Image Compression Coprocessor (ICC)

The KS0143 (ICC : Image Compression Coprocessor) integrated circuit is a very high performance programmable processor optimized for the execution of DCT-based image compression algorithms such as MPEG-1, JPEG, and H.261. In combination with commonly available RISC or DSP processors, the KS0143 and its companion chip, the KS0144 (MEC : Motion Estimation Coprocessor), form a complete, highly programmable computing solution for compressing and decompressing both still and motion video images.

RELATED PRODUCTS

- KS0144 Motion Estimation Coprocessor (MEC)
- VFEB® Hardware Evaluation Kit
- Aspen / Keystone Reference Board
- ICCST® Software Tool Kit

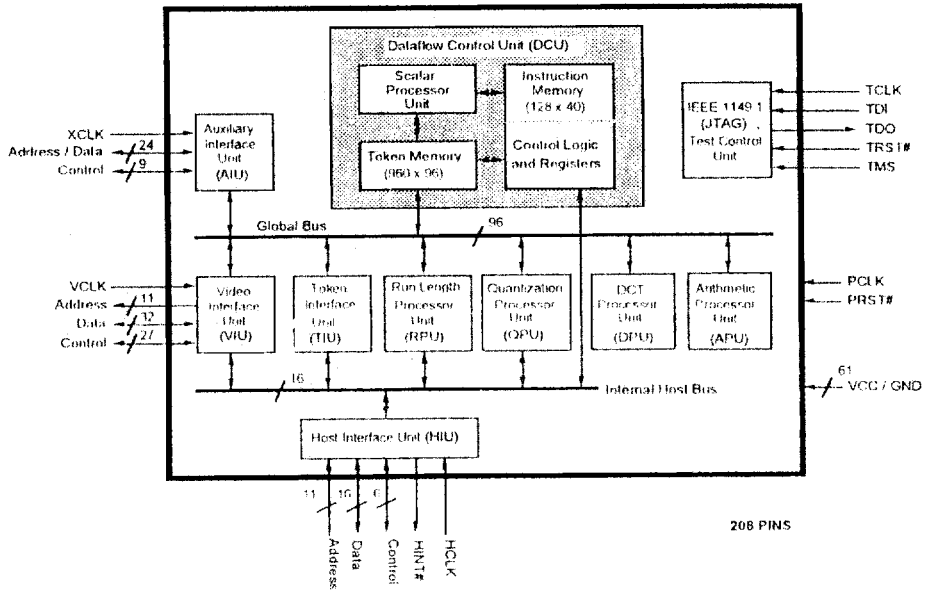
APPLICATIONS

- Video Conferencing and Telephony
- Video Editing
- Video CD-ROM Mastering
- Image Storage and Retrieval
- Multimedia-based Systems

FEATURES

- Compliant with MPEG-1 (ISO/ IEC International Standard 11172-2), JPEG (ISO/ IEC International Standard 10918-1), and ITU-T Recommendation H.261.
- Graph-based dataflow programming using high-level instructions downloadable to on-chip RAM.
- Multiple internal processing units optimized for very high performance execution of widely used image compression operations.
- Memory-mapped host processor interface including 16-bit data bus, 11-bit address bus, and support for high performance burst transfers.
- High speed 32-bit video data bus for accessing multiple video frame memories required during the execution of compression algorithms.
- Auxiliary interface bus permitting the connection of user-definable external processing chips to the KS0143.
- 50 MHz clock, 6W maximum power dissipation.
- 208-pin Metal Quad Flat Pack (MQUAD™), cavity down package.

FUNCTIONAL BLOCK DIAGRAM



SAMSUNG ELECTRONICS

INTRODUCTION TO THE KS0143 / KS0144 CHIP SET

The KS0143 Image Compression Coprocessor (ICC), together with its companion chip, the KS0144 Motion Estimation Coprocessor (MEC), form the heart of a programmable video codec for implementing the JPEG, MPEG-1, and H.261 compression standards. The codec architecture shown in Figure 1.1 may be easily programmed to implement any of the following applications at 30 frames per second:

- H.261 codec processing CIF (352h x 288v) images
- MPEG-1 SIF (352h x 240v) encoder with up to two B frames per Prame and half-pel motion estimation
- MPEG-1 SIF decoder
- JPEG CCIR-601 encoder or decoder

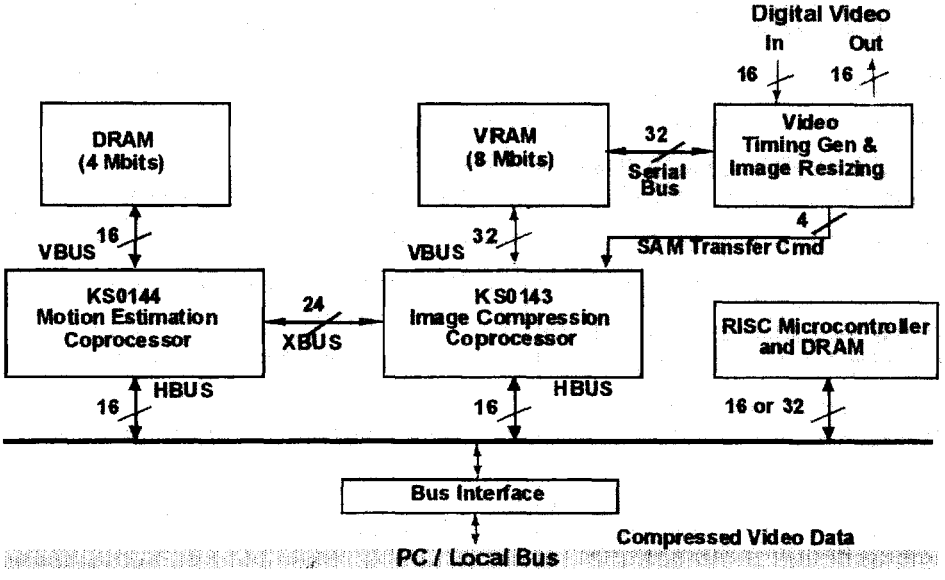
The KS0143 implements all portions of these standards other than motion estimation, variable length coding (VLC), and bit stream syntax handling. The KS0144 executes user-defined motion estimation search algorithms and is only required in systems implementing MPEG-1 or H.261 motion-predictive encoders.

VLC and bit stream syntax handling are typically performed using an off-the-shelf RISC microcontroller which also controls both the KS0143 and KS0144 via their host processor interfaces. The KS0143 / KS0144 chip set does not perform color space conversion or image resizing

VIDEOFLOW ® Architecture

The KS0143 / KS0144 chip set is based on a very high performance parallel processing architecture. Its parallel processing KS0144hanism has been implemented using a unique vector dataflow architecture which delivers exceptionally high performance and decouples the user from the complexities of a parallel processor. This vector dataflow (VIDEOFLOW) architecture combines dataflow control with vector instruction execution, allowing a user to control a very complex parallel processor via simple program flowgraphs. In addition, the VIDEOFLOW architecture is highly scalable and modular, allowing Array Microsystems to readily produce architecturally compatible derivative products.

Figure 1.1 - Typical Application
Programmable video codec for JPEG, MPEG-1, and H.261-based multimedia applications



Samsung Electronics Co., LTD.
Sa #14, Nongseo-Ri, Kihung-Eup, Youngin-Kun,
Kyungki-D, Korea 449-900
Suwon P.O. Box #105

Tel : +82 (331) 280-9454, 280-9484
FAX : =84 (331) 280-9459
Email : sspds @ jupiter. info. samsung. co.kr

REVISION HISTORY	
Revision Number	Release Date
SMP9601DS V1.0	7/93
SMP9601DS V1.1	7/93
SMP9601DS V1.2	1/95
KS0143DS V1.3	4/96

Product Ordering Information :

Part Number	Speed	Processing	Temperature Range (case)	Voltage Range	Pin Count	Package Type
KS0143-50	50 MHz	Commercial	0 × to 100 × C	4.75 to 5.25V	208	MQUAD™
KS0143-40	40 MHz	Commercial	0 × to 100 × C	4.75 to 5.25V	208	MQUAD™

Trademark acknowledgments :

KS0143/KS0144 is a trademark of Samsung Electronics Co.,LTD
VIDEO FLOW is a trademark of Array Microsystems Inc.
Intel and 1960 are trademarks of intel Corporation
29KTM is trademark of Advanced Micro Devices, Inc

WARNING - LIFE SUPPORT APPLICATIONS POLCY

Samsung Electronics products should notbge used within Life Support Systems without specific written consent of Samsung
Electronics Co., LTD. A Life Support System is a product or system intended to support or sustain life which, if it fails, can
be reasonably expected to result in significant personal injury or death.

NOTICE - DOCUMENTATION SUPPORT POLICY

The information in this publication is subject to change without notice. Samsung assumes no responsibility for any errors
or omissions, and disclaims responsibility for any consequences rsulting from the use of the information included herein.
Additionally, Samsung assumes no responsibility for the functioning of undescribed features or parameters.

NOTICE - APPLICATIONS SUPPORT POLOCY

Samsung Electronics Co., LTD . assumes no liability for applications assistance or customer product design.

NOTICE - PRODUCT AVAILABILITY

Samsung Electronics Co., LTD. reserves the right to make changes to product specifications or to discontinue products without notice.

NOTICE - STATED OR IMPLIED WARRANTY

This publication neither states nor implies any warranty of any kind, including byt not limited toimplied warranties of *merchantability or fitness for a particular application.*

NOTICE - COPY PERMISSION

Permission is hereby expressly granted to copy this publication for informatinal purposes only.
Copying this material for any other use is strictly prohibited.

KS0143 / KS0144 Chip Set Performance

The parallel processing architectures embodied in the KS0143 / KS0144 chip set permit it to deliver exceptional performance for the JPEG, MPEG-1, and H.261 standards. Table 1.1 shows representative real-time (i.e. 30 frames per second) applications of the standards using the 40 and 50 MHz versions of the chip set. The KS0144 is not required for H.261 decoder, MPEG-1 decoder, or JPEG applications.

A unique feature of the chip set, owing to the programming style and very high performance of the KS0143, is its ability to simultaneously process multiple image channels. For example, for multipoint video conferencing, a single KS0143/KS0144 chip set may be used to implement an H.261 video codec capable of simultaneously encoding a single image sequence (e.g. the one being transmitted) and decoding multiple received bit streams. Also, for MPEG-1 multimedia applications running in a windowed environment, a single KS0143 can decode two different SIF image bit streams in real time.

As shown in Table 1.1, another unique feature of the chip set is its ability to decode MPEG-1 images as they are encoded. This Òreal-time previewÓ feature is the result of the high performance and programmability of the chip set. Real-time previewing allows the user of MPEG-1 multimedia applications to create bit streams and visually monitor encoding results without the need for additional hardware for MPEG-1 decoding.

Furthermore, the ability of the KS0143 chip to real-time JPEG encode or decode the large image sizes shown in Table 1.1 is extremely useful in video editing applications. These applications typically operate with very high resolution images and compress them prior to storing them on disk.

High Speed, Compatible Bus Interfaces

As shown in Figure 1.1, the KS0143 and KS0144 have three bus interfaces called HBUS, XBUS, and VBUS which are compatible between the two chips. Each bus interface operates synchronously with its own dedicated clock (HCLK, XCLK, and VCLK). A separate clock (PCLK) is used for internal data processing and has a maximum frequency of 40 or 50 MHz depending upon the speed grade.

The HBUS is a 16-bit, 33 MHz synchronous host processor interface optimized for connection to the buses of RISC microcontrollers from families such as the Intel i960® and Advanced Micro Devices 29K®. The KS0143 and KS0144 operate exclusively as slaves on the HBUS. The host processor (RISC microcontroller) accesses registers and memories within these chips via memory-mapped accesses.

Both the KS0143 and KS0144 contain on-chip program memories which are typically loaded by the RISC microcontroller over the HBUS. The KS0143 instruction RAM is 128 words by 40 bits and the KS0144 instruction RAM is 768 words by 32 bits.

The highest bandwidth utilization of the HBUS occurs during the transfer of zero-run length coded symbols between the KS0143 and host processor during real-time compression and decompression operations. The HBUS also provides a level-sensitive interrupt from the KS0143 or KS0144 to the host processor.

The XBUS is a 24-bit, 40 MHz synchronous auxiliary processor bus used primarily for vector data transfers between the KS0143 and KS0144. The user may also define KS0143 "external" instructions to direct transfers over the XBUS which may be connected to processors other than the KS0144.

Table 1.1-KS0143 / KS0144 Performance Benchmarks at 30 Frams Per Second

	KS0143CQ-50 KS0144CQ-50	KS0143CQ-40 KS0144CQ-40
H.261 Codec	CIF (352h × 288v) encode + 1CIF decode, or QCIF (176h × 144V) encode + 6 QCIF deocdes	QCIF encode + 1 CIF decode @ 15 fps, or QCIF encode + 6 QCIF decodes @ 30 fps
MPEG-1 Encoder	SIF (352h × 240V) IBP encode with real-time preview	SIF (352h × 240V) IBP encode, or SIF (320h × 240V) IBP encode with real-time preview
MPEG-1 Decoder	2 SIF (352h × 240V) IBP decodes at 30fps, or 1 SIF IBP decode @ 60fps	2 SIF (352 × 240V) IBP decodes @ 30 fps, or 1 SIF IBP decode @ 60 fps
JPEG Encode or Decoder	720h × 480V, 4:2:2 YCrCb	720h × 480V, 4:2:2 YCrCb

The KS0143 is always the master of the XBUS, and up to four slave processors may be connected to it. In particular, up to four KS0144 chips may be connected to a single KS0143 for the purpose of allowing more searches in a given search area or allowing a search area to be expanded beyond the capacity of a single KS0144. During motion estimation operations, the XBUS is used by the KS0143 to send 16x16 blocks of luminance pels to the KS0144 for prediction. The KS0144 in turn, uses the XBUS to return motion vectors and their corresponding 16x16 pel blocks found during motion estimation searches.

The VBUS is a multi-master, 32-bit memory bus designed to be directly connected to the data, address, and control pins of DRAMs and VRAMs. The timing characteristics of signals on the VBUS are determined by user-programmable registers within the KS0143 and KS0144. VBUS data transfers may be dynamically resized to 16 or 32-bits.

The VBUS data interface is divided into byte-wide lanes, each lane transferring one 8-bit pel. In 16-bit mode, the VBUS transfers two pels at a time and in 32-bit mode transfers four pels at a time. All transfers are performed as fast page mode accesses at a maximum rate of one column access every 50 nsec, allowing a 16 x 16 pel block to be read or written in about 4.5 msec. The VBUS has an 11-bit address bus and three bits for memory selection which can access up to eight different images up to 4096 by 4096 pels each. A programmable CAS-before-RAS refresh controller is also included in the VBUS interface logic of both the KS0143 and KS0144.

The VBUS on both the KS0143 and KS0144 may be awarded to other bus masters via serial daisy chained bus arbitration. For example, an external bus master may be used to transfer images into or out of DRAMs connected to the KS0143. If an application permits, the KS0143 and KS0144 VBUS interfaces may be connected together in order to share the same memories. In this case, either the KS0143 or KS0144 is programmed to be at the head of the arbitration daisy chain, and one chip requests the VBUS from the other.

A unique feature of the KS0143 VBUS interface logic allows the KS0143 to transfer data between the serial and dynamic memory portions of VRAMs. This allows the serial port on VRAMs rather than the random access port on DRAMs to be used for image input/output, making the interface with external logic simpler and faster. Internal VRAM data transfers may be requested by either a video timing generator which asserts the KS0143 Óvideo transfer commandÓ pins (VCMD[3:0]) or the host processor through a memory mapped register.

Input, output, and scratch images are stored in DRAM- or VRAM-based memories connected to the VBUS on the KS0143 and KS0144. The 4 Mbit memory shown in Figure 1.1 stores the reference images used during H.261 and MPEG-1 motion estimation searches and may be implemented using a single 256K by 16-bit DRAM. The 8 Mbit memory stores input and output images as well as H.261 and MPEG-1 decoder reference images. It may be implemented using two 256K by 16-bit or four 256K by 8-bit VRAMs depending on how image input/output is performed. It is also possible to use DRAMs in place of the VRAMs.

VIDEOFLOW Support Tools

Array Microsystems, Inc. supplies a Hardware Evaluation Kit and Software Tool Kit to help customers design their KS0143/KS0144-based systems.

VFEB Hardware Evaluation Kit provides a hardware vehicle for designers and application developers wishing to explore the real-time performance and multi-standard versatility of the KS0143 and KS0144 chips. It consists of a PC plug-in board for the ISA or VL bus, embedded software, a graphical user interface (GUI) that is compatible with the Windows™ operating system, and User Guide.

The KS0143/KS0144-based PC plug-in board and GUI allow the user to perform real-time (30 fps) demonstrations of the MPEG-1, JPEG, and H.261 standards for both video encoding and decoding. For encoding, users can connect their own analog video to the board and store compressed video to disk in real-time. Decoding demonstrations can read compressed video from either disk or CD-ROM and display the results on a video monitor.

The ICCST Software Tool Kit provides a UNIX workstation-based software development environment for designers wishing to architect, program, simulate, and verify video compression solutions based on the KS0143 and KS0144 chips. The tool kit contains:

- Programming tools consisting of a graphical flowgraph editor, assembler and linker for the KS0143, and a C-compiler and assembler for the KS0144
- Simulation tools consisting of an KS0143/KS0144 system simulator, configuration tools, program debugging aids, and various graphical displays for monitoring program performance
- Image analysis tools supporting the display of image sequences at or near real-time on a workstation monitor with optional graphical overlays of color-coded compression algorithm decisions and motion vectors produced by the simulator
- Comprehensive user and reference guide
- Demonstration package containing programs and simulation examples for the MPEG-1, JPEG, and H.261 video compression standards

Section 2 - KS0143 FUNCTIONAL DESCRIPTION

The functional description of the KS0143 that follows refers to the block diagram that appears on the front page. Architecturally, the KS0143 is a programmable parallel processor consisting of a central controller and seven processor/bus interface units interconnected by a 96-bit global bus. The controller, called the Dataflow Control Unit (DCU), contains a 128 word by 40-bit instruction RAM and a large token memory which is used to hold instruction execution results.

Each of the processor and bus interface units (generically called functional units), is optimized to execute a subset of the KS0143 instruction set. The DCU schedules instructions for execution at run-time on the appropriate functional units, sends instructions and operands to these units over the global bus, and receives instruction results for storage in the token memory. Allocation and deallocation of token memory storage is handled by the DCU as a background task and is not explicitly controlled by the programmer.

The DCU and all functional units are clocked by PCLK which has a maximum frequency of 50 MHz. The collective KS0143 functional units have a peak throughput of over 1 billion operations per second. The global bus is capable of transferring eight 12-bit words per PCLK cycle, yielding a maximum burst throughput of 400 Mwords (or 600 Mbytes) per second.

Dataflow Computing Model

The KS0143 uses the DCU to implement a dataflow computing model. Dataflow computers such as the KS0143, share the primary characteristic that they are data-driven rather than control-driven like common microprocessors. Dataflow computers do not use a program counter to determine instruction execution order. Dataflow computers execute or "fire" an instruction when the operands, the resources, and a place to put the results are available. In a dataflow computer which contains multiple processor units (like the KS0143), many instructions may meet these criteria at the same time and can be executed in parallel.

Programs for dataflow computers are represented using directed graphs consisting of instruction nodes interconnected by arcs representing the flow of data between instructions. The data objects which travel on these flowgraph arcs are called "tokens".

The KS0143 is a static dataflow computer which allows only a single token to exist on an arc at any one time. This means that an instruction cannot execute again until its result from a previous execution has been consumed by all of its destination instructions.

The flowgraph-based programming used in the KS0143 makes it very easy for the user to specify the parallel execution of instructions. No complicated compiler technology is required as with VLIW (very long instruction word) and SIMD (single instruction - multiple data) architectures. The KS0143 programmer simply specifies how data flows between instructions by interconnecting them using the graphical flowgraph editor/assembler supplied with the VIDEOFLOW Software Tools. The DCU does the rest by scheduling instruction executions at run-time.

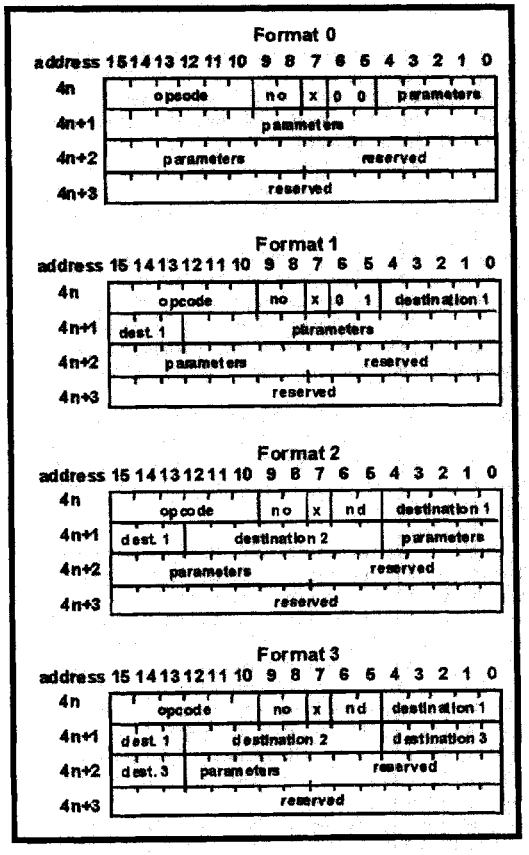
The KS0143 further simplifies programming by using high-level instructions to perform arithmetic operations such as DCT and quantization on multiple data blocks. This allows programs to be very compact and therefore easy to write and understand.

DCU Operation

The DCU continually scans the KS0143 instruction RAM for instructions which are ready for execution. It checks the ready status of each instruction by consulting bookkeeping memories which tell it when the instruction operands (if any) are present in the DCU token memory and the result token (if any) from a previous execution of the instruction has been consumed by all of its destinations. The DCU also checks whether the functional unit required to execute the instruction is idle. Once an instruction is ready, the DCU dispatches both it and its operands to the appropriate functional unit over the global bus or, in some cases, executes the instruction itself.

If an operand is not needed by any other instruction after it is transmitted, the storage it occupies in the token memory is deallocated so that it may be reused by result tokens. After a functional unit finishes executing an instruction, it requests service from the DCU. The DCU subsequently allocates space in the token memory for the functional unit result token (if any) and directs the functional unit to send the result to the token memory over the global bus.

Instruction Formats and Field Definitions



Field	Definitions
opcode	x=0 : Internal function code x=1 : opcode (5:2) is external function code opcode (1:0) aux. processor address
no	number of operands (0,1, or 2)
x	external instruction flag (1=external)
nd	actual number of destinations
destination	Instruction destination address bits 7:1 =Instruction address bit0=operand port (0=OP1, 1=OP2)
parameters	parameter field (defination determined by opcode)

Instruction Set

The KS0143 instruction set consists of 56 internal instructions plus 64 user-definable external instructions. The internal instructions are executed by the on-chip functional units or the DCU. The external instructions are executed by up to four off-chip (i.e. auxiliary) processors such as the KS0144. Each of these instructions processes zero, one, or two operand tokens and creates zero or one result tokens.

A user's KS0143 program consists of one or more flowgraphs which are loaded into the DCU instruction RAM by the host processor as part of system initialization.

Figure 2.1 shows that KS0143 instructions appear to the host as 64-bit words, but only the most significant 40 bits of each instruction are used. Each KS0143 instruction falls into one of four formats based on its maximum number of destinations.

The nd field specifies the actual number of destinations. For internal instructions (i.e. those for which x = 0), the opcode field specifies one of the KS0143 56 internal instruction types. For external instructions, opcode is undefined except for its two least significant bits which specify the address of the auxiliary processor required to execute the instruction. Parameter bits have definitions which may vary from instruction to instruction.

The KS0143 instruction set is specifically designed to handle the real-time algorithm requirements of the H.261, MPEG-1, and baseline JPEG video compression standards and is divided into the following functional categories:

- Arithmetic instructions perform operations such as addition, subtraction, forward and inverse DCT (discrete cosine ransform), and forward and inverse quantization.
- Logical instructions perform boolean operations.
- Host Interface instructions allow data in a variety of formats to be transferred between the KS0143 and the host processor via the HBUS.
- Descriptor Modification instructions allow descriptive information about a token to be altered.
- Dataflow Control instructions control the passing of tokens between instructions based on data or flag-dependent conditions.
- Video Memory instructions transfer tokens of image data between the KS0143 and video memories connected to the VBUS.
- External instructions execute on auxiliary processors (e.g. the KS0144) which are connected to the XBUS.

KS0143 internal instructions are listed in Tables 3.1 through 3.6 by functional category. The tables give the following information about each instruction:

- the instruction mnemonic,
- numerical operation code in hex (opcode),
- number of required operands (no),
- instruction format (fmt),
- the unit which executes the instruction.
- a brief operational description, and
- a list of the instruction parameters.

For maximum performance, each of the KS0143 functional units is optimized to execute a particular subset of instructions. Some units only communicate with the global bus. These units include the:

- DCT Processor Unit (DPU),
- Quantization Processor Unit (QPU), and
- Arithmetic Processor Unit (APU).

Other functional units are primarily responsible for interfacing the KS0143 to the VBUS, HBUS, and XBUS and may also perform a moderate amount of computation. These include the:

- Video Interface Unit (VIU),
- Token Interface Unit (TIU),
- Runength Processor Unit (RPU), and
- Auxiliary Interface Unit (AIU).

The Host Interface Unit (HIU), shown in the Functional Block Diagram, is not connected to the global bus but translates HBUS accesses by the host processor into accesses on an internal host bus which is connected to most of the functional units.

In addition to managing instruction executions, the DCU is capable of quickly modifying token descriptor contents. Note that the DCU itself is responsible for executing all of the instructions listed in Tables 3.2, 3.4, and 3.5. These operations are performed by the DCU Scalar Processor Unit shown in the Functional Block Diagram. While the DCU is executing an instruction, it cannot communicate with any of the functional units over the global bus.

All of the instruction mnemonics and parameter names that appear in Tables 3.1 - 3.6 are compatible with the VIDEOFLOW Software Tool Kit graphical flowgraph editor/assembler. The flowgraph editor allows the programmer to select KS0143 instructions from a menu, specify parameter values, and graphically interconnect different instructions to form flowgraphs. The numerical contents of all instruction fields are then automatically generated. Figure 2.2 shows a

Token Types

KS0143 instructions produce and consume two types of tokens: control tokens and data tokens. Both token types have a common data structure called the token descriptor. This is a single 96-bit word which for programming purposes, is organized as twelve 8-bit bytes having the fields shown in Figure 2.3. The most important field is the type field of byte 0 which identifies the token as a control or data token. Token descriptor contents are manipulated primarily using Descriptor Modification instructions (Table 3.4).

A control token consists of only a token descriptor. The data token consists of a token descriptor plus one to four data blocks. Each data block consists of 64 twelve-bit words which are generally interpreted by instructions as an 8x8, two's complement integer array. The number of data blocks making up a data token is indicated by the nblocks field within byte 0 of the token descriptor.

KS0143 instructions generally act on both token descriptors and data blocks. With data tokens, the information contained in the token descriptor relates to the token data blocks. For example, it may contain video memory coordinates, motion vectors associated with the data blocks, or other parameters to be applied to the data blocks during processing.

Both control and data tokens are stored in DCU token memory. Token data blocks are stored in the token memory in units called block allocation units (BAUs), each of which can hold up to two data blocks. A single BAU may not be split between two tokens, so each data token requires one or two full BAUs. The token memory can store a total of 52 BAUs which is sufficient for all JPEG, MPEG-1, and H.261 applications. The token memory also stores a total of 128 token descriptors (one for every instruction in the KS0143 instruction RAM). This is a sufficient capacity for all KS0143 programs, since each instruction in a flowgraph can produce only one result token.

Figure 2.2 - JPEG Decoder Flowgraph

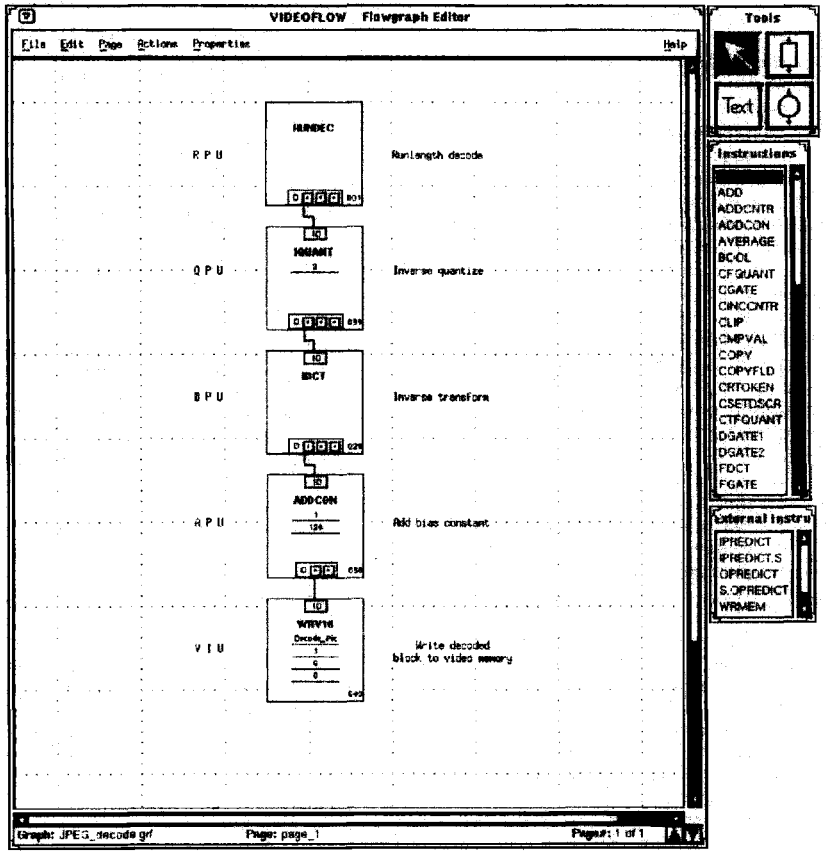
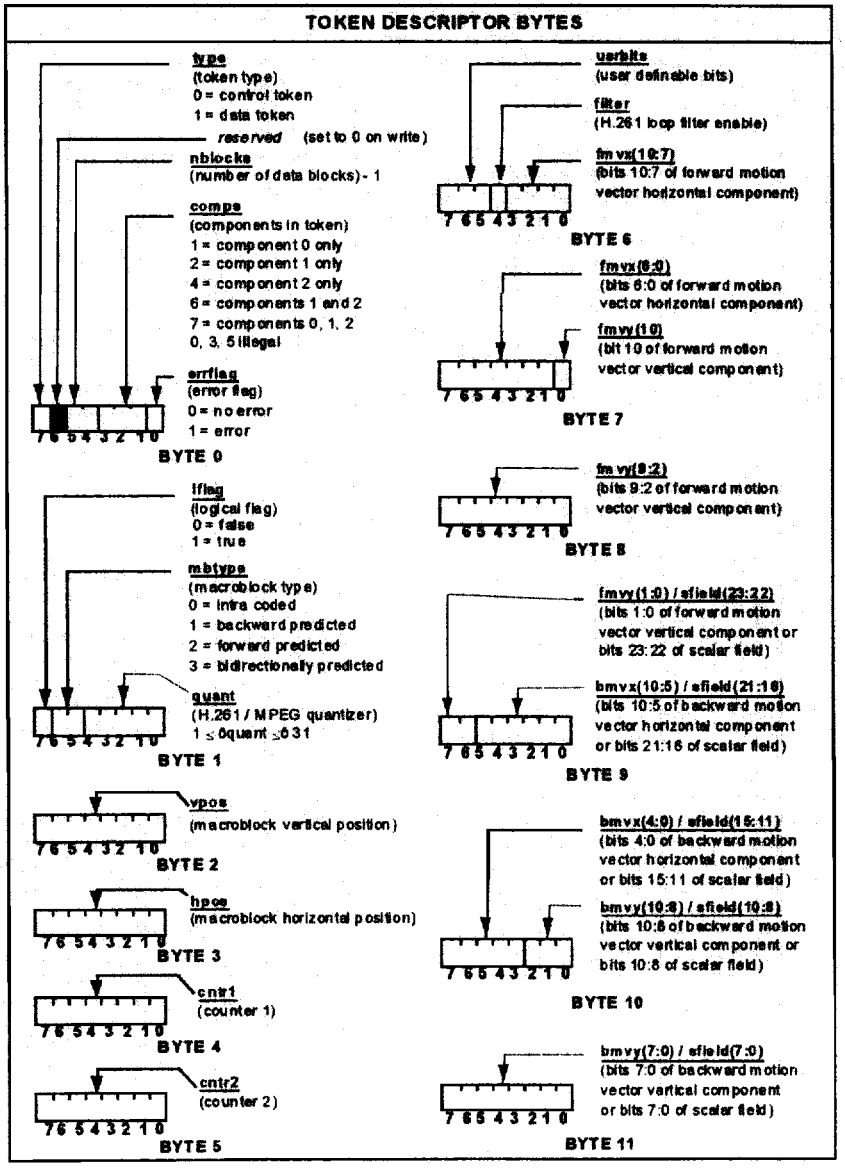


Figure 2.3 - Token Descriptor Programmer's Model



Estimating KS0143 Performance

The best way to determine program performance is to simulate it using the KS0143 / KS0144800 VIDEOFLOW Software Tool Kit. The real-time statistical displays of the simulation manager show functional unit and global bus utilization during program execution. The tool kit instruction trace capability provides a detailed timing profile of each instruction execution, allowing instruction parallelism bottlenecks to be readily identified.

Execution times for each of the 56 KS0143 internal instructions are listed in Table 3.7 in PCLKs. Two execution times are listed for each instruction (DCU Cycles and Unit Cycles). Both are functions of the number of data blocks making up instruction operand tokens. For non-DCU instructions, the clock cycles listed under DCU Cycles are used by the DCU to transfer tokens over the global bus to and from the functional units. Cycles listed under Unit Cycles are used by the functional units to both transfer and process tokens. Transfers over the global bus involving a given functional unit are not overlapped with data processing within that unit. This means Unit Cycles are the sum of DCU Cycles and token processing cycles. DCU instructions are executed entirely within the DCU, so their Unit Cycles are not applicable.

The execution time of an KS0143 program may be estimated in the manner described in the following sequence.

1. Determine the computational loading of each KS0143 functional unit by adding together the Unit Cycles of all the instructions in the program which are assigned to that unit. The time associated with each instruction should include all activities contributing to the execution time of that instruction, *e.g. video bus transfer time, host bus transfer time, amortized overhead for video memory refresh, etc.*
2. Estimate DCU loading by adding up the DCU Cycles across all instructions in the program.
3. Pick the maximum time across all functional units and the DCU. This is an ideal estimate of program execution time.

Note: The ideal estimate does not include additional delays due to factors like DCU instruction operand polling or lack of synchronization between the firings and durations of different instructions. A more realistic estimate can be obtained by dividing the ideal time by a derating factor of 0.75. Note that derating factors are highly program dependent, and 0.75 is a pessimistic value of evaluation of a wide range of programs by Array Microsystems.

Example Calculation:

Assume a program which contains FDCT, IDCT, FQUANT and DDCONT instructions.

	Unit Cycles	DCU Cycles
FDCT	(80n+37) × (20nsec)	(16n+4) × (20nsec)
IDCT	(80n+35) × (20nsec)	(16n+4) × (20nsec)
FQUANT	(80n+21) × (20nsec)	(16n+4) × (20nsec)
ADDCON	(48n+15) × (20nsec)	(16n+4) × (20nsec)

	DCU	DPU	QPU	APU
FDCT	1.36usec	7.14usec		
IDCT	1.36usec	7.10usec		
FQUANT	1.36usec		6.82usec	
ADDCON	1.36usec			4.14usec
Unit Loading	5.44usec	14.24usec	6.82usec	4.14usec

Assume PCLK = 50Mhz (20 nsec), n=4 (16 × 16 pel block)

The ideal execution time is 14.24 msec. A derated estimate is 18.99 msec (14.24/0.75)

Functional Unit Memory Maps

All functional units except the DPU, APU, and AIU have registers and/or memories which the host processor may access using the HBUS. These registers and memories are described in Section 4. All register and memory names used are compatible with the KS0143 / KS0144800 VIDEOFLOW Software Tool Kit. Note that all addresses shown in the memory maps included in Section 4 are in hexadecimal and all shaded areas should be set to zero when written and are undefined when read unless otherwise noted.

Video Interface Unit (VIU)

The VIU (Section 4.1) supports all VBUS operations such as fast page mode read and write, memory refresh, and SAM (serial access memory) transfers. Arbitration logic within the VIU selects which internal operation has access to the VBUS at any point in time.

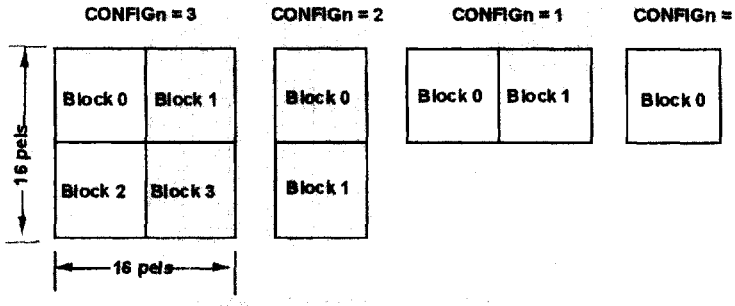
The VIU also supports reading blocks of image data which are offset by motion vectors stored in fmvx, fmvy, bmvx, and bmvy token descriptor fields (Figure 2.3). Motion vectors are interpreted as having full or half pel resolution depending on the state of the FULLFMV and FULLBMV registers in the DCU (Section 4.6). Full pel motion vectors follow the H.261 standard while half pel vectors follow the MPEG-1 standard.

The KS0143 programming model treats a logical image as a series of logical image components (such as RGB or YUV) each up to 4096 by 4096 pels in size. Logical images are partitioned into "macroblocks" consisting of spatially-related components. Each macroblock component consists of one, two, or four 8x8 pel blocks as defined by its CONFIG register in the DCU memory map (Section 4.6).

A macroblock position is derived from the vpos (vertical position) and hpos (horizontal position) token descriptor fields of the VIU instruction operand token (Figure 2.3). These coordinates are further modified by factors such as motion displacement, interleave, and interpolation. The VIU transacts macroblocks with the DCU in the form of one or more tokens depending on the CONFIG registers. The comps field in the data token descriptor together with the values of the CONFIG registers (Section 4.6) determine the number of 8x8 pel blocks contained in the data token.

Figures 2.5 and 2.6 show how macroblocks may be assembled from tokens. The example 4:2:0 MPEG/H.261 macroblock configuration (Figure 2.5) appears to the KS0143 program as two tokens. Token 0 contains the image luminance data (Logical Image Component 0; comps = 1) configured as four 8x8 pel blocks (CONFIG0 = 3). Token 1 contains the image chrominance data (Logical Image Components 1, 2; comps = 6) configured as single 8x8 pel blocks (CONFIG1 = 0, CONFIG2 = 0). A macroblock can also fit into one token as in the case of the CCIR-601 YUV 2:1:1 image (Figure 2.6).

Figure 2.4 - Component Block Configuration



Logical image components are physically distributed across two or four 8-bit wide memory banks connected to the VBUS. The number of banks depends on whether 16- or 32-bit access mode is selected. More than one memory bank may be stored on a single memory chip; for example, a 256K by 16-bit DRAM stores two 512 by 512 eight-bit banks. The VIU transacts image data with physical memory in the form of 2x1 or 2x2 pel tiles depending on whether 16-bit or 32-bit access mode is selected.

In 16-bit access mode (Figure 2.5), pels from even numbered lines of a logical image component are stored in one memory bank and all pels from odd numbered lines are stored in the other. The two memory banks may be connected to video data buses VD0 and VD1 (Low Bus mode), or VD2 and VD3 (High Bus mode). Tiles are organized as two rows by one column pel arrays in order to minimize RAS cycle time in DRAM memory systems. The example configuration in Figure 2.5 shows that an 8x8 pel block can be fetched using only four RAS cycles.

In 32-bit access mode (Figure 2.6), even numbered pels from even lines are stored in bank 0, even numbered pels from odd lines are stored in bank 1, odd numbered pels from even numbered lines are stored in bank 2, and odd numbered pels from odd numbered lines are stored in bank 3. Each of the four memory banks is connected to a different video data bus. Using 32-bit access mode, tiles are dimensioned as 2x2 pels. Again, fewer RAS cycles are required, but if a motion vector is involved, the number of extra pels that are fetched is minimized.

Figure2.5 - Example Macroblack 4;2;0 Configuratin Using 16-bit Bus Mode

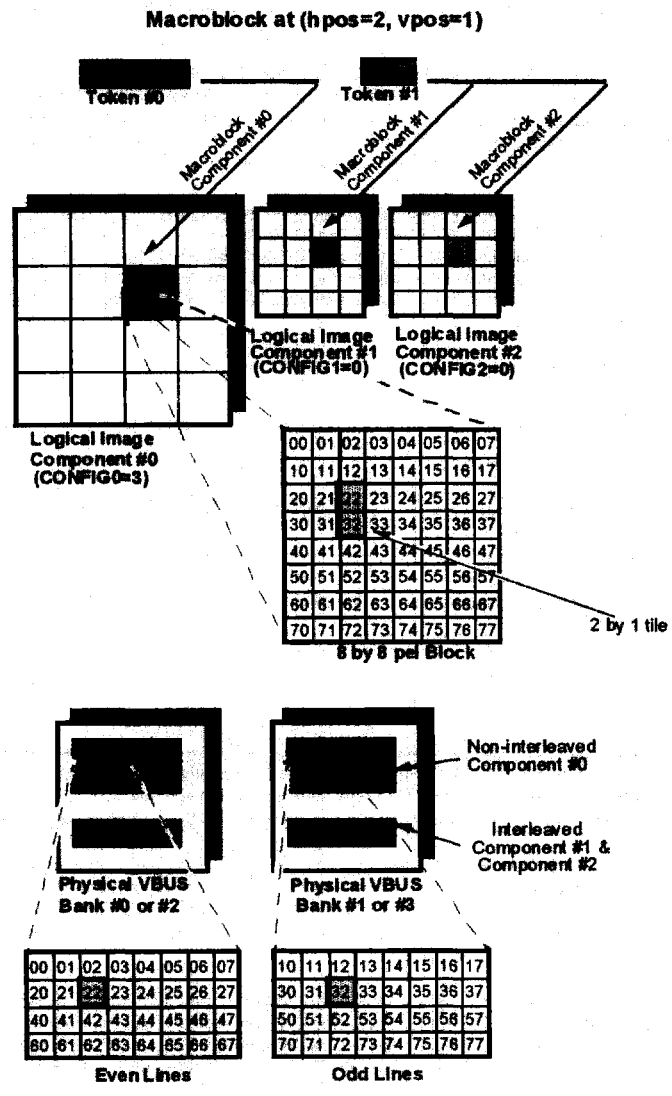
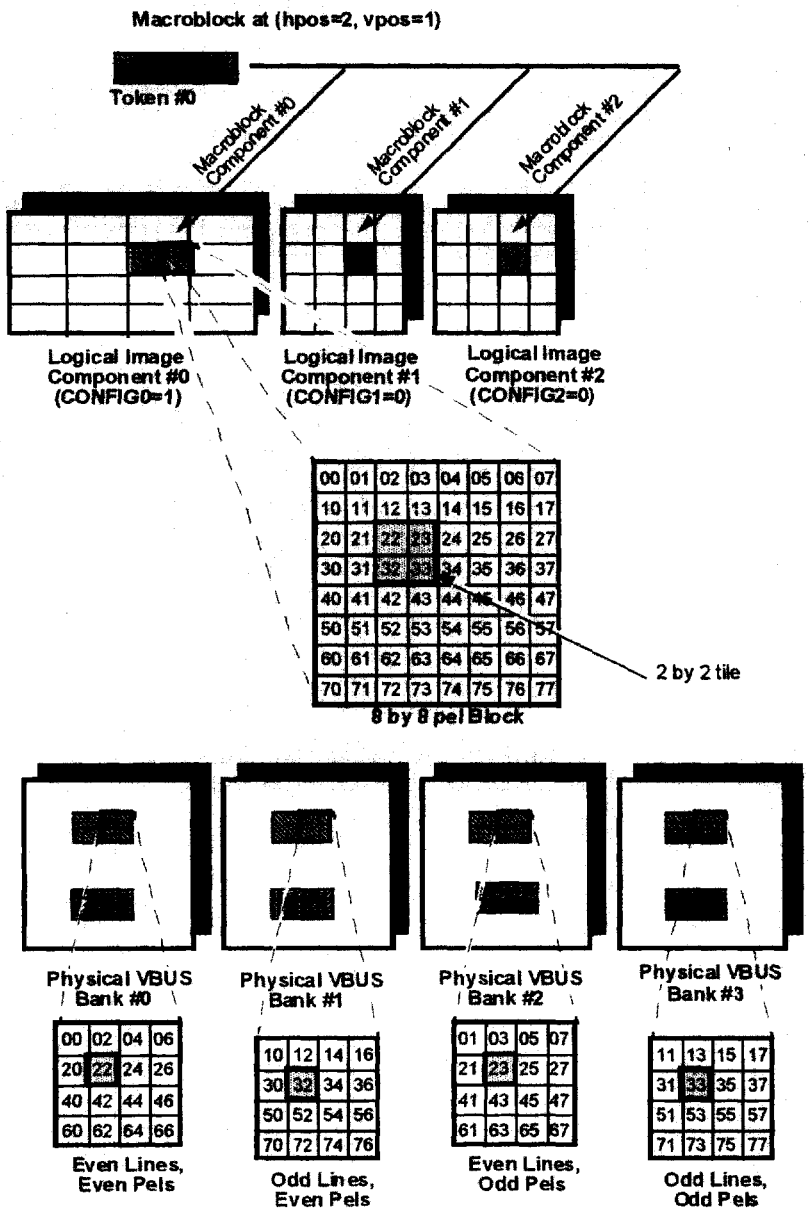


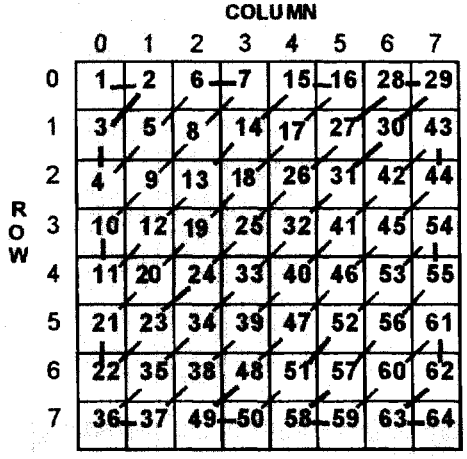
Figure 2.6 - Example Macroblock 2:1:1 Con figuration Using 32 bit Bus Mode



The example configuration in Figure 2.6 shows an 8x8 pel block motion offset by one row can be loaded using only five RAS cycles.
The VIU also supports features such as pel interleaving and line swapping which simplify the transfer of pels to and from external devices.

Run Length Processor Unit (RPU)
The RPU (Section 4.3) is responsible for converting data tokens between the internal format supported by all the functional units and the zero-run length coded format used for transferring compressed image data between the KS0143 and its host processor. The host processor then converts between the zero-run length coded format and the bit stream syntax defined by the particular compression standard being implemented.
Each 8x8 data block within a data token is converted to the zero-run length coded format by first linearly ordering its words according to the zigzag scheme shown in Figure 2.7. The result is then coded as a sequence of 16-bit words containing (zero-run, level) pairs or symbols where Òzero-runÓ is the number of consecutive zero values preceding the value ÒlevelÓ. The zigzag ordering convention conforms to the JPEG, MPEG-1, and H.261 standards.

Figure 2.7- Data Block Zigzag Ordering



SAMSUNG ELECTRONICS

Figure 2.8 shows the various elements of the zero-run length coded format. Figure 2.8(a) shows the basic structure of a token coded in this format. It begins with two to six header words which contain a subset of the fields in the token's corresponding token descriptor. The number of header words depends on the token descriptor mbtype field which indicates which motion vectors (if any) are associated with the token. Header word formats appear in Figures 2.8(c) through 2.8(f)
The header words are followed by nblocks+1 sequences of zero-run length coded symbols, i.e. one sequence for every data block in the token. As shown in Figure 2.8(b), each symbol consists of either one or two 16-bit words, depending on the level. Each sequence is terminated by an end-of-block marker word. The last end-of-block marker is followed by an end-of-token marker word.
The RUNENC and RUNENC.S instructions (Table 3.3) convert operand data tokens into zero-run length coded symbols and header words, while the RUNDEC instruction does the inverse. The symbols and headers reside in a 256 word by 16-bit bidirectional FIFO within the RPU. One end of the FIFO is accessed by the RPU while the other is accessed by the host processor via the HIU.
From the programmer's viewpoint, the FIFO is partitioned into two sections called INFIFO and OUTFIFO whose relative lengths may be configured by the host processor. The host always writes zero-run length coded symbols and header words to INFIFO and reads them from OUTFIFO. The access ports for both INFIFO and OUTFIFO appear in the RPU memory map as does a status register for informing the host of FIFO fullness. The RPU also contains logic for generating a level-sensitive interrupt to the host processor via the HINT# pin. The interrupt may be the result of the occurrence of one or more of 14 different interrupt conditions as defined by the INTSTAT register in the RPU memory map (Section 4.3).

Token Interface Unit (TIU)
The TIU (Section 4.5) contains a single 264 word by 12-bit memory called the Token Passing Buffer (TPB) which is used to pass tokens in their raw form between the KS0143 and the host processor. At any point in time the TPB may hold a single data or control token. The TIU writes the TPB with the operand token of a SNOOP instruction, and the host subsequently reads it in the format shown in the TIU memory map.

Figure 2.8 - Zero - Run Length Coded Data Formats

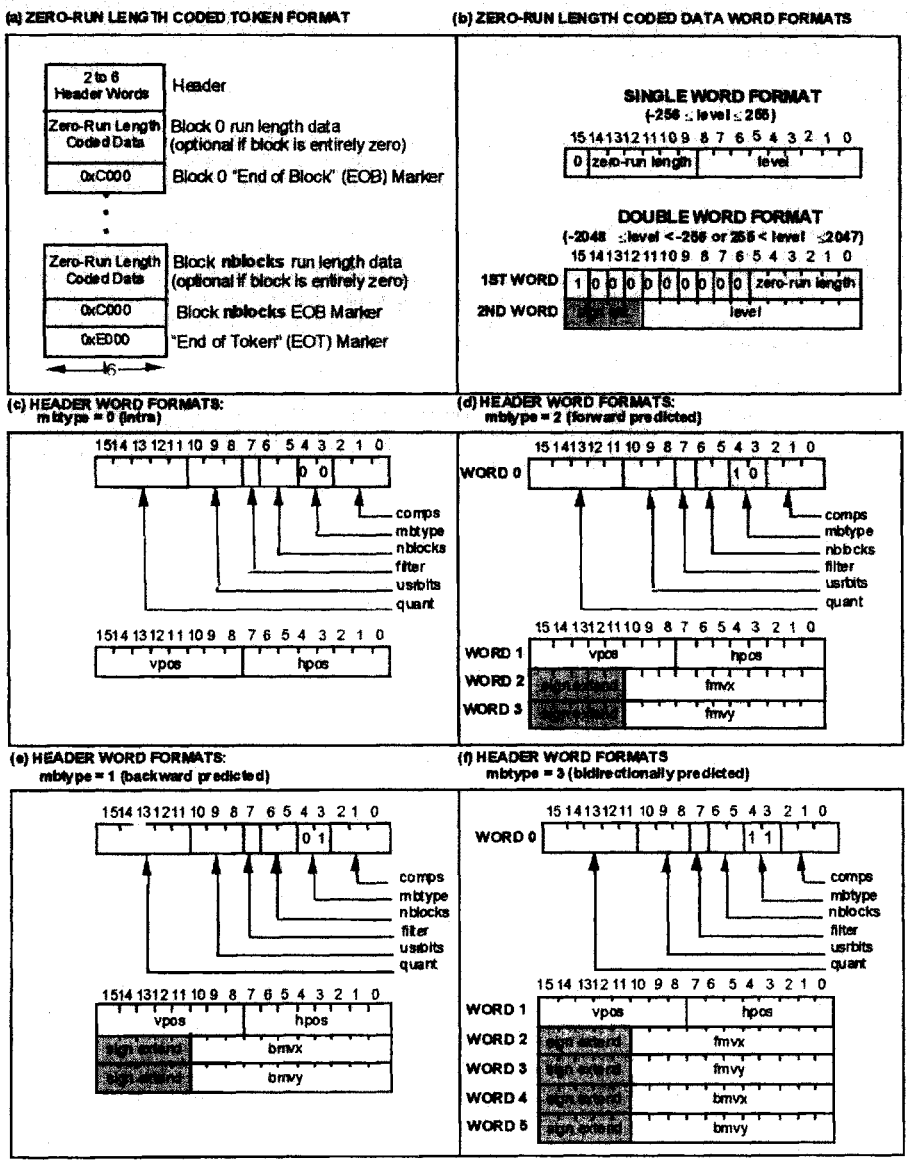
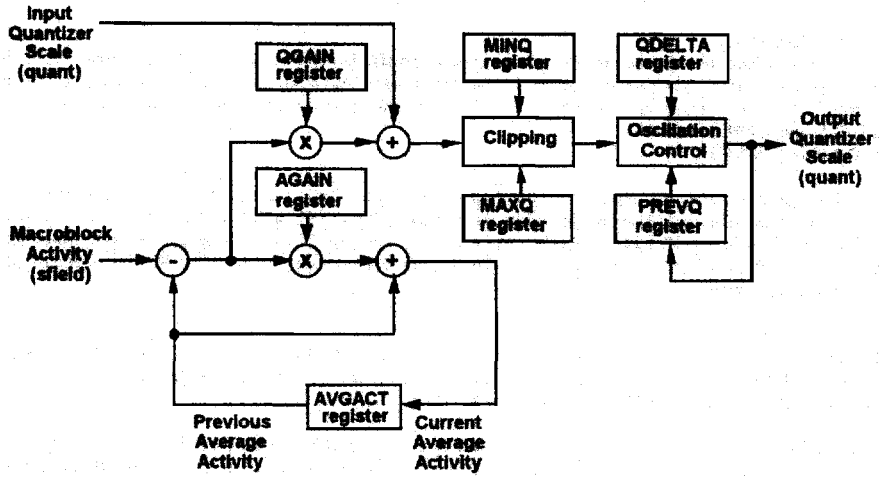


Figure 2.9 - ADPATQ Instruction Execution



With SNEAK instructions, the host writes a token into the TPB using the format shown in Section 4.5. As a result, the TIU writes the token back to the DCU over the global bus. The host controls the TPB via the TPBSEL, TPBFULL, and TPBADR registers which reside in the DCU memory map (Section 4.6). An interrupt condition is also defined that notifies the host when the TPB is empty or full (see INTSTAT; Section 4.3).

The programmer may regard SNEAK and SNOOP instructions as software probes allowing a token to be injected or sampled at any point in the KS0143 program. For example, they may be used for program debugging, starting flowgraph execution, or signaling the host processor of various events. SNOOP and SNEAK may also interface with the entry and return points, respectively, of procedures implemented by the host processor.

Quantization Processor Unit (QPU)

The QPU (Section 4.4) implements the JPEG, MPEG-1, and H.261 forward and inverse quantization algorithms. It also performs variance, mean square, and adaptive quantizer computations. The JPEG, MPEG-1, and H.261 inverse quantization algorithms fully conform to all three standards. The MPEG-1 and H.261 forward quantization algorithms are derived from MPEG-1 ÒVideo Simulation Model 3Ó (SM3) and H.261 ÒReference Model 8Ó (RM8), respectively.

In place of the SM3 flat matrix, the QPU version of the SM3 non-intra forward quantization algorithm fully incorporates the non-intra quantization matrix of the MPEG-1 standard.

The adaptive thresholding algorithm implemented in the TFQUANT and CTFQUANT instructions is taken from Section 7.2.1 of SM3.

The MPEG-1 quantizer_scale or the H.261 macroblock quantizer is stored in the token descriptor quant field. The QPU provides three stages for adaptively modifying quant during MPEG-1 and H.261 encoding via the SETQUANT, ADAPTQ, CFQUANT, and CTFQUANT instructions. During the first stage, quant is initialized by the SETQUANT instruction which copies the contents of the DCU QUANTREG register (Section 4.6) into the quant field of the SETQUANT operand token. QUANTREG is modified by the host processor as needed to achieve a targeted bit rate.

During the second stage, the ADAPTQ instruction modifies quant as a linear function of changes in instantaneous image activity relative to average image activity, generally pushing quant higher for positive changes and lower for negative changes. This decreases quant in relatively flat image areas and increases it in busy areas in order to achieve good perceived quality across the entire reconstructed image without compromising compression. Prior to the execution of ADAPTQ, image activity is computed for the luminance token of a macroblock by the VAR instruction with the results stored in the sfield token descriptor field.

A block diagram of the ADAPTQ instruction execution appears in Figure 2.9. The various registers depicted in the block diagram are defined in the QPU register descriptions of Section 4.4.

During the third stage, the CFQUANT or CTFQUANT instruction automatically increases quant as needed to prevent excessive quantization clipping, and then performs forward quantization.

Excessive clipping causes image artifacts. These artifacts are especially visible for DCT coefficients closer to the DC coefficient. CFQUANT and CTFQUANT use the contents of the QPU LASTCLIP register to determine which coefficients to protect from excessive clipping. Details appear in the description of the LASTCLIP register in Section 4.4.

DCT Processor Unit (DPU)
The DPU performs forward and inverse discrete cosine transforms (DCTs) on 8x8 blocks of data. Forward DCTs process 9-bit twoÕs complement data into 12-bit twoÕs complement results. Inverse DCTs process 12-bit twoÕs complement data into 9-bit twoÕs complement results. The accuracy of the inverse DCT performed by the DPU fully conforms to the require ments of the JPEG, MPEG-1, and H.261 standards.
Arithmetic Processor Unit (APU)

The APU is the only functional unit capable of operating on two data token operands performing binary operations such as addition and subtraction. The most complex operation performed by the APU is the loop filter function as defined in the H.261 standard. The loop filter is conditionally invoked by the FILTER instruction (Table 3.1) depending on whether the filter token descriptor bit is set in the instruction operand token. Bidirectional predictions as defined in the MPEG-1 standard are computed using the AVERAGE instruction.

Host Interface Unit (HIU)
All host interface bus cycles supported by the HIU transfer 16-bit words synchronous to HCLK. Both burst and non-burst cycles are supported. Each non-burst read by the host processor requires four HCLK cycles (HMODE pin = 0). Each non-burst write requires two or three HCLK cycles (HMODE = 1 or 0, respectively).

Burst cycles transfer multiple words at the rate of one every one or two HCLK cycles (HMODE = 1 or 0, respectively). The host processor need only supply the address for the first word in a burst since the other addresses are generated internally by the HIU.

When accessing multiple words at the same address (e.g. the RPU INFIFO or OUTFIFO port), the host processor may disable internal incrementing of burst transfer addresses by writing a "1" to the BINCDIS register beforehand (Section 4.2).

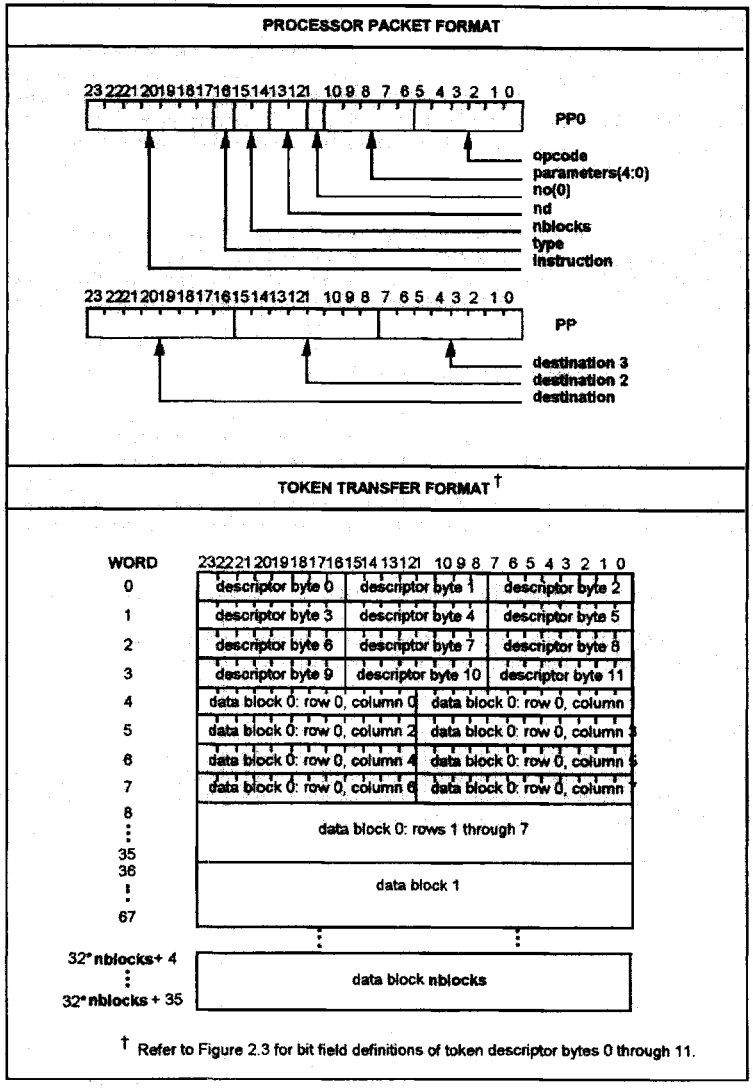
All burst cycles other than burst reads with HMODE = 1 are terminated by the deassertion of the HBRST pin during the final word transfer and may be up to 2048 words long. A burst read cycle with HMODE = 1 can only be up to 257 words long and requires that its length be written beforehand into the HIU BLENGTH register.

Auxiliary Interface Unit (AIU)
The DCU dispatches each external instruction to the AIU, which then transmits both the instruction and its operand token (if any) to the appropriate auxiliary processor unit via the XBUS. The auxiliary processor then executes the instruction and returns its result to the AIU, which then notifies the DCU. Each external instruction may process either zero or one operand token and create either zero or one result token. An auxiliary processor may execute up to four different instructions in parallel. For example, the KS0144 can simultaneously execute three external instruction classes called IPREDICT, OPREDICT, and WRMEM.

Figure 2.1 shows the least significant two bits (bits 1:0) of an external instruction opcode field selects the auxiliary processor to be used. Bits 3:2 of opcode specify the parallel execution class of the instruction. Instructions from different classes may execute in parallel, but only one instruction may execute at a time from any given class. The DCU maintains a 16-bit external instruction status table to track which classes have instructions executing on which auxiliary processors at any point in time.

The AIU uses auxiliary processor write cycles to transmit external instructions and operands over the XBUS. These cycles are 24-bit word data transfers consisting of two processor packet words (PP0 and PP1) followed by zero to 132 words of token data. The processor packet and token transfer formats appear in Figure 2.10.
One 24-bit word is transmitted every XCLK cycle. The processor packet consists of fields copied from the external instruction and nblocks and type fields copied from the operand token descriptor, if it exists.

Figure 2.10 - Auxally Processor Bus Data



The operand token is transmitted as four words containing the token descriptor followed by 32 words for each of (nblocks+1) token data blocks if the token is a data token. An auxiliary processor sends results back to the AIU by first requesting an auxiliary processor read cycle from the AIU.

The selected auxiliary processor does this by asserting its XRQST# pins (one of four) on the KS0143. The AIU acknowledges the auxiliary processor by broadcasting its address on the XBUS. The auxiliary processor then responds by sending back a processor packet corresponding to the instruction completing execution. The processor packet is followed by zero to 132 words of result token data, all of which have the formats shown in Figure 2.10. Depending on external instruction execution times, it is possible that several auxiliary processor write cycles may take place for different instructions before an auxiliary processor read cycle occurs.

Dataflow Control Unit (DCU)

The instructions executed by the DCU create tokens, modify token descriptors, copy tokens, and control program dataflow. The CRTOKEN instruction bootstraps flowgraph execution by firing only once and creating a token from a parameter list. Token descriptor modification instructions operate on either arbitrary bits within the token descriptor bytes of Figure 2.3 or on predefined fields such as hpos and vpos. Token copying instructions copy either all or part of one token to another.

Dataflow control instructions fall into two groups. Instructions in the first group, such as FGATE and DGATE1, act like spigots which turn dataflow on and off by either passing or annihilating tokens.

Instructions in the other group (semaphore instructions), do not turn dataflow on and off but rather delay turning it on until certain events occur. Semaphore instructions may be used to create dataflow multiplexers, guard the entrances to flowgraphs so as to prevent token memory overflow, and synchronize flowgraph execution with the host processor.

The semaphore instructions are INITSEM, TSTSEM, TSTDEC, and INCSEM. These instructions atomically manipulate and/or test the contents of the semaphore registers (SEMREG0, SEMREG1, SEMREG2, and SEMREG3) in the DCU memory map (Section 4.6).

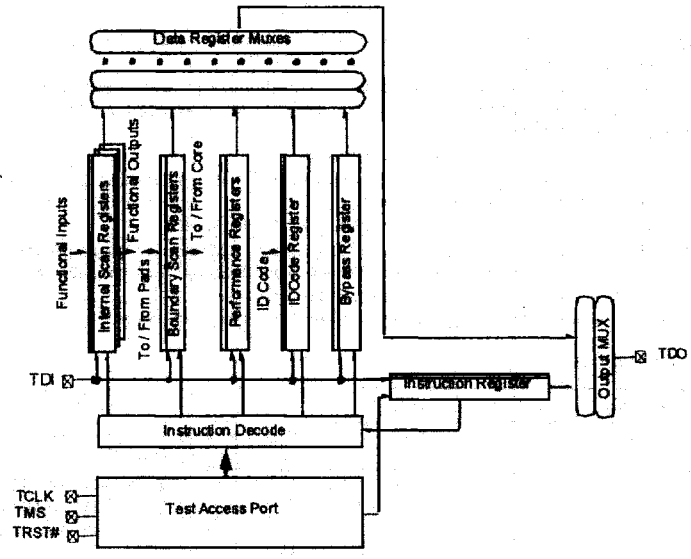
The semaphore registers act like global variables, allowing dataflow in different branches of a flowgraph to be synchronized.

For example, the TSTSEM instruction holds its operand token at its input until a semaphore matches the cntr1 field in the operand token's token descriptor. Typically, the semaphore's value is changed by a INCSEM instruction elsewhere in the program.

Test Control Unit (TCU)

A simplified block diagram of the KS0143 Test Control Unit (TCU), associated registers, and instruction set appear in Figure 2.11. The TCU supports boundary scan in compliance with the IEEE 1149.1 (JTAG) standard and also allows external logic to read and write the KS0143's internal scan paths. For further details regarding the KS0143 boundary and internal scan paths, please contact the Array Microsystems technical applications support group.

Figure 2.11 - Test Control Unit (TCU)



Instruction Set

SAMPLE/PRELOADLoad/sample boundary registers through I/O pins

EXTESTStimulate/observe I/O pins through serial test port

INTESTStimulate/observe IC core through serial test port

BYPASSQuick bypass of serial data input to serial data out

IDCODERead hardwired ID code through serial test port

PERFORMANCEEnable/observe internal performance ring counter

INSCANStimulate/observe internal data registers through serial test port (with reset)

INSAMPLEStimulate/observe internal data registers through serial test port (without reset)

SECTION3 - KS0143 INSTRUCTION SET

This section contains Tables 3.1 through 3.6 which describe the various instructions that the KS0143 executes internally. The table columns are defined as follows:

- Mnemonic is the instruction symbolic name as defined by the VIDEOFLOW Software Tools flowgraph editor.
- opcode and no are the contents of the corresponding instruction fields as shown in Figure 2.1:
  - > opcode is the instruction operation code.
  - > no identifies the number of operands required by the instruction.
- fmt identifies which of the four formats the instruction is (see Figure 2.1). It also identifies the maximum number of destinations the instruction can have in an KS0143 program.
- Unit identifies the KS0143 functional unit responsible for executing the instruction.
- Description contains the basic operation performed by the instruction including any required parameter fields. Parameter names are those defined by the VIDEOFLOW Software Tools flowgraph editor. The numbers preceding each parameter name identify the bit positions the parameter occupies within the instruction.

Table 3.7 contains the instruction timing for each of the instructions relative to PCLK cycles.

Table 3.1 - KS0143 Arithmetic Instructions

Mnemonic	op-code	no	fmt	Unit	Description
FILTER	01	1	3	APU	Conditionally performs H.261 loop filter on data token based on filter descriptor bit (no parameters)
A_{DD}	02	2	3	APU	Adds two data token and optionally clips the sum to the range [0.255] (0) clip: 1=enable clipping, 0=disable clipping
AVERAGE	03	2	3	APU	Adds two data tokens, halves the sum, and merges motion vectors (no parameters)
SUBTRACK	04	2	3	APU	Subtracts two data tokens (no parameters)
ADDCON	05	1	2	APU	Adds const to data token and optionally clips the sum to the range [0,255] (9) clip : 1=enable clipping, 0=disable clipping (8:0) const : constant with a range [-256,255]
CLIP	06	1	1	APU	Clips data token to fall within the range [minval, maxval] (15:8) minval : constant with the range [0,255] (7:0) maxval: constant with the range [0,255]
FDCT	07	1	3	DPU	Performs 8x8 inverse DCT on each of the data blocks within a data token (no parameters)
IDCT	08	1	3	DPU	Performs 8x8 inverse DCT on each of the data blocks within a data token (no parameters)
FQUANT	09	1	3	QPU	Forward quantizes and clips data token (4:2) clipsel (clipping range limits): 0 = ± 127, 1= ± 255, 2= ± 511, 3= ± 1023, 4= ± 2047 (!:0) algsel (quantization algorithm): 0=H.261, 1=MPEG, 2=JPEG
TFQUANT	0A	1	3	QPU	Adaptively thresholds, forward quantizes, and clips data token (4:2) clipsel (clipping range limits): 0= ± 127, 1= ± 255, 2= ± 511, 3= ± 1023, 4 (0) algsel (quantization algorithm): 0=H.261, 1=MPEG

Table 3.1 - KS0143 Arithmetic Instructions (Continued)

Mnemonic	op-code	no	fmt	Unit	Description
IQUANT	0B	1	3	QPU	Inverse quantizes data down (1:0) algsel (quantization algorithm): 0=H.261, 1=MPEG, 2=JPEG
MEANSQ	0C	1	3	QPU	Computers mean square value of Component 0 within data token (no parameters)
VAR	0D	1	3	QPU	Computes variance or spatial activity of Component 0 within data token (0) varsel : 0=compute variance, 1=compute spatial activity
ADAPTQ	34	1	3	QPU	Adapts quantizer quant field based on relative measure of spatial activity (no parameters)
CTFQUANT	38	1	3	QPU	Adaptively thresholds, forward quantizes and clips data token increasing quantizer as needed to prevent excessive quantization clipping (4:2) clipsel (clipping range limits): 0= ± 127, 1= ± 255, 2= ± 511, 3= ± 1023, 4= ± 2047 (0) algsel (quantization algorithm): 0=H.261, 1=MPEG
CFQUANT	39	1	3	QPU	Forward quantizes and clips data token, increasing quantizer as needed to prevent excessive quantization clipping (4:2) clipsel (clipping range limits): 0=clipsel (clipping range limits): 0= ± 127, 1= ± 255, 2= ± 511, 3= ± 1023, 4= ± 2047 (0) algsel (quantization algorithm): 0=H.261, 1=MPEG

Table 3.2 -KS0143 Logical Instructions

Mnemonic	op-code	no	fmt	Unit	Description
CMPVAL	0F	2	3	DCU	Compares the sfield descriptor fields of two tokens (no parameters)
TSTVAL	10	1	1	DCU	Compares the sfield descriptor field of a token against a constant (20:0) const : constant with a range [-2^{20}, 2^{20}-1]
TSTDSCR	11	1	1	DCU	Logically matches a specified token descriptor bit field against constant (19:6) bytesel: descriptor byte select with a range [0,11] (15:8) mask: boolean mask applied to selected byte (7:0) bconst: boolean constant used for bit-wise comparision
BOOL	12	2	3	DCU	Performs a logical operation between two token descriptor lflag fields (2) compl1: 1= complement lflag of first operand token (1) compl2: 1= complement lflag of second operand token (0) op (logical operation): 0=OR, 1=AND
TSTCNTR	35	1	2	DCU	Compares a specified token descriptor counter field against a constant (9:8) cntrsel (counter field): 0= vpos, 1=hpos, 2=cntr1, 3=cntr2 (7:0) const: constant with the range [0,255]

Table 3.3 - Host Interface Instructions

Mnemonic	op-code	no	fmt	Unit	Description
RUNENC	2B	1	0	RPU	Reorders token data block words in zigzag fashion, encodes them as (zero - run, level) pairs, and palces them in the RPU OUTFIFO for access by host (np parameters)
RUNENC.S	2C	1	3	RPU	Same as RUNENC, *except signals after writing all data to the* RPU OUTFIFO (no parameters)
RUNDEC	2D	0	3	RPU	Reads a series of (zero - run, level) pairs from the RPU INFIFO, decodes and inverse zigzags, and creates a data token from the result (no parameters)
SNOOP	2E	1	0	TIU	Copies token into Token Passing Buffer for access by host processor (no parameters)
SNEAK	30	0	3	TIU	Creates token from contents of Token Passing Buffer (no parameters)

Table 3.4 - KS0143 Descriptor Modification Instructions

Mnemonic	op-code	no	fmt	Unit	Description
SUBVAL	0E	2	3	DCU	Subtracts the sfield descriptor fields of two tokens (0) scale: 0=subtract the sfield descriptor bits of two tokens 1=divide the first sfield by two before subtracting
COPY	13	1	3	DCU	Copies an entire token or only the token's descriptor (0) copysel: 0= copy entire token 1= copy token descriptor only
CRTOKEN	15	0	3	DCU	Creates a control token based on instruction parameters (4:3) nblocks : result descriptor nblocks field (2:0) comps : result descriptor comps field
INCCNTR	16	1	2	DCU	Unconditionally adds const to a specified token descriptor counter field (9:8) cntrsel (counter field) : 0=vpos, 1=hpos, 2=cntr, 3=cntr2 (7:0) const : constant with the range [-128,255]
CINCCNTR	17	2	2	DCU	Conditionally adds a constant to a specified token descriptor counter field (10) lsense : iflag descriptor bit value required to enable increment operation (9:8) cntrsel : 0=vpos, 1=hpos, 2=cntr1, 3=cntr2 (7:0) const : constant with the range [-128,256]
ADDCNTR	18	2	3	DCU	Adds the specified token descriptor counter fields from two tokens (3:2) cntrsel1 (operand 1 counter field): 0=vpos, 1=hpos, 2=cntr1, 3=cntr2 (1:0) cntrsel2 (operand 2 counter field): 0=vpos, 1=hpos, 2=cntr1, 3=cntr2
SETDSCR	1B	1	1	DCU	Unconditionally sets a specified token descriptor bit field to a constant (19:16) bytesel : descriptor byte select with the range [0,11] (15:8) mask: boolean mask specifying bits in selected byte (7:0) bconst : boolean constant specifying values of selected bits
COPYFLD	37	2	2	DCU	Copies a specified token descriptor bit field from one token to another (12) copyall: 1=copu entire descriptor to output token (11:8) bytesel : descriptor byte select with the range [0,11] (7/:0) mask : boolean mask specifying bits in selected byte

Table 3.4 - KS0143 Descriptor Modification Instructions (Contiuned)

Mnemonic	op-code	no	fmt	Unit	Description
CSETDSCR	1C	2	1	DCU	Conditionally sets a specified token descriptor bit field to a constant (20) lsense : lflag descriptor bit value required to enable set operation (19:16) bytesel: descriptor byte select with the range [0,11] (15:8) mask: boolean mask specifying bits in selected byte (7:0) bconst: boolean constant specifying values of selected bits
SETQUANT	1D	1	3	DCU	Sets token descriptor quant field to the contents of the QUANTREG register (no parameters)

Table 3.5 - KS0143 Dataflow Control Unit Instructions

Mnemonic	op-code	no	fmt	Unit	Description
DGATE1	14	1	1	DCU	Gates token based on value of bit field in token's token descriptor (20) lsense : 0=pass token if bit field does not equal value 1=pass token if bit field equals value (19,16) bytesel : descriptor byte select with the range [0,11] (15:8) mask : boolean mask specifying bits in selected byte (7:0) bconst : boolean constant specifying values of selected bits
DGATE2	2F	2	1	DCU	Gates Operand 1 based on value of bit field in token descriptor of Operand 2 (20) lsense : 0=pass token if bit field does not equal value 1=pass token if bit field equals value (19:16) bytesel : descriptor byte select with the range [0,11] (15:8) mask : boolean mask specifying bits in selected byte (7:0) const : constant with the range [0,255]
CGATE	36	1	2	DCU	Gates token based on value of counter field in token's descriptor (10) lsense : 0=pass token if bit field does not equal value 1=pass token if bit field equals value (9:8) cntrsel (counter field): 0= vpos, 1=hpos, 2=cntr1, 3=cntr2 (7:0) const : constant with the range [0,255]
MINIMAX	31	2	3	DCU	Selects token having minimum or maximum sfield descriptor field (4) maxsel: 0=select minimum, 1=select maximum
GATE	1E	2	3	DCU	Gates token based on the lflag descriptor bit of another token (4) lsense : value of lflag in Operand 2 require to pass Operand 1
FGATE	1F	1	3	DCU	Gates token based on the state of selected bit in GFLAG register (4) lsense : 0=pass token if selected bit in GFLAG is 0 1=pass token if selected bit in GFLAG is 1 (3:0) flagsel : GFLAG bit select with the range [0,15]
INITSEM	22	1	2	DCU	Sets selected bits of a semaphore register to a constant (11:10) semreg : semaphore register select with the range [0,3] (9:8) semnib (semaphore nibble select): 0=low nibble, 1=high nibble, 2=entire register (7:0) const : constant
TSTSEM	20	1	2	DCU	Suspends dataflow until selected cntr1 bit field in token descriptor matches bit field in selected sempaphore register (11:10) semreg : semaphore register select with the range [0,3] (8) semnib (semaphore nibble select): 0=low nibble, 1=high nibble (3:0) mask : boolean mask specifying bits in cntr1 and selected nibble

Table 3.5 - KS0143 Dataflow Control Unit Instructions (Continued)

Mnemonic	op-code	no	fmt	Unit	Description
TSTDEC	3A	1	2	DCU	Suspends data flow until selected semaphore register is greater than or eual to a constant, then subtracts the constant from semaphore register (11:10) semreg: semaphore register select with the range [0,3] (9:8) semnib (semaphore nibble select): 0= low nibble, 1= high nibble, 2=entire register (7:4) const1 : constant with the range [0,15]; select if nblocks descriptor field in operand token is 2 or 3 (3:0) const2 : constant with the range [0,15]; selected if nblocks descriptor field in operand token is 0 or 1
INCSEM	21	1	2	DCU	Adds constant to selceted semaphore register (11:10) semreg ; semaphore register selecft with the range [0,3] (9:8) semnib (semaphore nibble select): 0= low nibble, 1= high nibble, 2= entire register (7:4) const1 : constant with the range [0,15]; selected if nblocks descriptor field in operand token is 2 or 3 (3:0) const2 : constant with the range [0,15]; selected if nblocks descriptor field in operand token is 0 or 1

Table 3.6 - KS0143 Video Memory Instructions

Mnemonic	op-code	no	fmt	Unit	Description
RDV16	23	1	2	VIU	Reads data token from 16-bit wide memory (12:10) memsel : memory select 0,1,2,3=uses Low Bus : 4,5,6,7 uses High Bus (9:8) corgsel: component origin register select with the range[0,3] (7:4) horgsel: subpicture horizontal origin divided by 128 with the range [0,15] (3:0) vorgsel: subpicture vertical origin divided by 128 with the range [0,15]
RDV16FMV	24	1	2	VIU	Reads data token from 16-bit wide memory offset by forward motion vector (12:10) memsel : memory select 0,1,2,3=uses Low Bus: 4,5,6,7 uses High Bus (9:8) corgsel: component origin register select with the range [0,3] (7:4) horgsel: subpicture horizontal origin divided by 128 with the range [0,15] (3:0) vorgsel: subpicture vertical origin divided by 128 with the range [0,15]
RDV16BMV	25	1	2	VIU	Reads data token from 16-bit wide memory offset by backward motion vector (12:10) memsel: memory select: 0,1,2,3=uses Low Bus;4,5,6,7 uses High Bus (9:8) corgsel: component origin register select with the range [0,3] (7:4) horgsel: subpicture horizontal origin divided by 128 with the range [0,15] (3:0) vorgsel: subpicture vertical origin divided by 128 with the range [0,15]

Table 3.6 - KS0143 Video Memory Instructions (Continued)

Mnemonic	op-code	no	fmt	Unit	Description
WRV16	26	1	0	VIU	Writes data token to 16-bit wide memory (12:10) memsel: memory select 0,1,2,3=uses Low Bus: 4,5,6,7 uses High Bus (9:8) corgsel: component origin register select with the range[0,3] (7:4) horgsel: subpicture horizontal origin divided by 128 with the range [0,15] (3:0) vorgsel: subpicture horizontal origin divided by 128 with the range [0,15]
WRV16.S	32	1	2	VIU	Writes data token to 16-bit wide memory and signals whern finished (12:10) memsel: memory origin register select with the range [0,3] (9:8) corgsel: component origin register select with the range [0,3] (7:4) horgsel: subpicture horizontal origin divided by 128 with the range [0,15] (3:0) vorgsel: subpicture vertical origin divide by 128 with the range [0,15]
RDV32	27	1	2	VIU	Reads data token from 32-bit wide memroy (12:10) memsel: memory select [0,7] (9:8) corgsel: domponent origin register select with the range [0,3] (7:4) horgsel: subpictute horizontal origin divide by 128 with the range [0,15] (3:0) vorgsel: subpicture vertical origin divided by 128 with the range [0,15]
RDV32FMV	28	1	2	VIU	Reads data token from 32-bit wide memory (12:10) memsel: memory select [0,7] (9:8) corgsel: component origin register select with the range [0,3] (7:4) horgsel: subpicture horizontal origin divide by 128 with the range [0,15] (3:0) vorgsel: subpicture vertical origin divided by 128 with the range [0,15]
RDV32BMV	29	1	2	VIU	Reads data token from 32-bit wide memroy (12:10) memsel: memory select [0,7] (9:8) corgsel: domponent origin register select with the range [0,3] (7:4) horgsel: subpictute horizontal origin divide by 128 with the range [0,15] (3:0) vorgsel: subpicture vertical origin divided by 128 with the range [0,15]
WRV32	2A	1	0	VIU	Writes data token to 32-bit wide memory (12:10) memsel: memory select [0,7] (9:8) corgsel: component origin register select with the range [0,3] (7:4) horgsel: subpicture horizontal origin divide by 128 with the range [0,15] (3:0) vorgsel: subpicture vertical origin divide by 128 with the range [0,15]
WRV32.S	33	1	2	VIU	Writes data token to 32-bit wide memory (12:10) memsel: memory select [0,7] (9:8) corgsel: component origin register select with the range [0,3] (7:4) horgsel: subpicture horizontal origin divide by 128 with the range [0,15] (3:0) vorgsel: subpicture vertical origin divide by 128 with the range [0,15]

3-10

Table 3.7- Instruction Timing in PCLK Cycles

Mnemonic	Unit	DCU Cycles	Unit Cycles	Qualifiers	Notes
ARITHMETIC INSTRUCTIONS					
A_{DD}	APU	24n+5	56n+16		1
ADDCON	APU	16n+4	48n+15		1
CLIP	APU	16n+4	48n+15		1
AVERAGE	APU	24n+5	56n+16		1
SUBTRACT	APU	24n+5	56n+16		1
FILTER	APU	16n+4	80n+27	token descriptor bit filter=1	1
FILTER	APU	16n+4	16n+6	token descriptor bit filter=0	1
FDCT	DPU	16n+4	80n+37		1
IDCT	DPU	16n+4	80n+35		1
FQUANT	QPU	16n+4	80n+21	algsel=0(H.261) or algsel=2(JPEG)	1
FQUANT	QPU	16n+4	144n+33	algsel=1(MPEG)	1
TFQUANT	QPU	16n+4	80n+21	algsel=0(H.261)	1
TFQUANT	QPU	16n+4	144n+33	algsel=1(MPEG)	1
C[T]FQUANT	QPU	16n+4	144n+33	algsel=1(MPEG)	1
C[T]FQUANT	QPU	16n+4	80n+21	algsel=0(H.261), clipsel=0, and quant >7	1
C[T]FQUANT	QPU	16n+4	80n+21	algsel=0(H.261), clipsel=1, and quant >3	1
C[T]FQUANT	QPU	16n+4	80n+21	algsel=0(H.261), clipsel=2, and quant >1	1
C[T]FQUANT	QPU	16n+4	80n+21	algsel=0(H.261), clipsel=3	1
C[T]FQUANT	QPU	16n+4	(84+p)n+14	algsel=0(H.261), clipsel=0, and quant $\leq$ 7	1.2
C[T]FQUANT	QPU	16n+4	(84+p)n+14	algsel=0(H.261), clipsel=1, and quany $\leq$ 3	1.2
C[T]FQUANT	QPU	16n+4	(84+p)n+14	algsel=0(H.261), clipsel=2, and quant =1	1.2
IQUANT	QPU	16n+4	80n+21	algsel=0(H.261), or algsel=2(JPEG)	1
IQUANT	QPU	16n+4	144n+30	algsel=1(MPEG)	1
MEANSQ	QPU	8n+4	8n+64q+27		1.3
VAR	QPU	8n+4	8n+64q+27	varsel=0	1.3
VAR	QPU	8n+4	8n+71q+14	varsel=1	1.3
ADAPTQ	QPU	8n+4	8n+18		1
LOGICAL INSTRUCTIONS					
CMPVAL	DCU	10	not applicate		
TSTVAL	DCU	10	not applicate		
TSTDSCR	DCU	10	not applicate		
TSTCNTR	DCU	10	not applicate		
BOOL	DCU	10	not applicate		
DESCRIPTOR MODIFICATION INSRUCTIONS					
SUBVAL	DCU	10	not applicate		
COPY	DCU	10	not applicate		
CRTOKEN	DCU	10	not applicate	executes only once between chip resets	
INCCNTR	DCU	10	not applicate		
CINCCNTR	DCU	10	not applicate		
ADDCNTR	DCU	10	not applicate		
COPYFLD	DCU	10	not applicate		
SETDSCR	DCU	10	not applicate		
CSETDSCR	DCU	10	not applicate		
SETQUANT	DCU	10	not applicate		

Table 3.7 - Instruction Timing in PCLK Cycles (Continued)

Mnemonic	Unit	DCU Cycles	Unit Cycles	Qualifiers	Notes
DATAFLOW CONTROL INSTRUCTIONS					
DGATE1	DCU	10	not applicate		
DGATE2	DCU	10	not applicate		
CGATE	DCU	10	not applicate		
MINIMAX	DCU	10	not applicate		11
GATE	DCU	10 or 11	not applicate		11
FGATE	DCU	10 or 11	not applicate		
INITSEM	DCU	10	not applicate		4
TSTSEM	DCU	10	not applicate		4
TSTDEC	DCU	10	not applicate		
INCSEM	DCU	10	not applicate		
VIDEO MEMORY INSTRUCTIONS					
RDV16	DCU	10	8n+4+r		1,5,6
RDV16FMV	DCU	10	8n+4+r		1,5,6
RDV16BMV	DCU	10	8n+4+r		1,5,6
WRV16	DCU	10	8n+4+r		1,5,6
WRV16.S	DCU	10 or 11	8n+4+r		1,5,6
RDV32	DCU	10 or 11	8n+4+r		1,5,6
RDV32FMV	DCU	10	8n+4+r		1,5,6
RDV32BMV	DCU	10	8n+4+r		1,5,6
WRV32	DCU	10	8n+4+r		1,5,6
WRV32.S	DCU	10	8n+4+r		1,5,6
Mnemonic	**Unit**	**DCU Cycles**	**Unit Cycles**	**Qualifiers**	**Notes**
HOST INTERFACE INSTRUCTIONS					
RUNENC	RPU	8n+2	73n+s+10	mbtype=0	1,7,8
RUNENC	RPU	8n+2	73n+s+12	mbtype=1	1,7,8
RUNENC	RPU	8n+2	73n+s+12	mbtype=2	1,7,8
RUNENC	RPU	8n+2	73n+s+14	mbtype=3	1,7,8
RUNENC.S	RPU	8n+4	73n+s+12	mbtype=0	1,7,8
RUNENC.S	RPU	8n+4	73n+s+14	mbtype=1	1,7,8
RUNENC.S	RPU	8n+4	73n+s+14	mbtype=2	1,7,8
RUNENC.S	RPU	8n+4	73n+s+16	mbtype=3	1,7,8
RUNDEC	RPU	8n+2	74n+s+9	mbtype=0	1,7,8
RUNDEC	RPU	8n+2	74n+s+11	mbtype=1	1,7,8
RUNDEC	RPU	8n+2	74n+s+11	mbtype=2	1,8,9
RUNDEC	RPU	8n+2	74n+s+13	mbtype=3	1,8,9
SNOOP	RPU	8n+2	8n+2+t		1,10
SHEAK	RPU	8n+2	8n+2	time assumes token is already in Token Passing Buffer	1

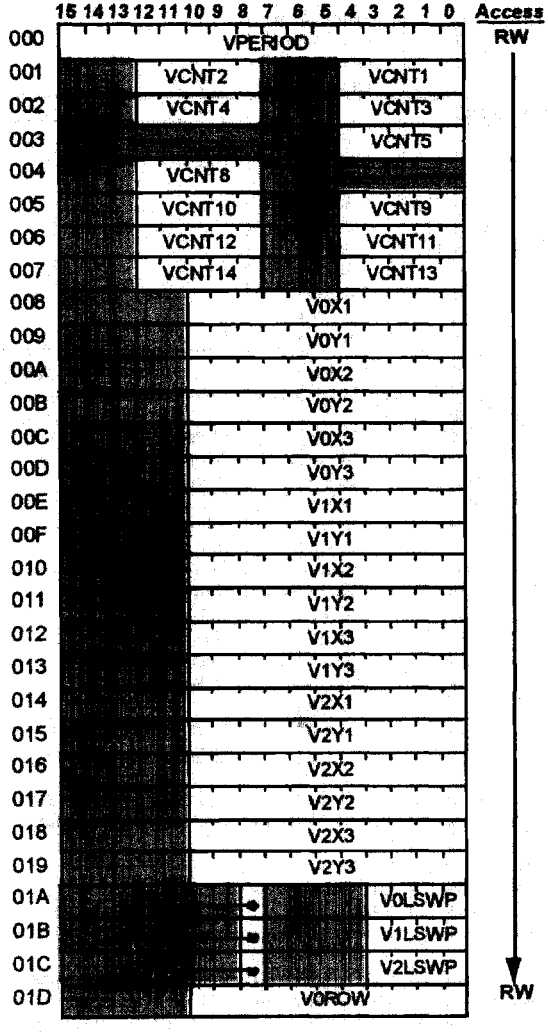
Notes:

1. n=numgber of data blocks in each operand token; n=0 if operand is control token
2. p=2((LASTCLIP+1)/2) where LASTCLIP is contents of QPU LASTCLIP register; " " denotes division with rounding to nearest integer (half values round up)
3. q=number of data blocks in component 0
4. Does not execute until semaphore test conditions satisfied
5. Time may increase due to refresh, SAM transfer, or VBUS arbitration running concurrently with instruction
6. r=time to read or write data token on video bus; depends on component CONFIG registers, VCLK period, and video timing setup
7. Instruction execution is delayed if RPU OUTFIFO fills completely during execution
8. s=number of words in token with levelN <-256 or level >255
9. Time assumes host has completely loaded zero-run length token into RPU INFIFO
10. t=time require for host processor to empty Token Passing Buffer
11. GATE/FGATE instructions take 11 cycles if token is annihilated

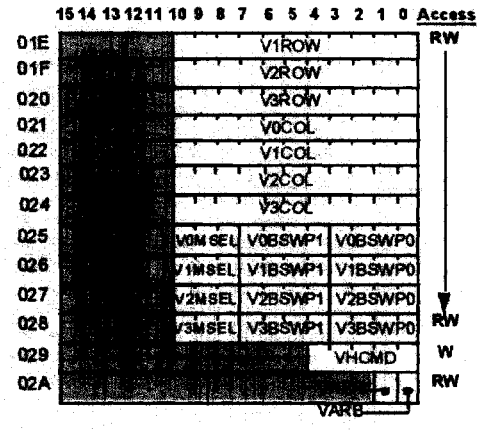
SECTION 4 - KS0143 REGISTER AND MEMORY MAPS

This section contains bit map diagrams and descriptions of KS0143 registers and memories which can be accessed by the external host processor. Unless indicated otherwise, shaded bits in the bit map diagrams have undefined contents. If these bits are written, they should be written with zeroes. Within each description table there is a column describing the Reset State of the register. Unless indicated otherwise, the Reset State shown is taken in response to either the assertion of the PRST# pin or the host processor writing to the GRESET register. Any Reset State annotated with asterisks (**) is only assumed in response to the assertion of the PRST# pin.

4.1 Video Interface Unit Registers



4.1 Video Interface Unit Registers (Continued)



Register Name	Address	Reset State	Description
VPERIOD (16 bits)	000 (15:0)	none	VIDEO BUS REFRESH PERIOD-the number of VCLK periods between successive CAS# - before -RAS# refresh cycles: need only be initialized if refresh is actived via the VREN register.
VCNT1 (5 bits)	001 (4:0)	none	VIDEO TIMING REGISTER 1- row address to RAS# delay ; the number of VCLK periods between the assertion of a row address on VA10:0 and the falling edge of the assertion of VRAS# during fast page mode or SAM transfer memory cycles. VCNT1 must be nonzero and the relation (VCNT1+ VCNT3)> VCNT2> VCNT1 must be satisfied.
VCNT2 (5 bits)	001 (12:8)	none	VIDEO TIMING REGISTER 2- row address to RAS# delay ; the number of VCLK periods between the assertion of a row address on VA10:0 and the falling edge of the assertion of VRAS# during fast page mode or SAM transfer memory cycles. VCNT2 must be nonzero and the relation (VCNT1+ VCNT3)> VCNT2> VCNT1 must be satisfied.
VCNT3 (5 bits)	002 (4:0)	none	VIDEO TIMING REGISTER 3- row address to RAS# delay ; the number of VCLK periods between the assertion of a row address on VA10:0 and the falling edge of the assertion of VRAS# during fast page mode or SAM transfer memory cycles. VCNT3 must be nonzero and the relation (VCNT1+ VCNT3)> VCNT2> VCNT1 must be satisfied.
VCNT4 (5 bits)	002 (12:8)	none	VIDEO TIMING REGISTER 4 - CAS# cycle time; the number of VCLK periods between successive filing edges of VCASO# during fast page mode memory cycles; must be nonzero.
VCNT5 (5 bits)	003 (4:0)	none	VIDEO TIMING REGISTER 5 - RAS# precharge time; the number of VCLK periods VRAS# is deasserted at the end of any fast page mode, SAM transfer, or fresh memory cycle; must be nonzero. NOTE: RAS# precharge time is held two additional VCLK cycles between row accesses.
VCNT8 (5 bits)	004 (12:8)	none	VIDEO TIMING REGISTER 8 - CAS# pulse width; the number of VCLK periods between the falling edge of each assertion and the rising edge of each deassertion of VCASO# during fast page mode memory cycles;must be nonzero.
VCNT9 (5 bits)	005 (4:0)	none	VIDEO TIMING REGISTER 9 - RAS# hold time; the number of VCLK periods between falling edge of the last assertion of VCASO# and the rising edge of the deassertion of VRAS# during fast page mode memory cycles; must be nonzero.

4.1 - Video Interface Unit Registers (Continued)

Register Name	Address	Reset State	Description
VCNT10 (5 bits)	005 (12:8)	none	VIDEO TIMING REGISTER 10 - CAS# to RAS# delay; the number of VCLK periods between the falling edge VCAS# assertion and the falling edge of VRAS# assertion during a CAS - before - RAS refresh memory cycle; must be nonzero.
VCNT11 (5 bits)	006 (4:0)	none	VIDEO TIMING REGISTER 11 - CAS# hold time; the number of VCLK periods between the falling edge of VRAS# assertion and the rising edge of VCASO# deassertion during a CAS - before - RAS refresh; must be nonzero.
VCNT12 (5 bits)	006 (12:8)	none	VIDEO TIMING REGISTER 12 - RAS# pulse width; the number of VCLK periods between the falling edge of VRAS# assertion and the rising edge of VRAS# deassertion during a CAS - before - RAS refresh; must be nonzero.
VCNT13 (5 bits)	007 (4:0)	none	VIDEO TIMING REGISTER 13 - OE# hold time; the number of VCLK periods between the falling edge of VCASOS# assertion and the rising edge of VOE# deassertion during a SAM transfer cycle; must be nonzero.
VCNT14 (5 bits)	007 (12:8)	none	VIDEO TIMING REGISTER 14 - RAS# / CAS# transfer hold time; the number of VCLK periods between the rising edge VOE# deassertion and the rising edge of VRAS# and VCASO# deassertions at the end of a SAM transfer cycle; must be nonzero.
V0INTLV V1INTLV V2INTLV (1 bit)	01A (8) 01B (8) 01C (8)	none	COMPONENT 0,1, AND 2 INTERLEAVE ENABLE REGISTERS - each register selects whether pels from its corresponding image component are stored in successive columns in video memory or in every other column; i.e., if an image has U and V chrominance components, it is advantageous to enable interleaving for each so that U and V chrominance components, it is advantageous to enable interleaving for each so that U and V samples may be stored in pairs within each row of video memory. V0INTLV, V1INTLV, and V2INTLV correspond to components 0, 1 and 2, and interleaving is enabled for a component if its interleave register is set to "1".
V0LSWP V1LSWP V2LSWP (4 bits)	01A (3:0) 01B (3:0) 01C (3:0)	none	COMPONENT 0,1, AND 2 LINE SWAP ENABLE REGISTERS - ViLSWP(i) is the j-th line swap enable bit for component I, where j is specified by the corgsel parameter of video memory instructions. If line swap enable bit="1", pels from even numbered lines (where the first line is line number 0) of the corresponding image component are stored in odd numbered video memory banks. Pels from odd numbered lines are stored in even numbered memory banks. If the bit is "0", even lines are stored in even banks, and odd lines are stored in odd banks (a memory "bank" is a memory connected to one of the KS0143 four video data buses; bank 0 is connected to VDD, etc). Line swapping is used to store luminance and chrominance pels from the same line in different banks, allowing them to be input or output at the same time.
V0X1 V0X2 V0X3 V1X1 V1X2 V1X3 V2X1 V2X2 V2X3 (11 bits)	008 (10:0) 00A (10:0) 00C (10:0) 00E (10:0) 010 (10:0) 012 (10:0) 014 (10:0) 016 (10:0) 018 (10:0)	none	COMPONENT 0, 1, AND 2 HORIZONTAL OFFSET REGISTERS- these registers contain horizontal address offsets which are added to the horizontal page offsets described by the horsel parameters in video memory instructions as part of DRAM column address formation. ViXj, where j is specified by the corgsel parameter of video memory instructions, is the j-th horizontal offset register for component I; the registers for j =1,2, and 3 reside (as shown here) in the KS0143 memory map and are user - definable, whereas the registers for j = 0 are "hardwired" to contain zero.
V0Y1 V0Y2 V0Y3 V1Y1 V1Y2 V1Y3 V2Y1 V2Y2 V2Y3 (11 bits)	008 (10:0) 00A (10:0) 00D (10:0) 00F (10:0) 011 (10:0) 013 (10:0) 015 (10:0) 017 (10:0) 019 (10:0)	none	COMPONENT 0, 1, AND 2 VERTICAL OFFSET REGISTERS - these registers contain vertical address offsets which are added to the vertical page offsets described by the vorgsel parameters in video memory instructions as part of DRAM row address formation. ViYj, where j is specified by the corgsel parameter of video memory instructions., is the j-th vertical offset register for component I; the registers for j=1,2, and 3 reside (as shown here) in the KS0143 memory map and are user-definable, whereas the registers for j=-0 are "hardwired" to contain zero.

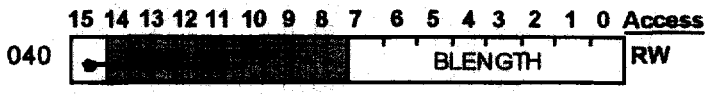
4.1 - Video Interface Unit Registers (Continued)

Register Name	Address	Reset State	Description
VHCMD (5 bits)	029 (4:0)	none	HOST SAM TRANSFER COMMAND REGISTER-this register may be written to by the host processor to select one of the KS0143's four SAM transfer row address counters (i.e. V0CNTR, V1CNTR, V2CNTR, or V3CNTR) for the purpose of incrementing or resetting it or commanding the KS0143 to use it to perform a SAM transfer. Counters 0 and 1 are used for SAM - to - DRAM transfers while counters 2 and 3 are used for DRAM - to - SAM transfers. The VCMD3:0 pins serve the same function as bits 3.0 of the VHCMD register except that they may be tied directly to an external video timing generator. To be effective, a given 4 bit command must be written to VHCMD(3:0) twice, first with VHCMD(4)=0 and then with VHCMD(4)=1. Bits 1:0 of VHCMD select the row address counter, and bits 3.2 specify the operation to be performed on the selected counter as follows: VHCMD(3:2) OPERATION 00 initialize counter from its corresponding starting row addr register 01 increment counter by 1 10 transfer VRAM row indicated by counter 11 transfer VRAM row indicated by counter; then increment counter by 1
V0COL V1COL V2COL V3COL (11 bits)	021(10:0) 022(10:0) 023(10:0) 024(10:0)	none	SAM TRANSFER TAP ADDRESS REGISTERS - these registers contain the starting or "tap" addresses for shifting out of or into VRAM serial access memory (SAM); the contents of these registers are output by the KS0143 as column addresses during SAM transfer cycles. Each register is associated with one of the KS0143's four SAM transfer row address counters; V0COL, V1COL, V2COL, and V3COL are associated with V0CNTR, V1CNTR, V2CNTR, and V3CNTR, respectively.
V0ROW V1ROW V2ROW V3ROW (11 bits)	01D(10:0) 01E(10:0) 01F(10:0) 020(10:0)	none	SAM TRANSFER STARTING ROW ADDRESS REGISTERS - these registers contain the starting row addresses used to initialize the KS0143's four SAM transfer row address counters; V0ROW, V1ROW, V2ROW, and V3ROW are associated with V0CNTR, V1CNTR, V2CNTR, and V3CNTR, respectively.
VOMSEL V1MSEL V2MSEL V2MSEL (3 bits)	025(10:8) 026(10:8) 027(10:8) 028(10:8)	none	SAM TRANSFER MEMORY SELECT REGISTERS - the contents of these registers are output on the VMSEL2:0 pins during SAM transfers; V0MSEL, V1MSEL, V2MSEL and V3MSEL are associated with V0CNTR, V1CNTR, V2CNTR, and V3CNTR, respectively.

4.1 - Video Interface Unit Registers (*Continued*)

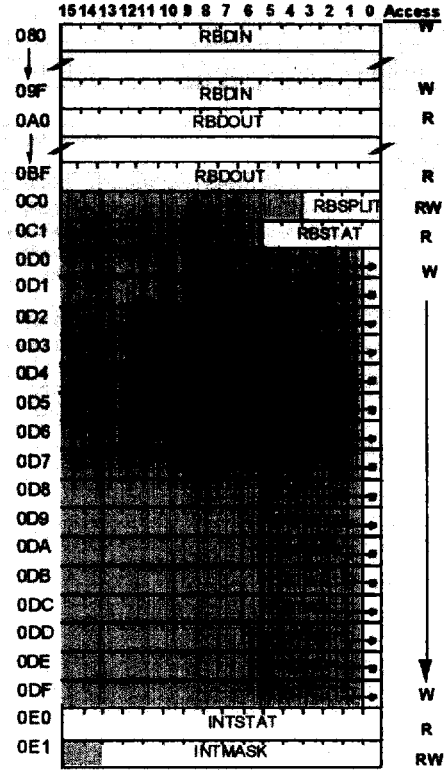
Register Name	Address	Reset State	Description
V0BSWP0 V0BSWP1 V1BSWP0 V1BSWP1 V2BSWP0 V2BSWP1 V3BSWP0 V3BSWP1 (4 bits)	025(3:0) 025(7:4) 026(3:0) 026(7:4) 027(3:0) 027(7:4) 028(3:0) 028(7:4)	none	SAM TRANSFER MEMORY BANK SWAP REGISTERS - these registers are used to select which of the four memory banks (each of which is connected to one of the KS0143's four video data buses) are accessed during SAM transfers. Each register bit corresponds to a memory bank (e.g. bit 0 corresponds to bank 0, etc.). When a bit is set to "1", the bank's VRAS# and VCASO# pins are active during a transfer. Registers are selected for use during transfers by way of their association with the four SAM transfer row address counters (V0BSWP0 and V0BSWP1 are associated with V0CNTR, V1BSWP0 and V1BSWP1 are associated with V1CNTR, etc.). Transfers involving a given counter alternate between the two swap registers associated with that counter; e.g. even numbered transfers (where the first transfer is number 0) involving V0CNTR use swap register V0BSWP0 and odd numbered transfers use V0BSWP1.
VREN (1 bit)	02A (1)	none	VIDEO MEMORY REFRESH FUNCTION ENABLE REGISTER - if set to "1", the CAS#-before-RAS# refresh function on the KS0143 is enabled and starts running immediately; if set to ÒOÓ, the function is disabled.
VARB (1 bit)	02A (0)	none	VIDEO BUS ARBITRATION MASTER ENABLE REGISTER - if set to "1", the KS0143 is internally configured as the highest priority master on the video bus arbitration daisy chain (i.e. the KS0143 ignores its VGRNTI pin and puts its own requests for use of the video bus above those of external requests from other devices).

4.2 - Host Interface Unit Register



Register Name	Address	Reset State	Description
BINCDIS (1 bit)	040 (15)	0**	HOST BUS BURST INCREMENT DISABLE REGISTER - if this register is set to "1", the KS0143 does not internally increment addresses during burst transfers; otherwise, it increments normally. Setting BINCDIS to "1" is useful when the host needs to read or write the same KS0143 memory location several times in succession (e.g. when polling a register or accessing the RPU rate buffer).
BLENGTH (8 bits)	040 (7:0)	0**	HOST BUS MODE 1 BURST READ CYCLE LENGTH REGISTER - if the host processor wishes to perform a Mode 1 (i.e. single cycle) burst read cycle with the KS0143, this register must first be initialized with two less than the actual number of words to be transferred during the cycle. This register need not be initialized for any other type of read or write cycles, including Mode1 burst writes.

4.3 - RUN Length Processor Registers

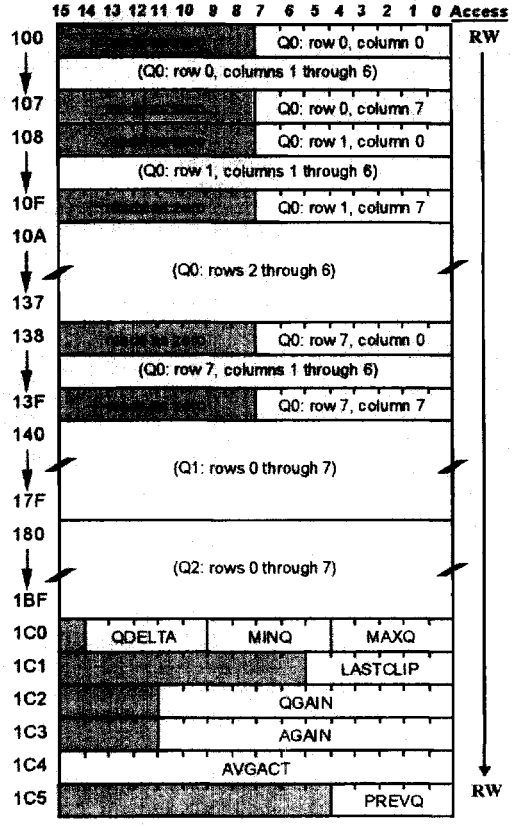


Register Name	Address	Reset Start	Description
RBDIN (16 bits)	080 - 09F (15:0)	none	RATE BUFFER DATA INPUT PORT - a write-only input port to the INFIFO portion of the RPU rate buffer; used by the host processor to write compressed image data to the KS0143. Any of the 32 addresses may be used to access this register for burst-rate transfers.
RBDOUT (16 bits)	0A0 - 0BF (15:0)	none	RATE BUFFER DATA OUTPUT PORT - a read-only output port from the OUTFIFO portion of the RPU rate buffer; used by the host processor to read compressed image data from the KS0143. Any of the 32 addresses may be used to access this register for burst-rate transfers.
RBSPLIT (4 bits)	0C0 (3:0)	none	RATE BUFFER PARTITION CONTROL REGISTER - determines how the 256 word by 16 bit RPU rate buffer is split between its INFIFO and OUTFIFO portions. If the most significant bit of RBSPLIT is zero, INFIFO is (RBSPLIT x 32) words long, while OUTFIFO is 256 - (RBSPLIT x 32) words long; if the most significant bit of RBSPLIT is one, INFIFO is 256 words long and OUTFIFO is nonexistent.

4.3 - Run Length Processor Registers (Continued)

Register Name	Address	Reset Start	Description
RBSTAT (6 bits)	0C1 (5:0)	none	RATE BUFFER STATUS REGISTER - a read-only register that indicates the fullness of the RPU rate buffer; the most significant three bits of RBSTAT correspond to INFIFO, and the least significant three bits correspond to OUTFIFO. These bits are interpreted as follows (k = 0 for OUTFIFO and k = 1 for INFIFO): RBSTAT(3k+2:3k) DEFINITION 100 FIFO empty 000 FIFO not empty but £ 25% full 001 FIFO more than 25% full £ 50% full 011 FIFO more than 50% full but £ 75% full 010 FIFO more than 75% full but < 100% full 110 FIFO completely full The host may also obtain this status information by enabling one or more of KS0143 interrupts 0 through 11.
INSTAT (Read status) (16 bits)	0E0 (15:0)	0	INTERRUPT STATUS REGISTER - gives the status of KS0143 interrupt conditions. If a bit in this register is set to '1", the corresponding interrupt condition as noted in the following table has occurred. The host processor typically reads this register after receiving an interrupt from the KS0143. Once a status bit is set to "1", it continues to assert an interrupt until: 1) the KS0143 is reset or, 2) the host writes a "0" to that bit. See description of INSTAT (Clear Status) READ STATUS BIT POSITION — STATUS BIT DEFINITION — CLEAR STATUS BIT ADDRESS 0 OUTFIFO completely full 0D0 1 OUTFIFO more than 75% full but<100% full 0D1 2 OUTFIFO more than 50% full but ≤ 75% full 0D2 3 OUTFIFO more than 25% full but ≤ 50% full 0D3 4 OUTFIFO not empty but ≤ 25% full 0D4 5 OUTFIFO empty 0D5 6 INFIFO completely full 0D6 7 INFIFO more than 75% full but < 100% full 0D7 8 INFIFO more than 50% full but ≤ 75% full 0D8 9 INFIFO more than 25% full but ≤ 25% full 0F9 10 INFIFO not empty but ≤ 25% full 0DA 11 INFIFO empty 0DB 12 TPBSEL=3: TIU Token Passing Buffer contains a new token resulting from the execution of a SNOOP instruction TPBSEL=2: TIU Token Passing Buffer is now empty resulting from the execution of a SNEAK instruction. 0DC 13 KS0143 Error 0DD 14 reserved 0DE 15 reserved 0DF
INSTAT (Clear status) (1 bit)	0DD-0DF (0)	0	INTSTAT appears at multiple addresses (0D0 - 0DF) in the KS0143 memory map (shown on the previous page) for clearing the interrupt status condition. Write a zero to the address corresponding to the interrupt that is to be cleared (see table above).
INTMASK (16 bits)	0E1 (15:0)	0	INTERRUPT MASK REGISTER - each bit in this register corresponds to an interrupt condition and is logically "and-ed" with that condition prior to its being loaded into the INTSTAT register; if a mask bit is "0", the corresponding interrupt condition is unconditionally set to "0" (i.e. disabled). The HINT# pin is then asserted if any of the bits in INTSTAT is set to "1". The two most significant bits of INTMASK should always be set to "0" since the corresponding bits in INTSTAT are currently reserved.

4.4 Qiamtozatopm Processor Register

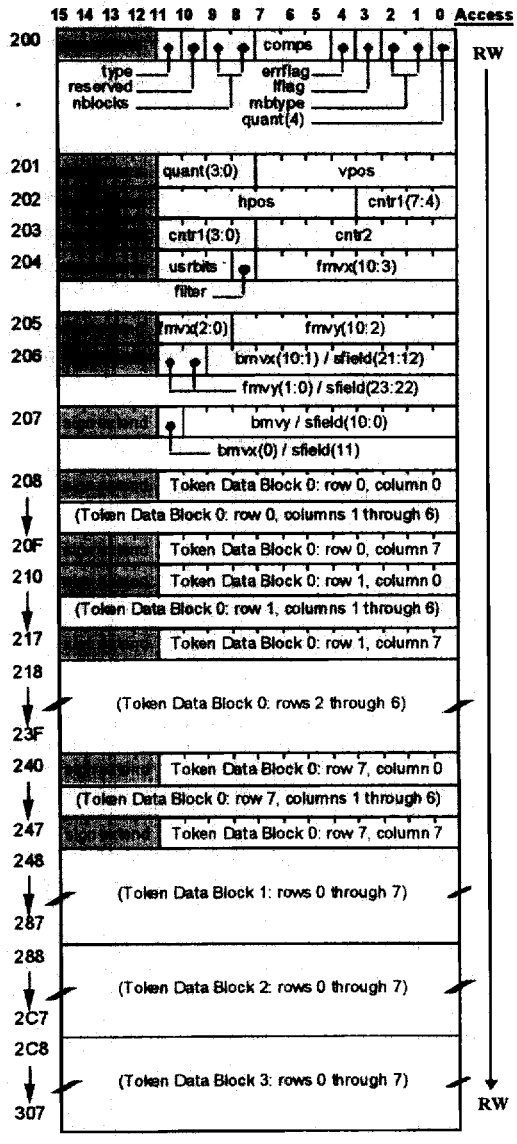


SAMSUNG ELECTRONICS

4. 4 Quantization Processor Registers (Continued)

Register Name	Address	Reset Start	Description
Q0[0..7,0..7] (8 bits) Q1[0..7,0..7] (8 bits) Q2[0..7,0..7] (8 bits)	100 - 13F (7:0) 140 - 17F (7:0) 180 - 1BF (7:0)	none	QUANTIZATION MATRICES - each of these three matrices consists of eight rows by eight columns by eight bits and is addressed in the memory map in "row major" format. For applications using JPEG quantization, each matrix is used to forward or inverse quantize pels from its like-numbered component (i.e. Q0 corresponds to component 0, etc.). For applications using MPEG quantization, Q0 is the intra-mode quantization matrix, while Q1 is the non-intra matrix; Q2 is not used. Finally, for H.261 quantization, no quantization matrices are used. Quantization matrix entries and quantizers (i.e. token descriptor quant fields) having a zero value force corresponding forward or inverse quantized results to be zero.
QDELTA (5 bits) MINQ (5 bits) MAXQ (5 bits) QGAIN (12 bits) AGAIN (12 bits) AVGACT (16 bits) PREVQ (5 bits)	1C0 (14:10) 1C0 (9:5) 1C0 (4:0) 1C2 (11:0) 1C3 (11:0) 1C4 (15:0) 1C5 (4:0)	none	ADAPTIVE QUANTIZATION CONTROL REGISTERS - used by the KS0143's ADAPTQ instruction to modify the 5 bit quantizers (stored in the token descriptor quant field of ADAPTQ input tokens) as a linear function of changes in instantaneous image activity relative to average image activity; the tendency of ADAPTQ is to push quantizers higher for relative increases in image activity and lower for relative decreases. Instantaneous image activity is computed prior to the ADAPTQ instruction using the VAR instruction and is stored in the sfield(15:0) token descriptor field of the ADAPTQ input token. As shown in Figure 2.9, ADAPTQ computes a new quantizer by subtracting its running estimate of average image activity (stored in the AVGACT register) from the instantaneous activity, multiplying the difference by QGAIN, scaling the product, and then adding it to the input quantizer. This new quantizer is then clipped as needed to fall inclusively between MINQ and MAXQ. QDELTA establishes a guardband which may be used to prevent the quantizer from changing too often; i.e. if clipping is not required and the new quantizer is within QDELTA of PREVQ, the new quantizer is reset to PREVQ (i.e. the quantizer computed by the previous execution of ADAPTQ). Finally, PREVQ and AVGACT are both updated. AVGACT is updated by multiplying the previously computed activity difference by AGAIN, scaling the product, and then adding it to AVGACT. Note that QGAIN controls the rate at which the quantizer adapts to changes in image activity, and AGAIN controls the rate at which the estimate of average image activity reacts to changes in instantaneous activity. QGAIN and AGAIN each contain an unsigned positive fractional integer with the binary point to the left of the most significant bit. AVGACT contains an unsigned positive integer. QDELTA and the token descriptor quant field are always interpreted as containing unsigned positive integers.
LASTCLIP (6 bits)	1C1 (5:0)	none	QUANTIZATION CLIPPING CONTROL REGISTER - used by the KS0143 CFQUANT and CTFQUANT instructions to prevent excessive quantization clipping of a selected number of DCT coefficients during MPEG or H.261 forward quantization. In quantizing one or more 8x8 blocks of DCT coefficients, the CFQUANT and CTFQUANT instructions examine coefficients 0 through LASTCLIP of all blocks in zigzag order. This determines the smallest quantizer which will not cause excessive quantization clipping of these coefficients (note: if a token is intra-coded, coefficient 0 of each block is skipped during the clipping control operation). If the new minimum quantizer is greater than quant, then it replaces the quantizer supplied by the input token quant token descriptor field for use in quantization of all data blocks.

4.5 Token Interface Unit Register

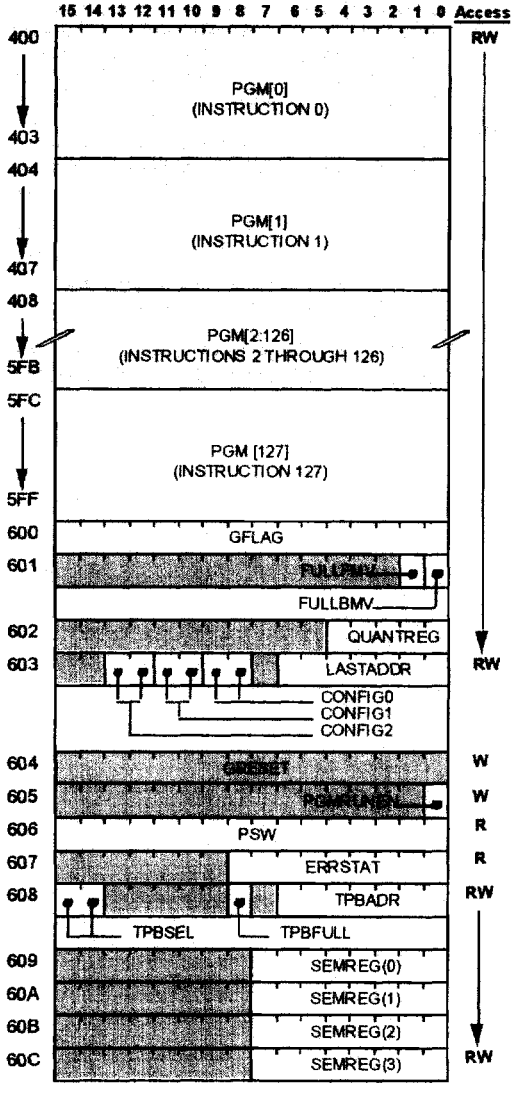


SAMSUNG ELECTRONICS

4.6 Token Interface Unit Registers (Continued)

Register Name	Address	Reset Start	Description
TPB (16 bits)	200 - 307 (15:0)	none	TOKEN PASSING BUFFER (TPB) MEMORY - used in conjunction with KS0143 SNEAK and SNOOP instructions to pass data or control tokens between the KS0143 and host processor. The host processor may write a token to the TPB for copying by a SNEAK instruction into the dataflow of an KS0143 program. Similarly, the host processor may read a token from the TPB which was written by a SNOOP instruction from an KS0143 program. The host controls usage of the 264 word TPB via the TPBSEL, TPBADR, and TPBFULL registers and may enable KS0143 interrupt 12 (see the INTSTAT register description) to signal it when the TPB is empty or full. The TPB may hold only one data or control token at any point in time. The first eight words of the TPB (200 - 207) contain the token descriptor of a control or data token. The remaining 256 words may contain up to four 8x8 data blocks. The four most significant bits of each 16-bit word in the TPB are ignored when written by the host processor and sign-extended (i.e. contain four copies of bit 11) when read.

4.6 Dataflow Control Unit Register



4.6 Dataflow Control Registers (*Continued*)

Register Name	Address	Reset Start	Description
PGM[0:127] (64 bits)	400 - 5FF (15:0)	none	PROGRAM MEMORY - contains the user KS0143 program. Each 64-bit word is mapped onto four consecutive memory locations and contains one instruction. Only the most significant 40 bits of each instruction are defined. The host processor may only read or write the program memory when the DCU is not running.
GFLAG (16 bits)	600 (15:0)	none	GLOBAL FLAG REGISTER - contains bits used by FGATE instructions in the user program to control program dataflow. Each bit is treated as a boolean constant. GFLAG is typically written before the user program starts to execute. The host processor however, may write to GFLAG during program execution without interfering with the simultaneous execution of an FGATE instruction.
FULLFMV (1 bit)	601 (1)	none	FORWARD MOTION VECTOR ÒFULL PELÓ FLAG - used by RDV16FMV and RDV32FMV instructions in the user program to determine whether the forward motion vector stored in the token descriptor of an operand token has full or half pel resolution. 1 = full pel resolution, 0 = half pel resolution.
FULLBMV (1 bit)	601 (0)	none	BACKWARD MOTION VECTOR ÒFULL PELÓ FLAG - used by RDV16BMV and RDV32BMV instructions in the user program to determine whether the backward motion vector stored in the token descriptor of an operand token has full or half pel resolution. 1 = full pel resolution, 0 = half pel resolution.
QUANTREG (5 bits)	602 (4:0)	none	QUANTIZER REGISTER - the contents of QUANTREG are written to the quant field of operand tokens by the SETQUANT instruction. QUANTREG is typically written by the host processor during the execution of MPEG or H.261 encoder programs to set a "target" quantizer value to control the compressed bit rate. Other instructions such as ADAPTQ, CFQUANT, and CTFQUANT may then be used to modify the quantizer value based on image quality criteria. The host processor may write to QUANTREG during program execution without interfering with a SETQUANT instruction which may be executing at the same time.
CONFIG0 (2 bits) CONFIG1 (2 bits) CONFIG2 (2 bits)	603 (9:8) 603 (11:10) 603 (13:12)	none	COMPONENT CONFIGURATION REGISTERS - each of these registers describes the number and geometric configuration of 8 by 8 blocks of image pels making up an image component for the purpose of reading and writing image memories and storing pels in data tokens. In the following table, CONFIGn is the configuration register for component n. A component may have one of the following four possible configurations: CONFIGn EFINITION 0 a single 8x8 pel block 1 two horizontally aligned 8x8 pel blocks 2 two vertically aligned 8x8 pel blocks 3 four 8x8 pel blocks arranged in a 2x2 block configuration Configurations are assigned to components in sets since they also specify the relative horizontal and vertical sampling ratios between different components. For example, the configuration assignments for MPEG and H.261 YUV components (assuming Y = component 0, U = component 1, and V = component 2) are CONFIG0 = 3, CONFIG1 = 0, and CONFIG2 = 0 since the horizontal and vertical spatial resolutions of the U and V components are each half those of the Y component.
LASTADDR (7 bits)	603 (6:0)	none	LAST PROGRAM ADDRESS - the highest address in the user program. The DCU assumes that the user program is located between address 0 and address LASTADDR in the program memory.
GRESET (16 bits)	604 (15:0)	none	GLOBAL RESET - a write - only register written by the host processor to perform a "soft" reset of the KS0143. Writing any value to GRESET halts execution of the user program and brings the entire KS0143 to a predefined initial state. All registers and memories in the KS0143 memory map assume their reset states except where noted.
PGMRUNEN (1 bit)	605 (0)	0	PROGRAM RUN ENABLE - a write-only bit written by the host processor to enable or disable execution of the user program. 1 = start execution, 0 = halt execution.

4.6 - Dataflow Control Registers (Continued)

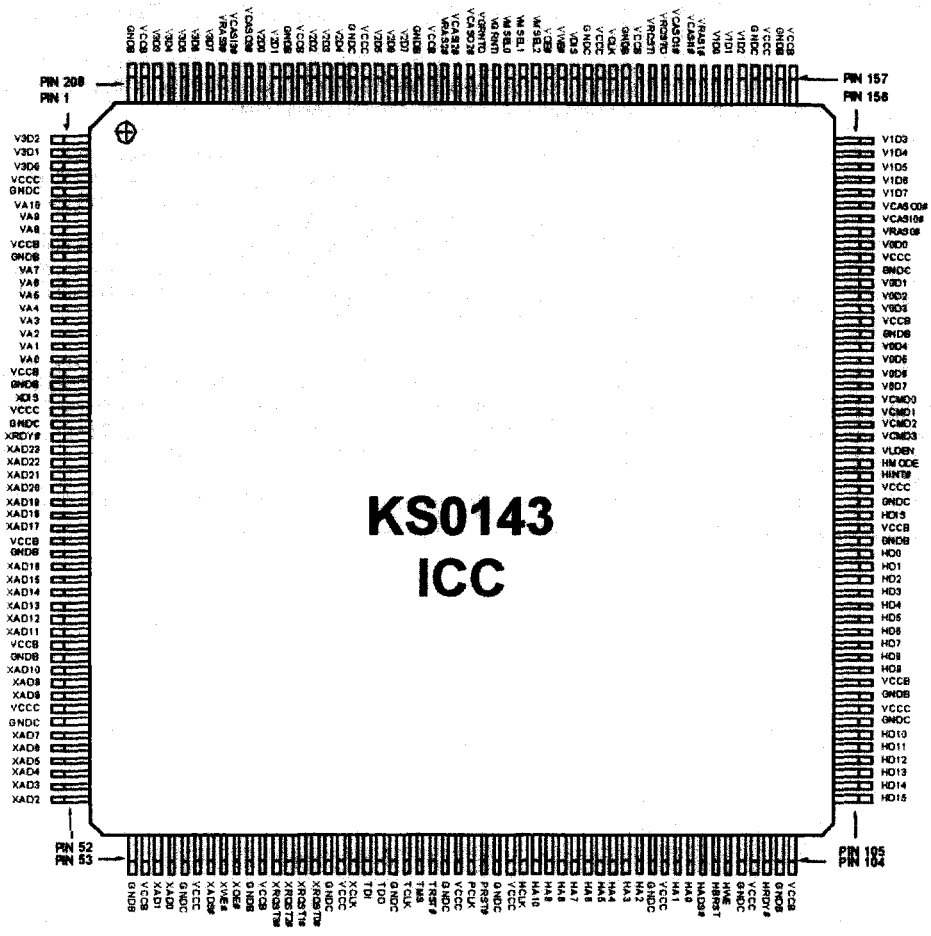
Register Name	Address	Reset Start	Description
PSW (16 bits)	606 (15:0)	none	PROCESSOR STATUS WORD - this read-only register may be read at any time by the host processor to check the "busy" status of the KS0143 internal functional units as well as any auxiliary processors hooked to the XBUS. Its contents are defined as follows (if a bit is set to a "1", the corresponding condition is satisfied): PSW bit — DEFINITION 0 — Dataflow Control Unit is running 1 — Token Interface Unit is enabled for SNEAK or SNOOP and TPB is full 2 — Video Interface Unit is busy executing an instruction 3 — Run Length Processor is busy executing an instruction 4 — DCT Processor Unit is busy executing an instruction 5 — Quantization Processor unit is busy executing an instruction 6 — Arithmetic Processor Unit is busy executing an instruction 7 — Aux Interface Unit is busy communicating with DCU or aux processor 8 — reserved 9 — reserved 10 — reserved 11 — reserved 12 — Auxiliary Processor Unit 0 is busy executing an instruction 13 — Auxiliary Processor Unit 1 is busy executing an instruction 14 — Auxiliary Processor Unit 2 is busy executing an instruction 15 — Auxiliary Processor Unit 3 is busy executing an instruction
ERRSTAT (9 bits)	607 (8:0)	0	ERROR STATUS REGISTER - this read-only register is read by the host processor to determine the source of various KS0143 errors detected by the DCU during program execution; the host may determine the existence of an error condition by either polling ERRSTAT or allowing itself to be interrupted by KS0143 interrupt 13 (see INTSTAT register description). ERRSTAT contents are defined as follows: ERRSTAT bits — DEFINITION 8:7 — error code: 0 = no error 1 = errflag bit set in result token descriptor 2 = undefined instruction opcode field 3 = token memory overflow 6:0 — address of KS0143 instruction causing error
TPBSEL (2 bits)	608 (15:14)	0	TOKEN PASSING BUFFER FUNCTION SELECT - written by the host processor to specify the function of the TPB with regard to the servicing of KS0143 SNEAK and SNOOP instructions. The contents of this register are defined as follows: TPBSEL — DEFINITION 0 — disable the execution of all SNEAK and SNOOP instructions 1 — disable the execution of all SNEAK and SNOOP instructions 2 — enable the execution of the SNEAK instr at the addr specified by TPBADR 3 — enable the execution of any SNOOP instruction
TPBADR (7 bits)	608 (6:0)	0	TOKEN PASSING BUFFER INSTRUCTION ADDRESS - contains the address of either the: · SNEAK instruction to be enabled for execution when TPBSEL = 2 · last SNOOP instruction executed and whose operand token resides in the TPB when TPBSEL = 3.

4.6 - Dataflow Control Registers (Continued)

Register Name	Address	Reset Start	Description
TPBFULL (1 bit)	608 (8)	0	TOKEN PASSING BUFFER FULL FLAG - indicates whether or not the TPB contains valid data: · SNEAK instructions: TPBFULL is set to a "1" by the host processor after it has placed a new token in the TPB. The KS0143 sets TPBFULL to "0" after the SNEAK instruction specified by the TPBADR has executed. · SNOOP instructions: TPBFULL is set to a "1" by the KS0143 after it has placed a SNOOP instruction operand token in the TPB. The KS0143 will not load another token into the TPB until the host processor sets TPBFULL to "0" once it has finished accessing the TPB.
SEMREG0 (8 bits) SEMREG1 (8 bits) SEMREG2 (8 bits) SEMREG3 (8 bits)	609 (7:0) 60A (7:0) 60B (7:0) 60C (7:0)	0 0 0 0	SEMAPHORE REGISTERS - These registers are used as semaphores within the KS0143 parallel processor environment to synchronize token dataflow within different portions of a program flowgraph. The KS0143 instructions which use these registers are INITSEM, TSTSEM, TSTDEC, and INCSEM. Applications of semaphores not involving host processor access of these registers include the formation of "time division" token multiplexers using TSTSEM and INCSEM instructions and the controlling of token memory utilization via TSTDEC and INCSEM instructions. In addition, the host processor may control program dataflow by writing to semaphores which are then tested using the TSTSEM instruction. The host processor may read semaphores at any time and write them whenever the user program halts.

SECTION 5 - KS0143 PIN ASSIGNMENTS AND DESCRIPTIONS

Figure 5.1 - KS0143 Pin Layout



SAMSUNG ELECTRONICS

Table 5.1 - KS0143 Singal Name to Pin Number Correspondence

Name	Pin Number	Name	Pin Number	Name	Pin Number	Name	Pin Number
HA0	96	V0D3	143	VCASI1#	165	XAD22	26
HA1	95	V0D4	140	VCASI2#	183	XAD23	25
HA2	92	V0D5	139	VCASI3#	200	XADS#	59
HA3	91	V0D6	138	VCLK	171	XCLK	70
HA4	90	V0D7	137	VCMD0	136	XDIS	21
HA5	89	V1D0	163	VCMD1	135	XOE#	61
HA6	88	V1D1	162	VCMD2	134	XRDY#	24
HA7	87	V1D2	161	VCMD3	133	XSRQST0#	67
HA8	86	V1D3	156	VDIS	174	XSRQST1#	66
HA9	85	V1D4	155	VGRNTI	180	XRQST2#	65
HA10	84	V1D5	154	VGRNTO	181	XRQST3#	64
HADS#	97	V1D6	153	VLDEN	132	XWE#	60
HBRST	98	V1D7	152	VMSEL0	179		
HCLK	83	V2D0	198	VMSEL1	178		
HD0	124	V2D1	197	VMSEL2	177		
HD1	123	V2D2	194	VOE#	176	VCCB (+5V)	9,19,32,40, 54,63,104, 114,126,142, 157,169,185, 195,207
HD2	122	V2D3	193	VRAS0#	149		
HD3	121	V2D4	192	VRAS1#	164		
HD4	120	V2D5	189	VRAS2#	184		
HD5	119	V2D6	188	VRAS3#	201		
HD6	118	V2D7	187	VRQSTI	168		
HD7	117	V3D0	3	VRQSTO	167		
HD8	116	V3D1	2	VWE#	175		
HD9	115	V3D2	1	XAD0	56	VCCC (+5V)	4,22,45,58, 69,78,82,94, 101,112,129, 147,159,172, 190
HD10	110	V3D3	206	XAD1	55		
HD11	109	V3D4	205	XAD2	52		
HD12	108	V3D5	204	XAD3	51		
HD13	107	V3D6	203	XAD4	50		
HD14	106	V3D7	202	XAD5	49		
HD15	105	VA0	18	XAD6	48		
HDIS	127	VA1	17	XAD7	47	GNDB (ground)	10,22,33,41, 53,62,103, 113,125,141, 158,170,186, 196,208
HINT#	130	VA2	16	XAD8	44		
HMODE	131	VA3	15	XAD9	43		
HRDY#	102	VA4	14	XAD10	42		
HWE	99	VA5	13	XAD11	39		
PCLK	79	VA6	12	XAD12	38		
PRST#	80	VA7	11	XAD13	37		
TCLK	74	VA8	8	XAD14	36		
TDI	71	VA9	7	XAD15	35	GNDC (ground)	5,23,46,57, 68,73,77,81, 93,100,111, 128,146,160, 173,191
TDO	72	VA10	6	XAD16	34		
TMS	75	VCASO0#	151	XAD17	31		
TRST#	76	VCASO1#	166	XAD18	30		
V0D0	148	VCASO2#	182	XAD19	29		
V0D1	145	VCASO3#	199	XAD20	28		
V0D2	144	VCASI0#	150	XAD21	27		

Table 5.2 - KS0143 Pin Number to Signal Name Correspondence

Name	Pin Number	Name	Pin Number	Name	Pin Number	Name	Pin Number
1	V3D2	53	GNDB	105	HD15	157	VCCB
2	V3D1	54	VCCB	106	HD14	158	GNDB
3	V3D0	55	XAD1	107	HD13	159	VCCC
4	VCCC	56	XAD0	108	HD12	160	GNDC
5	GNDC	57	GNDC	109	HD11	161	V1D2
6	VA10	58	VCCC	110	HD10	162	V1D1
7	VA9	59	XADS#	111	GNDC	163	V1D0
8	VA8	60	XWE#	112	VCCC	164	VRAS1#
9	VCCB	61	XOE#	113	GNDB	165	VCASI1#
10	GNDB	62	GNDB	114	VCCB	166	VCASO1#
11	VA7	63	VCCB	115	HD9	167	VRQSTO
12	VA6	64	XRQST3#	116	HD8	168	VRQSTI
13	VA5	65	XRQST2#	117	HD7	169	VCCB
14	VA4	66	XRQST1#	118	HD6	170	GNDB
15	VA3	67	XRQST0#	119	HD5	171	VCLK
16	VA2	68	GNDC	120	HD4	172	VCCC
17	VA1	69	VCCC	121	HD3	173	GNDC
18	VA0	70	XCLK	122	HD2	174	VDIS
19	VCCB	71	TDI	123	HD1	175	VWE#
20	GNDB	72	TDO	124	HD0	176	VOE#
21	XDIS	73	GNDC	125	GNDB	177	VMSEL2
22	VCCC	74	TCLK	126	VCCB	178	VMSEL1
23	GNDC	75	TMS	127	HDIS	179	VMSEL0
24	XRDY#	76	TRST#	128	GNDC	180	VGRNTI
25	XAD23	77	GNDC	129	VCCC	181	VGRNTO
26	XAD22	78	VCCC	130	HINT#	182	VCASO2#
27	XAD21	79	PCLK	131	HMODE	183	VCASI2#
28	XAD20	80	PRST#	132	VLDEN	184	VRAS2#
29	XAD19	81	GNDC	133	VCMD3	185	VCCB
30	XAD18	82	VCCC	134	VCMD2	186	GNDB
31	XAD17	83	HCLK	135	VCMD1	187	V2D7
32	VCCB	84	HA10	136	VCMD0	188	V2D6
33	GNDB	85	HA9	137	V0D7	189	V2D5
34	XAD16	86	HA8	138	V0D6	190	VCCC
35	XAD15	87	HA7	139	V0D5	191	GNDC
36	XAD14	88	HA6	140	V0D4	192	V2D4
37	XAD13	89	HA5	141	GNDB	193	V2D3
38	XAD12	90	HA4	142	VCCB	194	V2D2
39	XAD11	91	HA3	143	V0D3	195	VCCB
40	VCCB	92	HA2	144	V0D2	196	GNDB
41	GNDB	93	GNDC	145	V0D1	197	V2D1
42	XAD10	94	VCCC	146	GNDC	198	V2D0
43	XAD9	95	HA1	147	VCCC	199	VCASO3#
44	XAD8	96	HA0	148	V0D0	200	VCASI3#
45	VCCC	97	HADS#	149	VRAS0#	201	VRAS3#
46	GNDC	98	HBRST	150	VCASI0#	202	V3D7
47	XAD7	99	HWE	151	VCASO0#	203	V3D6
48	XAD6	100	GNDC	152	V1D7	204	V3D5
49	XAD5	101	VCCC	153	V1D6	205	V3D4
50	XAD4	102	HRDY#	154	V1D5	206	V3D3
51	XAD3	103	GNDB	155	V1D4	207	VCCB
52	XAD2	104	VCCB	156	V1D3	208	GNDB

Table 5.3 - KS0143 Pin Descriptions

Name (1)	Type (2)	PU/PD (3)	Description
			VIDEO MEMORY INTERFACE (VBUS)
VA[10:0]	TSO	PU	VIDEO ADDRESS BUS - multiplexed row or column address asserted in conjunction with VRAS# and VCASO#. Connected to the DRAM or VRAM address pins.
VMSEL[2:0]	TSO	PU	VIDEO MEMORY SELECT - selects the desired DRAM or VRAM from those that are connected to the video data bus. The state of the VMSEL pins is determined by a video memory instruction parameter. VMSEL pins are driven to the low state while the KS0143 is performing a memory refresh.
V0D[7:0] V1D[7:0] V2D[7:0] V3D[7:0]	I/O I/O I/O I/O	PU PU PU PU	VIDEO DATA BUSES - four buses that transfer video data between the off-chip DRAMs or VRAMs and the KS0143. Each bus transfers a single 8-bit pel during the execution of 16-bit video memory instructions. V0D and V1D or V2D and V3D are used to transfer pels as selected by an instruction parameter. All four buses are used simultaneously to transfer pels during the execution of 32-bit video memory instructions.
VRAS[3:0]#	TSO	PU	VIDEO MEMORY ROW ADDRESS STROBES - when active (low), enables the VA[10:0] pins to select the row address of the selected DRAM or VRAM. Connected to the RAS# pins of memories connected to the KS0143.
VCASO[3:0]#	TSO	PU	VIDEO MEMORY COLUMN ADDRESS STROBE OUTPUT - when active (low), enables the VA[10:0] pins to select the column address of the selected DRAM or VRAM. Connected to the CAS# pins of memories connected to the KS0143.
VCAS[3:0]#	I	PU	VIDEO MEMORY COLUMN ADDRESS STROBE INPUT - when active (low), latches the data from the corresponding video data bus during the execution of video memory read instructions. When common page mode DRAMs and VRAMs are used, the VCAS# pins should be connected to the corresponding VCASO# pins using as short a path as possible. If hyper page mode DRAMs and VRAMs are being used, the VCAS# pins should be tied to the low state.
VWE#	TSO	PU	VIDEO MEMORY WRITE ENABLE - when active (low), enables writing to the selected DRAM or VRAM. Connected to the WE# pins of all memories connected to the KS0143.
VOE#	TSO	PU	VIDEO MEMORY OUTPUT ENABLE - when active (low), enables reading from the selected DRAM or VRAM. Connected to the OE# pins of all memories connected to the KS0143.
VDIS	I	PU	VIDEO BUS DISABLE - when active (high), places a high impedance state on the output drivers of the following video bus pins: VA, VMSEL, V0D, V1D, V2D, V3D, VRAS#, VCASO#, VWE#, and VOE#.
VRQSTI VRQSTO	I O	PD none	VIDEO BUS REQUEST DAISY CHAIN (INPUT and OUTPUT) - two signals used by the potential bus masters to request the video bus during arbitration. VRQSTI connects to VRQSTO of the next lower priority bus master. VRQSTO connects to VRQSTI of the next higher priority bus master. VRQSTI of the lowest priority bus master is tied to the low state.
VGRNTI VGRNTO	I O	PD none	VIDEO BUS GRANT DAISY CHAIN (INPUT and OUTPUT) - two signals used in the awarding of the video bus to a bus master during bus arbitration. VGRNTI connects to VGRNTO of the next higher priority bus master. VGRNTO connects to VGRNTI of the next lower priority bus master. VGRNTI of the highest priority master is tied to either the low or high state. If the KS0143 has not been granted the video bus, the following output drivers are placed into the high impedance state: VA, VMSEL,V0D, V1D, V2D, V3D, VRAS#, VCASO#, VWE#, and VOE#.
VCLK	I	PD	VIDEO BUS CLOCK - the master clock input used by the KS0143 for asserting and sampling all signals on the video bus. The KS0143 controls the timing of all memory read, write, and SAM transfer operations using programmable registers (VCNTn) which count periods of VCLK.

Table 5.3 - KS0143 Pin Description (Continued)

Name (1)	Type (2)	PU/PD (3)	Description
VCMD[3:0] VLDEN	I I	PD PD	VIDEO BUS SAM TRANSFER CONTROLS - five pins that allow an external video timing generator to command the KS0143 to transfer data between the serial access memory (SAM) and DRAM portions of VRAMs connected to the video bus. These pins control the initialization and incrementing of internal address counters associated with these transfers. The KS0143 enables transfers on selected video data buses via corresponding VRAS# and VCASO# pins. VLDEN enables the synchronous (with respect to VCLK) loading of a command from the VCMD pins. As a general rule, successive command inputs must be separated from one another by at least eight VCLK periods.
			HOST PROCESSOR INTERFACE (HBUS)
HCLK	I	PD	HOST INTERFACE CLOCK - the clock used by the KS0143 to assert and /or sample signals on the host interface bus (with the exception of HDIS). HBUS signals are synchronously sampled or asserted relative to the rising edge of HCLK. HCLK is tied to the bus clock of the host processor.
HA[10:0]	I	PU	HOST ADDRESS BUS - used to select an external memory map location for reading or writing.
HADS#	I	PU	HOST ADDRESS STROBE - when asserted (low), indicates a new host bus cycle is starting. The contents of HA[10:0], HWE, and HMODE are all asserted in conjunction with HADS#.
HD[15:0]	I/O	PU	HOST DATA BUS - used to transfer 16-bit words between the host and the external memory map.
HWE	I	PU	HOST WRITE ENABLE - indicates that a write or read operation is taking place. Asserted (high), in conjunction with HADS#, indicates a write operation, while deasserted indicates read.
HBRST	I	PD	HOST BURST CYCLE REQUEST - when asserted (high), requests another transfer at the next sequential address. The address for the next transfer is internally generated by the KS0143. HBRST is deasserted at the end of a read or write data transfer.
HDIS	I	PU	HOST BUS DISABLE - when asserted (high), forces the Host Data Bus output drivers to the high impedance state regardless of any other control signals.
HMODE	I	PD	HOST BUS TRANSFER MODE - asserted in conjunction with HADS# to select the data transfer timing mode for the next read or write cycle. Asserted (high) indicates Timing Mode 1, deasserted indicates Timing Mode 0.
HRDY#	OD	none	HOST BUS DATA READY- when asserted (low), indicates that valid data is present on the host bus for either a read or write operation. HRDY# is an open drain output requiring external pull-up/pull-down resistors.
HINT#	O	none	HOST INTERRUPT - when asserted (low), indicates the occurrence of various internal events as selected by the INTMASK register. HINT# is a level-sensitive signal and remains asserted until the host processor ÒclearsÓ it by writing to the INTSTAT register.
			AUXILIARY PROCESSOR INTERFACE (XBUS)
XAD[23:0]	I/O	PU	AUXILIARY PROCESSOR ADDRESS or DATA BUS - multiplexed bus that communicates processor addresses, external instructions, and/or token data between the KS0143 and selected auxiliary processor.
XADS#	O	none	AUXILIARY PROCESSOR ADDRESS STROBE - when asserted (low), indicates the start of a new read or write cycle between an auxiliary processor and KS0143. The auxiliary processor address is simultaneously asserted on XAD[1:0].
XWE#	O	none	AUXILIARY PROCESSOR WRITE ENABLE - asserted (low) by the KS0143 with XADS# to signal the beginning of an auxiliary processor write cycle. Deasserted by the KS0143 to indicate a read.
XOE#	O	none	AUXILIARY PROCESSOR OUTPUT ENABLE - when asserted (low) by the KS0143, enables the XAD[23:0] output drivers of an auxiliary processor for a bus read operation. The auxiliary processor was previously selected during the assertion of XADS#.

Table 5.3 - KS0143 Pin Descriptions (Continued)

Name (1)	Type (2)	PU/PD (3)	Description
XDIS	I	PU	AUXILIARY BUS DISABLE - when asserted (high), forces the XAD bus output drivers of the auxiliary processor into a high impedance state regardless of any other control signals.
XRDY#	I	PU	AUXILIARY PROCESSOR DATA READY- when asserted (low), indicates that the selected auxiliary processor is ready to accept or send data on the bus.
XRQST[3:0]#	I	PU	AUXILIARY PROCESSOR SERVICE REQUEST - when asserted (low), indicates that an auxiliary processor has completed executing an instruction previously sent to it by the KS0143. Each pin is connected to only one auxiliary processor, i.e. XRQSTn# is asserted by auxiliary processor n, where n = 0, 1, 2, or 3.
XCLK	I	PD	AUXILIARY INTERFACE BUS CLOCK- the master clock used by the KS0143 for asserting and sampling all signals on the auxiliary bus.
			IEEE 1149.1 TEST INTERFACE
TDI	I	PD	TEST DATA INPUT - serial data input shifted into an internal scan path or the boundary scan path for the purpose of testing internal logic or I/O buffers; synchronous to the rising edge of TCLK.
TDO	O	none	TEST DATA OUTPUT - serial data output from an internal scan path or the boundary scan path; synchronous to the falling edge of TCLK.
TCLK	I	PD	TEST CLOCK - strobes data from or to the test access port.
TMS	I	PU	TEST MODE SELECT - when asserted (high), controls the operation of the test access port.
TRST#	I	PU	TEST RESET - when asserted (low), asynchronously resets the test access port. When not using the IEEE test access port for testing, tie this signal to the low state.
			PROCESSOR CLOCK AND RESET
PCLK	I	PD	PROCESSOR CLOCK - the primary clock for all KS0143 internal processing and data transfers.
PRST#	I	PU	PROCESSOR RESET - when asserted (low), asynchronously sets the KS0143 to its initial state.
			POWER SUPPLY
GNDB, GNDC			CHIP GROUND - 31 pins which must be connected to a board ground plane.
VCCB, VCCC			CHIP POWER - 30 pins which must be connected to a board V_{CC} plane (nominally, +5.0V)

Notes:

1. Name Definitions : > Pin names ending in "#" are active - low
   > Pin names not ending in "#" are active - high
   > Names with an "x:y" designation indicate a bus of pins numbered "y" to "x", where "x" is the most significant bit.
2. Type Definitions :
   > I = Input
   O = Output
   > I/O = Bidirectional
   > TSO = Tri - state output
   > OD = Open drain output
3. PU/PD Definitions :
   > PD = An active pull - down device (~200K ohms) is present (built into the pin's circuitry; see Figure 7.6)
   > PU = An active pull - up device (~200K ohms) is present (built into the pin's circuitry; see Figure 7.6)
   > none = no pull - up or pull- down device is present

SECTION 6 - PROGRAMMING THE VBUS TIMING CONTROL REGISTERS

Section 4.1 describes the KS0143 VCNTN registers which may be programmed by the host processor to support a wide range of dram and vram devices at various clock speeds. these registers set the relationship and duration of the various strobes used to access memory on the vbus.
video memory bus timing is established in multiples of vclk clock periods as shown in figures 6.1 through 6.3. in the following paragraphs, the symbol tv refers to the vclk duration (tv = 1/vclk). for example, if VCLK is 40 MHz, then t_v = 25ns.

NOTE:

The VCNTN registers must not be set to the value 0.

Page mode memory cycle timing

The KS0143 accesses the selected dram and/or vram using fast page mode memory cycles (see figure 6.1). The first step in programming the vcntn registers for fast page mode operation is to select the cas Cycle time with the vcnt4 and vcnt8 registers. the vcnt8 register contents represent the active (low) cas pulse width and must be large enough to satisfy the minimum cas pulse width (TCAS), the access time from cas (TCAC), and the column address hold time (TCAH) specified for the memories in use:

VCNT8 * TV ≥ MAX (TCAS, TCAC, TCAH)

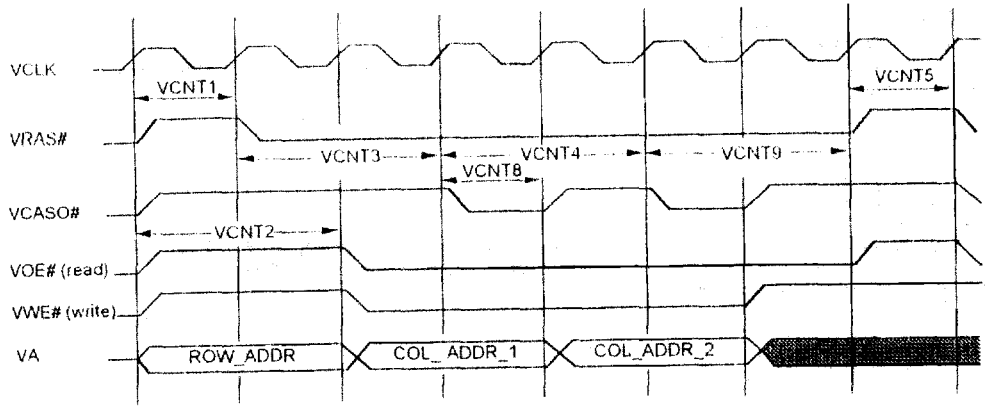
The VCNT4 register contents represent the cycle time of the cas signal and must be selected to satisfy The minimum cas cycle time (TPC) without causing a violation of the minimum cas precharge time (TCP) or The column address setup time (TASC). This is described by the following relation:

VCNT4 * TV ≥ MAX (TPC,VCNT8 * TV + TCP ,VCNT8 * TV + TASC)

The VCTN4 register contents represent the cycle time of the cas signal and must be selected to satisfy the minimum cas cycle time (TPC) without causing a violation of the minimum cas precharge time (TOP) or the column address setup time (TASC). This is described by the following relation:

VCNT3 * TV ≥ MAX (TRCD, TRAC - VCNT8 * TV, TCSH - VCNT8 * TV)

Figure 6.1 - VBUS FAST PAGE MODE TIMING DIAGRAM

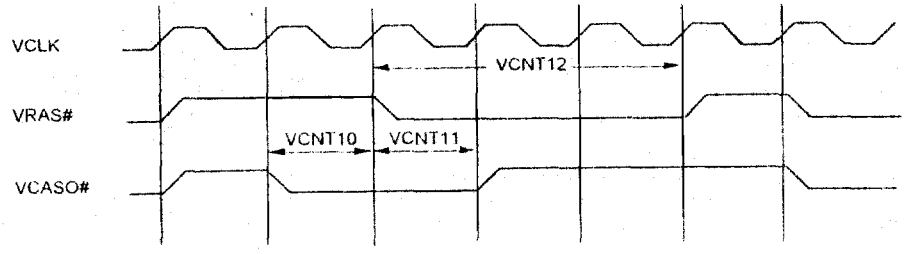


SAMSUNG ELECTRONICS

The VCNT9 register contents must be set to at least VCNT4 and possibly more to satisfy the RAS hold time (tRSH):
VCNT9 * tv ≥ max (VCNT4 * tv, tRSH)

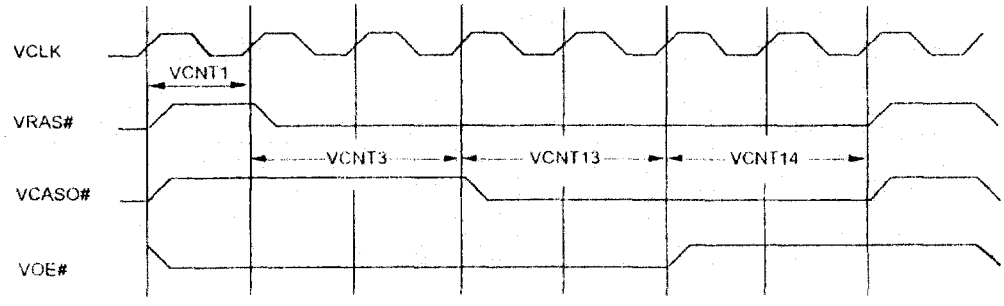
Refresh Timing
If the refresh feature of the KS0143 is used, the refresh timing parameters must be initialized (see Figure 6.2). For CAS before RAS refresh cycles, the contents of VCNT10 and VCNT11 can typically be set to "1" since both the CAS setup time (tCSR) and the CAS hold time (tCHR) are less than one clock cycle (tv). VCNT12, however, must meet the minimum RAS pulse width (tRAS) which is usually 3 to 4 cycles:
VCNT12 * tv ≥ 2 tRAS

Figure 6.2 - VBUS Refresh Timing Diagram



VRAM - SAM Transfer Cycle Timing
If VRAMs are used on the KS0143, the SAM transfer cycle timing parameters must also be initialized (see Figure 6.3). The VCNT14 register contents can typically be set to "1" since the Data Transfer to RAS precharge delay (tTRD) time is usually less than one clock cycle. The VCNT13 register contents must be set to a large enough value to meet the minimum RAS pulse width (tRAS):
VCNT13 * tv ≥ tRAS - (VCNT3 + VCNT14) * tv
The contents of VCNT1 register usually remains set to "1" since the Write setup (tWSR) and Data Transfer setup (tTLS) time is 0 for all standard VRAMs.

Figure 6.3 - VBUS SAM Transfer Cycle Timing Diagram



SECTION 7 - ELECTRICAL SPECIFICATIONS

Table 7.1 - Absolute Maximum Ratings

Parameter	Max Rating
Power Supply Voltage	-0.5 V to 7.0
DC Output Voltage (applied in HI - Z state)	-0.5V to 7.0V
Low Level Output Current	20mA
Case Temperature Under Bias	-55 ℃ to 110 ℃
Storage Temperature	65 ℃ to 150 ℃

Note : Operation at any Max Rating is not limplied. Ratings are provided for guidance purposes only and not tested.
Exposure to absolute maximum rating conditions over extended periods may affect device reliability.

Table 7.2 - Operating Conditions

Parameter	Symbol	Test Conditions	Limits		
			Min	Max	Units
Supply Voltage	V_{CC}		4.75	5.25	V
Case Temperature	T_C		0	100	℃

Table 7.3 - DC Elcetrical Characteristics (within operating conditions)

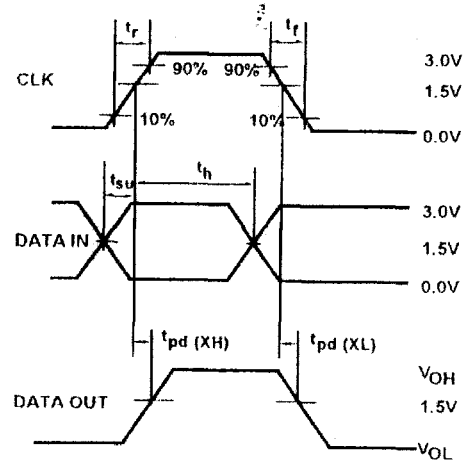
Parameter	Symbol	Test Conditions	Limits		
			Min	Max	Units
Standby Current	KS0143Q	V_{CC}=Max, f=0.0MHz (all clocks) XDIS=HDIS=VDIS=5.0V V_{IH}=5.0V, V_{IL}=0.0V, I_{OUT}=0.0mA		200	mA
Dynamic Supply Current	KS0143A	V_{CC}=Max, f=50.0MHz (PCLK) f=25MHz (XCLK, VCLK, HCLK) V_{IH}=5.0V, V_{IL}=0.0V, I_{OUT}=0.0mA		1200	mA
Input High Voltage	V_{IH}		2.0		V
Input Low Voltage	V_{IL}			0.8	V
Output High Voltage	V_{OH}	V_{CC}=Min, I_{OH}=-4.0mA	2.4		V
Output Low Voltage	V_{OL}	V_{CC}=Min, I_{OL}=8.0mA		0.4	V
Output Low Voltage Open Drain	V_{OLD}	V_{CC}=Min, I_{OLD}=30.0mA		0.4	V
Input High Current Pull - up	I_{IHU}	V_{CC}=Max, V_{IN}=V_{CC}		5.0	μA
Input High Current Pull - down	I_{IHD}	V_{CC}=Max, V_{IN}=V_{CC}		30.0	μA
Input Low Current Pull - down	I_{ILU}	V_{CC}=Max, V_{IN}=0.0V	-30.0		μA
Input Low Current Pull - down	I_{ILD}	V_{CC}=Max, V_{IN}=0.0V	-5.0		μA
Output Hi - Z Current Pull - up	I_{OZH}	V_{CC}=Max, V_{OUT}=V_{CC}		5.0	μA
Output Hi - Z Current Pull - up	I_{OZL}	V_{CC}=Max, V_{OUT}=0.0V	-30.0		μA
Input Clamp Voltage	V_{IC+}	V_{CC}=0.0V, I_{IN}=1mA	0.3	1.5	V
Input Clamp Voltage	V_{IC-}	V_{CC}=5.0V, I_{IN}=-1mA	-1.5	-0.3	V
Input Pin Capacitance**	C_{IN}	V_{CC}=5.0v, T_C=25 ℃		7.0	pF
I/O Pin Capacitance**	$C_{I/O}$	V_{CC}=5.0V, T_C=5 ℃		10.0	pF

** = Parameter is guranteed by design and characterization data, but not tested.

Table 7.4 - AC Test Conditions

Parameter	Test Condition
AC Switching Waveform Input Rise (tr) Time and Fall (tf)Time (10-90%) See Figures 7.1 and 7.2	4.0 ns (Max)
AC Switching Waveform Input Timing Reference Level See Figures 7.1 and 7.2	1.5V
AC Switching Waveform Output Timing Reference Level See Figures 7.1 and 7.2	1.5V
Output Load: See AC Output Equivalent Load Configuration (Figure 7.3) See Open Drain Equivalent Load Configuration (Figure 7.4) See Hi-Z Equivalent Load Configuration (Figure 7.5)	
Tc	0 × C to 100 × C
Vcc	5.0V ± 5%

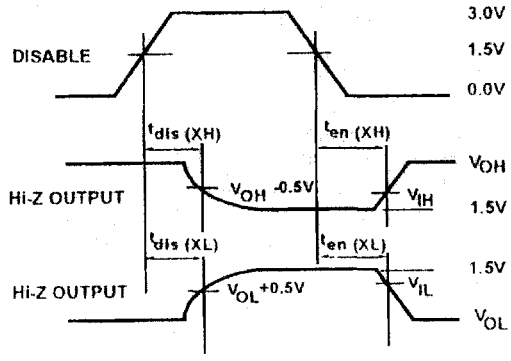
Figure 7.1 - AC Switching Diagram



3-10

SAMSUNG ELECTRONICS

Figure 7.2 - Hi - Z Ouput Swithching Waveform



SAMSUNG ELECTRONICS

Figure 7.3 - AC Output Equivalent Load Configuration

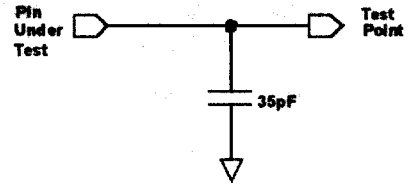


Figure 7.4 - Open Drain Equivalent Load Configuration

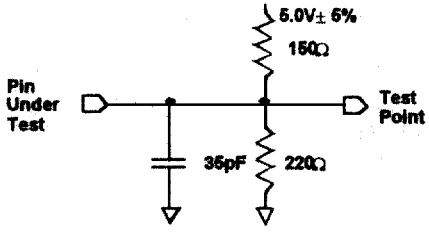


Figure 7.5 - HI - Z Output Equivalent Load Configuration

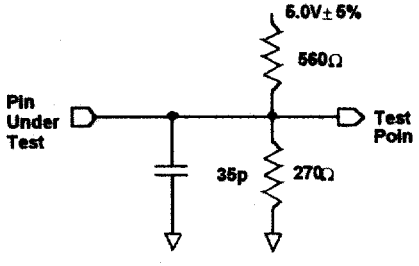


Figure 7.6 - Pull-up and Pull-down Circuits

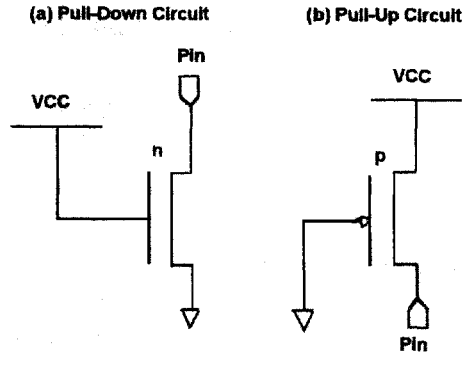


Figure 7.7 - HBUS State Transition Diagram

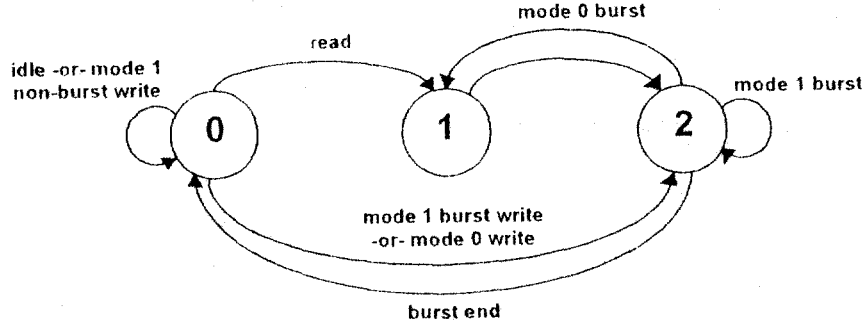


Table 7.5 - HBUS Timing Parameters

Identifier	Symbol	Description	KS0143CQ-60 Min	KS0143CQ-60 Max	KS0143CQ-40 Min	KS0143CQ-40 Max	Units	Notes
Host Bus Clock								
H1	tc (HCLK)	Host bus clock period			30		ns	2
H2	tw (HCLKH)	Host bus clock high pulse width			12		ns	
H3	twb (HCLKL)	Host bus clock low pulse width			12		ns	
	tr (HCLK)	Host bus clock rise time				4	ns	1
	tf (HCLK)	Host bus clock fall time				4	ns	1
Host Bus Control Strobe Timing								
H4	tsu (HS)	Host address strobe setup time			2		ns	
H5	th (HS)	Host address strobe hold time			6		ns	
H6	tsu (HM)	Host bus mode setup time			8		ns	
H7	th (HM)	Host bus mode hold time			2		ns	
H8	tsu (HW)	Host write enable setup time			8		ns	
H9	th (HW)	Host write enable hold time			2		ns	
H10	tsu (HBL)	Host burst low setup time			8		ns	
H11	th(HBL)	Host burst low hold time			2		ns	
H12	tsu (HBH)	Host burst high setup time			8		ns	
H13	th (HBR)	Host burst high hold time			2		ns	
H14	tpd (HRL)	Host ready low delay				21	ns	
H15	tv (HRL)	Host ready low valid			3		ns	
H16	tpd (HRH)	Host ready high delay				21	ns	
H17	tv (HRH)	Host ready high valid			3		ns	
Host Bus Address and Data Timing								
H18	tsu (HA)	Host address setup time			8		ns	
H19	th (HA)	Host address hold time			2		ns	
H20	ta (HD)	Host output data access time				19	ns	
H21	tv (HD)	Host output data valid time			5		ns	
H22	tv (HDT)	Host terminal output data valid time			5		ns	
H23	tsu (HD)	Host input data setup time			8		ns	
H24	th (HD)	Host input data hold time			2		ns	
H25	ten (HD)	Host clock to host data bus driven				25	ns	1
H26	tdis (HD)	Host clock to host data bus tri-state				25	ns	1
H27	ten (HE)	Host bus enable to host data bus valid				35	ns	
H28	tdis (HE)	Host bus enable to host data bus tri-state				35	ns	
Host Interrupt Timing								
H29	tpd (HI)	Host interrupt delay				24	ns	
H30	tv (HI)	Host interrupt valid			5		ns	

Notes : 1. Parameter is guaranteed by design and characterization data, but not tested.

2. $t_{c(HCLK)}$ must be $\geq t_{c(PCLK)}$

Figure 7.8 - Non - Burst Host Read Cycle Waveform
(See Figure 7.7 for a diagram of the state transitions)

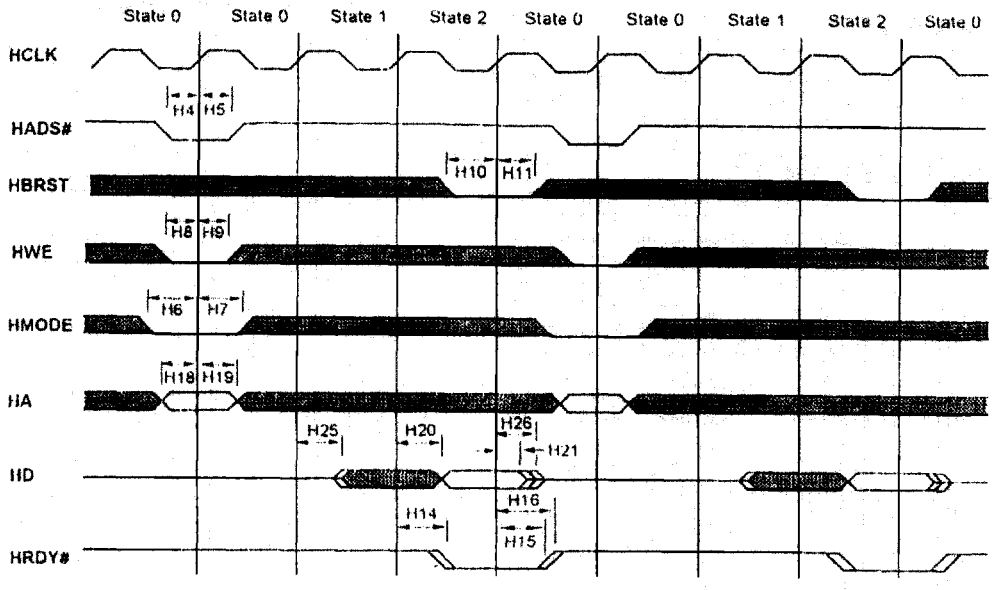


Figure 7.9 - Mode 0 Non-Burst Host Write Cycle Waveform
(See Figure 7.7 for a diagram of the state transitions)

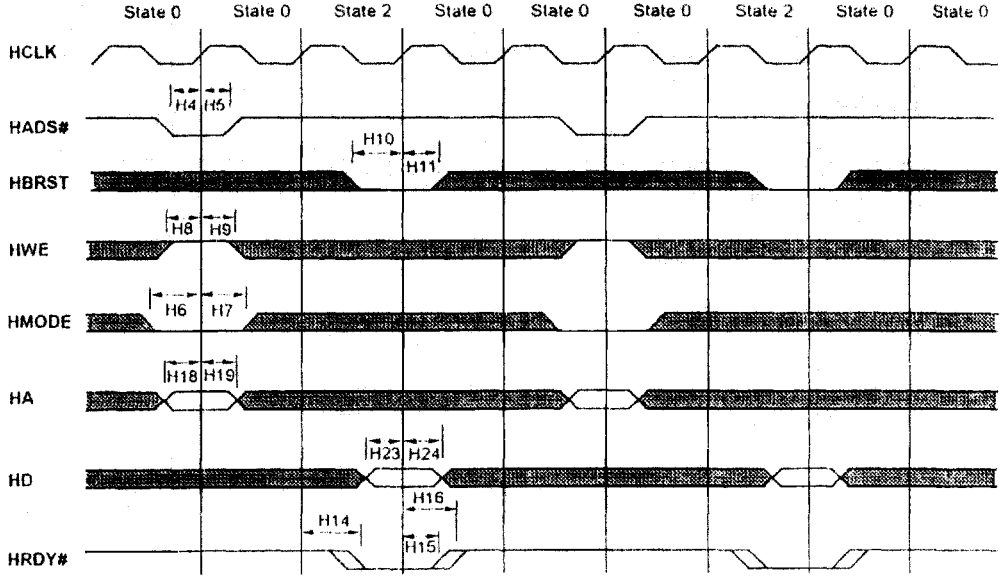


Figure 7.10 - Mode 1 Non Burst Host Write Waveform
(See Figure 7.7 for a diagram of the state transitions)

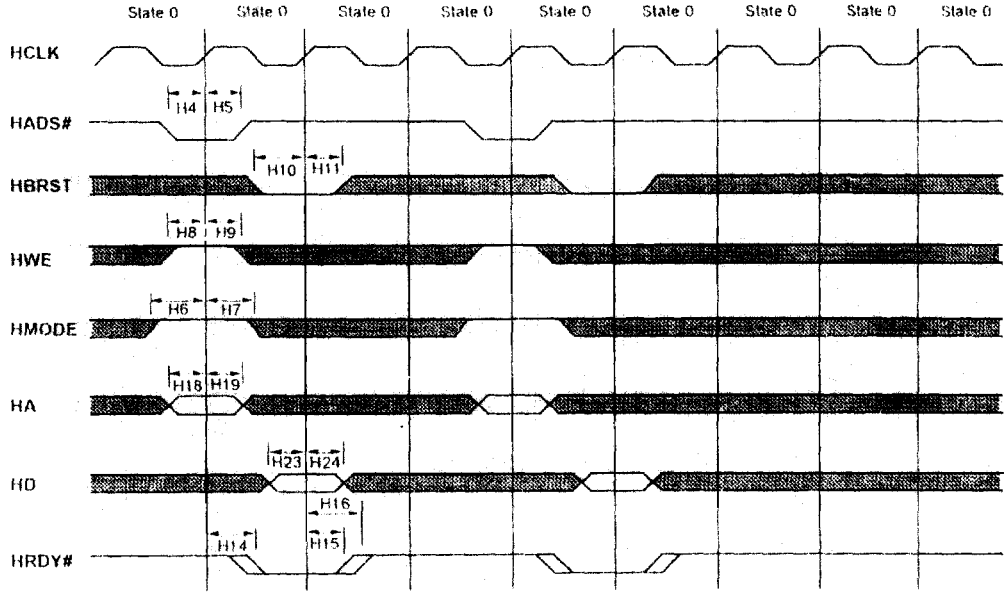
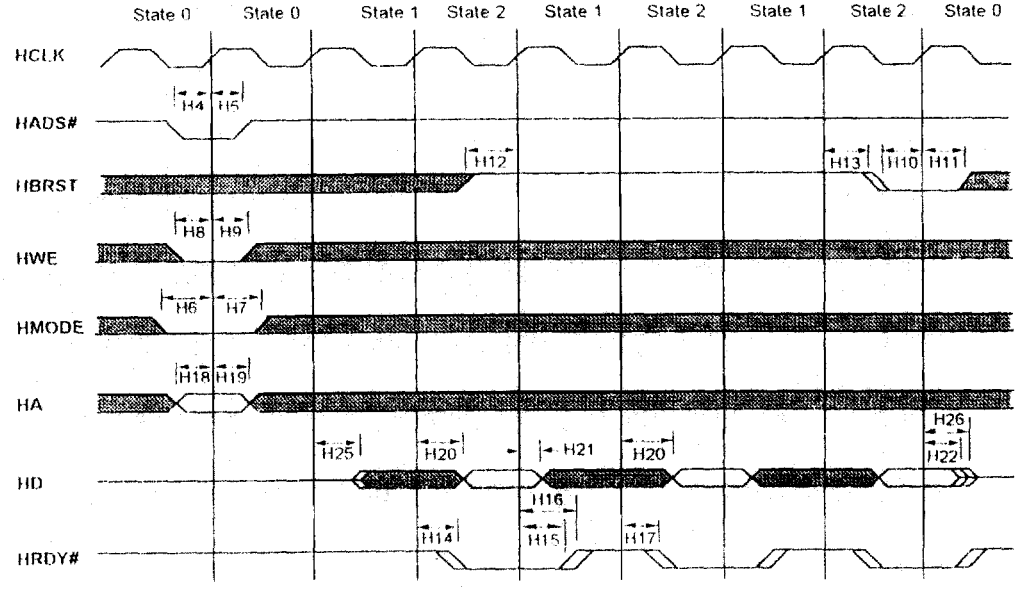


Figure 7.11 - Mode 0 Burst Host Read Cycle Waveform
(See Figure 7.7 for a diagram of the state transitions)



SAMSUNG ELECTRONICS

Figure 7.12 - Mode 1 Burst Host Read Cycle (BLENGTH = 2) Waveform
(See Figure 7.7 for a diagram of the state transitions)

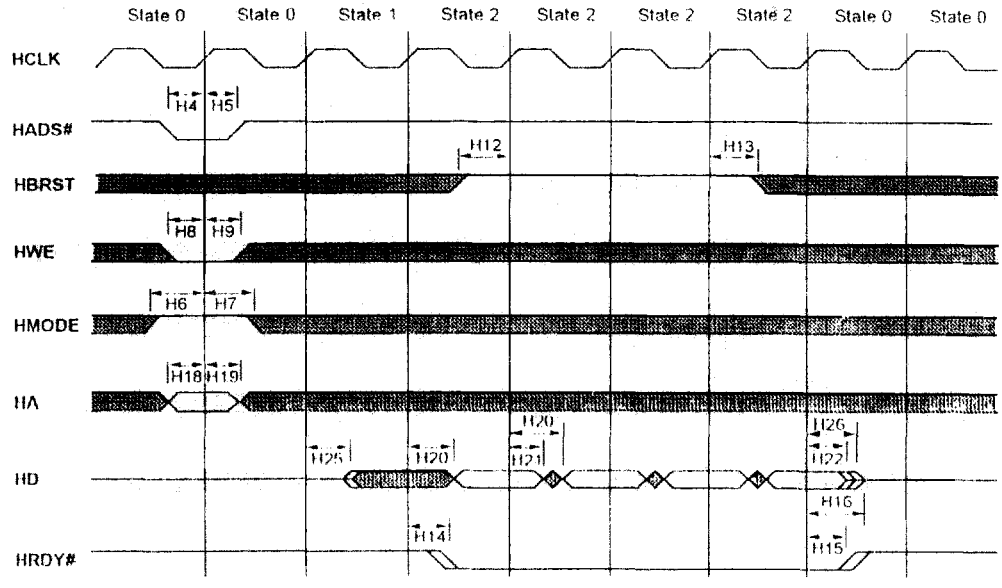
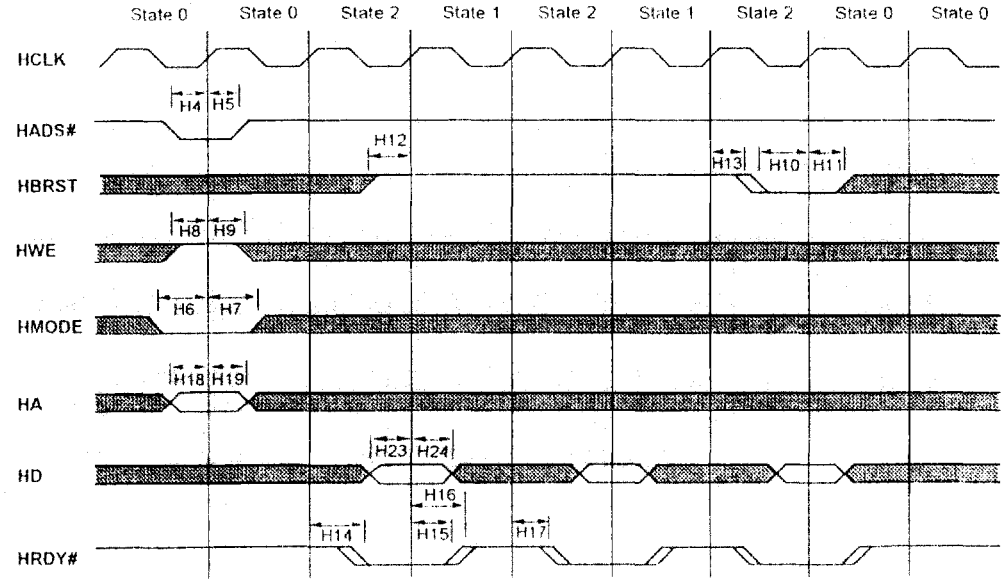


Figure 7.13 - Mode 0 Burst Host Write Cycle Waveform
(See Figure 7.7 for a diagram of the state transitions)



SAMSUNG ELECTRONICS

Figure 7.14 - Mode 1 Burst Host Write Cycle Waveform
(See Figure 7.7 for a diagram of the state transitions)

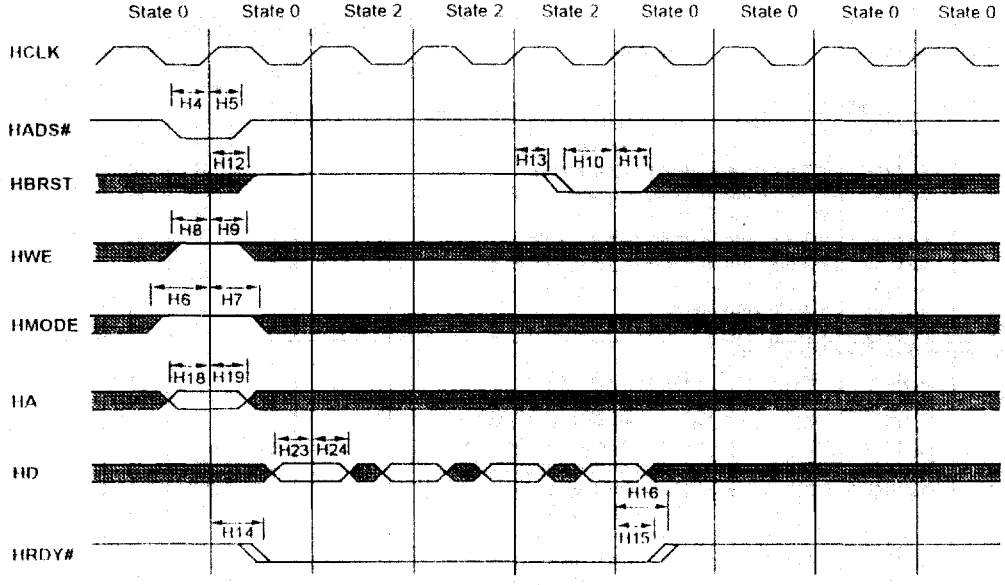
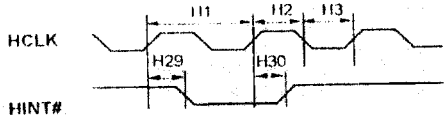


Figure 7.15 - Host Interrupt Waveform



NOTE: HINT# remains low until host processor clears interrupts by writing to INTSTAT register.

Figure 7.16 - Host Bus Disable Waveform

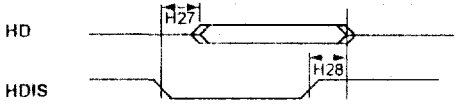


Table 7.6 - VBUS Timing Parameters

Identifier	Symbol	Description	KS0143CQ-60		KS0143CQ-40		Units	Notes
			Min	Max	Min	Max		
Video Memory Tri-State Timing								
V1	t_{en} (VM)	VCLK to video memroy select valid				28	ns	
V2	t_{dis} (VM)	VCLK to video memory select tri - state				28	ns	1
V3	t_{en} (VA)	VCLK to video memory address valid				30	ns	
V4	t_{dis} (VA)	VCLK to video memroy address tri - state				30	ns	1
V5	t_{en} (VD)	VCLK to video memory data driven				30	ns	
V6	t_{dis} (VD)	VCLK to video memory data tri - state				30	ns	1
V7	t_{ea} (VE)	VCLK to video memory output enable driven				26	ns	
V8	t_{en} (VW)	VCLK to video memroy select valid				26	ns	
V9	t_{en} (VB)	VDIS to output valid				35	ns	
V10	t_{dis} (VB/)	VDIS to output tri - state				35	ns	1
Video Memory Bus Page Mode Timing								
V11	t_v (VM)	Video memroy select valid after VCLK			5		ns	
V12	t_{pd} (VRL)	VCLK to video memroy row addr strobe low				20	ns	
V13	t_v (VRL)	Video memory row addr strobe low after VCLK			5		ns	
V14	t_{pd} (VCL)	VCLK to video memory column addr strobe low			5	20	ns	
V15	t_{pd} (VCH)	VCLK to video memory column addr strobe high			5	20	ns	
V16	t_{pd} (VA)	VCLK to video memory addr prop delay			7	22	ns	
V17	t_{pd} (VWL)	VCLK to video memory write enable				20	ns	
V18	t_v (VWL)	Video memory write enable after VCLK			5		ns	
V19	t_{pd} (VEL)	VCLK to video memory outut enable				20	ns	
V20	t_v (VEL)	Video memory output enable after VCLK			5		ns	
V21	t_{pd} (VD)	VCLK to video memory data outptu prop delay			7	22	ns	
V22	t_{su} (VSH)	VCAl# to VCLK setup time			5		ns	
V23	t_h (VSH)	VCAl# to VCLK hold time			4		ns	
V24	t_w (VSL)	VCASl# pulse width low			16		ns	
V25	t_{su} (VDS)	VnD to VCASl# high setup time			6		ns	
V26	t_h (VDS)	VnD to VCASl# high hold time			0		ns	
V27	t_{su} (VSL)	VCASl# to VCLK setup time			7		ns	1
V28	t_h (VSL)	VCASl# to VCLK hold time			2		ns	1
V29	t_{su} (VDC)	VnD to VCLK setup time (VCASl# low)			5		ns	
V30	t_h (VDC)	VnD to VCLK hold time (VCASl# low)			5		ns	
Video Memory Bus Strobe Timing								
V31	t_w (VCL)	VCASO# low width		T_8-3			ns	3
V32	t_w (VCH)	VCASO# high width		T_4-T_8-3			ns	3
V33	t_c (VC)	VCASO# cycle time		T_4-1			ns	3
V34	t_d (VRLCL)	VRAS# low to VCASO# low		T_3-3			ns	3
V35	t_d (VCLCH)	VCASO# low to VCASO# high		T_8-3			ns	3
V36	t_d (VRLCH)	VRAS# low to VCASO# high		T_3+T_8-3			ns	3
V37	t_d (VCLRL)	VCASO# low to VRAS# low (refresh cycle)		T_{10}-3			ns	3
V38	t_d (VRLCLr)	VCASO# low after VRAS# low (refresh cycle)		T_{11}-3			ns	3
V39	tw (VCLt)	VCASl# low width (transfer cycle)		T_{13}+T_{14}-3				

Table 7.6 - VBUS Timing Parameters (Continued)

Identifier	Symbol	Description	KS0143CQ-60 Min	KS0143CQ-60 Max	KS0143CQ-40 Min	KS0143CQ-40 Max	Units	Notes
Video Memory Bus Clock								
V40	t_c (VCLK)	Video memory bus clock period			25		ns	2
V41	t_w (VCLKH)	Video memory bus clock high pulse width			10		ns	
V42	t_w (VCLK)	Video memory bus clock low pulse width			10		ns	
	t_r (VCLKH)	Video memory bus clock rise time				4	ns	1
	t_f (VCLKH)	Video memory bus clock fall time				4	ns	1
Video Memory Bus Arbitration								
V43	t_{su} (VQ)	Video mem bus request input setup time			3		ns	
V44	t_h (VQ)	Vleo mem bus request input hold time			4		ns	
V45	t_{su} (VG)	Video mem bus grant input setup time			3		ns	
V46	t_h (VCL)	Video mem bus grant input hold time			4		ns	
V47	t_{pd} (VQ)	Video mem bus request output prop delay			5	20	ns	
V48	t_{pd} (VG)	Video mem bus grant output prop delay			5	20	ns	
Video Memory Serial Command Bus								
V49	t_{su} (VT)	Video mem serial command load setup time			8		ns	
V50	t_h (VT)	Video mem serial command hold time			2		ns	
V51	t_{su} (VN)	Video mem serial tranefer command setu time			8		ns	
V52	t_h (VN)	Video mem serial command hold time			2		ns	

Notes: 1. Parameter is guaranteed by design and characterization data, but not tested.
2. tc(VCLK) must be S t_c(PCLK)
3. $T_a = t_c$(VCLK) × VCNTn

3-10

Figure 7.17 - Page Mode Read Cycle Waveforms

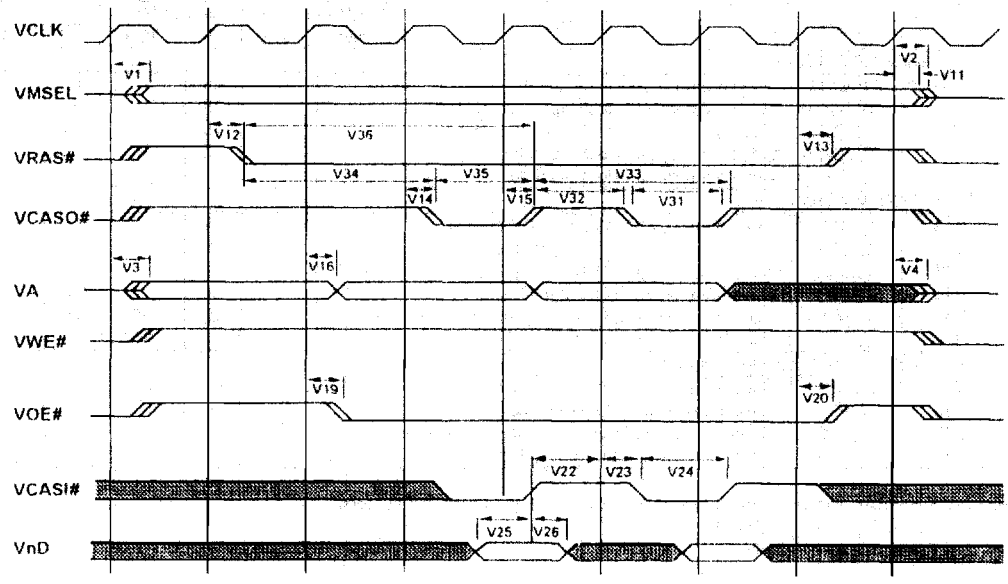


Figure 7.18 - Hyper Page Mode Read Cycle Waveforms

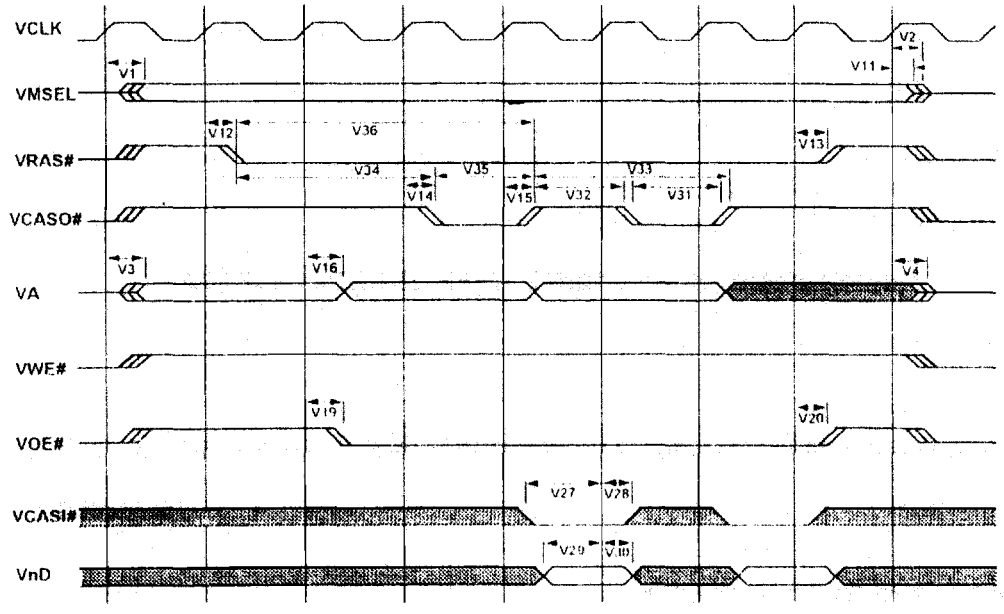
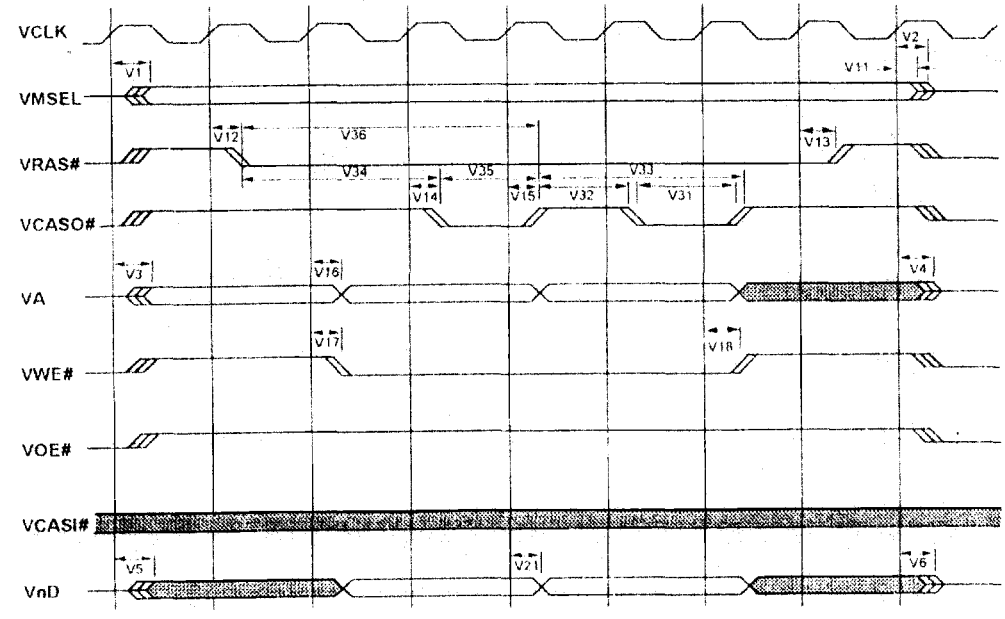


Figure 7.19 - Page Mode Write Cycle Waveforms



3-10

Figure 7.20 - CAS Before RAS Refresh Waveforms

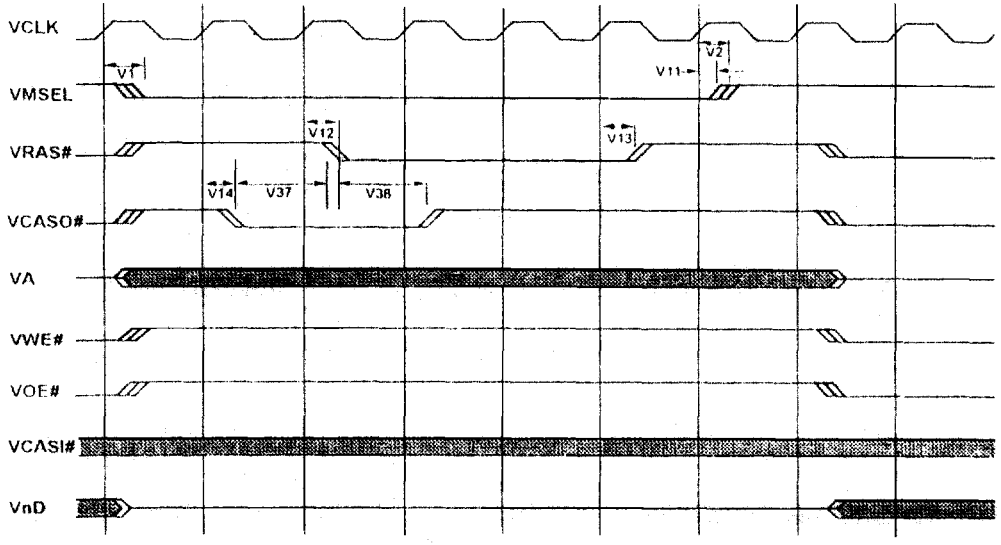


Figure 7.21 - SAM to DRAM Transfer Waveforms

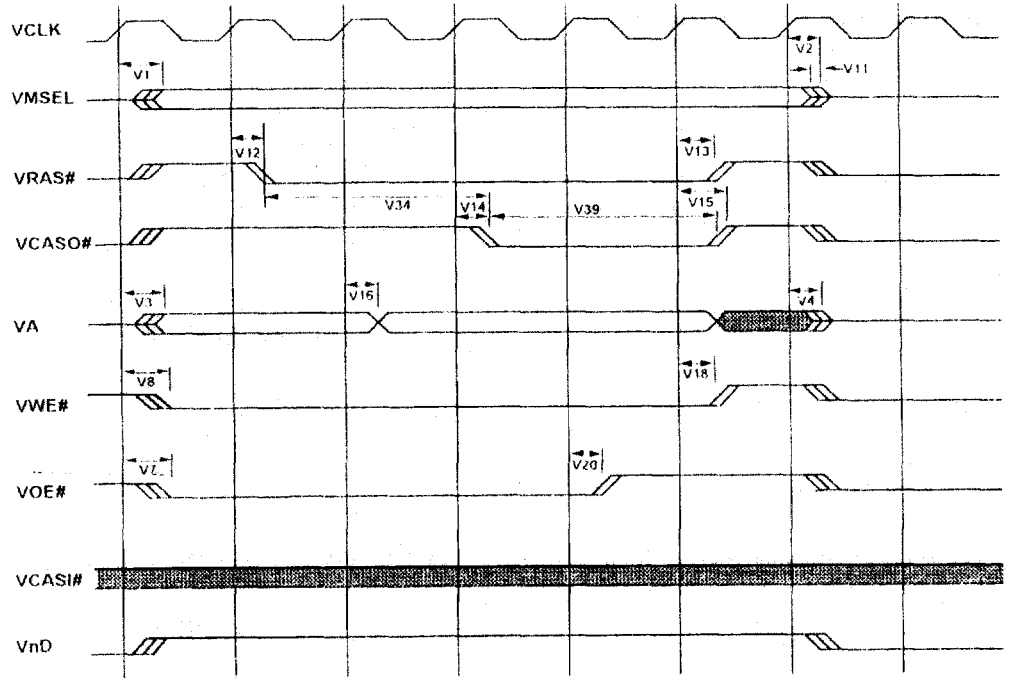


Figure 7.22 - DRAM to SAM Transfer Waveforms

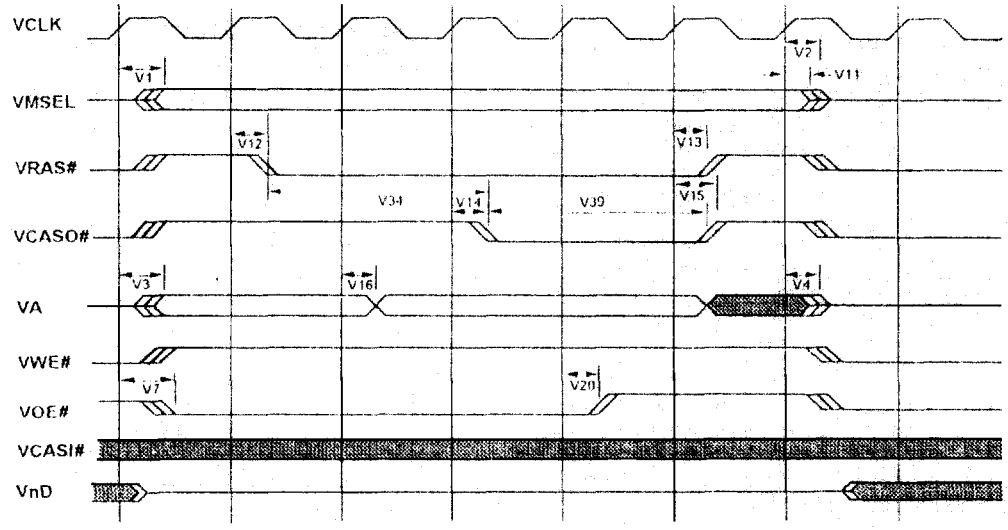


Figure 7.23 - Video Bus Arbitration Waveform

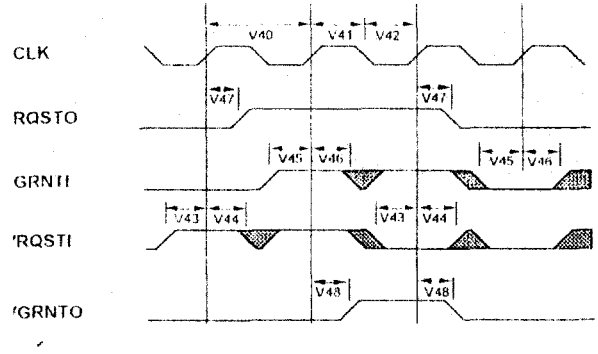


Figure 7.24 - Video Memory Serial Command Bus Waveform

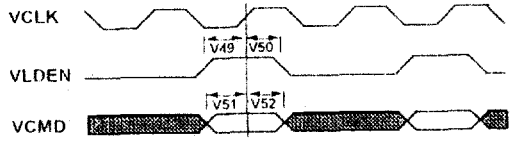
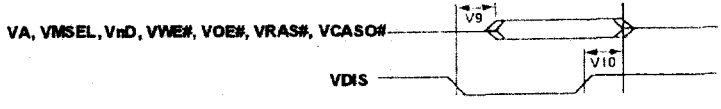


Figure 7.25 - Video Bus Disable Waveform



3-10

Table 7.7 - XBUS Timing Parameters

Identifier	Symbol	Description	KS0143CQ-50 Min	KS0143CQ-50 Max	KS0143CQ-40 Min	KS0143CQ-40 Max	Units	Notes
Auxiliary Bus Clock								
X1	t_c(XCLK)	Auxiliary bus clock period			25		ns	2
X2	t_w(XCLKH)	Auxiliary bus clock high pulse width			10		ns	
X3	t_w(XCLKL)	Auxiliary bus clock low pulse width			10		ns	
	t_r(XCLK)	Auxiliary bus clock rise time				4	ns	1
	t_f(XCLK)	Auxiliary bus clock fall time				4	ns	1
Auxiliary Bus Arbitration								
X4	t_{su}(XQ)	Auxiliary request setup time			3		ns	
X5	t_h(XQ)	Auxiliary request hold time			5		ns	
Auxiliary Bus Strobe Timing								
X6	tpd(XSL)	Auxiliary address strobe delay				23	ns	
X7	tv(XSL)	Auxiliary address strobe valid			9		ns	
X8	tpd(XWL)	Auxiliary write enable delay				23	ns	
X9	tv(XWL)	Auxiliary write enable valid			9		ns	
X10	tpd(XEL)	Auxiliary output enable low delay			9	23	ns	
X11	tpd(XEH)	Auxiliary output enable high delay			9	23	ns	
X12	tsu(XR)	Auxiliary processor ready setup time			3		ns	
X13	th(XR)	Auxiliary processor ready hold time			5		ns	
Auxiliary Bus Data Transfer Timing								
X14	t_a(XA)	Auxiliary address access time				24	ns	
X15	t_v(XA)	Auxiliary address valid time			8		ns	
X16	t_a(XD)	Auxiliary output data access time				24	ns	
X17	t_v(XD)	Auxiliary output data valid time			8		ns	
X18	t_{su}(XD)	Auxiliary input data setup time			0		ns	
X19	t_h(XD)	Auxiliary input data hold time			8		ns	
X20	t_{en}(XA)	Auxiliary bus enable to address valid				25	ns	
X21	t_{dis}(XA)	Auxiliary bus disable to address tri - state				25	ns	1
X22	t_{en}(XD)	Auxiliary bus enable to data valid				25	ns	
X23	t_{dis}(XD)	Auxiliary bus disable to data tri - state				25	ns	1
X24	t_{pd}(XD)	Auxiliary output data display			8	24	ns	

Notes: 1. Parameter is guaranteed by design and characterization data, but not tested.

2. t_c(XCLK) must be $\geq t_c$(PCLK)

Figure 7.26 - XBUS Read Cycle Waveforms (start)

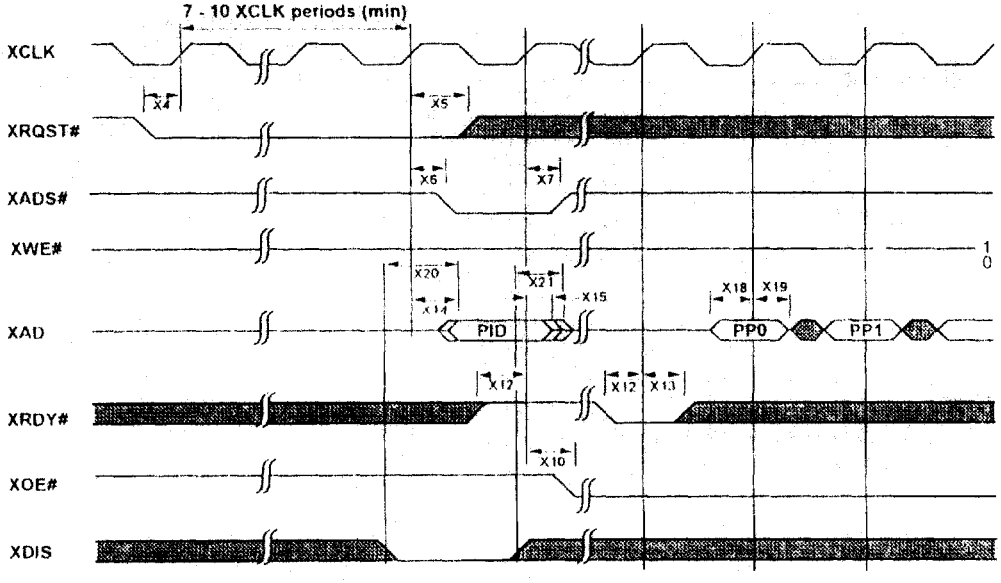


Figure 7.27 - XBUS Read Cycle Waveforms (end)

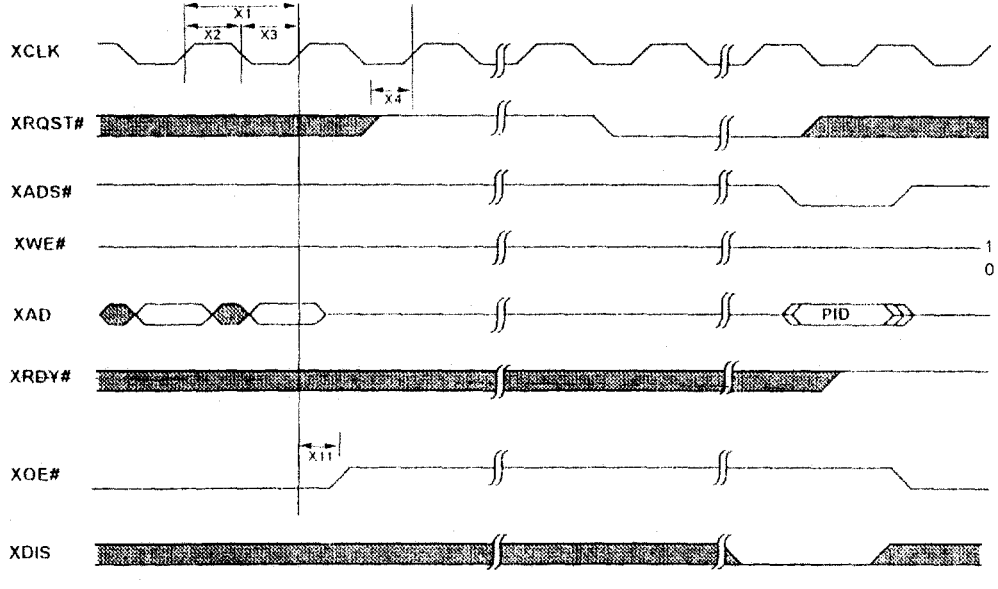


Table 7.8 - Test Bus Timing Parameters

Identifier	Symbol	Description	KS0143CQ - 50 Min	KS0143CQ - 50 Max	KS0143CQ - 40 Min	KS0143CQ - 40 Max	Unit	Notes
Test Bus Clock								
T1	tc(TCLK)	Test bus clock period			100		ns	
T2	tw(TCLKH)	Test bus clock high pulse width			40		ns	
T3	tw(TCLKL)	Test bus clock low pulse width			40		ns	
	tr(TCLK)	Test bus clock rise time				4	ns	1
	tf(TCLK)	Test bus clock fall time				4	ns	1
Test Bus Timing								
T4	tsu(TI)	TMS, TDI setup time			10		ns	2
T5	th(TI)	TMS, TDI hold time			10		ns	2
T6	ten(TDO)	Test clock low to TDO driven			4	30	ns	3
T7	tpd(TDO)	Test clock low to TDO delay			4	30	ns	3
T8	tdis(TDO)	Test clock low or reset pulse low to TDO tri - state			4	30	ns	1,3
T9	tw(TRSTL)	Test reset low pulse width			100		ns	

Notes:

1. Parameter is guaranteed by design and characterization data, but not tested.
2. Input setup/hold times are with respect to the positive edge of TCLK.
3. Output delay times are with respect to the negative edge of TCLK.

Figure 7.28 - XBUS Write Cycle Waveform

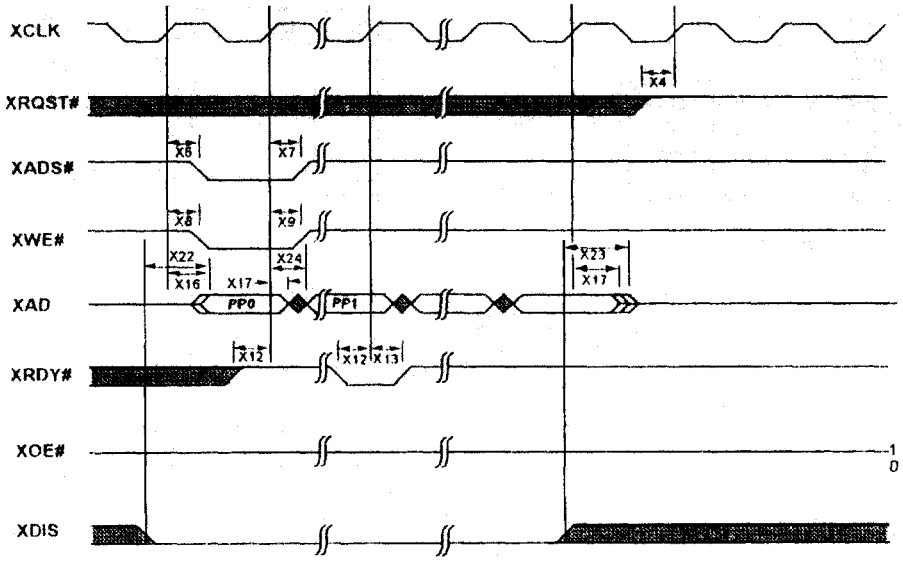


Figure 7.29 - Test Bus Timing Waveforms

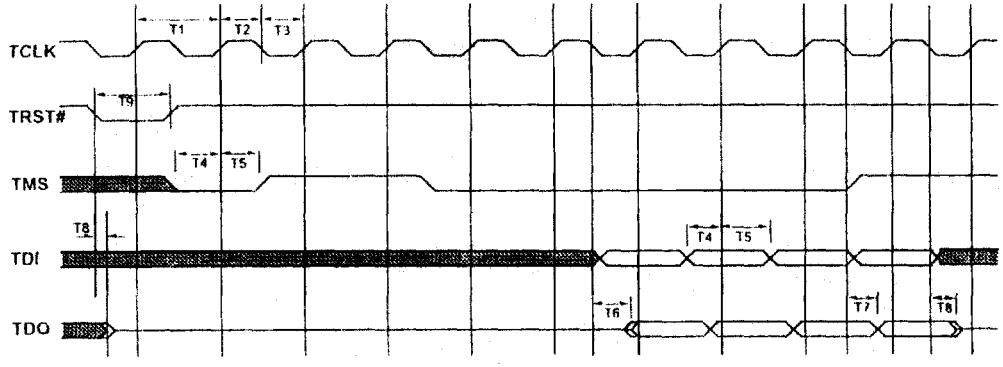
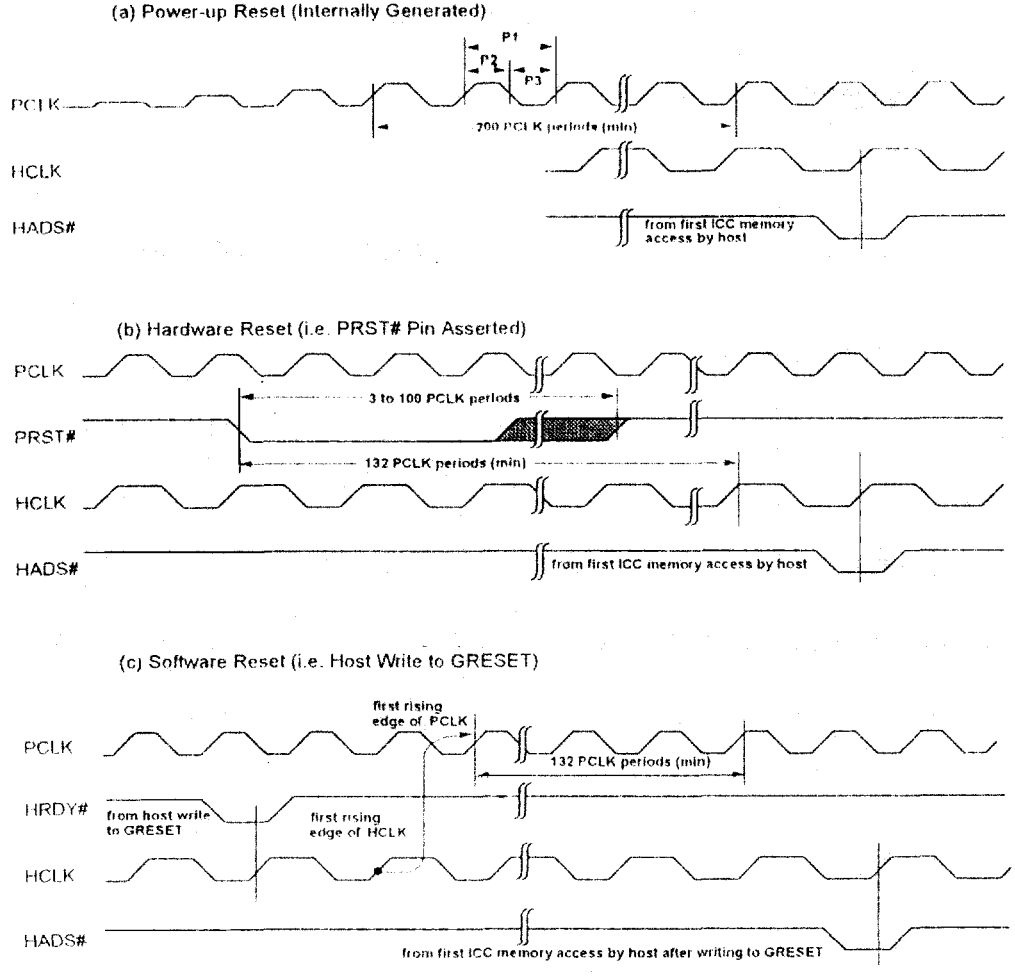


Table 7.9 - Processor Clock (PCLK) Timing Parameters

Identifier	Symbol	Description	KS0143CQ-50 Min	KS0143CQ-50 Max	KS0143CQ-40 Min	KS0143CQ-40 Max	Units	Notes
Processor Clock								
P1	t_c(PCLK)	Processor clock period			25		ns	
P2	t_w(PCLKH)	Processor clock high pulse width			10		ns	
P3	t_w(PCLKL)	Processor clock low pulse width			10		ns	
	t_r(PCLK)	Processor clock rise time				4	ns	1
	t_f(PCLK)	Processor clock fall time				4	ns	1

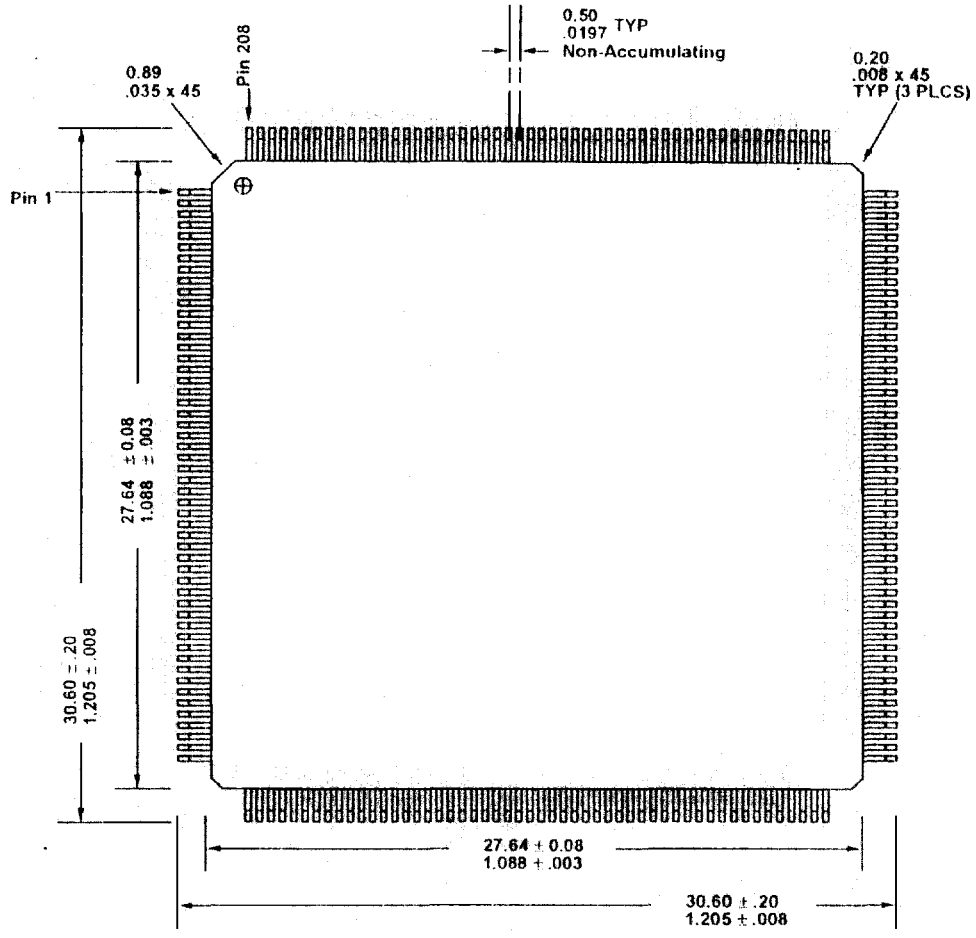
Notes : 1 = Parameter is guaranteed by design and characterization, data, but not tested.

Figure 7.30 - Reset Waveforms



SECTION 8 -MECHANICAL SPECIFICATIONS

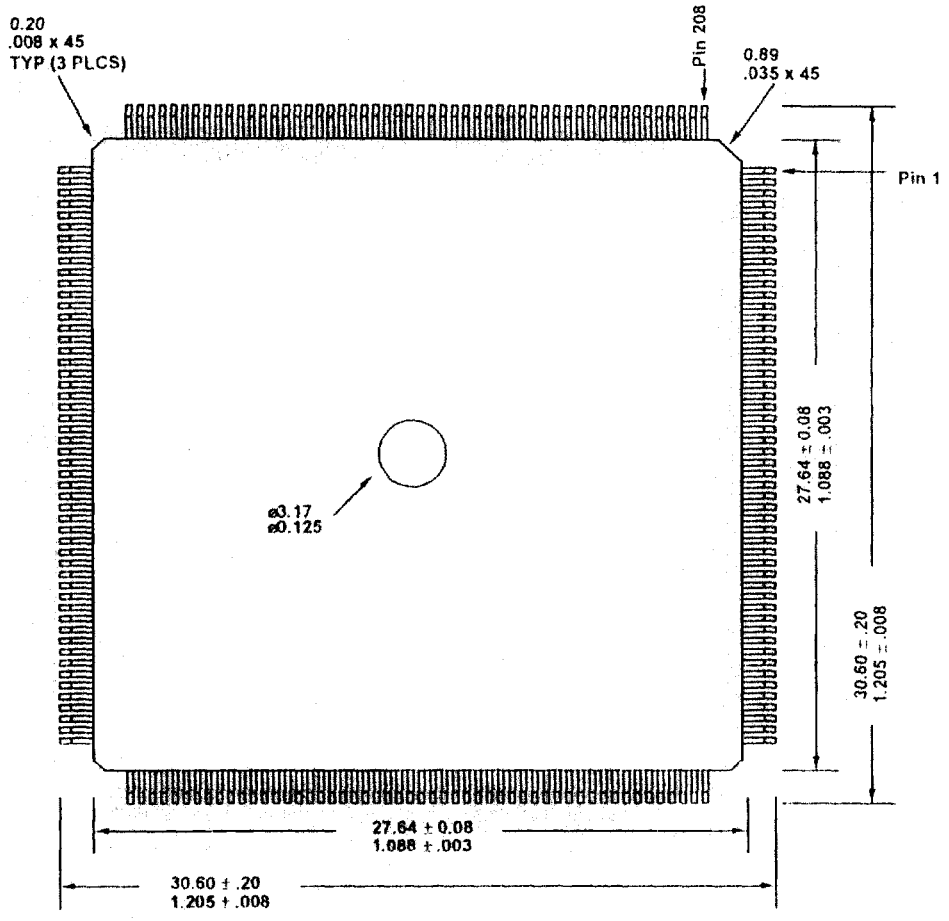
Figure 8.1 -Package Dimensions (top view, cavity down)



Notes:

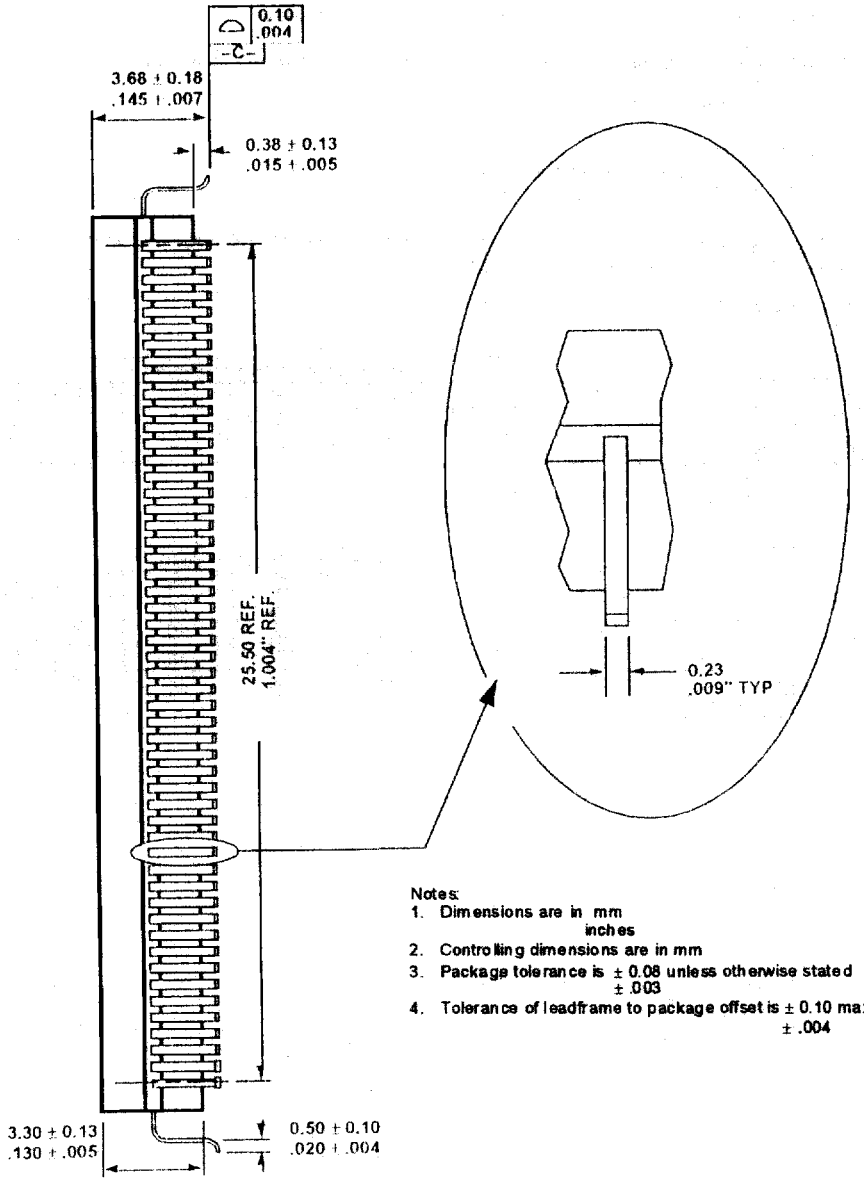
1. Dimensions are in mm inches
2. Controlling dimensions are in mm
3. Package tolerance is ± 0.08 unless otherwise stated ± .003
4. Tolerance of leadframe to package offset is ± 0.10 max ± .004

Figure 8.2 - Package Dimensions (bottom view, cavity down)



Notes:
1.0Dimensions are in mm inches
2.0Controlling dimensions are in mm
3.0Package tolerance is ± 0.08 unless otherwise stated ± .003
4.0Tolerance of leadframe to package offset is ± 0.10 max ± .004

Figure 8.3 - Package Dimensions (Side View, cavity down)



Notes:
1. Dimensions are in mm
   inches
2. Controlling dimensions are in mm
3. Package tolerance is ± 0.08 unless otherwise stated
   ± .003
4. Tolerance of leadframe to package offset is ± 0.10 max
   ± .004

Package Thermal Specifications

The KS0143 / KS0144KS0143 / KS0144KS0143 is specified for operation when the case temperature (Tc) is within the range of 0× C to 100× C.

The case temperature is measured at the center of the top surface of the package.

The maximum allowable ambient temperature (TA) can then be calculated using the following equation:

TA [max] = TC [max] - (P * (q JA - q JC))

where: P = Maximum Power Dissipation
q JC = Thermal Resistance from junction to case
q JA = Thermal Resistance from junction to ambient

The following tables give the thermal resistance parameters and the maximum allowable ambient temperatures for the 208 MQUAD™ package under various airflow conditions

Table 8.3 - 208 MQUAD Thermal Resistance (×C/Watt)

Parameter	Airflow (Linear Feet Per Minute)						
	0	100	200	400	600	800	1000
θ JC (junction -to-case)	3	3	3	3	3	3	3
θ JA(junction -to-ambient)	21	18	16	12	9	7	6

Table 8.4 - Maximum Ambient Temperature at Various Operating Conditions (×C)

Parameter	Airflow (Linear Feet Per Minute)						
	0	100	200	400	600	800	1000
TA		10	22	46	64	76	82
(without heat sink)	10	25	35	55	70	80	85
TA	TBD	TBD	TBD	TBD	TBD	TBD	TBD
(with heat sink) **	TBD	TBD	TBD	TBD	TBD	TBD	TBD

** = Heat sink to be determined