011001 | AND <dest>, <src> | A,<pregs> | 1 | 1 | AND A, P2:0 |
| | | 1010001 | | A,<dregs> | 1 | 1 | AND A, D0:1 |
| | | 1010100 | | A,<limm> | 2 | 2 | AND A, #%1234 |
| | | 1010101 | | A,<memind> | 1 | 3 | AND A, @@P1:0 |
| | | 1010001 | | A,<direct> | 1 | 1 | AND A, %2C |
| | | 1010001 | | A,<regind> | 1 | 1 | AND A, @P1:2+LOOP |
| | | 1010000 | | A,<hwregs> | 1 | 1 | AND A, EXT3 |
| | | | | | | | |
| CALL | Subroutine Call | 0010100 | CALL <cc>, <address> | <cc>,<direct> | 2 | 2 | CALL sub1 |
| | | 0010100 | | <direct> | 2 | 2 | CALL Z, sub2 |
| CCF | Clear Carry Flag | 1001010 | CCF | None | 1 | 1 | CCF |
| CIEF | Clear Carry Flag | 1001010 | CIEF | None | 1 | 1 | CIEF |
| COPF | Clear OP Flag | 1001010 | COPF | None | 1 | 1 | COPF |

Table 73 Instruction Op Codes (Continued)

| Inst | Description | Op Code | Synopsis | Operands | Words | Cycles | Examples |
|------|-------------|---------|--------------------|----------------|-------|--------|---------------|
| CP | Comparison | | CP<src1>,<src2> | | | | |
| | | 0111001 | | A, <pregs> | 1 | 1 | CP A, P0:0 |
| | | 0110001 | | A, <dregs> | 1 | 1 | CP A, D3:1 |
| | | 0110101 | | A, <memind> | 1 | 3 | CP A, @@P0:0 |
| | | 0110011 | | A, <direct> | 1 | 1 | CP A, %FF |
| | | 0110001 | | A, <regind> | 1 | 1 | CP A, @P2:1+ |
| | | 0110000 | | A, <hwregs> | 1 | 1 | CP A, STACK |
| | | 0110100 | | A, <limm> | 2 | 2 | CP A, #%FFCF |
| DEC | Decrement | 1001000 | DEC [<cc>,<dest> | <cc>, A | 1 | 1 | DEC NZ, A |
| | | 1001000 | | A | 1 | 1 | DEC A |
| INC | Increment | 1001000 | INC [<cc>,<dest> | <cc>, A | 1 | 1 | INC PL, A |
| | | 1001000 | | A | 1 | 1 | INC A |
| JP | Jump | 0100110 | JP [<cc>,<address> | <cc>, <direct> | 2 | 2 | JP NIE, Label |
| | | 0100110 | | <direct> | 2 | 2 | JP Label |

Table 73 Instruction Op Codes (Continued)

| Inst | Description | Op Code | Synopsis | Operands | Words | Cycles | Examples |
|------|------------------------------|----------|--|--------------------|-------|--------|------------------|
| LD | Load Destination with Source | 00000000 | LD <dest>, <src> | A, <hwregs> | 1 | 1 | LD A, X |
| | | 00000001 | | A, <dregs> | 1 | 1 | LD A, D0:0 |
| | | 00010001 | | A, <pregs> | 1 | 1 | LD A, P0:1 |
| | | 00000001 | | A, <regind> | 1 | 1 | LD A, @P1:1 |
| | | 00001001 | | A, <memind> | 1 | 3 | LD A, @D0:0 |
| | | 00000011 | | A, <direct> | 1 | 1 | LD A, 124 |
| | | 00001111 | | <direct>, A | 1 | 1 | LD 124, A |
| | | 00001000 | | <dregs>, <hwregs> | 1 | 1 | LD D0:0, EXT7 |
| | | 00011000 | | <pregs>, <simm> | 1 | 1 | LD P1:1, #%FA |
| | | 00010100 | | <pregs>, <hwregs> | 1 | 1 | LD P1:1, EXT1 |
| | | 00001110 | | <regind>, <limm> | 1 | 1 | LD @P1:1, #%1234 |
| | | 00000010 | | <regind>, <hwregs> | 1 | 1 | LD @PM+, X |
| | | 00010001 | | <hwregs>, <pregs> | 1 | 1 | LD Y, P0:0 |
| | | 00000001 | | <hwregs>, <dregs> | 1 | 1 | LD SR, D0:0 |
| | | 00001000 | | <hwregs>, <limm> | 2 | 2 | LD PC, #%1234 |
| | | 01001001 | | <hwregs>, <accind> | 1 | 3 | LD X, @A |
| | | 00001001 | | <hwregs>, <dregs> | 1 | 3 | LD Y, D0:0 |
| | | 00000001 | | <hwregs>, <regind> | 1 | 1 | LD A, @P0:0-LOOP |
| | | 00000000 | | <hwregs>, <hwregs> | 1 | 1 | LD X, EXT6 |
| | | | When <dest> is <hwregs>, <dest> cannot be P. When <dest> is <hwregs> and <src> is <hwregs>, <dest> cannot be EXTn if <src> is EXTn, <dest> cannot be X if <src> is X, <dest> cannot be SR if <src> is SR. When <src> is <accind> <dest> cannot be A. | | | | |

Table 73 Instruction Op Codes (Continued)

| Inst | Description | Op Code | Synopsis | Operands | Words | Cycles | Examples |
|------|-----------------------------|---------|---------------------|-------------|-------|--------|------------------|
| NEG | Negate | 1001000 | NEG <cc>, A | <cc>, A | 1 | 1 | NEG NZ,A |
| | | 1001000 | | A | 1 | 1 | NEG A |
| NOP | No Operation | 0000000 | NOP | None | 1 | 1 | NOP |
| OR | Bitwise OR | 1101001 | OR <dest>, <src> | A, <pregs> | 1 | 1 | OR A, P0:1 |
| | | 1100001 | | A, <dregs> | 1 | 1 | OR A, D0:1 |
| | | 1100100 | | A, <limm> | 2 | 2 | OR A, #%202 |
| | | 1100101 | | A, <memind> | 1 | 3 | OR A, @@P2:1+ |
| | | 1100011 | | A, <direct> | 1 | 1 | OR A, %2C |
| | | 1100001 | | A, <regind> | 1 | 1 | OR A, @P1:0-LOOP |
| | | 1100000 | | A, <hwregs> | 1 | 1 | OR A, EXT6 |
| POP | Pop a Value from the Stack | 0001010 | POP <dest> | <pregs> | 1 | 1 | POP P0:0 |
| | | 0000100 | | <dregs> | 1 | 1 | POP D0:1 |
| | | 0000010 | | <regind> | 1 | 1 | POP @P0:0 |
| | | 0000000 | | A | 1 | 1 | POP A |
| PUSH | Push a Value onto the Stack | 0001001 | PUSH <src>, <pregs> | <pregs> | 1 | 1 | PUSH P0:0 |
| | | 0000001 | | <dregs> | 1 | 1 | PUSH D0:1 |
| | | 0000001 | | <regind> | 1 | 1 | PUSH @P0:0 |
| | | 0000000 | | <hwregs> | 1 | 1 | PUSH BU.S |
| | | 0000100 | | <limm> | 2 | 2 | PUSH #%2345 |
| | | 0100101 | | <accind> | 1 | 3 | PUSH @A |
| | | 0000101 | | <memind> | 1 | 3 | PUSH @@P0:0 |
| RET | Return from Subroutine | 0000000 | RET | None | 1 | 2 | RET |
| RL | Rotate Left | 1001000 | RL <cc>, A | <cc>, A | 1 | 1 | RL NZ, A |
| | | 1001000 | | A | 1 | 1 | RL A |
| RR | Rotate Right | 1001000 | RR <cc>, A | <cc>, A | 1 | 1 | RR C, A |
| | | 1001000 | | A | 1 | 1 | RR A |
| SCF | Set C Flag | 1001010 | SCF | None | 1 | 1 | SCF |
| SIEF | Set IE Flag | 1001010 | SIEF | None | 1 | 1 | SIEF |

Table 73 Instruction Op Codes (Continued)

| Inst | Description | Op Code | Synopsis | Operands | Words | Cycles | Examples |
|------|----------------------|---------|-------------------|-------------|-------|--------|-------------------|
| SLL | Shift Left Logical | 1001000 | SLL | [<cc>,) A | 1 | 1 | SLL NZ, A |
| | | 1001000 | | A | 1 | 1 | SLL A |
| SOPF | Set OP Flag | 1001010 | SOPF | None | 1 | 1 | SOPF |
| SRA | Shift Right | 1001000 | SRA <cc>, A | <cc>, A | 1 | 1 | SRA NZ, A |
| | Arithmetic | 1001000 | | A | 1 | 1 | SRA A |
| SUB | Subtract | 0011001 | SUB <dest>, <src> | A, <pregs> | 1 | 1 | SUB A, P1:1 |
| | | 0010011 | | A, <dregs> | 1 | 1 | SUB A, D0:1 |
| | | 0010100 | | A, <limm> | 2 | 2 | SUB A, #%2C2C |
| | | 0010101 | | A, <memind> | 1 | 3 | SUB A, @D0:1 |
| | | 0010011 | | A, <direct> | 1 | 1 | SUB A, %15 |
| | | 0010001 | | A, <regind> | 1 | 1 | SUB A, @P2:0-LOOP |
| | | 0010000 | | A, <hwregs> | 1 | 1 | SUB A, STACK |
| XOR | Bitwise Exclusive OR | | XOR <dest>, <src> | | | | |
| | | 1111001 | | A, <pregs> | 1 | 1 | XOR A, P2:0 |
| | | 1110001 | | A, <dregs> | 1 | 1 | XOR A, D0:1 |
| | | 1110100 | | A, <limm> | 2 | 2 | XOR A, #%3933 |
| | | 1110001 | | A, <memind> | 1 | 3 | XOR A, @P2:1+ |
| | | 1110011 | | A, <direct> | 1 | 1 | XOR A, %2F |
| | | 1110001 | | A, <regind> | 1 | 1 | XOR A, @P2:0 |
| | | 1110000 | | A, <hwregs> | 1 | 1 | XOR A, BUS |

Instruction Descriptions

The DSP instruction set consists of 30 basic instructions, optimized for high-code density and reduced execution time. Single-cycle instruction execution is possible because of the Z90T366 pipeline and system architecture. Table 74 contains a description for each instruction.

Table 74 Instruction Descriptions

| Mnemonic | Mnemonic Expansion |
|----------------------|--|
| Instruction Operands | Lists the types of addressing methods for a specific instruction (ABS A or ABS <cc>, A). |
| Instruction Format | Displays instruction format for register indirect addressing. |
| Operation | Displays operation sequence. |
| Affected Flags | Lists flags affected by operation. |
| Description | Describes the instruction operation. |
| Examples | A simple example displays the instruction operation and how registers are affected. The example includes initialization, instruction and result. It also includes the number of cycles and instruction length. |

Each Assembly Instruction includes an example for each addressing mode available for the specific instruction.

The mnemonics listed in Table 75 are used in the instruction format.

Table 75 Instruction Format Mnemonics

| Mnemonic | Description | Mnemonic | Description |
|-----------|--------------------------|----------|-------------------|
| A | Address | dest | Destination Value |
| am | Accumulator Modification | fm | Flag Modification |
| b | RAM Bank | op | Op Code |
| cc | Condition Code | rp | Register Pointer |
| const exp | Constant Expression | s | Source Address |
| d | Destination Address | src | Source Value |

ABS Absolute Value ABS

Instruction Word

| | | | | | | | | | | | | | | | |
|---------|----|----|----|----|----|---|----------------|----|----|----|----|------------------|---|---|---|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | cc | cc | cc | cc | cc | 0 | 1 | 1 | 1 |
| Op Code | | | | | | | Condition Code | | | | | ACC Modification | | | |

Syntax

ABS <cc>, A

ABS A

Operation

If ACC < 0 then -(ACC) -> ACC

Flags: N: Set if the accumulator has 800000h (see below).

Description

If the contents of the accumulator are determined to be less than 0 (a negative number), the absolute value of the accumulator is calculated (accumulator replaced by its two's complement value). Using the condition code provides an additional method to evaluate a status flag before the absolute value of the accumulator is calculated.

► **Note:** If the accumulator contains 800000h, the ABS A instruction stores the value of the two's complement at address 800000h and sets the Overflow and Negative status bits. There is no overflow protection.

Example 1

ABS A

Initialization: Accumulator contains FFEB00h
 SR contains 0000h

Instruction: ABS A

Result: Accumulator contains 001500h
 SR contains 1000h

This example uses one word of memory and executes in one machine cycle. Because the value in the accumulator is less than zero, the two's complement is performed and the result is placed in the accumulator $ABS(FFEBh) = 001500h$. The carry bit is set.

Example 2

`ABS <cc>, A`

Initialization: Accumulator contains 456400h

Instruction: `ABS MI, A`

Result: Accumulator contains 456400h

This example uses one word of memory and executes in one machine cycle. The condition code (negative bit) is not set because the accumulator value is positive; therefore, the instruction is not executed.

```
<A> = 456400h
      ABS MI, A
<A> = 456400h
```

Example:

```
<A> = C56400h
      ABS MI, A
<A> = 3AC900h
```

ADD Addition ADD

Instruction Word

| | | | | | | | | | | | | | | | |
|---------|----|----|----|----|----|---|-------------|-------------|---|---|---|--------|---|---|---|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 1 | b | 0 | 0 | 0 | 0 | s | s | s | s |
| Op Code | | | | | | | RAM Bank | Destination | | | | Source | | | |

Syntax

```

ADD      A,      <regind>
ADD      A,      <memind>
ADD      A,      <limm>
ADD      A,      <hwregs>
ADD      A,      <direct>
ADD      A,      <pregs>
ADD      A,      <dregs>

```

Operation

ACC + <source> -> ACC

Flags: C Set if carry from the most significant bit is found.
N: Set if result in the accumulator is negative.
Z: Set if result is 0.
OV: Set if addition exceeds upper (FFFFFFh)
 or lower (800000h) limit of the accumulator.

Description

The addressed data memory operand is added to the accumulator. The result is loaded into the accumulator.

► **Note:** The lower eight bits of the accumulator are unchanged while the add instruction is executed.

Example 1

ADD A, <regind>

Initialization: Accumulator contains 123456h
P0:0 contains 4Dh
RAM Bank1: 4Dh contains 8746h

Instruction: ADD A, @P0:0

Result: A contains 997A56h
@P0:0 contains 746h

This example uses one word of memory and executes in one machine cycle. The pointer P0:0 contains the RAM register location (4Dh). The contents of RAM register 4Dh are added to the accumulator to obtain the sum (874600h + 123456h = 997A56h). The sum is contained in the accumulator and the pointer is left unchanged. The direct addressing equivalent is ADD A, %4D or ADD A, 77 (4Dh = 77 decimal).

Example 2

ADD A, <memind>

Initialization: Accumulator contains 123400h
P0:0 contains 21h
RAM Bank0: 21h contains 247Ah
ROM Address: 247Ah contains 0C12h

Instruction: ADD A, @@P0:0

Result: A contains 1E4600h
P0:0 contains 21h
RAM Bank0: 21h contains 247Bh

This example uses one word of memory and executes in three machine cycles. The pointer P0:0 contains the RAM register location (21h). The contents of this register have a ROM address. This address refers to the ROM data placed in the specified accumulator by an AND instruction 123400h + 0C1200h = 1E4600h. When memory indirect addressing is used, the ROM address is automatically incremented. to provide a convenient method of accessing sequential data. Using ADD A, @@P0:0+ performs the same operation and also increments the P0:0 content to 22h.

Example 3

ADD A, <limm>

Initialization: Accumulator contains 123400h

Instruction: ADD A, #0C12

Result: A contains 1E4600h

This example uses two words of memory and executes in two machine cycles. The immediate operand 0C12h is added to the accumulator to obtain the sum $123400h + 0C1200h = 1E4600h$.

Example 4

ADD A, <hwregs>

Initialization: Accumulator contains 23400Hh
Register X contains 0C12h

Instruction: ADD A, X

Result: A contains 1E4600h

This example uses one word of memory and executes in one machine cycle. The contents of register X are added to the accumulator to obtain the sum. $123400h + 0C1200h = 1E4600h$. All hardware registers can transfer from <hwregs>.

Example 5

ADD A, <direct>

Initialization: Accumulator contains 123400h
RAM Bank0: F3h contains 0C12h

Instruction: ADD A, %F3

Result: A contains 1E4600h

This example uses one word of memory and executes in one machine cycle. Register F3h is added to the accumulator to obtain the sum. $123400h + 0C1200h = 1E4600h$. An equivalent instruction is ADD A, 243 (F3h = 243 decimal).

Example 6

ADD A, <pregs>

Initialization: Accumulator contains 123400h
P0:0 contains 56h

Instruction: ADD A, P0:0

Result: Accumulator contains 128A00h

This example uses one word of memory and executes in one machine cycle. The contents of the pointer register $P0:0$ is added to the accumulator. $123400h + 005600h = 128A00h$. The Pointer Register is connected to the lower 8 bits of the D-bus. The D-bus is connected to the upper 16-bits of the P-bus. This causes the pointer register operand to become $005600h$ before being added to the accumulator.

Example 7

ADD A,<dregs>

Initialization: Accumulator contains 123400h
 D0: 1 contains 8746h

Instruction: ADD A,D0:1

Result: A contains 997A00h
 D0: 1 contains 8746h

This example uses one word of memory and executes in one machine cycle. The contents of the data pointer $D0:1$ are added to the accumulator. The sum is contained in the accumulator and the pointer is left unchanged. The data pointer contains 8746h.

AND

BITWISE AND

AND

Instruction Word

| | | | | | | | | | | | | | | | |
|---------|----|----|----|----|----|---|-------------|-------------|---|---|---|--------|---|---|---|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 1 | 0 | 1 | 1 | 1 | 0 | 0 | b | 0 | 0 | 0 | 0 | s | s | s | s |
| Op Code | | | | | | | RAM Bank | Destination | | | | Source | | | |

Syntax

```

AND A, <regind>
AND A, <memind>
AND A, <limm>
AND A, <hwregs>
AND A, <direct>
AND A, <pregs>
AND A, <dregs>
AND A, <simm>

```

Operation

<accumulator>. AND.<source> -> <accumulator>

Flags: N: Set if accumulator result is less than 0.
Z: Set if accumulator result is 0.

Description

Data is stored in the specified accumulator by an AND instruction. The lower eight bits of the accumulator are cleared when this instruction is executed.

Example 1

```
AND A, <regind>
```

Initialization: Accumulator contains 123456h
P0A contains 45h
RAM Bank1: 45h contains 8746h

Instruction: AND A, @P0:1

Result: Accumulator contains 020400h

This example uses one word of memory and executes in one machine cycle. The data in RAM Bank1, referenced by RAM pointer 0, is stored in the specified accumulator using an AND instruction $123456h.AND.874600h = 020400h$.

Example 2

AND A, <memind>

Initialization: Accumulator contains 123400h
P0:0 contains 45h
RAM Bank0: 45h contains 047Ah
ROM Address: 047Ah contains 8746h

Instruction: AND A, @@P0:0

Result: A contains 020400h
P0:0 contains 45h
RAM Bank0: 45h contains 247Bh

This example uses one word of memory and executes in three machine cycles. The pointer P0:0 contains the RAM register location (45h). The contents of this register has a ROM address. This address refers to the ROM data that is placed in the specified accumulator by an AND instruction $123400h.AND.874600h = 020400h$. With memory-indirect addressing, the ROM address is automatically incremented. This provides a convenient method to access sequential data. Using AND A, @@P0:0+ performs the same operation and also increments the P0:0 content to 46h.

Example 3

AND A, <limm>

Initialization: Accumulator contains 362400h

Instruction: AND A, #%1234

Result: Accumulator contains 122400h

This example uses two words of memory and executes in two machine cycles. The immediate operand 1234h and an accumulator address are processed with an AND instruction to produce the result, $362400h.AND.123400h = 122400h$.

Example 4

AND A, <simm>

Initialization: Accumulator contains 123456h

Instruction: AND A, #%1F

Result: Accumulator contains 001400h

This example uses one word of memory and executes in one machine cycle. The data in the immediate field and the contents of the accumulator are processed with an AND instruction. $123456h \text{ AND } 001F00h = 001400h$.

Example 5

AND A, <hwregs>

Initialization: Accumulator contains 123400h
Register X contains 0C12h

Instruction: AND A, X

Result: A contains 001000h

This example uses one word of memory and executes in one machine cycle. Use an AND instruction to send the contents of Register X to the accumulator to obtain the result $123400h \text{ AND } 0C1200h = 001000h$. All hardware registers can transfer from <hwregs>.

Example 6

AND A, <direct>

Initialization: Accumulator contains 123400h
RAM Bank0: F3h contains 0C12h

Instruction: AND A, %F3

Result: A contains 001000h

This example uses one word of memory and executes in one machine cycle. Use an AND instruction to send Register F3h to the accumulator to obtain the result $123400h \text{ AND } 0C1200h = 001000h$. An equivalent instruction is AND A, 243 (F3h = 243 decimal).

Example 7

AND A, <pregs>

Initialization: Accumulator contains 123400h
P0:0 contains 56h

Instruction: AND A, P0:0

Result: Accumulator contains 001400h

This example uses one word of memory and executes in one machine cycle. Use an `AND` instruction to send the contents of the pointer register `P0:0` to the accumulator `123400h.AND.005600h = 001400h`. The Pointer Register is connected to the lower 8 bits of the D-bus. The D-bus is connected to the upper 16 bits of the P-bus. This action changes the pointer register operand to `005600h` before being added to the accumulator.

Example 8

`AND A, <dregs>`

Initialization: Accumulator contains 123400h
D0:1 contains 2645h

Instruction: `AND A, D0:1`

Result: Accumulator contains 020400h

This example uses one word of memory and executes in one machine cycle. The data register, `D0:1`, references the operand `2645h`. Use an `AND` instruction to send this data register to the accumulator to produce the result `123400h.AND.2645h = 020400h`.

CALL

SUBROUTINE CALL

CALL

Instruction Word

| | | | | | | | | | | | | | | | |
|---------|----|----|----|----|----|---|---|----------------|----|----|----|----------|---|---|---|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | b | cc | cc | cc | cc | s | s | s |
| Op Code | | | | | | | | Condition Code | | | | Not Used | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| a | a | a | a | a | a | a | a | a | a | a | a | a | a | a | a |

16-Bit Address

Syntax

CALL <cc>, <direct>

CALL <direct>

Operation

PC + 2 → STACK

16-Bit Address → PC

Flags: None

Description

The current Program Counter (PC) register content is incremented by two and placed on the stack. The address of the specified label in the CALL instruction is then placed in the PC register. The jump is made to the appropriate subroutine via the PC. The condition code option is used if CALL is executed.

Example 1

CALL <direct>

Initialization: PC contains 1FFBh
FFT2 subroutine address contains F234h
Stack Level 0 contains 0025h

Instruction: CALL FFT2

Result: PC contains F234h
Stack Level 0 contains 1FFDh
Stack Level 1 contains 0025h

This example uses two words of memory and executes in two machine cycles. The call to the subroutine FFT2 places PC+2 (1 FFDh) on the stack. All information currently on the stack is pushed up the stack. The subroutine address is then placed in the PC register. The processor executes the next instruction addressed by the PC, the FFT2 subroutine.

Example 2

CALL <cc>, <direct>

Initialization: PC contains 1FFBh
 FFT2 subroutine address contains F234h
 Stack Level 0 contains 0025h
 UO (User Zero Bit) contains 1

Instruction: CALL UO, FFT2

Result: PC= F234h
 Stack Level 0= 1FFDh
 Stack Level 1= 0025h

This example uses two words of memory and executes in two machine cycles. The condition code UO is tested by the processor before executing the CALL instruction. Because the UO bit is enabled, the CALL routine is executed exactly like the example above. The condition code UO is input to the processor that determines if subroutine FFT2 is used. Another CALL instruction can determine if another subroutine, FFT1, is used.

Example 3

CALL <direct>

Initialization: PC contains 1FFBh
 FFT2 subroutine address contains F234h
 Stack Level 0 contains 0025h

Instruction: CALL FFT2

Result: PC contains F234h
 Stack Level 0 contains 1FFDh
 Stack Level 1 contains 0025h

This example uses two words of memory and executes in two machine cycles. The call to the subroutine FFT2 places PC+2 (1 FFDh) on the stack. All information currently on the stack is pushed up the stack. The subroutine address is then placed in the PC register. The processor executes the next instruction addressed by the PC, the FFT2 subroutine.

CCF

CLEAR CARRY FLAG

CCF

Instruction Word

| | | | | | | | | | | | | | | | |
|---------|----|----|----|----|----|---|----------------|---|---|---|---|-------------------|---|---|---|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| Op Code | | | | | | | Condition Code | | | | | Flag Modification | | | |

Syntax

CCF

Operation

Zero → Carry Bit

Flags: C: Set to 0.

Description

The Clear Carry Flag instruction resets the carry flag with a 0.

Example

CCF

Initialization: SR contains 3000h

Instruction: CCF

Result: SR contains 2000h
C contains 0

This example uses one word of memory and executes in one machine cycle.

CIEF CLEAR INTERRUPT ENABLE FLAG CIEF

Instruction Word

| | | | | | | | | | | | | | | | |
|---------|----|----|----|----|----|---|----------------|---|---|---|---|-------------------|---|---|---|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| Op Code | | | | | | | Condition Code | | | | | Flag Modification | | | |

Syntax

CIEF

Operation

Zero → IE bit

Flags: IE: Set to 0.

Description

The Clear Interrupt Enable Flag instruction sets the IE flag to 0.

Example

CIEF

Initialization: SR contains 3080h

Instruction: CIEF

Result: SR contains 3000h
IE contains 0

This example uses one word of memory and executes in one machine cycle.

COPF CLEAR OVERFLOW PROTECTION FLAG COPCLEAR

Instruction Word

| | | | | | | | | | | | | | | | |
|---------|----|----|----|----|----|---|----------------|---|---|---|---|-------------------|---|---|---|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| Op Code | | | | | | | Condition Code | | | | | Flag Modification | | | |

Syntax

COPF

Operation

Zero → OP bit

Flags: P: Set to 0.

The Clear Overflow Protection Flag instruction resets the OP flag to 0.

Example

COPF

Initialization: SR contains 0100h

Instruction: COPF

Result: SR contains 0000h
 OP contains 0

This example uses one word of memory and executes in one machine cycle.

protection bit is set and an overflow occurs after the compare is executed. The accumulator updates with the appropriate low (800000h) or high (7FFFFFFh) limit.

Example 1

CP A, <regind>

Initialization: A contains 7A2500h
 P2:1 contains A4h
 RAM Bank1: A4h contains 5463h

Instruction: CP A, @P2:1

Result: A contains 7A2500h
 SR contains 1000h

This example uses one word of memory and executes in one machine cycle. The operand referenced by P2:1 is subtracted from the accumulator. $7A2500h - 546300h = 25C200h$. Because the comparison does not yield a 0, Bit 0 is not set. The content of the P2:1 register is appended with eight additional 0 bits. For consistent comparisons, the accumulator must contain zeros in the lower eight bits.

The accumulator is unaffected by the operation.

Example 2

CP A, <memind>

Initialization: A contains 7A2500h
 P2:1 contains A4h
 RAM Bank1: A4h contains 5463h
 ROM Address: 5463h contains 0C12h

Instruction: CP A, @@P2:1

Result: A contains 7A2500h
 P2:1 contains A4h
 RAM Bank1: A4h contains 5464h
 SR contains 1000h

This example uses one word of memory and executes in three machine cycles. The pointer P2:1 contains the RAM register location (A4h). The contents of this register have a ROM address. This address refers to the ROM data compared to the accumulator $7A2500h - 0C1200h = 6E1300h$. Because the comparison does not yield a 0, Bit 0 is not set. With memory indirect addressing,

the ROM address is automatically incremented. Using `CP A, @@P2:1 +` performs the same operation and also increments `P2:1` content to `A5h`.

Example 3

`CP A, <limm>`

Initialization: A contains `7A2500h`

Instruction: `CP A, #%7A25`

Result: A contains `7A2500h`
SR contains `3000h`

This example uses two words of memory and executes in two machine cycles. The immediate operand is compared to the accumulator. Because they are equal, the Zero Flag is set.

Example 4

`CP A, <hwregs>`

Initialization: A contains `7A2500h`
SR contains `0000h`
BUS contains `7A25h`

Instruction: `CP A, BUS`

Result: A contains `7A2500h`
SR contains `3000h`

This example uses one word of memory and executes in one machine cycle. The `<hwreg>` operand is subtracted from the accumulator. Because the two operands are equal, the zero-status bit is set high. `<hwregs>` can be compared from all hardware registers.

Example 5

`CP A, <direct>`

Initialization: Accumulator contains `7A2500h`
RAM Bank0 F3h contains `5463h`

Instruction: `CP A, %F3`

Result: A contains `7A2500h`
SR contains `1000h`

This example uses one word of memory and executes in one machine cycle. Register F3h is compared to the accumulator. $7A2500h - 546300h = 25C200h$. An equivalent instruction is CP A, 243 (173h = 243 decimal).

Example 6

CP A, <pregs>

Initialization: Accumulator contains 123400h
P0:0 contains 56h

Instruction: CP A, P0:0

Result: Accumulator contains 123400h
SR contains 1000h

This example uses one word of memory and executes in one machine cycle. The contents of the pointer register P0:0 are compared to the accumulator. $123400h - 005600h = 11DE00h$. The Pointer Register is connected to the lower 8 bits of the D-bus. The D-bus is connected to the upper 16 bits of the P-bus. This action changes the pointer register operand to 005600h before being compared to the accumulator.

Example 7

CP A, <dregs>

Initialization: A contains 7A2500h
D2:1 contains 5463h

Instruction: CP A, D2:1

Result: A contains 7A2500h
SR contains 1000h

This example uses one word of memory and executes in one machine cycle. The contents of the data pointer D2:1 are compared to the accumulator. $7A2500h - 546300h = 25C200h$.

DEC

DECREMENT

DEC

Instruction Word

| | | | | | | | | | | | | | | | |
|---------|----|----|----|----|----|---|----------------|----|----|----|----|------------------|---|---|---|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | cc | cc | cc | cc | cc | 0 | 1 | 0 | 1 |
| Op Code | | | | | | | Condition Code | | | | | ACC Modification | | | |

Syntax

DEC A

DEC <cc>, A

Operation

ACC - 1 → ACC

Flags: C: Set if carry is required for operation
 Z: Set if result is 0.
 N: Set if decrement results in a value less than 0.
 OV: Set if upper (7FFFFFFh) or lower (800000h) limits are exceeded.

Description

The accumulator decrements by 1. A condition code can be used to test for a specific condition before decrementing.

Example 1

DEC A

Initialization: A contains 7A2500h

Instruction: DEC A

Result: A contains 7A24FFh

This example uses one word of memory and executes in one machine cycle. The value in the accumulator decrements by 1.

Example 2

DEC <cc>, A

Initialization: A contains 7A2500h



Instruction: DEC MI, A

Result: A contains 7A2500h

This example uses one word of memory and executes in one machine cycle.
Because the accumulator is not negative, the decrement instruction is not
executed.

INC

INCREMENT

INC

Instruction Word

| | | | | | | | | | | | | | | | |
|---------|----|----|----|----|----|---|----------------|----|----|----|----|------------------|---|---|---|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | cc | cc | cc | cc | cc | 0 | 1 | 0 | 0 |
| Op Code | | | | | | | Condition Code | | | | | ACC Modification | | | |

Syntax

INC <cc>, A

INC A

Operation

ACC + 1 → ACC

Flags: C: Set if carry is required for operation.
 Z: Set if result is 0.
 N: Set if results in a value less than 0.
 OV: Set if upper (7FFFFFFh) or lower (800000h) limits are exceeded.

Description

The Increment instruction adds one to the accumulator. A condition code can be used to test for a specific condition for an increment to occur.

Example 1

INC <cc>, A

Initialization: A contains 7A2500h

Instruction: INC MI, A

Result: A contains 7A2500h

This example uses one word of memory and executes in one machine cycle. Because the accumulator is not negative, the increment instruction is not executed.

Example 2

INC A



Initialization: A contains 7A2500h

Instruction: INC A

Result: A contains 7A2501h

This example uses one word of memory and executes in one machine cycle. The value in the accumulator is incremented by 1.

JP

JUMP

JP

Instruction Word

| | | | | | | | | | | | | | | | |
|---------|----|----|----|----|----|---|----------------|----|----|----|----|----------|---|---|---|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 | 1 | 0 | cc | cc | cc | cc | cc | 0 | 0 | 0 | 0 |
| Op Code | | | | | | | Condition Code | | | | | Not Used | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| a | a | a | a | a | a | a | a | a | a | a | a | a | a | a | a |

16-Bit Address

Syntax

```
JP      <cc>,      <direct>
JP      <direct>
```

Operation

16-Bit address → PC

Flags: None

Description

The instruction places the address of the referenced ROM location in the Program Counter (PC). Because the processor obtains its next instruction address from the PC, the processor jumps to the appropriate location. A condition code can be used to test for a specific condition for a JUMP to occur.

Example 1

```
JP <cc>, <direct>
```

Initialization: Routine 1 address contains 1455h
 PC contains 1343h
 User 0 input contains 1

Instruction: JP NUO, Routine 1

Result: PC contains 1343h

This example uses two words of memory and executes in two machine cycles. Because the User 0 input is set high, the condition code is not met. Therefore,

the JUMP instruction does not execute. The User 0 input is used to control the flow of the software.

Example 2

JP <direct>

Initialization: Routine 1 address contains 1455h
 PC contains 1343h

Instruction: JP Routine 1

Result: PC contains 1455h

This example uses two words of memory and executes in two machine cycles. The value in the program counter is replaced by the Routine 1 address (1455h).

LD

LOAD

LD

Instruction Word

| | | | | | | | | | | | | | | | |
|---------|----|----|----|----|----|---|-------------|-------------|---|---|---|--------|---|---|---|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | b | 0 | 0 | 1 | 1 | s | s | s | s |
| Op Code | | | | | | | RAM Bank | Destination | | | | Source | | | |

Syntax

| | | |
|----|------------|----------|
| LD | A, | <pregs> |
| LD | <direct> , | A |
| LD | <hwregs> , | <dregs> |
| LD | A, | <dregs> |
| LD | <dregs> , | <hwregs> |
| LD | <hwregs> , | <limm> |
| LD | A, | <memind> |
| LD | <pregs> , | <simm> |
| LD | <hwregs> , | <accind> |
| LD | A, | <direct> |
| LD | <pregs> , | <hwreg> |
| LD | <hwregs> , | <memind> |
| LD | A, | <regind> |
| LD | <regind> , | <limm> |
| LD | <hwregs> , | <regind> |
| LD | A, | <hwregs> |
| LD | <hwregs> , | <pregs> |
| LD | <hwregs> , | <hwregs> |

Operation

<source> → <destination>

Flags: None

Description

The **LOAD** command provides the ability to transfer data to several different locations in the processor including hardware registers, accumulator, stack, pointers and memory. All transfers across the various internal buses are transparent to the user.

Notes:

1. A load using the **X** or **Y** register provides an automatic multiply operation. This means the operand can be obtained from any register location.
2. The **P** register is a read-only register, therefore the destination of the load cannot be the **P** register.
3. **LD EXTN, EXTN** is not allowed.
4. A **LOAD** instruction to the accumulator clears the lower 8 bits of the 24-bit accumulator.

Example 1

LD A, <regind>

Initialization: A contains 7A2500h
 P2:1 contains A4h
 RAM Bank1: A4h contains 5463h

Instruction: **LD A, @P2:1**

Result: A contains 546300h

This example uses one word of memory and executes in one machine cycle. Indirect addressing through the pointer registers provides access to RAM data. The data in RAM Bank 1, register A4 is transferred to the accumulator. The contents of the **P2:1** register are appended with eight additional 0 bits. This is added to verify a correct arithmetic comparison.

Example 2

LD A, <memind>

Initialization: Accumulator contains 123400h
 P0:0 contains 21h
 RAM Bank0: 21h contains 247Ah
 ROM Address: 247Ah equals OC1 2h

Instruction: LD A, @@P0:0
Result: A contains 0C1 200h
P0:0 contains 21h
RAM Bank0: 21h contains 247Bh

This example uses one word of memory and executes in three machine cycles. The pointer P0:0 contains the RAM register location (21h). The contents of this register have a ROM address. This address refers to the ROM data that loads to the accumulator. When memory indirect addressing is used, the ROM address is automatically incremented. Using LD A, @@P0:0+ has the same result, also incrementing the P0:0 content to 22h.

Example 3

LD A, <limm>

Initialization: Accumulator contains 123400h
Instruction: LD A, #2474
Result: A contains 247400h

This example uses two words of memory and executes in two machine cycles. The immediate operand 2474h loads to the accumulator.

Example 4

LD A, <hwregs>

Initialization: Accumulator contains 123400h
Register X contains 0C12h
Instruction: LD A, X
Result: A contains 0C1200h

This example uses one word of memory and executes in one machine cycle. The contents of Register x are loaded to the accumulator. All hardware registers can transfer from <hwregs>.

Example 5

LD A, <direct>

Initialization: Accumulator contains 123400h
RAM Bank0 F3h contains 0C12h
Instruction: LD A, %F3

Result: A contains 0C1200h

This example uses one word of memory and executes in one machine cycle. Register F3h is loaded to the accumulator. An equivalent instruction is LD A, 243 (F3h = 243 decimal).

Example 6

LD A, <dregs>

Initialization: Accumulator contains 123400h
D0: 1 contains 8746h

Instruction: LD A, D0:1

Result: A contains 874600h
D0: 1 contains 8746h

This example uses one word of memory and executes in one machine cycle. The contents of the data pointer D0:1 load to the accumulator.

Example 7

LD A, <pregs>

Initialization: Accumulator contains 123400h
P0:0 contains 56h

Instruction: LD A, P0:0

Result: Accumulator contains 005600h

This example uses one word of memory and executes in one machine cycle. The contents of the pointer register P0:0 are loaded to the accumulator. The Pointer Register is connected to the lower 8 bits of the D-bus. The D-bus is connected to the upper 16 bits of the P-bus. This operation causes the pointer register operand to become 005600h before being loaded into the accumulator.

Example 8

LD <direct>, A

Initialization: Accumulator contains 123400h
RAM Bank0: 3Ch contains 5678h

Instruction: LD %3C, A

Result: Accumulator contains 123400h
RAM Bank0: 3Ch contains 1234h

This example uses one word of memory and executes in one machine cycle. The current value in the accumulator is loaded to the register addressed by the instruction (3Ch). An equivalent instruction is LD 60, A. (3Ch = 60 decimal).

Example 9

LD <pregs>, <simm>

Initialization: P2:0 contains 2Fh
RAM Bank0: 3Fh contains 254645h

Instruction: LD P2:0, #3F

Result: P2:0 contains 3Fh
RAM Bank0: 3Fh contains 254645h

This example uses one word of memory and executes in one machine cycle. The immediate data (3Fh) is loaded into the pointer register P2:0. This action provides a convenient method for initializing pointer registers.

Example 10

LD <pregs>, <hwregs>

Initialization: P0:1 contains 2Fh
Y contains 2376h

Instruction: LD P0:1, Y

Result: P0:1 contains 76h
Y contains 2376h

This example uses one word of memory and executes in one machine cycle. The lower 8-bits of the Y register are transferred to the pointer register P0:1. All hardware registers can transfer from <hwregs>.

Example 11

LD <regind>, <limm>

Initialization: P0:1 contains 2Fh
RAM Bank0: 2Fh contains 2376h

Instruction: LD @P0:1, #35B8

Result: P0:1 contains 2Fh
RAM Bank1: 2Fh contains 35B8h

This example uses two words of memory and executes in two machine cycles. The immediate operand 35B8h is transferred to the Register 2Fh of RAM Bank1.

Example 12

LD <hwregs>, <pregs>

Initialization: P2:0 contains 2Fh
X Register contains 8B87h

Instruction: LD X,P2:0

Result: P2:0 contains 2Fh
X Register contains 002Fh

This example uses one word of memory and executes in one machine cycle. The contents of the P2:0 pointer (2Fh) are loaded into the X register. Transfer to <hwreg> is possible to all hardware registers except the read-only P register.

Example 13

LD <hwregs>, <dregs>

Initialization: D2:0 contains 3C87h
Accumulator contains 8BB722h

Instruction: LD A,D2:0

Result: D2:0 contains 3C87h
Accumulator contains 3C8700h

This example uses one word of memory and executes in one machine cycle. The contents of the D2:0 pointer (3C87h) are loaded into the accumulator. Transfer to <hwregs> is possible to all hardware registers except the read-only P register.

Example 14

LD <hwregs>, <limm>

Initialization: Stack0 contains 8B2Fh
Stack1 contains 0000h

Instruction: LD Stack, #35B8

Result: Stack0 contains 35B8h
Stack1 contains 8B2Fh

This example uses two words of memory and executes in two machine cycles. The immediate data is pushed onto the stack as previous stack data is pushed up

the stack. Transfer to <hwregs> is possible to all hardware registers except the read-only P register.

Example 15

LD <hwregs>, <accind>

Initialization: EXT7 Register contains 8B87h
Accumulator contains 77B6h
ROM 77B6h contains 387Dh

Instruction: LD EXT7, @A

Result: EXT7 Register contains 387Dh
Accumulator contains 77B6h

This example uses one word of memory and executes in one machine cycle. The contents of the ROM Register 77B6h (387Dh) are loaded into External Register 7. Transfer to <hwregs> is possible to all hardware registers except the read-only P register and the accumulator register.

Example 16

LD <hwregs>, <memind>

Initialization: Y Register contains 1234h
P0:0 contains 21h
RAM Bank0: 21h contains 247Ah
ROM Address: 247Ah contains 0C12h

Instruction: LID Y, @@P0:0

Result: Y Register contains 0C12h
P0:0 contains 21h
RAM Bank0: 21h contains 247Bh

This example uses one word of memory and executes in three machine cycles. The pointer P0:0 contains the RAM register location (21h). The contents of this register have a ROM address that refers to the ROM data that loads to the Y register. Transfer to <hwregs> is possible to all hardware registers except the read-only P register. When memory indirect addressing is used the ROM address is automatically incremented. Using LD A, @@P0:0+ performs the same operation and also increments the P0:0 content to 22h.

Example 17

LD <hwregs>, <regind>

Initialization: X Register contains 7A25h
 P21 contains A4h
 RAM Bank1: A4h contains 5463h

Instruction: LD X, @P2:1

Result: X Register contains 5463h

This example uses one word of memory and executes in one machine cycle. Indirect addressing through the pointer registers provides access to RAM data. The data in RAM bank 1, register A4 is transferred to the X register. Transfer to <hwreg> is possible to all hardware registers except the read-only P register.

Example 18

LD <hwregs>, <hwregs>

Initialization: X Register contains 7A25h
 EXT5 Register contains 789Ah

Instruction: LD X, EXT5

Result: X Register contains 789Ah
 EXT5 Register contains 789Ah

This example uses one word of memory and executes in one machine cycle. The EXT5 Register contents are transferred to the X register. Transfer to <hwregs> is possible to all hardware registers except the read-only P register.

NEG

NEGATE

NEG

Instruction Word

| | | | | | | | | | | | | | | | |
|---------|----|----|----|----|----|---|----------------|----|----|----|----|------------------|---|---|---|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | cc | cc | cc | cc | cc | 0 | 1 | 1 | 0 |
| Op Code | | | | | | | Condition Code | | | | | ACC Modification | | | |

Syntax

NEG A

NEG <cc>, A

Operation

-ACC → ACC

Flags: N Set if result is a negative number.

Two special cases are:

1. If ACC contains 000000 after execution, then N and OV are cleared, and Z and C are set
2. If ACC contains 800000 after execution, then N and OV are set; and Z and C are cleared.

The accumulator is replaced with a negative of the current value. To achieve this state, the two's complement is performed.

Example 1

NEG A

Initialization: A contains 003654h

Instruction: NEG A

Result: A contains FFC9ACh

This example uses one word of memory and executes in one machine cycle.

Example 2

NEG <CC>, A

Initialization: A contains 000111h
 Carry bit contains 1

Instruction: NEG C, A

Result: A contains FFFEEFh

This example uses one word of memory and executes in one machine cycle.

NOP

NO OPERATION

NOP

Instruction Word

| | | | | | | | | | | | | | | | |
|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Syntax

NOP

Operation

PC+ 1 → PC

Flags: None

Description

The NOP instruction causes the processor to continue operation for one cycle without affecting previous registers and I/O.

OR

BITWISE OR

OR

Instruction Word

| | | | | | | | | | | | | | | | |
|---------|----|----|----|----|----|---|-------------|-------------|---|---|---|--------|---|---|---|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 1 | b | 0 | 0 | 0 | 0 | s | s | s | s |
| Op Code | | | | | | | RAM Bank | Destination | | | | Source | | | |

Syntax

OR A, <regind>

OR A, <memind>

OR A, <limm>

OR A, <hwregs>

OR A, <direct>

OR A, <pregs>

OR A, <dregs>

OR A, <simm>

Operation

ACC .OR. source → ACC

Flags: N Set If result in accumulator is negative.
 Z Set If result is 0.

Description

The accumulator performs an OR instruction on the contents of the specified register. The upper 16 bits of the accumulator are used. The result is placed in the accumulator. The OR instruction is frequently used to compare specific bits to assist in program control.

The lower eight bits of the accumulator are unchanged after execution of the OR instruction.

Example 1

OR A, <regind>

Initialization: Accumulator contains 3264A0h
P0:0 contains E2h
RAM Bank0: E2h contains 1126h

Instruction: OR A, @P0:0

Result: A contains 336600h

This example uses one word of memory and executes in one machine cycle. Use an OR instruction to reference the operand P0:0 with the upper 16 bits of the accumulator. The result is stored in the accumulator. 3264A0h OR. 1126A0h = 3366A0h.

Example 2

OR A, <memind>

Initialization: A contains 3264A0h
P2:1 contains A4h
RAM Bank 1: A4h contains 5463h
ROM Address: 5463h contains 1126h

Instruction: OR A, @@P2:1

Result A contains 3366A0h
P2:1 contains A4h
RAM Bank0: A4h contains 5464h
SR contains 0000h

This example uses one word of memory and executes in three machine cycles. The pointer P2:1 contains the RAM register location (A4h). The contents of this register have a ROM address. This address refers to the ROM data that is compared to the accumulator. 3264A0h.OR.112600h = 3366A0h. With memory indirect addressing, the ROM address is automatically incremented. Using ORA, @@P0:0+ performs the same operation and also increments the P0:0 content to A5h.

Example 3

OR A, <limm>

Initialization: A contains 3264A0h



Instruction: OR A, #%1126

Result: A contains 3366A0h
 SR contains 0000h

This example uses two words of memory and executes in two machine cycles.
The accumulator performs an OR instruction on the immediate data.

POP POP VALUE FROM STACK POP

Instruction Word

| | | | | | | | | | | | | | | | |
|---------|----|----|----|----|----|----|-------------|-------------|---|---|---|--------|---|---|---|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| op | op | op | op | op | op | op | b | d | d | d | d | s | s | s | s |
| Op Code | | | | | | | RAM Bank | Destination | | | | Source | | | |

Syntax

```
POP <pregs>
POP <dregs>
POP <regind>
POP <hwregs>
```

Operation

```
STACK 0 -> <destination>
Stack n -> Stack N-1
```

Flags: None

Description

The current value of the stack is copied to the specified register. Because the stack is a last-in, first-out (LIFO) hard-wired architecture, the copy and shift of data remaining in the stack are all performed in a single cycle.

The POP instruction provides the ability to control information sent to the stack, making it possible to expand the stack using software.

Example 1

```
POP <pregs>
```

Initialization: Stack 1 contains 0426h
 Stack 0 contains 0C06h
 P0:0 contains 24h

Instruction: POP P0:0

Result Stack 0 contains 0426h
 P0:0 contains 06h

This example uses one word of memory and executes in one machine cycle. The destination of `Stack 0` (item on top of stack) is `P0:0`. The 8-LSBs of the data in `stack 0` are loaded into `P0:0`. At transfer, `Stack 1` is automatically moved to `Stack 0`.

Example 2

`POP <dregs>`

Initialization: Stack 1 contains 0426h
 Stack 0 contains 0C06h
 D0:0 contains 7676h

Instruction: POP D0:0

Result: Stack 0 contains 0426h
 D0:0 contains 0C06h

This example uses one word of memory and executes in one machine cycle. The destination of the `Stack 0` (item on top of stack) is given by `D0:0`. When transferred, `Stack 1` is automatically moved to `Stack 0`.

Example 3

`POP <regind>`

Initialization: Stack 1 contains 0426h
 Stack 0 contains 0C06h
 P0:0 contains 24h
 RAM Bank0: 24h contains 42A4h

Instruction: POP @P0:0

Result Stack 0 contains 0426h
 RAM Bank0: 24h contains 0C06h

This example uses one word of memory and executes in one machine cycle. The destination of the `Stack 0` (item on top of stack) is given by `P0:0`. 24h is the register location in RAM Bank0 to which the stack item is transferred. At transfer, `Stack 1` is automatically moved to `Stack 0`.

Example 4

POP <hwregs>

Initialization: Stack 1 contains 0426h
 Stack 0 contains 0C06h
 X Register contains 089Ch

Instruction: POP X

Result: Stack 0 contains 0426h
 X Register contains 0C06h

This example uses one word of memory and executes in one machine cycle. The destination of the Stack 0 (item on top of stack) is given by the X Register. At transfer, Stack 1 is automatically moved to Stack 0. Transfer to <hwregs> is possible to all hardware registers except the read-only P register.

PUSH PUSH VALUE ONTO STACK PUSH

Instruction Word

| | | | | | | | | | | | | | | | |
|---------|----|----|----|----|----|---|-------------|-------------|---|---|---|--------|---|---|---|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 1 | b | 0 | 0 | 0 | 0 | s | s | s | s |
| Op Code | | | | | | | RAM Bank | Destination | | | | Source | | | |

Syntax

```

PUSH <pregs>
PUSH <dregs>
PUSH <memind>
PUSH <accind>
PUSH <regind>
PUSH <hwregs>
PUSH <direct>
PUSH <limm>

```

Operation

```

<source> -> Stack
Stack n -> Stack n+ 1

```

Flags: None

Description

The contents of the specified register are placed on the stack. Because the stack is a last-in, first-out (LIFO) hard-wired architecture, the placement and shifting of current stack data is performed in a single cycle.

The **PUSH** instruction provides the ability to control information sent to the stack, making it possible to expand the stack through software.

Example 1

```

PUSH <pregs>

```

Initialization: Stack 0 contains 0C06h
 P1:1 contains A4h

Instruction: PUSH P1:1

Result Stack 1 contains 0C06h
 Stack 0 contains 00A4h

This example uses one word of memory and executes in one machine cycle. The pointer P1:1 contains the 8-bit value A4h. The 16-bit value, 00A4h, is pushed onto the stack. At transfer, Stack 0 is automatically moved to Stack 1.

Example 2

PUSH <dregs>

Initialization: Stack 1 contains 0426h
 Stack 0 contains 0C06h
 D1:1 contains 42A4h

Instruction: PUSH D1:1

Result Stack 1 contains 0C06h
 Stack 0 contains 42A4h

This example uses one word of memory and executes in one machine cycle. The pointer D1:1 is pushed onto the stack. At transfer, Stack 0 is automatically moved to Stack 1.

Example 3

PUSH <memind>

Initialization: Stack 1 contains 0426h
 Stack 0 contains 0C06h
 RAM Bank0: 24h contains 42A4h
 P1:1 contains A4h
 RAM Bank1: A4h contains 5463h
 ROM Address 5463h contains 1126h

Instruction: PUSH @@P1:1

Result: Stack 1 contains 0C06h
 Stack 0 contains 1126h
 RAM Bank1: A4h contains 5464h

This example uses one word of memory and executes in three machine cycles. When memory indirect addressing is used, the ROM address is automatically incremented. Using `OR A, @@P0:0+` performs the same operation and also increments the `P0:0` content to `A5h`.

Example 4

`PUSH <accind>`

Initialization: Stack 1 contains `0426h`
 Stack 0 contains `0C06h`
 Accumulator contains `42A4h`
 ROM address `42A4h` contains `4C45h`

Instruction: `PUSH @A`

Result Stack 1 contains `0C06h`
 Stack 0 contains `4C45h`

This example uses one word of memory and executes in one machine cycle. Indirect addressing with the accumulator points to the ROM memory (`42A4h`). The data in this location is pushed onto the stack. At transfer, Stack 0 is automatically moved to Stack 1.

Example 5

`PUSH <regind>`

Initialization: Stack 1 contains `0426h`
 Stack 0 contains `0C06h`
 `P1:1` contains `A4h`
 RAM Bank1: `A4h` contains `42A4h`

Instruction: `PUSH @P1:1`

Result: Stack 1 contains `0C06h`
 Stack 0 contains `42A4h`
 RAM Bank1: `A4h` contains `42A4h`

This example uses one word of memory and executes in one machine cycle. The pointer `P1:1` contains the RAM register location (`A4h`). The data at this location is pushed onto the stack. At transfer, Stack 0 is automatically moved to Stack 1.

Example 6

`PUSH <hwregs>`

Initialization: Stack 1 contains 0426h
 Stack 0 contains 0C06h
 X Register contains 42A4h

Instruction: PUSH X

Result: Stack 1 contains 0C06h
 Stack 0 contains 42A4h

This example uses one word of memory and executes in one machine cycle. The data in the X register is pushed onto the stack. At transfer, Stack 0 is automatically moved to Stack 1. Transfer from <hwregs> is possible from all hardware registers.

Example 7

PUSH <direct>

Initialization: Stack 1 contains 0426h
 Stack 0 contains 0C06h
 RAM Bank0: 24h contains 42A4h

Instruction: PUSH %24

Result: Stack 1 contains 0C06h
 Stack 0 contains 42A4h

This example uses one word of memory and executes in one machine cycle. The instruction (24h) provides the direct register address. The value contained in this register is pushed onto the stack (42A4h). At transfer, Stack 0 is automatically moved to Stack 1.

Example 8

PUSH <limm>

Initialization: Stack 1 contains 0426h
 Stack 0 contains 0C06h

Instruction: PUSH #5757

Result: Stack 1 contains 0C06h
 Stack 0 contains 5757h

This example uses two words of memory and executes in two machine cycles. The immediate operand 5757h is pushed onto the stack. At transfer, Stack 0 is automatically moved to Stack 1.

RET RETURN FROM SUBROUTINE RET

Instruction Word

| | | | | | | | | | | | | | | | |
|---------|----|----|----|----|----|---|-------------|-------------|---|---|---|--------|---|---|---|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 |
| Op Code | | | | | | | RAM Bank | Destination | | | | Source | | | |

Syntax

RET

Operation

Stack 0 → PC

Stack n → Stack n-1

Flags: None

Description

The current stack information is popped from the stack and placed in the Program Counter (PC) register. The jump is made from the subroutine via the PC.

Example

RET

Initialization: Stack 1 contains 0624
 Stack 0 contains 0401
 PC contains 06DF

Instruction: RET

Result: Stack 0 contains 0624
 PC contains 0401h

This example uses one word of memory and executes in one machine cycle.

RL ROTATE LEFT RL

Instruction Word

| | | | | | | | | | | | | | | | |
|---------|----|----|----|----|----|---|----------------|----|----|----|----|------------------|---|---|---|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | cc | cc | cc | cc | cc | 0 | 1 | 1 | 1 |
| Op Code | | | | | | | Condition Code | | | | | ACC Modification | | | |

Syntax

RL A

RL <cc>,A

Operation

C <= 23 ----- < ----- 8 <= C

Flags: N Set if result of accumulator is negative
 Z Set if result is zero.
 C Set if MSB is set before rotate.

Description

The upper 16 bits of the accumulator are rotated left through the carry bit. The lower 8 bits remain unchanged while the resultant LSB, bit 0, is placed with the value 0 (see the accumulator section).

Example 1

RL A

Initialization: A contains 226A84h
 Carry bit contains 1

Instruction: RL A

Result: A contains 44D584h
 Carry bit contains 0

This example uses one word of memory and executes in one machine cycle. The upper 16 bits (226Ah) are shifted left through the carry bit to produce 44D5h. The lower 8 bits (84h) remain unchanged.

Example 2

RL <CC>, A

Initialization: A contains 226A84h
 Carry bit contains 0, Z=0

Instruction: RL Z, A

Result: A contains 226A84h

This example uses one word of memory and executes in one machine cycle. The condition code 0 is not set; therefore, the instruction is not executed.

RR ROTATE RIGHT RR

Instruction Word

| | | | | | | | | | | | | | | | |
|---------|----|----|----|----|----|---|----------------|----|----|----|----|------------------|---|---|---|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | cc | cc | cc | cc | cc | 0 | 0 | 0 | 0 |
| Op Code | | | | | | | Condition Code | | | | | ACC Modification | | | |

Syntax

RR A

RR <cc>, A

Operation

C =>23 ---- > 8 = => 7 - -> -- 0 -> discarded

Flags: N Set if result of accumulator is negative.
Z Set if result is 0.
C Set if LSB is set before rotate.

Description

The upper 16 bits of ACC are rotated right through the carry bit. The MSB of the lower 8 bits also obtains the data shifted from the LSB of upper 16 bits. The lower 8 bits are shifted right with the LSB being discarded.

Example 1

RR A

Initialization: A contains 226A84h
Carry bit contains 0

Instruction: RR A

Result: A contains 113542h

This example uses one word of memory and executes in one machine cycle. The upper 16 bits (226Ah) are shifted right through the carry bit to produce 1135h. The lower 8 bits (84h) are shifted right to provide 42h.

Example 2

RR <cc>, A



Initialization: A contains 226A84h
 Carry bit contains 0, Z=0

Instruction: RR Z, A

Result: A contains 226A84h

This example uses one word of memory and executes in one machine cycle. The condition code 0 is not set; therefore, the instruction is not executed.

SCF

SET CARRY FLAG

SCF

Instruction Word

| | | | | | | | | | | | | | | | |
|---------|----|----|----|----|----|----|----------------|----|----|----|----|-------------------|----|----|----|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| op | op | op | op | op | op | op | cc | cc | cc | cc | cc | fm | fm | fm | fm |
| Op Code | | | | | | | Condition Code | | | | | Flag Modification | | | |

Syntax

SCF

Operation

1 → Carry Bit

Flags: C Set to 1.

Description

The Set Carry Flag instruction places a one in the carry bit (bit 12 of the Status Register).

Example

SCF

Initialization: SR contains 2000h

Instruction: SCF

Result: SR contains 3000h
C contains 1

This example uses one word of memory and executes in one machine cycle.

SIEF SET INTERRUPT ENABLE FLAG SIEF

Instruction Word

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---------|----|----|----|----|----|---|----------------|----|----|----|----|-------------------|---|---|---|
| 1 | 0 | 0 | 1 | 0 | 1 | 0 | cc | cc | cc | cc | cc | 0 | 1 | 0 | 1 |
| Op Code | | | | | | | Condition Code | | | | | Flag Modification | | | |

Syntax

SIEF

Operation

1 → IE bit

Flags: None

Description

The instruction places a 1 in bit 7 of the status register and is used to enable interrupts.

Example

SIEF

Initialization: SR contains 3000h

Instruction: SIEF

Result: SR contains 3080h
IE contains 1

This example uses one word of memory and executes in one machine cycle.

SLL

SHIFT LEFT LOGICAL

SLL

Instruction Word

| | | | | | | | | | | | | | | | |
|---------|----|----|----|----|----|---|----------------|----|----|----|----|------------------|---|---|---|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | cc | cc | cc | cc | cc | 0 | 0 | 1 | 1 |
| Op Code | | | | | | | Condition Code | | | | | ACC Modification | | | |

Syntax

SLL A

SLL <cc>, A

Operation

discarded ← C ≤ 23 - - - - < - - - -0 ≤ 0

Flags: N Set if result of accumulator is negative
 (bit 23 set to 1).
 Z Set if result is 0.
 C Set if MSB is set before shift.

Description

All 24 bits of the accumulator are shifted left through the carry bit. The MSB, bit 23, passes through the carry bit before being discarded. The LSB, bit 0, is filled with a zero. Subsequent shifts cause additional zeroes to be shifted in.

Example 1

SLL A

Initialization: A contains 226A84h
 Carry bit contains 0

Instruction: SLL A

Result: A contains 226A84h

This example uses one word of memory and executes in one machine cycle. All 24 bits of the accumulator are shifted left through the carry bit, producing the result 44D508h.

Example 2

SLL <CC>, A

Initialization: A contains 226A84h
 Carry bit contains 0

Instruction: SLL MI, A

Result: A contains 226A84h

This example uses one word of memory and executes in one machine cycle. The condition code N is not set, and the instruction is not executed.

SOPF SET OVERFLOW PROTECTION FLAG SOPF

Instruction Word

| | | | | | | | | | | | | | | | |
|---------|----|----|----|----|----|---|----------------|---|---|---|---|-------------------|---|---|---|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| Op Code | | | | | | | Condition Code | | | | | Flag Modification | | | |

Syntax

SOPF

Operation

1 → OP bit

Flags: None

Description

The Set Overflow Protection Flag instruction places a one in bit 8 of the status register. If an ALU operation exceeds the limits of the processor, the overflow protection sets the overflow flag (OV) and holds the limit value in the accumulator.

Example

SOPF

Initialization: SR contains 0000h

Instruction: SOPF

Result: SR contains 0100h
OP contains 1

This example uses one word of memory and executes in one machine cycle.

SRA SHIFT RIGHT ARITHMETIC SRA

Instruction Word

| | | | | | | | | | | | | | | | |
|---------|----|----|----|----|----|---|----------------|----|----|----|----|------------------|---|---|---|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | cc | cc | cc | cc | cc | 0 | 0 | 0 | 0 |
| Op Code | | | | | | | Condition Code | | | | | ACC Modification | | | |

Syntax

SRA A

SRA <cc>, A

Operation

23 => 23 --- > ---- 0 => discarded

Flags: N Set if result of accumulator is negative.
Z Set if result is 0.
C Set if LSB is set before the shift.

Description

All 24 bits of the accumulator are shifted right with sign extension through the carry bit. The MSB, bit 23, is replicated in vacated bits. The LSB, bit 0, is passed through the carry before it is discarded.

Example 1

SRA A

Initialization: A contains 226A84h
Carry bit contains 0

Instruction: SRA A

Result: A contains 113542h
Carry bit contains 0

This example uses one word of memory and executes in one machine cycle. All 24 bits of the accumulator are shifted right. The MSB, bit 23, is copied into bit 22. The LSB, bit 0, is discarded.

Example 2

SRA <cc>, A

Initialization: A contains 226A84h
 Carry bit contains 0, N=0

Instruction: SRA A

Result: A contains 113542h
 Carry bit contains 0

This example uses one word of memory and executes in one machine cycle. The condition code is set; therefore, the instruction is executed. The initialization of the accumulator sets the PL (NN) condition code.

SUB

SUBTRACT

SUB

Instruction Word

| | | | | | | | | | | | | | | | |
|---------|----|----|----|----|----|---|-------------|-------------|---|---|---|--------|---|---|---|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | b | 0 | 0 | 0 | 0 | s | s | s | s |
| Op Code | | | | | | | RAM Bank | Destination | | | | Source | | | |

Syntax

SUB A, <regind>

SUB A, <memind>

SUB A, <limm>

SUB A, <hwregs>

SUB A, <direct>

SUB A, <pregs>

SUB A, <dregs>

SUB A, <simm>

Operation

ACC - (Source) → ACC

| | |
|----------|---|
| Flags: C | Set if a carry from the most significant bit is performed. |
| N | Set if the result in the accumulator is negative. |
| Z | Set if the result is 0. |
| OV | Set if the addition exceeds the upper (7FFFFFFh) or lower (800000h) limit of the accumulator. |

Description

The addressed data memory operand is subtracted from the accumulator. The result is loaded into the accumulator.

The lower eight bits of the accumulator are cleared by the execution of the subtract instruction.

Example 1

SUB A, <regind>

Initialization: Accumulator contains 874600h
P0:1 contains 45h
RAM Bank1: 45h contains 1234h

Instruction: SUB A, @P0:1

Result: A contains 751200h
@P0:1 contains 1234h

This example uses one word of memory and executes in one machine cycle. The contents of the register pointed by P0:1 are subtracted from the accumulator. The difference is contained in the accumulator and the pointer is left unchanged. The register pointer contains 45h. Because the pointer references RAM Bank1, the absolute register is 145h (325). Therefore, the contents of register 145h are subtracted from the accumulator. $874600h - 123400h = 751200h$. The direct addressing equivalent is SUB A, %145 or SUB A, 325.

Example 2

SUB A, <memind>

Initialization: Accumulator contains 874600h
P0:0 contains 21h
RAM Bank0: 21h contains 247Ah
ROM Address: 247Ah contains 1234H

Instruction: SUB A, @@P0:0

Result: A contains 751200h
P0:0 contains 22h
RAM Bank0: 21 h contains 247Bh

This example uses one word of memory and executes in three machine cycles. The pointer is used for memory indirect addressing. The pointer contains the address of the RAM (address 247Ah). The RAM contains the address of the requested ROM data (data 247Ah). This operand is subtracted from the accumulator. $874600h - 123400h = 751200h$.

Example 3

SUB A, <limm>

Initialization: Accumulator contains 874600h

Instruction: SUB A, #%1234

Result: Accumulator contains 751200h

This example uses two words of memory and executes in two machine cycles. The immediate operand 8746h is subtracted from the accumulator. $874600h - 123400h = 751200h$.

Example 4

SUB A, <hwregs>

Initialization: Accumulator contains 874600h
Register X contains 1234h

Instruction: SUB A, X

Result: A contains 751200h

This example uses one word of memory and executes in one machine cycle. The contents of Register X are subtracted from the accumulator. $874600h - 123400h = 751200h$. Transfer from <hwregs> is possible from all hardware registers.

Example 5

SUB A, <direct>

Initialization: Accumulator contains 874600h
RAM Bank0: F3h contains 1234h

Instruction: SUB A, %F3

Result: A contains 751200h

This example uses one word of memory and executes in one machine cycle. The contents of register F3h are subtracted from the accumulator. $874600h - 123400h = 751200h$. An equivalent instruction is SUB A, 243 (F3h = 243 decimal).

Example 6

SUB A, <dregs>

Initialization: Accumulator contains 874600h
D0: 1 contains 1234h

Instruction: SUB A, D0:1

Result: A contains 751200h
 D0:1 contains 1234h

This example uses one word of memory and executes in one machine cycle. The contents of the data pointer D0:1 are subtracted from the accumulator. The difference is contained in the accumulator and the pointer is left unchanged.

Example 7

SUB A, <pregs>

Initialization: Accumulator contains 874600h
 P0:0 contains 56h

Instruction: SUB A, P0:0

Result: Accumulator contains 86F000h

This example uses one word of memory and executes in one machine cycle. The contents of pointer register P0:0 are subtracted from the accumulator. $874600h - 005600h = 86F000h$. The Pointer Register is connected to the lower 8 bits of the D-bus. The D-bus is connected to the upper 16 bits of the P-bus. This causes the pointer register operand to become 005600h before it is subtracted from the accumulator.

XOR BITWISE EXCLUSIVE OR XOR

Instruction Word

| | | | | | | | | | | | | | | | |
|---------|----|----|----|----|----|---|-------------|-------------|---|---|---|--------|---|---|---|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 | 1 | b | 0 | 0 | 0 | 0 | s | s | s | s |
| Op Code | | | | | | | RAM Bank | Destination | | | | Source | | | |

Syntax

```

XOR A, <regind>
XOR A, <memind>
XOR A, <limm>
XOR A, <hwregs>
XOR A, <direct>
XOR A, <pregs>
XOR A, <dregs>
XOR A, <simm>

```

Operation

A.XOR.<operand> -> A

Flags: C Set if carry from the most significant bit is performed.

N Set if result in the accumulator is negative.

Z Set if result is 0.

OV Set if operation exceeds upper (7FFFFFFh) or lower (800000h) limit of accumulator.

Description

With the accumulator, perform an XOR instruction on the addressed data memory operand. The result loads into the accumulator.

The lower eight bits of the accumulator are cleared by the XOR instruction.

Example 1

XOR A, <regind>

Initialization: Accumulator contains 005600h
P0:0 contains 00h
RAM Bank0: 00h contains 1234h

Instruction: XOR A, @P0:0

Result: A contains 126200h

This example uses one word of memory and executes in one machine cycle. The pointer is used for memory indirect addressing. The pointer contains the address of the RAM (address 00h). Location 00h has operand 1234h. With the accumulator, 005600h, perform an XOR instruction to obtain the result 126200h.

Example 2

XOR A, <memind>

Initialization: A contains 3264A0h
P2:1 contains A4h
RAM Bank1: A4h contains 5463h
ROM Address: 5463h contains 1126h

Instruction: XOR A, @@P2:1

Result: A contains 2342A0h
P2:1 contains A41h
RAM Bank1: A4h contains 5464h
SR contains 0000h

This example uses one word of memory and executes in three machine cycles. The pointer P2:1 contains the RAM register location(A4h). The contents of this register have a ROM address. This address refers to the ROM data compared to the accumulator. $3264A0h.XOR.112600h = 2342A0h$. When indirect memory addressing is used, the ROM address is automatically incremented. Using XOR A, @@P2:1+ performs the same operation and also increments the P21 content to A5h.

Example 3

XOR A, <limm>

Initialization: A contains 3264A0h

Instruction: XOR A, #%1126

Result: A contains 2342A0h
SR contains 0000h

This example uses two words of memory and executes in two machine cycles. Perform an XOR instruction on the immediate data.

Example 4

XOR A, <hwreg>

Initialization: A contains 3264A0h
SR contains 0000h
BUS contains 1126h

Instruction: XOR A, BUS

Result: A contains 2342A0h
SR contains

This example uses one word of memory and executes in one machine cycle. With the accumulator, perform an XOR instruction on the <hwreg> operand.

Example 5

XOR A, <direct>

Initialization: Accumulator contains 3264A0h
RAM Bank0: F3h contains 1126h

Instruction: XOR A, %F3

Result: A contains 2342A0h
SR contains 0000h

This example uses one word of memory and executes in one machine cycle. Register F3h is compared to the accumulator. $3264A0h \text{ XOR } 112600h = 2342A0h$. An equivalent instruction is XOR A, 243 (F3h = 243 decimal).

5 System Design Considerations

The Z90T366 provides the ability to

- decode closed-caption transmissions
- display characters on the screen
- manipulate analog and digital control circuits
- monitor keypad button and infrared signals directly
- generate OSD if the Z90T366 receives vertical and horizontal synchronization signals

In a typical system, normal transmission is received and demodulated. The signals received from the color decoder and deflection unit control the CRT display. To display characters generated by the Z90T366 requires a video multiplexor which enables the CRT display's RGB signals and synchronization to be controlled by the video outputs from the processor. When the controller has to display a character on the screen, the multiplexor is switched, and the processor's video signals appear on the display.

The band-limited, A/C-coupled composite video signal is clamped internally to the negative reference voltage (REF-) during the back porch interval. It is then passed to the analog-to-digital converter through a 6:1 multiplexor. The digital signal is then decoded to extract the closed-caption text embedded in the video signal. The characters received are generated as video signals and are then passed to the display.

When a detectable composite video signal is received, the SYNC separator extracts the horizontal and vertical synchronization signals and passes them to the deflection module of the television. The FLYBACK signals from the deflection coils are fed back to the Z90T366. The controller uses these signals to align its video signals with those of the normal display. If the composite video signal is not present, video synchronization is provided by the controller. In this case, the SYNC signal pins are set to be outputs. The pins then feed to the deflection unit which controls the display. The SYNC generators can be configured to provide either HSYNC and VSYNC, or H-FLYBACK and V-FLYBACK.

Analog functions such as volume and color controls can be controlled by pulse width modulated outputs from the Z90T366. Additional digital controls like channel fine tuning are controlled via the serial I²C bus.

An infrared remote control receiver can be directly decoded through the capture register, and keypad input can be scanned by directly controlling I/O pins as keyscan ports.

The processor clock is available by referencing an internal phase locked loop to an external 32.768 KHz crystal oscillator. The oscillator minimizes EMI emissions from the clock circuitry. The internal system clock frequency can be selected as 12.058 MHz in normal operation or 32.768 KHz in low power consumption SLEEP mode (usually used if there is a general system power failure). The Z90T366's STOP mode suspends processor clocking for a power-down.

Program, display, and character graphics memory are on the chip, eliminating the requirement for external memory components. Characters can be displayed as two or three times normal size. Smoothing and fringing circuits enhance display appearance.

Figure 28 diagrams a typical application of the Z90T366 Television Controller as an embedded controller in a television.

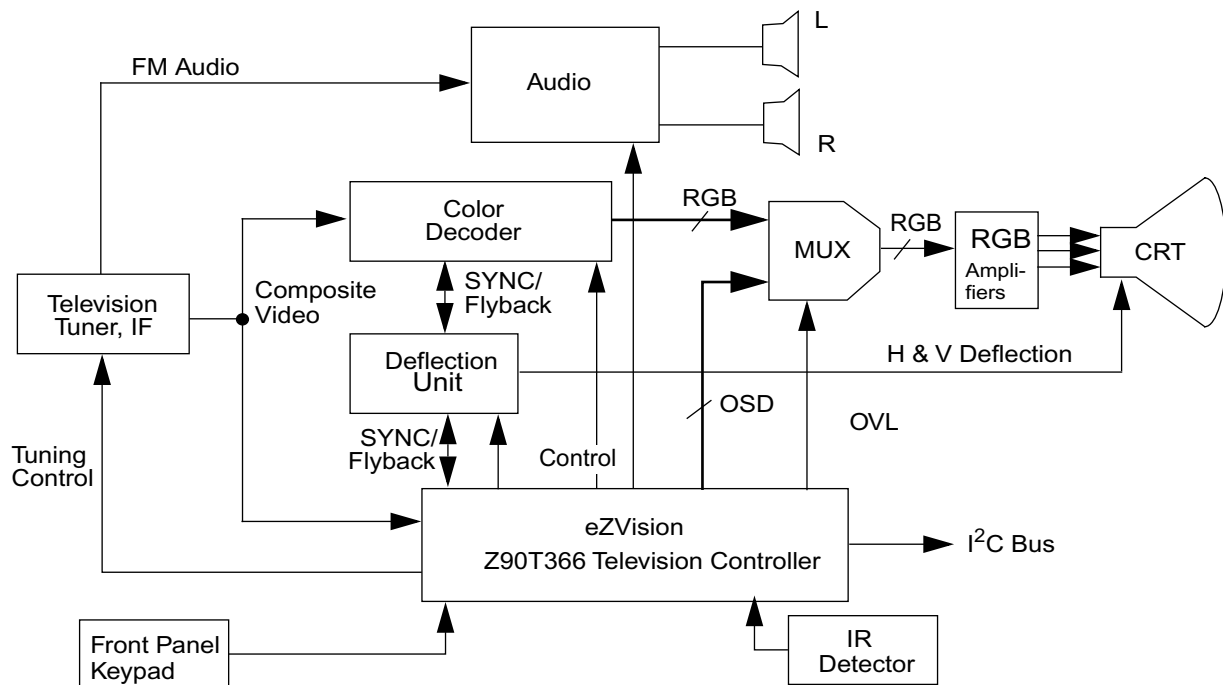


Figure 28 System Block Diagram

6 Electrical Characteristics

Table 76 Absolute Maximum and Minimum Ratings

| Sym | Parameter | Min | Max | Units | Conditions |
|----------|-----------------------|------|--------------|-------|-------------------------------|
| V_{CC} | Power supply voltage | 0 | 7 | V | |
| V_{ID} | Input voltage | -0.3 | $V_{CC}+0.3$ | V | Digital inputs |
| V_{IA} | Input voltage | -0.3 | $V_{CC}+0.3$ | V | Analog inputs (A/D0–A/D4) |
| V_O | Output voltage | -0.3 | $V_{CC}+0.3$ | V | All push-pull digital outputs |
| V_O | Output voltage | -0.3 | $V_{CC}+0.3$ | V | PWM outputs |
| I_{OH} | Output current high | | -10 | mA | one pin |
| I_{OH} | Output current high | | -100 | mA | all pins |
| I_{OL} | Output current low | | 20 | mA | one pin |
| I_{OL} | Output current low | | 200 | mA | all pins |
| T_A | Operating temperature | 0 | 70 | °C | |
| T_S | Storage temperature | -65 | 150 | °C | |

Table 77 DC Characteristics

| Symbol | Parameter | Min | Typ | Max | Units | Conditions |
|-----------|---------------------------------|---------------|------|--------------|-------|-------------------------------|
| V_{CC} | Power supply voltage | 4.75 | 5.00 | 5.25 | V | |
| V_{IL} | Input voltage low | 0 | 0.4 | $0.2 V_{CC}$ | V | |
| V_{IH} | Input voltage high | $0.7V_{CC}$ | 3.6 | V_{CC} | V | |
| V_{IHR} | Input voltage High on Reset pin | $0.75 V_{CC}$ | 4.2 | V_{CC} | V | |
| V_{PU} | Maximum pull-up voltage | | | V_{CC} | V | For PWM |
| V_{OL} | Output voltage low | | 0.16 | 0.4 | V | @ $I_{OL} = 1 \text{ mA}$ |
| V_{OH} | Output voltage high | $V_{CC}-0.4$ | 4.75 | | V | @ $I_{OH} = -0.75 \text{ mA}$ |
| I_{OL} | Output current low | 7.2 | 12 | | mA | @ $V_{OL} = 0.4\text{v}$ |
| I_{OL1} | Output current low | 12 | 20 | | mA | @ $V_{OL} = 0.8\text{v}$ |

Table 77 DC Characteristics (Continued)

| Symbol | Parameter | Min | Typ | Max | Units | Conditions |
|-----------|--------------------------|--------------|------|--------------|---------|---|
| I_{OH} | Output current high | 4.0 | 7.0 | | mA | @ $V_{OH} = V_{CC} - 0.4$ |
| I_{OH1} | Output current high | 16 | 20 | | mA | @ $V_{OH} = 2.4v$ |
| V_{XL} | Input voltage XTAL1 low | | | $0.3 V_{CC}$ | V | External clock generator driven |
| V_{XH} | Input voltage XTAL1 high | $0.6 V_{CC}$ | | | V | |
| I_{IR} | Reset input current | 25 | 90 | 150 | μA | $V_{RL} = 0V$ |
| I_{IL} | Input leakage | -3.0 | 0.01 | 3.0 | μA | @ 0V and V_{CC} |
| I_{CC} | Supply current | | 60 | 100 | mA | All ports are inputs, RGB is in digital mode |
| I_{CC} | Supply current | | 30 | 50 | mA | All ports are inputs, RGB is in analog mode |
| I_{CC1} | Supply current | | 5 | 10 | mA | Sleep mode @ 32.768 KHz |
| I_{CC2} | Supply current | | 50 | 100 | μA | Stop mode all PWM outputs are @ $V_{IN} = 0V$ |

6.1 DC Peripherals

Table 78 R, G, and B Analog Output

| | Output Voltage (30 K Ω Load) | | | Settling Time |
|-----------|-------------------------------------|--------------------|--------------------|----------------------------|
| | @ $V_{CC} = 4.75V$ | @ $V_{CC} = 5.00V$ | @ $V_{CC} = 5.25V$ | 70% of DC Level, 10pF Load |
| data = 00 | 0.00V–0.65V | 0.00V–0.70V | 0.00V–0.75V | <50ns |
| data = 01 | 1.70V–0.20V | 1.80V–0.20V | 1.90V–0.20V | |
| data = 10 | 2.80V–0.25V | 2.90V–0.25V | 3.00V–0.25V | |
| data = 11 | 3.90V–0.30V | 4.00V–0.30V | 4.10V–0.30V | |

Table 79 ADC0/Small Range*

| Sym | Parameter | Min | Typ | Max | Units |
|------------------|---------------------------|-------------------------------|-------------------------------|-------------------------------|-------|
| U_{R-} | Clamping voltage at ADC0 | 1.0 | 1.5 | 2.0 | V |
| ADC ₀ | Input voltage for level 0 | $U_{R-}-(U_{R+}+U_{R-})/15$ | U_{R-} | $U_{R-}+(U_{R+}-U_{R-})/15$ | V |
| ADC ₁ | Input voltage for level 1 | U_{R-} | $U_{R-}+(U_{R+}-U_{R-})/15$ | $U_{R-}+2(U_{R+}-U_{R-})/15$ | V |
| ADC ₂ | Input voltage for level 2 | $U_{R-}+(U_{R+}-U_{R-})/15$ | $U_{R-}+2(U_{R+}-U_{R-})/15$ | $U_{R-}+3(U_{R+}-U_{R-})/15$ | V |
| ADC ₃ | Input voltage for level 3 | $U_{R-}+2(U_{R+}-U_{R-})/15$ | $U_{R-}+3(U_{R+}-U_{R-})/15$ | $U_{R-}+4(U_{R+}-U_{R-})/15$ | V |
| ADC ₄ | Input voltage for level 4 | $U_{R-}+3(U_{R+}-U_{R-})/15$ | $U_{R-}+4(U_{R+}-U_{R-})/15$ | $U_{R-}+5(U_{R+}-U_{R-})/15$ | V |
| ADC ₅ | Input voltage for level 5 | $U_{R-}+4(U_{R+}-U_{R-})/15$ | $U_{R-}+5(U_{R+}-U_{R-})/15$ | $U_{R-}+6(U_{R+}-U_{R-})/15$ | V |
| ADC ₆ | Input voltage for level 6 | $U_{R-}+5(U_{R+}-U_{R-})/15$ | $U_{R-}+6(U_{R+}-U_{R-})/15$ | $U_{R-}+7(U_{R+}-U_{R-})/15$ | V |
| ADC ₇ | Input voltage for level 7 | $U_{R-}+6(U_{R+}-U_{R-})/15$ | $U_{R-}+7(U_{R+}-U_{R-})/15$ | $U_{R-}+8(U_{R+}-U_{R-})/15$ | V |
| ADC ₈ | Input voltage for level 8 | $U_{R-}+7(U_{R+}-U_{R-})/15$ | $U_{R-}+8(U_{R+}-U_{R-})/15$ | $U_{R-}+9(U_{R+}-U_{R-})/15$ | V |
| ADC ₉ | Input voltage for level 9 | $U_{R-}+8(U_{R+}-U_{R-})/15$ | $U_{R-}+9(U_{R+}-U_{R-})/15$ | $U_{R-}+10(U_{R+}-U_{R-})/15$ | V |
| ADC _A | Input voltage for level A | $U_{R-}+9(U_{R+}-U_{R-})/15$ | $U_{R-}+10(U_{R+}-U_{R-})/15$ | $U_{R-}+11(U_{R+}-U_{R-})/15$ | V |
| ADC _B | Input voltage for level B | $U_{R-}+10(U_{R+}-U_{R-})/15$ | $U_{R-}+11(U_{R+}-U_{R-})/15$ | $U_{R-}+12(U_{R+}-U_{R-})/15$ | V |
| ADC _C | Input voltage for level C | $U_{R-}+11(U_{R+}-U_{R-})/15$ | $U_{R-}+12(U_{R+}-U_{R-})/15$ | $U_{R-}+13(U_{R+}-U_{R-})/15$ | V |
| ADC _D | Input voltage for level D | $U_{R-}+12(U_{R+}-U_{R-})/15$ | $U_{R-}+13(U_{R+}-U_{R-})/15$ | $U_{R-}+14(U_{R+}-U_{R-})/15$ | V |
| ADC _E | Input voltage for level E | $U_{R-}+13(U_{R+}-U_{R-})/15$ | $U_{R-}+14(U_{R+}-U_{R-})/15$ | $U_{R-}+15(U_{R+}-U_{R-})/15$ | V |
| ADC _F | Input voltage for level F | $U_{R-}+14(U_{R+}-U_{R-})/15$ | U_{R+} | $U_{R-}+16(U_{R+}-U_{R-})/15$ | V |
| R _{IN} | Input impedance | 1.0 | | | MΩ |

Note: * The Input voltage level indicated in the table is a switch point from one ADC level to another. To be in the middle of the level half, LSB $((U_{R+}-U_{R-})/(15*2))$ must be added. The Input voltage must be prorated accordingly if V_{CC} is changed. 1.5–2.0V; TA = 0°C to 70°C; VCC = 4.75V to 5.25V

Table 80 ADC1-ADC4/Full range

| Data from ADC | Input voltage (volts) | Data from ADC | Input voltage (volts) | Data from ADC | Input voltage (volts) | Data from ADC | Input voltage (volts) |
|--------------------------------------|-----------------------|---------------|-----------------------|---------------|-----------------------|---------------|-----------------------|
| 0000 | 0.000 ± 0.15 | 0100 | 1.250 ± 0.15 | 1000 | 2.500 ± 0.15 | 1100 | 3.750 ± 0.15 |
| 0001 | 0.312 ± 0.15 | 0101 | 1.563 ± 0.15 | 1001 | 2.813 ± 0.15 | 1101 | 4.063 ± 0.15 |
| 0010 | 0.625 ± 0.15 | 0110 | 1.875 ± 0.15 | 1010 | 3.125 ± 0.15 | 1110 | 4.375 ± 0.15 |
| 0011 | 0.934 ± 0.15 | 0111 | 2.188 ± 0.15 | 1011 | 3.438 ± 0.15 | 1111 | 4.688 ± 0.15 |
| Input impedance | | > 1.0 MΩ | | | | | |
| Note: 0-5.0V; Vcc = 5.0V; Temp=0-70C | | | | | | | |

6.2 AC Characteristics

Table 81 lists the AC characteristics.

Table 81 AC Characteristics

| Sym | Parameter | Min | Typ | Max | Units |
|-----------------------------------|--|-----|-----|-------|-----------|
| T _{PC} | Input clock period | 16 | 32 | 100 | μ sec |
| T _{RC} , T _{FC} | Clock input Rise and Fall | | 12 | | nsec |
| TD _{POR} | Power on reset delay | 0.8 | 1.2 | | sec |
| TW _{RES} | Power on reset minimum width | | | 5 TPC | μ sec |
| TD _{HS} | H _{SYNC} incoming signal width | 1 | 10 | 15 | μ sec |
| TD _{VS} | V _{SYNC} incoming signal width | 1 | 200 | 10000 | μ sec |
| TD _{ES} | Time delay between leading edge of V _{SYNC} and H _{SYNC} in EVEN field | -12 | 0 | +12 | μ sec |
| TD _{OS} | Time delay between leading edge of V _{SYNC} and H _{SYNC} in ODD field | 20 | 32 | 44 | μ sec |
| TW _{HVS} | H _{SYNC} /V _{SYNC} edge width | | 0.5 | 2.0 | μ sec |

Note: *All timing of the I²C bus interface are defined by related specifications of the I²C bus interface.
Note: T_A = 0°C to 70°C; V_{CC}=4.75V to 5.25V; F_{OSC}=32,768Hz

6.3 ANALOG RGB

The RGB outputs in analog mode are controlled current sources with an internal load. These outputs display gamma-corrected, V_{CC} prorated characteristics. See Table 82, Table 83, and Figure 29.

Table 82 RGB Voltage Specification

| Parameter | Min | Typical | Max | Units |
|--------------------|--------|-------------------------|--------|-------|
| Supply voltage | 4.5 | 5.0 | 5.5 | V |
| Full scale voltage | 1.8V | 1.9V (0.40 * V_{CC}) | 2.2V | V |
| 2/3 scale voltage | 1.5V | 1.6V (0.33 * V_{CC}) | 1.815V | V |
| 1/3 scale voltage | 1.125V | 1.2V (0.25 * V_{CC}) | 1.375V | V |
| Zero scale voltage | — | 0.0V (0.00 * V_{CC}) | +0.1 V | V |

Note: *Measured with 3.9 k Ω load.

Table 83 RGB Time Specification

| Parameter | Min | Typical | Max | Units |
|------------------|-----|---------|-----|-------|
| Output rise time | | 50 | 65 | ns |
| Output fall time | | 50 | 65 | ns |

Note: *Measured with 4 k Ω resistor in parallel with 30 pF capacitor load.

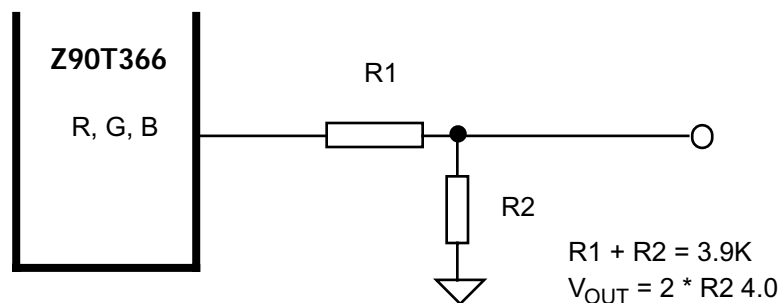


Figure 29 Recommended Application Schematics

7 Packaging

Figure 30 and Table 84 indicate the controlling dimensions.

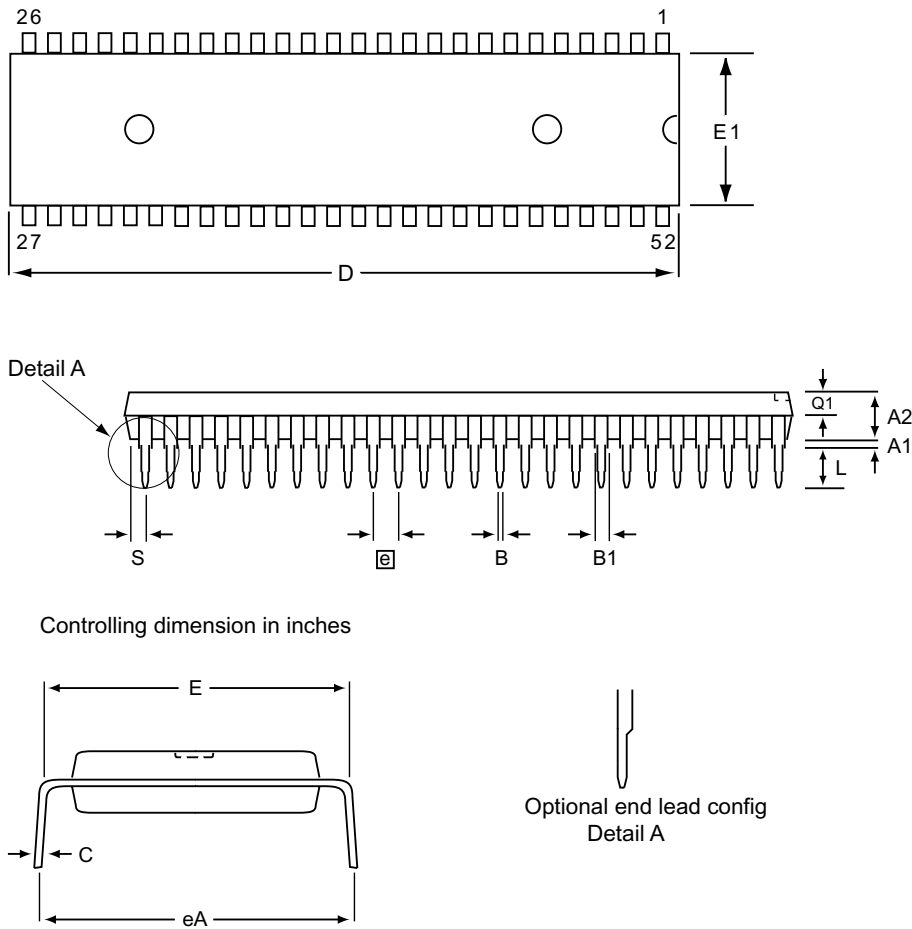



Figure 30 52-Pin SDIP Package Dimensions

Table 84 Controlling Dimensions

| Symbol | Millimeter | | Inch | |
|--|------------|-------|-----------|-------|
| | Min | Max | Min | Max |
| A1 | 0.51 | — | 0.020 | — |
| A2 | 3.25 | 3.94 | 0.128 | 0.155 |
| B | 0.38 | 0.53 | 0.015 | 0.021 |
| B1 | 0.89 | 1.14 | 0.035 | 0.045 |
| C | 0.23 | 0.38 | 0.009 | 0.015 |
| D | — | 47.50 | — | 1.870 |
| E | 15.24 | 15.75 | 0.600 | 0.620 |
| E1 | 13.72 | 14.10 | 0.540 | 0.555 |
|  | 1.778 TYP | | 0.070 TYP | |
| eA | 15.49 | 16.76 | 0.610 | 0.660 |
| L | 3.05 | 3.68 | 0.120 | 0.145 |
| Q1 | 1.52 | 1.91 | 0.060 | 0.075 |
| S | 0.64 | 1.78 | 0.025 | 0.070 |

8 Ordering Information

| Device | Order number | Description |
|--|-----------------------|--|
| Z90T361 64 KWord OTP | Z90T361PZ016SC | 16 MHZ, 64 KWord OTP Television Controller with OSD |
| Z90T366 64 KWord ROM | Z90T366PZ016SC Rxxxx* | 16 MHZ, 64 KWord Mask ROM Television Controller with OSD |
| Emulation Kit | Z9036900ZEM | Emulator/ Programmer |
| OTP Adapter | Z903xx00100ZDP | OTP programming adapter |
| Evaluation Board | Z9034600ZCO | OSD Evaluation Board |
| Note: * xxxx is a unique ROM number assigned to each customer code | | |

ROM Code Submission

ROM Code Submission Instructions

ROM Code can be submitted on ZiLOG's web site at <http://www.zilog.com>.

Top Mark Information

Mark Permanency: 3X soak into Alpha 2110 at 63° to 70°C, for 30 seconds duration each soak. Mechanical brush after each soak.



Customer Feedback Form

eZVision Z90T366 Product Specification

If there are any problems while operating this product, or any inaccuracies in the specification, please copy and complete this form, then mail or fax it to ZiLOG. Suggestions welcome!

Customer Information

| | |
|----------------|---------|
| Name | Country |
| Company | Phone |
| Address | Fax |
| City/State/Zip | E-Mail |

Product Information

| |
|--------------------------------|
| Serial # or Board Fab #/Rev. # |
| Software Version |
| Document Number |
| Host Computer Description/Type |

Return Information

ZiLOG
System Test/Customer Support
910 E. Hamilton Avenue, Suite 110, MS 4-3
Campbell, CA 95008
Fax: (408) 558-8536 Email: tools@zillog.com

Problem Description or Suggestion

Provide a complete description of the problem or suggestion. For specific problems, include all steps leading up to the occurrence of the problem. Attach additional pages as necessary.

| |
|--|
| |
| |
| |