1.0 INTRODUCTION

The AHA5140 is a single-chip Reed-Solomon error corrector, buffer manager and QIC® tape formatter.

The device provides a frame-level control interface to free the microprocessor for other tasks. Individual block-level transfers are handled in hardware. During read-while-write, block rewrites are automatically scheduled. During reads incoming blocks are automatically directed to the appropriate DRAM location. In addition to frame-level control, the microprocessor can still take direct control of transfers to handle unusual or proprietary operations.

Host data transfers are through DMA connection to an external controller. DMA rates support up to ATAPI PIO mode 3 (11.1 MB/s).

The tape formatter has read/write compatibility with QIC-3095 and read compatibility with QIC-80, QIC-3010 and QIC-3020. In addition, several features within the IC allow reading and writing of the single channel TRAVAN-5 format.

1.1 FEATURES

HOST INTERFACE:

- Connects to external SCSI or ATAPI chips
- Host data rate of up to 11.1 MB/sec while maintaining maximum microprocessor and tape data rates and DRAM refresh

TAPE CONTROL AND FORMATS:

- Frame-level control of tape read and write operation
- Optional block-level control during unusual or proprietary operations
- Fully meets QIC-3095 standard including read-while-write support
- Reads QIC-80, 3010, 3020 and 3095 formats; Writes QIC-3095 format
- Reads and writes TRAVAN-5 formats
- Output either NRZI or pulse data
- Connects to popular read/write channel ICs
- Up to 1.66 MB/sec tape channel rate

REED-SOLOMON ECC:

- 6 or 10 byte redundancy to support QIC-3095 and TRAVAN-5
- Interleave-level control of ECC encoding operation
- ECC decoding based on erasures detected by CRC
- ECC erasures stored in DRAM for off-line decoding

MICROPROCESSOR INTERFACE:

- Direct connection to Intel 80188, 80186, 80196 microprocessors
- High-speed block copy support of DRAM
- SRAM Emulation mode eliminates separate SRAM in the system

DRAM INTERFACE

- Supports 60 ns 256K through 4MB DRAM
- EDO DRAM supported but not required

GENERAL:

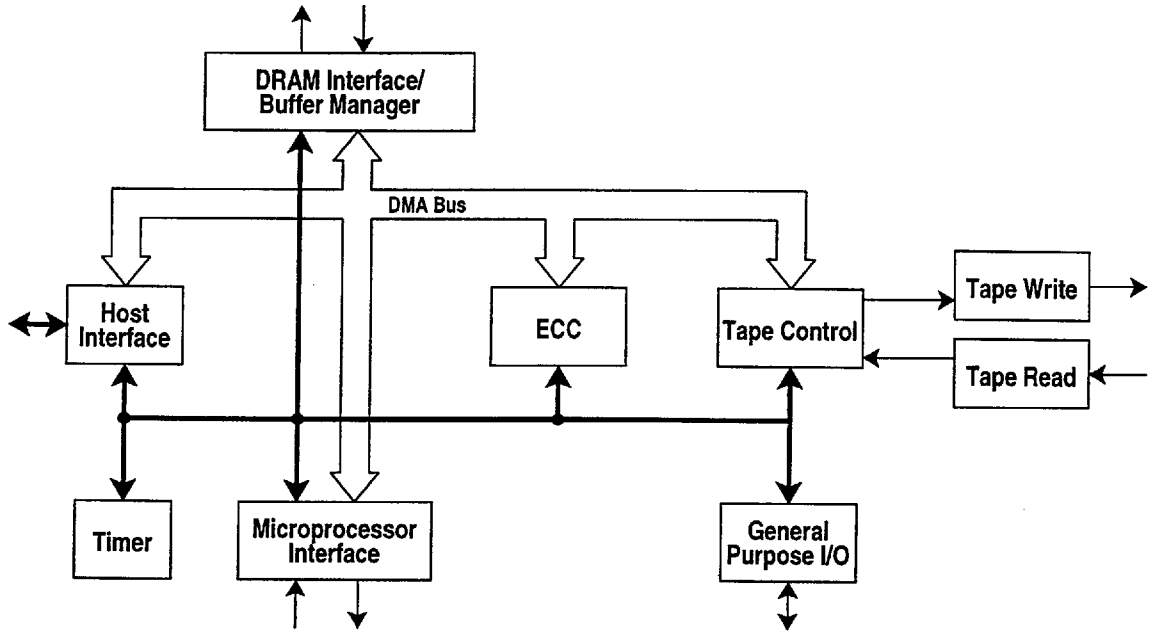
- General purpose, programmable counter/timer
- Programmable clock generation for system, tape write channel and data separator reference
- Tristatable outputs to facilitate system testing
- 8-bit general purpose I/O port with programmable direction and interrupt capability
- 5.0 volts supply

1.2 DOCUMENT CONVENTIONS

The following typographic conventions were followed in the generation of this specification.

- **SIGNALS OR PINS**
- *Register Names*
- **INDIVIDUAL REGISTER BITS**
- **Interrupts**
- ***Commands***

Figure 1: Block Diagram



2.0 MICROPROCESSOR INTERFACE

The AHA5140 provides a generic 8-bit multiplexed address/data port compatible with common microcontrollers. All internal register reads and writes go through this port. The microprocessor interface is also responsible for two types of transfers with the DRAM buffer. Firstly the processor can indirectly access the buffer through an address and data register. Secondly the chip can emulate a SRAM using part of the buffer. The following sections discuss all three responsibilities of the microprocessor interface.

2.1 REGISTER READS AND WRITES

Register accesses follow a standard multiplexed address/data flow. An access is initiated by driving **UPCSN** low. The address on **UPAD[7:0]** is latched on the falling edge of the address latch enable **UPALE**. **UPRDN** and **UPWRN** indicate reads and writes respectively. The microprocessor wait line, **UPRDY**, is driven active as soon as **UPCSN** goes low. **UPRDY** releases synchronous to **SYSCLK**.

The microprocessor can also use fixed internal wait states for register accesses. The only exception is the *DRAM Data* register with wait generation enabled. See the "Accessing DRAM Buffer" section below.

The internal register map takes up 128 address locations. Therefore **UPCSN** must be a decode of no less than seven bits of address.

Some 16-bit processors do not fully support operation with 8-bit ports. These processors can only write even addresses to the least significant byte of the data bus. To support such processors, the AHA5140 can shift the address bus right 1 bit. Then the external firmware sees adjacent registers at addresses 0,2,4,...etc. This feature is enabled in the *System Configuration* register.

2.2 SRAM EMULATION

The AHA5140 emulates a 64Kx8 SRAM. The data is actually stored in the first 64K of the DRAM buffer. A separate chip select, **UPSCSN** is provided to distinguish these accesses from the internal register.

UPRDY will always assert after **UPSCSN** asserts. In the case of a read, **UPRDY** will remain active until the DRAM arbitration occurs and the read data is present on **UPAD[7:0]**. In the case of a write, **UPRDY** will remain active until SRAM controller has arbitrated for a DRAM access. Waiting a write prevents coherency problems with a write followed by a read to the same address. It also prevents the next read from having to wait for a previous write to finish.

For SRAM emulation accesses the minimum bandwidth to the DRAM buffer is 1 MB/s. This occurs when Tape and Host are active.

If SRAM emulation is not desired, **UPSCSN** should be tied high and **UPA[15:8]** tied high or low.

2.3 ACCESSING DRAM BUFFER

The DRAM buffer is accessed through an address pointer and data register. To read or write a range of memory, the microprocessor writes the address pointer and then repeatedly reads or writes the data register. DRAM arbitrations are variable in length so wait states or polling are required to pace the transfers.

2.3.1 DRAM WRITES

To write a location the microprocessor programs the desired address into *DRAM Address* from least to most significant byte with bit 23 clear. Then the processor writes the data byte to *DRAM Data*. This causes the controller to schedule a DRAM access and leaves the microcontroller free to perform other tasks. After completing the write, *DRAM Address* is automatically incremented. A subsequent write to *DRAM Data* will go to the next location in DRAM. Hence the microprocessor only needs to write the address once to write a range of memory.

While the buffer interface is busy writing a DRAM location, the microprocessor must not write *DRAM Address* or *DRAM Data*. In all modes, the **DBUSY** bit in *DRAM Configuration* will be set if it is not safe to write. The microprocessor can poll this register and wait for **DBUSY** to clear before writing either register. In addition, automatic wait state insertion can be enabled via **DWTE** in *DRAM Configuration*. The AHA5140 will then use **UPRDY** to insert wait states during write cycles to ensure any pending DRAM access has finished before completing a write to *DRAM Address* or *DRAM Data*.

2.3.2 DRAM READ

To read a location, the microprocessor programs the desired address into *DRAM Address* from least to most significant byte with bit 23 set. This initiates a DRAM access to prefetch the data into *DRAM Data*. The processor then reads the data from *DRAM Data*. After the read, *DRAM Address* is automatically incremented and a prefetch of the new address initiated. Hence the microprocessor only needs to write the address once to read a range of memory.

While the buffer interface is busy prefetching a DRAM location, the microprocessor must not read *DRAM Data*. Also if the buffer interface is still completing a previous access, the microprocessor must not yet write *DRAM Address*. In all modes, the **DBUSY** bit in *DRAM Configuration* will be set if it is not safe to read *DRAM Data* or write *DRAM Address*. The microprocessor can poll this register and wait for **DBUSY** to clear before accessing either register. In addition, automatic wait state insertion can be enabled via **DWTE** in *DRAM Configuration*. The AHA5140 will then use **UPRDY** to insert wait states during read or write cycles to ensure any pending access completes before updating *DRAM Address* or reading *DRAM Data*.

2.3.3 DATA BYTE SKIPPING

To facilitate reading and writing QIC-3095 and TRAVAN-5 control bytes, AHA5140 can modify the automatic-increment feature of *DRAM Address* to skip over data bytes in a frame. When **DDBS** in *DRAM Configuration* is set, *DRAM Address* will increment by 512 instead of by 1 after each eight bytes. This allows the microprocessor to write all of the control bytes for a frame with only one write to *DRAM Address*. The required byte counter is reset each time the most significant byte of *DRAM Address* is written.

2.3.4 BANDWIDTH

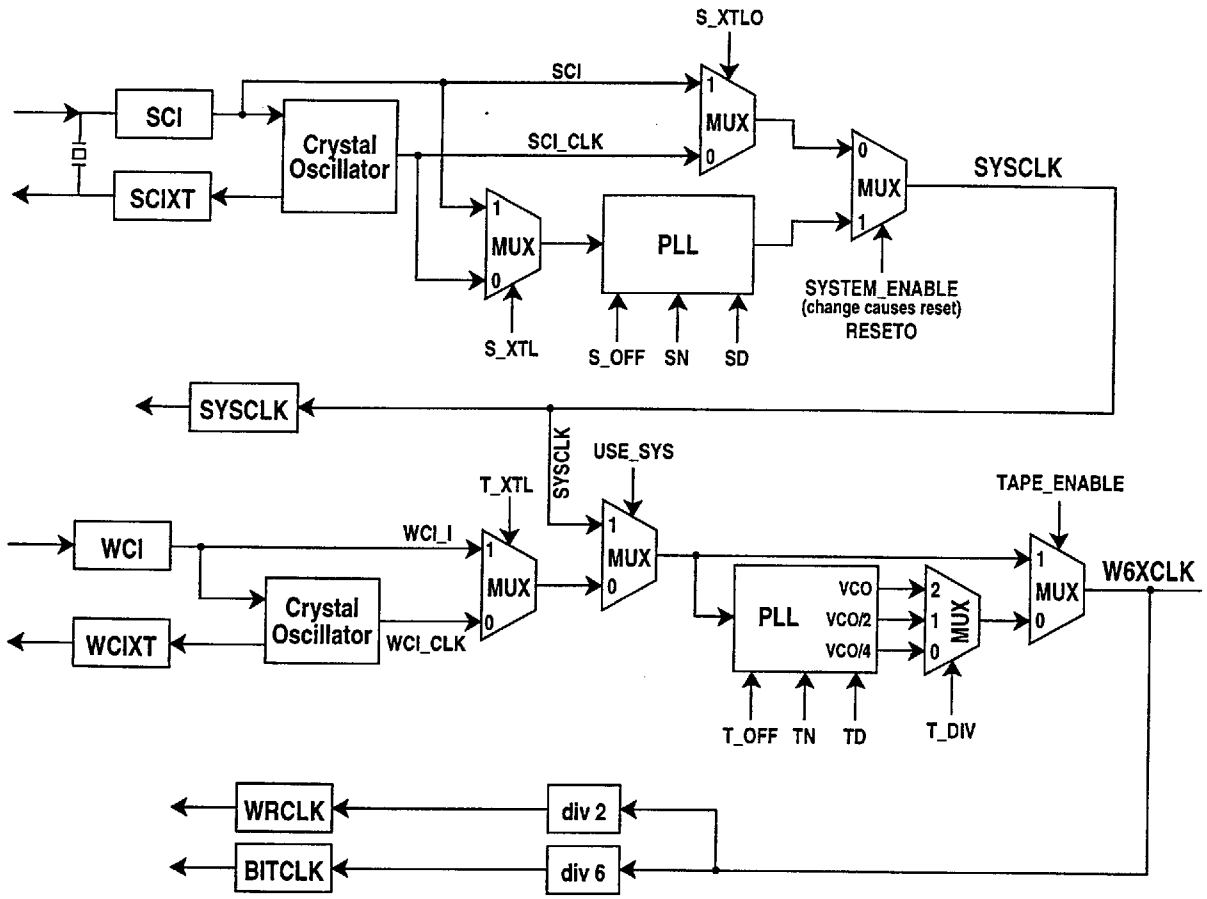
For indirect accesses the minimum DRAM buffer bandwidth is 1 MB/s. This occurs when Tape and Host are active.

3.0 CLOCKS

The AHA5140 has two on-board clock generation phase locked loops (PLLs). One of the PLLs generates the system clock, **SYSCLK**, and the other generates the tape clock. Figure 2 shows a block diagram of the clock generation on the AHA5140. This clock structure allows for considerable flexibility in the system design.

The **System PLL** is driven from a crystal oscillator which can either be connected to an external crystal or driven directly with an external clock source. The source type and frequency is programmed in *System Denominator*, *System Numerator* and *Clock Configuration* registers. The output of this clock circuit generates **SYSCLK**. **SYSCLK** can be used to clock other ICs in the system. However, since **SYSCLK** will glitch during clock configuration, those chips using it should be reset with **RESETO**.

Figure 2: AHA5140 Clock Generation Block Diagram



The clock network for the tape interface should be fed from either a crystal or an external clock source. It can also feed with **SYSCLK**. However, this is not recommended because of the compounded clock jitter that results from feeding one PLL from another. The **Tape PLL** is programmable with *Tape Numerator*, *Tape Denominator* and *Clock Configuration* registers to generate a wide range of frequencies.

Refer to AHA5140 Application Note or contact AHA for the register programming for particular applications.

4.0 GENERAL I/O PORT

The AHA5140 provides a general purpose 8-bit I/O port for communicating with the drive mechanism or other parts of the system. The **GIO[7:0]** pins can always be read in the *GIO Data* register. All of the pins can become outputs. The direction is programmable in pairs of bits by *GIO Control*. Each pin also acts as an interrupt. Interrupts can be edge or level sensitive. Interrupt sensitivity is programmable by nibble in the *GIO Control* register. A pin not used as an interrupt can be masked via the *GIO Mask* register.

Level sensitive interrupts can be used to combine other system interrupts into the AHA5140's **UPINTI** interrupt output.

5.0 COUNTER / TIMER

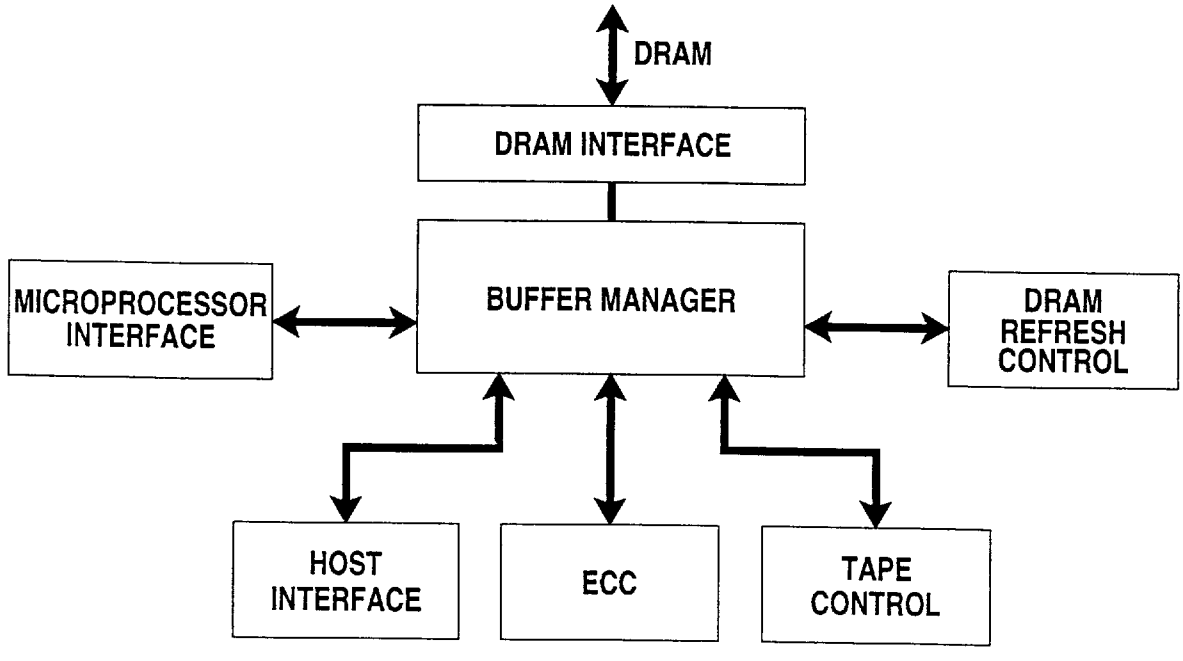
The AHA5140 provides a general purpose 16-bit counter/timer for use by the drive firmware. In addition, hardware support is added to utilize the counter for tape operations. The counter can be enabled to count or halted. When enabled, it counts every time a selectable event occurs. This can be every **SYSCLK**, every low-frequency amble pattern written to tape, every **Amble Found** interrupt, or every **Rewrite** interrupt. The event source can also be prescaled (divided) by 1, 4, 64, or 1024. The counter can only be written when halted but can be safely read at anytime.

6.0 BUFFER MANAGER

All access to the external DRAM buffer memory is done through the DMA sub-system and controlled by the buffer manager. Figure 3 shows a block diagram of the DMA sub-system. It is the buffer manager that arbitrates between the DRAM requests and assigns access according to the arbitration rules. The design of the system assures none of the rules are violated. The rules governing arbitration are listed below. All times and rates assume **SYSCLK** is operating at 40 MHz.

1) Allow a CAS-before-RAS DRAM refresh cycle to occur at least every 620 **SYSCLK** cycles.
2) The Tape Control receives enough bandwidth to assure that the supported formats can be read or written at a channel rate of up to 1.66 MB/s (19.9 Mb/s).
3) The Processor Interface has rules for both bandwidth and latency. The microprocessor is allowed up to 1.0 MB/s of DRAM bandwidth with a typical latency of 1.5µs. The typical latency is assured for all microprocessors accesses accept when a DRAM refresh is scheduled to occur. When the refresh and microprocessor are both requested, the latency is less then 2.5µs. If the microprocessor is using all of its bandwidth, the average latency is less than 1.6µs.
4) The Host Interface is allowed up to 11.1 MB/s of DRAM bandwidth. The AHA5140 could sustain this host rate indefinitely while maintaining the Tape Control, the Microprocessor Interface and the Refresh channels at their maximum rates. In practice, however, a host transfer at this rate would exhaust the external DRAM buffer memory relatively quickly. It is this fact that allows the ECC channel sufficient bandwidth to operate.
5) The ECC Channel operates at a lower priority than the other channels. It only gets DRAM bandwidth not used by the other channels. When all of the other channels are operating at their maximum rate, the ECC Channel will not get any DRAM bandwidth. However, as soon as any of the channels drops below peak performance, the ECC channel will grab the available cycles. As a rule of thumb, the ECC channel get one byte for every two bytes *not* transferred by the sum of the other channels. Under normal circumstances, it is the Host Interface that is unable to sustain its maximum rate. As soon as the host transfer exhausts the external DRAM buffer memory, the ECC channel data rate increases to about 5 MB/s.

Figure 3: DMA Sub-System



7.0 HOST INTERFACE

The host interface to the IC is through a generic DMA-style interface. The DMA-style interface allows for connection to an external controller such as a SCSI or ATAPI chip.

When host transfers are idle, the memory address and transfer count registers can be used to copy data from one section of the external buffer memory to another.

7.1 HOST DATA MANAGER

The **Host Data Manager** has two DMA channels for accessing the external buffer memory. Each DMA channel has a memory address pointer and a data transfer count as well as control, status and interrupts. However, the channels do share the same data path and consequently only one can be active at a time. The two channels can be configured so that one channel starts immediately upon the completion of the other. This allows a host data transfer to be mapped into non-contiguous memory locations without interruption. This feature is needed to split a host transfer into frames. The channels are referred to as *DMA A* and *DMA B*. The microprocessor can determine which channel is scheduled to be used next and the status of the channels by reading the *Host Status* register.

Host data transfers can be *non-blocked* or *blocked* as controlled by the **HBM** (Host Block Mode) bit in the *Host Configuration 2* register. Non-blocked transfers always place host data in consecutive memory locations. In blocked transfer mode, the DMA system reserves room in the external buffer memory for the microprocessor to later fill with the tape formats control bytes.

7.1.1 NON-BLOCKED TRANSFERS

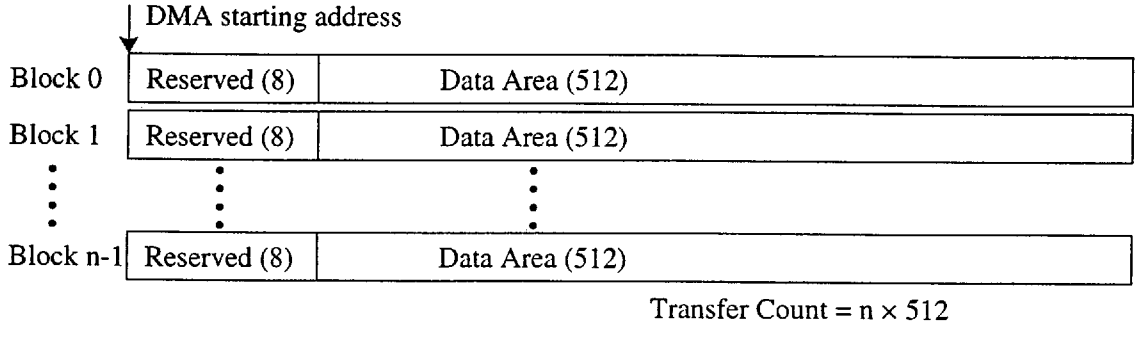
Non-blocked transfers are used for transferring sections of contiguous memory to or from the host. This type of transfer would not normally be used for tape data which is blocked. In addition, memory-to-memory copies are performed with non-block transfers.

Non-blocked transfers can be performed on either *DMA A* or *DMA B* when the **HBM** (Host Block Mode) bit is inactive. However, the DMA channels are not separately configurable for block mode: either both are blocked or both are non-blocked. With non-blocked transfers, the DMA address register is incremented by one and the DMA count register is decremented by one for each byte transferred to or from external buffer memory. When the DMA count expires, a DMA Done interrupt (either **Host DMA A Done** or **Host DMA B Done**) is generated.

7.1.2 BLOCKED TRANSFERS

Blocked transfers are used to transfer data formatted for the tape to or from the host. Tape formatted data has an area reserved for control bytes placed before the data area for the block. A block mode transfer will transfer only bytes from the data area. The figure below shows the memory structure for an '*n*' block transfer in QIC-3095. The memory structure depends upon the tape format selected in the *Tape Format* register.

Blocked transfers can be performed on either *DMA A* or *DMA B* when the **HBM** (Host Block Mode) bit in *Host Configuration 2* is active. However, the DMA channels are not separately configurable for block mode: either both are blocked or both are non-blocked. To accomplish block mode transfers, the DMA address register automatically skips the locations that are reserved for control bytes. The algorithm used to do this adds the number of control bytes (8 for QIC-3095 and TRAVAN-5) to the address register when the DMA is enabled and after every full block (512 bytes for QIC-3095 and TRAVAN-5) is transferred. Since the reserved area is skipped when the DMA is enabled, the starting address for a DMA should be the address for the first control byte not the address of the first data byte. For every byte in the data area, the address register increments by one. The DMA counter register is decremented by one of every byte transferred. The counter is not decremented for bytes that are skipped (control bytes).



If a block mode is disabled (with ***Host A Disable*** or ***Host B Disable***) before it completes, the Host Interface must be reset (with ***Host Reset***) before the next block mode DMA can be enabled.

7.1.2.1 CROSSING FRAME BOUNDARIES

The blocked transfer does not skip the positions reserved in the tape format for the ECC blocks. The microprocessor should setup the DMAs such that one completes at the end of the last data block in a frame. The next DMA should then start at the beginning of the next frame. The second DMA can be enabled before the first completes to avoid a pause in the data flow to the host. The following example demonstrates the steps to do this. It assumes a 32Kbyte transfer starting at block 30. This procedure assumes that both DMA channels start out disabled.

1) Program *Host DMA A Address* for the beginning of block 30. Program *Host DMA A Count* for the number of data bytes remaining in the frame (for QIC-3095 that is 22 × 512=11,264 bytes). Optionally mask **Host DMA A Done** interrupt.
2) Program *Host DMA B Address* for the beginning of the next frame. Program *Host DMA B Count* for the number of data bytes remaining in the transfer (for QIC-3095 that is 32,768-11,264=21,504 bytes).
3) Enable DMA A with ***Host A Enable*** and enable DMA B with ***Host B Enable***. Note that this must be done with two writes to the *Host Command* register.
4) Wait for **Host DMA B Done** interrupt.

7.1.2.2 TRANSFER DOES NOT START AT THE BEGINNING OF A BLOCK

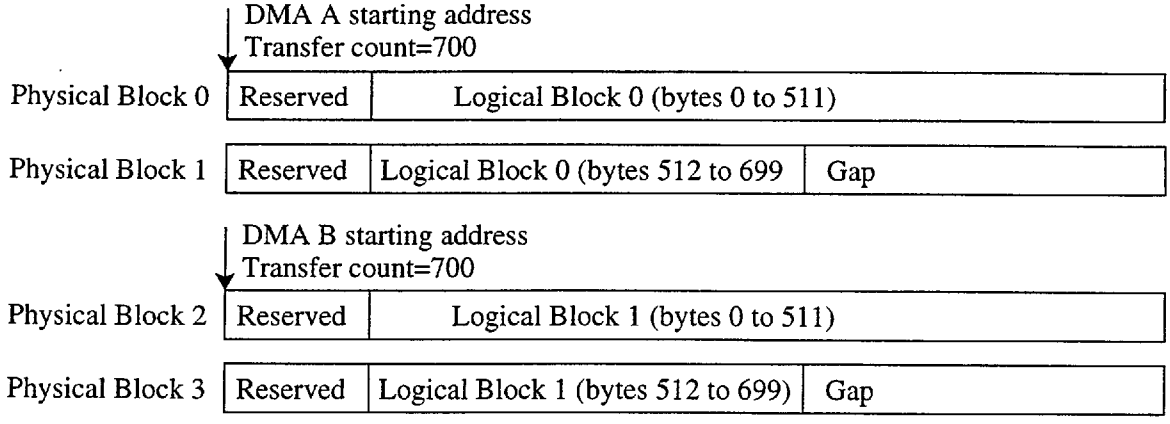
The DMA address generation algorithm assumes that host transfers start at the beginning of a block. While this is almost always the case, situations may arise where the host transfer must start at a location other than the beginning of a block. The following example demonstrates the steps to do this. This procedure assumes a 1K byte transfers starting at byte 20 of a block, a QIC-3095 format.

1) Program *Host DMA A Address* for the address of the first byte minus 8 (8 is automatically added when the DMA is enabled). Program *Host DMA A Count* for the number of bytes between the first byte and the end of the data area of the block. For the example, *Host DMA A Address* would be programmed to 20-8=12 and *Host DMA A Count* to 512-20=492. Enable DMA with the ***Host A Enable*** command.
2) Program *Host DMA B Address* for the beginning (first control byte not the first data byte) of the second block. Program *Host DMA B Count* for the remainder of the transfer. For the example, *Host DMA B Address* would be programmed for the beginning of the next block and *Host DMA B Count* to 1024-492=532. Enable DMA with the ***Host B Enable*** command.
3) Wait for **Host DMA A Done** and **Host DMA B Done** interrupts.

7.1.2.3 LOGICAL BLOCK IS NOT MULTIPLE OF PHYSICAL BLOCK

The QIC tape formats required that every logical block begins on at the beginning of a physical block. When the logical block length is multiple of the physical block length, this also means that every logical block will end at the end of the physical block. However, when the logical block length is not a multiple of the physical block length, the end of logical block will not be aligned with the end of physical block. This fact necessitates gaps in the memory map. As an example, the figure below shows how the two, 700 byte logical blocks would be DMA'ed. The steps involved in setting up these transfers are:

1) Set the *Host DMA A Address* register for the beginning of a block. Set the *Host DMA A Count* register for the length of the transfer, 700 bytes. Enable the transfer with the ***Host DMA A Enable*** command.
2) Either before or after the **Host DMA A Done** interrupt, setup DMA B. Set the *Host DMA B Address* register for beginning of the third physical block. Set the *Host DMA B Count* to 700 bytes. Enable with ***Host DMA B Enable*** command.
3) After the **Host DMA A Done** interrupt, another the DMA A channel can be re-initialized and enabled.



7.1.3 MEMORY-TO-MEMORY COPY

The host DMA channels can be used to copy data from one section of memory to another. This copy is performed at a rate of approximately one half the maximum host data rate. This mode is enabled with the **COPY** bit in *Host Configuration 2*.

In copy mode, the source of the data is specified with *Host DMA A Address* and the destination with *Host DMA B Address*. The number of bytes to be transferred must be specified in both *Host DMA A Count* and *Host DMA B Count*. Block mode transfers must be disabled during a copy operation and the **Host DMA A Done** interrupt should be masked.

To start the copy operation, enable issue ***Host A Enable*** and ***Host B Enable*** commands during the same microprocessor write cycle. The **Host DMA B Done** interrupt will occur when the copy is finished.

7.1.4 PARITY ERRORS

Parity errors can occur on either the host interface pins or on the DRAM interface. Each source of parity errors has its own interrupt; **Host Bus Parity Error** and **Host DRAM Parity Error**. Other than the interrupt, the module handles parity errors from either source the same way. When a parity error is detected the enabled **DMA** is put in the *error* state.

8.0 ECC

The ECC module performs both the encoding and decoding operations for the QIC-3095 and TRAVAN-5 formats. The error correction code implemented is a Reed-Solomon code in GF(256). The field is generated from the following primitive polynomial:

$$F(x) = x^8 + x^7 + x^2 + x + 1$$

The generator polynomial for the code is a function of the redundancy, *r*, and therefore depends upon the programming of the *ECC Format* register. However, the general form of the polynomial is:

$$G(x) = \prod_{i=0}^{r-1} (x + 2^i)$$

A redundancy of either 6 or 10 is supported.

8.1 BACKGROUND

The memory format expected by the ECC is determined by the programming of the *ECC Format* register. Figure 4 and Figure 5 show the memory matrix for the supported tape formats. Encoding and decoding are performed on ECC codewords which are columns when the frame is viewed as this matrix. The bytes in a column in the same interleave are in the same codeword. The bytes in that column from a different interleave are part of a different codeword. Processing on one codeword completes before the next codeword is started. Multiple codewords, up to an entire interleave, can be processed in one operation before an interrupt is generated. The number of codewords in an operation is programmed in *ECC Codewords*.

In addition to specifying the number of blocks, the starting address for the operation needs to be programmed in *ECC DMA Address*. The starting address for a codeword is the memory address for the first byte in the codeword. This byte is always in the first block of the interleave. When performing ECC on an entire interleave, program the starting address to the memory address of the control byte of the first block.

Selection between encoding and decoding is done with **ENCODE** bit in the *ECC Configuration* register. The *ECC Command* register controls the starting and stopping of the ECC.

Figure 4: Memory Matrix for QIC-3095 Format

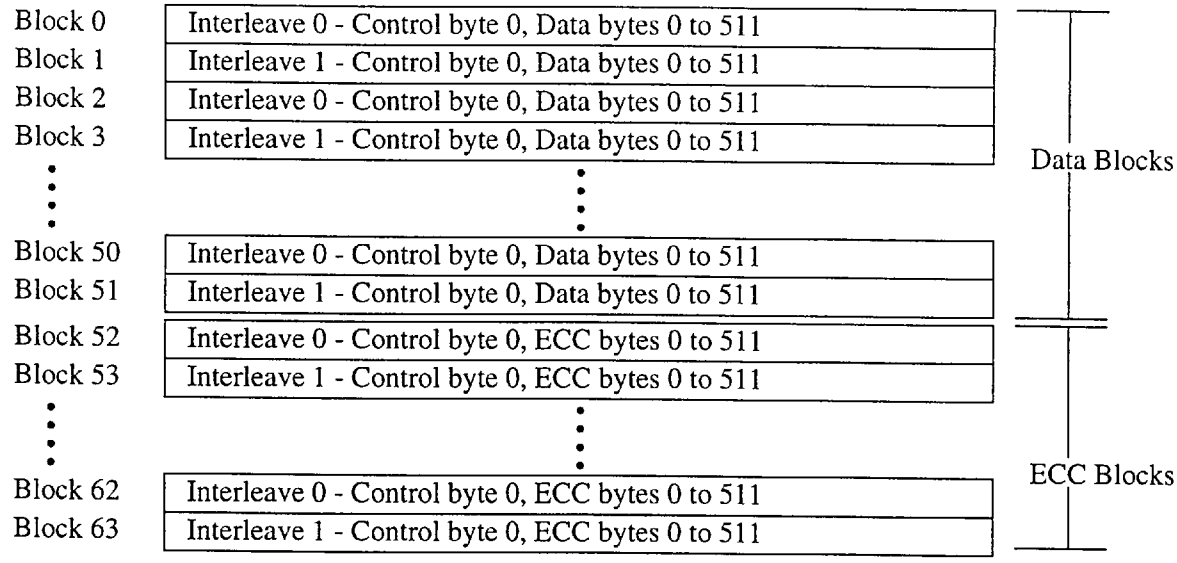
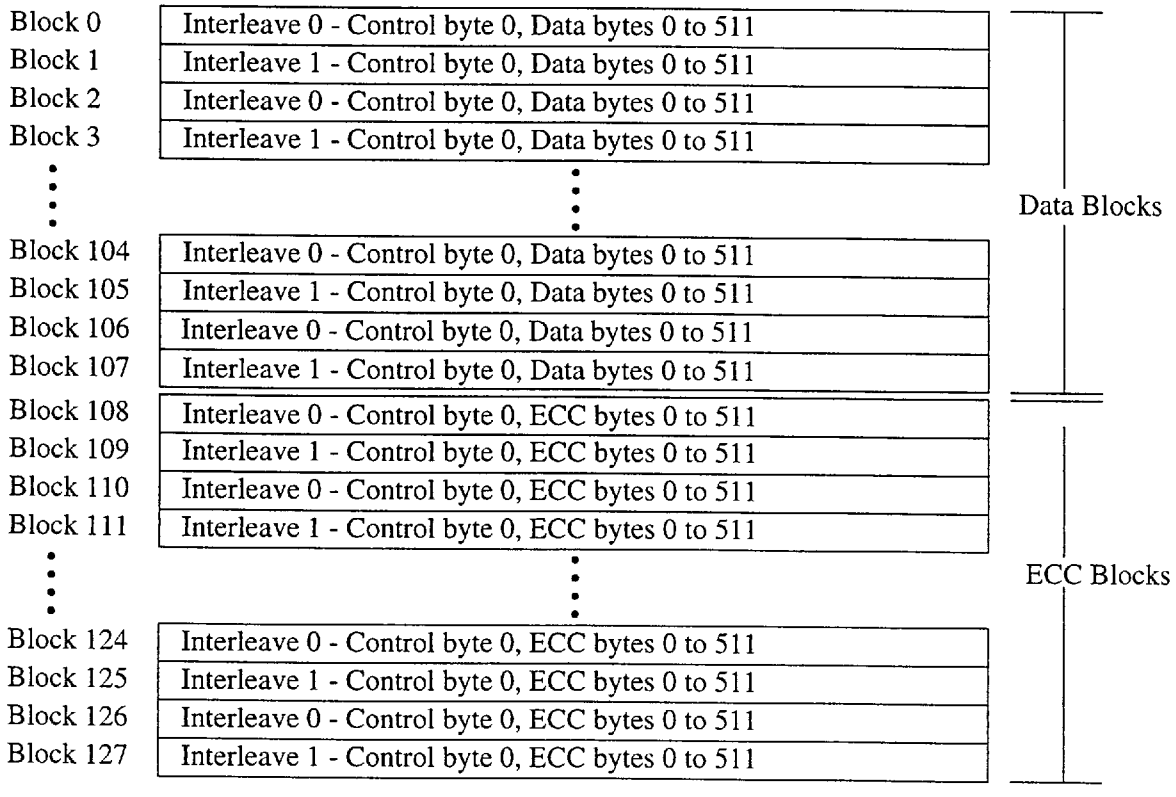


Figure 5: Memory Matrix for TRAVAN-5 Format



8.2 ECC ENCODING

The ECC encoding operation generates the codewords for a frame and writes them to external buffer memory. After encoding, the frame is ready to be written to tape. The module can be programmed to process up to one interleave at a time. The following procedure can be used to ECC encode a frame:

1) Setup interleave 0 by programming *ECC DMA Address* with the address for control byte 0 of block 0 and the codeword counter register, *ECC Codewords* with the number codewords in the interleave, 513 (201hex). Select ECC encoding with the **ENCODE** bit. Start the operation with the ***START*** command.
2) Wait for **ECC Done** interrupt.

3) Setup interleave 1 by programming *ECC DMA Address* with the address for control byte 0 of block 1 and the codeword counter register, *ECC Codewords* with the number blocks in the interleave, 513 (201hex). ECC encoding should already be selected in the **ENCODE** bit. Start the operation with the ***START*** command.
4) Wait for ***ECC Done*** interrupt.

If DRAM parity checking is enabled, a parity error is possible. If a parity error is detected, an **ECC Parity Error** interrupt is generated and encoding terminates as if a ***STOP*** command was received. The codeword containing the parity error can be determined by reading the *ECC Codewords* register. If desired, the encoding operation can be re-started on the codeword with the error.

8.3 ECC DECODING

The ECC decoding operation corrects the data bytes that are flagged as being in error. The error flags for a frame are referred to as the "erasure vector." The erasure vector is generated from the CRC check that is performed while reading from tape. The memory address for the erasure vector must be programmed in the *ECC Erasure Address* register. In addition, the decoding operation must know which interleave is being decoded. This is programmed in the **INTR** bit. The ECC module loads the erasure vector immediately after a ***START*** command is received.

The module can be programmed to decode up to one interleave at a time. Upon the completion of each codeword, an **ECC Codeword Done** interrupt is generated. If the module is unable to decode a codeword it generates an **ECC Codeword Uncorrectable** interrupt. An uncorrectable codeword causes the decode operation to terminate. The *ECC Codewords* register can be read to determine which codeword caused the error. Since under certain circumstances the decoder does not know a codeword will cause an error until after correction has been attempted, several bytes in the codeword may be modified before the operation is terminated.

A general procedure for decoding a QIC-3095 frame is shown below. This procedure assumes that all codewords in the frame are correctable by the decoder:

1) Setup interleave 0 by programming *ECC DMA Address* with the address for control byte 0 of block 0 and the codeword counter register, *ECC Codewords* with the number codewords in the interleave, 513 (201hex). Program the erasure vector location in *ECC Erasure Address* (for this case, $EEA = EDA + 33,273$). Program the interleave, **INTR** to zero. Select ECC decoding by setting the **ENCODE** bit to zero. Start the operation with the ***START*** command.
2) Wait for **ECC Done** interrupt.
3) Setup interleave 1 by programming *ECC DMA Address* with the address for control byte 0 of block 1 and the codeword counter register, *ECC Codewords* with the number of codewords in the interleave, 513 (201hex). The *ECC Erasure Address* register must be re-programmed with the same value as was written in step 1. Program the interleave, **INTR** to one. ECC decoding should already be selected. Start the operation with the ***START*** command.
4) Wait for **ECC Done** interrupt.

If DRAM parity checking is enabled, a parity error is possible. If a parity error is detected, an **ECC Parity Error** interrupt is generated and decoding terminates as if a ***STOP*** command was received. During decoding, parity errors can occur while reading the erasure vector, while reading in a codeword or while performing corrections on a codeword. Operation can be re-started on the codeword with the error.

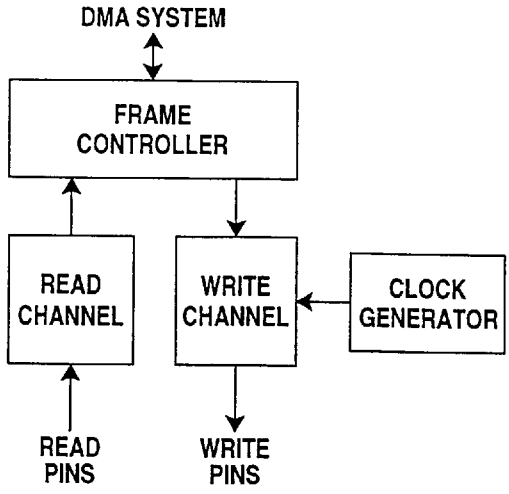
9.0 TAPE

9.1 TAPE OVERVIEW

9.1.1 TAPE BLOCK DIAGRAM

The tape channel formats data going to and from tape. It supports read and write for both QIC-3095 and the TRAVAN-5 format as well as read compatibility for QIC-80, QIC-3010 and QIC-3020. The tape channel provides frame level control logic to provide a high-level user interface. The tape channel consists of four main sections: a write channel, read channel, frame controller, and clock generator as shown in Figure 6. As the figure indicates, the read and write channels are independent. This provides for full read-while-write support.

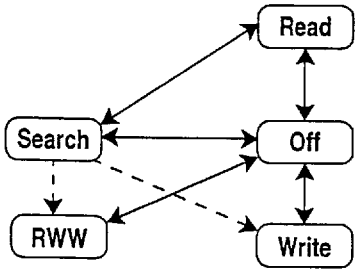
Figure 6: Tape Block Diagram



9.1.2 TAPE CHANNEL MODES

The tape channel has five modes of operation selected by the *Tape Control* register. These are Off, Write, Read-While-Write, Search, and Read. The simplest is Off which acts as a reset preventing any data transfers or interrupts and clearing all state machines and internal storage. In Write mode, data is sent from DRAM to tape but no read channel interrupts occur. In Read-While-Write mode, data is written to tape and the read channel reports all interrupt and status information. In Search mode, all of the read interrupts and status are reported but no data is sent to DRAM. In Read mode, all read channel interrupts are active and data is sent to DRAM. The mode can be switched from Off to any other mode and vice versa. In addition the controller can go from Off to Search to Write or Read-While-Write as shown in Figure 7.

Figure 7: Tape Operating Modes



9.1.3 DATA TRANSFERS

A tape operation is performed by first setting the appropriate mode and then enabling a data transfer. The tape controller contains two address pointers which are used as initial addresses for DMA transfers. One is for the "current" and one for the "next" transfer. Transfers are single blocks or entire frames. Since the AHA5140 is intended for frame level operation, these pointers are called frame pointers. A frame is enabled by writing the *Frame Control* register. As each block/frame completes, the other pointer becomes the current pointer. The *Frame Control* register provides a bit indicating which pointer is the next. The microprocessor enables the first frame, prearms the second, and then only needs to prearm a new frame as each one finishes.

9.1.4 FRAME FORMAT

The Frame controller assumes a particular frame format in memory. Each block is placed sequentially in memory with no gaps. The erasure vector (CRC status) information used for reads is stored immediately after the last block of the frame. The CRC status of each block is stored as a single bit. Each byte represents eight blocks. The first block is stored in bit 7 and the eighth block in bit 0. The microprocessor can examine erasure information by indirectly accessing DRAM.

The following subsections describe the physical formats supported and then discusses each of the tape operations.

9.2 PHYSICAL FORMATS

The physical formats used by the tape channel are selected in the *Tape Format* register. The particular block marker, modulation scheme, control bytes, data bytes, and CRC are set according to the selected format. There are two formats supported. One is a RLL format defined by QIC-3095. The other is an MFM format defined by QIC-80, QIC-3010 and QIC-3020. The following sections describe the details of each format.

9.2.1 RLL SUPPORT

The block format for QIC-3095 and TRAVAN-5 has the following fields and (unencoded) byte lengths.

2	*8*	*512*	*4*
Block Marker	Control Bytes	Data Bytes	CRC

A two block write would have the following sequence and byte lengths.

TIMED	*13,20,30*		*1*	*13,20,30*		*1*	*TIMED*
Default Amble	Preamble	Block	Postamble	Preamble	Block	Postamble	Default Amble

9.2.1.1 AMBLE (RLL)

After enabling a Write or Read-While-Write, the tape channel will output a default amble pattern. The default pattern is:

```
000000010000000100000001        (low frequency amble)
```

When a frame or block is enabled, the tape controller will complete the current 24-bit default amble pattern and then switch to high frequency amble (normal preamble). The pattern is:

```
010101010101
```

The number of bytes (12 bit patterns) of normal preamble is selected in *Tape Format*. At the end of a block, the normal postamble is always one byte (12 bits).

9.2.1.2 BLOCK MARKER (RLL)

The block marker pattern is:

```
010 101 010 000 000 100 000 010
```

During a read, the block marker must be preceded by 24 bits of high frequency preamble to be considered valid.

9.2.1.3 MODULATION CODE (RLL)

The modulation code is RLL 1:7 as defined by the following table where "X" means the inverse of the previous bit:

DATA BITS	*ENCODED BITS*
01	X00
10	010
11	X01
0001	X00 001
0010	X00 000
0011	010 001
0000	010 000
1110 1110	010 000 001 001
1011 1011	010 000 001 010

During a write, if the CRC ends in 00, an extra 01 will be inserted in the data stream to form a valid RLL encoding.

9.2.1.4 RANDOMIZATION (RLL)

Control and Data bytes can be randomized as selected in *Tape Control*. The polynomial used for randomization is:

$$x^{12} + x^6 + x^4 + x + 1$$

If enabled, randomization starts from the MSB of Control Byte 7 and goes through the LSB of Data Byte 511. All bits of the randomizer circuit are set to 1 prior to encoding. The same holds for derandomization.

9.2.1.5 CONTROL BYTES (RLL)

There are always 8 control bytes in a block. The physical block number is expected to be in first three control bytes (7, 6, and 5) and stored LSB first. When performing reads, the physical block number is extracted from the control bytes and stored in the *Incoming Block* register. When the third byte is loaded a **Control Bytes Available** interrupt is issued.

9.2.1.6 DATA BYTES (RLL)

The data field is always 512 bytes long.

9.2.1.7 CRC (RLL)

The CRC is generated bitwise on the data starting from the MSB of control byte 7 through the LSB of data byte 511. The CRC is calculated before randomization and RLL encoding. The generator polynomial is:

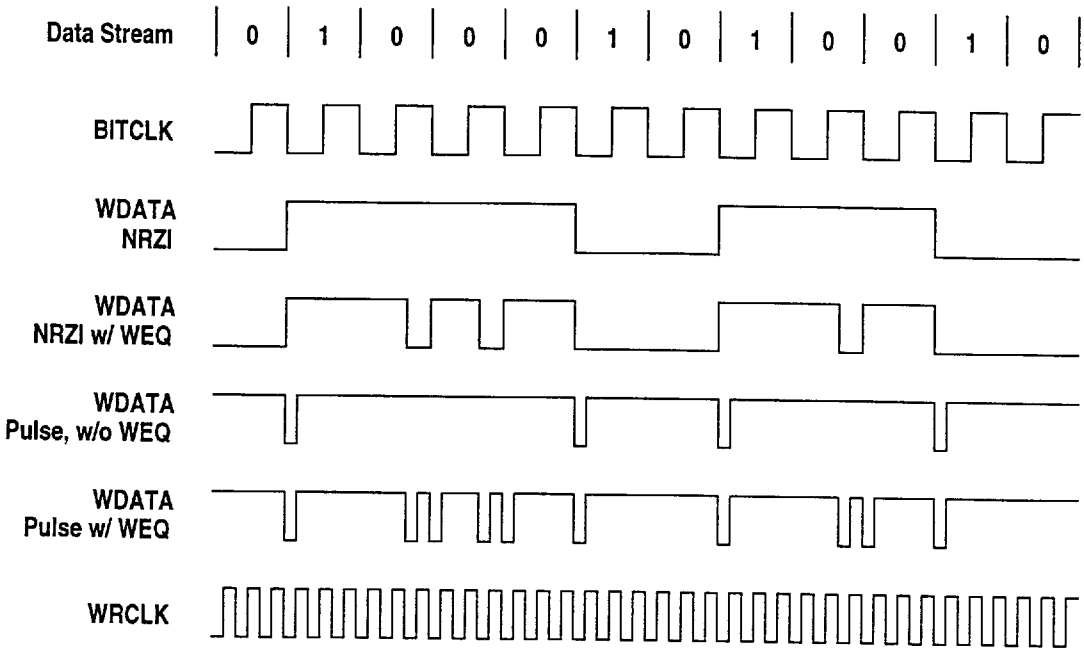
$$x^{32} + x^{28} + x^{26} + x^{19} + x^{17} + x^{10} + x^{6} + x^{2} + 1$$

All bits are set to 1 prior to CRC calculations. In the read channel, every block reports a CRC interrupt, either good or bad (**CRC Good** and **CRC Bad** respectively).

9.2.1.8 OUTPUT FORMAT (RLL)

The output stream can be in a NRZI or Pulse format. Either mode may also be write equalized. In Pulse format, each '1' in the data stream is a 1/6 bit cell pulse. If write equalization is enabled, two extra 1/6 cell pulses are inserted between any pair of zeros. In NRZI format, the output toggles for each '1' and, if write equalization is enabled, the output pulses for 1/3 a bit cell between any pairs of zeros. NRZI data changes with the falling edge of **WRCLK**. Figure 8 illustrates the output format.

Figure 8: Tape Channel Output Data Waveforms

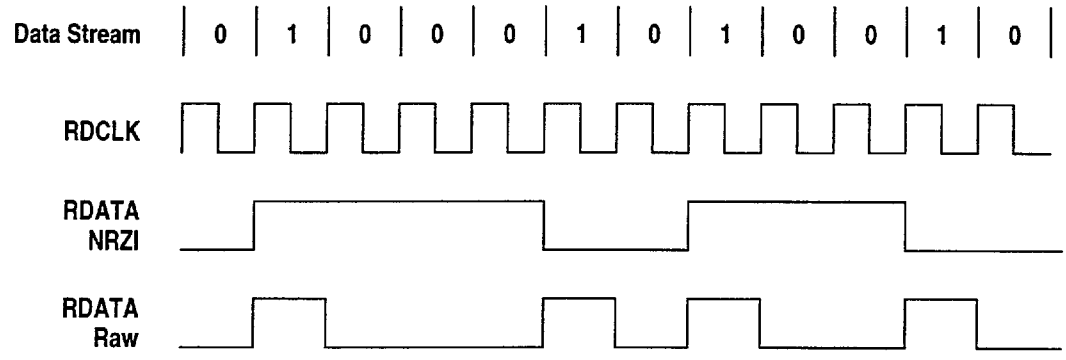


Write equalization can be turned on and off on the fly. The tape channel will change equalization modes on bit cell boundaries.

9.2.1.9 INPUT FORMAT (RLL)

Read channel input data can be in NRZI or raw data form. The value on **RDATA** is sampled on the rising edge of each **RDCLK**. Selection between the two modes is made in *Tape Control*. The following diagram illustrates the two modes.

Figure 9: Tape Channel Input Waveforms



9.2.2 MFM SUPPORT

The AHA5140 offers read compatibility for QIC-80, QIC-3010 and QIC-3020 formats (MFM). These formats read at low data rates so only block level support is provided. The chip must be placed in block mode and each DMA enabled seperately. The current and next block (frame) pointers can be thought of as active and prearm DMAs.

QIC-80, QIC-3010 and QIC-3020 contain three different block types, Segment Headers, Track Sector IDs, and Data Blocks. The AHA5140 detects all three and generates an interrupt for each. For each Segment Header, a **Segment Header Found** interrupt is generated. For a Track Sector ID block, the floppy track, side, and sector are stored into the *Incoming Block* register and a **Control Bytes Available** interrupt issued. A bad CRC for a Track Sector ID is reported via the **Track ID CRC Bad** interrupt. At the end of a Data Block the CRC is checked and a **CRC Good** or **CRC Bad** interrupt issued.

9.2.2.1 SEGMENT HEADER (MFM)

A Segment Header contains no information but the AHA5140 issues a **Segment Header Found** interrupt each time an Index Address Mark is detected on tape. An Index Address Mark is a specially encoded C2C2C2FC:

```
1 1 0 0 0 0 1 0 1 1 0 0 0 0 1 0 1 1 0 0 0 0 1 0 1 1 1 1 1 1 0 0  Raw C2C2C2FC
1010010001001000101001000100100010100100010010000101010101010100101 MFM
```

The index address mark must be preceded by 24 bits of sync mark 0x00, (1.5 bytes). The read channel does not search for Segment Headers while reading Track Sector IDs or Data Blocks.

9.2.2.2 TRACK SECTOR ID (MFM)

A Track Sector ID starts with a Sector ID Address Mark and contains the floppy track, side, and sector numbers for the following Data Block. The Sector ID Address Mark is a specially encoded 0xA1A1A1FE:

```
1 0 1 0 0 0 0 1 1 0 1 0 0 0 0 1 1 0 1 0 0 0 0 1 1 1 1 1 1 1 1 0  Raw A1A1A1FE
1000100100010010100010010001001010001001000100101010101010101000 MFM
```

When the read channel recognizes a Sector Address ID Mark preceded by 24 bits of sync mark (0x00, 1.5 bytes), it places the floppy track, side, and sector numbers into the *Incoming Block* register and issues a **Control Bytes Available** interrupt. These bytes are never written to DRAM. The CRC for the Track Sector ID is reported via the **Track ID CRC Bad** interrupt. This can be sampled as a status bit after the **Block Marker Found** for the following Data Block.

9.2.2.3 DATA BLOCK (MFM)

A Data Block starts with a Data Address Mark and contains 1024 bytes of data. The Data Address Mark is a specially encoded 0xA1A1A1FB:

```
1 0 1 0 0 0 0 1 1 0 1 0 0 0 0 1 1 0 1 0 0 0 0 1 1 1 1 1 1 0 1 1  Raw A1A1A1FB
1000100100010010100010010001001010001001000100101010101010001010 MFM
```

When the read channel recognizes a Data Address Mark preceded with 24 bits of sync mark (0x00, 1.5 bytes), it issues a **Block Marker Found** interrupt and reads the next 1024 bytes of data off tape. If the tape mode is set to Read, these bytes are sent to DRAM. At the end of the block it verifies the CRC and issues a **CRC Good** or **CRC Bad** interrupt.

9.2.2.4 MODULATION CODE (MFM)

The AHA5140 expects an input clock with a frequency twice that of the bit rate off tape. In particular "data" and "clock" bits are both sampled by **RDCLK**. The MFM decoder converts 2 bit pairs into single bits according to the following table:

UNENCODED PATTERN CURRENT (NEXT)	*ENCODED PATTERN*
00 (0)	Error
00 (1)	0
01 (0)	0
01 (1)	Error
10	1
11	Error

If any modulation errors are found within a Data Block, a **Modulation Error** interrupt is reported. Only one interrupt is generated per block.

9.2.2.5 CRC (MFM)

The CRC is calculated after MFM decoding and is based on the following polynomial:

$$x^{16} + x^{12} + x^{5} + 1$$

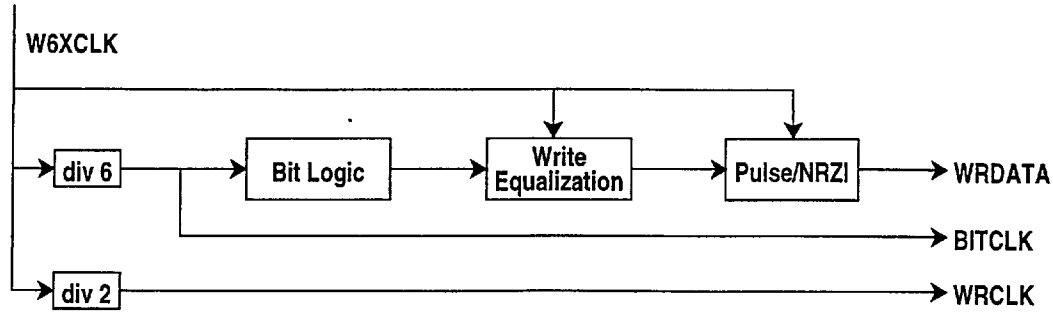
All CRC calculation bits are initialized to 1. For a Track Sector ID block the CRC is calculated from the most significant bit of the first Sector ID Address Mark byte to the least significant bit of the bytes per sector (4th) byte. For a Data Block the CRC is calculated from the most significant bit of the first Data Address Mark to the least significant bit of the 1024th data byte. Separate interrupts are generated for good and bad CRCs (**CRC Good** and **CRC Bad** respectively).

9.2.3 TAPE CHANNEL CLOCKS

The **Tape Interface** receives the **W6XCLK**. This clock is divided by 6 and optionally output as **BITCLK**. **BITCLK** is also used to clock the bit-level logic in the Tape Write Channel. In order to generate the write equalization pulses required by the format, the **W6XCLK** is used. The output data, **WRDATA**, is driven off chip from the rising edge of **W6XCLK**.

The read data input pin, **RDATA**, is sampled off the rising edge of each **RDCLK**. During normal operation, the internal read channel logic is clocked with **RDCLK**. For chip self test, an internal loopback mode is provided. In loopback mode, **RDCLK** is effectively driven by **BITCLK** and **RDATA** by the **WRDATA**. The selection between normal and loopback mode is made in the *Tape Clock* register.

Figure 10: Tape Channel Clocks



The read data input pin, **RDATA**, is sampled of the rising edge of each **RDCLK**. During normal operation, the internal read channel logic is run off **RDCLK**. For chip self test, an internal loopback mode is provided. In loopback mode, **RDCLK** is effectively driven by **BITCLK** and **RDATA** by the write channel. The selection between normal and loopback mode is made in the *Clock Configuration* register.

9.2.4 READ CHANNEL DATA RECOVERY CONTROL

The AHA5140 outputs two signals for data recovery control in the read channel IC. These signals are **RGATE** and **PLLSYNC**. **RGATE** is intended to switch the data recovery circuit from fixed to closed-loop clock mode and reset the VCO. The data recovery circuit can be switched from high gain to tracking mode with **PLLSYNC**.

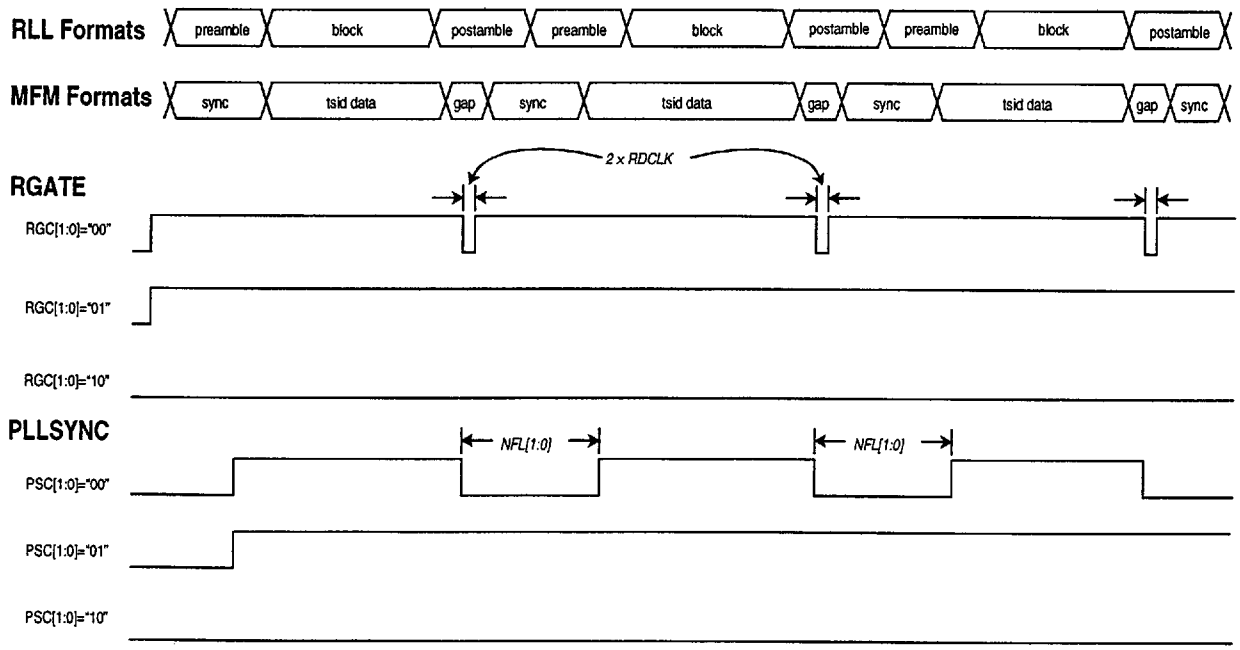
The operation of both **RGATE** and **PLLSYNC** are programmable in the *Read Channel Control* register. Figure 11 shows the operation of the signals under the various programmed options.

RGATE asserts active high when the AHA5140 is placed in Search, Read or Read-While-Write modes. The **RGC** then controls the conditional deassertion of the signal. Using **RGC**, **RGATE** can be programmed to pulse low for two **RDCLK** cycles after every CRC field. Implicit in this implementation is that **RDCLK** cycles even when **RGATE** is inactive. **RGATE** can also be programmed to be continuously active or inactive.

PSC and **NFL** control the operation of **PLLSYNC**. **PLLSYNC** asserts active high after a programmable amount of high frequency amble has been detected. The amble pattern detected is:

0 1 0 1 0 1 0 1 0 1 0 1

Figure 11: Read Channel Data Recovery Control



NFL specifies the number of repetitions of this pattern that are needed for assertion of **PLLSYNC**. Four pattern counts are provided: 4, 8, 16 or 21. If the programmed number of patterns is not detected before the block marker is detected, **PLLSYNC** will remain deasserted through the block and begin searching again during the next amble field. In all cases, the pattern counter is reset at the beginning of the amble field.

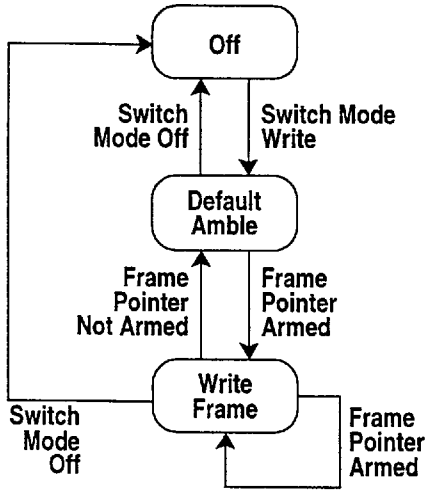
PLLSYNC can be programmed to deassert after every CRC field. After the deassertion, the AHA5140 begins the search for the programmed number of high frequency amble patterns. Once the pattern is found **PLLSYNC** is again asserted. Using **PSC**, **PLLSYNC** can also forced either inactive or active.

Notice that Figure 11 shows that **PLLSYNC** for the MFM formats is active before the first data block. This is because the first synchronization mark is most likely found before the first Track Sector ID (for example, at the Segment mark or in the data).

9.3 WRITE

Placing the AHA5140 in Write mode will cause the tape controller to turn on the write gate (**WGATEN**) and start to output default amble. All write channel interrupts are enabled. All read channel interrupts are disabled. Data is read from DRAM only after a frame pointer is enabled. Figure 12 illustrates the general write process.

Figure 12: Tape Write Process



9.3.1 FRAME LEVEL WRITE

The AHA5140 is capable of writing entire frames without user intervention. Frame level operation is enabled in the *Frame Control* register. When a frame is ready to go to tape, the user enables a frame pointer and the controller sets up all the DMAs for the entire frame. The *Frame 0(1) Pointer* registers are used as the bases to calculate the starting address of each tape block and remain as constants. The two frame pointers alternate to provide the starting address for the current and next frames. Initiating a write is accomplished with a simple four step process:

1) Program the *Drive Format*, *Tape Clock*, *Tape Control*, and *Tape Format* registers for the desired configuration and format.
2) Set the mode in *Tape Control* to Write. This starts default amble going to tape. Wait long enough to write long or extended preamble.
3) Write the *Expected Block* register with the physical block number of the first block in the frame. Write the *Frame 0 Pointer* with the address of the start of the frame in DRAM. Write the *Frame Control* register to enable frame 0.
4) When the second frame is ready, write the *Frame 1 Pointer* with the address of the second frame in DRAM. Write the *Frame Control* register to enable frame 1.

Note that the registers are organized sequentially so each step can be accomplished through block move instructions.

As the write of each frame is completed, **Frame 0(1) Done** interrupts are generated for each of the pointers respectively. The firmware need only to re-program and re-enable the corresponding frame pointer with each interrupt. The *Expected Block* register does not need to be rewritten after initiating the write. To maintain stream, a new frame must be enabled before the write channel completes the data bytes of the last block in the current frame.

The frame controller makes its decisions about what to do next at the end of each block (when switching from data to CRC bytes) using the following rules:

1) If the current frame pointer is now disabled, the write channel completes the current block and switches to default amble. This would occur if the microprocessor needs to abort a frame at the end of a track.
2) If the next block is within the current frame, the frame controller sets up a DMA for the next block.
3) If the current block is the last in a frame and the next frame is enabled, the frame controller sets up a DMA for the next block and issues a frame done interrupt (**Frame 0 Done** or **Frame 1 Done**).
4) If the current block is the last in a frame and the next frame is not enabled, the frame controller goes into an underrun and issues a **Unarmed Frame** interrupt. In an underrun, the frame controller either writes default amble or rewrites the last block at selected in the *Frame Control* register.

Completing a write is accomplished by not enabling any new frames after the last. Additionally, the underrun control bit in *Frame Control* should be set to write default amble. The controller will issue both a **Frame (0/1) Done** and an **Unarmed Frame** interrupt at the end of the last frame. The microprocessor then times this postamble before turning off the write gate **WGATEN** by switching the tape channel mode to Off.

When writing, the frame controller bases DMAs on the value in *Outgoing Block* register. When the first frame is enabled, the physical block number in *Expected Block* is stored in *Outgoing Block*. At the end of each block (when changing from data to CRC bytes), *Expected Block* and *Outgoing Block* are updated and a **Block Done** interrupt is issued. Usually they are both incremented by 1. Two exceptions occur around an underrun if forced streaming is enabled (i.e. the controller continually rewrites the last block). When entering such an underrun, *Expected Block* increments by 1 while *Outgoing Block* retains its previous value. Once in a forced stream, both registers remain at their new values.

9.3.2 BLOCK LEVEL WRITE

Data can be written on a block-by-block basis by clearing the frame support bit (**FSUP**) in *Frame Control*. When in block mode, the frame pointers become block pointers and enabling a pointer causes one block to be written to tape. The control mechanisms are virtually identical to frame level writes. After initiating the write, the firmware enables a new block after each **Frame 0(1) Done** interrupt. The *Expected Block* register, however, does not need to be written. A **Block Done** interrupt will occur at the same time as each **Frame 0(1) Done**. Failure to prearm a new block before the end of the data bytes for the current block will cause an underrun condition. If an underrun occurs, a **Unarmed Frame** interrupt is issued and the controller always drops into default amble.

9.4 READ-WHILE-WRITE

Just like a Write, when the tape channel is placed in Read-While-Write mode the write gate **WGATEN** turns on and default amble is output on **WRDATA**. Data is transferred from DRAM to the output pins. Unlike a Write, all write and read channel interrupts are active. The major differences are in the rules for which block to write next and the definition of a frame done.

9.4.1 FRAME LEVEL READ-WHILE-WRITE

The AHA5140 provides automated read-while-write capabilities on a frame basis. Frame level read-while-write is enabled by setting the tape **MODE** in *Tape Control* to Read-While-Write and then setting the **FSUP** bit in *Frame Control*. Enabling a Frame Pointer will write an entire frame including any necessary rewrites. The *Frame 0(1) Pointer* registers are used as bases to calculate the starting address of each tape block and remain as constants. The two frame pointers alternate to provide the starting address for the current and next frames.

The frame controller ensures that every physical block is read back with a good CRC. Blocks are written in ascending order, stepping back to rewrite blocks as necessary. A write is initiated with the following process:

1) Program the *Drive Format, Tape Clock, Tape Control*, and *Tape Format* registers for the desired configuration and format.
2) Set the mode in *Tape Control* to Read-While-Write. This starts default amble going to tape. Wait long enough to write long or extended preamble.
3) Write the *Expected Block* register with the physical block number of the first block of the first frame. Write the *Frame 0 Pointer* with the address of the start of the frame in DRAM. Write the *Frame Control* register to enable frame 0.
4) When the second frame is ready, write the *Frame 1 Pointer* with the address of the second frame in DRAM. Write the *Frame Control* register to enable frame 1.

Note that the registers are organized sequentially so each step can be accomplished through block move instructions.

As the write of each frame is completed, **Frame 0(1) Done** interrupts are generated for each of the pointers respectively. The firmware need only to re-program and re-enable the corresponding frame pointer with each interrupt. The *Expected Block* register does not need to be rewritten after initiating the write. To maintain stream, a new frame must be enabled before the write channel completes the data bytes of the last block in the current frame.

The *Expected Block* register contains the physical block number of the block the frame controller is trying to write to tape and read back with a good CRC. The *Outgoing Block* register contains the block number for the current block going to tape. In the normal case, *Outgoing Block* is greater than *Expected Block* by the head gap of the drive. An internal counter keeps track of the difference between these two registers.

The *Incoming Block* register in the read channel contains the physical block number for each block read from tape and is updated at the beginning of each block. The *Last Incoming Block* also contains the physical block number but is updated at the end of each block. *Last Incoming Block* also contains the CRC status of the reported block. The frame controller uses *Last Incoming Block* in making rewrite decisions.

When the first frame is enabled, the frame controller will set *Outgoing Block* to the value of *Expected Block*. At the end of each block (at the boundary between data and CRC bytes), the frame controller decides what block, if any, to write next according to the following rules:

1) If the current frame pointer is now disabled, the write channel finishes the current block and switches to default amble. This would occur if the microprocessor needs to abort a frame at the end of a track.
2) If *Last Incoming Block* matches *Expected Block* and has a good CRC, *Outgoing Block* and *Expected Block* are incremented by 1. This is the normal case. If *Expected Block* crosses a frame boundary, a **Frame 0(1) Done** interrupt is issued. If *Outgoing Block* is going to cross a frame boundary, the following additional rules take place.
   a) If the next frame is enabled, the first block of the next frame is written.
   b) If the next frame is not enabled, an **Unarmed Frame** interrupt is generated, *Expected Block* is incremented, and *Outgoing Block* is not incremented. If forced streaming is selected in *Frame Control*, the last block will be continually rewritten until the next frame is enabled.
3) If *Last Incoming Block* matches *Expected Block* and has a bad CRC, *Outgoing Block* is reset to *Expected Block*, *Expected Block* remains unchanged, and a **Rewrite** interrupt is issued. This is a normal rewrite.
4) If the counter recording the difference between *Outgoing Block* and *Expected Block* is greater than the head gap specified in *Drive Format*, *Outgoing Block* is reset to *Expected Block*, *Expected Block* remains unchanged, and a **Rewrite** interrupt is issued. This handles the case when a block is written but never read back.
5) If none of the above four conditions are true, *Outgoing Block* increments by 1 and the next block is written to tape. If *Outgoing Block* is going to cross a frame boundary but the next frame is not enabled, an **Unarmed Frame** interrupt is generated, *Outgoing Block* is not incremented and the last block is rewritten until the frame is complete (or a rewrite occurs).

After a **Rewrite** interrupt, *Expected Block* contains the block that was rewritten. A rewrite counter tracks how many times *Expected Block* is rewritten. After the 15th rewrite fails, *Expected Block* and *Outgoing Block* are increment by 1 as if the block was good and a **Max Rewrites** interrupt is generated.

In the event of a short-term underrun, the frame controller will complete writing all of the blocks for the current frame before writing any blocks from the new frame. In any underrun (including end of data) the difference counter monitoring *Outgoing Block* and *Expected Block* will increment in cases where *Outgoing Block* does not. This assures that the tape controller will still rewrite blocks that are never read back from tape.

A write sequence is completed by simply not arming the next frame. When *Outgoing Block* reaches the next frame, an **Unarmed Frame** interrupt is issued and the last block is rewritten until the current frame is completed. After the **Frame 0(1) Done**, the controller will write default amble or continue re-writing the last block depending on the setting of underrun control (**URC**) in *Frame Control*.

9.4.2 BLOCK LEVEL READ-WHILE-WRITE

Data can be written with read-while-write verification on a block-by-block basis by clearing the frame support bit (**FSUP**) in *Frame Control*. When in block mode, the frame pointers become block pointers and enabling a pointer causes one block to be written to tape. The control mechanisms are virtually identical to a block level Write except that read channel interrupts are active. *Expected Block* and *Outgoing Block* have no meaning in block mode.

After initiating the write, the firmware enables a new block after each **Frame 0(1) Done** interrupt. At this time, the microprocessor needs to decide if a rewrite should take place. In order to make this decision the firmware needs the physical block number and CRC status of the last block read from tape. This information is contained in *Last Incoming Block*. If the user can guarantee that a read block will not finish during the **Frame 0(1) Done** interrupt service routine, *Last Incoming Block* can be read directly. Otherwise the **CRC Good** and **CRC Bad** service routines must read *Last Incoming Block* and provide some form of handshaking mechanism with the **Frame 0(1) Done** routine. The decision process for which block to write next is identical to that outlined in Frame Level Support above.

A **Block Done** interrupt will occur at the same time as each **Frame 0(1) Done**. Failure to prearm a

new block before the end of the data bytes for the current block will cause an underrun condition. If an underrun occurs, a **Unarmed Frame** interrupt is issued and the controller drops into default amble.

9.5 APPEND

Appends are accomplished by searching for an appendable point, usually the end of data, and then switching to a Write or Read-While-Write. The Search mode enables the read channel so it reports control bytes and all interrupts while not transferring data. As each block is detected the physical block numbers are stored in *Incoming Block* and a **Control Bytes Available** interrupt issued. The **CRC Good** and **CRC Bad** interrupts are available if the control bytes need to be verified. Once the firmware knows the track and physical block numbers of the last frame on tape, it can:

1) Search for the physical blocks from the last frame on tape
2) Wait for an **Amble Found** interrupt
3) Switch to a Write or Read-While-Write.

9.6 READ

9.6.1 FRAME LEVEL READ

The AHA5140 offers automated reading of entire frames. The physical block numbers coming off tape are examined real-time and the appropriate DMAs started. The *Frame 0 Pointer* and *Frame 1 Pointer* registers indicate where to place the current and next frame in DRAM. Once a frame pointer is enabled the controller will read all of the frame it can. The frame controller also generates a table in DRAM of which blocks passed and failed CRC checks for use by the ECC unit.

If the tape channel is placed in Search mode, all of the read interrupts are reported but no data is transferred. Search is useful for locating the position on tape via physical block numbers. The block numbers are stored in *Incoming Block* and can be read after a **Control Bytes Available** interrupt. When switching from a Search to a Read, data is transferred starting at the beginning of the first block found after entering Read mode. When switching from a Read to a Search, the last block of the Read is completed before shutting off further data transfers. Reads are initiated by the following steps:

1) If necessary place tape controller in Search mode to find correct location on tape.
2) Write the physical block number of the first block of the first frame into *Expected Block*.
3) Write the starting address of the first frame in DRAM into *Frame 0 Pointer*.
4) Enable the *Frame 0 Pointer* via *Frame Control*.
5) When the next frame is free in DRAM, set *Frame 1 Pointer* to the next frame and enable it.

As the read for each frame completes, a **Frame 0(1) Done** interrupt tells the firmware to re-program and re-enable the respective frame pointer. *Expected Block* does not need to be rewritten after starting a read.

When a frame pointer is enabled, the frame controller sets the erasure vector (CRC vector) in DRAM to all 1's. As each block comes in, a DMA is set up according to the following rules:

1) If no frames are enabled, all incoming blocks are discarded.
2) If the *Incoming Block* is less than *Expected Block*, the block is discarded.
3) If the *Incoming Block* is within the current frame (as defined by *Expected Block*), the erasure information for the block is fetched from DRAM. If the block hasn't been seen with a good CRC, the block is transferred. Otherwise it is discarded.
4) If the *Incoming Block* is within the next frame and the next frame is enabled, the erasure information is fetched. If the block hasn't been seen with a good CRC, the block is transferred. Otherwise the block is discarded. If the *Incoming Block* is far enough into the next frame and has a valid CRC, a **Frame 0(1) Done** interrupt is issued for the current frame. This distance is set by **RWD** in *Drive Format* and is 8 for QIC-3095.
5) If the *Incoming Block* is within the next frame but the next frame is not enabled, the block is discarded. If the CRC is valid a **Unarmed Frame** interrupt is issued. If in addition to a valid CRC, the *Incoming Block* is **RWD** or more blocks into the next frame, a **Frame 0(1) Done** interrupt is issued for the current frame.
6) If the *Incoming Block* is greater than the current or next frame, the block is discarded. If the CRC is valid then an **Out of Window** interrupt is generated indicating the desired frames were completely missed.

The AHA5140 read architecture allows for multiple passes to be made on the same frame. If the erasure vector initialization feature is turned off (via **EVINIT** in *Frame Control*) on a subsequent pass, only blocks which haven't been seen get transferred. Hence multiple passes may make otherwise uncorrectable frames good.

To maintain stream, the next frame must be enabled prior to one complete block before the first block of the next frame. If a frame is manually disabled by the microprocessor, the controller will complete the transfer of the current block if one is still in progress.

9.6.2 BLOCK LEVEL READ

Data can be read on a block-by-block basis by clearing the frame support bit (**FSUP**) in *Frame Control*. QIC-80, QIC-3010 and QIC-3020 formats must be read in block mode. When in block mode, the frame pointers become block pointers and enabling a pointer causes one block to be read from tape. The Expected Block register is undefined is this mode.

After the control bytes for QIC-3095 or Track ID bytes for QIC-80/3010/20 are read into [illegible]

11.0 REGISTERS

Multi-byte registers are organized little endian and are long-word aligned. To support the 196 Peripheral Transaction Server, the interrupt registers are contiguous in memory.

The IC supports two addressing modes for internal registers. In the default mode **UPAD[6:0]** drives the internal address bus. In 8x186 mode, **UPAD[7:1]** drives the internal bus effectively making all addresses occur on even locations. In

11.1 REGISTER SUMMARY

ADDRESS	REGISTER NAME
0x00	System Configuration (Normal addressing)
0x01	System Configuration (8×186 addressing)
0x02	Version
0x03	Clock Configuration
0x04	System PLL Control
0x05	System PLL Control
0x06	Tape PLL Control
0x07	Tape PLL Control
0x08	reserved
0x09	reserved
0x0A	reserved
0x0B	reserved
0x0C	reserved
0x0D	reserved
0x0E	reserved
0x0F	reserved
0x10	reserved
0x11	reserved
0x12	reserved
0x13	reserved
0x14	reserved
0x15	reserved
0x16	reserved
0x17	reserved
0x18	reserved
0x19	reserved
0x1A	reserved
0x1B	reserved
0x1C	reserved
0x1D	reserved
0x1E	reserved
0x1F	reserved
0x20	reserved
0x21	reserved
0x22	reserved
0x23	reserved
0x24	ECC DMA Address [7:0]
0x25	ECC DMA Address [15:8]
0x26	ECC DMA Address [23:16]
0x27	reserved
0x28	ECC Erasure Address [7:0]
0x29	ECC Erasure Address [15:8]
0x2A	ECC Erasure Address [23:16]
0x2B	reserved
0x2C	ECC Codewords [7:0]
0x2D	ECC Codewords [15:8]
0x2E	ECC Command / ECC Status
0x2F	ECC Configuration
0x30	Host DMA A Address [7:0]
0x31	Host DMA A Address [15:8]
0x32	Host DMA A Address [23:16]
0x33	reserved
0x34	Host DMA A Counter [7:0]
0x35	Host DMA A Counter [15:8]
0x36	reserved
0x37	reserved
0x38	Host DMA B Address [7:0]
0x39	Host DMA B Address [15:8]
0x3A	Host DMA B Address [23:16]
0x3B	reserved
0x3C	Host DMA B Counter [7:0]
0x3D	Host DMA B Counter [15:8]
0x3E	reserved
0x3F	reserved

ADDRESS	REGISTER NAME
0x40	Host Configuration 1
0x41	Host Configuration 2
0x42	Host Command / Host Status
0x43	reserved
0x44	DRAM Address [7:0]
0x45	DRAM Address [15:8]
0x46	DRAM Address [23:16]
0x47	DRAM Wait State
0x48	DRAM Data
0x49	DRAM Configuration
0x4A	GIO Data
0x4B	GIO Control
0x4C	Tape Test Configuration
0x4D	ECC Format
0x4E	reserved
0x4F	reserved
0x50	Counter Timer [7:0]
0x51	Counter Timer [15:8]
0x52	Counter Configuration
0x53	Read Channel Control
0x54	Drive Format
0x55	Tape Format
0x56	Expected Block [7:0]
0x57	Expected Block [15:8]
0x58	Frame 0 Pointer
0x59	Frame 0 Pointer
0x5A	Frame 0 Pointer
0x5B	reserved
0x5C	Frame 1 Pointer
0x5D	Frame 1 Pointer
0x5E	Frame 1 Pointer
0x5F	reserved
0x60	Frame Control
0x61	Tape Control
0x62	Outgoing Block [7:0]
0x63	Outgoing Block [15:8]
0x64	Incoming Block [7:0]
0x65	Incoming Block [15:8]
0x66	Incoming Block [23:16]
0x67	reserved
0x68	Last Incoming Block [7:0]
0x69	Last Incoming Block [15:8]
0x6A	Last Incoming Block [23:16]
0x6B	Last Incoming Block [31:24]
0x6C	Tape Current Memory Address [7:0]
0x6D	Tape Current Memory Address [15:8]
0x6E	Tape Current Memory Address [23:16]
0x6F	reserved
0x70	Host Interface Interrupts
0x71	ECC and Other Interrupts
0x72	Tape Frame Interrupts
0x73	Tape Block Interrupts
0x74	GIO Interrupt
0x75	reserved
0x76	reserved
0x77	reserved
0x78	Host Interface Interrupts Mask
0x79	ECC and Other Interrupts Mask
0x7A	Tape Frame Interrupt Mask
0x7B	Tape Block Interrupt Mask
0x7C	GIO Mask
0x7D	reserved
0x7E	reserved
0x7F	reserved

11.2 TAPE REGISTERS

11.2.1 DRIVE FORMAT

Read/Write
Reset Value (hex) = XX

	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
0x54	WNRZI[1:0]		RNRZI	reserved	GAP[3:0]			

RNRZI - NRZI Input. When set to 1, **RDATA** is expected to be in NRZI format. When clear, **RDATA** is expected to be plain serial data.

WNRZI[1:0]-Controls data format on **WRDATA** output pin according to the following table:

WNRZI[1:0]	*FORMAT*
00	Output is in pulse format
01	Output is in NRZI format
10	Output is in raw format
11	reserved

GAP[3:0] - Head Gap. The distance in blocks between the read and write heads of the drive. Program with the next higher integer from the actual gap. For Read-While-Write, this sets how far ahead of *Expected Block Outgoing Block* can get before a read error must have occurred.

11.2.2 READ CHANNEL CONTROL

Read/Write
Reset Value(hex)=0x3F

	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
0x53	reserved		RGC[1:0]		PSC[1:0]		NFL[1:0]	

RGC[1:0] - RGATE Control. It is illegal to change bit 0 of this register except when the tape mode is Off. Bit 1 can be changed at anytime.

RGC[1:0]	*DESCRIPTION*
00	**RGATE** goes active when the tape operating mode is switched to Search, Read or Read-While-Write. Upon the reading of each CRC field, **RGATE** pulses inactive for 2 **RDCLK** cycles. **RGATE** is deasserted as soon as the operating mode switches.
01	**RGATE** goes active when the tape operating mode is switched to Search, Read or Read-While-Write. It remains active while in that operating mode. As soon as the mode is switched, **RGATE** is deasserted.
1X	**RGATE** is deasserted and remains deasserted.

PSC[1:0] - PLLSYNC Control. It is illegal to change bit 0 of this register except when the tape mode is Off. Bit 1 can be changed at anytime.

PSC[1:0]	*DESCRIPTION*
00	**PLLSYNC** deasserts after every CRC field is read. **PLLSYNC** remains inactive until the amount of amble programmed in **NFL** is found. Once found, **PLLSYNC** asserts.
01	**PLLSYNC** asserts when the programmed amount of amble is found. Once active, **PLLSYNC** remains high until the tape mode switches from Search, Read, or Read-While-Write mode. When the mode switches, **PLLSYNC** is deasserted.
1X	**PLLSYNC** is deasserted and remains deasserted.

NFL[1:0] - Normal Amble Found Pattern Length. Sets how many 12-bit high frequency patterns will cause the **PLLSYNC** to assert when **PSC**= "00." The pattern that is searched for is "010101010101."

It is only legal to change these bits when the tape mode is Off.

PSC[1:0]	DESCRIPTION
00	Searches for 4 occurrences of the pattern
01	Searches for 8 occurrences of the pattern
10	Searches for 16 occurrences of the pattern
11	Searches for 21 occurrences of the pattern

11.2.3 TAPE FORMAT

Read/Write
Reset Value (hex) = XX

	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
0x55	RWD[1:0]		AFL[1:0]		PAL[1:0]		FRMT[1:0]	

RWD[1:0] -Rewrite Distance. The RLL formats define a value k where once a drive writes block N+k, it can no longer rewrite block N. For QIC-3095 this value is 8. Provided for future formats.

RWD[1:0]	MAXIMUM DISTANCE
00	8
01	12
10	16
11	reserved

AFL[1:0] - Amble Found Interrupt Length. Determines the number of 8-bit amble pattern repetitions required to trigger an **Amble Found** interrupt. See table below

PAL[1:0] - Preamble Length. Sets the number of 12 bit patterns of high frequency preamble before each block. The bits are only operative when writing to tape. See table below.

FRMT[1:0]- Selects the specific format. Used to determine block marker pattern, modulation code, block length, and CRC.

VALUE	AFL[1:0]	PAL[1:0]	FRMT[1:0]
00	63	13	test
01	127	20	MFM
10	255	30	RLL
11	511	reserved	reserved

11.2.4 TAPE CONTROL

Read/Write
Reset Value (hex) = 00

	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
0x61	reserved			WEQ	RAND	MODE[2:0]		

WEQ - Write Equalization. When set to 1, turns on write equalization. Independent of **NRZ** setting.

RAND - Randomization. When set enables randomization.

MODE[2:0]-Set Tape Channel Operating Mode. The mode must be set to Off when switching modes with the following exceptions: Read to Search, Search to Read, Search to Write, Search to Read-

While-Write. The Off state acts as a reset for all tape channel state machines and the enable state machines inside the frame controller.

MODE[2:0]	*OPERATING MODE*
000	Off
100	Search
101	Read
110	Write
111	Read-While-Write
Others	reserved

11.2.5 EXPECTED BLOCK

Read/Write
Reset Value (hex) = XXXX

	bit7	*bit6*	*bit5*	*bit4*	*bit3*	*bit2*	*bit1*	*bit0*
0x56	**EXBLK[7:0]**							
0x57	**EXBLK[15:8]**							

EXBLK[15:0]-Expected Block. For any write to tape, the *Expected Block* register is programmed with the physical block number of the first block going to tape. During a Write it contains the physical block number of the block going to tape. During a Read-While-Write it contains the physical block number of the next block the controller is trying to write and then read back with good CRC. For either of the write modes, *Expected Block* is updated at the end of each block written as indicated by the **Block Done** interrupt. *Expected Block* will increment by 1 or maintain its previous value. *Expected Block* will remain stable until 1 byte before the end of the data field of the next block.

For a read from tape, *Expected Block* is initially programmed with the physical block number of the first block of the first frame the user wants to read from tape. *Expected Block* is used to determine if the *Incoming Block* is from the current frame, next (armed) frame, or outside both. *Expected Block* increments to the beginning of the next frame whenever a **Frame 0(1) Done** interrupt occurs and is otherwise stable.

11.2.6 FRAME 0 POINTER

Read/Write
Reset Value (hex) = XXXXXX

	bit7	*bit6*	*bit5*	*bit4*	*bit3*	*bit2*	*bit1*	*bit0*
0x58	**FP0[7:0]**							
0x59	**FP0[15:8]**							
0x5A	reserved			**FP0[21:16]**				
0x5B	reserved							

FP0[21:0] - Contains the address of the beginning of a frame in memory. Used as the base address for DMA calculations and does not change. In block mode, indicates the beginning of the next block to be read or written. The *Frame 0 Pointer* must be written before its enable bit **F0EN** is set. Alternates with the *Frame 1 Pointer*.

11.2.7 FRAME 1 POINTER

Read/Write
Reset Value (hex) = XXXXXX

	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
0x5C	FP1[7:0]							
0x5D	FP1[15:8]							
0x5E	reserved			FP1[21:16]				
0x5F	reserved							

FP1[21:0] - Contains the address of the beginning of a frame in memory. Used as the base address for DMA calculations and does not change. In block mode, indicates the beginning of the next block to be read or written. The *Frame 1 Pointer* must be written before its enable bit **F1EN** is set. Alternates with the *Frame 0 Pointer*.

11.2.8 FRAME CONTROL

Contains the control bits to enable frame level operation and to enable or disable frames and blocks. This register is reset whenever the **MODE** in *Tape Control* is set to Off.

Read/Write
Reset Value (hex) = 00

	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
0x60	URC	FSIZ	EVINIT	FSUP	NEXT	FDIS	F1EN	F0EN

URC - Underrun Control. Sets options for handling an underrun during a write. If set the frame controller will continue to rewrite the last block until new data arrives or the frame is disabled. If clear, frame controller will drop into default amble. If **URC** is cleared during an underrun, the tape controller will complete the current block before writing postamble.

FSIZ - Frame Size. When **FSUP** is set, selects frame size. zero = 64, one = 128. Has no effect when **FSUP** is clear.

EVINIT - Erasure Vector Initialization. When set, the frame controller does not initialize the erasure vector table when frames are enabled. Used to make progressive passes on hard to read frames. Normally clear.

FSUP - Frame Support. Selects between block and frame level support. When set, frame level control is enabled. Enabling a frame pointer will read or write an entire frame from or to tape. The **URC**, and **FSIZ** bits in this register have meaning. When **FSUP** is clear, block level control is enabled. Enabling a frame pointer will read or write a single block from tape.

NEXT - Indicates which frame pointer is next to be enabled. zero = frame 0, one = frame 1. Read-Only.

FDIS - Frame Disable. Writing a one to this bit disables both frame pointers. Writing a '0' does nothing. Disabling takes precedence over enabling. When a frame is disabled it will complete the current block. Always reads as a '0'. If an immediate termination of a read or write is required, change the tape **MODE** in *Tape Control* to "Off". The tape **MODE** must be toggled to "off" after a ***frame disable*** command is received prior to the next ***frame 0(1) enable*** command. In addition, the **URC** bit must be cleared when writing the ***frame disable*** command.

F1EN - Frame 1 Enable. Writing a '1' enables the *Frame 1 Pointer*. Writing a '0' does nothing. Reading indicates enable status of frame 1 (one = enabled, zero = disabled). The tape channel **MODE** in *Tape Control* must be set to its proper value before setting Frame 1 Enable. **F1EN** is cleared by a manual disable (writing a one to **FDIS**) or when a **Frame 1 Done** interrupt occurs. During a manual disable, **F1EN** will remain set until the current block completes.

F0EN - Frame 0 Enable. Writing a '1' enables the *Frame 0 Pointer*. Writing a '0' does nothing. Reading indicates enable status of frame 0 (one = enabled, zero = disabled). The tape channel **MODE** in *Tape Control* must be set to its proper value before setting Frame 0 Enable. **F0EN** is cleared by a manual disable (writing a one to **FDIS**) or when a **Frame 0 Done** interrupt occurs. During a manual disable, **F0EN** will remain set until the current block completes.

11.2.9 INCOMING BLOCK

Read Only
Reset Value (hex) = XXXXXX

	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
0x64	**INBLK[7:0]** (PBN[7:0] / Floppy Track)							
0x65	**INBLK[15:8]** (PBN[15:8] / Floppy Side)							
0x66	**INBLK[23:16]** (PBN[23:16] / Floppy Sector)							
0x67	reserved							

INBLK[23:0]-In RLL formats, contains the physical block number of the last block detected on tape. **INBLK[23:0]** is updated as the first three control bytes of a block are read from tape. A **Control Bytes Available** interrupt is issued immediately after the last byte is loaded. Remains stable until the next block marker is read from tape.

In MFM formats, contains the floppy track, side, and sector numbers of the last Track ID Block detected on tape. Updated as the first three bytes of a Track ID Block are read from tape. A **Control Bytes Available** interrupt is issued immediately after the last byte is loaded. *Incoming Block* remains stable until the next sector ID address mark is read from tape.

11.2.10 LAST INCOMING BLOCK

Read Only
Reset Value (hex) = XXXXXX

	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
0x68	**LINBLK[7:0]** (PBN[7:0] / Floppy Track)							
0x69	**LINBLK[15:8]** (PBN[15:8] / Floppy Side)							
0x6A	**LINBLK[23:16]** (PBN[23:16] / Floppy Sector)							
0x6B	reserved					**WRT**	**TCRC**	**ICRC**

LINBLK[23:0]-Loaded with the contents of *Incoming Block* (**INBLK[23:0]**) after every **CRC Good** or **CRC Bad** interrupt. Provides ability to read physical block numbers and their corresponding CRC status at the same time. Necessary because the contents of *Incoming Block* are overwritten shortly after a CRC interrupt by the next block off tape. Stable until the next **CRC Good** or **CRC Bad** interrupt.

WRT - Written to DRAM. This bit is set if the block reported by **LINBLK** was written to DRAM. Clear in all other cases. Updated at every **CRC Good** or **CRC Bad** interrupt.

TCRC - CRC Bit for Track Sector ID Blocks. In MFM formats, this bit indicates if the Track Sector ID block had a CRC error (and hence the contents of **LINBLK[23:0]** are suspect). A one indicates an error, a zero indicates a pass. In RLL formats, this bit mimics **ICRC**. Updated at every **CRC Good** or **CRC Bad** interrupt.

ICRC - Incoming CRC. For RLL formats, indicates if the block reported by **LINBLK[23:0]** had a good CRC. A one indicates an error, a zero indicates a pass. In MFM formats, this bit indicates CRC status for the last Data Block. Updated at every **CRC Good** or **CRC Bad** interrupt.

11.2.11 OUTGOING BLOCK

Read Only
Reset Value (hex) = XXXX

	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
0x62	OUTBLK[7:0]							
0x63	OUTBLK[15:8]							

OUTBLK[15:0]-The *Outgoing Block* register contains the lower 16 bits of the physical block number of the currently being written to tape during a Write or Read-While-Write. *Outgoing Block* is always updated at the end of a block going to tape as indicated by a **Block Done** interrupt. The register remains stable until 1 byte prior to the end of the next block. In the normal case, *Outgoing Block* is incremented by 1. In the case of a rewrite, *Outgoing Block* is set to *Expected Block*. When the tape channel mode is set to Write or Read-While-Write and the first frame is enabled, *Outgoing Block* is reset to the value of *Expected Block*.

11.2.12 TAPE CURRENT MEMORY ADDRESS

Read Only
Reset Value (hex) = XXXXXX

	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
0x6C	TCMA[7:0]							
0x6D	TCMA[15:8]							
0x6E	TCMA[21:16]							
0x6F	reserved							

TCMA[21:0]-Contains the DRAM address pointer for the tape channel DMA. Provided for firmware debug purposes. Only stable when no data is being transferred.

11.2.13 TAPE FRAME INTERRUPTS

Reading this register returns the current value of each of the interrupts. The setting of interrupt bits in this registers is not effected by the current interrupt mask setting. The interrupt mask only effects the generation of the interrupts signal.

Interrupts are cleared by writing a '1' to the corresponding bit in this register. Writing a '0' has no effect on the bit.

Read/Write
Reset Value (hex) = 00

	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
0x72	AFI	CBAI	MXI	RWI	OWI	UFI	F1DI	F0DI

AFI - **Amble Found**. Indicates an amount of amble selected in *Tape Format* was detected on tape. Occurs in Search, Read, and Read-While-Write modes.

CBAI - **Control Bytes Available**. Indicates *Incoming Block* has been updated with a new block detected on tape. For QIC-3095, generated after the third byte of a block (the last physical block number) is read from tape and placed in the tape FIFO. For QIC-80/3010/3020 formats, **Control Bytes Available** is generated after the first three bytes of a Track Sector ID block are loaded into *Incoming Block*. Occurs in Search, Read, and Read-While-Write modes.

MXI - **Max Rewrites**. Indicates the frame controller re-wrote a block the maximum allowable number of times without successfully reading it back and moved on to the subsequent block. The offending block is one less than the value of *Expected Block*. **Max Rewrites** is generated coincident with **Block Done.** Only occurs with frame level control in Read-While-Write mode.

RWI - **Rewrite**. The frame controller initiated a block rewrite. Reported at the end of a block coincident with **Block Done**. The offending block can be read from *Expected Block*.Only occurs with frame level control in Read-While-Write mode.

OWI - **Out of Window**. Indicates the Incoming Block has a physical block number greater than either of the armed frames but the CRC is valid. Indicates initiating a read too late on the tape or a severe gap in the data. Offending block is stored in *Incoming Block*. Occurs with frame level control in Search and Read modes. Firmware Timing Problem????

UFI - **Unarmed Frame**. Indicates the frame controller needed data from an unarmed frame to continue reading or writing. For a write, indicates a failure to prearm the next frame in time or reaching the last frame to be written to tape. Generated at the end of an outgoing block coincident with **Block Done**. For a read, indicates a failure to prearm the next frame in data or reaching the last frame to be read from tape. Generated within 10 **SYSCLK**s after a **CRC Good** or **CRC Bad** interrupt. Occurs in Write, Read-While-Write, and Read modes.

F1DI - **Frame 1 Done**. Indicates completion of frame 1. All other functions identical to **Frame 0 Done**.

F0DI - **Frame 0 Done**. Indicates completion of frame 0. See Read and Write sections for definition of completions. Generated coincident with **Block Done** in write modes and coincident with **CRC Good** or **CRC Bad** during a read. Occurs in Write, Read-While-Write, and Read modes.

11.2.14 TAPE BLOCK INTERRUPTS

Reading this register returns the current value of each of the interrupts. The setting of interrupt bits in this registers is not effected by the current interrupt mask setting. The interrupt mask only effects the generation of the interrupts signal.

Interrupts are cleared by writing a '1' to the corresponding bit in this register. Writing a '0' has no effect on the bit.

Read/Write
Reset Value (hex) = 00

	bit7	*bit6*	*bit5*	*bit4*	*bit3*	*bit2*	*bit1*	*bit0*
0x73	**TPEI**	**BDI**	**CRCBI**	**CRCGI**	**MERRI**	**TCBI**	**SHFI**	**BMFI**

TPEI - **Tape Parity Error**. The tape channel read a byte from DRAM with bad parity. Generated when the byte is transferred from the DRAM interface to the tape channel. When the tape channel encounters a parity error it posts the interrupt and continues processing. If the microprocessor then chooses to disable both frames, the parity is known to have occurred in the current or previous block. Occurs in Read, Write, and Read-While-Write modes.

BDI - **Block Done**. Generated when the last byte of the data field for a block is output from the chip. The decision as to which block, if any, to write next is made at this point. During any write, this event updates the *Expected Block* and *Outgoing Block* registers. Occurs in Write and Read-While-Write modes.

CRCBI - **CRC Bad**. Generated at the end of a block read from tape. Indicates a CRC failure for the block. Occurs in Search, Read, and Read-While-Write modes.

CRCGI - **CRC Good**. Generated at the end of a block read from tape. Indicates the CRC check for the block passed. Occurs in Search, Read, and Read-While-Write modes.

MERRI - **Modulation Error**. Generated during the read of a block whenever an invalid RLL or MFM code is encountered. Only one interrupt will be reported per block. Reported at least one **RDCLK** before and **CRC Good** or **CRC Bad** interrupt. Occurs in Search, Read, and Read-While-Write modes.

TCBI - **Track ID CRC Bad**. Generated in MFM formats when the CRC for the Track ID Block is bad. Can be sampled after a **Block Marker Found** interrupt. Normally masked to act as a status bit.

SHFI - **Segment Header Found**. Generated in MFM formats when a Segment Header is detected. Occurs during Search and Read modes.

BMFI - **Block Marker Found**. Generated when a block marker pattern (RLL) or data sync mark (MFM) is read from tape. Once a block is detected, a new block marker will not be recognized until the block is finished. RLL block markers must be preceded by 24 bits of preamble to be considered valid. MFM data sync marker must be preceded by 24 bits of 0x00. Occurs during Search, Read and Read-While-Write modes.

11.2.15 TAPE FRAME INTERRUPT MASK

The *Frame Interrupt Mask* register determines if each interrupt in *Tape Frame Interrupts* can cause **UPINTI** to go active. Masked interrupts are still reported in the *Tape Frame Interrupts* register. A one masks the interrupt and zero enables it.

Read/Write
Reset Value (hex) = FF

	bit7	*bit6*	*bit5*	*bit4*	*bit3*	*bit2*	*bit1*	*bit0*
0x7A	**AFM**	**CBAM**	**MXM**	**RWM**	**OWM**	**UFM**	**F1DM**	**F0DM**

AFM - Mask bit for **Amble Found**. **(AFI)**.

CBAM - Mask bit for **Control Bytes Available**. **(CBAI)**

MXM - Mask bit for **Max Rewrites**. **(MXI)**

RWM - Mask bit for **Rewrite**. **(RWI)**

OWM - Mask bit for **Out of Window**. **(OWI)**

UFM - Mask bit for **Unarmed Frame**. **(UFI)**

F1DM - Mask bit for **Frame 1 Done**. **(F1DI)**

F0DM - Mask bit for **Frame 0 Done**. **(F0DI)**.

11.2.16 TAPE TEST CONFIGURATION

Write Only
Reset Value (hex)=00

	bit7	*bit6*	*bit5*	*bit4*	*bit3*	*bit2*	*bit1*	*bit0*
0x4C	reserved			**TT1[2:0]**			**TT0[2:0]**	

TTx[2:0]	*DESCRIPTION*
000	**BLKSTN** - Active Low block start. Indicates the detection of the block marker by the read channel. Driven from rising edge of **RDCLK**. Active for one **RDCLK** period
001	**CRCBN** - Active Low CRC Bad. Indicates a block with a invalid CRC was read from tape. Driven from rising edge of **RDCLK**. Active for one **RDCLK** period
010	**CRCGN** - Active Low CRC Good. Indicates a block with a valid CRC was read from tape. Driven from rising edge of **RDCLK**. Active for one **RDCLK** period
011	**MERRN** - Active Low modulation error. Indicates a modulation error occurred during a read from tape. Pulses for each decode error. Driven from rising edge of **RDCLK**. Active for one **RDCLK** period
100	Reserved
101	Reserved
110	Reserved
111	Reserved

TT1[2:0] - Selects the source for the **TAPE[1]** output signal. The source is determined according to the above table.

TT0[2:0] - Selects the source for the **TAPE[0]** output signal. The source is determined according to the above table.

11.2.17 TAPE BLOCK INTERRUPT MASK

The *Tape Block Interrupt Mask* register determines if each interrupt in *Tape Block Interrupts* can cause **UPINTI** to go active. Masked interrupts are still reported in the *Tape Block Interrupts* register. A one masks the interrupt and zero enables it.

Read/Write
Reset Value (hex) = FF

	bit7	*bit6*	*bit5*	*bit4*	*bit3*	*bit2*	*bit1*	*bit0*
0x7B	TPEM	BDM	CRCBM	CRCGM	MERRM	TCBM	HMFM	BMFM

TPEM - Mask bit for **Tape Parity Error. (TPEI)**

BDM - Mask bit for **Block Done. (BDI)**

CRCBM - Mask bit for **CRC Bad. (CRCBI)**

CRCGM - Mask bit for **CRC Good. (CRCGI)**

MERRM - Mask bit for **Modulation Error. (MERRI)**

TCBM - Mask bit for **Track ID CRC Bad. (TCBI)**

HMFM - Mask bit for **Header Marker Found. (HMFI)**

BMFM - Mask bit for **Block Marker Found. (BMFI)**

11.3 MICROPROCESSOR INTERFACE REGISTERS

11.3.1 DRAM ADDRESS

The DRAM Address register indicates location and direction of indirect DRAM accesses.

Read/Write
Reset Value (hex) = XXXXXX

	bit7	*bit6*	*bit5*	*bit4*	*bit3*	*bit2*	*bit1*	*bit0*
0x44	DA[7:0]							
0x45	DA[15:8]							
0x46	DRWN	reserved	DA[21:16]					

DA[21:0] - DRAM Address Pointer. Points to the location DRAM to be read or written. If **DRWN** is set for a read, a DRAM prefetch will be initiated after **DA[21:16]** is written.

DRWN - DRAM Read/Write Not. Sets the direction of indirect DRAM transfers. When set indicates a read from DRAM, when clear a write. Causes a read prefetch after *DRAM Address* is written. This bit is required for the controller to minimize number of wait states required for an access.

11.3.2 DRAM DATA

Read/Write
Reset Value (hex) = XX

	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
0x48	DD[7:0]							

DD[7:0] - DRAM Read/Write Data. The *DRAM Data* register is used by the microprocessor in indirectly access the DRAM buffer. It contains write data going to DRAM or read data coming from DRAM. When *DRAM Data* is written, a write to DRAM is scheduled by the controller. When complete, *DRAM Data* is incremented by 1. If **DDBS** in *DRAM Configuration* is set, DRAM Data will increment by 512 after every eight byte. While the write to DRAM is taking place, the microprocessor must not write *DRAM Data* (or *DRAM Address*). The **DBUSY** bit in *DRAM Configuration* indicates if the DRAM is still busy. When *DRAM Data* is read, *DRAM Address* is automatically incremented as in a write and then a DRAM read is scheduled. While the DRAM read is taking place, the microprocessor must not use the contents of *DRAM Data*. **DBUSY** can be read to see when *DRAM Data* is valid. Also, if automatic wait state generation is active, **UPRDY** will extend a read of *DRAM Data* until it updates and its contents are present on the outside of the chip. The same wait process occurs after *DRAM Address* is written for a DRAM read.

11.3.3 DRAM CONFIGURATION

The *DRAM Configuration* register sets options for accessing the DRAM and enables parity.

Read/Write
Reset Value (hex) = 00

	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
0x49	DPE	DPSZ[1:0]		DINIT	DPO	DDBS	DWTE	DBUSY

DPE - DRAM Parity Enable. When active, DRAM parity is checked (according to **DPO**). When inactive, the output enable for the **MDP** pin is always enabled. Parity is always generated but not checked.

DPSZ[1:0] -DRAM Page Size. Selects the starting position in the 22-bit internal address for the DRAM row address. The address bits are numbered from 0 to 21 therefore if the row address starts at bit 9, the DRAM column has 9 bits (bits 0 to 8).

DPSZ	ROW ADDRESS START POSITION
00	bit 9
01	bit 10
10	bit 11
11	reserved

DINIT - Enable DRAM initialization sequence. When this is active, the interface to the external DRAM is reset thus preventing all DRAM activity. Many DRAMs require a period of inactivity after power on before normal operation can be performed. This bit should be active during the DRAM initializing sequence.

DPO - DRAM Parity odd. When active, odd parity is generated and checked (if **DPE** is active). When inactive even parity is enabled.

DDBS - DRAM Data Byte Skip. Selects how *DRAM Address* is automatically incremented. When clear, *DRAM Address* is incremented by 1 after each read or write to *DRAM Data*. When set, *DRAM Address* is incremented by 512 after every eighth read or write to *DRAM Data* and by 1 for all others.

DWTE - DRAM Wait Enable. Controls whether the AHA5140 generates extra microprocessor wait states when writing *DRAM Address* or reading and writing *DRAM Data*. Wait state generation is enabled when **DWTE** is set and disabled when clear. During a write to *DRAM Address* or *DRAM Data*, **UPRDY** will remain active until any previous writes to DRAM have completed. During a read from *DRAM Data*, **UPRDY** will remain active until read data is ready.

DBUSY - DRAM Data Busy. Indicates if a microprocessor DRAM access is still pending. Is set immediately following any write to *DRAM Address* or any access to *DRAM Data*. If **DBUSY** was set by a write to *DRAM Address*, it will clear when the addressed data byte is loaded into *DRAM Data*. If **DBUSY** was set by a write to *DRAM Data*, **DBUSY** will clear after the byte is written to DRAM. In the case of a Write to *DRAM Address* followed by a Write to *DRAM Data*, **DBUSY** will remain set until the write is complete.

11.3.4 SYSTEM CONFIGURATION

The *System Configuration* register sets microprocessor options and is used for production (scan) testing. Please note that accidental writes of ones to reserved bits in this register can have catastrophic results. *System Configuration* appears at two locations so that it can be written regardless of the setting of **AMODE** described below.

Read/Write
Reset Value (hex) = 00

	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
0x00, 0x01	reserved						**SFTRES**	**AMODE**

SFTRES - Software Reset. Writing a '1' to this bit pulses the internal power-on reset bringing the chip to a idle state. **SYSCLK** still runs during this period.

AMODE - Address Mode. When set, the internal address bus is driven by **UPAD[7:1]** instead of **UPAD[6:0]**, effectively making every internal register appear at an even address. This mode is also referred to as the 8x186 addressing mode.

11.3.5 VERSION

The *Version* register contains the revision number of the die.

Read Only
Reset Value (hex) = 11

	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
0x02	**REV[3:0]**				**PROCESS[3:0]**			

REV[3:0] - Logical and/or electrical version of die. First revision is number one.

PROCESS[3:0]-For a given revision, indicates fabrication process. All processes for a given revision meet the same logical and electrical specifications. Initial value is one.

11.3.6 DRAM WAIT STATE

Support for 1Mbit DRAMs (in 1M x 1 or 256K x 4 configurations) has been added with the addition of a DRAM wait state in the **MCASN** timing. The wait state is enabled by setting the **CASWS** bit in the *DRAM Wait State* register. When the DRAM wait state is active, the total available DRAM bandwidth is decreased. The available bandwidth by channel is as follows:

Tape Control - 1.0MB/s (12.0 Mb/s)
Microprocessor Interface - 1.0 MB/s
Host Interface - 6.9 MB/s

Reset Value (hex)=00

	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
0x47	reserved		CASWS	reserved				

CASWS - Controls insertion of extra clock pulse in the **MCASN** low timing. This bit must be programmed high for support of 1Megabit DRAMs (1M x1 or 256K x 4).

11.4 GIO REGISTERS

11.4.1 GIO DATA

Read/Write
Reset Value (hex) = XX

	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
0x4A	GIOD[7:0]							

GIOD[7:0]- GIO Read/Write Data. The *GIO Data* register contains read/write data for the **GIO[7:0]** pins. Reading the register always returns the current value of the **GIO[7:0]** pins. When pins are configured as outputs, writing *GIO Data* sets the value of the corresponding GIO pin. When configured as an input, writing *GIO Data* has no immediate effect. However if the pins are switched to outputs, they will drive the value of the last write.

11.4.2 GIO INTERRUPT

Read/Write
Reset Value (hex) = XX

	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
0x74	GIOI[7:0]							

GIOI[7:0] - The *GIO Interrupt* register contains interrupt status for GIO pins. All pins act as interrupts but may be masked via *GIO Mask*. Interrupts can be posted for signal levels or transitions (edges) as set in *GIO Control*. If a pin is programmed as an edge interrupt, the corresponding **GIOI** bit will be set whenever the specified transition occurs. The interrupt is cleared by writing a one. If a pin is programmed as a level interrupt, the bit is set whenever the GIO pin is at the specified value and writing ones has no effect. For all kinds of interrupts, writing zeros to *GIO Interrupt* has no effect

11.4.3 GIO MASK

Read/Write
Reset Value (hex) = FF

	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
0x7C	GIOM[7:0]							

GIOM[7:0]- GIO Mask. The *GIO Mask* register controls whether interrupts in *GIO Interrupt* cause **UPINTI** to go active. If the bit is set, the interrupt is masked. If the bit is clear, **UPINTI** goes active whenever the corresponding bit in *GIO Interrupt* is set. Note that writing an output pin may cause an interrupt condition so the user needs to mask output pins.

11.4.4 GIO CONTROL

The *GIO Control* register determines the direction and interrupt sensitivity of the **GIO[7:0]** pins.

Read/Write
Reset Value (hex) = 00

	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
0x4B	GEL74	GEL30	GRF74	GRF30	GDIR76	GDIR54	GDIR32	GDIR10

GEL74 - Edge/Level interrupt setting for **GIO[7:4]**. When set, **GIO[7:4]** are edge level interrupts. The corresponding GIO Interrupt bit will be set when the pin rises or falls as set by **GRF74**. When clear, **GIO[7:4]** are edge level interrupts. The corresponding GIO Interrupt bit will be set whenever the GIO pin is at the level specified by **GRF74**.

GEL30 - Edge/Level interrupt setting for **GIO[3:0]**. When set, **GIO[3:0]** are edge level interrupts. The corresponding GIO Interrupt bit will be set when the pin rises or falls as set by **GRF30**. When clear, **GIO[3:0]** are edge level interrupts. The corresponding GIO Interrupt bit will be set whenever the GIO pin is at the level specified by **GRF30**.

GRF74 - Rise/Fall Polarity for **GIO[7:4]**. When set, **GIO[7:4]** are rising edge or high level interrupts depending on **GEL74**. When clear, **GIO[7:4]** are falling edge or low level interrupts.

GRF30 - Rise/Fall Polarity for **GIO[3:0]**. When set, **GIO[3:0]** are rising edge or high level interrupts depending on **GEL30**. When clear, **GIO[3:0]** are falling edge or low level interrupts.

GDIR76 - Direction for **GIO[7:6]**. When set **GIO[7:6]** are outputs. When clear **GIO[7:6]** are inputs.

GDIR54 - Direction for **GIO[5:4]**. When set **GIO[5:4]** are outputs. When clear **GIO[5:4]** are inputs.

GDIR32 - Direction for **GIO[3:2]**. When set **GIO[3:2]** are outputs. When clear **GIO[3:2]** are inputs.

GDIR10 - Direction for **GIO[1:0]**. When set **GIO[1:0]** are outputs. When clear **GIO[1:0]** are inputs.

11.5 COUNTER REGISTERS

11.5.1 COUNTER TIMER

Read/Write
Reset Value (hex) = 0000

	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
0x50	CNTR[7:0]							
0x51	CNTR[15:8]							

CNTR[15:0]-The *Counter Timer* register counts events from a selectable input source. It is provided particularly for counting amble in the tape channel. The count enabling and the event source are set in *Counter Configuration.* Although it is a two byte counter, the contents can be safely read at any time so long as **CNTR[7:0]** is read first. *Counter Timer* can be preset to any value (when disabled) and only increments. When the counter rolls over from 0xFFFF to 0x0000, a **Timer** interrupt is recorded in *ECC and Other Interrupts.*

11.5.2 COUNTER CONFIGURATION

The *Counter Configuration* register determines the operational mode of the counter/timer.

Read/Write
Reset Value (hex) = 00

	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
0x52	reserved			CNTPS		CNTS[1:0]		CNTE

CNTE - Counter Enable. When set the counter increments each time the selected event occurs. When clear counter halts at current value.

CNTS[1:0] -Counter Event Select. Determines what event causes the counter to increment, if enabled via the **CNTE** bit.

CNTS[1:0]	SOURCE
00	**SYSCLK**, counts every clock.
01	Amble Pattern Written. Increments every time the tape channel writes a 24-bit default (low-frequency) amble pattern. Useful to time amble.
10	Amble Found Events. Increments every time an **Amble Found** interrupt is posted regardless of whether that interrupt is masked or already set. Useful to measure amble lengths on tape.
11	Rewrites. Increments every time a **Rewrite** interrupt is posted. Useful to quantify quality of data written to tape.

CNTPS - Counter Event Prescale. Provides a selectable divisor for the input source. Especially useful when **SYSCLK** is the event source.

CTPA[1:0]	DIVISOR
00	1
01	4
10	64
11	1024

11.6 HOST REGISTERS

11.6.1 HOST DMA A ADDRESS

Read/Write
Reset Value (hex) = XXXXXX

	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
0x30	HDAA[7:0]							
0x31	HDAA[15:8]							
0x32	reserved			HDAA[21:16]				
0x33	reserved							

HDAA[21:0]-Host DMA **A** Address. Contains the address for the next DRAM location to be accessed by pointer **A**. This value is incremented after each DRAM access. If control byte skipping is enabled with **HBM**, the incrementing of this address will skip the DRAM locations reserved for the blocks control bytes. In this mode, the control byte offset (8, in the case of QIC-3095) is added to the initial value programmed by the microprocessor when the DMA is enabled. The control byte offset is then added after each physical block (512 for QIC-3095). This register should only be written when host DMA **A** is disabled. It can be read at any time but since it is actively used during the DMA, the value read by sequential reads may not accurately reflect the value in the register. This is only an issue when a DMA transfer occurs between the sequential reads.

11.6.2 HOST DMA A COUNTER

Read/Write
Reset Value (hex) = XXXX

	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
0x34	HDAC[7:0]							
0x35	HDAC[15:0]							

HDAC[15:0]-Host DMA **A** Counter. Contains the number of bytes remaining in the DMA transfer. The register can be programmed by the microprocessor only when DMA **A** is disabled. It is decremented for every byte transferred by DMA **A**. When the counter expires, DMA **A** is automatically disabled and the **Host DMA A Done** interrupt is generated. This register can be read at any time but since it is actively used during the DMA, the value read by sequential reads may not accurately reflect the value in the register.

11.6.3 HOST DMA B ADDRESS

Read/Write
Reset Value (hex) = XXXXXX

	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
0x38	HDBA[7:0]							
0x39	HDBA[15:8]							
0x3A	reserved			HDBA[21:16]				
0x3B	reserved							

HDBA[21:0]-Host DMA **B** Address. Contains the address for the next DRAM location to be accessed by pointer **B**. This value is incremented after each DRAM access. If control byte skipping is enabled with **HBM**, the incrementing of this address will skip the DRAM locations reserved for the blocks control bytes. In this mode, the control byte offset (8, in the case of QIC-3095) is added to the initial value programmed by the microprocessor when the DMA is enabled. The control byte offset is then added after each physical block (512 for QIC-3095). This register should only be written when host DMA **B** is disabled. It can be read at any time but since it is actively used

during the DMA, the value read by sequential reads may not accurately reflect the value in the register. This is only an issue when a DMA transfer occurs between the sequential reads.

11.6.4 HOST DMA B COUNTER

Read/Write
Reset Value (hex) = XXXX

	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
0x3C	HDBC[7:0]							
0x3D	HDBC[15:0]							

HDBC[15:0]-Host DMA **B** Counter. Contains the number of bytes remaining in the DMA transfer. The register can be programmed by the microprocessor only when DMA **B** is disabled. It is decremented for every byte transferred by DMA **B**. When the counter expires, DMA **B** is automatically disabled and the **Host DMA B Done** interrupt is generated. This register can be read at any time but since it is actively used during the DMA, the value read by sequential reads may not accurately reflect the value in the register.

11.6.5 HOST INTERFACE INTERRUPTS

Reading this register returns the current value of each of the interrupts. The setting of interrupt bits in this registers is not effected by the current interrupt mask setting. The interrupt mask only effects the generation of the interrupts signal.

Interrupts are cleared by writing a '1' to the corresponding bit in this register. Writing a '0' has no effect on the bit.

Read/Write
Reset Value (hex) = 00

	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
0x70	reserved				HDBDI	HDADI	HBPEI	HDPEI

HDBDI - **Host DMA B Done** Interrupt. Generated when after the last byte in host DMA **B** is transferred to or from the external DRAM buffer. Host DMA **B** is disabled when this interrupt is generated. The Host DMA **B** counter and address can be written when the DMA is disabled.

HDADI - **Host DMA A Done** Interrupt. Generated when after the last byte in host DMA **A** is transferred to or from the external DRAM buffer. Host DMA **A** is disabled when this interrupt is generated. The Host DMA **A** counter and address can be written when the DMA is disabled.

HBPEI - **Host Bus Parity Error** Interrupt. Generated when a parity error is detected on the host data bus. Parity is checked at the pins. If a parity error occurs, the active DMA channel is halted.

HDPEI - **Host DRAM Parity Error** Interrupt. Generated when a parity error is detected when the host channel is reading DRAM. If a parity error occurs, the active DMA channel is halted. The microprocessor can determine which byte in the transfer had the parity error by reading the DMA counter for the active channel.

11.6.6 HOST INTERFACE INTERRUPTS MASK

Writing a '1' to the mask bit for an interrupt prevents that interrupt from causing the interrupt signal to be asserted. Masking an interrupt does not effect the setting of the status register. A '0' in the mask bit allows the corresponding interrupt to assert the interrupt signal. Reading this register returns the current mask settings.

Read/Write
Reset Value (hex) = FF

	bit7	*bit6*	*bit5*	*bit4*	*bit3*	*bit2*	*bit1*	*bit0*
0x78	reserved				HDBDM	HDADM	HBPEM	HDPEM

HDBDM - Mask bit for **Host DMA B Done** Interrupt. (**HDBDI**)

HDADM - Mask bit for **Host DMA A Done** Interrupt. (**HDADI**)

HBPEM - Mask bit for **Host Bus Parity Error** Interrupt. (**HBPEI**)

HDPEM - Mask bit for **Host DRAM Parity Error** Interrupt. (**HDPEI**)

11.6.7 HOST COMMAND

Commands are initiated by writing a '1' to the corresponding bit. Writing a '0' has no effect on a command that may be executing. The status of a command is determined by reading the *Host Status* register. Writing a '1' to a command that is already executing may either have no effect or, if the command happened to be just completing, it would cause the command to begin again.

Write Only

	bit7	*bit6*	*bit5*	*bit4*	*bit3*	*bit2*	*bit1*	*bit0*
0x42	reserved			RST	HBE	HBD	HAE	HAD

RST - Reset. Writing a '1' to this bit initiates a reset of the data path elements and state machines in the Host Interface. Writing a '0' has no effect.

HBE - ***Host B Enable***. Writing a '1" to this bit enables DMA **B**. **B** remains enabled until it is automatically disabled when the **Host DMA B Done** interrupt occurs or until the microprocessor disables the DMA with a ***Host B Disable*** command.

HBD - ***Host B Disable***. Writing a '1" to this bit disables DMA **B**. A DMA typically only needs to be disabled as part of an error recovery scheme. If DMA **B** is active when this command is received, the DMA transfer will stop and if DMA **A** is enabled, it will immediately start. If DMA **B** is not active, this command prevents it from starting.

HAE - ***Host A Enable***. Writing a '1' to this bit enables DMA **A**. **A** remains enabled until it is automatically disabled when the **Host DMA A Done** interrupt occurs or until the microprocessor disables the DMA with a ***Host A Disable*** command.

HAD - ***Host A Disable***. Writing a '1' to this bit disables DMA **A**. A DMA typically only needs to be disabled as part of an error recovery scheme. If DMA **A** is active when this command is received, the DMA transfer will stop and if DMA **B** is enabled, it will immediately start If DMA **A** is not active, this command prevents it from starting.

11.6.8 HOST STATUS

This register reports the status of the current operations. It can be read at any time.

Read Only
Reset Value (binary) = xxx1 0000

	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
0x42	reserved				HBS[1:0]		HAS[1:0]	

HBS - Host DMA **B** Status. Reports the status of the DMA **B**.

HBS[1:0]	STATE	DESCRIPTION
00	**B Disabled**	DMA **B** is disabled. This state is active after reset, after a DMA successfully completes or after a ***Host B Disable*** command. Will remain in this state until a ***Host B Enable*** command is received. This is the only state in which *Host DMA B Address* and *Host DMA B Count* can be written.
01	**B Enabled**	DMA **B** is enabled and waiting to become active. The DMA will become active as soon as DMA **A** is disabled. DMA **B** can be disabled with a ***Host B Disable*** command. *Host DMA B Address* and *Host DMA B Count* are stable for reading in this state.
10	**B Active**	When in this state, DMA **B** is actively being used for DMA transfers. This state will remain active until the DMA successful completes, a DMA error (such as parity error) is encountered or until a ***Host B Disable*** command is received. *Host DMA B Address* and *Host DMA B Count* can be read in this state however the registers are actively being used and may be in the process of transitioning.
11	**B Error**	This state is only active after an error in the DMA operation is encountered. The source of the error is reported in the *Host Interrupts* register. When in this state, the DMA is considered to be still enabled but halted. The only means of restarting host DMA transfers is for the *Host B Disable* command to be received. *Host DMA B Address* and *Host DMA B Count* are stable can be read in this state. The value in *Host DMA B Count* will indicate location at which the DMA error occurred.

HAS - Host DMA **A** Status. Reports the status of the DMA **A**.

HAS[1:0]	STATE	DESCRIPTION
00	**A Disabled**	DMA **A** is disabled. This state is active after reset, after a DMA successfully completes or after a ***Host A Disable*** command. Will remain in this state until a ***Host A Enable*** command is received. This is the only state in which *Host DMA A Address* and *Host DMA A Count* can be written.
01	**A Enabled**	DMA **A** is enabled and waiting to become active. The DMA will become active as soon as DMA **B** is disabled. DMA **A** can be disabled with a ***Host A Disable*** command. *Host DMA A Address* and *Host DMA A Count* are stable for reading in this state.
10	**A Active**	When in this state, DMA **A** is actively being used for DMA transfers. This state will remain active until the DMA successful completes, a DMA error (such as a parity error) is encountered or until a ***Host A Disable*** command is received. *Host DMA A Address* and *Host DMA A Count* can be read in this state however the registers are actively being used and may be in the process of transitioning.

HAS[1:0]	STATE	DESCRIPTION
11	**A Error**	This state is only active after an error in the DMA operation is encountered. The source of the error is reported in the *Host Interrupts* register. When in this state, the DMA is considered to be still enabled but halted. The only means of restarting host DMA transfers is for the ***Host A Disable*** command to be received. *Host DMA A Address* and *Host DMA A Count* are stable can be read in this state. The value in *Host DMA A Count* will indicate location at which the DMA error occurred.

11.6.9 HOST CONFIGURATION 1

Read/Write
Reset Value (hex) = 48

	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
0x40	reserved	**HAL[1:0]**		**HAH[1:0]**		**HRAL**	**HPE**	**HRD**

HAL[1:0] - Host **HACKN** Active (low) pulsewidth. The host acknowledge signal is held low for the binary encoded value specified in this register. This value is measured in number of **SYSCLK**s. The value of '00' is undefined and should not be used when the host port is enabled.

HAH[1:0] - Host **HACKN** Inactive (high) pulsewidth. The host acknowledge signal is held high for the binary encoded value specified in this register. This value is measured in number of **SYSCLK**s. The value of '00' is undefined and should not be used when the host port is enabled.

HRAL - Host Read Active Low. When this bit is set the **HRD** signal is active low (i.e. low when reading from DRAM, high when writing to DRAM). When clear, the **HRD** is active high. Note that this signal does *not* affect the polarity of the **HRD** register bit.

HPE - Host Parity Enable. When active, parity generation and checking is enabled on the ICs host port. Only odd parity is supported.

HRD - Host read. Selects the host data transfer direction. When active, the data is read from DRAM. Data is written to DRAM when the bit is low.

11.6.10 HOST CONFIGURATION 2

Read/Write
Reset Value (hex) = 00

	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
0x41	reserved			**HBC[3:0]**		reserved	**COPY**	**HBM**

HBC[3:0] - Host Byte Count. Number of bytes in host channel.

COPY - Memory Copy Mode. In this mode, DRAM to DRAM copies are allowed with *DMA A* as the source and *DMA B* as the destination. Both *DMA A* and *DMA B* must be in the 'disabled' state. This is determined by reading **HAS** (Host DMA A Status) and **HBS** (Host DMA B Status). Copy mode is enabled when this register is active. When inactive, normal host DMAs are enabled.

HBM - Host Block Mode. When active block mode transfers are enabled. In block mode transfers, the DMA address register skips memory locations that are reserved for the control bytes for the block. The number of locations that are skipped and the number of bytes between the skips depends upon the tape format programmed in the *Tape Format* register. The value in this register applies to both DMA channels. This register may only be changed with both DMA channels are disabled.

11.7 CLOCK CONFIGURATION REGISTERS

11.7.1 SYSTEM PLL CONTROL

This register is asynchronously reset on active edge of **RESETN**. No other reset effects the contents of this register. Regardless of the relative frequency of the microprocessor and the AHA5140, this register can be written. However, it can only be read according to the microprocessor read timing diagrams.

Read/Write
Reset Value (hex) = 0x0000

	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
0x04	reserved	S-XTL	reserved			SD[3:0]		
0x05	S-OFF	reserved				SN[2:0]		

S-XTL - System Crystal Control. When set, system PLL is fed from **SCI**. When the bit is clear, system oscillator output feeds the system PLL.

SD[3:0] - Denominator for system PLL frequency generation.

S-OFF - System PLL Control. When set, system PLL is off. When cleared, PLL is on.

SN[2:0] - Numerator for system PLL frequency generation.

11.7.2 TAPE PLL CONTROL

This register is asynchronously reset on active edge of **RESETN**. No other reset effects the contents of this register. Regardless of the relative frequency of the microprocessor and the AHA5140, this register can be written. However, it can only be read according to the microprocessor read timing diagrams.

Read/Write
Reset Value (hex) = 0x0000

	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
0x06	reserved	T-XTL	reserved			TD[4:0]		
0x07	T-OFF	reserved				TN[2:0]		

T-XTL - Tape Crystal Control. When set, tape PLL is allowed to be fed from **WCI**. When the bit is clear, the tape oscillator output feeds the tape PLL.

TD[4:0] - Denominator for tape PLL frequency generation.

T-OFF - Tape PLL Off. When set, tape PLL is off. When cleared, PLL is on.

TN[2:0] - Numerator for tape PLL frequency generation.

11.7.3 CLOCK CONFIGURATION

Read/Write
Reset Value (hex) = 0x0000

	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
0x03	T_DIV[1:0]		reserved	S_XTLO	SYSTEM_ ENABLE	TAPE_ ENABLE	USE_SYS	LOOP

T_DIV[1:0] -Selects which PLL output is used for the tape clock.

T_DIV[1:0]	*W6XCLK SOURCE*
00	VCO/4
01	VCO/2
10	VCO
11	VCO

Must be programmed to output VCO if **TAPE_ENABLE** is used to bypass the tape PLL.

S_XTLO - When high selects source for **SYSCLK** driver to be **SCI**. When low the source for **SYSCLK** is the output of the system oscillator. This signal only effects **SYSCLK** when **SYSTEM_ENABLE** is low.

SYSTEM_ENABLE -When set the System PLL is enabled to drive **SYSCLK**. When the bit is cleared, **SYSCLK** is driven depending upon **S_XTLO**. When this bit changes, a 15 sci_clk pulse is generated on **RESETO** pin. In addition, a soft chip reset cycle is performed.

TAPE_ENABLE -When set (and **T_DIV** is programmed to output VCO) the tape PLL is in bypass mode. In bypass mode, the input to the PLL is fed directly to W6XCLK. When cleared the tape PLL is enabled to drive W6XCLK.

USE_SYS - When set the input to the tape PLL is driven from **SYSCLK**. When the bit is cleared, **WCI** drives the tape PLL.

LOOP - Tape loopback mode is enabled when this bit is set. In loop back mode **WRDATA** drives to **RDATA** and **WRCLK** to **RDCLK**. When loop back mode is disabled, **RDATA** and **RDCLK** are driven with the value at the pins.

11.8 ECC REGISTERS

11.8.1 ECC DMA ADDRESS

Read/Write
Reset Value (hex) = XXXXXX

	bit7	*bit6*	*bit5*	*bit4*	*bit3*	*bit2*	*bit1*	*bit0*
0x24	**EDA[7:0]**							
0x25	**EDA[15:8]**							
0x26	reserved			**EDA[21:16]**				
0x27	reserved							

EDA[21:0] - ECC DMA Address. Contains the address for the ECC DMA access. This register is normally programmed by the microprocessor with the address of control byte 0 for the first block in the interleave (see Figure 4 and Figure 5 on page 9). However, ECC can be performed on a partial interleave. When this is being done, the address programmed must only be in the first block of the interleave. The internal control for this register automatically accesses the bytes in the codeword in the proper order. In the case of QIC-3095, the offset between consecutive bytes in the codeword is 1040 (8 control bytes + 512 data bytes × 2 interleaves).

11.8.2 ECC ERASURE ADDRESS

Read/Write
Reset Value (hex) = XX

	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
0x28	EEA[7:0]							
0x29	EEA[15:8]							
0x2A	reserved	INTR	EEA[21:16]					
0x2B	reserved							

INTR - ECC Interleave selection. Bit is high when processing the odd number blocks and low when processing the even blocks.

EEA[21:0] -ECC Erasure Address. Programmed by the microprocessor to contain the starting address for the ECC erasure information. The erasure information is written to memory while reading tape and is based upon the CRC checks for the blocks in the frame. The starting address for a frames erasure vector is fixed relative to the frame. It begins immediately after the last byte in frame and has one bit for each block. The erasure flags for consecutive blocks are stored in consecutive bits. If the erasure bit is set, the corresponding block is flagged with a CRC error. This register must be re-programmed after decoding of a interleave.

11.8.3 ECC CODEWORDS

Read/Write
Reset Value (hex) = XXXX

	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
0x2D	reserved						ECWS[9:8]	
0x2C	ECWS[7:0]							

ECWS[9:0] -Specifies the number of ECC codewords to encode or decode. When encoding or decoding an entire interleave, this register should be programmed to 513(dec)=201(hex). However, any part of a interleave all the way down to a single codeword can be done in a single operation. This register is decremented upon the correct completion of every codeword; it is not decremented for uncorrectable codeword. When the register decrements to zero, an **ECC DONE** interrupt is generated.

11.8.4 ECC CONFIGURATION

Read/Write
Reset Value (hex) = 00

	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
0x2F	reserved[6:0]							ENCODE

reserved[6:0] -ECC reserved. Reserved for internal use.

ENCODE - ECC Encoding. When active, ECC encoding is performed. ECC Decoding is selected when this register is inactive. This register may only be changed when the ECC is idle.

11.8.5 ECC COMMAND

Commands are initiated by writing a '1' to the corresponding bit. Writing a '0' has no effect on a command that may be executing. The status of a command is determined by reading the *ECC Status* register. Writing a '1' to a command that is already executing my either have no effect or, if the command happened to be just completing, it would cause the command to begin again.

Write Only

	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
0x2E	reserved						**STOP**	**START**

STOP - Causes an active ECC operation to immediately stop. The codeword being encoded or decoded when an operation is terminated may be only partially complete. The partially complete codeword is corrupted and cannot be re-operated upon by the ECC. This command acts as a module reset.

START - Starts ECC encoding or decoding operation, depending upon the setting of the **ENCODE** register. The operation will continue until it completes, encounters and uncorrectable codeword or is halted with **STOP** command. The current status of the ECC operation can be determined from the *ECC Status* register.

11.8.6 ECC STATUS

This register reports the status of the current operations. It can be read at any time.

Read Only
Reset Value (binary) = 0000 0000

	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
0x2E	reserved						**PHASE[1:0]**	

PHASE[1:0]-Reports current status of the ECC operation.

PHASE [1:0]	STATE	DESCRIPTION
00	**ECC IDLE**	No ECC operation is in progress. All ECC registers are stable for reading or writing.
01	**ECC INIT**	An ECC operation has been started and is the processor initializing itself. During decoding, this is the state where the ECC Erasure information of the interleave is loaded.
10	**ECC In**	An encoding or decoding operation is reading a codeword from memory. The codeword being read can be determined from the **ECWS** register.
11	**ECC Out**	An encoding operation is outputting ECC Bytes or the decoding operation is outputting corrections. The active codeword can be determined from the **ECWS** register. At the completion of this state an *ECC Codeword Done* interrupt is generated.

11.8.7 ECC AND OTHER INTERRUPTS

Reading this register returns the current value of each of the interrupts. The setting of interrupt bits in this registers is not effected by the current interrupt mask setting. The interrupt mask only effects the generation of the interrupts signal.

Interrupts are cleared by writing a '1' to the corresponding bit in this register. Writing a '0' has no effect on the bit.

Read/Write
Reset Value (hex) = 00

	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
0x71	EDI	ECWDI	ECWUI	EPEI	SEPEI	DAPEI	reserved	TIMI

EDI - **ECC Done** Interrupt. Generated after the completion of the last codeword in the ECC operation.

ECWDI - **ECC Codeword Done** Interrupt. Generated after the completion of each ECC codeword.

ECWUI - **ECC Codeword Uncorrectable** Interrupt. Generated when decoding and an uncorrectable codeword is detected. The codeword number that caused the interrupt can be determined from the *ECC Codewords* register and the address for the beginning of the codeword is in the *ECC DMA Address* register.

EPEI - ECC DRAM Parity Error Interrupt. Generated when a parity error is detected when the ECC channel is reading DRAM. The ECC channel reads DRAM during state *ECC Init*, *ECC In* and when decoding, *ECC Out*. The microprocessor can determine which ECC codeword contains the parity error by reading the *ECC Codewords* register.

SEPEI - **SRAM Emulation Parity Error**. A parity error occurred while reading DRAM as part of an SRAM emulation read. Failed location is *DRAM Address* - 1.

DAPEI - **DRAM Access Parity Error**. A parity error occurred while indirectly reading DRAM through *DRAM Address* and *DRAM Data*. Failed location is contained in DRAM Address if the last operation was a write to *DRAM Address*. Otherwise it is *DRAM Address* - 1.

TIMI - **Timer**. The *Counter Timer* register rolled over from 0xFFFF to 0x0000.

11.8.8 ECC AND OTHER INTERRUPTS MASK

Writing a '1' to the mask bit for an interrupt prevents that interrupt from causing the interrupt signal to be asserted. Masking an interrupt does not effect the setting of the status register. A '0' in the mask bit allows the corresponding interrupt to assert the interrupt signal. Reading this register returns the current mask settings.

Read/Write
Reset Value (hex) = FF

	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
0x79	EDM	ECWDM	ECWUM	EPEM	SEPEM	DAPEM	reserved	TIMM

EDM - Mask bit for **ECC Done. (EDI)**

ECWDM - Mask bit for **ECC Codeword Done. (ECWDI)**

ECWUM - Mask bit for **ECC Codeword Uncorrectable. (ECWUI)**

EPEM - Mask bit for **ECC DRAM Parity Error. (EPEI)**

SEPEM - Mask bit for **SRAM Emulation Parity Error. (SEPEI)**

DAPEM - Mask bit for **DRAM Access Parity Error. (DAPEI)**

TIMM - Mask bit for **Timer. (TIMI)**

11.8.9 ECC FORMAT

Read/Write
Reset Value (hex) = XX

	bit7	*bit6*	*bit5*	*bit4*	*bit3*	*bit2*	*bit1*	*bit0*
0x4D	reserved					**FSIZ**	**ECC-FRMT[1:0]**	

FSIZ - ECC Frame Size. If set, the frame has 128 blocks. If cleared, the frame has 64 blocks.

ECC-FRMT[1:0]-ECC Format. Sets level of redundancy for the ECC code.

ECC-FRMT[1:0]	*REDUNDANCY*
00	6
10	10
11	reserved

12.0 SIGNAL DESCRIPTIONS AND SPECIFICATIONS

Unused input pins should be tied inactive through a resistor.

12.1 SYSTEM SIGNALS

SIGNAL	TYPE	DESCRIPTION
SCI	I	Input for external crystal. Clock input when external oscillator used.
SCIXT	O	Output for external crystal. Leave open if no crystal used.
SYSCLK	O	System Clock Output.
RESETN	I	Active Low System Reset.
TRISN	I	Active Low Tristate. Asynchronously forces all outputs into high impedance.
RESETO	O	Active low when **RESETN** is active or when **SYSTEM_ENABLE** bit is changed by the microprocessor. This signal should be used as the power-on-reset for any device using **SYSCLK** for a clock. The reason for this is that **SYSCLK** may glitch anytime **SYSTEM_ENABLE** is changed.

12.2 MICROPROCESSOR SIGNALS

SIGNAL	TYPE	DESCRIPTION
UPCSN	I	Active Low Chip Select, Register Accesses. Active when microprocessor is performing a read or write to internal registers.
UPWRN	I	Active Low Write Enable. Combined with **UPCSN** indicates a write to the chip.
UPRDN	I	Active Low Read Enable. Combined with **UPCSN** indicates a read from the chip.
UPSCSN	I	Active Low Chip Select, SRAM Emulation Accesses. Active when microprocessor is performing a read or write to the emulated SRAM.
UPALE	I	Microprocessor Address Latch Enable. Active high signal which latches the contents of **UPA[15:8]** and **UPAD[7:0]**.
UPA[15:8]	I	Microprocessor Address. Contains upper byte of address for SRAM emulation accesses.
UPAD[7:0]	I/O	Microprocessor Address/Data. Contains Address from microprocessor followed by data going to and from microprocessor. Driven when **UPRDN** and **UPCSN** or **UPSCSN** are active.
UPINTI	O	Active High Interrupt Request. High whenever an unmasked bit in any interrupt register is set.
UPRDY	OD	Active Chip Ready. When low, indicates the chip needs more time to complete an access. When high chip has completed access. Goes active immediately after **UPCSN** or **UPSCSN** and releases synchronous to **SYSCLK**.

12.3 HOST SIGNALS

The Host signals are only available when the SCSI control is disabled. When the SCSI controller is enabled, these signals are redefined as SCSI Signals.

SIGNAL	TYPE	NAME AND DESCRIPTION
HD[7:0]	I/O	Host data bus. Bus is driven whenever the host port is configured to be read with the **HRD** bit in the *Host Configuration 1* register. Write data is sampled on **SYSCLK** when **HACKN** is deasserted. Read data is driven from the active edge of **HACKN**.
HDP	I/O	Host data bus parity. Optional odd parity associated with **HD** bus. Driven and latched with **HD**. If parity is disabled, this signal must be tied low.
HACKN	O	Host acknowledge output. Strobes data into and out of the device. The active and inactive pulsewidths are programmed in the *Host Configuration 1* register. Active low.
HREQ	I	Host request input. Treated as asynchronous input until an active edge is detected. During a host transfer, this input is sampled synchronously on the rising edge of **SYSCLK** that generates the inactive edge of **HACKN**. Once this is sampled inactive, an asynchronous rising edge detect is again used. Active high.
HRD	O	Host Read. Active when the host port is being read by an external device. Inactive when the host port is being written. Driven directly from the **HRD** bit in the *Host Configuration 1* register.

12.4 TAPE SIGNALS

SIGNAL	TYPE	DESCRIPTION
WCI	I	Write Clock Input. Can be driven directly by an external oscillator or connected to a crystal along with **WCIXT**.
WCIXT	O	Write Clock Crystal. Connected to external crystal along with **WCI**. If an external oscillator is used, **WCIXT** should remain unconnected.
WRCLK	O	Write Clock Output. Frequency is three time the tape bit rate
WRDATA	O	Write Data Output. Clocked off falling edge of **WRCLK**.
WGATEN	O	Write Gate Not. Active low when tape channel is in Write or Read-While-Write.
BITCLK	O	Bit Rate Clock. Provides bit rate reference for external read PLL.
RDCLK	I	Read Clock Input. Used to clock read data.
RDATA	I	Read Data Input. Format is NRZI or raw data as per *Tape Control* register. Values are sampled on rising edge of **RDCLK**.
PLLSYNC	O	Output to read data recovery IC. Switches data recovery from high gain to tracking mode.
RGATE	O	Output to read data recovery IC. Switches data recovery circuit from fixed to closed-loop clock generation.
TAPE[1:0]	O	Tape Status Outputs. Output value depends upon programming in *Tape Test Configuration* register.

12.5 DRAM SIGNALS

SIGNAL	TYPE	DESCRIPTION
MRASN	O	DRAM Row Address Strobe.
MCASN	O	DRAM Column Address Strobe.
MWEN	O	DRAM Active Low Write Enable.
MOEN	O	DRAM Output Enable.
MA[11:0]	O	DRAM Address Bus.
MD[7:0]	I/O	DRAM Data Bus.
MDP	I/O	DRAM Data Bus Parity. If no parity DRAM exists. Parity must be disabled with DPE and this signal must be unconnected.

12.6 OTHER SIGNALS

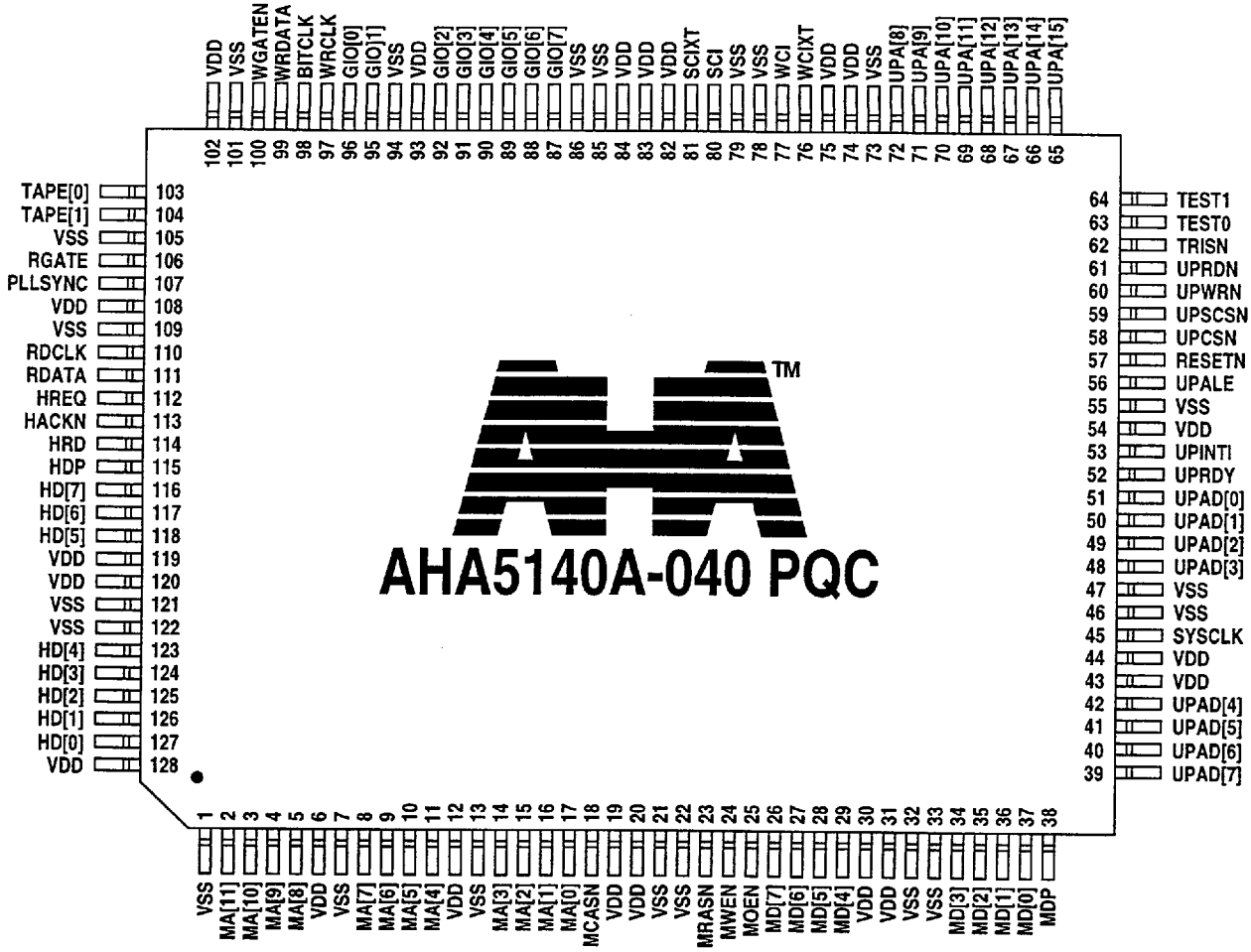
SIGNAL	TYPE	DESCRIPTION
GIO[7:0]	I/O	General I/O Port.

13.0 PINOUT

Please consult with Advanced Hardware Architectures for the most up-to-date pinout and packaging information.

PIN	SIGNAL	TYPE	LOAD	PIN	SIGNAL	TYPE	LOAD	PIN	SIGNAL	TYPE	LOAD
1	VSS	POWER	n/a	44	VDD	POWER	n/a	87	GIO[7]	CMOS	50
2	MA[11]	TTL (4mA)	40	45	SYSCLK	CMOS	50	88	GIO[6]	CMOS	50
3	MA[10]	TTL (4mA)	40	46	VSS	POWER	n/a	89	GIO[5]	CMOS	50
4	MA[9]	TTL (4mA)	40	47	VSS	POWER	n/a	90	GIO[4]	CMOS	50
5	MA[8]	TTL (4mA)	40	48	UPAD[3]	TTL (4mA)	60	91	GIO[3]	CMOS	50
6	VDD	POWER	n/a	49	UPAD[2]	TTL (4mA)	60	92	GIO[2]	CMOS	50
7	VSS	POWER	n/a	50	UPAD[1]	TTL (4mA)	60	93	VDD	POWER	n/a
8	MA[7]	TTL (4mA)	40	51	UPAD[0]	TTL (4mA)	60	94	VSS	POWER	n/a
9	MA[6]	TTL (4mA)	40	52	UPRDY	TTL (8mA) od	60	95	GIO[1]	CMOS	50
10	MA[5]	TTL (4mA)	40	53	UPINTI	TTL (4mA)	60	96	GIO[0]	CMOS	50
11	MA[4]	TTL (4mA)	40	54	RESETO			97	WRCLK	CMOS	35
12	VDD	POWER	n/a	55	VDD	POWER	n/a	98	BITCLK	CMOS	35
13	VSS	POWER	n/a	56	VSS	POWER	n/a	99	WRDATA	CMOS	35
14	MA[3]	TTL (4mA)	40	57	UPALE	TTL	input	100	WGATEN	CMOS	35
15	MA[2]	TTL (4mA)	40	58	RESETN	CMOS s	input	101	VSS	POWER	n/a
16	MA[1]	TTL (4mA)	40	59	UPCSN	TTL	input	102	VDD	POWER	n/a
17	MA[0]	TTL (4mA)	40	60	UPSCSN	TTL	input	103	TAPE[0]	TTL (2mA)	35
18	MCASN	TTL (12mA)	50	61	UPWRN	TTL	input	104	TAPE[1]	TTL (2mA)	35
19	VDD	POWER	n/a	62	UPRDN	TTL	input	105	VSS	POWER	n/a
20	VDD	POWER	n/a	63	TRISN	CMOS s	input	106	RGATE	CMOS	35
21	VSS	POWER	n/a	64	TEST1	TTL	input	107	PLLSYNC	CMOS	35
22	VSS	POWER	n/a	65	UPA[15]	TTL	input	108	VDD	POWER	n/a
23	MRASN	TTL (12mA)	50	66	UPA[14]	TTL	input	109	VSS	POWER	n/a
24	MWEN	TTL (4mA)	50	67	UPA[13]	TTL	input	110	RDCLK	CMOS	input
25	MOEN	TTL (4mA)	50	68	UPA[12]	TTL	input	111	RDATA	CMOS	input
26	MD[7]	TTL (4mA)	50	69	UPA[11]	TTL	input	112	HREQ	TTL	input
27	MD[6]	TTL (4mA)	50	70	UPA[10]	TTL	input	113	HACKN	TTL (4mA)	35
28	MD[5]	TTL (4mA)	50	71	UPA[9]	TTL	input	114	HRD	TTL (4mA)	35
29	MD[4]	TTL (4mA)	50	72	UPA[8]	TTL	input	115	HDP	TTL (4mA)	35
30	VDD	POWER	n/a	73	VSS	POWER	n/a	116	HD[7]	TTL (4mA)	35
31	VDD	POWER	n/a	74	VDD	POWER	n/a	117	HD[6]	TTL (4mA)	35
32	VSS	POWER	n/a	75	VDD	POWER	n/a	118	HD[5]	TTL (4mA)	35
33	VSS	POWER	n/a	76	WCIXT	CRYSTAL	input	119	VDD	POWER	n/a
34	MD[3]	TTL (4mA)	50	77	WCI	CRYSTAL	n/a	120	VDD	POWER	n/a
35	MD[2]	TTL (4mA)	50	78	VSS	POWER	n/a	121	VSS	POWER	n/a
36	MD[1]	TTL (4mA)	50	79	VSS	POWER	n/a	122	VSS	POWER	n/a
37	MD[0]	TTL (4mA)	50	80	SCI	CRYSTAL	input	123	HD[4]	TTL (4mA)	35
38	MDP	TTL (4mA)	50	81	SCIXT	CRYSTAL	n/a	124	HD[3]	TTL (4mA)	35
39	UPAD[7]	TTL (4mA)	60	82	VDD	POWER	n/a	125	HD[2]	TTL (4mA)	35
40	UPAD[6]	TTL (4mA)	60	83	VDD	POWER	n/a	126	HD[1]	TTL (4mA)	35
41	UPAD[5]	TTL (4mA)	60	84	VDD	POWER	n/a	127	HD[0]	TTL (4mA)	35
42	UPAD[4]	TTL (4mA)	60	85	VSS	POWER	n/a	128	VDD	POWER	n/a
43	VDD	POWER	n/a	86	VSS	POWER	n/a				

Figure 13: Pinout



14.0 DC ELECTRICAL CHARACTERISTICS

ABSOLUTE MAXIMUM STRESS RATINGS					
SYMBOL	CHARACTERISTICS	MIN	MAX	UNITS	TEST CONDITIONS
Tstg	Storage temperature	-55	150	°C	
Vdd	Supply voltage	-0.5	6.0	V	
Vin	Input voltage	Vss-0.5	Vdd+0.5	V	

OPERATING CONDITIONS					
SYMBOL	CHARACTERISTICS	MIN	MAX	UNITS	TEST CONDITIONS
Vdd	Supply voltage	4.75	5.25	V	
Idd	Supply current		20	mA	Static
Idd	Supply current		320	mA	Dynamic – with typical operation
Ta	Operating temperature	0	70	°C	

INPUTS (TTL)					
SYMBOL	CHARACTERISTICS	MIN	MAX	UNITS	TEST CONDITIONS
Vih	Input high voltage	2.0	Vdd+0.5	V	
Vil	Input low voltage	Vss-0.5	0.8	V	
Iil	Input leakage	-10	10	µA	0<Vin<Vdd
Cin	Self load capacitance		10	pF	

OUTPUTS (TTL)					
SYMBOL	CHARACTERISTICS	MIN	MAX	UNITS	TEST CONDITIONS
Voh	Output high voltage	2.4	Vdd	V	Ioh=4mA
Vol	Output low voltage	Vss	0.4	V	Iol=4mA
Vol	Output low voltage – UPRDY	Vss	0.5	V	Iol=4mA
Ioh	Output high current	-4		mA	Voh=2.4V
Iol	Output low current		4	mA	Vol=0.4
Ioz	High impedance leakage		10	µA	0<Vout<Vdd
Cout	Self load capacitance		10	pF	
Cl	Load capacitance			pF	See Table

INPUTS (CMOS)					
SYMBOL	CHARACTERISTICS	MIN	MAX	UNITS	TEST CONDITIONS
Vih	Input high voltage	0.7×Vdd	Vdd+0.5	V	
Vil	Input low voltage	Vss-0.5	0.3×Vdd	V	
Iil	Input leakage	-10	10	µA	0<Vin<Vdd
Cin	Self load capacitance		10	pF	

INPUTS (CRYSTAL – SCI, WCI)					
SYMBOL	CHARACTERISTICS	MIN	MAX	UNITS	TEST CONDITIONS
Vih	Input high voltage	0.85×Vdd	Vdd+0.5	V	
Vil	Input low voltage	Vss-0.5	0.15×Vdd	V	

OUTPUTS (CMOS)					
SYMBOL	CHARACTERISTICS	MIN	MAX	UNITS	TEST CONDITIONS
Voh	Output high voltage	0.7×Vdd	Vdd	V	
Vol	Output low voltage	Vss	0.3×Vdd	V	
Ioz	High impedance leakage		10	µA	
Cout	Self load capacitance		10	pF	
Cl	Load capacitance			pF	See Table

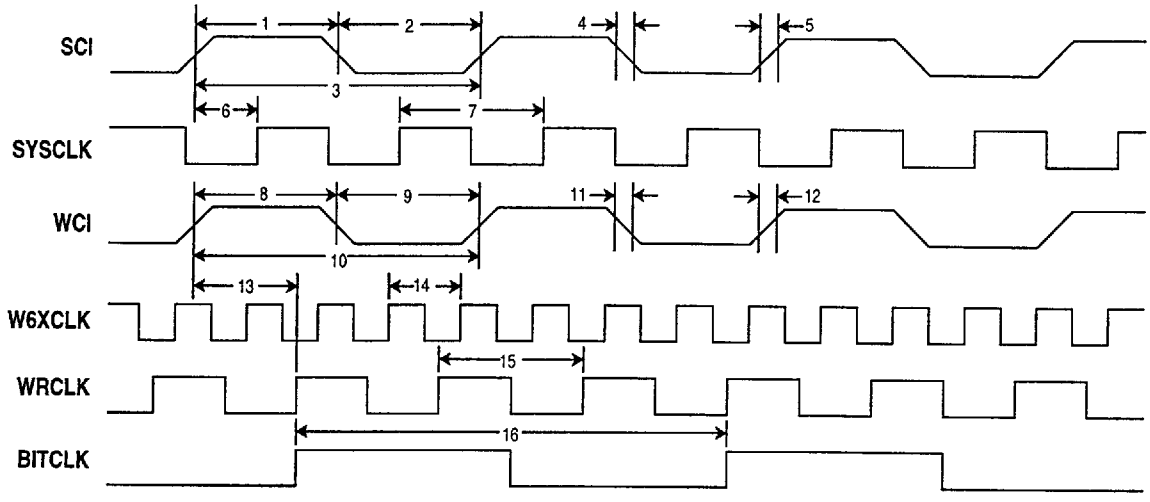
15.0 AC ELECTRICAL CHARACTERISTICS

Notes:

1) Unless otherwise specified, all timing measurements are measured from 1.5V.

2) Unless otherwise specified, the maximum transition time (Vil to Vih) on inputs is 3ns.System Timing

15.1 CLOCK TIMING



NUMBER	PARAMETER	MIN	MAX	UNITS	NOTES
1	**SCI** High Time	10		ns	1
2	**SCI** Low Time	10		ns	1
3	**SCI** Period	25		ns	1
4	**SCI** Fall Time		4	ns	1
5	**SCI** Rise Time		4	ns	1
6	**SCI** to **SYSCLK**		15	ns	1, 2
			15	ns	1, 3
7	**SYSCLK** Period	25	25	ns	4
8	**WCI** High Time	5		ns	5
9	**WCI** Low Time	5		ns	5
10	**WCI** Period	16		ns	5
11	**WCI** Fall Time		4	ns	5
12	**WCI** Rise Time		4	ns	5
13	**WCI** to **WRCLK**		25	ns	5, 6
			25	ns	5, 7
14	**W6XCLK** Period	8.3		ns	
15	**WRCLK** Period	16.6		ns	
16	**BITCLK** Period	50		ns	

Notes:

1) Timing requirements when **SCI** *is driven with external oscillator.*

*2) System PLL is bypassed (***SYSTEM_ENABLE***=0)*

*3) System PLL is enabled (***SYSTEM_ENABLE***=1).*

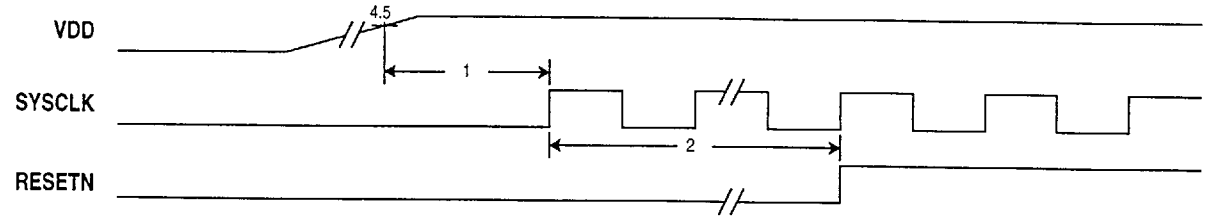
4) The frequency of **SYSCLK** *must be 40MHz (25 ns period) for full operation. However, during special conditions such as the chip reset, a lower frequency on* **SYSCLK** *is allowed.*

5) Timing requirements when **WCI** *is driven with external oscillator.*

*6) Tape PLL is bypassed (***TAPE_ENABLE***=1)*

*7) Tape PLL is enabled (***TAPE_ENABLE***=0).*

15.2 RESET TIMING



NUMBER	PARAMETER	MIN	MAX	UNITS	NOTES
1	Power on to **SYSCLK**		1	us	1
			10	ms	2
2	**SYSCLK** to **RESETN** inactive	10		**SYSCLK**s	

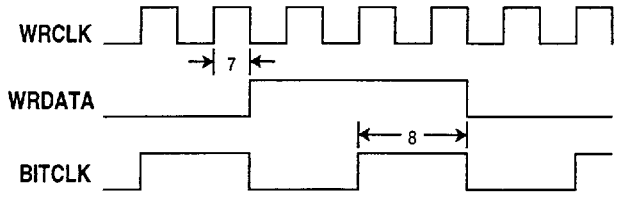
Notes:

1) Using external oscillator.

2) Using internal oscillator.

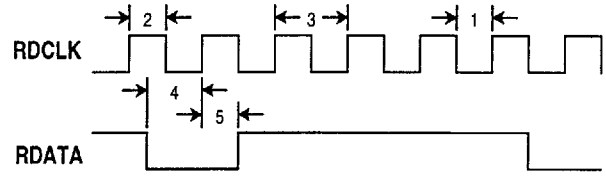
15.3 TAPE TIMING

15.3.1 WRITE DATA OUTPUT



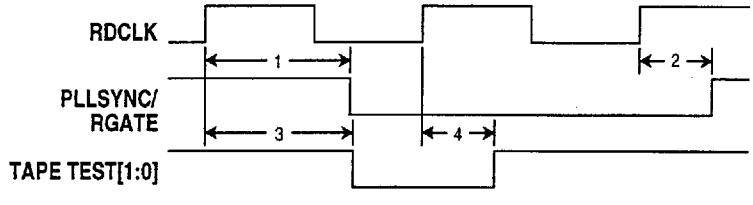
NUMBER	PARAMETER	MIN	MAX	UNITS	NOTES
7	**WRCLK** to **WRDATA** delay	-10	+10	ns	
8	**BITCLK** to **WRDATA** delay	-10	+10	ns	

15.3.2 READ CLOCK AND DATA



NUMBER	PARAMETER	MIN	MAX	UNITS	NOTES
1	**RDCLK** Low Time	15		ns	
2	**RDCLK** High Time	15		ns	
3	**RDCLK** Period	44		ns	
4	**RDATA** Setup	10		ns	
5	**RDATA** Hold	5		ns	

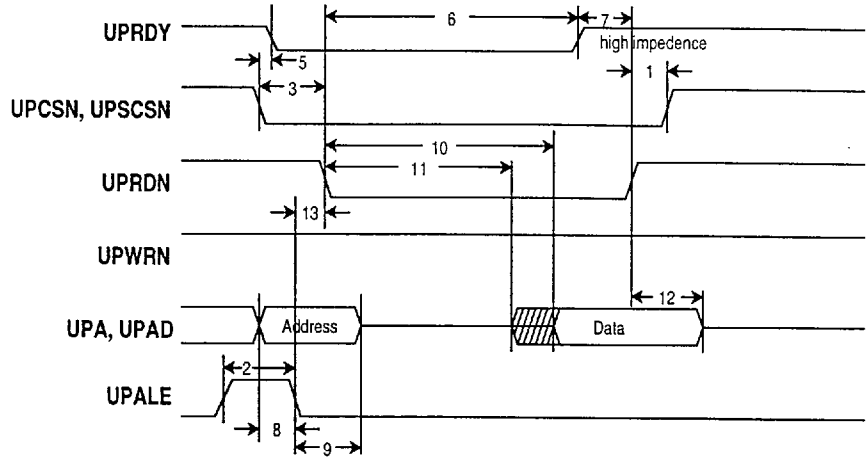
15.3.3 READ STATUS OUTPUTS



NUMBER	PARAMETER	MIN	MAX	UNITS	NOTES
1	**RDCLK** to **PLLSYNC/RGATE** delay		20	ns	
2	**RDCLK** to **PLLSYNC/RGATE** hold	3		ns	
3	**RDCLK** to **TAPE TEST[1:0]** delay		20	ns	
4	**RDCLK** to **TAPE TEST[1:0]** hold	3		ns	

15.4 MICROPROCESSOR TIMING

15.4.1 MICROPROCESSOR READ TIMING



NUMBER	PARAMETER	MIN	MAX	UNITS	NOTES
1	**UPRDN** rise to **UPCSN, UPSCSN** rise	0		ns	
2	**UPALE** pulsewidth.	15		ns	
3	**UPCSN, UPSCSN** fall to **UPRDN** fall	0		ns	
5	**UPCSN, UPSCSN** fall to **UPRDY** fall		20	ns	
6a	**UPRDN** fall to **UPRDY** rise		7	SYSCLKs	1
6b	**UPRDN** fall to **UPRDY** rise		100	SYSCLKs	2
6c	**UPRDN** fall to **UPRDY** rise		100	SYSCLKs	3
7	**UPRDY** rise to **UPRDN** rise	0		ns	
8	Address valid setup to **UPALE** fall	10		ns	
9	Address valid hold from **UPALE** fall	5		ns	
10a	**UPRDN** fall to **UPAD** valid		6	SYSCLKs	1
10b	**UPRDN** fall to **UPAD** valid		6	SYSCLKs	2
10c	**UPRDN** fall to **UPAD** valid		100	SYSCLKs	3
11	**UPRDN** fall to **UPAD** driven		2	SYSCLKs	
12	**UPRDN** rise to **UPAD** tristate	0	10	ns	
13	**UPALE** fall to **UPRDN** fall	0		ns	

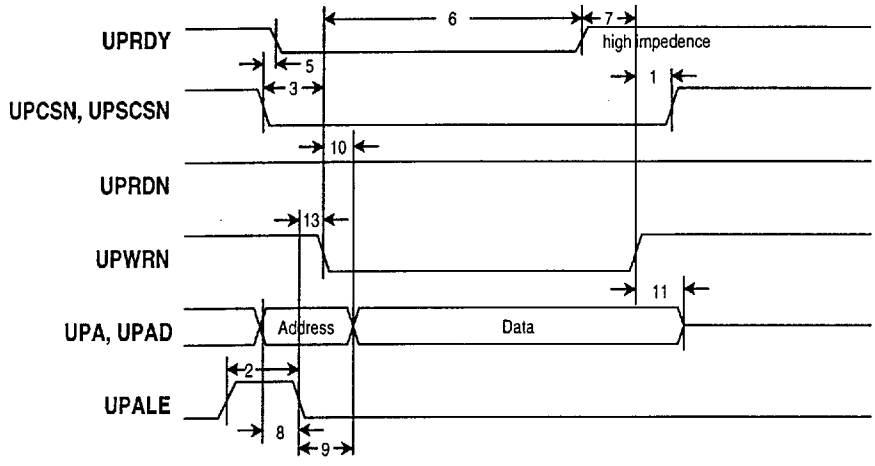
Notes:

1) Typical register access.

2) Accessing DRAM ADDRESS or DRAM DATA with wait states enabled.

3) SRAM emulation mode.

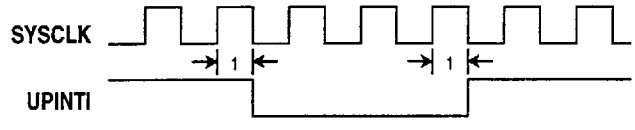
15.4.2 MICROPROCESSOR WRITE TIMING



NUMBER	PARAMETER	MIN	MAX	UNITS	NOTES
1	**UPWRN** rise to **UPCSN, UPSCSN** rise	0		ns	
2	**UPALE** pulsewidth	15		ns	
3	**UPCSN, UPSCSN** fall to **UPWRN** fall	0		ns	
5	**UPCSN, UPSCSN** fall to **UPRDY** fall		20	ns	
6a	**UPWRN** fall to **UPRDY** rise		6	**SYSCLKs**	1
6b	**UPWRN** fall to **UPRDY** rise		100	**SYSCLKs**	2
6c	**UPWRN** fall to **UPRDY** rise		100	**SYSCLKs**	3
7	**UPRDY** rise to **UPWRN** rise	0		ns	
8	Address valid setup to **UPALE** fall	10		ns	
9	Address valid hold from **UPALE** fall	5		ns	
10	**UPWRN** fall to **UPAD** data valid		3	**SYSCLKs**	1
11	**UPWRN** rise to **UPAD** data valid hold	0		ns	
13	**UPALE** fall to **UPWRN** fall	0		ns	

Notes:
1) Typical register access.
2) Accessing DRAM ADDRESS or DRAM DATA with wait states enabled.
3) SRAM emulation mode.

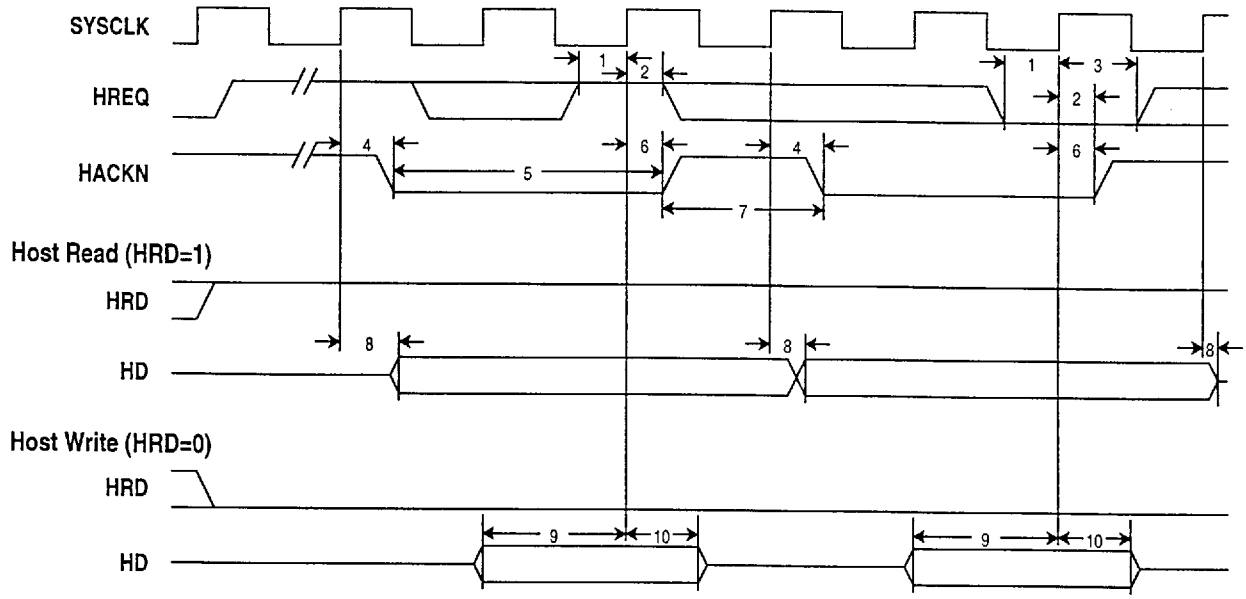
15.4.3 MICROPROCESSOR INTERRUPT TIMING



NUMBER	PARAMETER	MIN	MAX	UNITS	NOTES
1	**SYSCLK** to **UPINTI**		18	ns	

15.5 HOST TIMING

15.5.1 HOST DMA TIMING



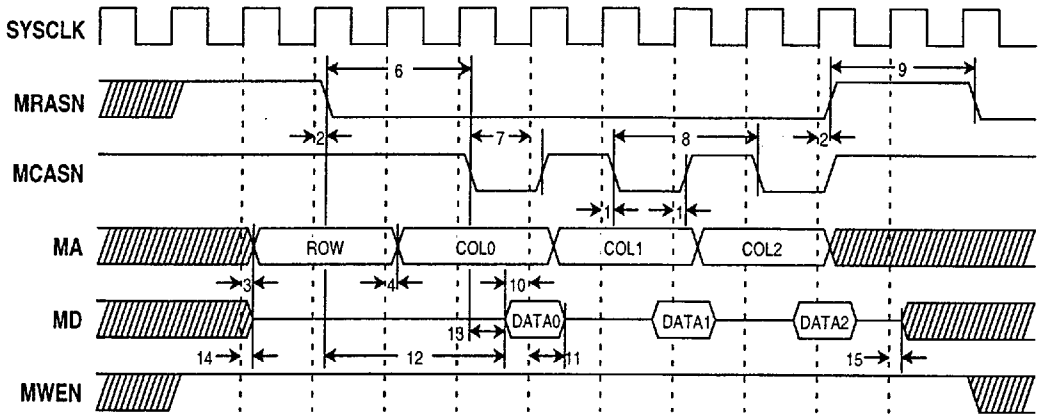
NUMBER	PARAMETER	MIN	MAX	UNITS	NOTES
1	**HREQ** setup to **SYSCLK**	12		ns	1
2	**HREQ** hold from **SYSCLK**	0		ns	1
3	**HREQ** active delay	12		ns	2
4	**HACKN** active edge delay		15	ns	
5	**HACKN** active pulsewidth			**SYSCLK**s	3
6	**HACKN** inactive edge delay		15	ns	
7	**HACKN** inactive pulsewidth			**SYSCLK**s	3
8	**HD** valid data delay		15	ns	
9	**HD** setup to **SYSCLK**	12		ns	
10	**HD** hold from **SYSCLK**	0		ns	

Notes:

1) *During the handshaking operation* **HREQ** *is sampled synchronously. As long as* **HREQ** *is sampled active, the setup and hold times must be met.*
2) *Once* **HREQ** *is sampled inactive, the next active is synchronized with two cascaded flip-flops on the active edge of* **SYSCLK**.
3) *The* **HACKN** *active and inactive pulsewidths are individually programmed in Host Configuration 1. The active edge of* **HACKN** *is faster than the inactive edge by less than 10ns.*

15.6 DRAM TIMING

15.6.1 DRAM READ



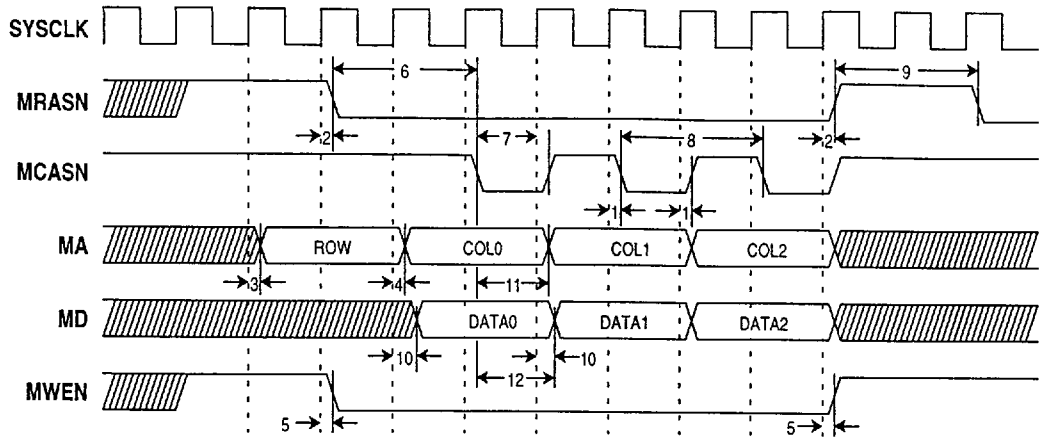
NUMBER	PARAMETER	MIN	MAX	UNITS	NOTES
1	**MCASN** delay	1	6	ns	
2	**MRASN** delay	1	6	ns	
3	**MA** delay for row address valid	1	9	ns	
4	**MA** delay for column address valid	1	9	ns	
6	**MRASN** to **MCASN** delay	2ts-5	2ts+5	ns	
7	**MCASN** active pulsewidth	ts-5	ts+5	ns	1
8	**MCASN** cycle time	ts-1	ts+1	ns	
9	**MRASN** inactive to **MRASN** active	2ts-5	2ts+5	ns	
10	**MD** setup time	5		ns	
11	**MD** hold time		0	ns	
12	access time from **MRASN** active		3ts-5	ns	
13	access time from **MCASN** active		ts-5	ns	
14	**MD** high impedance delay	1	5	ns	
15	**MD** drive delay	1	5	ns	

ts is **SYSCLK** *period*

Notes:

1) Timing is verified by worst case timing simulations. It is not verified in production test.

15.6.2 DRAM WRITE TIMING



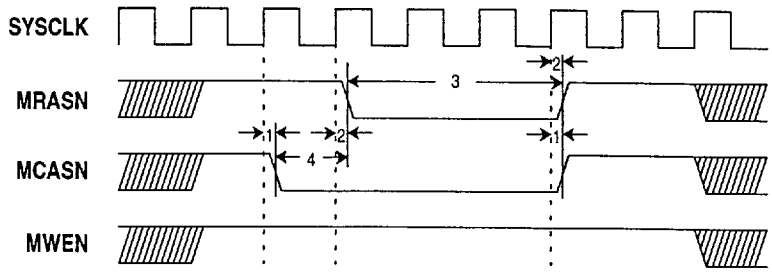
NUMBER	PARAMETER	MIN	MAX	UNITS	NOTES
1	**MCASN** delay	1	6	ns	
2	**MRASN** delay	1	6	ns	
3	**MA** delay for row address valid	1	9	ns	
4	**MA** delay for column address valid	1	9	ns	
5	**MWEN/MOEN** delay	1	11	ns	
6	**MRASN** to **MCASN** delay	ts-5	ts+5	ns	
7	**MCASN** active pulsewidth	ts-5	ts+5	ns	1
8	**MCASN** cycle time	ts-1	ts+1	ns	
9	**MRASN** inactive to **MRASN** active	ts-4		ns	
10	**MD** delay	1	11	ns	
11	**MA** hold from **MCASN** active	ts-5	ts+5	ns	
12	**MD** hold from **MCASN** active	ts-5	ts+5	ns	

ts is **SYSCLK** *period*

Notes:

1) This timing is verified by worst case timing simulations. It is not tested in the production test flow.

15.6.3 DRAM REFRESH



NUMBER	PARAMETER	MIN	MAX	UNITS	NOTES
1	**MCASN** delay	1	6	ns	
2	**MRASN** delay	1	6	ns	
3	**MRASN** pulsewidth	3ts-4	3ts+4	ns	
4	**MCASN** to **MRASN** delay	ts-4	ts+4	ns	

ts is **SYSCLK** *period*

16.0 PACKAGING

Please consult with Advanced Hardware Architectures for the most up-to-date pinout and packaging information.

Dimensions

SYMBOL	NUMBER OF PIN AND SPECIFICATION DIMENSION		
	128		
	RB		
	MIN	NOM	MAX
(LCA)	26		
(LCB)	38		
ExD		14x20	
E	13.9	14	14.1
D	19.9	20	20.1
A	2.846		
A1	0.196	0.348	0.50
A2	2.65	2.75	2.85
[e]		0.5	
b	0.20	0.30	0.40
c	0.10	0.15	0.20
M	0.005		
x			0.13
y			0.10
θ°	0		7
Z_E		0.825	
Z_D		0.575	
L	0.60	0.80	1.00
L_1		1.95	
L_2		1.15	
H_E	17.50	17.90	18.30
H_D	23.50	23.90	24.30
P			0.53
R	-0.05		0.18

Note: All dimensions are in millimeters

Figure 14: Package Cross-Section Diagram

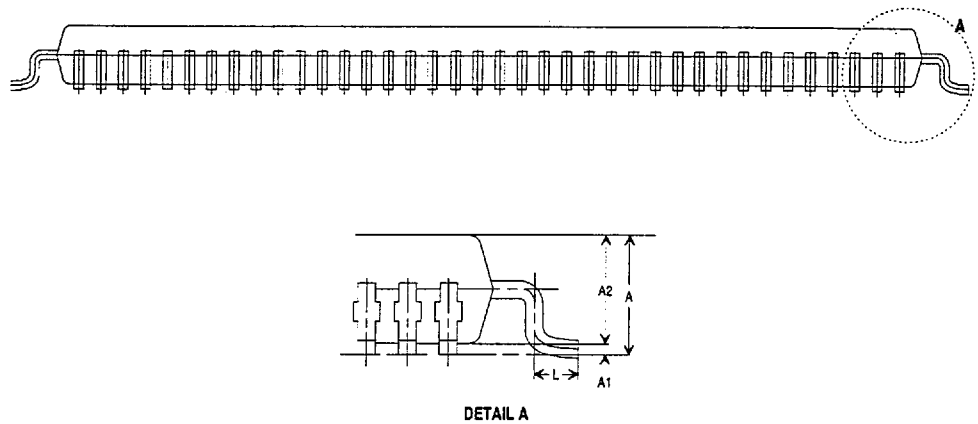
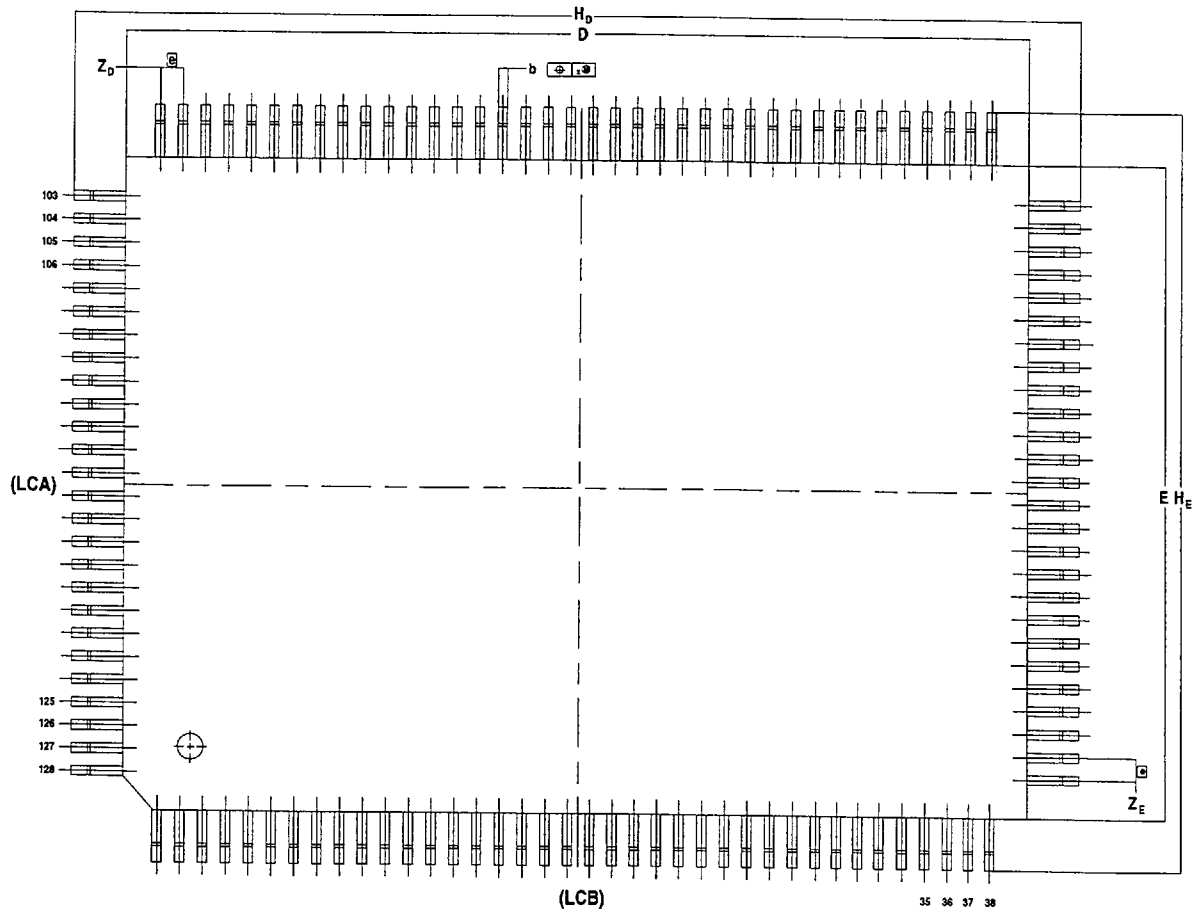


Figure 15: Package Diagram



Complete Package Drawing Available Upon Request

17.0 ORDERING INFORMATION

17.1 AVAILABLE PARTS

PART NUMBER	DESCRIPTION
AHA5140A-040 PQC	QIC® Tape Data Format Controller

17.2 PART NUMBERING

AHA	5140	A-	040	P	Q	C
Manufacturer	Device Number	Revision Level	Speed Designation	Package Material	Package Type	Test Specification

Device Number:
5140
Revision Letter:
A
Package Material Codes:
P Plastic
Package Type Codes:
Q Q - Quad Flat Pack
Test Specifications:
C Commercial 0°C to +70°C

18.0 RELATED TECHNICAL PUBLICATIONS

DOCUMENT #	DESCRIPTION
PB5140	AHA Product Brief – AHA5140 QIC Tape Data Format Controller
ANFC01	AHA Application Note – AHA5140 Initialization Overview
QIC-3095 MC	Serial Recorded Magnetic Tape Minicartridge for Information Interchange
TRAVAN-520	TRAVAN-5 Serial Recorded Magnetic Tape Format for Information Interchange