Areté Systems Corporation 1100 and 1200

DMC4 Four-Channel DMA Memory Controller Board Functional Specifications

February 4, 1986 *Part Number FS-01181-02*

Functional Specifications

DMC4 - Four-Channel DMA

TABLE OF CONTENTS

Section 1. Introduction

Section 2. PMB (Processor Memory Bus)

Section 3. ICB (Interprocessor Communication Bus)

Section 4. DTB (Data Transfer Bus)

Section 5. DMA Channels

Functional Specifications

Four-Channel DMA

Section One. Introduction

This section describes the hardware functions of the Arete Four-channel DMA (direct memory access) Memory Controller Board, or DMC4. This board is a primary element in the Arete bus structure. The DMC connects to three of the system-level buses in the Arete 1000 series systems:

1. The PMB (processor memory bus).
2. The ICB (interprocessor communication bus).
3. The DTB (data transfer bus).

The PMB is used for transferring data between bus masters and main system memory. Bus masters consist of up to four CPU boards, and four DMA channels. The refresh controller uses the PMB to refresh main memory, but no data transfers occur during a refresh cycle. The system supports up to four memory boards with 8 megabytes of RAM (random access memory) on each board. Each memory board is controlled and accessed via the PMB bus.

The ICB is used by the master CPU to transfer small amounts of data at relatively low speeds to the DMC, as well as any other boards in the system. On the DMC, the ICB is used for configuring the memory error correction logic, and for reading board status.

The DTB is used for transferring large blocks of data quickly between I/O processors and main memory (or other I/O processors). The DMC4 outputs a master ID code every 160 nanoseconds, and a DTB master may then transfer 32 bits of data to a slave during the 160 nanosecond period. This results in a 25 megabyte per second transfer rate. The DMC4 has two DMA channels, which allow I/O processors to transfer data to and from main memory.

A block diagram of DMC functions is shown in Figure 1-1. Table 1 lists the many PALs (programmable array logic) on the DMC board. These PALs control the various processes that occur on the DMC and other boards that share the system buses. The circuits used to implement these three bus interfaces are described in the following sections on the PMB, the ICB, and the DTB. In addition, there is a section devoted to DMA operation.

Figure 1-1. DMC4 Block Diagram

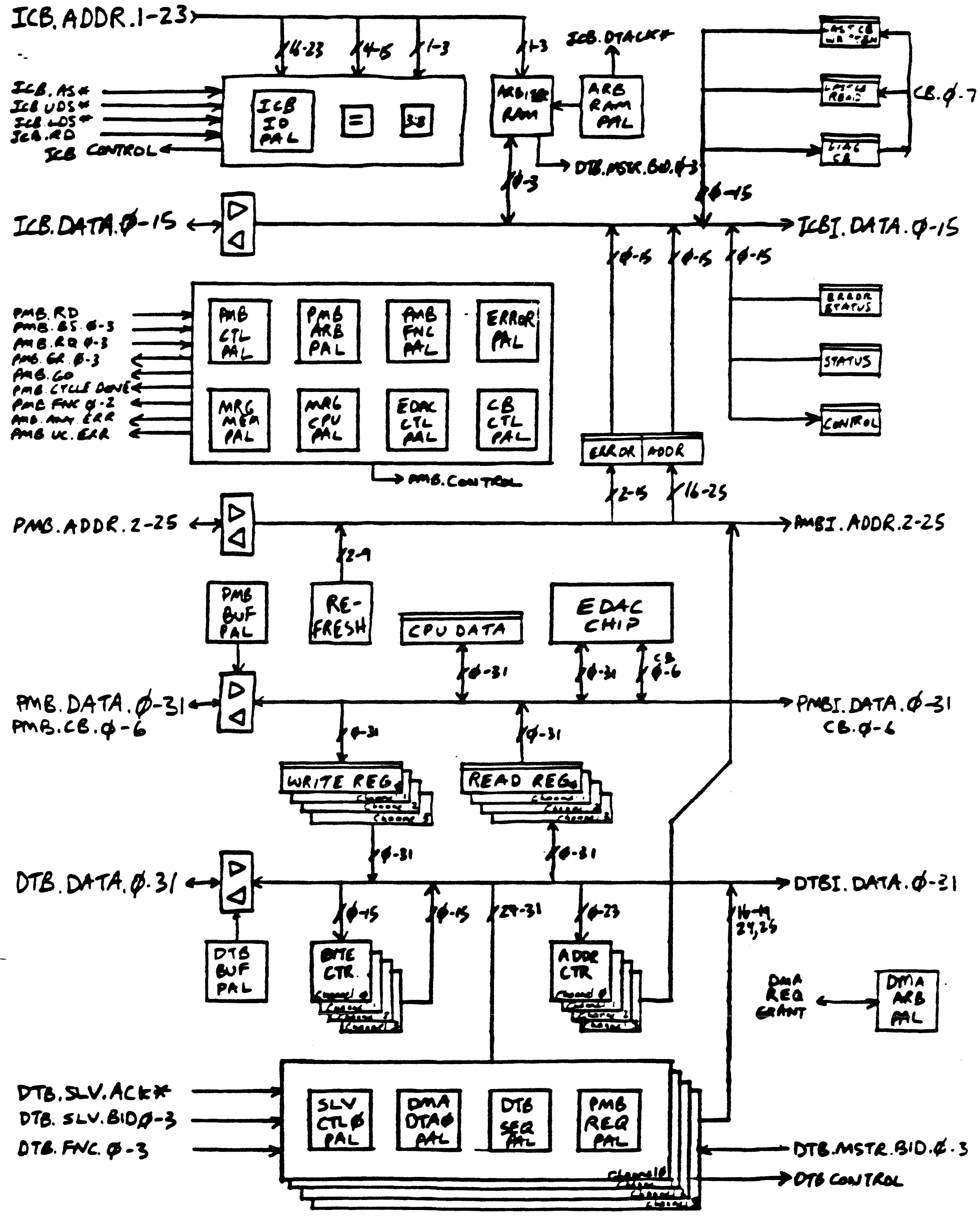

Table 1. DMC PALs (Programmable Array Logic)

PAL Name	Summary of Conditions
PMB CTL	PMB.CYCLE.DONE, CYCLE START, GRANT ENABLE, State codes.
PMB ARB	PMB.GO; Grants for refresh, DMA, and CPUs 0-3.
PMB BUF	Enables and changes direction for PMB data and check bit drivers.
PMB FNC	PMB.FNC.0-1, Read cycle, Read modify write cycle, latch PMB signals.
MRG MEM	Byte output enables 0-3 for memory data.
MRG CPU	Byte output enables 0-3 for CPU data.
CB CTL	Internal PMB check bit output enables and latch enable.
EDAC CTL	Three EDAC chip control signals.
ERROR	PMB.ANY.ERR, PMB.UC.ERR, other error signals, latch error address and syndrome bits, latch check bits, abort DMA.
ICB IO	ICB interface control signals.
ARB RAM	Arbiter RAM control signals, DTACK.
DTB BUF	Enable and change direction for DTB data drivers, and DTB write.
DMA ARB	DMA channels 0,1 grants, DMA request.
SLV CTL	DMA Channel load long word count, load address, slave ID, match, read status, transferring, slave acknowledge.
DMA DTA	DMA channel sets and reset for data sent and data received J-K flip-flops.
DTB SEQ	DMA channel locked, busy, transfer enable, abort, DTB synchronized PMB cycle done.
PMB REQ	DMA channel decrement byte count, increment address, request, common DMA to PMB write signal.

Section Two. PMB (Processor Memory Bus)

The PMB provides the 32-bit data path between main memory and the CPUs and DMA channels in the system. Up to four CPUs can dynamically share up to 16 megabytes of main memory on the PMB. Thus, the PMB provides a highly efficient means of tightly coupling main memory to the computational subsystem. The PMB includes 32 bits of data, 26 address bits, and control signals.

PMB Bus Arbiter

The PMB bus arbiter accepts requests for use of the PMB by the various PMB bus masters. The highest priority requester is then granted use of the bus to transfer data to or from main memory. There are six request inputs for the PMB: 4 CPU requests, 1 DMA request (for the four DMA channels), and the refresh request. Following is a list of the PMB request priorities:

Device	PMB Request Priority
Refresh	0 (highest)
DMA channels 0-3	1
CPU 0	2
CPU 1	3
CPU 2	4
CPU 3	5 (lowest)

These requests are activated on the rising edge of the PMB clock (25 MHz) by the various bus requesters, and are then synchronized again on the DMC by the PMB clock. The PMB arbiter PAL (PMB ARB) uses one set of request and grant signals to interface with the four DMA channels.

The PMB CTL PAL begins a PMB cycle by activating the grant enable signal (GR.ENB*). This signal allows the PMB ARB PAL to issue a bus grant to the highest priority requester. The PMB ARB PAL performs both the priority encoding and grant decoding functions simultaneously. A particular grant output is activated when GR.ENB* is active, the corresponding request is active, no grants are currently active, and no higher requests are active. The requester must not drive the PMB bus once its grant has been deactivated.

The PMB CTL PAL detects what type of cycle is taking place and issues the PMB.CYC.DONE* signal after the appropriate number of clock periods have elapsed to indicate the completion of the cycle. The following chart shows the number of PMB clock cycles, or states, required for each type of cycle:

Type of Cycle	Number of States	PMB States
Refresh	8	S0 - S7
32 bit write	8	S0 - S7
Read with no error	10	S0 - S9
Read with error	13	S0 - S12
Read/Modify/Write	15	S0 - S14

PMB.CYC.DONE* occurs during the last clock period of the cycle. The CPU uses PMB.CYCLE.DONE to latch data on a read cycle and the memory board uses it to return to an idle state and wait for the next cycle.

Function Code State Machine

The function code state machine PAL (PMB FNC) performs several functions. When PMB.CYC.START* is detected, it first activates the CLK.PMB.SIGS* signal. This latches the current state of the PMB byte select lines, and a code representing the current bus master. This information is used in the event of a memory error. The PMB data is also latched on a write or RMW cycle. Two of the PMB function codes (PMB.FNC.0-1*) are then enabled onto the bus during states S2 and S3. The function code state machine uses the PMB byte select lines, the PMB read signal, and the DMA read signal to determine the proper function code:

FNC.2*	FNC.1*	FNC.0*	Description
0	x	x	Refresh
1	0	0	8,16,24 Bit Write(Read/Modify/Write)
1	0	1	32 Bit Write
1	1	0	Read(Up to 32 Bits)
1	1	1	No Operation

The function code bit (PMB.FNC.2*) is dedicated as the refresh function code, and is derived from the refresh grant signal. These three function codes are used by the memory and CPU boards to determine the type of PMB bus cycle.

The UC.ERR* output of the ERROR PAL goes active during S10 if an uncorrectable error occurs during a read or RMW cycle. It stays active until PMB.CYC.DONE* occurs, and is used by the CPU board to generate a bus error for the CPU.

PMB Data Control Buffers

The 32-bit PMB data bus is buffered with bidirectional transceivers. These bus buffers are controlled by the PMB buffer control PAL (PMB BUF). All four transceivers have a single enable and a single directional control. The inputs to the PMB BUF PAL provide the type of PMB cycle and the current PMB state within the cycle. For a refresh cycle, the buffers are not enabled.

For a read cycle, the data buffers are enabled at state S1 and are disabled when PMB.CYCLE.DONE occurs. The directional control allows data to be received from the PMB onto the DMC board. The PMB data bus is driven by the memory board, and data is latched directly by the current bus master. The DMC only reads the data to perform error detection and correction. When one of the DMA channels is the active PMB master, the memory data is read through the buffers and passed through the DMC to the DTB bus.

For a read cycle with a memory error, the buffers are disabled at S9 and the direction of the buffers is changed at S10. The buffers are enabled again at S11. This allows the corrected data to be placed on the PMB bus by the DMC. By this time in the cycle, the memory board has disabled its data bus drivers, allowing the DMC to drive the bus.

For a RMW (read modify write) cycle, the write data is first latched on the DMC. The corresponding 32-bit location is then read from memory and corrected. At this time, the buffers are disabled,turned around, and then enabled as is done for the read-with-error cycle. The new CPU bytes and corrected memory bytes are then merged and enabled onto the PMB bus to be written to memory.

For a 32-bit write cycle, the buffers are enabled for the entire cycle, and are directed toward the DMC. The current bus master drives the data bus, and the memory board latches and stores the data. The DMC uses the data to generate the seven check bits (PMB.CB0 through PMB.CB6), which are also written into memory.

When one of the DMA channels is the current PMB bus master, the PMB buffers are not enabled until state S2, because the buffer direction bit for DMA transfers is not valid until S1. If the buffers were enabled at S1, as with the other cycles, there would be bus contention for a memory write cycle. The DMA channels enable their bus buffers when the grant is received at S0.

PMB Check Bit Buffers (Control)

The PMB has a 7-bit check bit bus for the memory boards. During a read or a read modify write cycle, these check bits are read by the DMC and used to detect and correct any errors that may have occurred. During write or read modify write cycles, new check bits are generated by the DMC and written into memory.

The PMB BUF PAL is used to control the enable and the direction specification of the bidirectional check bit buffer. The direction bit is set at S6 to read the check bits during a read or read modify write cycle. At S9 the direction is changed to write the check bits to the bus. For a 32-bit write cycle, the direction is always toward the memory boards.

The check bit buffer is enabled at S3 for a 32-bit write cycle, thus ensuring that the check bits are valid at the memory boards in time for the write cycle. For other cycles, the buffer is enabled at S6. The buffer is then enabled at S6. The buffer is then disabled at S8 after the check bits have been latched in the EDAC chip. For a read modify write cycle, the buffer is again enabled at S11 to allow the new check bits to be written to memory.

EDAC (Error Detection and Correction)

A special 32-bit EDAC chip is used on the DMC. This device has the capability to generate a 7-bit check byte for each 32-bit data value. Each 32- bit word in memory has a corresponding check byte. When data is read from memory, the check byte and data are both processed by the EDAC chip to determine if an error has occurred. If a single-bit error has occurred, the EDAC corrects the error and produces a syndrome to indicate which bit was in error. Multiple bit errors are uncorrectable and cause a bus error to be generated on the CPU board.

Four PALs are used to control the EDAC and associated buffers and latches. Included in the PMB CTL PAL is a counter whose outputs provide state information to other PALs. The counter starts when PMB.CYC.START occurs. Its outputs are the binary representation of the current state of the PMB cycle. The PALs use this state information to determine when to activate their outputs. The counter is then cleared when PMB.CYC.DONE occurs.

The EDAC control PAL (EDAC CTL) has three outputs. EDAC.S0 and EDAC.S1 determine the operating mode of the EDAC chip as follows:

EDAC.S1	EDAC.S0	Summary of Conditions
0	0	Error detector disabled, check bit generator enabled, corrector disabled, generated check bits out to check bits, open input data latch.
0	1	Error detector enabled, check bit generator disabled, corrector disabled, input check bits out to check bits, open input data latch.
1	0	Error detector enabled, check bit generator disabled, corrector disabled, syndrome out to check bits, open data input latch, open check bit input latch.
1	1	Error detector enabled, check bit generator disabled, corrector enabled, syndrome out to check bits, data and check bits held in input latches.

During a read cycle, EDAC.S1 and EDAC.S0 both start out low. At state S5 EDAC.S1 goes high, thus enabling the error detector, disabling the check bit generator, and opening the check bit input latch for incoming check bits from memory. At state S8 EDAC.S0 goes high, thus latching the data and check bits in the EDAC chip and enabling the data corrector. EDAC.S1 and EDAC.S0 both remain high throughout the rest of the cycle. If an error occurs, the corrected data and syndromes are available at the outputs of the EDAC.

During a 32-bit write cycle, EDAC.S1 and EDAC.S0 both remain low throughout the cycle. This mode is used only for generating new check bits.

During a read modify write cycle, EDAC.S1 and EDAC.S0 both start out low. At state S5 EDAC.S1 goes high. This enables the error detector, disables the check bit generator, and opens the check bit input latch for incoming check bits from memory. At state S8 EDAC.S0 goes high, thereby latching the data and check bits in the EDAC chip and enabling the data corrector. EDAC.S1 and EDAC.S0 both go low again in state S10. The check bit generator is enabled, and the input data latches are opened up so that new check bits can be generated.

The LED.BO signal controls the data output latch of the EDAC chip. During a read modify write cycle, LED.BO goes high at S9, thus latching the corrected data in the EDAC output latches. LED.BO remains high until the end of the cycle.

If error correcting has been disabled (INH.EDAC high), EDAC.S0 is prevented from going high at state S7. This prevents the data corrector from being enabled. LED.BO goes high at S8 for a read with error or a read modify write cycle if INH.EDAC is high. This causes the uncorrected input data to be latched in the EDAC output latch. The data from the output latch is then written to memory for a read modify write cycle, or sent to the current bus master for a read cycle.

There are two PALs used for merging bytes during a read modify write cycle. If the current PMB master is writing less than four bytes to memory, then the corresponding 32-bit word must first be read from memory and then must be corrected. The new bytes to be written to memory are merged with the old corrected bytes, forming a new 32-bit word to be written to memory.

The source of the new bytes is a 32-bit latch controlled by the CPU data byte merge control PAL (MRG CPU). The source of the old, corrected bytes is the EDAC chip. The PAL that controls the byte enable outputs of the EDAC chip to merge the memory data bytes is called the MRG MEM PAL. For each byte being written to memory, either the latch or the EDAC is enabled to supply that particular byte. The two

PALs enable the latch or EDAC outputs from state S10 until the end of the cycle.

For a read with error cycle, the MRG MEM PAL enables the corrected data out of the EDAC chip from state S10 until the end of the cycle.

The check bit control PAL, CB CTL controls the check bit buffers and latches. The OE.CB output enables the EDAC check bit output buffer. OE.CB goes active at S3 for 32-bit write cycles, and at S10 for read with error or read modify write cycles.

If the DIAG bit is set, OE.CB.DIAG goes active instead of OE.CB. In this case, the check bits to be written to memory are provided by a diagnostic check bit latch instead of the EDAC. The diagnostic check bit latch outputs to the internal check bits. The diagnostic check bits are written into the latch via the ICB bus. For every 32-bit write or read modify write cycle, the check bits being written to memory are stored in a latch. The CLK.CB signal latches the check bits during S4 for a 32-bit write, and during S13 for a read modify write. This last-check-bits-written latch may then be read via the ICB for diagnostic purposes.

For every read or read modify write cycle, the check bits read from memory are stored in a latch. The ERR.CHK signal latches the read check bits into this latch. This last-check-bits-read latch may be read over the ICB for diagnostic purposes.

Error Reporting

Memory error reporting is controlled by the ERROR PAL. For read or read modify write cycles, the ERR.CHK signal goes active during S7. This signal causes a flip-flop to be set on the rising edge of S8. The output of this flip-flop (ERR.CHK.SYNC) gates through the SGL.BIT.ERR signal for read or read modify write cycles. The output of this gate (ANY.ERR) is valid only during state S8.

The MEM.ERR output of the ERROR PAL goes active at S9 if ANY.ERR is true, and stays active until the ICB status port is read. The MEM.ERR bit can be read via the ICB, and causes an ICB interrupt when it is active.

The LAT.ERR output of the ERROR PAL goes active at the same time as MEM.ERR, but only stays active until the end of the cycle. It is used by the MRG MEM byte merging PAL when an error occurs during a read cycle.

The UC.ERR output of the ERROR PAL goes active during S10 if an uncorrectable error occurs during a read or a read modify write cycle. It stays active until PMB.CYC.DONE occurs and is used by the CPU board to generate a bus error for the CPU.

The CLK.ERR.REG goes active during state S9 if no previous memory error is pending. This signal latches the syndromes out of the EDAC chip, and the grant, byte select, and PMB.RD signals. This error information may then be read by the CPU via the ICB.

The CLK.ADDR.ERR.REG signal goes active during S2 if no previous memory error is pending. It is used to latch the current PMB address. If an error occurs, this address is held in a latch and may then be read via the ICB.

The ABORT.DMA signal goes active during S10 if an uncorrectable error occurs during a memory to DMA channel transfer. This signal causes the active DMA channel to be reset.

PMB Cycle Descriptions

The following is a state-by-state description of each type of PMB cycle.

32-Bit Write Cycle

S0-2 CPU REQUEST goes active.

S0-1 REQUEST is synchronized to PMB.CLK so it can be sampled by the arbiter state machine on the next clock edge.

S0 GO, GRANT, and CYCLE START go active. CPU or DMA drives address and data on the bus.

S1 Address, data, byte selects, and PMB.RD are valid on the bus. DMA channel drives write signal, if applicable. The data buffers are enabled so check bit generation can begin.

S2 Write function code is driven onto the bus. Grant code, byte selects, and data are latched with CLK.PMB.SIGS*. EDAC check bit output buffer enabled. If this is a DMA write, data buffers are not enabled till now.

S3 Check bit buffer enabled to drive generated check bits on the bus. This buffer is not enabled until now because the state machines do not know that a write is occurring until S2.

S4 Function codes are disabled. Check bits are latched in last-check bits written latch. WRITE goes active on memory board.

S5 RAM WRITE ENABLE still active on memory board.

S6 Disable GRANT and GO. CPU or DMA disables its address and data drivers.

S7 CYCLE DONE goes active.

S0 Data buffers and check bit buffers are disabled and their direction is changed. Activate GO, GRANT, and CYCLE START for the next cycle, if back to back.

Read Cycle With No Error

S0-2 CPU or DMA request goes active.

S0-1 Request is synchronized to PMB.CLK so it can be sampled by the arbiter state machine on the next clock edge.

S0 GO, GRANT, and CYCLE START go active. CPU or DMA drives address on the bus.

S1 ADDRESS, BYTE SELECTS, and PMB.RD are valid on the bus. Data buffers are enabled.

S2 Read function code driven onto the bus. Grant code and BYTE SELECTS are latched with CLK.PMB.SIGS*. The latched read signal is set. Address is latched in error address latch in case of an error.

S3 EDAC check bit output buffer disabled.

S4 Function codes are disabled.

S5 Disable GRANT and GO.

S6 Change direction of check bit buffer to prepare to read the check bits from the bus. Set EDAC.S1 high to put EDAC chip in read mode (open up input latches for data, check bits). Memory board starts driving the data and check bits on the bus. Enable the check bit buffer so that the check bits can be read.

S7 RAM data and check bits are now valid.

S8 Set EDAC.S0 high to latch the data and check bits in the EDAC chip, and to start correcting the data in case there is an error. Set the memory error flip-flop to enable any possible error. Disable the check bit buffer in anticipation of outputting the syndromes from the EDAC chip if an error occurs.

S9 Activate CYCLE DONE. The CPU board latches the data on the next rising edge of the PMB clock. Change the direction of the check bit buffer. Check bits are latched in 'last check bits read' latch.

S0 Disable LATCHED READ. Disable and change direction of data buffer and set EDAC.S0 and EDAC.S1 low again to prepare for next cycle. Activate GO and GRANT for the next cycle, if back to back.

Read Cycle With Error

S0-2 CPU or DMA request goes active.

S0-1 Request is synchronized to PMB.CLK so it can be sampled by the arbiter state machine on the next clock edge.

S0 GO, GRANT, and CYCLE START go active. CPU or DMA drives address on the bus.

S1 ADDRESS, BYTE SELECTS, and PMB.RD are valid on the bus.

S2 Read function code is driven onto the bus. Grant code and BYTE SELECTS are latched with CLK.PMB.SIGS*. The latched read signal is set. Address is latched in error address latch in case of an error.

S3 EDAC check bit output buffer disabled.

S4 Function codes are disabled.

S5 Disable GRANT and GO.

S6 Change direction of check bit buffer to prepare to read the check bits from the bus. Set EDAC.S1 high to put EDAC chip in read mode (open up input latches for data, check bits). Memory board starts driving the data and check bits on the bus. Enable the check bit buffer so that the check bits can be read.

S7 RAM data and check bits are now valid.

S8 Set EDAC.S0 high to latch the data and check bits in the EDAC chip, and to start correcting the data in case there is an error. Set the memory error flip-flop to enable any possible error. Errors are enabled onto the bus. Disable the check bit buffer in anticipation of outputting the syndromes from the EDAC chip.

S9 Set the memory error flip-flop. Activate errors and disable data buffers so that the bus can be turned around. Enable the output check bit buffer of the EDAC chip to get the syndrome. Change direction of the check bit buffer. Check bits latched in 'last check bits read' latch.

S10 Activate UNCORRECTABLE ERROR if one has occurred. The corrected syndrome is now valid. Latch the syndrome in an error latch. The memory board stops driving data and check bits at this state. Change direction of data buffers to prepare for driving corrected data onto the bus. Enable the output data buffers from the EDAC chip to get the corrected data.

S11 The PMB data bus is now floating, and the data buffers are enabled to drive the corrected data onto the bus.

S12 Activate CYCLE DONE. The CPU board latches the corrected data on the next rising edge of the PMB clock.

S0 Disable LATCHED READ. Disable and change direction of data buffer and set EDAC.S0 and EDAC.S1 low again to prepare for next cycle. Disable EDAC chip data and check bit outputs. activate GO and GRANT for the next cycle, if back to back.

Read/Modify/Write Cycle With or Without Error

S0-2 CPU request goes active.

S0-1 Request is synchronized to PMB.CLK so it can be sampled by the arbiter state machine on the next clock edge.

S0 GO, GRANT, and CYCLE START go active. The CPU board drives address and data on the bus.

S1 Address, new data bytes, byte selects, and PMB.RD are valid on the bus. Data buffers are enabled.

S2 RMW function code is driven onto the bus. Grant code, byte selects and data are latched with CLK.PMB.SIGS*. The read/modify/write cycle signal is set. The address is latched in error address latch in case of an error.

S3 EDAC check bit output buffer disabled.

S4 Function codes are disabled.

S5 Disable GRANT and GO.

S6 The CPU board disables its address and data buffers. Change direction of check bit buffer to prepare to read the check bits from the bus. Set EDAC.S1 high to put EDAC chip in read mode (open up input latches for data, check bits). The memory board starts driving the data and check bits on the bus. Enable the check bit buffer so that the check bits can be read.

S7 RAM data and check bits are now valid.

S8 Set EDAC.S0 high to latch the data and check bits in the EDAC chip, and to start correcting the data in case there is an error. The memory error flip-flop is set to enable any possible error. Any error is enabled onto the bus if it is present. The check bit buffer is disabled in anticipation of outputting the syndromes from the EDAC chip if an error occurs.

S9 Disable data buffers so that the bus can be turned around. Latch the corrected data in the EDAC chip output latch. Enable the output check bit buffer of the EDAC chip to get the syndrome if there was an error. Change the direction of the check bit buffer. Check bits latched in 'last check bits read' latch.

S10 Activate UNCORRECTABLE ERROR till end of cycle if one has occurred. If an error has occurred, latch the syndrome in an error latch. The memory board stops driving data and check bits at this state. Change direction of data buffers to prepare for driving new and corrected data onto the bus. Enable the output data buffers from the EDAC chip to get the memory data bytes and the CPU holding latch for bytes which are being written by the CPU. Set EDAC.S0 and EDAC.S1 low to start generating the new check bits.

S11 The PMB data bus is now floating, and the data and check bit buffers are enabled to drive the merged data onto the bus.

S12 Data and check bits are valid on the bus 10 nanoseconds after the rising edge of S12.

S13 Memory board activates WRITE ENABLE. Check bits are latched in 'last check bits written' latch.

S14 Activate CYCLE DONE. RAM WRITE ENABLE still active on memory board.

S0 Disable READ/MODIFY/WRITE signal. Data buffers and check bit buffers are disabled and their direction is changed. EDAC chip data and check bit outputs are disabled. Activate GO and GRANT for the next cycle, if back to back.

Refresh

The dynamic RAMs used on the memory boards must be refreshed a minimum of 256 times every four microseconds. The request, PMB.RFSH.RQ*, requests a PMB cycle from the PMB arbiter. The refresh generator has highest priority on the PMB bus. When the PMB arbiter grants the request with PMB.GR.RFSH*, the output of an 8-bit refresh address counter is enabled onto the PMB address bus bits 2 through 9. The refresh request flip-flop is also cleared. The memory boards detect the refresh function code, and perform a refresh cycle. When PMB.GR.RFSH* goes inactive, the address driver is disabled and the refresh address counter is incremented in preparation for the next refresh cycle.

System Timing Generation

The bus operates synchronously at 25 MHz. The output of a 50 MHz oscillator is first divided by two producing a 25 MHz clock (SYS.CLK). The 25 MHz clock is then divided again to produce a 12.5 MHz clock (HALF.SYS.CLK), which is used by the CPU board(s). Clock phasing is important to guarantee synchronization of the various system components. Both SYS.CLK and HALF.SYS.CLK rise on the same edge of the 50 MHz oscillator output, because of final synchronization with the 50 MHz clock.

To minimize clock undershoot and overshoot on the DMC, the system clock is split into 11 different clocks. These drive the PMB.CLK into all areas of the board.

The DTB (data transfer bus) operates at one-fourth the clock rate of the PMB. The same counter and flip-flops, plus associated gates, are used to divide the 25 MHz system clock by four to produce a 25%

duty cycle, 6.25MHz clock. The high portion of the clock is 40 nanoseconds wide, and the low portion is 120 nanoseconds wide.

Section Three. ICB (Interprocessor Communication Bus)

The ICB interface on the DMC is used to configure the PMB error detection and correction circuit, and for reading back memory and DMA status. It provides synchronization and control of all the CPUs in the system. The master CPU uses the ICB to control the operation of the slave CPUs, and to monitor their status. The ICB also reads and writes the DTB master ID arbiter RAM.

Note that each board slot connects to the ICB in the Series 1000 system, and has a unique 64K byte segment in the two megabytes of ICB address space. Status and control registers for the board are located at the top of its memory segment. Each slot has a 5 bit hardwired slot address that determines which 64K byte memory segment the board occupies. Another register in the memory segment, linked through the ICB, stores the board type currently in that slot. At power up, the master CPU checks the board type from each memory segment, so the system can do a completely automatic configuration check.The ICB has 16 data bits, 21 address bits, and a subset of the VME bus control signals.

DMC Reset

There are two sources for a DMC reset. The first source, SYS.RESET*, is issued to the entire system for either a power-on or a manual system reset. When SYS.RESET* occurs, a register is asynchronously cleared, and DMC.RESET* is generated. When SYS.RESET* goes inactive, DMC.RESET* goes inactive two DTB clock periods (320 nanoseconds) later, synchronous with DTB.CLK.

The other source for a DMC reset is an output bit of the ICB output control port. When this bit is set by an ICB master, a 320 nanosecond pulse is generated on the DMC.RESET* signal. To preserve the data in the main system dynamic RAM, the refresh controller must be operating continuously. This is the reason that a pulse is generated on DMC.RESET* when the ICB resets the board. It is assumed that the SYS.RESET* is also pulsed when a manual system reset is activated. If the SYS.RESET* signal is held active for long, the data in RAM is lost.

ICB Data Buffers, Control

The 16-bit ICB data bus (ICB.DATA.0-15) is buffered with bidirectional transceivers. These devices are enabled when ICB address bits 16 through 20 match the DMC slot address, the ICB address strobe (ICB.AS*) is active, and one or both of the ICB data strobes (ICB.LDS*, ICB.UDS*) are active. The direction of the buffers is determined by the ICB read signal, ICB.RD.

I/O Decoders

The first level of address decoding is comparing the five bit board slot address with ICB address bits 16 through 20. If they match and ICB.AS* is valid, then ICB.SEL* is activated. Address bits 21 through 23 are not currently used and are pulled to logic level 1.

The second level of decoding is performed with a 20L8 PAL called ICB IO. This PAL monitors ICB.SEL*, address bits 4 through 15, the data strobes, and the ICB read signal. The ICB.IO.WR* and ICB.IO.RD* outputs are activated for ICB writes to the DMC control ports, and reads from the status ports.

Two decoders further decode the address to produce the read and write strobes to the particular I/O ports. The output port decoder has an additional control called IO.WR.STB*, which causes the write strobes to be disabled before the end of a cycle. This control provides adequate data hold time for the I/O output ports. It is generated by the arbiter RAM PAL, ARB RAM, and is 160 nanoseconds wide.

The SEL.IO* output of the PAL goes active whenever the status or control ports are being accessed, and SEL.ARB.RAM* goes active for accesses to the DTB arbiter RAM. These two outputs are used to determine when to enable the ICB data transfer acknowledge (ICB.DTACK*) signal. The ARB.RD.DATA.EN* output enables a data buffer during reads from the DTB arbiter RAM so that the data from the arbiter RAM may be sent to the master CPU.

I/O Control and Status Ports

The ICB writes out 16 bits to a control port on the DMC. All of the necessary control signals are included in these 16 bits. The outputs are all set to zero when the DMC is reset. In addition, the ICB reads from a 16 bit status port on the DMC. This register contains all the information the CPU needs during normal operation, including some read back of bits that are written in the control port.

ICB Memory Map

A map of all registers that can be accessed by the ICB on the DMC is shown in Figure 3-1.

Data Transfer Acknowledge (DTACK)

The ARB.RAM state machine is also used to generate the data transfer acknowledge signal (ICB.DTACK*) to the ICB. For read cycles, this signal is activated when the data is available, and for write cycles it is activated when the data has been stored.

The SEL.IO* signal is first synchronized with DTB.CLK before being sampled by the state machine. For an I/O read or write cycle, the state machine pulses SET.DTACK* as soon as it detects an active SEL.IO* input. The trailing edge of SET.DTACK* sets a flip-flop, the output of which enables ICB.DTACK*. At this time, the flip-flop which synchronizes SEL.IO* is cleared.

ICB.AS* goes inactive for a minimum of 65 nanoseconds between cycles, but the state machine is clocked with a 120 nanosecond clock. Therefore, the SEL.IO* and SEL.ARB.RAM* signals had to be synchronized with external flip-flops so that they could be cleared asynchronously at the end of the cycle.Otherwise, the state machine would have hung up at the end of its internal cycle waiting for the end of the ICB bus cycle. The DTACK flip-flop is then cleared when ICB.AS* goes inactive at the end of the cycle.

Figure 3-1. Memory Map (Page One)

660000 - 6607FE READ or WRITE DTB Arbiter RAM

15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
1	1	1	1	1	1	1	1	1	1	1	1	DTB.MSTR.BID.3-0			

66FFF0 READ Memory Error Address Most Significant Word

15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
1	1	1	1	1	1	PMB.ADDR.25 - 16									

66FFF2 READ Memory Error Address Least Significant Word

15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
PMB.ADDR.15-2														ACC.1-0	

ACC.1-0 = ACCess type: 00 - system data, 01 - system code, 10 - user data, 11 - user code.

66FFF4 READ Last PMB Check Bits Written to Memory

15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
1	1	1	1	1	1	1	1	0	PMB.CB.6-0						

66FFF6 READ Memory Error Status Information

15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
GR.STAT.2-0			RD	PMB.BS.3-0				MB	SYNDROME BITS FROM EDAC CHIP						

GR.STAT.2-0 = PMB GRant STATus: 0-3 = DMA 0-3, 4-7 = CPU 0-3, RD = 1 Read Cycle, MB = Multiple Bit Error.

66FFFA WRITE Diagnostic PMB Check Bits

15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
x	x	x	x	x	x	x	x	x	PMB.CB.6-0						

66FFFA READ Last Check Bits Read from Memory

15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
1	1	1	1	1	1	1	1	0	PMB.CB.6-0						

Figure 3-1. Memory Map (Page Two)

66FFFC READ DMC4 Status

15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
0	DI	DG	B4	B3	B2	B1	B0	1	IE	1	1	EE	ME	1B	0B

DI=DMC.INT	1=DMC4 has activated ICB interrupt 5 because of a memory error; to clear this bit use MEM.ERR.INT.ENA or read this port.
DG=DIAG	Read back of control port bit 13.
B4=BOARD.TYPE.4	Board-type bit 4;set to logic 0.
B3=BOARD.TYPE.3	Board-type bit 3;set to logic 1.
B2=BOARD.TYPE.2	Board-type bit 2;set to logic 1.
B1=BOARD.TYPE.1	Board-type bit 1;set to logic 0.
B0=BOARD.TYPE.0	Board-type bit 0;set to logic 1.
IE=INH.EDAC*	Read back of control port bit 12.
ME=MEM.ERR	If set to 1, a memory error has occurred. Read this port to clear bit.
0B=DMA0.BUSY*	If set to 0, DMA channel 0 is transferring data.
1B=DMA1.BUSY*	If set to 0, DMA channel 1 is transferring data.

66FFFE WRITE DMC4 Control

15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
IR	EE	DG	IE	RL	GL	I1	I0	YL	x	x	x	3R	2R	1R	0R

IR=ICB RESET	If set to 1, resets the DMC board.
EE=MEM.ERR.INT.ENA	If set to 1, enables interrupt for memory error.
DG=DIAG	If set to 1, write diagnostic check bits.
IE=INH.EDAC*	If set to 0, no error correction.
RL=Red LED	If set to 1, turn on red LED.
GL=Green LED	If set to 1, turn on green LED.
I1=DTB.ARB.INX.1	Selects one of four DTB arbitration sequences.
I0=DTB.ARB.INX.0	Selects one of four DTB arbitration sequences.
YL=Yellow LED	If set to 0, turn on yellow LED.
R3=DMA.RESET.3	If set to 1, resets DMA channel 3.
R2=DMA.RESET.2	If set to 1, resets DMA channel 2.
R1=DMA.RESET.1	If set to 1, resets DMA channel 1.
R0=DMA.RESET.0	If set to 1, resets DMA channel 0.

Section Four. DTB (Data Transfer Bus)

The DTB provides a data path between main memory and I/O boards or between I/O boards. It is used for transferring 32-bit words between I/O boards and between main memory and I/O boards. There can be up to 16 master DTB devices and 16 slave DTB devices. Each master and each slave have a four bit ID or identifier number (DTB.MSTR.BID.0-3, DTB.SLV.BID.0-3). The DMC4 has a DTB arbiter RAM that outputs a periodic sequence of DTB master ID numbers.

The master ID number is updated every DTB clock period (160 nanoseconds). A new master ID can be output every clock period, or the same master can be given the bus for several cycles. Master devices, which use the DTB more heavily, could have their IDs occur more often in the sequence than less heavily used devices.

When a master recognizes its ID number, it has the opportunity to write or read data to or from a slave. It then outputs a slave ID and a function code (DTB.FNC.0-3), which describes the type of transfer to take place. If the slave device recognizes its ID number and can perform the transfer indicated by the function codes, it enables the slave acknowledge signal, DTB.SLV.ACK*, and transfers the requested data either to or from the master. The entire sequence of events from the master ID to the slave acknowledge and data transfer occurs asynchronously during one DTB clock period.

If a master recognizes its ID but has nothing to transfer, it outputs a null function code. If a slave cannot perform a requested operation, it simply does not enable slave acknowledge. The master then can continue to send the same slave ID and function code until it gets a response. The following is a list of the function codes and a description of each:

Function Code	Description
0000	Reserved.
0001	Set Long Word Count (SET BC).
0010	Set Address (SET ADDR).
0011	Reserved.
0100	Read Data (READ DATA).
0101	Read Last Data (READ LAST).
0110	Write Data (WRITE DATA).
0111	Write Last Data (WRITE LAST).
1000	Send Interrupt (SEND INT).
1001	Spare.
1010	Not Used.
1011	Read Status (READ STAT).
1100	Spare.
1101	Spare.
1110	Spare.
1111	NOP.

83 150

The DMC4 has four DTB masters or slaves, each consisting of a DMA channel between the DTB and PMB. These DMA channels can be used as slaves by DTB masters to set up a data transfer to or from main memory in which the DMC4 DMA channel becomes the master for the actual transfer of data.

DTB Data Buffers, Control

The DTB on the DMC4 is buffered with bidirectional transceivers. The enable and the direction of data transfer for these buffers is controlled by the DTB BUF PAL. The PAL enables the buffers whenever the DTB slave ID or master ID matches the ID of one of the DMA channels (0, 1, 2, or 3). Normally the buffers are pointing towards the DTB in preparation of a read cycle. The buffers point towards the DMC for the following types of DTB cycles: WRITE DATA, WRITE LAST, SEND INT, SET ADDR, or SET BC. In these five cases, data is being transferred from the DTB to the DMC.

DTB Arbiter RAM

As described above, the arbiter RAM outputs a periodic sequence of DTB master IDs. A free-running 8-bit counter provides the address for the RAM, thus generating a 256 code sequence. There are also two I/O output bits (DTB.ARB.INX.0 and DTB.ARB.INX.1), which are the top two address bits for the RAM. Changing these bits causes an entirely different 256 code master ID sequence to be selected.

During system operation, four different ID sequences can be selected without having to update the RAM. The signals DTB.ARB.INX.0 and DTB.ARB.INX.1 are synchronized to DTB.CLK with flip-flops to guarantee a stable RAM address during selection of a new sequence. The DTB arbiter RAM is read or written via the ICB. It occupies a 2-Kbyte addressing space, and is accessed four bits at a time. The ARB RAM PAL state machine is used to control the reading and writing of the RAM.

When SEL.ARB.RAM* is activated for a read cycle, ICB.ARB.ACC* first goes active. This causes the RAM chip select to go inactive, the address multiplexers to select the ICB address instead of the free running counter, and the free running counter to be disabled. On the next DTB clock edge, RAM chip select and STB.ARB.DAT are activated. When STB.ARB.DAT goes high, the data from the RAM is latched. On the trailing edge of SET.DTACK*, a flip-flop is set low, activating ICB.DTACK*. The RAM data is then read by the ICB. The ICB.DTACK* flip-flop is cleared when ICB.AS* is cleared.

The arbiter RAM counters are disabled for three DTB clock periods while the RAM is being read. During this time, the DTB master IDs that appear on the bus are: 1111, value read from RAM, 1111. When the ICB access is complete, the master ID sequence resumes from the point where it was interrupted.

When SEL.ARB.RAM* is activated for a write cycle, ICB.ARB.ACC* first goes active. This causes the RAM chip select to go inactive, the address multiplexers to select the ICB address instead of the free running counter, and the free running counter to be disabled.

On the next DTB clock edge, RAM chip select, RAM write enable, and the write data buffer are enabled. Then the chip select and write enable are disabled, thus storing the data into the RAM. The chip select is disabled for one clock period before writing to allow the RAM outputs to float before the write data buffer is enabled.

The chip select and write enable are both disabled at the same time. This prevents the RAM from enabling its output drivers while the write data buffer is still enabled. The write data buffer is disabled one clock period after chip select and write enable are disabled to provide adequate data hold time.

ICB.DTACK* is set and cleared as with the read cycle. The arbiter RAM counters are disabled for four DTB clock periods while the RAM is being written. During this time, the DTB master IDs that appear on the bus are: 1111, write data to RAM, write data to RAM, 1111. When the ICB access is complete, the master ID sequence resumes from the point where it was interrupted.

Section Five. DMA Channels

The DMC4 has four DMA channels, which transfer 32-bit words between the DTB and the PMB. Each DMA channel is recognized by the system as either a DTB slave device or a DTB master device, depending on its activity. Before a channel can be a master, it must be a slave in order to be configured by its master. After the channel has been set up, it is the master for the duration of the data transfer. Each channel consists of a 24-bit address counter, a 16-bit long word counter, two 32-bit data latches (one for each direction) between the PMB and DTB, and four control PALs.

DMA Request Arbiter

The four DMA channels have their own PMB bus arbiter, the DMA ARB PAL. When any one of the DMA channels requires use of the PMB, the DMA ARB PAL state machine first requests use of the PMB by activating PMB.RQ.DMA*. When the main PMB bus arbiter responds with PMB.GR.DMA* and GR.ENB*, the DMA ARB PAL enables the PMB grant signal to the DMA channel with the highest priority. Since GR.ENB* goes active one cycle before PMB.GR.DMA*, the DMA REQ PAL has one clock period to determine which DMA.GR.EN* output to enable.

The DMA ARB PAL has an internal rotating priority encoder. After each DMA transfer, the priority of the four channels is rotated. If only one DMA channel is requesting the bus, then it is granted the bus each time, independent of the current priority. The priorities are only used when there are simultaneous DMA requests.

DMA Resets, Aborts

There are four sources for a DMA channel reset:

1. DMC board reset.
2. ICB DMA reset.
3. PMB abort DMA due to uncorrectable memory error (ABORT.DMA).
4. DTB SEND INT command with data bit 31 high (ABORT.DMA*).

Only ABORT.DMA* resets a single channel. If an uncorrectable error occurs during a DMA transfer, a flip-flop is set. The output of this flip-flop can then be read on data bit 24 by a DTB master during a read status command. Also available on a read status command, are the current long word count, slave board ID, and the lock bit, which indicates a busy channel. The flip-flop is cleared at the end of a DTB read status command.

DMA Data Latches, Control

Each DMA channel has two 32-bit latches between the PMB and DTB. The latch that transfers data from the DTB to PMB is clocked when the DMA.DATA.RECEIVED signal goes active for the corresponding DMA channel. DMA.DATA.RECEIVED goes active when data has been successfully received from the DTB, and is controlled by the DMA data flow control PAL (DMA DTA) for each channel.

This 32-bit latch is enabled onto the internal PMB data bus when the DMA direction bit indicates a transfer to the PMB (DMA.TO.MEM.XFER* = 0), and a PMB grant has been issued for the corresponding DMA channel.

The latch that transfers data from the PMB to the DTB is clocked on the rising edge of CLK.PMB.DATA. This signal is reset by a grant to the corresponding DMA channel, meaning that a read is occurring on the PMB. Then the signal is set by the next PMB.CYCLE.DONE - when the read data is valid. In this way, the rising edge of CLK.PMB.DATA occurs just after PMB.CYC.DONE*, when the data is guaranteed to be valid.

This 32-bit latch is enabled onto the internal DTB data bus when the DTB function codes indicate a WRITE DATA or WRITE LAST (DTB.FNC.WR* = 0) and a master match occurs for the corresponding DMA channel. The master match signal is generated for each channel when its master ID is is active on the DTB and it indicates that the channel is now master of the bus.

PMB Request State Machine

A PMB Request PAL, PMB REQ, is used by each channel. This state machine performs four functions:

1. Requests the PMB bus for a DMA channel.
2. Increments a DMA channel address counter.
3. Decrements the DMA channel long word counter.
4. Generates DMA.PMB.WR*, an internal signal similar to the PMB.RD signal.

Because the PMB REQ PAL is clocked with the PMB.CLK, all of the asynchronous DMA input signals are first synchronized to the PMB.CLK. For a DTB to PMB transfer (DMA.TO.MEM* = 0), the PMB.RQ.DMA* is activated when DMA.DATA.RCVD* goes active. This occurs when a word has been received from the DTB and needs to be written to main memory.

For a PMB to DTB transfer (DMA.TO.MEM* = 1), the PMB.RQ.DMA* is activated when DMA.DATA.SENT* goes active. This occurs when a word has been sent over the DTB to a DTB master, and the data latch is empty and waiting for another word from main memory. The PMB.RQ.DMA* signal is disabled when DMA.GR.EN* and PMB.GR.DMA* have both gone active, indicating that the DMA channel has been given control of the PMB bus.

The address counters are incremented and the long word counters decremented when PMB.GR.DMA* goes inactive. At this point in the PMB cycle, the address is no longer driven onto the PMB address bus. On a DMA to PMB write cycle, the CPU does not drive the PMB.RD signal low to indicate a write. Therefore, a signal is needed by the DMC4 to tell it to output a write function code. This signal is DMA.PMB.WR*, a tri-state signal shared by all four channels, but only driven by the channel which is active on the PMB.

The state machine waits for PMB.CYC.DONE*, indicating that the data has been transferred to or from main memory. Finally, the state machine waits for DMA.DATA.RCVD* (or DMA.DATA.SENT*) to go inactive, before going back to the idle state. Once in the idle state, the state machine again waits for DMA.DATA.RCVD* or DMA.DATA.SENT* to go active for the next transfer.

DTB Sequencer State Machine

Each of the four DMA channels on the DMC4 has a DTB Sequencer PAL, DTB SEQ. The DTB SEQ PAL includes two internal state machines and an unrelated latched output. This output, ABORT*, is activated by a DTB master during a SEND INT command if data bit 31 is high. This function allows a DTB master to reset a DMA channel.

The first state machine has three outputs:

1. DMA.LOCK*.
2. DMA.BUSY*.
3. DMA.XFERENB*.

When a DTB master directs a SET BC command to a DMA channel, the DMA long word counter is loaded, and data bits 24 to 31 are latched. Data bits 24 to 27 are the master's board ID and bit 28 is the direction bit (DMA.TO.MEM*). If DTB data bit 28 is low during a SET BC command, DMA.TO.MEM* is set low, indicating a DTB to memory transfer. If DTB data bit 28 is high during a SET BC command, DMA.TO.MEM* is set high, indicating a memory to DTB transfer. The DMA.LOCK* signal goes active to indicate that the DMA channel has been configured and is now reserved for this specific data transfer. Next, a DTB master sends a SET ADDR command to the DMA channel. This not only loads the address counter, but also turns on DMA.BUSY*, which indicates that the DMA channel is enabled, and ready to begin transferring data. A copy of DMA.BUSY* is DMA.XFER.ENB*, which acts exactly the same. All three outputs are deactivated at the same time.

For a PMB to DTB transfer, DMA.BUSY* goes inactive when the DTB master executes a READ LAST command, preventing any more data from being transferred from main memory. For a DTB to PMB transfer DMA.BUSY* is not disabled until a WRITE LAST command is followed by PMB.DMA.DONE* going active. This allows the last word to be transferred to memory before DMA.BUSY* goes inactive. These three outputs are used by the slave control and DMA DTA PALs described in following sections.

The second state machine has only one output: PMB.DMA.DONE*. This output is essentially PMB.CYC.DONE* synchronized with DTB.CLK. This state machine waits for PMB.DMA.ACTV to go active, then inactive for each word transferred. PMB.DMA.ACTV is the opposite phase of CLK.PMB.DATA, and is active during the PMB part of a DMA transfer. When PMB.DMA.ACTV goes inactive at the end of a PMB transfer, PMB.DMA.DONE* is activated for one DTB.CLK period.

Slave Recognizer

The slave PAL (SLV CTL) monitors the DTB slave board ID and function codes, as well as the status of a DMA channel. Each DMA channel has a slave PAL. The DMA.SLV.MATCH* output goes active whenever the DTB slave board ID matches the ID of the corresponding DMA channel.

The DMA.LD.BC* output goes active for a DTB SET BC command when the slave board ID matches, and the DMA channel is not busy or locked. Data bits 0 to 16 are latched as the long word count and bits 23 to 28 are latched control signals. The DMA.LD.ADDR* output goes active for a DTB SET ADDR command when the slave board ID matches, and the DMA channel is not busy. The DMA.LD.ADDR* signal loads the DMA address counters. The RD.STAT* output goes active for a DTB RD STAT command when the slave board ID matches. The output DMA.DTB.XFER is active when DMA.XFER.ENB* is active and DMA.DATA.RCVD* is false for a PMB to DTB transfer or DMA.DATA.SENT* false for a DTB to PMB transfer. The DMA.SLV.ACK* output goes active for the following commands and conditions:

Command	Conditions
SET BC	DMA not busy or locked.
SET ADDR	DMA not busy but locked.
SEND INT	XFER.ENB* false, DMA.DATA.RCVD* false for PMB to DTB or DMA.DATA.SENT* false for DTB to PMB.
RD STAT	Any time.

When DMA.SLV.ACK* goes active, a tri-state buffer is enabled, which drives DTB.SLV.ACK* onto the DTB bus. This signal notifies a DTB master that the requested DTB command can be executed.

DMA Data Flow Controller

Each DMA channel has a DMA data flow controller. This PAL has four outputs which set and clear the DMA.DATA.RCVD* and DMA.DATA.SENT* flip-flops. The separate flip-flops external to the PAL were required because the set-and-hold equations were too complex to fit in the PAL.

The data received flip-flop is set when a slave match occurs, the DMA channel is enabled for transfers, a DTB WRITE DATA or WRITE LAST command is executed, and the data received flip-flop is not currently set. The flip-flop is cleared when PMB.DMA.DONE* occurs for a DTB to memory transfer. The flip-flop is also cleared when the DMA channel is initialized with a SET ADDR command. No data transfers to memory occur until the flip-flop is set.

The data-sent flip-flop is set when a slave match occurs, the DMA channel is enabled for transfers, a DTB READ DATA is executed, and the data sent flip-flop is not currently set. The flip-flop is also set when the DMA channel is initialized with a SET ADDR command. This causes the first word to be read from main memory. The flip-flop is cleared when PMB.DMA.DONE* occurs for a PMB to DTB transfer.

DMA Long Word Counters

Each DMA channel has a 16 bit long word counter. The counter is loaded with the number of long words to be transferred by a DTB master executing the SET BC command. The outputs of the counter are buffered and enabled onto the internal DTB data bus when a READ STAT command has been issued to the channel by the DTB master. The long word count is accessible only by the DTB master.

DMA Address Counters

Each DMA channel has a 24-bit address counter. During a DMA transfer over the PMB bus, all four byte select lines are true since 32-bit words are always transferred. With four byte selects and a 24-bit counter, the effective address is 26 bits, allowing up to 64 Mbytes to be addressed. The counter is loaded with the starting DMA address by a DTB master executing the SET ADDR command. The outputs of the counters are buffered and enabled onto the PMB address bus when a PMB grant has been issued to a DMA channel.

DMA Registers

The following are the DMA registers for each channel that are accessible by DTB masters:

LOAD LONG WORD COUNT

```
 3 3 2|2|2 2 2 2|2 2 2 2 1 1 1 1|1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0
 1 0 9|8|7 6 5 4|3 2 1 0 9 8 7 6|5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0
------+-+-------+---------------+-------------------------------
 X X X|W|~~SLV~~.BID|X X X X X X X X|       LONG WORD COUNT 15 - 0
------+-+-------+---------------+-------------------------------
```

W=0 DTB to PMB transfer, W=1 PMB to DTB transfer
SLV.BID = Slave Board ID 3-0

Master BID

LOAD ADDRESS

```
 3 3 2 2 2 2 2 2|2 2 2 2 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0
 1 0 9 8 7 6 5 4|3 2 1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0
----------------+-----------------------------------------------
 X X X X X X X X|               PMB ADDRESS 25 - 2
----------------+-----------------------------------------------
```

READ STATUS

```
 3 3 2 2 2 2|2|2|2 2 2 2|1 1 1 1|1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0
 1 0 9 8 7 6|5|4|3 2 1 0|9 8 7 6|5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0
------------+-+-+-------+-------+-------------------------------
 X X X X X X|L|U|X X X X|SLV.BID|       LONG WORD COUNT 15 - 0
------------+-+-+-------+-------+-------------------------------
```

L = DMA channel not locked   U = Uncorrectable error has occurred during a transfer on this channel   SLV.BID = Slave Board ID 3-0

SEND INTERRUPT

```
 3|3 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0
 1|0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0
--+-------------------------------------------------------------
 A|X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X
--+-------------------------------------------------------------
```

A = Abort the DMA operation in progress on this channel

DMA Registers

The following are the DMA registers for each channel that are accessible by DTB masters:

LOAD LONG WORD COUNT

```
 3 3 2|2|2 2 2 2|2 2 2 2 1 1 1 1|1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0
 1 0 9|8|7 6 5 4|3 2 1 0 9 8 7 6|5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0
------+-+-------+---------------+-------------------------------
 X X X|W|SLV.BID|X X X X X X X X|      LONG WORD COUNT 15 - 0
------+-+-------+---------------+-------------------------------
W=0 DTB to PMB transfer, W=1 PMB to DTB transfer
SLV.BID = Slave Board ID 3-0
```

Master BID

LOAD ADDRESS

```
 3 3 2 2 2 2 2 2|2 2 2 2 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0
 1 0 9 8 7 6 5 4|3 2 1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0
----------------+-----------------------------------------------
 X X X X X X X X|                  PMB ADDRESS 25 - 2
----------------+-----------------------------------------------
```

reset?? how??

READ STATUS

```
 3 3 2 2 2 2|2|2|2 2 2 2|1 1 1 1|1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0
 1 0 9 8 7 6|5|4|3 2 1 0|9 8 7 6|5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0
------------+-+-+-------+-------+-------------------------------
 X X X X X X|L|U|X X X X|SLV.BID|      LONG WORD COUNT 15 - 0
------------+-+-+-------+-------+-------------------------------
L = DMA channel not locked     U = Uncorrectable error has occurred
during a transfer on this channel    SLV.BID = Slave Board ID 3-0
```

SEND INTERRUPT

```
 3|3 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0
 1|0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0
--+-------------------------------------------------------------
 A|X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X
--+-------------------------------------------------------------
A = Abort the DMA operation in progress on this channel
```

Inter-Processor Communication Bus:

The Inter-Processor Communication Bus (ICB) is one of the three major Arete 1000 system busses.
The ICB is primarily used as a system wide control and status bus that the master processor uses to coordinate the functions of the system. It is also used as a relatively low speed I/O bus used to service character oriented I/O controllers.

The following is a specification of three areas of ICB usage. A functional bus specification covers the bus signals and timing characteristics. A description of the standard ICB hardware interface is discussed. A description of the standard ICB virtual register interface is discussed.

ICB Functional Specification:

The ICB is a relatively low speed asynchronous parallel data transfer bus. It is capable of 8 or 16 bit data transfers throughout a 16 Mbyte address space. It is currently a single master, multiple slave implementation. The ICB supports up to seven non-vectored interrupts. The ICB is derived from the VME Data Transfer Bus. It is roughly equivalent to the VME A24 D16 BTO(20) IH(1-7) I(1-7). The VME Arbiter, Requester, Sequential Access, and Environmental options are not supported.

Signal Description:

The following is a description of ICB Signals. It includes their mnemonics, connector pin numbers, termination values on each end of the bus, and a brief description of their function. An asterisk (*) is used as a signal suffix to indicate negative assertion of that signal. Assertion levels are otherwise active high.

Signal	Pin	Term	Description
ICB.DATA.00	A01	1K-1K	Three State Data Bus
ICB.DATA.01	A02	1K-1K	Three State Data Bus
ICB.DATA.02	A03	1K-1K	Three State Data Bus
ICB.DATA.03	A04	1K-1K	Three State Data Bus
ICB.DATA.04	A05	1K-1K	Three State Data Bus
ICB.DATA.05	A06	1K-1K	Three State Data Bus
ICB.DATA.06	A07	1K-1K	Three State Data Bus
ICB.DATA.07	A08	1K-1K	Three State Data Bus
ICB.DATA.08	C01	1K-1K	Three State Data Bus
ICB.DATA.09	C02	1K-1K	Three State Data Bus
ICB.DATA.10	C03	1K-1K	Three State Data Bus
ICB.DATA.11	C04	1K-1K	Three State Data Bus
ICB.DATA.12	C05	1K-1K	Three State Data Bus
ICB.DATA.13	C06	1K-1K	Three State Data Bus
ICB.DATA.14	C07	1K-1K	Three State Data Bus
ICB.DATA.15	C08	1K-1K	Three State Data Bus
ICB.ADDR.01	A30	1K-1K	Three State Address Bus
ICB.ADDR.02	A29	1K-1K	Three State Address Bus
ICB.ADDR.03	A28	1K-1K	Three State Address Bus
ICB.ADDR.04	A27	1K-1K	Three State Address Bus
ICB.ADDR.05	A26	1K-1K	Three State Address Bus
ICB.ADDR.06	A25	1K-1K	Three State Address Bus
ICB.ADDR.07	A24	1K-1K	Three State Address Bus
ICB.ADDR.08	C30	1K-1K	Three State Address Bus
ICB.ADDR.09	C29	1K-1K	Three State Address Bus
ICB.ADDR.10	C28	1K-1K	Three State Address Bus
ICB.ADDR.11	C27	1K-1K	Three State Address Bus
ICB.ADDR.12	C26	1K-1K	Three State Address Bus
ICB.ADDR.13	C25	1K-1K	Three State Address Bus

ICB.ADDR.14	C24	1K-1K	Three State Address Bus
ICB.ADDR.15	C23	1K-1K	Three State Address Bus
ICB.ADDR.16	C22	1K-1K	Three State Address Bus
ICB.ADDR.17	C21	1K-1K	Three State Address Bus
ICB.ADDR.18	C20	1K-1K	Three State Address Bus
ICB.ADDR.19	C19	1K-1K	Three State Address Bus
ICB.ADDR.20	C18	1K-1K	Three State Address Bus
ICB.ADDR.21	C17	1K-1K	Three State Address Bus
ICB.ADDR.22	C16	1K-1K	Three State Address Bus
ICB.ADDR.23	C15	1K-1K	Three State Address Bus
ICB.IRQ.01*	B30	220-220	Interrupt Request Level 1
ICB.IRQ.02*	B29	220-220	Interrupt Request Level 2
ICB.IRQ.03*	B28	220-220	Interrupt Request Level 3
ICB.IRQ.04*	B27	220-220	Interrupt Request Level 4
ICB.IRQ.05*	B26	220-220	Interrupt Request Level 5
ICB.IRQ.06*	B25	220-220	Interrupt Request Level 6
ICB.IRQ.07*	B24	220-220	Interrupt Request Level 7
IACK.IN*	A21	None	Interrupt Daisy Chain (Currently Not Used)
IACK.OUT*	A22	None	Interrupt Daisy Chain (Currently Not Used)
BG.00.IN*	B04	None	Bus Grant In (Currently Not Used)
BG.01.IN*	B06	None	Bus Grant In (Currently Not Used)
BG.02.IN*	B08	None	Bus Grant In (Currently Not Used)
BG.03.IN*	B10	None	Bus Grant In (Currently Not Used)
BG.00.OUT*	B05	None	Bus Grant Out (Currently Not Used)
BG.01.OUT*	B07	None	Bus Grant Out (Currently Not Used)
BG.02.OUT*	B09	None	Bus Grant Out (Currently Not Used)
BG.03.OUT*	B11	None	Bus Grant Out (Currently Not Used)
ICB.AS*	A18	220-220	Address Strobe
ICB.UDS*	A12	220-220	Upper Data Strobe
ICB.LDS*	A13	220-220	Lower Data Strobe
ICB.RD	A14	220-220	Read/Write*
ICB.DTACK*	A16	220-220	Asynchronous Data Transfer Acknowledge
ICB.BERR*	C11	220-220	Asynchronous Bus Error Indication
ICB.CLK	A10	220-220	ICB Clock
SYS.RESET*	C12	220-220	System Reset
GND	A09	None	Ground
GND	A11	None	Ground
GND	A15	None	Ground
GND	A17	None	Ground
GND	A19	None	Ground
GND	B21	None	Ground
GND	B23	None	Ground
GND	C09	None	Ground
VCC	A32	None	+5 Volts
VCC	B32	None	+5 Volts
VCC	C32	None	+5 Volts
+12V	C31	None	+12 Volts
-12V	A31	None	-12 Volts
RESERVED	A20	None	Unused Reserved
RESERVED	A23	None	Unused Reserved
RESERVED	B01	None	Unused Reserved
RESERVED	B02	None	Unused Reserved
RESERVED	B03	None	Unused Reserved

RESERVED	B12	None	Unused Reserved
RESERVED	B13	None	Unused Reserved
RESERVED	B14	None	Unused Reserved
RESERVED	B15	None	Unused Reserved
RESERVED	B16	None	Unused Reserved
RESERVED	B17	None	Unused Reserved
RESERVED	B18	None	Unused Reserved
RESERVED	B19	None	Unused Reserved
RESERVED	B20	None	Unused Reserved
RESERVED	B22	None	Unused Reserved
RESERVED	B31	None	Unused Reserved
RESERVED	C10	None	Unused Reserved
RESERVED	C13	None	Unused Reserved
RESERVED	C14	None	Unused Reserved

ICB.DATA.00-15: Three state bidirectional data lines providing a sixteen bit data path between Bus Master and Slave.

ICB.ADDR.01-23: Three state address lines driven by the Bus Master specifying a memory address.

ICB.IRQ.01-07*: Open collector lines driven by the interrupter prioritize interrupt requests where ICB.IRQ.07* is highest and ICB.IRQ.01* is lowest.

IACK.IN*: Totem pole driven interrupt priority daisy chain driven by next highest priority devices IACK.OUT*. These lines are currently not used.

IACK.OUT*: Totem pole driven interrupt priority daisy chain driven to the next highest priority devices IACK.IN*. These lines are currently not used.

BG.00-03.IN*: Totem pole driven lines indicate that this board may become the bus master. These lines are currently not used.

BG.00-03.OUT*: Totem pole driven lines indicate that the next board in the daisy chain may become the bus master. These lines are currently not used.

ICB.AS*: Address Strobe is an open collector driven signal that indicates the bus master has a valid address on the bus.

ICB.UDS*: Upper Data Strobe is an open collector driven signal driven by the bus master indicates that the data transfer will occur on the upper byte (ICB.DATA.08-15) of the word.

ICB.LDS*: Lower Data Strobe is an open collector driven signal driven by the bus master indicates that the data transfer will occur on the lower byte (ICB.DATA.00-07) of the word.

ICB.RD: An open collector driven signal that when high specifies that the data transfer is a read from the slave to the bus master. When low this signal indicates a write operation from the bus master to the slave.

ICB.DTACK*: Data Transfer Acknowledge is an open collector driven signal generated by the bus slave. This signal indicates that valid data is available on the bus during a read cycle or that data has been accepted on a write cycle. The bus master will extend the bus cycle until ICB.DTACK* or ICB.BERR* is asserted.

ICB.BERR*: Bus Error is an open collector driven signal generated by the slave. It indicates that an unrecoverable error has occurred during the bus cycle and that it should be aborted. ICB.BERR* may be asserted at the time of or in lieu of ICB.DTACK* to terminate a cycle.

ICB.CLK: A constant square wave clock used for general system timing or syncronizing ICB control signals. This signal is currently the main CPU clock.

SYS.RESET*: An open collector driven signal that when driven will reset the entire system.

GND: GROUND (0 volts).

VCC: VCC (+5 volts DC).

+12V: (+12 volts DC).

-12V: (-12 volts DC).

RESERVED: Unused reserved for extensions.

ICB Operation:

There are two basic bus cycle types supported across the ICB. A Read cycles and write cycles are supported, VME style read-modify-write cycles are currently not supported by either the current CPU or IOCPs. The following paragraphs explain the control signals and bus operation during these cycles.

Data transfer across the ICB involves the address bus (ICB.ADDR.XX), the data bus (ICB.DATA.XX), and control lines (ICB.AS*, ICB.UDS*, ICB.LDS*, ICB.RD, ICB.DTACK*, and ICB.BERR*). The address and data busses are separate parallel busses and the various control signals are used to provide a completely asynchronous bus cycle.

Read Cycle:

During a read cycle the bus master is transferring specific data from the slave. Byte or word transfers are allowed. To initiate a transfer, the master places the valid address on the bus and asserts ICB.RD. It then asserts ICB.AS* to indicate the valid address. It also asserts ICB.LDS* and/or ICB.UDS* to indicate whether the transfer is occurring over the low order and/or high order bytes of the word.

At this point the master waits for the asynchronous data transfer acknowledge (ICB.DTACK*) signal from the slave. This signal indicates that the data is now available on the bus. The master then receives the data and negates the strobe signals (ICB.AS*, ICB.UDS*, and ICB.LDS*). The slave then negates the ICB.DTACK* signal.

If the slave does not assert the ICB.DTACK* signal by the required set-up time, the master will enter wait states. It will continue to insert wait states until the ICB.DTACK* signal is asserted.

In particular unrecoverable error conditions it is possible to terminate a cycle by asserting the ICB.BERR* signal. This signal can be asserted either at the same time as or in lieu of ICB.DTACK*. Terminating a cycle in this manner causes the master to enter error handling routines.

Below is a flowchart outlining the read cycle operation:

```
        Master                                        Slave

---------------------------------
|1) Assert ICB.RD                |
|2) Place ICB.ADDRs on bus       |
|3) Assert ICB.AS*               |         ---------------------------------
|4) Assert ICB.UDS* and ICB.LDS* | ---->  |1) Decode ICB.ADDR                |
---------------------------------         |2) Place read data on ICB.DATA    |
                                          |3) Assert ICB.DTACK*           |
 --------------------------------
|1) Latch data from ICB.DATA     | <---- |                                |
|2) Negate ICB.UDS* and ICB.LDS*|         ---------------------------------
|3) Negate ICB.AS*               |        ---------------------------------
|                                | ----> |1) Remove ICB.DATA              |
 --------------------------------        |2) Negate ICB.DTACK*            |
                                          ---------------------------------
```

Write Cycle:

During a write cycle the bus master is transferring specific data to the slave. Byte or word transfers are allowed. To initiate a transfer, the master places the valid address on the bus and negates ICB.RD. It then asserts ICB.AS* to indicate the valid address. It also asserts ICB.LDS* and/or ICB.UDS* to indicate whether the transfer is occurring over the low order and/or high order bytes of the word.

At this point the master waits for the asynchronous data transfer acknowledge (ICB.DTACK*) signal from the slave. This signal indicates that the slave has now accepted the data from the bus. The master terminates the cycle by negating the strobe signals (ICB.AS*, ICB.UDS*, and ICB.LDS*). The slave then negates the ICB.DTACK* signal.

If the slave does not assert the ICB.DTACK* signal by the required set-up time, the master will enter wait states. It will continue to insert wait states until the ICB.DTACK* signal is asserted.

In particular unrecoverable error conditions it is possible to terminate a cycle by asserting the ICB.BERR* signal. This signal can be asserted either at the same time as or in lieu of ICB.DTACK*. Terminating a cycle in this manner causes the master to enter error handling routines.

Below is a flowchart outlining the write cycle operation:

```
        Master                                        Slave

 --------------------------------
|1) Negate ICB.RD                |
|2) Place  ICB.ADDRs on bus      |
|3) Assert ICB.AS*               |        ---------------------------------
|4) Place  ICB.DATA on bus       |       |                                |
|5) Assert ICB.UDS* and ICB.LDS*| ----> |1) Decode ICB.ADDR              |
 --------------------------------        |2) Accept data from ICB.DATA    |
 --------------------------------        |3) Assert ICB.DTACK*            |
|                                | <---- |                                |
|1) Negate ICB.UDS* and ICB.LDS*|         ---------------------------------
|2) Negate ICB.AS*               |        ---------------------------------
|3) Remove ICB.DATA              | ----> |1) Negate ICB.DTACK*            |
 --------------------------------        |                                |
                                          ---------------------------------
```

ICB Signal Timing:

The following tables specify timing for ICB cycle in both Read and Write operation. The theoretical "worst case" timing is presented ,as well as typical "as measured" timing. Worst case timing should be assumed when interfacing to the ICB.

Timing:

Parameter	Theoretical	Actual
Address Setup Time	29	54
Address Hold Time	26	108
Data Setup Time (R)	-	-
Data Setup Time (W)	~~23~~	~~20~~
Data Hold Time (R)	-	-
Data Hold Time (W)	79	80
RD Setup Time	-	50

RD Hold Time	-	66
AS assert to DS assert (R)	6	8
AS assert to DS assert (W)	74	72
AS negate to DS negate (R)	7	5
AS negate to DS negate (W)	7	5
DTACK, BERR Setup Time	-	-
DTACK, BERR Hold Time	-	-
Clock Period	80	80

ICB Interface Registers:

The ICB supports a standard hardware and software interface scheme to allow standard methods of configuring and initializing all cards in the system. The interface includes capabilities for resetting, interrupting, halting devices, Test and Set, NMI, Board Ready, Board identification, and general status and control.

There are different classes of ICB hardware interfaces. Level one is a basic interface that provides Board Type register, 12 bit status and control registers, and reset capability. This is the basis for extended ICB interfaces. The basic ICB interface is a minimal hardware implementation that provides a five bit Board Type identification register that allows a board to be uniquely identified. It also provides 12 bit general purpose control and status registers. It also provides a standard method for resetting a board. Every board has a minimum of these features.

There are additional levels of extended ICB hardware interfaces, each providing greater functionality. In addition to the level one interface, the complete interface includes two way control or status with handshake, 64 Kbyte dual ported memory interface, full interrupt capabilities, standard methods for resetting and halting devices, and test and set.

The level one interface is intended as a minimal ICB implementation to be used on Non-intelligent or Non-IOCP cards. The extended interfaces are intended to offer various levels of design flexibility depending on the requirements.

Each card occupies 64 Kbytes of address space on the ICB. The top two words of that address space are dedicated to hardware registers. Those are byte or word addressable registers that can be written and read by the ICB Master. The two control words at the top of ICB address space are located at hex addresses XXFFFC and XXFFFE. Both locations may be read and written, thus four bytes of control and four bytes of status exist. The layout of those four status bytes are as illustrated in figure 1. The following is a summary of those functions.

Master Writing to 0xXXFFFE:

ICB Command - (Bit 0-11) - A twelve bit command register used for passing commands or control to the bus slave. If the slave supports the Command/Status Interrupt Extension a slave interrupt will be generated when this Command register is written. That interrupt will be cleared when the slave reads the ICB Command.

HALT - (Bit 12) - Active high halt signal when set causes board to stop processing. Board will remain halted until the HALT bit is negated.

INT - (Bit 13) - Active high interrupt when set causes the board to enter ICB interrupt handling. INT is cleared by

NMI - (Bit 14) - Active high non-maskable interrupt when set causes the board to enter NMI handling. NMI is cleared by

RESET - (Bit 15) - Active high board reset when written causes a one shot to reset that board only. A reset is caused only on the low to high transition of RESET.

Master Writing to 0xXXFFFC:

ND - (Bit 0-15) - Not Defined Reserved.

Master Reading from 0xXXFFFE:

ICB Status - (Bit 0-11) - A twelve bit status register used for passing commands or control to the bus master. If the slave supports the Command/Status Interrupt Extension, the master will be interrupted when the slave writes this register. That interrupt will be cleared when the slave reads the ICB Command register.

ND - (Bit 12) - Not Defined Reserved.

INTRQ - (Bit 13-15) - Interrupt Request status contains the interrupt level that the slave is currently using.

Master Reading from 0xXXFFFC:

ICB Extensions:

If the board supports ICB extensions beyond level one this register indicates which extensions are provided. These bits indicate whether the board supports complete interrupt capabilities, complete 64K dual ported memory, virtual register interface, and ICB bus master capability. Bits 0-4 are used to identify these extensions.

INT - (Bit 0) - The INT Bit indicates that the board supports complete ICB interrupt capabilities. This means that the slave can interrupt the master and the master can interrupt the slave.

DPM - (Bit 1) - The DPM Bit indicates that the board supports a dual ported memory interface across the ICB. This is a 64Kb shared memory "window" between the board and the bus master. This extensions also includes hardware Test and Set logic to control ownership of the dual ported memory.

VREG - (Bit 2) - The VREG Bit indicates that the board supports a standard virtual register interface.

BM - (Bit 3) - The BM Bit indicates that the board is capable of becoming an ICB bus master.

ND - (Bit 4) - Not Defined Reserved.

ICB Ready - (Bit 5) - The ICB Ready Bit indicates that the board has completed self test, mapped out EPROM, and has set up its virtual registers.

ND - (Bit 6) - Not Defined Reserved.

TAS - (Bit 7) - The Test and Set bit (TAS) indicates the "ownership" of the shared dual ported memory of an extended ICB interface.

Board Type - (Bit 8-12) - A five bit Board Type identification register allows the unique identification of the board type in that card slot.

Extended Interface - (Bit 13) - Extended Interface bit indicates whether the board supports any extensions beyond the level one ICB interface. If the Extended Interface bit is true the actual extensions are indicated in the ICB Extension bits.

SIP - (Bit 14) - Slave Interrupt Pending

MIP - (Bit 15) - Master Interrupt Pending