Vero Component Specification
Version 2.0.1

Document Number BOC-CS-08943

April 18, 1995

Contact: Robert Eng
Owner: G. C. Keller

Co-Processor Hardware Development
Dept. 22U
Internal Zip 2111
Boca Raton

Unclassified

Preface

This document describes the functions of the Vero VLSI component. The main intention of this chip is to provide an efficient means of interfacing several relatively slow serial communication chips to a high-speed bursting CFE bus. This is done through a dual-bus structure, eight independent DMA channels, linked-list chaining and automatic appended I/O operations.

Comments

| Any comments and/or suggestions should be directed to Robert Eng.

Acknowledgements

This document would not have been possible without the help of:

Jay Ackerman,
Serafin Garcia,
Gary Hoch,
Toan Pham and
Eric Stelzer.

Revisions

- 01-22-91 to 03-07-91. Original input to the document.
- 03-14-91 to 04-09-91. Version 0.2.2 updates.
  - Added address parity checking to Local Bus
  - New definition for Burst Last signal
  - New definition for Exception signal
  - New AIB Data Steering table
  - New AIB Ready signal definition
  - New AIB Detect signal definition
  - New AIB Clock signal definition
  - Added Local Bus Exception to Miami sourced interrupts
  - Added Local Bus Exception to Brighton sourced interrupts
  - Rearranged Brighton interrupt table
  - Added Local Bus Configuration register
  - Added full 64KB DMA byte count capability
  - Rearranged TQC, MPR, and CPR in CDT register map
  - Added restriction to usage of DCCR and GDCR channel enable command
  - Changed buffer queuing to allow 4K-1 buffers instead of 4K
  - Corrected Queuing flow example
  - Added 2 Local Bus input interrupt signals
  - Added a high priority AIB interrupt input
  - Changed interrupt vector assignments
  - Added Interrupt Status register
  - Added bits to AIB Command/Status register
  - Added AIB presence detect bit to PDR
  - Changed AIB Slave Read and Write cycles to drive Chip Select sooner
- 08-01-91 to 08-23-91. Version 0.3 updates.

- Changed to Toshiba technology VLSI
- Changed ground to power pin ratio
- Added AIB wrap capability
- Added AIB Wrap Enable signal
- Changed AIB INT signal description
- Rearranged Brighton interrupt code table
- Changed signal description for all LSSD test pins
- Added Local Bus Exception and AIB Bus parity status to DISR
- Added gate array ID and queuing enable/disable function to LBCR
- Rearranged Interrupt vector assignments
- Removed generic RHPI and EOI functions of interrupt controller
- Added AIB INT0 EOI command
- Added AIB INT1 EOI command
- Added wrap buffer enable and wrap mode bits to ACSR
- Added AIB Wrap Control register
- Added AIB Wrap Address/Data register

• 09-01-91 to 01-24-91. Version 0.4 updates.

- Changed format of document to Bookmaster.
- Changed signal description section to table format.
- Changed timings to reflect 25MHz performance.
- Changed timings for AIB Bus (no revision code used here)
- Added Pin name/Pin number cross reference appendix.
- Added Local Bus Timing appendix.
- Added ICT I/O Mapping appendix.

• 02-11-92 to 04-08-92. Version 0.4.1 updates.

- Removed revision code 1, 2, and 3 from document.
- Changed Queue Tail Pointer register to 'read only' in 1.7, “Vero Register Address Map” on page 19.
- Changed Interrupt Status register to 'read only' in 1.7, “Vero Register Address Map” on page 19.
- Added Bit 31 in 2.2.1, “Channel Control Register (CCR)” on page 23.
- Updated definition of Bit 0 (enable/disable) in 2.2.1, “Channel Control Register (CCR)” on page 23.
- Clarified definition of GDCR stop command in 2.2.3, “Global DMA Command Register (GDCR)” on page 27.
- Updated Queue Count initialization requirements in 2.2.5, “Transfer/Queue Count Register (TQC)” on page 29.
- Changed QTP to 'read only' in 2.2.10, “Queue Tail Pointer Register (QTP)” on page 33.
- Added Section 2.3.1, “Linked List Chaining/Stopping Matrix” on page 35.
- Added more description in 2.4, “Queuing Operation (NOT Supported)” on page 36.
- Changed DISR to 'read only' in 2.5.1, “DMA Interrupt Status Registers (DISR)” on page 40.
- Changed Queue underrun status bit definition in 2.5.1, “DMA Interrupt Status Registers (DISR)” on page 40.
- Changed bit names to QE and PE in 2.6.2, “Local Bus Configuration register (LBCR)” on page 42.
- Changed polarity of queuing enable bit in 2.6.2, “Local Bus Configuration register (LBCR)” on page 42.
- Changed polarity of LED enable value in 5.1.2, “LED Enable Register (LER)” on page 62.
- Changed T7 min to 0ns in B.2.1, “AIB Slave Read Cycle” on page 72.
- Changed T7 min to 0ns in B.2.3, “AIB Interrupt Acknowledge Cycle” on page 76.
- Documented Local Bus timings for CFE incompatibility in Appendix C, “Local Bus Timings” on page 79.

• March 17, 1993. Version 1.1.0 updates.

- Note that Version 1.0.0 was never formally released
- added block diagram (Figure 1 on page 12)

- changed local bus signal name L_PAR to L_ADP (for consistency with Miami and Brighton)
- removed A_RDY support during DMA
- removed AIB wrap support
- clarified A_DETECT use
- clarified Brighton and Miami interrupt pins
- added secondary address select
- removed TC followed by EOP support
- corrected conflicting AIB OP information
- clarified when writing QR causes interrupt
- removed APR bit from ACSR
- removed DFCR register (had no function)
- clarified use of PDR
- the RESET COMMAND values are in the process of being verified and should not be relied upon

- July 15, 1993. Version 2.0.0 updates.
  - Clarified XINT bus is active low
  - Clarified present dectect bits are non-inverted
  - Clarified Queue Pointer Address in QR is invalid when read
  - Indicated Queueing is not supported
  - Changed RESET COMMAND values
  - Changed CS active and inactive pulse width values
  - Clarified the use of bit 31 and bit 0 of CCR
  - Added active time for -A_RESET signal
  - Changed max DMA byte count from '64kb' to '64kb-1'

| • April 18, 1995. Version 2.0.1 updates.

| – Declassified document.
| – Changed contact person to Robert Eng.
| – Reworded note not to write zero DMA byte count and to initialize DMA byte count for all DMA
| channels; added note to linked list chaining description not to write zero DMA byte count.
| – Added Electrical Characteristics section to Appendix (temperature, power, voltage).
| – Added Packaging section to Appendix.

Contents

Figures

Tables

1.0 Vero Component Overview

1.1 General

The major functions of the Vero module are highlighted below.

- Toshiba (0.8 micron) 12.8mm Semi-custom chip; 278,964 transistors
  - 8 channel dual-bus DMA controller
  - Common Front End (CFE) Local Bus interface
  - Application Interface Bus (AIB bus) interface
  - Interrupt controller (80960 expanded mode interrupt support)
  - 208 pin PQFP; 174 signal I/O, 25 Ground, 8 Vdd
  - 25MHz operation

1.2 RAS Highlights

- Address/data parity generation and checking on Local Bus
- Data parity generation and checking on AIB Bus
- No write only registers

1.3 Performance

The design point of the Vero module is to provide a high performance DMA subsystem for Juno Adapter daughter card designs. In addition, the 80960 support functions of interrupt controller and AIB Bus interface are integrated into the module. The DMA unit is capable of supporting 0 wait state Local Bus accesses for durations of 16-byte bursted transfers. This translates into a peak bandwidth of approximately 57MB/s. In the Juno Adapter implementation however, the Brighton module will induce wait states into Vero's bus cycle that reduce its Local Bus bandwidth capability down to about 33MB/s. This is not to say that Vero can sustain 33MB/s on the Local Bus. Vero's sustained rate on the Local Bus will be a function of other masters such as Miami also accessing the Local Bus.

In addition to being a Local Bus master, the Vero module is also an AIB Bus master for DMA operations. Here, the DMA is capable of running continuous 160ns bus cycles to 4-byte wide devices. This means that for a 4-byte wide device capable of supporting a cycle time of 160ns, a bursted rate of 25MB/s is possible.

1.4 Sustained Throughput

The throughput of the Vero chip is dependent on specific AIB implementations. It is intended that AIB's capable of bursted bandwidths of 25MB/s and sustained bandwiths of at least 10MB/s be possible using the Vero module. Other factors, however, can influence the real data throughput of the module. These include:

- AIB device buffer size
- Frequency of linked list chaining
- Frequency of DMA executed AIB I/O operations (AIB OP's)
- Local bus utilization by other master devices

1.5 Vero Block Diagram

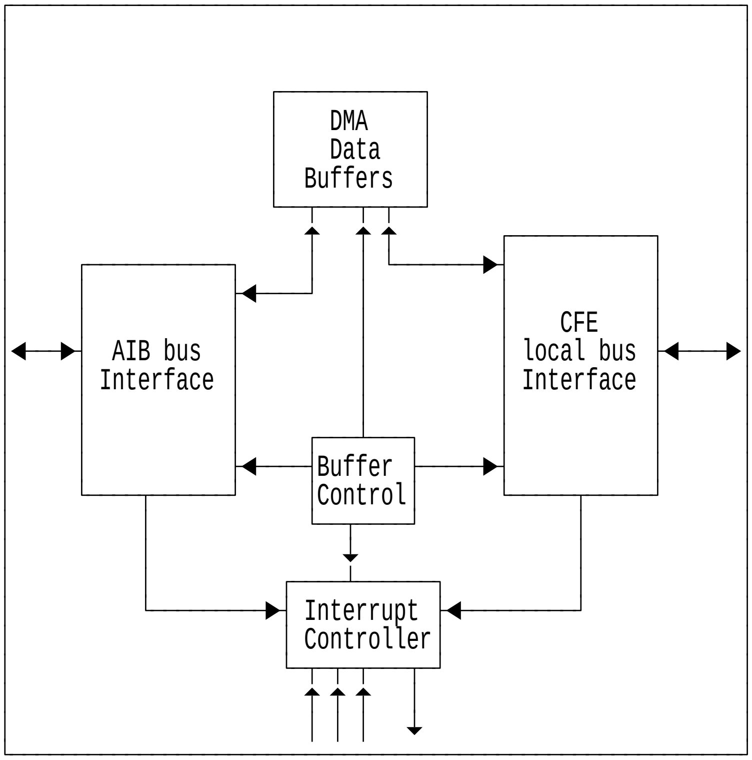


Figure 1. Major Functional Blocks in Vero

1.6 Vero Signal Description

The following tables provide a description of each signal in the Vero module. See Appendix A, "Vero Pin Name/Number Cross Reference" on page 63 for module pin number information.

1.6.1 Vero Local Bus Signals (CFE Bus)

Table 1. Vero Local Bus Signals

Name	Type	Description
L_AD(31-0)	**I/O**	**Local Address/Data(31-0)** are used for both address generation and data transfer on the Local Bus. The address is driven by Vero when it owns the Local Bus and received otherwise. Data is driven by the device providing the data.
L_ADP(3-0)	**I/O**	**Local Bus Address/Data Parity(3-0)** are used to generate and check data parity on the Local Bus.
L_W/-R	**I/O**	**Local Write/-Read** is used for Local Bus transfers to establish the direction of data flow. This signal is driven when Vero is the Local Bus master and received otherwise.
-L_RDY	**I/O**	**-Local Ready** indicates the termination of a data transfer. This signal is driven by Vero as a slave and received when Vero is the Local Bus master.
-L_BLAST	**O**	**-Local Burst Last** is driven by Vero as a master to indicate the last transfer of a burst access.
-L_ADS	**I/O**	**-Local Address Strobe** indicates valid address and the start of a new bus access. This signal is driven by Vero as a master and received by Vero as a slave.
-L_BE(3-0)	**I/O**	**-Local Byte Enables(3-0)** select which of the four bytes addressed are active during an access. These signals are driven by Vero as a master and received by Vero as a slave.
-L_REQ	**O**	**-Local Bus Request** is driven by Vero to request use of the Local Bus for transfers to and from the Vero DMA channels.
-L_GRANT	**I**	**-Local Bus Grant** is received by Vero to detect that a Vero bus request has been granted.
-L_EXCPT	**I/O**	**-Local Exception** is received by Vero when it is the Local Bus master if the slave detects a critical error, ie. Local Bus parity error. Vero will halt the current transfer and post an interrupt in the appropriate DMA interrupt status register (see 2.5.1, "DMA Interrupt Status Registers (DISR)" on page 40). As a slave on the Local Bus Vero drives this signal when it detects bad data parity on a write to one of its internal registers. No interrupt or status is reported in this case.

1.6.2 Vero AIB Bus Signals

Table 2 (Page 1 of 2). Vero AIB Bus Signals

Name	Type	Description
A_AD(31-0)	**I/O**	**AIB Address/Data(31-0)** are the multiplexed address and data bits for the AIB interface. For DMA data transfer to a DMA slave AIB device these lines are data only. All DMA data transfers use implicit addressing of the I/O device with the DACK signals. For 80960 data transfer to the AIB an address is generated on AD0-18 during the first clock cycle and then data is presented on subsequent cycles.
A_PAR(3-0)	**I/O**	**AIB Data Parity(3-0)** are the data parity data bits that are driven by the device supplying data during a data transfer cycle. Each bit is the generated odd parity bit for the corresponding byte of data. Vero generates the parity bits during writes and can check them during reads.
-A_ALE	**O**	**-AIB Address Latch Enable** is provided during 80960 data transfer to allow an external address latch to demultiplex the AD0-18 signals. Address is valid before and after the falling edge of this signal.
-A_BE(3-0):	**I/O**	**-AIB Byte Enables(3-0)** are driven along with address during 80960 data transfer to the AIB. These four signals indicate to the AIB which data bytes are valid for a given cycle for write operations. They are used during read operations to tell the AIB which data bytes are considered to be valid for the read access. During DMA read operations to a DMA slave AIB device, these signals are driven by the DMA device to indicate to the DMA controller which data bytes are valid for that cycle. During DMA write operations, these signals are driven by the the DMA controller to indicate to the device how many data bytes are valid for that cycle.
-A_WR	**O**	**-AIB Write** is driven active during a write operation to an AIB slave device.
-A_RD	**O**	**-AIB Read** is driven valid during a read operation to an AIB slave device.
-A_DEN	**O**	**-AIB Data Enable** is driven active during the data phase of a cycle to an AIB slave device. For reads, it is an indication that it is now alright to enable data on to the bus. For writes, it is an indication that data is now valid on the bus.
-A_CS(0-4)	**O**	**-AIB Chip Select(0-4)** are driven active when a valid address decode is detected for the address ranges defined in the CSD0-4 registers.
+A_RDY	**I**	**+AIB Ready** is used to pace the cycle if the active cycle time for an AIB slave device is greater than the programmed cycle time (see 4.2.1, “Chip Select Definition registers (CSD0-4)” on page 54). The AIB bus is normally 'ready', and this signal is driven low by the slave device to extend the cycle time. A_RDY is not supported during DMA transfers.
-A_RESET	**O**	**-AIB Reset** is used to reset the AIB. It is driven active for 60ns after the rising edge of the -RESET input signal, and can also be driven active under program control of the 80960 to reset just the AIB.
-A_ERROR	**I**	**-AIB Error input** is driven by the AIB when any AIB defined critical error occurs. This will cause vector #232 to be generated. (see Table 12 on page 44).

Table 2 (Page 2 of 2). Vero AIB Bus Signals

Name	Type	Description
A_CLK	**O**	**AIB Clock** is the 25MHz AIB reference clock.
-D_REQ(0-7)	**I**	**-DMA Request(0-7)** are the eight DMA request signals that AIB devices use to request DMA service. These signals are driven active asynchronously and driven inactive in response to the device receiving the corresponding DACK signal.
-D_ACK(0-7)	**O**	**-DMA Acknowledge(0-7)** are the eight DMA acknowledge signals that the DMA controller drives to the AIB Bus when the corresponding DREQ is being serviced. The duration of these signals are individually programmable within the DAPW registers to allow for both fast and slow devices to coexist on the AIB with minimal external logic.
-D_EOP(0-7)	**I/O**	**-DMA End-of-Process(0-7)** used to signal various termination conditions for the DMA channels. For a DMA transmit channel, EOP can be either a synchronous output or an asynchronous input. As an output, EOP is driven active synchronous with the data DACK transfer on which a DMA terminal count condition occurs. For a DMA receive channel, EOP can be either a synchronous input or an asynchronous input. Either case can optionally cause a DMA channel to flush the current buffer, stop, interrupt, or chain. There is a separate EOP signal for each DMA channel.
-A_DETECT	**I**	**-AIB Detect input** must be driven low to allow VERO AIB bus to function. The state of this signal is readable in the 5.1.1, “Presence Detect Register (PDR)” on page 61.

1.6.3 Vero Interrupt Controller Signals

Table 3. Vero Interrupt Controller Signals

Name	Type	Description
PXINT(7-0)	**O**	**Processor Interrupt (7-0)** form an interrupt vector bus that is compatible with the 80960CA expanded mode interrupt structure. Vero internally manages the prioritization of interrupt input signals and translates those inputs into specific interrupt vectors which are placed on this bus. This bus is active low.
-A_INT(3-0)	**I**	**-AIB Interrupts(3-0)** are four separate interrupt input signals that are driven active by devices on the AIB when interrupt service by the 80960 is required. -AIB_INT(0) and -AIB_INT(1) can be optionally programmed to require the interrupting device to supply an 8-bit interrupt vector when the corresponding INTACK signal is driven active. Also, these two inputs can be used as direct interrupt inputs with the vector being automatically supplied by the Vero chip. -AIB_INT(2) and -AIB_INT(3) are always used as direct interrupt inputs with the vector automatically being supplied by the Vero chip. When operating -AIB_INT(0) or -AIB_INT(1) in interrupt acknowledge mode, the interrupt must be cleared at its source, AND the appropriate EOI to the interrupt controller must be issued, or interrupts below that prioritization level will be locked out (see 3.3.1, “AIB INT0 End-of-Interrupt (EOI0) command” on page 50 and 3.3.2, “AIB INT0 End-of-Interrupt (EOI1) command” on page 50).
-A_INTACK(1-0)	**O**	**-AIB Interrupt Acknowledge(1-0)** are two separate output signals to the AIB that are coupled to each of the -AIB_INT(0) and -AIB_INT(1) signals. When a device uses the -AIB_INT(1-0) signals and the Vero chip is programmed to request an external vector, it must be capable of responding to the corresponding -AIB_INTACK(1-0) signal by driving an 8-bit interrupt vector on lines A_AD(7-0) during the interrupt acknowledge cycle.
-L_INT(1-0)	**I**	**-Local Bus Interrupts(1-0)** are two separate interrupt input signals for devices that may reside on the Local Bus (or elsewhere). These inputs must be held active until the interrupt has been acknowledged by a read or write to an interrupt device status register, which clears the interrupt at its source.
M_INT(3-0)	**I**	**-Miami Interrupts(3-0)** are four encoded interrupt inputs that can interface to the Miami chip. These signals correspond to certain interrupts as shown in Table 12 on page 44. If unused these signals should be pulled up to +5V.
B_INT(3-0)	**I**	**-Brighton Interrupts(3-0)** are four encoded interrupt inputs that can interface to the Brighton chip. These signals correspond to certain interrupts as shown in Table 12 on page 44. If unused these signals should be pulled up to +5V.
-WDOG	**O**	**-Watch Dog Error** can be used to provide an output signal to the Miami chip when the watchdog timer interrupt is issued. It goes low when B_INT(3-0)='0000'.

1.6.4 Vero Miscellaneous Signals

Table 4. Vero Miscellaneous Signals

Name	Type	Description
PD(11-8) and PD (6-0)	**I**	**Presence Detect(11-8) and PD (6-0)** are used as dedicated digital inputs to Vero. (see 5.1.1, "Presence Detect Register (PDR)" on page 61).
-LED_EN	**O**	**-LED Enable** is a signal dedicated to enabling an external light emitting diode for diagnostic purposes. (see 5.1.2, "LED Enable Register (LER)" on page 62).
PRI/-SEC_ADDR	**I**	**Primary/-Secondary Address Select** is a dedicated input for selecting the address map on the Local Bus for Vero. This input will allow for two (2) Vero chips to be attacted to the CFE Bus. This pin should be pulled up to +5 external to the chip in order to select the primary address. This pin should be driven low external to the chip in order to select the secondary address. The primary address map will be from 1FF00000 h to 1FF8FFFF h. The secondary address map will be from 1FE00000 h to 1FE8FFFF h. (see 1.7.1, "Vero Primary and Secondary Address Map Information" on page 19). **Note:** This spec lists all addresses as primary addresses. For secondary addresses use 1FExxxxx instead of 1FFxxxxx.

1.6.5 Vero Clocks and Reset

Table 5 (Page 1 of 2). Vero Clocks and Reset

Name	Type	Description
L_OSC	**I**	**Local Oscillator** is the Local Bus input reference clock. All internal functional non-overlapping clocks are derived from this oscillator. All CFE Bus signals are referenced to this clock.
TST_CLKA	**I**	**LSSD A_Clock** is used during LSSD testing of the chip. This pin should be pulled up to +5v external to the chip during normal operation.
TST_CLKB	**I**	**LSSD B_Clock** is used during LSSD testing of the chip. This pin should be pulled up to +5v external to the chip during normal operation.
TST_CLKC	**I**	**LSSD C_Clock** is used during LSSD testing of the chip. This pin should be pulled up to +5v external to the chip during normal operation.
RAMTSTCLK		**LSSD Ram_Clock** is used during LSSD testing of the chip. This pin should be pulled up to +5v external to the chip during normal operation.
RAMTSTEN	**I**	**Ram Test Enable** is used to enable testing of the internal RAM's during LSSD test. This pin should be pulled up to +5v external to the chip during normal operation.

Table 5 (Page 2 of 2). Vero Clocks and Reset

Name	**Type**	**Description**
DI	**I**	**Driver Inhibit** is used to tri-state all I/O and outputs of the chip. This pin should be pulled up to +5v external to the chip during normal operation. This pin is tied low at the same time that the IN CIRCUIT TEST pin is tied high to tri-state all internal off chip drivers.
ICT_EN	**I**	**In-Circuit Test** places the chip in an I/O mapping mode for in-circuit testing (see Appendix D, “In-Circuit Test I/O Mapping” on page 81). This pin should be pulled up to +5v external to the chip during normal operation. The DRIVER INHIBIT pin must be tied high to put the chip into ICT mode.
COMP_RES	**I**	**Compensation Resistor** is a chip input used to adjust the speed of all internal off-chip drivers. This signal should be pulled up to +5v through a 909 +/- 1% ohm resistor.
-RESET	**I**	**Reset** is the reset input signal to the chip.

1.7 Vero Register Address Map

The memory map of all registers addressable within the Vero module is shown below. Detailed information is found in the indicated section. Each register is 4 byte aligned in the address space and should be accessed using 80960 'word' load (ld) and 'word' store (st) instructions.

All registers, when read, will return '0' values in those bits that are undefined for a specific register.

At the end of each register description section, that registers RESET conditions are shown. RESET COMMAND refers to the value that the register bits assume if a DMA channel is reset using either the DMA Channel Command Register (see 2.2.2, “DMA Channel Command Registers (DCCR 0-7)” on page 26) or the Global DMA Command Register (see 2.2.3, “Global DMA Command Register (GDCR)” on page 27). Also, a 'U' in the "Reset Conditions" section means undefined, and an 'S' means the value stays the same as before the reset command.

1.7.1 Vero Primary and Secondary Address Map Information

This document will only reference the primary addresses for the Vero registers. The primary and secondary addresses for the Vero registers are:

```
Primary Address Map:     1FF8000 h to 1FF8FFFF h
Secondary Address Map:   1FE8000 h to 1FE8FFFF h
```

Table 6. Vero DMA Registers

Name	**Local Bus Address**	**Access Type**	**Section**
DMA Channel 'x' AIB_ADDR 1/2	1FF8x000h	r/w	2.2.7
DMA Channel 'x' AIB_OP1 Data	1FF8x004h	r/w	2.2.8
DMA Channel 'x' AIB_OP2 Data	1FF8x008h	r/w	2.2.9
DMA Channel 'x' Memory Pointer	1FF8x00Ch	r/w	2.2.4
DMA Channel 'x' Transfer/Queue Count	1FF8x010h	r/w	2.2.5
DMA Channel 'x' Chain Pointer	1FF8x014h	r/w	2.2.6
DMA Channel 'x' Channel Control	1FF8x018h	r/w	2.2.1
DMA Channel 'x' Queue Tail Pointer	1FF8x01Ch	ro	2.2.10
DMA Channel 'x' Command	1FF8x020h	r/w	2.2.2
DMA Channel 'x' Queuing	1FF8x024h	r/w	2.4
DMA Channel 'x' Interrupt Status	1FF8x028h	ro	2.5.1
DMA Channel 'x' Fifo Residual Count	1FF8x02Ch	ro	2.6.1
Global DMA Command	1FF88000h	r/w	2.2.3
Local Bus Configuration	1FF88008h	r/w	2.6.2
Queuing Status	1FF8800Ch	r/w	2.4.2
Note: 'x' = DMA channel number.			

Table 7. Vero Interrupt Registers

Name	Local Bus Address	Access Type	Section
Interrupt Initialization	1FF88010h	r/w	3.2.1
Interrupt Mask	1FF88014h	r/w	3.2.2
Interrupt Status	1FF88018h	ro	3.2.3
AIB INT0 End-of-Interrupt	1FF89000h	wo (command)	3.3.1
AIB INT1 End-of-Interrupt	1FF8A000h	wo (command)	3.3.2

Table 8. Vero AIB Registers

Name	Local Bus Address	Access Type	Section
Chip Select Definition 0	1FF8B000h	r/w	4.2.1
Chip Select Definition 1	1FF8B004h	r/w	4.2.1
Chip Select Definition 2	1FF8B008h	r/w	4.2.1
Chip Select Definition 3	1FF8B00Ch	r/w	4.2.1
Chip Select Definition 4	1FF8B010h	r/w	4.2.1
DMA Acknowledge Pulse Width 0	1FF8B030h	r/w	4.2.2
DMA Acknowledge Pulse Width 1	1FF8B034h	r/w	4.2.2
DMA Acknowledge Pulse Width 2	1FF8B038h	r/w	4.2.2
DMA Acknowledge Pulse Width 3	1FF8B03Ch	r/w	4.2.2
DMA Acknowledge Pulse Width 4	1FF8B040h	r/w	4.2.2
DMA Acknowledge Pulse Width 5	1FF8B044h	r/w	4.2.2
DMA Acknowledge Pulse Width 6	1FF8B048h	r/w	4.2.2
DMA Acknowledge Pulse Width 7	1FF8B04Ch	r/w	4.2.2
AIB Command/Status	1FF8C000h	r/w	4.2.3

Table 9. Vero Miscellaneous Registers

Name	Local Bus Address	Access Type	Section
Presence Detect	1FF8D000h	ro	5.1.1
LED Enable	1FF8D004h	r/w	5.1.1

2.0 Vero DMA Controller

2.1 General

The functions of the DMA portion of the Vero module are highlighted below.

- 8 independent DMA channels
- split bus implementation (32-bit AIB Bus to 32-bit Local Bus)
- support of 8/16/32 bit AIB Bus devices
- 16 byte buffer per channel
- 16 byte burst capability on Local Bus (57 MB/s peak bandwidth)
- 32-bit 4GB addressability on Local Bus
- 16-bit 64KB addressability on AIB Bus for AIB OP's
- separate DREQ and DACK signals for each DMA channel
- programmable DACK cycle time
- 64KB byte count capability
- 8 word descriptor block for each channel
- linked list chaining of buffers on all channels
- chaining support for end-of-process and terminal count condition
- buffer queuing (up to 4K-1 buffers) on all channels
- automatic storage of residual transfer count on chain event
- automatic storage of AIB OP reads on chain event
- up to 2 programmable auto I/O operations on chain event
- 7 interrupt sources for each channel

Each of the 8 DMA channels are of equal function. Each channel can be programmed to support data transfer, 1) from the AIB Bus to the Local Bus (receive) or, 2) from the Local Bus to the AIB Bus (transmit). The service priority is fixed with CH0 having highest priority and CH7 having lowest priority.

Each channel is controlled by an 8 word Channel Descriptor Table (CDT). The program normally writes the desired CDT register values to memory resident structures called Channel Descriptor Blocks (CDB's). The program then writes the starting address of a CDB to the Queuing Register (see 2.4, “Queuing Operation (NOT Supported)” on page 36). At this point, the CDB is automatically fetched from memory and loaded into the CDT. Chains of CDB's are linked using this queuing mechanism. Alternately, the CDT registers can be programmed directly.

Once enabled, a channel will service DMA requests from the AIB Bus until one of a number of programmable conditions is reached. If the DMA channel is programmed to stop on one of these conditions, the channel can be re-enabled with a write to the Channel Control Register (CCR), or by queuing another buffer. Any condition that can stop the channel can also interrupt the 80960. Also, the channel has the ability to interrupt without stopping the channel. All of the options are programmable in the CCR and are described later.

The two types of DMA requests that come from the AIB Bus can be classified as a Transmit request (TR) or a Receive request (RR).

Once a TR is received, the DMA controller arbitrates for control of the Local Bus, and when granted control, bursts 16 bytes of data from memory into that channel's FIFO buffer. The data is then transferred to the requesting device across the AIB Bus until the FIFO buffer is empty. This allows data transfer to the AIB device to occur in the background of Local Bus activity.

For a receive DMA channel, the DMA controller will receive up to 16 bytes of data across the AIB Bus. It then arbitrate for control of the Local Bus. Once granted, the DMA controller bursts all 16 bytes into memory. This, again, allows data transfer from the AIB device to occur in the background of Local Bus activity.

DMA FIFO Buffer: As mentioned above, data transfer from Local Bus to AIB Bus is not direct, but rather, is buffered in a set of fifo's. There is a 16 byte deep fifo associated with each DMA channel that acts as an intermediate storage area for data. The fifos support high speed access so that all accesses to the Local Bus will move data into and out of the fifos instead of directly to the AIB Bus, which might support only slower devices. This allows all Local Bus DMA data transfer initiated by Vero to be high speed bursted accesses.**:** The fifo's are configured such that AIB Bus accesses and Local Bus accesses can occur simultaneously. One DMA channel can be performing a read or write on the AIB Bus, while at the same time, another channel is performing a read or write on the Local Bus.

2.2 Channel Descriptor Table

Table 10 shows the organization of the channel descriptor table for one of the DMA channels. This table is duplicated for each of the 8 DMA channels. The table exists in the address space of the 80960 and is accessible via 32-bit 'load' and 'store' type instructions.

Table 10. Vero Channel Descriptor Table (CDT)

Register Name	**Local Bus Address**	**Valid Bits**
Queue Tail Pointer (QTP)	1FF8x01Ch	31-0
Channel Control Register (CCR)	1FF8x018h	21-0
Chain Pointer Register (CPR)	1FF8x014h	31-0
Queue Count Register (TQC)	1FF8x010h	27-16
Transfer Count Register (TQC)	1FF8x010h	15-0
Memory Pointer Register (MPR)	1FF8x00Ch	31-0
AIB OP2 Data Register (AIB_OP2)	1FF8x008h	31-0
AIB OP1 Data Register (AIB_OP1)	1FF8x004h	31-0
AIB OP2 Address Register (AIB_ADDR2)	1FF8x000h	31-16
AIB OP1 Address Register (AIB_ADDR1)	1FF8x000h	15-0
Note: 'x' = DMA channel number.		

2.2.1 Channel Control Register (CCR)

Description: The CCR register controls the operational personality for a DMA channel.

Register Format

```
(80960 Address = 1FF8x018h) r/w     x = channel #
```

```
  15  14  13  12  11  10  9   8   7   6   5   4   3   2   1   0
   channel     channel     channel            channel
|  stopping | interrupt |  chaining |        definition        |   |
|  options  |  options  |  options  |         options          |   |
```

STP 2	STP 1	STP 0	INT 2	INT 1	INT 0	LCH 2	LCH 1	LCH 0	PORT SIZE	EOP DIR	EOP AEN	TR SYN	+T/ -R	EN/ DIS

```
  31  30                         22  21  20  19  18  17  16
                                       AIB        AIB
|    |                             | write |  operation   |
|    |          Reserved           |control|   control    |
```

GEN STA	Reserved	WR 1	WR 0	OP #2	OP #1	#IO OPS	#IO OPS

Bit Descriptions

- Bit 31. General purpose status bit. This bit can be used for any application dependent purpose. It has no function internal to the chip. It is updated by I/O writes, and during list chaining fetches.
- Bits 30-22. Reserved. Always program these bits to '0'.
- Bits 21-20. These two bits are encoded to indicate what data will be written to the AIB Bus on the second of two AIB OP's, if the first of the two AIB OP's is a read.

```
B21    B20
---    ---
 0      0     -- Write the value of AIB_OP2 Data Register
 0      1     -- Write the value that was read on first OP
 1      0     -- Write the complement of value read on first OP
 1      1     -- Reserved
```

- Bit 19. AIB OP #2, read or write. This bit is coupled with bits 16 and 17. It defines the second IO operation to the AIB Bus after a chain event to be a read or write. 0 = read. 1 = write. If the operation is a write, the data written is defined by bits 20 and 21 of the CCR. If the operation is a read, the data will be stored in memory.
- Bit 18. AIB OP #1, read or write. This bit is coupled with bits 16 and 17. It defines the first IO operation to the AIB Bus after a chain event to be a read or write. 0 = read. 1 = write. If the operation is a write, the data written is defined by bits of the AIB_OP1 register. If the operation is a read, the data will be stored in memory.
- Bits 17-16. Number of IO operations to AIB Bus upon chain event. These two bits are encoded to provide the DMA controller with the ability to execute up to 2 discrete IO operations to the AIB Bus following the recognition of a chain event. The address to which the first IO operation occurs is defined

by bits 0-15 in the AIB_ADDR 1/2 register. The address to which the second IO operation occurs is define by bits 16-31 in the AIB_ADDR 1/2 register.

```
00 = No IO operation upon chain event.
01 = One IO operation upon chain event.
10 = Two IO operations upon chain event.
11 = Reserved.
```

- Bit 15-13. Encoded DMA channel stopping options.

STP2	STP1	STP0	Stopping Option
0	0	0	Do not stop
0	0	1	TC=0
0	1	0	EOP
0	1	1	TC=0 "or" EOP
1	0	0	Reserved
1	0	1	Reserved
1	1	0	Reserved
1	1	1	Reserved

When a channel is stopped, the CCR enable/disable bit is reset. Also, a chaining condition takes precedence over a stopping condition if both occur at the same time since the chaining condition causes a new CCR enable/disable bit to be fetched from memory.

- Bit 12-10. Encoded DMA interrupt options.

INT2	INT1	INT0	Interrupt Option
0	0	0	Disabled
0	0	1	TC=0
0	1	0	EOP
0	1	1	TC=0 "or" EOP
1	0	0	Reserved
1	0	1	Reserved
1	1	0	Reserved
1	1	1	Reserved

- Bit 9-7. Encoded list chaining enabling options.

LCH2	LCH1	LCH0	Chaining Option
0	0	0	Disabled
0	0	1	TC=0
0	1	0	EOP
0	1	1	TC=0 "or" EOP
1	0	0	Reserved
1	0	1	Reserved
1	1	0	Reserved
1	1	1	NOP

The chaining NOP selection, when detected, immediately causes the next CDB in memory to fetched without processing the CDB containing the NOP.

- Bit 6-5. DMA port size. These two bits define the data port size of the DMA device.

PS1	PS0	Data Port Size
0	0	8-bit
0	1	16-bit
1	0	32-bit
1	1	Reserved

- Bit 4. End-of-Process direction control. If this bit is set to a '1', the EOP signal is an input. If this bit is set to a '0', the EOP signal is an output.

  If the channel is a transmit DMA channel, and this bit is '0', then the EOP signal is driven synchronously with the last byte transferred when TC=0. For a transmit channel, if this bit is '1', then EOP is an asynchronous input that can causing chaining based on how bits 7-9 are programmed. Asynchronous EOP's must only occur on 16-byte boundaries for Tx DMA.

  If the channel is a receive DMA channel, and this bit is '0', then the EOP signal is disabled as an input. For a receive channel, if this bit is '1', then EOP is an input that can causing chaining based on how bits 7-9 are programmed.

- Bit 3. EOP asynchronous input enable. When this bit is set to '1' the Vero chip is enabled to detect an asynchronous EOP input. When reset, the chip will not detect asynchronous EOP inputs. The Vero chip is ALWAYS enabled to detect synchronous EOP inputs.

- Bit 2. DMA transfer synchronization option. When this bit is set to '0' all DMA transfers are synchronized by the DREQ input pins of the Vero chip. The DREQ signal must become active for a DMA transfer to occur. When this bit is set to '1' DMA transfers are not synchronized to the DREQ input pins. DMA transfers will begin as soon as the CCR bit 0 is enabled and will continue until a channel stopping condition is reached.

- Bit 1. -Receive/+transmit indicator. 0 = channel is servicing receive DMA requests; 1 = channel is servicing transmit DMA requests. For receive requests, data transfer is FROM AIB Bus TO Local Bus. For transmit requests, data transfer is FROM Local Bus TO AIB Bus.

- Bit 0. +Enable/-disable channel. 0 = channel disabled to service DMA requests; 1 = channel enabled to service DMA requests. This bit can be used to disable the channel at any time. Re-enabling the channel will cause the DMA operation to start where it left off. Also, any programming option that disables the channel will be reflected in this bit. This bit is updated with its corresponding bit during list chaining reads from memory. This bit can also be set/reset via the DCCR. (see2.2.2, “DMA Channel Command Registers (DCCR 0-7)” on page 26)

  If queuing is disabled in the LBCR (see 2.6.2, “Local Bus Configuration register (LBCR)” on page 42), then this bit can not be set unless a non-zero value has been previously written to the Queue Count portion of the TQC register (see 2.2.5, “Transfer/Queue Count Register (TQC)” on page 29).

  Note: This bit or bit 0 of the DCCR registers alone should not be used as a status bit to determine if Vero is disabled, stopped, or reached the end of a chain. Instead the user should use both bit 31 and bit 0 of the CCR to determine the status of Vero during a linked list chaining. The following is an example assuming that bit 31 and bit 0 of the CCR are set in the main CDB's and are reset in the dummy CDB's :

Bit 31	Bit 0	Vero status
1	1	Running
1	0	Chaining
0	0	Stopped.

Reset Conditions

```
POWER-UP:       U000 0000 00UU UUUU 0000 0000 0UU1 0UU0
RESET COMMAND:  S000 0000 00SS SSSS SSSS SSSS SSSS SSS0
```

2.2.2 DMA Channel Command Registers (DCCR 0-7)

Description: The DCCR supports two commands. One allows for the bit manipulation of the CCR enable/disable bit. This bit allows CCR bit 0 to be set/reset without affecting the other CCR bits. A second command allows internal logic to be reset to a known state if an error condition causes a channel to stop before completion. A separate command register exists to control each DMA channel.**:** The command to enable a channel should not be issued with the same write that releases the channel from a reset command.

Register Format

```
(80960 Address = 1FF8x020h) r/w     x = channel #
```

31 2	1	0
Reserved	RES	DIS

Bit Descriptions

- Bit 7-2. Reserved. Always read as '0'.
- Bit 1. Reset channel command. Setting this bit to '1' causes a channel to be reset as shown by the Reset Command for each register. The channel is held in the reset state until the bit is set back to '0'.
- Bit 0. Disable channel command. Bit 0 = '0' defines the command to disable the channel. Bit 0 = '1' defines the command to enable the channel. Bit 0 of the CCR, if read, will reflect a disable command if it is issued through the DCCR.

  When a channel is disabled, any internally pending operations are performed before the channel stops servicing DMA requests.

Reset Conditions

```
POWER-UP:       0000 0000 0000 0000 0000 0000 0000 0000
RESET COMMAND:  0000 0000 0000 0000 0000 0000 0000 00S0
```

2.2.3 Global DMA Command Register (GDCR)

Description: The GDCR allows two commands to be issued globally to all DMA channels at once. It allows all DMA channels to be 'stopped' with one command or all DMA channels to be reset with one command. The 'stopped' state of each channel is reflected in the EN/DIS bit of the CCR.**:** The command to re-start all channels should not be issued with the same write that releases all channels from a reset command.

Register Format

```
(80960 Address = 1FF88000h) r/w
```

31 2	1	0
Reserved	RES	STP

Bit Descriptions

- Bits 7-2. Reserved. Always read as 0.
- Bit 1. Reset all channels command. Setting this bit to '1' causes all the channels to be reset as shown by the Reset Command for each register. All channels are held in the reset state until the bit is set back to '0'.
- Bit 0. Stop all channels command. If written to a '1' all channels are stopped. If written to a '0' all channels are re-started.

  When a channel is 'stopped', all internally pending operations are performed before the channel stops servicing DMA requests.

Reset Conditions

```
POWER-UP:       0000 0000 0000 0000 0000 0000 0000 0000
RESET COMMAND:  0000 0000 0000 0000 0000 0000 0000 00S0
```

2.2.4 Memory Pointer Register (MPR)

Description: The MPR contains the 32-bit Local Bus address of the next data byte to be DMA'ed. There are no alignment restrictions on the address programmed into this register.

Register Format

```
(80960 Address = 1FF8x00Ch) r/w     x = channel #

 31                                                            0
+--------------------------------------------------------------+
|                      Memory Pointer                          |
+--------------------------------------------------------------+
```

Reset Conditions

```
POWER-UP:       UUUU UUUU UUUU UUUU UUUU UUUU UUUU UUUU
RESET COMMAND:  SSSS SSSS SSSS SSSS SSSS SSSS SSSS SSSS
```

2.2.5 Transfer/Queue Count Register (TQC)

Description: The TQC register contains two separate count values. Bits 0-15 contains the current DMA transfer count in bytes. Values from 0001h to FFFFh can be programmed. This allows DMA transfers of from '1' to '64KB-1' to be transferred. This register is automatically saved into memory when a chain event occurs. This is shown in Figure 2 on page 34. When a receive DMA channel has been programmed with a port size of 32-bits, transfer count values 0001h, 0002h and 0003h are illegal and should not be programmed. When a receive DMA channel has been programmed with a port size of 16-bits, a transfer count value of 0001h is illegal and should not be programmed.

For a transmit DMA channel, a TC=0 condition is always defined as the transfer count value going from 0001h to 0000h. For a receive DMA channel, a TC=0 condition depends on the port size. For a 32-bit port a TC=0 condition is when the transfer count value is less than 4. For a 16-bit port a TC=0 condition is when the transfer count value is less than 2. For an 8-bit port a TC=0 condition is when the transfer count value is 0000h.

Bits 16-27 of this register contains a count of the number of CDB's that have been queued by the Queuing Register (QR). This allows up to '4K-1'(or 'FFF'h) CDB's to be queued per DMA channel. Each time the QR is written, this register is incremented by one. **The queue count should never be allowed to increment from 'FFF'h to '000'h.** Each time a CDB finishes processing (a stop condition or a chain condition defines the end of a CDB) this register is decremented by one. Software must initialize the Queue Count to '000'h if queuing is enabled in the LBCR (see 2.6.2, “Local Bus Configuration register (LBCR)” on page 42). If queuing is disabled in the LBCR then the Queue Count must be written to a non-zero value or the CCR enable/disable will not be able to be set to '1' to enable the DMA channel.

If the queue count register equals '0001'h and a chain event occurs, it will decrement to '0000'h. This means that another CDB has not been queued fast enough. In this case, the channel is automatically stopped and an error interrupt (queue underrun) is posted.

Register Format

```
(80960 Address = 1FF8x010h) r/w     x = channel #
```

31 28	27 16	15 0
Reserved	Queue Count	DMA Byte Count

Reset Conditions

```
POWER-UP:       0000 UUUU UUUU UUUU UUUU UUUU UUUU UUUU
RESET COMMAND:  0000 SSSS SSSS SSSS SSSS SSSS SSSS SSSS
```

Note: Never write a zero value to the DMA byte count portion, either directly or by linked list chaining. Also, a non-zero value must be written to the DMA byte count portion of this register for all DMA channels whenever a hardware reset has occurred.

2.2.6 Chain Pointer Register (CPR)

Description: The CPR contains the 32-bit address pointer that points to the memory location where the DMA controller will fetch the CDB for the next buffer when a chain event occurs. The lower 2 bits of this register should always be programmed to '00' so that the chain pointer is 4 byte aligned. The following two conditions define a chaining event.

1. terminal count has been reached and list chaining for terminal count is enabled in the CCR,
2. an End-Of-Process condition has occurred and list chaining for end-of-process is enabled in the CCR.

Note: The CPR should always point to a valid memory location. This location should contain a valid "dummy" CDB with the channel control set to stop the channel and disable list chaining.

Register Format

```
(80960 Address = 1FF8x014h) r/w     x = channel #
```

31	0
Chain Pointer	

Reset Conditions

```
POWER-UP:       UUUU UUUU UUUU UUUU UUUU UUUU UUUU UUUU
RESET COMMAND:  SSSS SSSS SSSS SSSS SSSS SSSS SSSS SSSS
```

2.2.7 AIB Address 1/2 Register (AIB_ADDR 1/2)

Description: The AIB_ADDR 1/2 register contains two separate 16-bit address fields. Bits 0-15 defines the first AIB address of two possible I/O operations that the hardware will perform when a chain event occurs. Bits 16-31 defines the second AIB address of two possible I/O operations that the hardware will perform when a chain event occurs. The first operation (read or write) is directed at the address defined by bits 0-15 of this register. For a write operation, the data is defined by the value in the AIB_OP1 DATA register. For read operations, the data read is stored in memory at the CPR address (the first address of the next CDB). Also, since the AIB supports an 19-bit (512KB) address space, devices which use this feature must reside within the lower 64K of the 512K AIB address space. The second operation (read or write) is directed at the address defined by bits 16-31 of this register. This operation is the same as the first except that the data written to the AIB is that contained in the AIB_OP2 DATA register or may be a pre-defined fixed value, as defined in CCR bits 20-21.

Register Format

```
(80960 Address = 1FF8x000h) r/w     x = channel #

 31                          16 15                            0
 ----------------------------------------------------------------
|     2nd AIB_OP Address         |     1st AIB_OP Address         |
 ----------------------------------------------------------------
```

Reset Conditions

```
POWER-UP:       UUUU UUUU UUUU UUUU UUUU UUUU UUUU UUUU
RESET COMMAND:  SSSS SSSS SSSS SSSS SSSS SSSS SSSS SSSS
```

2.2.8 AIB_OP1 DATA Register (AIB_OP1)

Description: The AIB_OP1 register contains the data field of the first of two operations upon a chain event. When a chain event occurs, the DMA controller can be programmed to perform up to two I/O operations to any AIB Bus address in the lower 64K of the AIB address space. The first operation (read or write) is directed at the address defined by bits 0-15 of the AIB_ADDR 1/2 register. For a write operation, the data is defined by this register.

Register Format

```
(80960 Address = 1FF8x004h) r/w     x = channel #
```

```
 31                                                          0
+-------------------------------------------------------------+
|                      1st AIB_OP Data                        |
+-------------------------------------------------------------+
```

Reset Conditions

```
POWER-UP:       UUUU UUUU UUUU UUUU UUUU UUUU UUUU UUUU
RESET COMMAND:  SSSS SSSS SSSS SSSS SSSS SSSS SSSS SSSS
```

2.2.9 AIB_OP2 DATA Register (AIB_OP2)

Description: The AIB_OP2 register contains the data field of the second of two operations upon a chain event. When a chain event occurs, the DMA controller can be programmed to perform up to two I/O operations to any AIB Bus address in the lower 64K of the AIB address space. The second operation (read or write) is directed at the address defined by bits 16-31 of the AIB_ADDR 1/2 register. For a write operation, the data is defined by this register or may be a pre-defined fixed value, as defined by bits 20-21 of the CCR.

Register Format

```
(80960 Address = 1FF8x008h) r/w     x = channel #
```

```
 31                                                          0
+-------------------------------------------------------------+
|                      2nd AIB_OP Data                        |
+-------------------------------------------------------------+
```

Reset Conditions

```
POWER-UP:       UUUU UUUU UUUU UUUU UUUU UUUU UUUU UUUU
RESET COMMAND:  SSSS SSSS SSSS SSSS SSSS SSSS SSSS SSSS
```

2.2.10 Queue Tail Pointer Register (QTP)

Description: The QTP register contains the starting memory address of the last CDB that was queued by the QR (see 2.4, “Queuing Operation (NOT Supported)” on page 36). The hardware manages this register so that it can keep track of CDB's that are to be processed by the DMA controller. This is a read only register.

Register Format

```
(80960 Address = 1FF8x01Ch) ro      x = channel #

 31                                                            0
+---------------------------------------------------------------+
|                      Queue Tail Pointer                       |
+---------------------------------------------------------------+
```

Reset Conditions

```
POWER-UP:       UUUU UUUU UUUU UUUU UUUU UUUU UUUU UUUU
RESET COMMAND:  SSSS SSSS SSSS SSSS SSSS SSSS SSSS SSSS
```

2.3 Linked List Chaining (LLC)

The Vero DMA controller supports a feature known as Linked List Chaining (LLC), sometimes called scatter/gather DMA. This feature allows the programmer to create, in memory, lists of different data buffer areas and buffer counts (CDB's) that can be automatically reloaded to the CDT after a terminal count or end-of-process condition for the present data buffer has occurred. The channel can optionally interrupt or not interrupt the 80960 when any of these conditions has occurred. Also, the DMA channel can independently be stopped or not stopped at this point. In addition certain control information for the channel can be changed during the chaining operation.

Within the CDT, a 32-bit address pointer (CPR) is maintained. This points to a CDB in memory. At the time of terminal count or end-of-process detection, if linked list chaining is enabled, the hardware will do three memory writes followed by seven memory reads starting at this chain pointer address. It stores into memory the present DMA transfer byte count, and any data read from the AIB Bus by AIB_OP1 or AIB_OP2. It then fetches seven new words of CDB information as detailed below. At the end of this operation, the CPR in the CDT will be the CPR value that was just read from the CDB in memory. This address is the starting address of the next CDB in the chain. This provides a mechanism to off-load the 80960 from processing DMA terminal count or end-of-process interrupts. In theory, this list of buffers could occupy any amount of free memory. This concept is detailed in Figure 2. Note: Make sure that linked list chaining does not load a DMA byte count of zero.

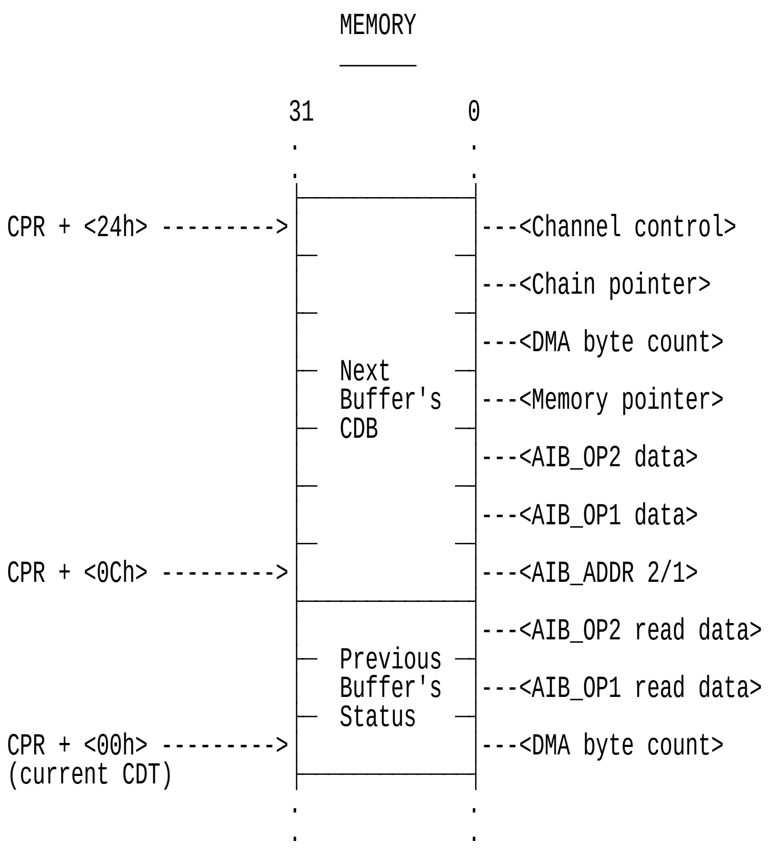


Figure 2. Linked List Chaining

Please note that the hardware will always perform the three memory writes regardless of whether the option to do reads from the AIB is chosen. If this option is not chosen the data written to these two fields will be indeterminate.

2.3.1 Linked List Chaining/Stopping Matrix

The CCR provides many different options for linked list chaining and stopping DMA channels (see 2.2.1, “Channel Control Register (CCR)” on page 23). The chaining options and the stopping options must be looked at together to determine the action a DMA channel takes when these conditions are encountered during DMA channel execution. The matrix below describes how a DMA channel operates for all possible combinations of chaining and stopping options being programmed into the CCR.

Summary: If chaining is disabled, the DMA channel is stopped by the programmed option. If chaining is enabled, the stop programming options are ignored unless the programmed stop option is different from the programmed chaining option. The state of the CCR enable bit in a memory CDB determines whether a DMA channel remains enabled or disabled after the chaining operation occurs.

		STOP OPTIONS			
		DO NOT STOP	TC=0	EOP	TC=0|EOP
CHAIN OPTIONS	DISABLED	1	2	2	3
	TC=0	4	4	5	5
	EOP	6	7	6	7
	TC=0|EOP	8	8	8	8

Figure 3. CCR Chaining/Stopping Matrix

1. DMA channel detects TC=0 but continues DMA'ing data to the next MPR address.
2. DMA channel stops servicing DMA requests when condition is detected.
3. DMA channel stops servicing DMA requests when either condition is detected.
4. DMA channel stops servicing DMA requests when TC=0 is detected, performs chaining operation, then is either re-enabled or stays disabled based on new CCR enable bit that is chained in.
5. DMA channel operates as in (4), but will stop servicing DMA requests, without chaining, if EOP is detected.
6. DMA channel stops servicing DMA requests when EOP is detected, performs chaining operation, then is either re-enabled or stays disabled based on new CCR enable bit that is chained in.
7. DMA channel stops servicing DMA requests when EOP is detected, performs chaining operation, then is either re-enabled or stays disabled based on new CCR enable bit that is chained in. If TC=0 occurs, the DMA channel stops servicing DMA requests with no chaining.
8. DMA channel stops servicing DMA requests when either condition is detected, performs chaining operation, then is either re-enabled or stays disabled based on new CCR enable bit that is chained in.

2.4 Queuing Operation (NOT Supported)

The Vero DMA controller supports hardware management of data buffers. This is done through the Queuing Register (QR). The QR is a read/write command register. There is one physical QR for all eight DMA channels. However, the register is writable at eight different addresses by the 80960. Each different address defines a queuing operation for a separate DMA channel.

The data written to the QR defines a pointer into memory. This pointer points to the next CDB to be linked on to a chain of CDB's. Only the most significant 30 bits of the QR pointer is defined as an address. This means the pointer will always be addressing the next CDB on a 4 byte boundary.

Software must use the QR to initialize a DMA channel as well as to queue another CDB. Also, the queue count register within the TQC register must be written to zero at initialization. Queuing operation can be optionally disabled in the LBCR (see 2.6.2, “Local Bus Configuration register (LBCR)” on page 42). If this is done, the queue count register within the TQC register must be written to a non-zero value at initialization.

The QR is a single resource shared by all DMA channels. In order to issue a queuing command, the software must first poll the QR Busy bit, bit 0 of of the QR. If this bit is set it indicates the queuing resource is currently owned by another process. This means either the hardware is now waiting for a write to the QR or the hardware is processing a recently queued CDB. In either case the software must wait (usually by polling) until the Busy bit is reset. If the software reads the QR Busy bit reset, the hardware automatically sets it (atomic operation) to lock out all other queuing requests. The software then writes the QR, and the command is executed. Typical latency to perform a queuing operation should be less than 2 microseconds. After completing the queue operation the hardware will reset the Busy bit. If another write to the QR is detected it will cause a Queuing Error interrupt to the 80960. The pending command will execute and the requested command will be ignored.

The hardware will perform three different operations depending on the value of the current queue count when a queuing operation takes place.

Queue Count = 0: The hardware fetches a complete CDB from memory at the address written to the QR. This is actually the identical operation of a list chaining event. It consists of **3** Local Bus writes followed by **7** Local Bus reads. The CDB offset in memory must account for the 3 writes. If linked list chaining is enabled for the first queued CDB, the CPR address value of that CDB should point to a pre-determined 'null' CDB. This 'null' CDB provides the first CDB with an area to save the 'write' information that is stored during linked list chaining. Also, the CCR value of the 'null' CDB should be '00000000'h to guarantee that the channel is disabled after the linked list chaining operation takes place.

Queue Count = 1: The hardware writes the address value written to the QR directly into the internal Chain Pointer register. No external Local Bus operation takes place. The hardware also writes the QR address value into the Queue Tail Pointer register.

Queue Count > 1: The hardware writes the address value written to the QR into the CPR location of the CDB pointed to by the QTP address value. This consists of **1** Local Bus write operation. The hardware then writes the QR address value into the Queue Tail Pointer register.

2.4.1 Queuing Register (QR)

Description: See 2.4, “Queuing Operation (NOT Supported)” on page 36.

Register Format

31 … 2	1	0
Queue Pointer Address (30 bits) (write only)	X	QR BUSY

Note: Queue Pointer Addess field is a write only field. Invalid address is returned when read.

- DMA channel 0 queuing address - 1FF80024h
- DMA channel 1 queuing address - 1FF81024h
- DMA channel 2 queuing address - 1FF82024h
- DMA channel 3 queuing address - 1FF83024h
- DMA channel 4 queuing address - 1FF84024h
- DMA channel 5 queuing address - 1FF85024h
- DMA channel 6 queuing address - 1FF86024h
- DMA channel 7 queuing address - 1FF87024h

Reset Conditions

```
POWER-UP:       UUUU UUUU UUUU UUUU UUUU UUUU UUUU UU00
RESET COMMAND:  SSSS SSSS SSSS SSSS SSSS SSSS SSSS SS0S
```

Note: The reset values indicated above is for DCCR command reset. When using GDCR command reset the hardware also reset the BUSY bit 0.

Queuing Flow Example:

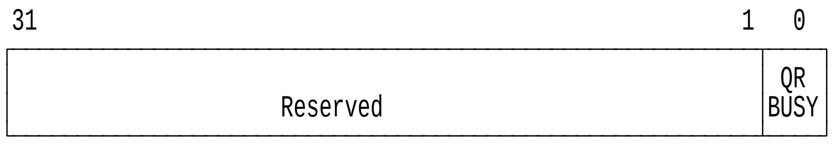
1. Software must write the queue count (bits 16-27 of the TQC) = '0000'h. This tells the hardware that 0 CDB's have so far been queued
2. Software sets up the first CDB in memory.
3. Software now checks for free QR by reading the BUSY bit.
    a. If not busy, go to 4 (hardware sets busy bit)
    b. If busy, go to 3
4. Software writes QR. The value written is an address pointer the next CDB in memory to be queued.
5. Hardware executes command.
6. Hardware resets busy bit upon completion

2.4.2 Queuing Status register (QSR)

Description: The QSR can be read to interrogate the QR Busy bit without risk of setting it as would normally occur during a queuing operation. Also, a write to this register address with any data will cause the QR busy bit to be reset.

Register Format

```
(80960 Address = 1FF8800Ch) r/w
```



Bit Descriptions

- Bits 31-1. Reserved.
- Bit 0. QR Busy bit. This bit, when set, indicates that a QR operation in currently in process.

Reset Conditions

```
POWER-UP:       0000 0000 0000 0000 0000 0000 0000 0000
RESET COMMAND:  0000 0000 0000 0000 0000 0000 0000 000S
```

Note: The reset values indicated above is for DCCR command reset. When using GDCR command reset the hardware also reset the BUSY bit 0.

2.5 DMA Interrupts

Each of the 8 DMA channels has 7 possible sources of interrupts to the 80960. Of these 7, two are classified as 'normal termination' interrupts and five are classified as 'error' interrupts. Normal termination interrupts are maskable in the CCR and error interrupts are non-maskable.

Each DMA channel has a fixed interrupt vector assigned to it. These 8 interrupt vectors are at priority level 8 within the 80960 interrupt model. The following vectors are assigned to the 8 DMA channels.

Table 11. Vero DMA Channel Vector Assignment

DMA Channel #	**Interrupt Vector #**
0	71
1	70
2	69
3	68
4	67
5	66
6	65
7	64

Each DMA channel has an interrupt status register (see 2.5.1, “DMA Interrupt Status Registers (DISR)” on page 40) associated with it that is read during the interrupt subroutine to determine the pending interrupt status bits for that channel. This read clears all active status bits.

2.5.1 DMA Interrupt Status Registers (DISR)

Description: Each DMA channel has a DMA Interrupt Status register (DISR) associated with it. This status register is cleared when read. Since 7 different conditions can cause the interrupt, this means that it is possible for multiple conditions to be present when the status register is read. However, since the register is cleared by the read, only one interrupt will be presented to the 80960. Therefore, the interrupt service routine must be capable of servicing all of the pending status bits.

Register Format

```
(80960 Address = 1FF8x028h) ro      x = channel #
```

31 8	7	6	5	4	3	2	1	0
Reserved	QE	QUE	LPR	LEX	APR	x	EOP	TC0

Bit Descriptions

- Bit 31 - 8. Reserved.
- Bit 7. QE. This is the queuing error status bit. It is set if a write takes place to the QR while the QR busy bit is set. The pending command will execute but the requested command will be ignored.
- Bit 6. QUE. This bit is the queue underrun status bit. It is set when the queue count decrements from '001'h to '000'h after a linked list chaining operation takes place. This can be interpreted by software as a completion indicator that all required CDBs have completed, or as an error indicator that no CDBs are currently queued. In this case, the channel is automatically stopped.
- Bit 5. LPR. This bit, when set, indicates a Local Bus parity error has occurred when the DMA channel was reading data from the Local Bus. In this case, the channel is automatically stopped.
- Bit 4. LEX. This bit, when set, indicates a Local Bus exception error occurred. This happens when a Local Bus slave drives the Exception signal while Vero is the Local Bus master. In this case, the Vero DMA channel is automatically stopped.
- Bit 3. APR. This bit, when set, indicates an AIB Bus parity error has occurred when the DMA channel was reading data from the AIB Bus. In this case, the channel is automatically stopped.
- Bit 2. Reserved.
- Bit 1. EOP. This bit, when set, indicates an end-of-process condition has occurred for the DMA channel. This bit is set upon detecting an EOP regardless of the state of its corresponding interrupt enable bit.
- Bit 0. TC0. This bit, when set, indicates a terminal count condition has been reached for the DMA channel. This bit is set upon detecting a TC regardless of the state of its corresponding interrupt enable bit.

Reset Conditions

```
POWER-UP:       0000 0000 0000 0000 0000 0000 0000 0000
RESET COMMAND:  0000 0000 0000 0000 0000 0000 SSSS S0SS
```

2.6 DMA Miscellaneous Registers

2.6.1 DMA FIFO Residual Count registers (DFRC0-7)

Description: The DFRC register can be read when a DMA channel is stopped or known to be inactive to determine how many bytes of data are in the FIFO. The register contains two 8-bit fields, one of which is the value of the AIB side counter, and the other of which is the value of the LBUS side counter.

For a transmit DMA channel, the LBUS side counter increments as bytes are loaded into the FIFO from the Local Bus, and the AIB side counter increments as these bytes are sent to the AIB Bus. Software must subtract the AIB side count value from the LBUS side count value to obtain a residual count value of many bytes are still in the fifo waiting to be sent to the AIB Bus.

For a receive DMA channel, the AIB side counter increments as bytes are loaded into the FIFO from the AIB Bus, and the LBUS side counter increments as these bytes are sent to the Local Bus. Software must subtract the LBUS side count value from the AIB side count value to obtain a residual count value of many bytes are still in the fifo waiting to be sent to the Local Bus.

Register Format

```
(80960 Address = 1FF8x02Ch) ro      x = channel #
```

31 16	15 8	7 0
Reserved	AIB side count value	LBUS side count value

Reset Conditions

```
POWER-UP:       0000 0000 0000 0000 0000 0000 0000 0000
RESET COMMAND:  0000 0000 0000 0000 0000 0000 0000 0000
```

2.6.2 Local Bus Configuration register (LBCR)

Description: The LBCR enables various options associated with the Local Bus operation of the chip.

Register Format

```
(80960 Address = 1FF88008h) r/w
```

31 28	27 2	1	0
GAID	Reserved	QE	PE

Bit Descriptions

- Bit 31-28. Gate array ID. These bits, which are read only, read as '0000' for revision 1 of the chip. The Gate array ID will NOT change for revision 2 of the chip.
- Bit 27-2. Reserved.
- Bit 1. Queuing enable. When reset, queuing operation through use of the Queuing register (see 2.4, “Queuing Operation (NOT Supported)” on page 36) is enabled. When set, queuing is disabled. Queuing disabled allows for circular buffer operation which is not possible when queuing is enabled.
- Bit 0. Address/data parity enable. When set, address/data parity checking on the Local Bus is enabled. When reset, address/data parity checking on the Local Bus is disabled. Address and data parity are always generated on the Local Bus.

Reset Conditions

```
POWER-UP:       0000 0000 0000 0000 0000 0000 0000 0000
RESET COMMAND:  0000 0000 0000 0000 0000 0000 0000 00SS
```

3.0 Vero Interrupt Controller

3.1 General

The general features of the interrupt controller subsystem are shown below.

- 4 encoded interrupt inputs from the Miami chip
- 4 encoded interrupt inputs from the Brighton chip
- 2 separate interrupt inputs from the Local Bus (reserved)
- 5 separate interrupt inputs from the AIB (1 high priority)
- support for both vectoring and non-vectoring AIB devices
- programmable single or double (8259 type) IACK pulse
- programmable IACK pulse width
- programmable interrupt masking of the Local Bus
- programmable interrupt masking of the AIB Bus inputs

The Vero module supports 80960 expanded mode interrupting. An 8-bit interrupt vector bus is provided to directly attach to the 80960. The Vero chip takes the interrupt inputs and translates these into interrupt vectors that it places on the interrupt bus for interpretation by the 80960. However, support is provided to allow 2 AIB devices to directly supply the 80960 with an interrupt vector. This is a programmable option. Table 12 on page 44 shows the priority scheme that is used to report various classes of interrupts back to the 80960.

Table 12. Vero DMA Channel Vector Assignment

80960 Priority Level	**Vector #**	**Interrupt Condition**	**Interrupt Source (see Note)**
31	255	Watchdog timeout	Brighton (0000)
31	254	Multi-bit ECC error	Brighton (0001)
31	253	Micro Channel NMI Command	Miami (0000)
30	243	Local Bus Exception	Brighton (0010)
30	242	Local Bus Parity/Exception error	Miami (0001)
30	241	Local Bus parity error w/80960 master	Brighton (0011)
30	240	AIB Bus read parity error w/80960 master	Vero
29	235	Memory protect error w/80960 master	Brighton (0100)
29	234	Memory protect error Miami master	Brighton (0101)
29	233	Reserved	Brighton (0110)
29	232	AIB error input signal	AIB
28	224	Local Bus INT0 input	Local Bus
26	209	Serial debug port - receive	Brighton (0111)
26	208	Serial debug port - transmit	Brighton (1000)
25-21	207-168	AIB INT0 (vector sourced by AIB device)	AIB
20-16	167-128	AIB INT1 (vector sourced by AIB device)	AIB
15	120	AIB INT2	AIB
14	112	AIB INT3	AIB
12	100	SCB Attention Port	Miami (0010)
12	99	Bus master EOT	Miami (0011)
12	98	Interrupt command	Miami (0100)
12	97	Bus master Channel 2	Miami (0101)
12	96	Bus master Channel 1	Miami (0110)
11	95-88	Reserved	Miami
10	80	Reserved	Brighton
8	71-64	Vero DMA channel 0-7	Vero
7	58	Local Bus INT1 input	Local Bus
4	35	Timer 1 (software)	Brighton (1010)
4	34	Timer 2 (time of day)	Brighton (1011)
4	33	Timer 3 (performance)	Brighton (1100)
4	32	Timer 4 (time slice)	Brighton (1101)
1	8	Single-bit ECC error	Brighton (1110)

Note: The value in parentheses corresponds to the signals **MIA_INT(3-0)** for Miami sourced interrupts, or to **BRI_INT(3-0)** for Brighton sourced interrupts.

The interrupt controller manages prioritizing the interrupts that these devices generate. The priority is fixed with 'single-bit ECC error' being the lowest and the NMI sources being the highest. Interrupts are not latched by the interrupt controller. The ouput of the interrupt controller to the 80960 interrupt bus reflects the value of the highest priority input that is active.

3.2 Programmable Options

3.2.1 Interrupt Initialization register (IIR)

Description: The IIR register controls the initialization parameters for the Vero chip interrupt controller logic. This register must be programmed before interrupts are generated from the AIB.

Register Format

```
(80960 Address = 1FF88010h) r/w


  15  14  13  12  11  10  9   8   7   6   5   4   3   2   1   0
| interrupt ack 1 control field | interrupt ack 0 control field |
|                               |                               |
| inactive  |  active    |control| inactive  |  active    |control|
```

INA PW2	INA PW1	INA PW0	ACK PW2	ACK PW1	ACK PW0	OTP ACK	IAC EN	INA PW2	INA PW1	INA PW0	ACK PW2	ACK PW1	ACK PW0	OTP ACK	IAC EN

Bit Descriptions

- Bits 31-16. Reserved.
- Bits 15-8. Interrupt level 1 control field.
- Bits 7-0. Interrupt level 0 control field.

Each 8 bit control field defines the interrupt controller operation for that interrupt level. All control fields are equivalent.

- IAC EN: Interrupt acknowledge enable. When set to '0' the INT0 (or INT1) input functions as a direct interrupt input from the AIB. No interrupt acknowledge cycle will be run. When using this mode an interrupt will generate the lowest vector in the defined range for the interrupt (168 for INT0, 128 for INT1). When set to '1' an interrupt will cause an interrupt acknowledge cycle to be run on the AIB Bus. The vector during the interrupt acknowledge cycle will be presented to the 80960 if it is within the defined range. If it is not within the defined range, the highest vector in the defined range (207 for INT0, 167 for INT1) will be given instead.
- OTP ACK: One or two pulse interrupt acknowledge signal. Some devices use an 8259 type interrupt acknowledge cycle (two pulse). Others require external logic to eliminate the first pulse. This bit should reduce or eliminate the need for that external logic.
  - OTP ACK=0: one pulse.
  - OTP ACK=1: two pulses.

  If two pulses are selected, an idle time is inserted between the two pulses for device recovery, the length of which is controlled by INA PW2-0.
- ACK PW2-0: Interrupt acknowledge cycle active pulse width. These three bits provide eight options for the interrupt acknowledge cycle active length. Some devices respond slower than others and therefore would require external logic to insert wait states into the interrupt acknowledge cycle. These bits should eliminate the need for this external logic. These bits allow a programmable active time from 120ns to 680ns in 80ns increments for the interrupt acknowledge pulse.

ACK PW2	ACK PW1	ACK PW0	INTACK 'low' (active) pulse width
0	0	0	120ns
0	0	1	200ns
0	1	0	280ns
0	1	1	360ns
1	0	0	440ns
1	0	1	520ns
1	1	0	600ns
1	1	1	680ns

- INA PW2-0: Interrupt acknowledge recovery (inactive) width. These three bits provide eight options for the interrupt acknowledge recovery length. Some devices are slower than others to release (float) data signals on the AIB Bus, and thus require a certain amount of bus idle time after the interrupt acknowledge cycle is completed. These bits allow a programmable inactive time from 120ns to 680ns in 80ns increments.

INA PW2	INA PW1	INA PW0	INTACK 'high' (inactive) pulse width
0	0	0	120ns
0	0	1	200ns
0	1	0	280ns
0	1	1	360ns
1	0	0	440ns
1	0	1	520ns
1	1	0	600ns
1	1	1	680ns

If the OTP ACK bit is 1 (two pulses), both interrupts acknowledge pulses will be the length set by ACK PW2-0, with an inactive time between and after them determined by INA PW2-0.

Either or both interrupt acknowledge cycles can be extended by the AIB READY signal in increments of 40ns.

Reset Conditions

```
POWER-UP:       0000 0000 0000 0000 UUUU UUU0 UUUU UUU0
RESET COMMAND:  0000 0000 0000 0000 SSSS SSSS SSSS SSSS
```

3.2.2 Interrupt Mask register (IMR)

Description: The IMR allows the four AIB interrupt inputs, the two Local Bus interrupt inputs, and the AIB Error input to be enabled and disabled from causing interrupts to the 80960 without programming the interrupting device directly. This allows for a point of synchronization between the 80960 and the interrupting device. Typically, devices contain their own interrupt enable functions. However, these are sometimes not usable on a realtime basis since spurious interrupts can result if an interrupt is disabled within a device immediately after its interrupt output has been asserted. This register prevents the possibility of these spurious interrupts if the device takes no precautions to do this.

Register Format

```
(80960 Address = 1FF88014h) r/w
```

31	7	6	5	4	3	2	1	0
Reserved		AER	LI1	LI0	AI3	AI2	AI1	AI0

Bit Descriptions

- Bit 31-7. Reserved.
- Bits 6. AIB_ERROR input mask bit. This bit can mask the AIB_ERROR input signal from generating an interrupt to the 80960. The disabling of these bits is synchronized with the interrupt input signal to prevent spurious interrupts when disabling. Interrupt are enabled when a bit is set to '1' and disabled when set to '0'.
- Bits 5-4. LB_INT1-0 mask bits. These bits can individually mask the 2 Local Bus interrupt inputs from generating an interrupt to the 80960. The disabling of these bits is synchronized with the interrupt input signal to prevent spurious interrupts when disabling. Interrupt are enabled when a bit is set to '1' and disabled when set to '0'.
- Bits 3-0. AIB_INT3-0 mask bits. These bits can individually mask the 4 AIB interrupt inputs from generating an interrupt to the 80960. The disabling of these bits is synchronized with the interrupt input signal to prevent spurious interrupts when disabling. Interrupt are enabled when a bit is set to '1' and disabled when set to '0'.

Reset Conditions

```
POWER-UP:       0000 0000 0000 0000 0000 0000 0000 0000
RESET COMMAND:  0000 0000 0000 0000 0000 0000 0SSS SSSS
```

3.2.3 Interrupt Status register (ISR)

Description: The ISR allows all Vero interrupt input sources to read through a single status register.

Register Format

```
(80960 Address = 1FF88018h) ro

 31                                                       0
 ___________________________________________________________
|                 Pending Interrupt Status                  |
|___________________________________________________________|
```

Bit Descriptions

- Bits 31-24. Reserved.
- Bits 23-20. AIB_INT 3-0 status bits.
- Bit 19. AIB error input status bit.
- Bit 18. AIB parity error status bit.
- Bits 17-16. Local Bus interrupt 1-0 status bits.
- Bits 15-12. Brighton interrupt 3-0 status bits.
- Bits 11-8. Miami interrupt 3-0 status bits.
- Bits 7-0. Vero DMA channel 7-0 status bits.

Reset Conditions

```
Not applicable
```

3.3 Commands

3.3.1 AIB INT0 End-of-Interrupt (EOI0) command

Description: The EOI0 command is used to inform the interrupt controller that the 80960 has cleared the AIB INT0 source. This command must be issued when for all interrupting AIB devices that use the Interrupt Acknowledge option in 3.2.1, “Interrupt Initialization register (IIR)” on page 46. This command should be issued after the command that goes directly to the AIB to clear the interrupting condition. The EOI0 command is issued by writing any data value to address '1FF89000'h.

Command Format

```
80960 ADDRESS = 1FF89000h
80960 DATA    = don't care
```

Reset Conditions

```
Not applicable
```

3.3.2 AIB INT0 End-of-Interrupt (EOI1) command

Description: The EOI1 command is used to inform the interrupt controller that the 80960 has cleared the AIB INT1 source. This command must be issued when for all interrupting AIB devices that use the Interrupt Acknowledge option in 3.2.1, “Interrupt Initialization register (IIR)” on page 46. This command should be issued after the command that goes directly to the AIB to clear the interrupting condition. The EOI1 command is issued by writing any data value to address '1FF8A000'h.

Command Format

```
80960 ADDRESS = 1FF8A000h
80960 DATA    = don't care
```

Reset Conditions

```
Not applicable
```

4.0 Vero AIB Bus Interface

4.1 General

The Vero module provides for an I/O bus (AIB) that is split from the Local Bus, where memory for program execution resides. This allows the DMA controller to only access the Local Bus when one of its 16 byte data buffers is empty (transmit) or full (receive). In this manner, only low speed non-bursted data movement occurs on the AIB bus, and only bursted data movement occurs on the Local Bus. This describes the nature of most data movement (DMA controlled) that occurs on the Juno Adapter.

The AIB Bus supports slave type devices. Slave devices can be directly accessed by the 80960 for initialization and status types of operations, and will typically use the Vero DMA controller to move data from the device to memory. The AIB Bus supports a 512K byte address space within the 80960 address space for addressing slave devices on the AIB. The starting 80960 address within this address space is 1FF00000h, and this extends up to 1FF80000h. Five separate regions can be defined within this 512KB address space with each region having a programmable number of wait states.

Only 80960 'stob' (store byte ordinal), 'stos' (store short ordinal), 'st' (store word), 'ldob' (load byte ordinal), 'ldos' (load short ordinal), and 'ld' (load word) are supported for direct data transfer to/from AIB Bus devices. Double, triple and quad word 80960 instructions are not supported to the AIB. Also, 80960 executable memory is not supported on the AIB.

4.1.1 AIB Arbiter

Contained within the Vero chip is the arbitration logic that determines which requesting master gains control of the AIB Bus to run its cycle. There are three requesting masters for the AIB Bus.

1. Local Bus master cycle to AIB
2. Vero DMA channel cycle to AIB
3. Interrupt Acknowledge cycle to AIB

The prioritization scheme is the order shown above. The Local Bus master will always be given highest priority. If another cycle is currently active on the AIB the Local Bus master will be granted the next cycle. Therefore, the maximum latency for the Local Bus master to begin its cycle is the maximum cycle time of a DMA channel cycle or interrupt acknowledge cycle.

The DMA channels also arbitrate amongst themselves. A DMA channel whose input -DREQ signal is held active will continue to be granted the AIB bus ahead of other DMA channels and interrupt acknowledge requests until its request is sampled inactive at the end of a DMA cycle, or until its DMA channel buffer inside Vero becomes full (for a receive channel) or empty (for a transmit channel). As stated above, a Local Bus master request for the AIB will interrupt this process and be granted immediately. Service priority among the DMA channels is fixed with channel 0 being the highest priority and channel 7 being the lowest priority.

The interrupt acknowledge cycle request is the lowest priority in the arbitration scheme. In order for this cycle request to be granted, both Local Bus master and DMA cycle requests must be inactive for at least one clock cycle. This can lead to extended servicing latencies based upon such things as the size of AIB device DMA fifos and AIB device DMA cycle time. For instance, for an 8-bit AIB device with a 16 byte DMA fifo, and a DMA cycle time of 200ns, the DMA controller will be able to run 16 200ns cycles (3.2 microseconds) before an interrupt acknowledge cycle is granted. It should be noted that using direct interrupt inputs from the AIB will provide more predictable interrupt response in a heavy DMA environment.

4.2 AIB Initialization registers

Before devices that exist on the AIB Bus can be accessed by the processor, a set of initialization registers must be programmed. These registers allow for a great degree of flexibility in the types of devices that can be supported on the AIB Bus. Many devices can be supported with a minimal amount of external interface hardware.

4.2.1 Chip Select Definition registers (CSD0-4)

Description: Each of the five chip select signals can be programmed for a base starting address on any 4KB boundary within the 512KB address space of the AIB. Also, each of the chip select signals can be programmed to decode a range of addresses from 4KB to 512KB. Also, each of the five chip select signals can be programmed to run a cycle from 240ns to 1120ns. This is done in 40ns increments. In addition, both active and inactive (restore) portions of the cycle can be varied. This should be suitable to accommodate most devices without having to design external READY circuitry. However, an external READY signal is also available to further extend cycles or to run variable length cycles based on other real-time changing conditions that might exist on the AIB. There is a separate CSD register for each chip select signal (total of five).

Register Format

```
(80960 ADDRESS = 1FF8B000h) CSD0 r/w
(80960 ADDRESS = 1FF8B004h) CSD1 r/w
(80960 ADDRESS = 1FF8B008h) CSD2 r/w
(80960 ADDRESS = 1FF8B00Ch) CSD3 r/w
(80960 ADDRESS = 1FF8B010h) CSD4 r/w
```

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AC3	AC2	AC1	AC0	IC2	IC1	IC0	X	X	RG2	RG1	RG0	X	DS1	DS0	ENA

31	23	22	21	20	19	18	17	16
Reserved		A18	A17	A16	A15	A14	A13	A12

CSD Register Bit Descriptions

- Bits 31-23. Reserved.
- Bits 22-16. Chip select base address. These seven bits determine the base address of the chip select address range. This document will only reference the primary addresses for the Vero AIB devices selected by CSD0-4. The primary and secondary addresses maps for the Vero AIB devices are:

  ```
  Primary Address Map:     1FF0000 h to 1FF7FFFF h
  Secondary Address Map:   1FE0000 h to 1FE7FFFF h
  ```

  Starting addresses can be on any 4KB boundary of the AIB address space. A value of '0000000'b chooses the starting address for a chip select as '1FF00000'h in the 80960 address space. A value of '1111111'b chooses the starting address for a chip select as '1FF7F000'b in the 80960 address space. These bits can be thought of as replacing bits 18-12 of the physical 80960 address.
- Bits 15-12. Active cycle time. These four bits determine the active cycle time for a particular chip select. Active time is defined as the time when the -A_RD or -A_WR signal is low (0 volts).

AC3	AC2	AC1	AC0	CS 'low' (active) pulse width
0	0	0	0	200ns
0	0	0	1	240ns
0	0	1	0	280ns
0	0	1	1	320ns
0	1	0	0	360ns
0	1	0	1	400ns
0	1	1	0	440ns
0	1	1	1	480ns
1	0	0	0	520ns
1	0	0	1	560ns
1	0	1	0	600ns
1	0	1	1	640ns
1	1	0	0	680ns
1	1	0	1	720ns
1	1	1	0	760ns
1	1	1	1	800ns

- Bits 11-9. Inactive cycle time. These two bits determine the inactive cycle time for a particular chip select. Inactive time is defined as the time when the -A_RD or -A_WR signal is high (5 volts).

IC2	IC1	IC0	CS 'high' (inactive) pulse width
0	0	0	80ns
0	0	1	120ns
0	1	0	160ns
0	1	1	200ns
1	0	0	240ns
1	0	1	280ns
1	1	0	320ns
1	1	1	360ns

- Bits 8-7. Reserved.
- Bits 6-4. Chip select address range bits. These bits are binary encoded to provide the range of addresses that are decoded for each chip select signal. The eight binary possibilities give decode ranges of 4K, 8K, 16K, 32K, 64K, 128K, 256K, and 512K addresses. There is a restriction that the base starting address selected in bits 22-16 must be on a boundary that is equal to the decode range selected by bits 6-4. For instance, if a decode range of 32K is selected the base starting address can be any 32K boundary.

RG2	RG1	RG0	Chip Select Decode Range
0	0	0	4K
0	0	1	8K
0	1	0	16K
0	1	1	32K
1	0	0	64K
1	0	1	128K
1	1	0	256K
1	1	1	512K

- Bits 2-1. Device size bits. These bits are encoded with the data bus size of the device in this chip select range. Device size options of 8, 16, and 32 bits are possible.

```
DS1  DS0       Device size
---  ---       -----------
 0    0          8-bit
 0    1         16-bit
 1    0         32-bit
 1    1        Reserved
```

- Bit 0. Chip select enable. This bit when set to '0' disables this chip select and when set to '1' enables the chip select.

Reset Conditions

```
POWER-UP:       UUUU UUUU UUUU UUUU UUUU UUU0 0UUU 0UU0
RESET COMMAND:  0000 0000 0SSS SSSS SSSS SSS0 0SSS 0SSS
```

4.2.2 DMA Acknowledge Pulse Width registers (DAPW0-7)

Description: These registers are used to program the pulse width of the DMA acknowledge (DACK) signal that the Vero chip sends to the DMA requesting device. Different devices require this signal to be a certain duration depending on how fast the device responds. These registers allow the DACK cycle to be optimized relative to the specific DMA device. The timings given below are nominal and can vary by '+' or '-' 3ns.

Register Format

```
(80960 ADDRESS = 1FF8B030h) r/w     DACK0 pulse width
(80960 ADDRESS = 1FF8B034h) r/w     DACK1 pulse width
(80960 ADDRESS = 1FF8B038h) r/w     DACK2 pulse width
(80960 ADDRESS = 1FF8B03Ch) r/w     DACK3 pulse width
(80960 ADDRESS = 1FF8B040h) r/w     DACK4 pulse width
(80960 ADDRESS = 1FF8B044h) r/w     DACK5 pulse width
(80960 ADDRESS = 1FF8B048h) r/w     DACK6 pulse width
(80960 ADDRESS = 1FF8B04Ch) r/w     DACK7 pulse width
```

31 8	7	6	5	4	3	2	1	0
Reserved	LP3	LP2	LP1	LP0	X	HP2	HP1	HP0

Bit Descriptions

- Bits 31-8. Reserved.
- Bits 7-4. DACK 'low' pulse width. These 4 bits are binary encoded to provide 10 options for setting the duration that the DACK signal will be 'low' (0 volts).

LP3	LP2	LP1	LP0	DACK 'low' (active) pulse width
0	0	0	0	80ns
0	0	0	1	120ns
0	0	1	0	160ns
0	0	1	1	200ns
0	1	0	0	240ns
0	1	0	1	280ns
0	1	1	0	320ns
0	1	1	1	360ns
1	0	0	0	400ns
1	0	0	1	440ns

- Bit 3. Reserved. Always program this bit to a '0'.
- Bits 2-0. DACK 'high' pulse width. These 3 bits are binary encoded to provide 8 options for setting the duration that the DACK signal will be 'high' (+5 volts).

HP2	HP1	HP0	DACK 'high' (inactive) pulse width
0	0	0	80ns
0	0	1	120ns
0	1	0	160ns
0	1	1	200ns
1	0	0	240ns
1	0	1	280ns
1	1	0	320ns
1	1	1	360ns

Reset Conditions

```
POWER-UP:       0000 0000 0000 0000 0000 0000 UUUU 0UUU
RESET COMMAND:  0000 0000 0000 0000 0000 0000 SSSS 0SSS
```

4.2.3 AIB Command/Status register (ACSR)

Description: The ACSR allows the AIB reset signal to be toggled under program control of the 80960. It should be written to a '1' to activate the reset signal and written back to '0' to deactivate it. The register also allows data parity checking to the AIB to enabled and disabled.**:** Reading this register will clear interrupt vector number 240, which is an AIB Bus read (from local bus) parity error condition has occurred (see Table 12 on page 44). This is a read/write register.

Register Format

```
(80960 Address = 1FF8C000h) r/w
```

31 7	6	5	4	3	2	1	0
Reserved	GEN	CHK	PAR	RSV	RSV	CLK	RST

Bit Descriptions

- Bits 31-7. Reserved.
- Bit 6. Parity Generate polarity. '0' = odd parity. '1' = even parity.
- Bit 5. Parity Check polarity. '0' = odd parity. '1' = even parity.
- Bit 4. Address/Data Parity enable. This bit should be set to enable address/data parity checking of the AIB data bus.
- Bit 3-2. Reserved.
- Bit 1. Clock enable. This bit is reset to enable the AIB_CLK signal (25MHz oscillator) to be presented to the AIB. It is set to disable the clock from propagating to the AIB.
- Bit 0. Reset. This bit is written to a '1' to activate the AIB 'reset' signal and is written to a '0' to deactivate the signal.

Reset Conditions

```
POWER-UP:       0000 0000 0000 0000 0000 0000 0000 0000
RESET COMMAND:  0000 0000 0000 0000 0000 0000 0SSS 00SS
```

5.0 Vero Miscellaneous Registers

5.1.1 Presence Detect Register (PDR)

Description: The PDR can be used to read static logic levels attached to Vero. These could be memory presence detect bits, cable ID's, etc. These bits are non-inverted when read.

Register Format

```
(80960 Address = 1FF8D000h) ro
```

31 13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	PDC	PDB	PDA	PD9	PD8	RSV	PD6	PD5	PD4	PD3	PD2	PD1	PD0

Bit Descriptions

- Bits 31-13. Reserved.
- Bit 12. AIB presence detect bit. (-A_DETECT signal)
- Bits 11-8. Presence detect bits. (PD(11-8) signals)
- Bit 7. Reserved.
- Bits 6-0. Presence detect bits. (PD(6-0) signals)

Reset Conditions

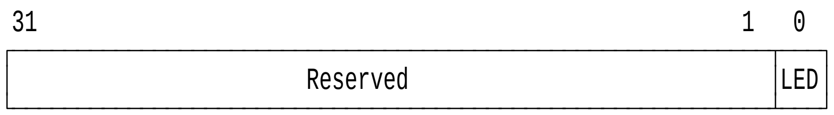
```
POWER-UP:       0000 0000 0000 0000 000U UUUU UUUU UUUU
RESET COMMAND:  0000 0000 0000 0000 000S SSSS SSSS SSSS
```

5.1.2 LED Enable Register (LER)

Description: The LER is used to allow an external LED to be enabled and disabled with a write operation by the 80960. It could also be used as a general purpose output.

Register Format

```
(80960 Address = 1FF8D004h) r/w
```



Bit Descriptions

- Bits 31-1. Reserved.
- Bit 0. LED enable. When set to '0' the LED_EN signal will be low. When set to '1' the LED_EN signal will be high. This is a 4 ma driver.

Reset Conditions

```
POWER-UP:       0000 0000 0000 0000 0000 0000 0000 0000
RESET COMMAND:  0000 0000 0000 0000 0000 0000 0000 000S
```

Appendix A. Vero Pin Name/Number Cross Reference

Table 13 (Page 1 of 5). Vero Pin Name/Number Cross Reference

Module Pin	Pin Name	Pin Type	Description
001	PD1	INPUT	PRESENCE DETECT 1
002	DI	INPUT	DRIVER INHIBIT
003	M_INT0	INPUT	MIAMI INT0
004	TST_CLKA	INPUT	LSSD A CLOCK
005	PD2	INPUT	PRESENCE DETECT 2
006	TST_CLKB	INPUT	LSSD B CLOCK
007	TST_CLKC	INPUT	LSSD C CLOCK
008	RAMTSTEN	INPUT	RAM TEST ENABLE
009	PD3	INPUT	PRESENCE DETECT 3
010	VDD	PWR	+5 VOLTS
011	ICT_EN	INPUT	IN CIRCUIT TEST ENABLE
012	TST_CLKR	INPUT	LSSD RAM CLOCK
013	PD4	INPUT	PRESENCE DETECT 4
014	PD11	INPUT	PRESENCE DETECT 11
015	VSS	GND	+0 VOLTS
016	PD10	INPUT	PRESENCE DETECT 10
017	PD5	INPUT	PRESENCE DETECT 5
018	PD9	INPUT	PRESENCE DETECT 9
019	PD8	INPUT	PRESENCE DETECT 8
020	A_DETECT	INPUT	AIB PRESENCE DETECT
021	PRI/SEC	INPUT	PRIMARY/SECONDARY ADDR
022	A_ERROR	CIO-4MA	ERROR IN
023	A_INTACK1	CIO-4MA	AIB INTERRUPT ACK 1
024	VSS	GND	+0 VOLTS
025	PD6	INPUT	PRESENCE DETECT 6
026	A_INTACK0	CIO-4MA	AIB INTERRUPT ACK 0
027	A_INT3	CIO-4MA	AIB INTERRUPT 3
028	A_INT2	CIO-4MA	AIB INTERRUPT 2
029	A_INT1	INPUT	AIB INTERRUPT 1
030	A_INT0	INPUT	AIB INTERRUPT 0
031	A_SPARE	CIO-4MA	RSVD
032	A_RDY	CIO-4MA	AIB READY
033	LED_EN	3ST-8MA	LED ENABLE
034	AAD0	CIO-4MA	AIB AD00
035	AAD1	CIO-4MA	AIB AD01
036	AAD2	CIO-4MA	AIB AD02
037	A_WRAPEN	CIO-4MA	RSVD
038	VSS	GND	+0 VOLTS
039	AAD3	CIO-4MA	AIB AD03
040	AAD4	CIO-4MA	AIB AD04
041	AAD5	CIO-4MA	AIB AD05
042	AAD6	CIO-4MA	AIB AD06
043	VDD	PWR	+5 VOLTS
044	AAD7	CIO-4MA	AIB AD07
045	VSS	GND	+0 VOLTS
046	AAD8	CIO-4MA	AIB AD08
047	AAD9	CIO-4MA	AIB AD09
048	AAD10	CIO-4MA	AIB AD10

Table 13 (Page 2 of 5). Vero Pin Name/Number Cross Reference

Module Pin	Pin Name	Pin Type	Description
049	A_RESET	CIO-4MA	AIB RESET
050	AAD11	CIO-4MA	AIB AD11
051	AAD12	CIO-4MA	AIB AD12
052	WDOG	CIO-4MA	WATCH DOG
053	VSS	GND	+0 VOLTS
054	A_DEN	CIO-4MA	AIB DATA ENABLE
055	AAD13	CIO-4MA	AIB AD13
056	AAD14	CIO-4MA	AIB AD14
057	AAD15	CIO-4MA	AIB AD15
058	AAD16	CIO-4MA	AIB AD16
059	AAD17	CIO-4MA	AIB AD17
060	VSS	GND	+0 VOLTS
061	VDD	PWR	+5 VOLTS
062	AAD18	CIO-4MA	AIB AD18
063	AAD19	CIO-4MA	AIB AD19
064	AAD20	CIO-4MA	AIB AD20
065	VSS	GND	+0 VOLTS
066	A_CLK	CIO-4MA	AIB CLOCK
067	VSS	GND	+0 VOLTS
068	AAD21	CIO-4MA	AIB AD21
069	AAD22	CIO-4MA	AIB AD22
070	VSS	GND	+0 VOLTS
071	A_ALE	CIO-4MA	AIB ALE
072	VSS	GND	+0 VOLTS
073	AAD23	CIO-4MA	AIB AD23
074	AAD24	CIO-4MA	AIB AD24
075	AAD25	CIO-4MA	AIB AD25
076	AAD26	CIO-4MA	AIB AD26
077	AAD27	CIO-4MA	AIB AD27
078	VSS	GND	+0 VOLTS
079	AAD28	CIO-4MA	AIB AD28
080	AAD29	CIO-4MA	AIB AD29
081	AAD30	CIO-4MA	AIB AD30
082	AAD31	CIO-4MA	AIB AD31
083	A_PAR0	CIO-4MA	AIB PARITY 0
084	VSS	GND	+0 VOLTS
085	A_PAR1	CIO-4MA	AIB PARITY 1
086	A_PAR2	CIO-4MA	AIB PARITY 2
087	A_PAR3	CIO-4MA	AIB PARITY 3
088	A_CS0	CIO-4MA	AIB CHIP SELECT 0
089	A_CS1	CIO-4MA	AIB CHIP SELECT 1
090	A_CS2	CIO-4MA	AIB CHIP SELECT 2
091	A_CS3	CIO-4MA	AIB CHIP SELECT 3
092	A_CS4	CIO-4MA	AIB CHIP SELECT 4
093	A_BE0	CIO-4MA	AIB BYTE ENABLE 0
094	A_BE1	CIO-4MA	AIB BYTE ENABLE 1
095	A_BE2	CIO-4MA	AIB BYTE ENABLE 2
096	VSS	GND	+0 VOLTS
097	VDD	PWR	+5 VOLTS
098	A_BE3	CIO-4MA	AIB BYTE ENABLE 3
099	D_ACK0	CIO-4MA	DMA ACKNOWLEDGE 0
100	D_ACK1	CIO-4MA	DMA ACKNOWLEDGE 1

Table 13 (Page 3 of 5). Vero Pin Name/Number Cross Reference

Module Pin	Pin Name	Pin Type	Description
101	D_ACK2	CIO-4MA	DMA ACKNOWLEDGE 2
102	D_ACK3	CIO-4MA	DMA ACKNOWLEDGE 3
103	D_ACK4	CIO-4MA	DMA ACKNOWLEDGE 4
104	D_ACK5	CIO-4MA	DMA ACKNOWLEDGE 5
105	D_ACK6	CIO-4MA	DMA ACKNOWLEDGE 6
106	D_ACK7	CIO-4MA	DMA ACKNOWLEDGE 7
107	A_RD	CIO-4MA	AIB READ
108	VSS	GND	+0 VOLTS
109	A_WR	CIO-4MA	AIB WRITE
110	A_EOP0	CIO-4MA	AIB EOP 0
111	A_EOP1	CIO-4MA	AIB EOP 1
112	A_EOP2	CIO-4MA	AIB EOP 2
113	A_EOP3	CIO-4MA	AIB EOP 3
114	VDD	PWR	+5 VOLTS
115	A_EOP4	CIO-4MA	AIB EOP 4
116	A_EOP5	CIO-4MA	AIB EOP 5
117	A_EOP6	CIO-4MA	AIB EOP 6
118	A_EOP7	CIO-4MA	AIB EOP 7
119	VSS	GND	+0 VOLTS
120	D_REQ0	INPUT	DMA REQUEST 0
121	D_REQ1	INPUT	DMA REQUEST 1
122	D_REQ2	INPUT	DMA REQUEST 2
123	D_REQ3	INPUT	DMA REQUEST 3
124	D_REQ4	INPUT	DMA REQUEST 4
125	L_ADS	CIO-4MA	LOCAL ADDRESS STROBE
126	VSS	GND	+0 VOLTS
127	L_OSC	INPUT	25 MHZ CLOCK INPUT
128	VSS	GND	+0 VOLTS
129	D_REQ5	INPUT	DMA REQUEST 5
130	D_REQ6	INPUT	DMA REQUEST 6
131	D_REQ7	INPUT	DMA REQUEST 7
132	L_BE3	CIO-4MA	LOCAL BYTE ENABLE 3
133	L_BE2	CIO-4MA	LOCAL BYTE ENABLE 2
134	L_BE1	CIO-4MA	LOCAL BYTE ENABLE 1
135	L_BE0	CIO-4MA	LOCAL BYTE ENABLE 0
136	VSS	GND	+0 VOLTS
137	L_ADP3	CIO-4MA	LOCAL PARITY 3
138	L_ADP2	CIO-4MA	LOCAL PARITY 2
139	L_ADP1	CIO-4MA	LOCAL PARITY 1
140	L_ADP0	CIO-4MA	LOCAL PARITY 0
141	LAD31	CIO-4MA	LOCAL AD31
142	VSS	GND	+0 VOLTS
143	LAD30	CIO-4MA	LOCAL AD30
144	LAD29	CIO-4MA	LOCAL AD29
145	LAD28	CIO-4MA	LOCAL AD28
146	LAD27	CIO-4MA	LOCAL AD27
147	VDD	PWR	+5 VOLTS
148	LAD26	CIO-4MA	LOCAL AD26
149	VSS	GND	+0 VOLTS
150	LAD25	CIO-4MA	LOCAL AD25
151	LAD24	CIO-4MA	LOCAL AD24
152	LAD23	CIO-4MA	LOCAL AD23

Table 13 (Page 4 of 5). Vero Pin Name/Number Cross Reference

Module Pin	Pin Name	Pin Type	Description
153	LAD22	CIO-4MA	LOCAL AD22
154	LAD21	CIO-4MA	LOCAL AD21
155	LAD20	CIO-4MA	LOCAL AD20
156	VSS	GND	+0 VOLTS
157	LAD19	CIO-4MA	LOCAL AD19
158	LAD18	CIO-4MA	LOCAL AD18
159	LAD17	CIO-4MA	LOCAL AD17
160	LAD16	CIO-4MA	LOCAL AD16
161	LAD15	CIO-4MA	LOCAL AD15
162	LAD14	CIO-4MA	LOCAL AD14
163	LAD13	CIO-4MA	LOCAL AD13
164	VSS	GND	+0 VOLTS
165	VDD	PWR	+5 VOLTS
166	LAD12	CIO-4MA	LOCAL AD12
167	LAD11	CIO-4MA	LOCAL AD11
168	LAD10	CIO-4MA	LOCAL AD10
169	LAD09	CIO-4MA	LOCAL AD09
170	LAD08	CIO-4MA	LOCAL AD08
171	VSS	GND	+0 VOLTS
172	LAD07	CIO-4MA	LOCAL AD07
173	PXINT7	3ST-4MA	PROCESSOR INT 7
174	LAD06	CIO-4MA	LOCAL AD06
175	LAD05	CIO-4MA	LOCAL AD05
176	LAD04	CIO-4MA	LOCAL AD04
177	PXINT6	3ST-4MA	PROCESSOR INT 6
178	LAD03	CIO-4MA	LOCAL AD03
179	VSS	GND	+0 VOLTS
180	LAD02	CIO-4MA	LOCAL AD02
181	PXINT5	3ST-4MA	PROCESSOR INT 5
182	LAD01	CIO-4MA	LOCAL AD01
183	LAD00	CIO-4MA	LOCAL AD00
184	PXINT4	3ST-4MA	PROCESSOR INT 4
185	L_W/R	CIO-4MA	LOCAL WRITE/-READ
186	L_RDY	CIO-4MA	LOCAL READY
187	L_REQ	CIO-4MA	LOCAL BUS REQUEST
188	PXINT3	3ST-4MA	PROCESSOR INT 3
189	L_EXCPT	CIO-4MA	LOCAL EXCEPTION
190	L_GRANT	INPUT	LOCAL BUS GRANT
191	L_BLAST	CIO-4MA	LOCAL BURST LAST
192	PXINT2	3ST-4MA	PROCESSOR INT 2
193	RESET	INPUT	RESET INPUT
194	L_INT0	INPUT	LOCAL INTERRUPT 0
195	L_INT1	INPUT	LOCAL INTERRUPT 1
196	PXINT1	3ST-4MA	PROCESSOR INT 1
197	B_INT3	INPUT	BRIGHTON INT 3
198	B_INT2	INPUT	BRIGHTON INT 2
199	B_INT1	INPUT	BRIGHTON INT 1
200	VSS	GND	+0 VOLTS
201	VDD	PWR	+5 VOLTS
202	B_INT0	INPUT	BRIGHTON INT 0
203	M_INT3	INPUT	MIAMI INT 3
204	PXINT0	3ST-4MA	PROCESSOR INT 0

Table 13 (Page 5 of 5). Vero Pin Name/Number Cross Reference

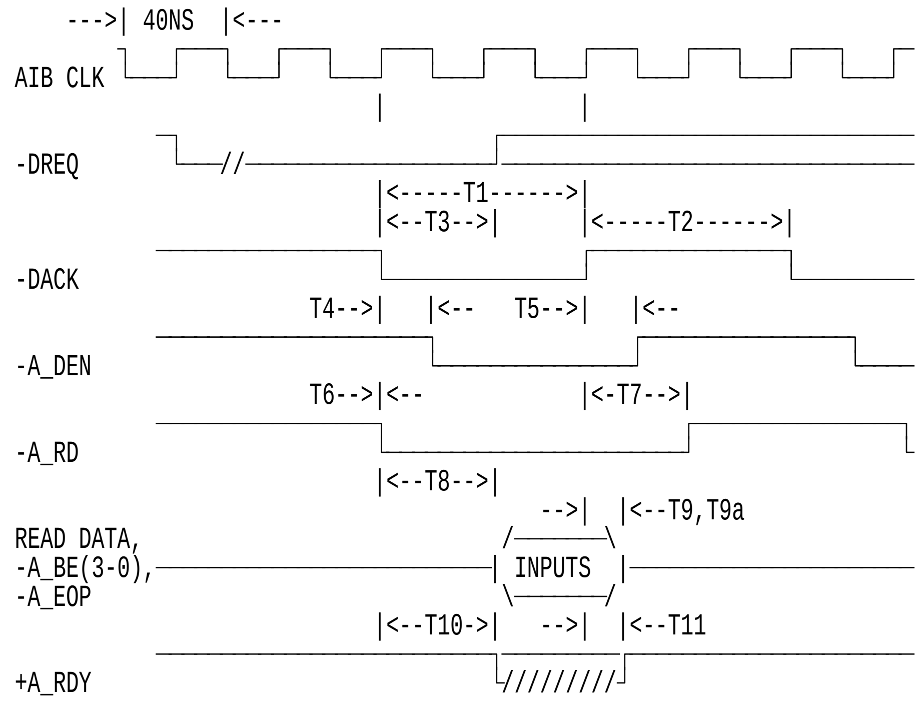
Module Pin	**Pin Name**	**Pin Type**	**Description**
205	M_INT2	INPUT	MIAMI INT 2
206	M_INT1	INPUT	MIAMI INT 1
207	COMP RES	INPUT	COMP. RESISTOR
208	PD0	INPUT	PRESENCE DETECT 0

Appendix B. AIB Bus Timings

The following timing diagrams show AIB Bus timings for the various types of cycles that run on the AIB interface. All timings are specified with 60pf load capacitance.

B.1 DMA Cycle Timings

B.1.1 DMA Read Cycle



		min	max		Notes
T1	: -DACK ACTIVE PULSE WIDTH	70	-	ns	1
T2	: -DACK INACTIVE PULSE WIDTH	77	-	ns	2
T3	: -DACK ACTIVE TO -DREQ INACTIVE	0	45	ns	1,3
T4	: -DACK ACTIVE TO -A_DEN ACTIVE	18	40	ns	
T5	: -DACK INACTIVE TO -A_DEN INACTIVE	20	40	ns	
T6	: -DACK ACTIVE TO -A_RD ACTIVE	0	10	ns	
T7	: -DACK INACTIVE TO -A_RD INACTIVE	35	45	ns	
T8	: -DACK ACTIVE TO -A_BE(3-0) VALID	-	45	ns	1,4
T8a	: -DACK ACTIVE TO READ DATA VALID	-	50	ns	1,4
T8b	: -DACK ACTIVE TO -A_EOP(7-0) VALID	-	50	ns	1,4
T9	: INPUT HOLD TIME FROM -DACK INACTIVE	0	-	ns	
T9a	: INPUTS 3-STATE FROM -DACK INACTIVE	-	70	ns	2,5
T10	: -DACK ACTIVE TO +A_RDY INACTIVE	-	40	ns	1,4
T11	: P_CLK HIGH TO +A_RDY ACTIVE	0	15	ns	4

Note 1 - This assumes programming of the DAPW register to 80ns active width. Add 40ns to this number for each corresponding 40ns pulse width increase.

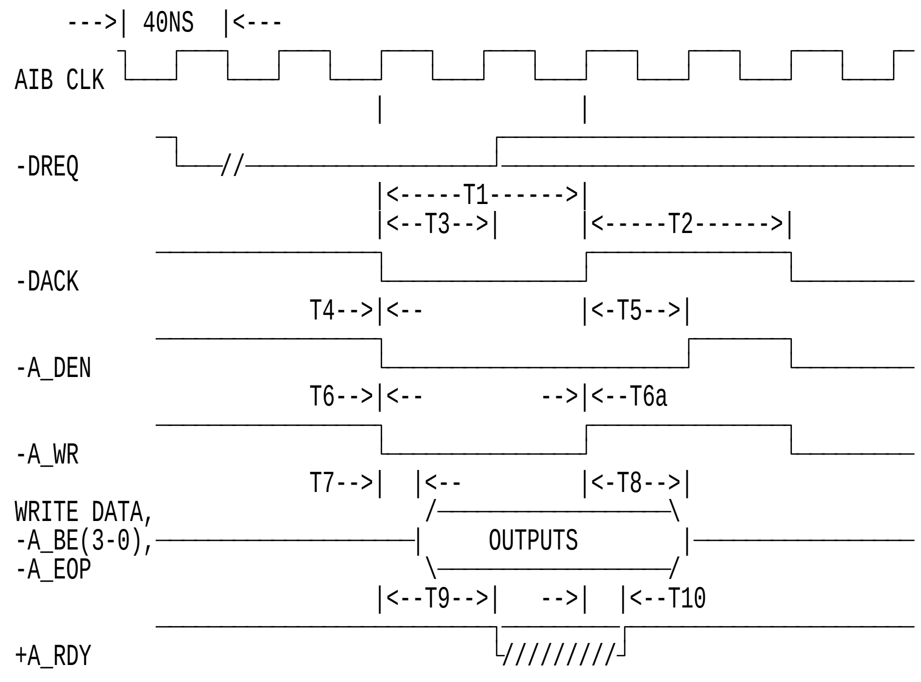
Note 2 - This assumes programming of the DAPW register to 80ns inactive width. Add 40ns to this number for each corresponding 40ns pulse width increase.

Note 3 - This timing is to assure that a -DREQ driven inactive by a device is sampled inactive. A -DREQ can be left active and will therefore be the next cycle run to the AIB.

Note 4 - The +A_RDY signal works in conjunction with the programming of the DAPW register. +A_RDY can be used to extend a cycle time beyond that which can be programmed into the DAPW. +A_RDY is first sampled after the DAPW programmed active time has elapsed. +A_RDY is sampled every 40ns thereafter and will extend the cycle for as long as it remains inactive. +A_RDY, if used, should be driven inactive combinatorially within 40ns of the -DACK active signal. It should be driven back active synchronously with AIB_CLK within 15ns.

Note 5 - Applies to Read data, BE(3-0), and EOP from the device driving these signals.

B.1.2 DMA Write Cycle



		min	max		Notes
T1	: -DACK ACTIVE PULSE WIDTH	70	-	ns	1
T2	: -DACK INACTIVE PULSE WIDTH	77	-	ns	2
T3	: -DACK ACTIVE TO -DREQ INACTIVE	0	45	ns	1,3
T4	: -DACK ACTIVE TO -A_DEN ACTIVE	0	10	ns	
T5	: -DACK INACTIVE TO -A_DEN INACTIVE	35	45	ns	
T6	: -DACK ACTIVE TO -A_WR ACTIVE	0	10	ns	
T6a	: -DACK INACTIVE TO -A_WR INACTIVE	-5	5	ns	
T7	: -DACK ACTIVE TO OUTPUTS VALID	0	28	ns	
T8	: OUTPUTS HOLD FROM -DACK INACTIVE	35	45	ns	
T9	: -DACK ACTIVE TO +A_RDY INACTIVE	-	40	ns	1,4
T10	: P_CLK HIGH TO +A_RDY ACTIVE	0	15	ns	4

Note 1 - This assumes programming of the DAPW register to 80ns active width. Add 40ns to this number for each corresponding 40ns pulse width increase.

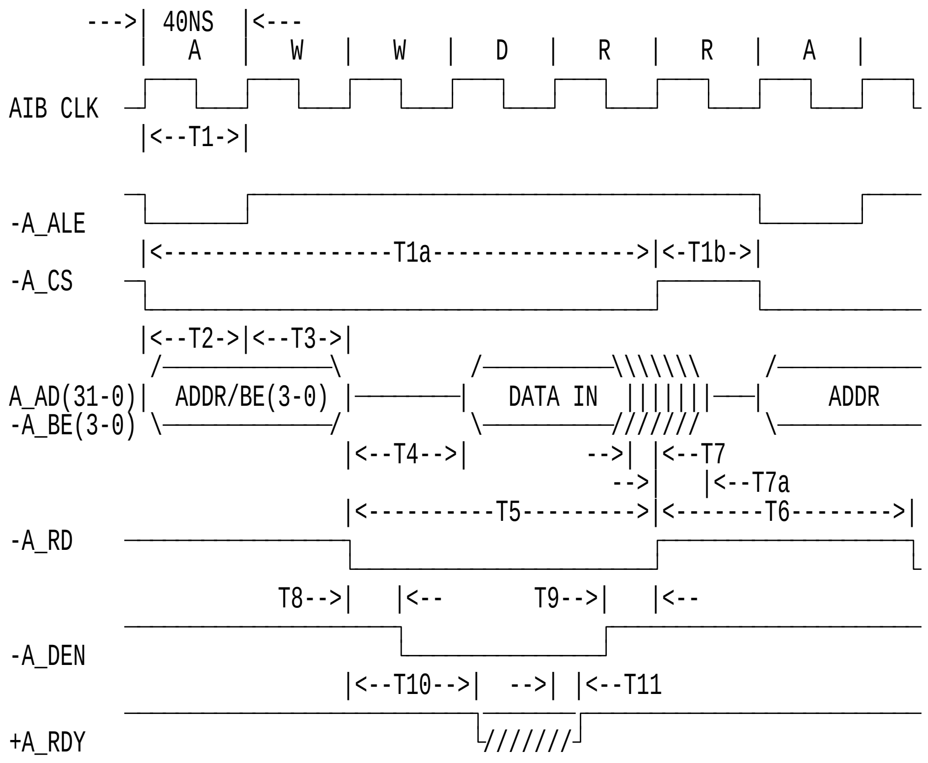
Note 2 - This assumes programming of the DAPW register to 80ns inactive width. Add 40ns to this number for each corresponding 40ns pulse width increase.

Note 3 - This timing is to assure that a -DREQ driven inactive by a device is sampled inactive. A -DREQ can be left active and will therefore be the next cycle run to the AIB.

Note 4 - The +A_RDY signal works in conjunction with the programming of the DAPW register. +A_RDY can be used to extend a cycle time beyond that which can be programmed into the DAPW. +A_RDY is first sampled after the DAPW programmed active time has elapsed. +A_RDY is sampled every 40ns thereafter and will extend the cycle for as long as it remains inactive. +A_RDY, if used, should be driven inactive combinatorially within 40ns of the -DACK active signal. It should be driven back active synchronously with AIB_CLK within 15ns.

B.2 AIB Slave Cycle Timings

B.2.1 AIB Slave Read Cycle



		min	max		Notes
T1	: -A_ALE ACTIVE PULSE WIDTH	35	45	ns	
T1a	: -A_CS ACTIVE PULSE WIDTH	190	-	ns	1
T1b	: -A_CS INACTIVE PULSE WIDTH	37	-	ns	2
T2	: ADDR/BE(3-0)/-A_CS SETUP TO -A_ALE INACTIVE	16	-	ns	
T3	: ADDRESS HOLD FROM -A_ALE INACTIVE	20	-	ns	0
T4	: -A_RD ACTIVE TO DATA VALID	-	40	ns	1
T5	: -A_RD ACTIVE PULSE WIDTH	110	-	ns	1,4
T6	: -A_RD INACTIVE PULSE WIDTH	117	-	ns	2,4
T7	: DATA HOLD FROM -A_RD INACTIVE	0	-	ns	
T7a	: DATA BUS 3-STATE FROM -A_RD INACTIVE	-	30	ns	2
T8	: -A_RD ACTIVE TO -A_DEN ACTIVE	18	37	ns	
T9	: -A_DEN INACTIVE TO -A_RD INACTIVE	8	22	ns	
T10	: -A_RD ACTIVE TO +A_RDY INACTIVE	0	18	ns	1,3
T11	: P_CLK HIGH TO +A_RDY ACTIVE	0	15	ns	3

Note 0 - -A_BE(3-0) are driven active with ADDRESS and are held valid until -A_DEN is driven inactive.

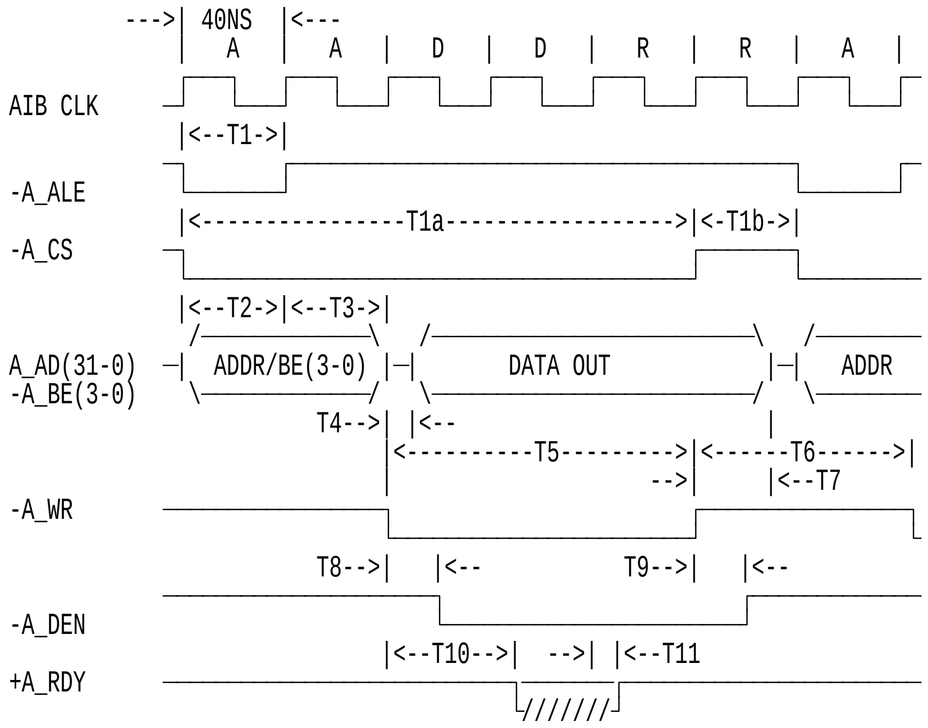
Note 1 - This assumes programming of the CSD register to 120ns active width. Add 40ns to this number for each corresponding 40ns pulse width increase.

Note 2 - This assumes programming of the CSD register to 120ns inactive width. Add 40ns to this number for each corresponding 40ns pulse width increase.

Note 3 - The +A_RDY signal works in conjunction with the programming of the CSD register. +A_RDY can be used to extend a cycle time beyond that which can be programmed into the CSD. +A_RDY is sampled internally with the first rising edge of AIB_CLK after -A_RD goes active (if CSD active time is 120ns). +A_RDY is sampled every 40ns thereafter and will extend the cycle for as long as it remains inactive. +A_RDY, if used, should be driven inactive combinatorially within 18ns of the -A_RD active signal. It should be driven back active synchronously with AIB_CLK within 15ns.

Note 4 - The minimum cycle time for a read cycle to the AIB is the sum of T5 and T6, or 240ns. The cycle time can be increased in 40ns increments by programming the CSD register, or through the use of +A_RDY.

B.2.2 AIB Slave Write Cycle



		min	max		Notes
T1	-A_ALE ACTIVE PULSE WIDTH	35	45	ns	
T1a	-A_CS ACTIVE PULSE WIDTH	190	-	ns	1
T1b	-A_CS INACTIVE PULSE WIDTH	37	-	ns	2
T2	ADDR/BE(3-0)/-A_CS SETUP TO -A_ALE INACTIV	16	-	ns	
T3	ADDRESS HOLD FROM -A_ALE INACTIVE	20	-	ns	0
T4	-A_WR ACTIVE TO DATA VALID	0	25	ns	
T5	-A_WR ACTIVE PULSE WIDTH	110	-	ns	1,4
T6	-A_WR INACTIVE PULSE WIDTH	117	-	ns	2,4
T7	DATA HOLD FROM -A_WR INACTIVE	35	-	ns	
T8	-A_WR ACTIVE TO -A_DEN ACTIVE	19	37	ns	
T9	-A_WR INACTIVE TO -A_DEN INACTIVE	19	37	ns	
T10	-A_WR ACTIVE TO +A_RDY INACTIVE	0	18	ns	1,3
T11	P_CLK HIGH TO +A_RDY ACTIVE	0	15	ns	3

Note 0 - -A_BE(3-0) are driven active with ADDRESS and are held valid until -A_DEN is driven inactive.

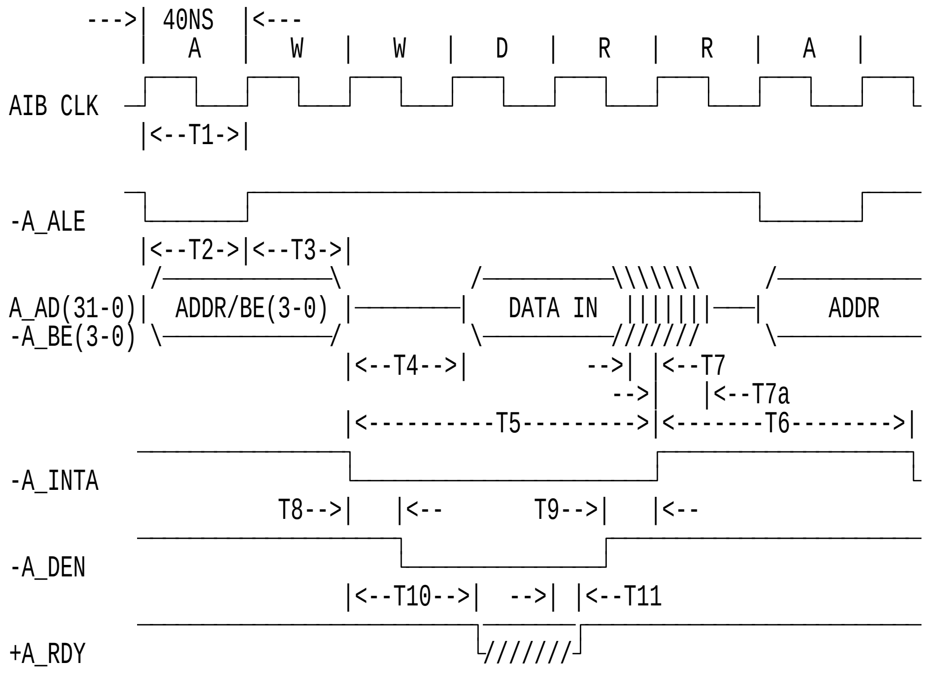
Note 1 - This assumes programming of the CSD register to 120ns active width. Add 40ns to this number for each corresponding 40ns pulse width increase.

Note 2 - This assumes programming of the CSD register to 120ns inactive width. Add 40ns to this number for each corresponding 40ns pulse width increase.

Note 3 - The +A_RDY signal works in conjunction with the programming of the CSD register. +A_RDY can be used to extend a cycle time beyond that which can be programmed into the CSD. +A_RDY is sampled internally with the first rising edge of AIB_CLK after -A_RD goes active (if CSD active time is 120ns). +A_RDY is sampled every 30ns thereafter and will extend the cycle for as long as it remains inactive. +A_RDY, if used, should be driven inactive combinatorially within 18ns of the -A_RD active signal. It should be driven back active synchronously with AIB_CLK within 15ns.

Note 4 - The minimum cycle time for a read cycle to the AIB is the sum of T5 and T6, or 240ns. The cycle time can be increased in 40ns increments by programming the CSD register, or through the use of +A_RDY.

B.2.3 AIB Interrupt Acknowledge Cycle



		min	max		Notes
T1	: -A_ALE ACTIVE PULSE WIDTH	35	45	ns	
T2	: ADDR/BE(3-0) SETUP TO -A_ALE INACTIVE	16	-	ns	
T3	: ADDRESS HOLD FROM -A_ALE INACTIVE	20	-	ns	0
T4	: -A_INTA ACTIVE TO DATA VALID	-	40	ns	1
T5	: -A_INTA ACTIVE PULSE WIDTH	110	-	ns	1,4
T6	: -A_INTA INACTIVE PULSE WIDTH	117	-	ns	2,4
T7	: DATA HOLD FROM -A_INTA INACTIVE	0	-	ns	
T7a	: DATA BUS 3-STATE FROM -A_INTA INACTIVE	-	30	ns	2
T8	: -A_INTA ACTIVE TO -A_DEN ACTIVE	20	37	ns	
T9	: -A_DEN INACTIVE TO -A_INTA INACTIVE	8	22	ns	
T10	: -A_INTA ACTIVE TO +A_RDY INACTIVE	0	18	ns	1,3
T11	: P_CLK HIGH TO +A_RDY ACTIVE	0	15	ns	3

Note 0 - -A_BE(0) is driven active with ADDRESS during an INTA cycle and is held valid until -A_DEN is driven inactive. Also, the address is driven to '7FFFC'h during the address phase of the INTA cycle.

Note 1 - This assumes programming of the IIR register to 120ns active width. Add 80ns to this number for each corresponding 80ns pulse width increase.

Note 2 - This assumes programming of the IIR register to 120ns inactive width. Add 80ns to this number for each corresponding 80ns pulse width increase.

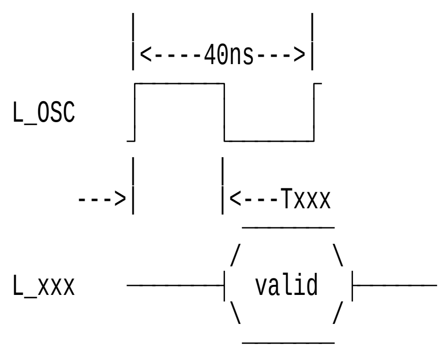
Note 3 - The +A_RDY signal works in conjunction with the programming of the IIR register. +A_RDY can be used to extend a cycle time beyond that which can be programmed into the IIR. +A_RDY is sampled internally with the first rising edge of AIB_CLK after -A_INTA goes active (if IIR active time is 120ns). +A_RDY is sampled every 40ns thereafter and will extend the cycle for as long as it remains inactive. +A_RDY, if used, should be driven inactive combinatorially within 18ns of the -A_INTA active signal. It should be driven back active synchronously with AIB_CLK within 15ns.

Note 4 - The minimum cycle time for an INTA cycle to the AIB is the sum of T5 and T6, or 240ns. The cycle time can be increased in 80ns increments by programming the IIR register, or through the use of +A_RDY.

Appendix C. Local Bus Timings

The Vero module, as a Local Bus master, conforms to the Commom Front End (CFE) bus specification from IBM Austin. When running at 25MHz, the Vero master conforms to the CFE spec. for signal loading up to 80pf. As a Local Bus slave, Vero conforms to all CFE bus timings except for ADS setup to L_OSC active. The CFE specifies a 7ns minimum setup time. For proper operation, ADS must have 10ns of setup time to L_OSC active.

Vero Local Bus Master



Txxx : Clock high to L_xxx active/inactive.

	MIN (ns) 40pf load	MAX (ns) 80pf load	
Trdy	6	25	
Tblst	6	25	
Tads	6	26	
Taddr	6	26	(includes parity)
Treq	6	27	
Tbe	6	29	
Twr	6	29	
Texcpt	6	29	
Tdata	6	32	(includes parity)

Note - L_REQ specified with 15pf MIN load and 25pf MAX load.
MAX timings can be derated .2ns/pf down to 50pf.

Appendix D. In-Circuit Test I/O Mapping

The Vero chip supports I/O mapping on an in-circuit-test fixture. This allows the chip to be stuck and open fault tested after it has been surface mounted on to the printed circuit board. By applying a known fixed test vector to chip input pins, an output signal can be made to toggle from one logic state to another. This can be observed on the card to determine whether the chip has been successfully mounted on the card. 171 of 174 signals i/o's can be verified with this test. Three signal i/o, the 33 power and ground pins, and the compensation resistor can not be verified this way.

ICT mode is entered when the three pins below are *lineheld*. During the *linehold* condition, application of the input vectors shown, will yield a known response on the corresponding output vector pin.

```
LINEHOLD - (002)=1;
           (011)=0;
           (193)=1;
```

ICT Vector #1

```
INVECT1(0..8) := (190)||(008)||(208)||(001)||(005)||
                 (009)||(013)||(017)||(004);
OUTVECT1      := (204);

INVECT1(0..8) := B'111111111' : OUTVECT1=0;
INVECT1(0..8) := B'011111111' : OUTVECT1=1;
INVECT1(0..8) := B'001111111' : OUTVECT1=0;
INVECT1(0..8) := B'000111111' : OUTVECT1=1;
INVECT1(0..8) := B'000011111' : OUTVECT1=0;
INVECT1(0..8) := B'000001111' : OUTVECT1=1;
INVECT1(0..8) := B'000000111' : OUTVECT1=0;
INVECT1(0..8) := B'000000011' : OUTVECT1=1;
INVECT1(0..8) := B'000000001' : OUTVECT1=0;
INVECT1(0..8) := B'000000000' : OUTVECT1=1;
```

ICT Vector #2

```
INVECT2(0..9) := (194)||(021)||(019)||(018)||(016)||
                 (014)||(195)||(202)||(199)||(006);
OUTVECT2      := (196);

INVECT2(0..9) := B'1111111111' : OUTVECT2=1;
INVECT2(0..9) := B'0111111111' : OUTVECT2=0;
INVECT2(0..9) := B'0011111111' : OUTVECT2=1;
INVECT2(0..9) := B'0001111111' : OUTVECT2=0;
INVECT2(0..9) := B'0000111111' : OUTVECT2=1;
INVECT2(0..9) := B'0000011111' : OUTVECT2=0;
INVECT2(0..9) := B'0000001111' : OUTVECT2=1;
INVECT2(0..9) := B'0000000111' : OUTVECT2=0;
INVECT2(0..9) := B'0000000011' : OUTVECT2=1;
INVECT2(0..9) := B'0000000001' : OUTVECT2=0;
INVECT2(0..9) := B'0000000000' : OUTVECT2=1;
```

ICT Vector #3

```
INVECT3(0..9) := (030)||(127)||(198)||(197)||(003)||
                 (206)||(205)||(203)||(120)||(007);
OUTVECT3      := (192);

INVECT3(0..9) := B'1111111111' : OUTVECT3=1;
INVECT3(0..9) := B'0111111111' : OUTVECT3=0;
INVECT3(0..9) := B'0011111111' : OUTVECT3=1;
INVECT3(0..9) := B'0001111111' : OUTVECT3=0;
INVECT3(0..9) := B'0000111111' : OUTVECT3=1;
INVECT3(0..9) := B'0000011111' : OUTVECT3=0;
INVECT3(0..9) := B'0000001111' : OUTVECT3=1;
INVECT3(0..9) := B'0000000111' : OUTVECT3=0;
INVECT3(0..9) := B'0000000011' : OUTVECT3=1;
INVECT3(0..9) := B'0000000001' : OUTVECT3=0;
INVECT3(0..9) := B'0000000000' : OUTVECT3=1;
```

ICT Vector #4

```
INVECT4(0..10) := (122)||(123)||(124)||(129)||(130)||
                  (131)||(029)||(020)||(025)||(121)||
                  (012);
OUTVECT4       := (188);

INVECT4(0..10) := B'11111111111' : OUTVECT4=0;
INVECT4(0..10) := B'01111111111' : OUTVECT4=1;
INVECT4(0..10) := B'00111111111' : OUTVECT4=0;
INVECT4(0..10) := B'00011111111' : OUTVECT4=1;
INVECT4(0..10) := B'00001111111' : OUTVECT4=0;
INVECT4(0..10) := B'00000111111' : OUTVECT4=1;
INVECT4(0..10) := B'00000011111' : OUTVECT4=0;
INVECT4(0..10) := B'00000001111' : OUTVECT4=1;
INVECT4(0..10) := B'00000000111' : OUTVECT4=0;
INVECT4(0..10) := B'00000000011' : OUTVECT4=1;
INVECT4(0..10) := B'00000000001' : OUTVECT4=0;
INVECT4(0..10) := B'00000000000' : OUTVECT4=1;
```

ICT Vector #5

```
INVECT5(0..27) := (135)||(189)||(185)||(134)||(133)||
                  (132)||(183)||(182)||(180)||(178)||
                  (176)||(175)||(174)||(172)||(170)||
                  (169)||(168)||(167)||(166)||(163)||
                  (162)||(161)||(160)||(159)||(158)||
                  (157)||(155)||(154);
OUTVECT5       := (184);

INVECT5(0..27) := B'1111111111111111111111111111' : OUTVECT5=1;
INVECT5(0..27) := B'0111111111111111111111111111' : OUTVECT5=0;
INVECT5(0..27) := B'0011111111111111111111111111' : OUTVECT5=1;
INVECT5(0..27) := B'0001111111111111111111111111' : OUTVECT5=0;
INVECT5(0..27) := B'0000111111111111111111111111' : OUTVECT5=1;
INVECT5(0..27) := B'0000011111111111111111111111' : OUTVECT5=0;
INVECT5(0..27) := B'0000001111111111111111111111' : OUTVECT5=1;
INVECT5(0..27) := B'0000000111111111111111111111' : OUTVECT5=0;
INVECT5(0..27) := B'0000000011111111111111111111' : OUTVECT5=1;
INVECT5(0..27) := B'0000000001111111111111111111' : OUTVECT5=0;
INVECT5(0..27) := B'0000000000111111111111111111' : OUTVECT5=1;
INVECT5(0..27) := B'0000000000011111111111111111' : OUTVECT5=0;
INVECT5(0..27) := B'0000000000001111111111111111' : OUTVECT5=1;
INVECT5(0..27) := B'0000000000000111111111111111' : OUTVECT5=0;
INVECT5(0..27) := B'0000000000000011111111111111' : OUTVECT5=1;
INVECT5(0..27) := B'0000000000000001111111111111' : OUTVECT5=0;
INVECT5(0..27) := B'0000000000000000111111111111' : OUTVECT5=1;
INVECT5(0..27) := B'0000000000000000011111111111' : OUTVECT5=0;
INVECT5(0..27) := B'0000000000000000001111111111' : OUTVECT5=1;
INVECT5(0..27) := B'0000000000000000000111111111' : OUTVECT5=0;
INVECT5(0..27) := B'0000000000000000000011111111' : OUTVECT5=1;
INVECT5(0..27) := B'0000000000000000000001111111' : OUTVECT5=0;
INVECT5(0..27) := B'0000000000000000000000111111' : OUTVECT5=1;
INVECT5(0..27) := B'0000000000000000000000011111' : OUTVECT5=0;
INVECT5(0..27) := B'0000000000000000000000001111' : OUTVECT5=1;
INVECT5(0..27) := B'0000000000000000000000000111' : OUTVECT5=0;
INVECT5(0..27) := B'0000000000000000000000000011' : OUTVECT5=1;
INVECT5(0..27) := B'0000000000000000000000000001' : OUTVECT5=0;
INVECT5(0..27) := B'0000000000000000000000000000' : OUTVECT5=1;
```

ICT Vector #6

```
INVECT6(0..29) := (150)||(148)||(146)||(145)||(144)||
                  (143)||(141)||(140)||(139)||(138)||
                  (137)||(110)||(111)||(112)||(113)||
                  (115)||(116)||(117)||(118)||(032)||
                  (093)||(094)||(095)||(098)||(022)||
                  (034)||(035)||(036)||(039)||(040);
OUTVECT6       := (181);

INVECT6(0..29) := B'111111111111111111111111111111' : OUTVECT6=1;
INVECT6(0..29) := B'011111111111111111111111111111' : OUTVECT6=0;
INVECT6(0..29) := B'001111111111111111111111111111' : OUTVECT6=1;
INVECT6(0..29) := B'000111111111111111111111111111' : OUTVECT6=0;
INVECT6(0..29) := B'000011111111111111111111111111' : OUTVECT6=1;
INVECT6(0..29) := B'000001111111111111111111111111' : OUTVECT6=0;
INVECT6(0..29) := B'000000111111111111111111111111' : OUTVECT6=1;
INVECT6(0..29) := B'000000011111111111111111111111' : OUTVECT6=0;
INVECT6(0..29) := B'000000001111111111111111111111' : OUTVECT6=1;
INVECT6(0..29) := B'000000000111111111111111111111' : OUTVECT6=0;
INVECT6(0..29) := B'000000000011111111111111111111' : OUTVECT6=1;
INVECT6(0..29) := B'000000000001111111111111111111' : OUTVECT6=0;
INVECT6(0..29) := B'000000000000111111111111111111' : OUTVECT6=1;
INVECT6(0..29) := B'000000000000011111111111111111' : OUTVECT6=0;
INVECT6(0..29) := B'000000000000001111111111111111' : OUTVECT6=1;
INVECT6(0..29) := B'000000000000000111111111111111' : OUTVECT6=0;
INVECT6(0..29) := B'000000000000000011111111111111' : OUTVECT6=1;
INVECT6(0..29) := B'000000000000000001111111111111' : OUTVECT6=0;
INVECT6(0..29) := B'000000000000000000111111111111' : OUTVECT6=1;
INVECT6(0..29) := B'000000000000000000011111111111' : OUTVECT6=0;
INVECT6(0..29) := B'000000000000000000001111111111' : OUTVECT6=1;
INVECT6(0..29) := B'000000000000000000000111111111' : OUTVECT6=0;
INVECT6(0..29) := B'000000000000000000000011111111' : OUTVECT6=1;
INVECT6(0..29) := B'000000000000000000000001111111' : OUTVECT6=0;
INVECT6(0..29) := B'000000000000000000000000111111' : OUTVECT6=1;
INVECT6(0..29) := B'000000000000000000000000011111' : OUTVECT6=0;
INVECT6(0..29) := B'000000000000000000000000001111' : OUTVECT6=1;
INVECT6(0..29) := B'000000000000000000000000000111' : OUTVECT6=0;
INVECT6(0..29) := B'000000000000000000000000000011' : OUTVECT6=1;
INVECT6(0..29) := B'000000000000000000000000000001' : OUTVECT6=0;
INVECT6(0..29) := B'000000000000000000000000000000' : OUTVECT6=1;
```

ICT Vector #7

```
INVECT7(0..29) := (041)||(042)||(044)||(046)||(047)||
                  (048)||(050)||(051)||(055)||(056)||
                  (057)||(058)||(059)||(062)||(063)||
                  (064)||(068)||(069)||(073)||(074)||
                  (075)||(076)||(077)||(079)||(080)||
                  (081)||(082)||(083)||(085)||(086);
OUTVECT7       := (177);

INVECT7(0..29) := B'111111111111111111111111111111' : OUTVECT7=1;
INVECT7(0..29) := B'011111111111111111111111111111' : OUTVECT7=0;
INVECT7(0..29) := B'001111111111111111111111111111' : OUTVECT7=1;
INVECT7(0..29) := B'000111111111111111111111111111' : OUTVECT7=0;
INVECT7(0..29) := B'000011111111111111111111111111' : OUTVECT7=1;
INVECT7(0..29) := B'000001111111111111111111111111' : OUTVECT7=0;
INVECT7(0..29) := B'000000111111111111111111111111' : OUTVECT7=1;
INVECT7(0..29) := B'000000011111111111111111111111' : OUTVECT7=0;
INVECT7(0..29) := B'000000001111111111111111111111' : OUTVECT7=1;
INVECT7(0..29) := B'000000000111111111111111111111' : OUTVECT7=0;
INVECT7(0..29) := B'000000000011111111111111111111' : OUTVECT7=1;
INVECT7(0..29) := B'000000000001111111111111111111' : OUTVECT7=0;
INVECT7(0..29) := B'000000000000111111111111111111' : OUTVECT7=1;
INVECT7(0..29) := B'000000000000011111111111111111' : OUTVECT7=0;
INVECT7(0..29) := B'000000000000001111111111111111' : OUTVECT7=1;
INVECT7(0..29) := B'000000000000000111111111111111' : OUTVECT7=0;
INVECT7(0..29) := B'000000000000000011111111111111' : OUTVECT7=1;
INVECT7(0..29) := B'000000000000000001111111111111' : OUTVECT7=0;
INVECT7(0..29) := B'000000000000000000111111111111' : OUTVECT7=1;
INVECT7(0..29) := B'000000000000000000011111111111' : OUTVECT7=0;
INVECT7(0..29) := B'000000000000000000001111111111' : OUTVECT7=1;
INVECT7(0..29) := B'000000000000000000000111111111' : OUTVECT7=0;
INVECT7(0..29) := B'000000000000000000000011111111' : OUTVECT7=1;
INVECT7(0..29) := B'000000000000000000000001111111' : OUTVECT7=0;
INVECT7(0..29) := B'000000000000000000000000111111' : OUTVECT7=1;
INVECT7(0..29) := B'000000000000000000000000011111' : OUTVECT7=0;
INVECT7(0..29) := B'000000000000000000000000001111' : OUTVECT7=1;
INVECT7(0..29) := B'000000000000000000000000000111' : OUTVECT7=0;
INVECT7(0..29) := B'000000000000000000000000000011' : OUTVECT7=1;
INVECT7(0..29) := B'000000000000000000000000000001' : OUTVECT7=0;
INVECT7(0..29) := B'000000000000000000000000000000' : OUTVECT7=1;
```

ICT Vector #8

```
INVECT8(0..34) := (066)||(087)||(071)||(054)||(107)||
                  (109)||(049)||(088)||(089)||(090)||
                  (091)||(092)||(026)||(023)||(099)||
                  (100)||(101)||(102)||(103)||(104)||
                  (105)||(106)||(191)||(052)||(033)||
                  (037)||(031)||(153)||(152)||(151)||
                  (187)||(028)||(027)||(125)||(186);
OUTVECT8       := (173);

INVECT8(0..34) := B'01000000000000000000000001111101111' : OUTVECT8=0;
INVECT8(0..34) := B'11000000000000000000000001111101111' : OUTVECT8=1;
INVECT8(0..34) := B'10000000000000000000000001111101111' : OUTVECT8=0;
INVECT8(0..34) := B'10100000000000000000000001111101111' : OUTVECT8=1;
INVECT8(0..34) := B'10110000000000000000000001111101111' : OUTVECT8=0;
INVECT8(0..34) := B'10111000000000000000000001111101111' : OUTVECT8=1;
INVECT8(0..34) := B'10111100000000000000000001111101111' : OUTVECT8=0;
INVECT8(0..34) := B'10111110000000000000000001111101111' : OUTVECT8=1;
INVECT8(0..34) := B'10111111000000000000000001111101111' : OUTVECT8=0;
INVECT8(0..34) := B'10111111100000000000000001111101111' : OUTVECT8=1;
INVECT8(0..34) := B'10111111110000000000000001111101111' : OUTVECT8=0;
INVECT8(0..34) := B'10111111111000000000000001111101111' : OUTVECT8=1;
INVECT8(0..34) := B'10111111111100000000000001111101111' : OUTVECT8=0;
INVECT8(0..34) := B'10111111111110000000000001111101111' : OUTVECT8=1;
INVECT8(0..34) := B'10111111111111000000000001111101111' : OUTVECT8=0;
INVECT8(0..34) := B'10111111111111100000000001111101111' : OUTVECT8=1;
INVECT8(0..34) := B'10111111111111110000000001111101111' : OUTVECT8=0;
INVECT8(0..34) := B'10111111111111111000000001111101111' : OUTVECT8=1;
INVECT8(0..34) := B'10111111111111111100000001111101111' : OUTVECT8=0;
INVECT8(0..34) := B'10111111111111111110000001111101111' : OUTVECT8=1;
INVECT8(0..34) := B'10111111111111111111000001111101111' : OUTVECT8=0;
INVECT8(0..34) := B'10111111111111111111100001111101111' : OUTVECT8=1;
INVECT8(0..34) := B'10111111111111111111110001111101111' : OUTVECT8=0;
INVECT8(0..34) := B'10111111111111111111111001111101111' : OUTVECT8=1;
INVECT8(0..34) := B'10111111111111111111111101111101111' : OUTVECT8=0;
INVECT8(0..34) := B'10111111111111111111111111111101111' : OUTVECT8=1;
INVECT8(0..34) := B'10111111111111111111111110111101111' : OUTVECT8=1;
INVECT8(0..34) := B'10111111111111111111111110011101111' : OUTVECT8=0;
INVECT8(0..34) := B'10111111111111111111111110001101111' : OUTVECT8=1;
INVECT8(0..34) := B'10111111111111111111111110000101111' : OUTVECT8=0;
INVECT8(0..34) := B'10111111111111111111111110000001111' : OUTVECT8=1;
INVECT8(0..34) := B'10111111111111111111111110000011111' : OUTVECT8=0;
INVECT8(0..34) := B'10111111111111111111111110000010111' : OUTVECT8=1;
INVECT8(0..34) := B'10111111111111111111111110000010011' : OUTVECT8=0;
INVECT8(0..34) := B'10111111111111111111111110000010001' : OUTVECT8=1;
INVECT8(0..34) := B'10111111111111111111111110000010000' : OUTVECT8=0;
```

Appendix E. Pre-Assembly Bake Requirement

Vero chip should be baked at 125 degress C for 24 hours prior to card assembly. Soldering of the part must be completed within 96 hours after baking.

Appendix F. Electrical Characteristics

F.1 D.C. Specifications

F.1.1 Operating Temperature

Operating Temperature - Junction:	10 Degrees C to 83 Degrees C
Operating Temperature - Case:	Maximum - 80 Degrees C

Junction temperature based on simulation temperature range
of 10 degrees C to 85 degrees C.

Maximum case temperature based on maximum power dissipation,
a junction/case (JC) thermal resistance of 3.0, and maximum
junction temperature.

F.1.2 Power Dissipation

Average:	0.6 Watts
Maximum:	0.8 Watts

F.1.3 Operating Voltage

Operating Voltage: 5 Volts +/- 10%

Appendix G. Packaging

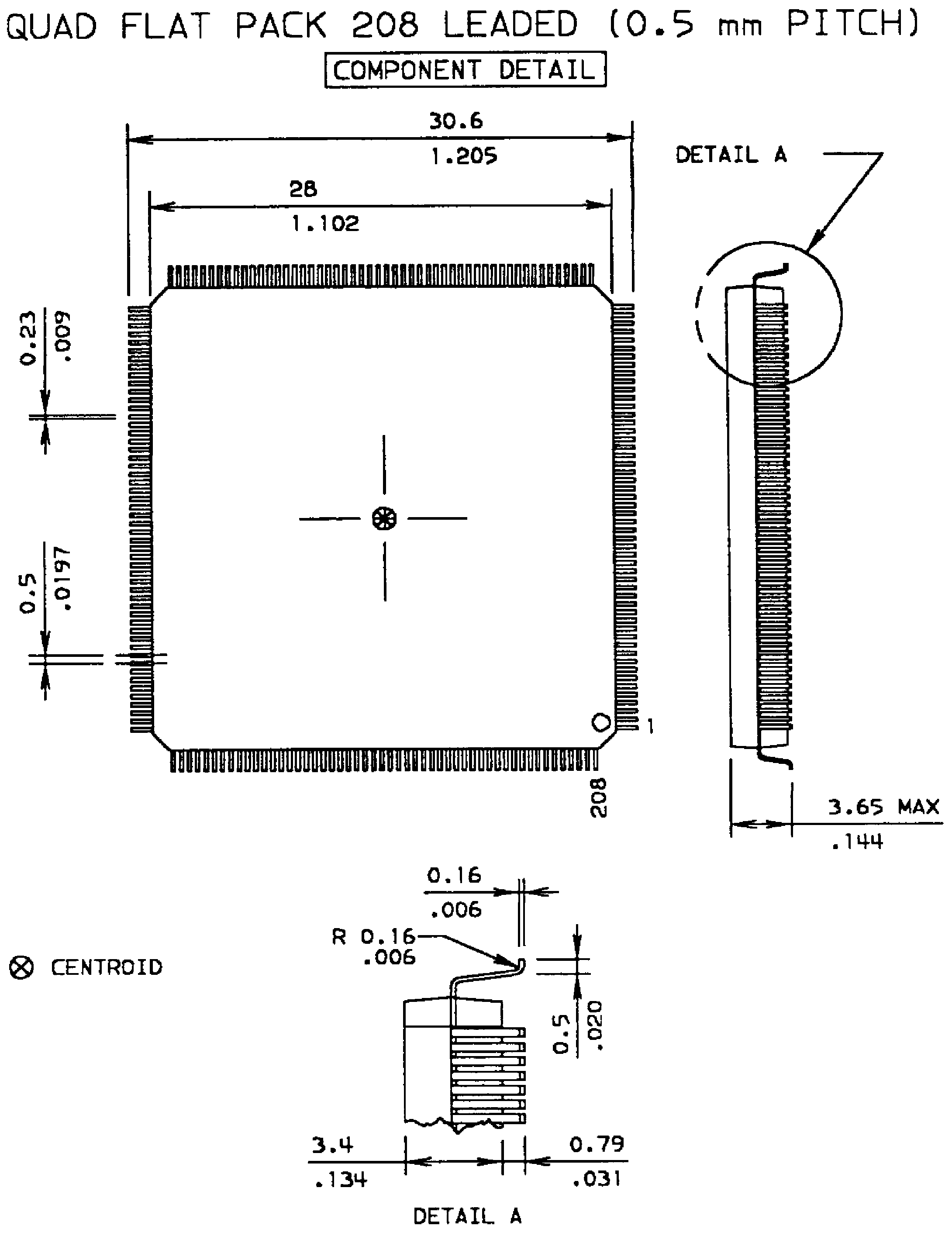
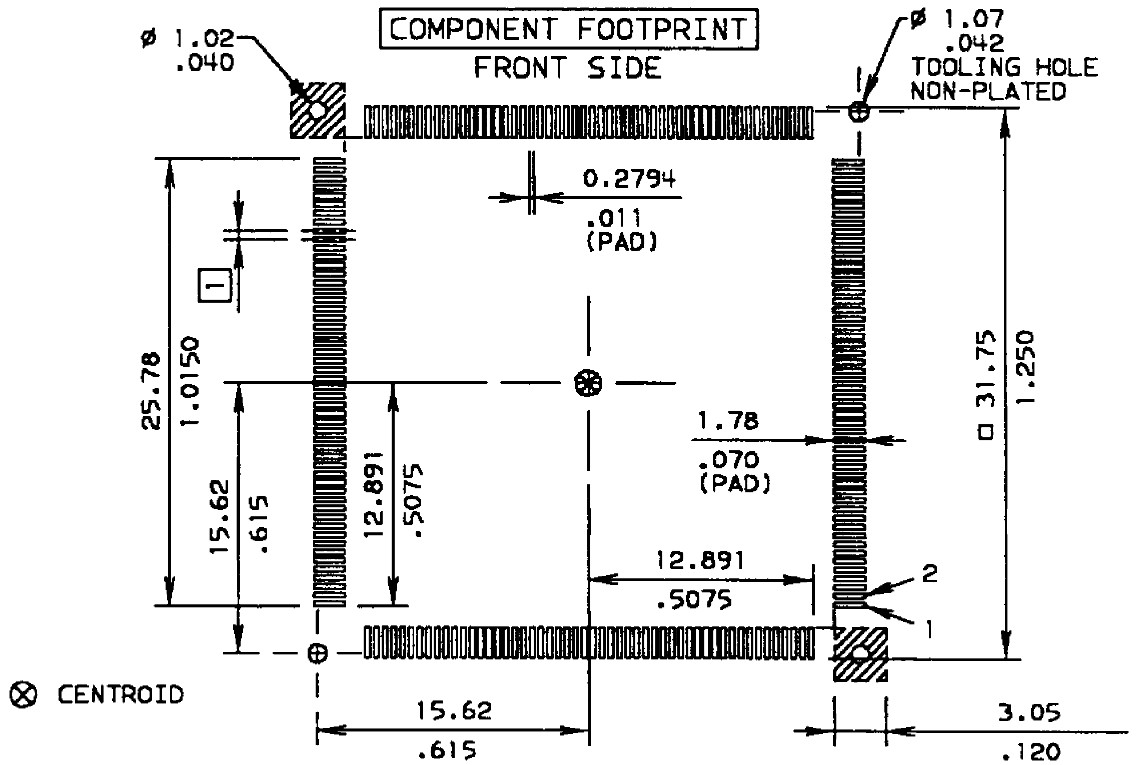


Figure 4. Package Information

QUAD FLAT PACK 208 LEADED (0.5 mm PITCH)



SMT SPACING (PITCH) FROM PAD CENTERLINE
• SPACING IS TO NEAREST 0.013 (.0005 IN)

PAD	SPACING $\boxed{1}$	PAD	SPACING $\boxed{1}$	PAD	SPACING $\boxed{1}$
1	0.000 (.0000)	19	9.004 (.3545)	37	17.996 (.7085)
2	0.495 (.0195)	20	9.450 (.3740)	38	18.504 (.7285)
3	1.003 (.0395)	21	9.995 (.3935)	39	18.999 (.7480)
4	1.498 (.0590)	22	10.503 (.4135)	40	19.495 (.7675)
5	1.994 (.0785)	23	10.998 (.4330)	41	20.003 (.7875)
6	2.502 (.0985)	24	11.506 (.4530)	42	20.498 (.8070)
7	2.997 (.1180)	25	12.002 (.4725)	43	21.006 (.8270)
8	3.505 (.1380)	26	12.497 (.4920)	44	21.501 (.8465)
9	4.001 (.1575)	27	13.005 (.5120)	45	21.996 (.8660)
10	4.496 (.1770)	28	13.500 (.5315)	46	22.504 (.8860)
11	5.004 (.1970)	29	13.995 (.5510)	47	23.000 (.9055)
12	5.499 (.2165)	30	14.503 (.5710)	48	23.495 (.9250)
13	5.994 (.2360)	31	14.999 (.5905)	49	24.003 (.9450)
14	6.502 (.2560)	32	15.494 (.6100)	50	24.498 (.9645)
15	6.998 (.2755)	33	16.002 (.6300)	51	25.006 (.9845)
16	7.506 (.2955)	34	16.497 (.6495)	52	25.502 (1.0040)
17	8.001 (.3150)	35	17.005 (.6695)		
18	8.496 (.3345)	36	17.501 (.6890)		

Figure 5. Circuit Board Footprint