Arix Corporation

System 90 - Memory Module

Functional Description

20 January 1989

1. INTRODUCTION

This document represents the functional specification for the Memory Module.

2. SCOPE

This functional specification includes the physical and electrical specifications, system compatibility, and a functional description of the Memory Module.

3. PHYSICAL SPECIFICATIONS

3.1 Physical Size

Height	14.437 inches
Length	15.750 inches
Width	less than .8 inches

3.2 Temperature Specifications

See Cooling Requirements.

3.3 Humidity Specifications

Operating	20% to 80% Non-condensing
Non-Operating	5% to 95% Non-condensing

3.4 Altitude Specifications

Operating	Sea Level to 8,000 feet @ 40 degrees C
Non-Operating	Sea Level to 40,000 feet

3.5 Cooling Requirements

The cooling requirement for the Memory Module is 12 cfm at 40 degrees C ambient temperature.

3.6 Physical Restrictions

There are no physical restrictions.

4. POWER REQUIREMENTS

The DC power requirements for the Memory Module are as follows: 14 Amps @ 5V; 70 watts.

5. SYSTEM COMPATIBILITY

The Memory Module can be used only on the Computational Subsystem Bus (CSS) in a System 90 machine.

6. FUNCTIONAL DESCRIPTION

6.1 Memory Module System Bus Interface

Command, address and data if applicable get clocked into the bus input registers every 50 nsec by the 20MHz system clock. The slot identifier field is compared with the memory module's slot number to see if the command is meant for this memory module. If there is a match, the top four bits of the address are checked. An access of the register section of the board is indicated by address bit 43 active high. Some of bits 42 through 40 and 57 are compared to zero for qualifying a memory access. The number of bits that must equal zero depends on the size of the board. If the transaction is not meant for this memory module, no action will be taken and a new command, address, and data will be clocked into the bus input register on the next rising edge of the system clock. Array accesses when the input pipeline register for the arrays is full and accesses to nonexistent status or control registers will get a NACK response from the memory module. The input register may not be used as a holding register for the memory module because it always gets new data clocked into it on every system clock rising edge.

6.2 Status And Control Registers

There are several status and control registers on the memory module. Each bit that can be written can be read. See the Memory Module Memory map in appendix 1 for details.

6.3 Memory Array Accesses

When a valid memory access gets clocked into the input registers, the address, and data if applicable will get clocked into the input pipeline register on the next system clock's rising edge. The input pipeline register holds commands until the arrays are ready to process them.

6.4 Input Pipeline Register

The pipeline register are two units wide and two units deep. The units here are 32 bits of address, 32 bits of data if applicable, and a few appropriate control bits. This register is implemented with nine 29520 pipeline registers. The 29520s are always used as a single four stage pipeline. Clocking data into the 29520 is done on the rising edge of the 20 MHz system clock, using the I1 and I0 control signals as shown in the following table:

center; c c l. I1 I0 ACTION _ 0 0 LOAD 0 1 NOT USED
1 0 NOT USED 1 1 HOLD

Whenever new data is latched into the 29520s, the data is latched into register A1. The previous content of A1 goes to A2, the content of A2 goes to B1, the content of B1 to B2. If B2 is already full, i.e. it contains a command (read or write) which has not yet been executed, the load will not be done, and the attempted command will be NACKed on the CSS bus. The memory board will be not ready in this condition. The outputs of these registers feed a four to one multiplexer which is controlled by the signals S1 and S0. One of the four registers is always being selected from the multiplexer. The encodings are as follows:

center; c c l. S1 S0 OUTPUT _ 0 0 B2 0 1 B1
1 0 A2 1 1 A1

The default when no outputs are being used is 00. To differentiate the default code from the READ.B2 code, the control signal READ.B2* must be tested. The output enable of the pipeline register is always active so that it always drives the bus.

6.5 Input Pipeline Register Control

The input pipeline register is controlled by four PALs.

The LDFL PAL generates the I0 and I1 signals used to steer input data to one, the other, or neither halves of the register. Four signals, one for each register, are also generated to tell if the register has any data presently in it. These signals are A1.FULL, A2.FULL, B1.FULL, and B2.FULL. The I0 and I1 signals can also be called LD.A1* and LD.B1*, respectively. The two signals are always asserted together and when neither is active the register is in the HOLD state.

The RDAR PAL keeps track of which registers contain read commands.

The READ PAL generates the S0 and S1 signals used to select the correct register for the pipeline register to output. Also generated are LATCH.ARRAY.0* and LATCH.ARRAY.1*, which latch the input registers to the 0 and 1 arrays. For a write, either the 0 input register or the 1 input register is latched, depending on address bit 02. Both 0 and 1 input registers are enabled on a read, so that the two arrays are synchronized for the read cycle.

6.6 Array Input Latches

The output of the pipeline registers feed the 74ALS374 array address and data latches. The outputs of the data latches go directly to the DRAM chips data inputs. The outputs of the array input address latches get multiplexed from twenty row and column address

lines to ten multiplexed address lines. Multiplexing is done with 74F241s, by connecting a row and a corresponding column address to each input having complementary output enables with their tri-state outputs tied together.

6.7 Array Organization

Each array contains four banks of dynamic RAM. The banks are 39 bits wide and 1 megabyte deep. The 39 bits consist of 32 bits of data and 7 check bits. Depopulation of the memory module is achieved by removing one or three banks from each array. In this way, two way interleaving is preserved with any stuffing option.

6.8 Array Control

Each of the two interleaved arrays has five control PALs for generating DRAM control signals. The two arrays can therefore operate independently on all cycles except a read.

The RAS PAL generates RAS and address multiplexer select for each of the four banks in the array.

The MERGE PAL generates the output enables for the array data input registers and the 632 data latches.

The EDAC PAL generates the control signals for the 632 EDAC chip, the error interrupt information latching, and the check bit registers.

The CASWE PAL generates the write enable signals for each bank of the memory array and the CAS signal for eack bank

The MEMCTL PAL contains a state counter and outputs signals to tell whether the cycle is a read, write, refresh, or partial write.

6.9 Array Outputs

The output of the DRAMs in the array feed a 74ALS244 whose outputs are shared with the bidirectional data and check bit pins of a 74AS632 EDAC chip, the input of another pipeline register used for output read data, the output of the array input data register, and the input to the DRAM array.

6.10 Output Pipeline Register

The pipeline register at the output of each array is organized as 32 bits wide and four deep. It is always used simultaneously with the output pipeline register of the other array. This is because the smallest read unit is 64 bits.

Control for this 64 bit wide register is handled by the OUTCTL PAL. Because of the simple organization of this pipeline register only three signals are needed to control it. The

LOAD* signal outputs from the PAL and connects to both the I0 and I1 inputs of the 29520. When active, the registers are loaded like a FIFO, when inactive, the register is in the hold state. The multiplexor signals, S0 and S1, are used as in the input pipeline register, except that they always start at 00 and increment to output the number of 8 byte words in the register.

6.11 Read Output

The output of the pipeline register goes into a 74F374 register for synchronization with the system clock before being output through a 74F241 and onto the system bus. On a read cycle, the memory module makes a request one clock before it loads the last 8 bytes of read data into the pipeline register. The GRANT from the arbiter goes to the J input of a 74F109 J-K flip-flop whose output is the output enable for the 74F241 output drivers. The K input of this 74F109, RD.DONE*, comes from the OUTCTL PAL, and becomes active after the last read data bytes are clocked into the output register. In this way, the memory module drives the bus for 1, 2, or 4 bus clocks, depending on whether it is an 8, 16, or 32 byte read. The clear of the 74F109 is connected to the BOARD.DISABLE line. With the 74F109 always cleared, the memory module never drives the system bus. If the GRANT from the arbiter comes immediately, the data can be clocked into the data output register during the same clock period that it is being driven onto the bus.

7. Cycle Descriptions

The memory module supports six types of memory operations, including three different types of read cycles. All memory cycles consist of a certain number of states, numbered S0, S1, S2, and so on. Each state is one system clock period, or 50 nsec. The memory cycles can be in any sequence and may start immediately after the preceding operation's last state. The types of memory operations with brief descriptions are as follows.

7.1 Write Cycle

The write cycle is of the late-write type. This means that while writing, the data output of the DRAM may be driving. This is acceptable because the tri-state drivers on the outputs of the DRAM array are turned off. The write cycle lasts five clock periods and will occur only if the LENGTH field of the bus type is equal to four bytes.

The write cycle begins the assertion of RAS in state S1. CAS is asserted in S2 and WE in S3. All three signals are de-asserted in S4. Write data and check bits are driven from S1-1/2 until S4-1/2. The signals EDAC.S1 and EDAC.S0 are both inactive for the duration of the cycle, putting the 74AS632 EDAC chip in the check bit generate mode. The check bits generated are valid about halfway through S2.

7.2 Read Cycle

The normal read cycle returns 8 bytes of data to the requester. The LENGTH field must equal 8 or less bytes for this this type of read. To simplify burst mode reads, the read data

is corrected whether or not an error has been detected. Therefore, most of the time correct data is being automatically corrected again. This procedure adds and extra clock period to the read cycle but it is made up for in control logic simplification.

The read cycle begins with RAS asserted in state S1. CAS is asserted in S2 and both signals are de-asserted in S4. Read data is valid just before S4. The EDAC chip control signals start with EDAC.S1 and EDAC.S0 equal to zero. In S2, EDAC.S1 is activated to put the EDAC chip into the detection mode, and EDAC.S0 is activated in S4 for the correction mode. The rising edge of EDAC.S0 latches the read data from the DRAMs. Both signals are deactivated after S5. The corrected data is driven by the 74AS632 EDAC chip starting in S5 and is valid about halfway through S5. The output pipeline register latches the corrected data on the next rising edge after S5.

7.3 16 Byte Read Cycle

The 16 byte read begins the same as a normal read cycle, but it does a nibble mode or page mode access when the 8 byte read would have completed. This type of read lasts 8 clock periods. When the cycle is a read and the LENGTH field is six (16 bytes), a 16 byte read will be done.

The read cycle begins with RAS asserted in state S1. CAS is asserted in S2 and only CAS is de-asserted in S4. Read data is valid just before S4. The EDAC chip control signals start with EDAC.S1 and EDAC.S0 equal to zero. In S2, EDAC.S1 is activated to put the EDAC chip into the detection mode, and EDAC.S0 is activated in S4 for the correction mode. The rising edge of EDAC.S0 latches the read data from the DRAMs. EDAC.S0 is deactivated after the rising edge of S6. The corrected data is driven by the 74AS632 EDAC chip starting in S5 and is valid about halfway through S5. The output pipeline register latches the corrected data on the rising edge of S6. The nibble mode or page mode access begins with CAS being asserted again in S5. Read data is valid just before S7. The EDAC chip control signal EDAC.S0 is activated in S7 and its rising edge latches the read data from the DRAMs. RAS and CAS are deactivated in S7. EDAC.S0 and EDAC.S1 are deactivated on the next rising edge after S8. The corrected data is driven by the 74AS632 EDAC chip starting in S8 and is valid about halfway through S8. The output pipeline register latches the corrected data on the next rising edge after S8.

7.4 32 Byte Read Cycle

The 32 byte read begins the same as a normal read cycle, but it does three nibble mode or page mode accesses when the 8 byte read would have completed. This read cycle takes 14 clock cycles to complete. When the cycle is a read and the LENGTH field is seven (32 bytes), the 32 byte read will occur.

The read cycle begins with RAS asserted in state S1. CAS is asserted in S2 and only CAS is de-asserted in S4. Read data is valid just before S4. The EDAC chip control signals start with EDAC.S1 and EDAC.S0 equal to zero. In S2, EDAC.S1 is activated to put the

EDAC chip into the detection mode, and EDAC.S0 is activated in S4 for the correction mode. The rising edge of EDAC.S0 latches the read data from the DRAMs. EDAC.S0 is deactivated after the rising edge of S6. The corrected data is driven by the 74AS632 EDAC chip starting in S5 and is valid about halfway through S5. The output pipeline register latches the corrected data on the rising edge of S6. The first nibble mode or page mode access begins with CAS being asserted again in S5. Read data is valid just before S7. The EDAC chip control signal EDAC.S0 is activated in S7 and its rising edge latches the read data from the DRAMs. CAS is deactivated in S7 and EDAC.S0 is deactivated on the rising edge of S9. The corrected data is driven by the 74AS632 EDAC chip starting in S8 and is valid about halfway through S8. The output pipeline register latches the corrected data on the rising edge of S9. The second nibble mode or page mode access begins with CAS being asserted again in S8. Read data is valid just before S10. The EDAC chip control signal EDAC.S0 is activated in S10 and its rising edge latches the read data from the DRAMs. CAS is deactivated in S10 and EDAC.S0 is deactivated on the rising edge of S12. The corrected data is driven by the 74AS632 EDAC chip starting in S11 and is valid about halfway through S8. The output pipeline register latches the corrected data on the rising edge of S9. The third nibble mode or page mode access begins with CAS being asserted again in S11. Read data is valid just before S13. The EDAC chip control signal EDAC.S0 is activated in S13 and its rising edge latches the read data from the DRAMs. RAS and CAS are deactivated in S13. EDAC.S0 and EDAC.S1 are deactivated on the next rising edge after S14. The corrected data is driven by the 74AS632 EDAC chip starting in S14 and is valid about halfway through S8. The output pipeline register latches the corrected data on the next rising edge after S14.

7.5 Partial Write Cycle

The partial write cycle is like a four byte read followed by a write. When a write command is clocked onto the memory module, the LENGTH field is checked to determine if a partial write will be needed. If the LENGTH is 1,2, or 3, the lower two address bits and the LENGTH field are used to generate byte selects for the bytes that will be changed in the partial write cycle. The new data bytes are merged with corrected data read from the location to form the new 4 byte data word which is then written. The data that is read is corrected whether or not there is an error, just like a read cycle. If there is an uncorrectable error, the cycle continues, and the possibly incorrect data bytes are written back into the location. The next read to this location will produce an uncorrectable error if one of the bad bytes was rewritten. The partial write cycle lasts nine clock periods.

The partial write cycle begins with RAS asserted in state S1. CAS is asserted in S2 and both signals are de-asserted in S8. Read data is valid just before S4. The EDAC chip control signals start with EDAC.S1 and EDAC.S0 equal to zero. In S2, EDAC.S1 is activated to put the EDAC chip into the detection mode, and EDAC.S0 is activated in S4 for the correction mode. The rising edge of EDAC.S0 latches the read data from the DRAMs. EDAC.S0 and EDAC.S1 are deactivated on the rising edge of S6. The corrected data is latched in the output latches of the EDAC chip by the rising edge of

LEDBO, which occurs after the rising edge of S6. The data bytes that are not being changed are then driven by the 74AS632 in S6, and the new bytes to be written are driven by the array input latches. Check bits are generated on the merged data and WE strobes for one clock period during S7. The various data output enables and LEDBO are turned off in S8, preparing the array for another cycle.

7.6 Refresh Cycle

The memory module uses CAS before RAS refresh to save address counter logic and extra address bus drivers. In the CAS before RAS refresh mode, an internal counter in the DRAM increments with each refresh operation, insuring that all 512 rows will be refreshed automatically without having to externally drive the address lines. The 512 rows need to be refreshed each 8 msec, so refresh cycles must occur every 15.625 usec. A counter clocked by the 20MHz system clock counts to 312, producing a refresh request every 15.6 usec.

The refresh cycle is initiated by the refresh request, which is the latched carry output of a counter. If the array is idle, the refresh cycle will start on the next rising edge of the system clock. If there is another cycle in progress, the refresh cycle will not start until the present cycle has completed. If there is a refresh request active while another cycle is going, the array will not activate the READY signal until after the refresh cycle to avoid read synchronization problems. The refresh request gets cleared by S2 of the refresh cycle.

The refresh cycle is six clock periods long. It begins with the activation of CAS in S1, and then RAS is asserted in S2. CAS is then de-asserted at S3, and RAS is deactivated during S5. The array is ready to begin a new cycle after S6.

7.7 Error Handling

Single bit or uncorrectable errors may be detected on a read or a partial write. Single bit errors are corrected and an interrupt is generated so that a CPU may keep track of corrected errors. The CPU that handles the interrupt may read the syndrome, which points out the bit in error, the address that the error occurred at, and the slot i.d. of the module that was performing the read. Double bit errors will always be detected but can not be corrected. Multiple bit errors may be detected but will not be corrected. Reading the error information kept for an error clears the registers that hold this information. Information on additional errors which occur before these registers are cleared is lost, that is, no new error information will be latched until the old information is read.

Single bit errors do not affect the operation of a read or partial write cycle, because data correction takes place automatically. On a read cycle, an uncorrectable error forces the error data response to be returned with the bad data. An interrupt is generated and the same error information is available as with a single bit error, except that the syndrome now only states that a multiple bit error has occurred. On a partial write, the data bytes are written even if there is an uncorrectable error. It is possible to write bad data back

into the location. This will not be detected until the next read to this location, when an uncorrectable error message will be generated if the bits in error were written back. Single bit or uncorrectable errors will not cause a 16 byte or 32 byte read to abort before completion and only the error information for the first error encountered will be kept.

7.8 Diagnostics

Diagnostics will do write-read tests with various patterns to find stuck data bits, shorted data bits, stuck address bits, shorted address bits, and data bits shorted to address bits. The address and the bit in error will be provided to the diagnostic user. Partial writes will also be used in place of writes to test the byte merging logic. The EDAC operation will be tested to see if the 74AS632 EDAC chip can generate correct check bits on a write and partial write, detect single bit and multiple bit errors, correct single bit errors, generate an interrupt on a single bit error, and return an uncorrectable error message on a multiple bit error. Read and write loops will also be available to the diagnostic user. All the control and status registers will be accessible and testable by the user.

8. Appendix A

MEMORY MODULE ECC SYNDROME TO BIT MAP

The following is a map of the ECC syndrome bits to bits-in-error for the Memory Module. During a read or read modify write cycle with an error, the syndrome register on the Memory Module latches a value that indicates which memory bits are in error. Double bit error indicates that a two bit uncorrectable error has been detected. Multiple bit errors are also uncorrectable.

ARRAY 0

center; 1 1. Syndrome Bit(s) in error _ xx 00 Multiple bit error (uncorrectable) xx 01 Double bit error (uncorrectable) xx 02 Double bit error (uncorrectable) xx 03 Multiple bit error (uncorrectable) xx 04 Double bit error (uncorrectable) xx 05 Multiple bit error (uncorrectable) xx 06 Multiple bit error (uncorrectable) xx 07 Double bit error (uncorrectable) xx 08 Double bit error (uncorrectable) xx 09 Multiple bit error (uncorrectable) xx 0a Data bit 07 xx 0b Double bit error (uncorrectable) xx 0c Multiple bit error (uncorrectable) xx 0d Double bit error (uncorrectable) xx 0e Double bit error (uncorrectable) xx 0f Data bit 06 xx 10 Double bit error (uncorrectable) xx 11 Multiple bit error (uncorrectable) xx 12 Data bit 05 xx 13 Double bit error (uncorrectable) xx 14 Data bit 04 xx 15 Double bit error (uncorrectable) xx 16 Double bit error (uncorrectable) xx 17 Data bit 03 xx 18 Data bit 02 xx 19 Double bit error (uncorrectable) xx 1a Double bit error (uncorrectable) xx 1b Data bit 01 xx 1c Double bit error (uncorrectable) xx 1d Data bit 00

ARRAY 0 (Continued)

center; l l. Syndrome Bit(s) in error _ xx 1e Multiple bit error (uncorrectable) xx 1f Double bit error (uncorrectable) xx 20 Double bit error (uncorrectable) xx 21 Multiple bit error (uncorrectable) xx 22 Data bit 37 xx 23 Double bit error (uncorrectable) xx 24 Data bit 36 xx 25 Double bit error (uncorrectable) xx 26 Double bit error (uncorrectable) xx 27 Data bit 35 xx 28 Data bit 34 xx 29 Double bit error (uncorrectable) xx 2a Double bit error (uncorrectable) xx 2b Data bit 33 xx 2c Double bit error (uncorrectable) xx 2d Data bit 32 xx 2e Multiple bit error (uncorrectable) xx 2f Double bit error (uncorrectable) xx 30 Data bit 30 xx 31 Double bit error (uncorrectable) xx 32 Double bit error (uncorrectable) xx 33 Multiple bit error (uncorrectable) xx 34 Double bit error (uncorrectable) xx 35 Data bit 31 xx 36 Multiple bit error (uncorrectable) xx 37 Double bit error (uncorrectable) xx 38 Double bit error (uncorrectable) xx 39 Multiple bit error (uncorrectable) xx 3a Multiple bit error (uncorrectable) xx 3b Double bit error (uncorrectable) xx 3c Multiple bit error (uncorrectable) xx 3d Double bit error (uncorrectable) xx 3e Double bit error (uncorrectable) xx 3f Check bit 6 xx 40 Double bit error (uncorrectable) xx 41 Multiple bit error (uncorrectable) xx 42 Multiple bit error (uncorrectable) xx 43 Double bit error (uncorrectable) xx 44 Multiple bit error (uncorrectable)

ARRAY 0 (Continued)

center; l l. Syndrome Bit(s) in error _ xx 45 Double bit error (uncorrectable) xx 46 Double bit error (uncorrectable) xx 47 Multiple bit error (uncorrectable) xx 48 Multiple bit error (uncorrectable) xx 49 Double bit error (uncorrectable) xx 4a Double bit error (uncorrectable) xx 4b Data bit 27 xx 4c Double bit error (uncorrectable) xx 4d Multiple bit error (uncorrectable) xx 4e Data bit 26 xx 4f Double bit error (uncorrectable) xx 50 Multiple bit error (uncorrectable) xx 51 Double bit error (uncorrectable) xx 52 Double bit error (uncorrectable) xx 53 Data bit 25 xx 54 Double bit error (uncorrectable) xx 55 Data bit 24 xx 56 Data bit 23 xx 57 Double bit error (uncorrectable) xx 58 Double bit error (uncorrectable) xx 59 Data bit 22 xx 5a Data bit 21 xx 5b Double bit error (uncorrectable) xx 5c Data bit 20 xx 5d Double bit error (uncorrectable) xx 5e Double bit error (uncorrectable) xx 5f Check bit 5 xx 60 Multiple bit error (uncorrectable) xx 61 Double bit error (uncorrectable) xx 62 Double bit error (uncorrectable) xx 63 Data bit 17 xx 64 Double bit error (uncorrectable) xx 65 Data bit 16 xx 66 Data bit 15 xx 67 Double bit error (uncorrectable) xx 68 Double bit error (uncorrectable) xx 69 Data bit 14 xx 6a Data bit 13 xx 6b Double bit error (uncorrectable)

ARRAY 0 (Continued)

center; 1 1. Syndrome Bit(s) in error _ xx 6c Data bit 12 xx 6d Double bit error (uncorrectable) xx 6e Double bit error (uncorrectable) xx 6f Check bit 4 xx 70 Double bit error (uncorrectable) xx 71 Data bit 10 xx 72 Multiple bit error (uncorrectable) xx 73 Double bit error (uncorrectable) xx 74 Data bit 11 xx 75 Double bit error (uncorrectable) xx 76 Double bit error (uncorrectable) xx 77 Check bit 3 xx 78 Multiple bit error (uncorrectable) xx 79 Double bit error (uncorrectable) xx 7a Double bit error (uncorrectable) xx 7b Check bit 2 xx 7c Double bit error (uncorrectable) xx 7d Check bit 1 xx 7e Check bit 0 xx 7f No errors detected

ARRAY 1

center;-1 1. Syndrome Bit(s) in error _ xx 00 Multiple bit error (uncorrectable) xx 01 Double bit error (uncorrectable) xx 02 Double bit error (uncorrectable) xx 03 Multiple bit error (uncorrectable) xx 04 Double bit error (uncorrectable) xx 05 Multiple bit error (uncorrectable) xx 06 Multiple bit error (uncorrectable) xx 07 Double bit error (uncorrectable) xx 08 Double bit error (uncorrectable) xx 09 Multiple bit error (uncorrectable) xx 0a Data bit 47 xx 0b Double bit error (uncorrectable) xx 0c Multiple bit error (uncorrectable) xx 0d Double bit error (uncorrectable)

ARRAY 1 (Continued)

center; 1 l. Syndrome Bit(s) in error _ xx 0e Double bit error (uncorrectable) xx 0f Data bit 46 xx 10 Double bit error (uncorrectable) xx 11 Multiple bit error (uncorrectable) xx 12 Data bit 45 xx 13 Double bit error (uncorrectable) xx 14 Data bit 44 xx 15 Double bit error (uncorrectable) xx 16 Double bit error (uncorrectable) xx 17 Data bit 43 xx 18 Data bit 42 xx 19 Double bit error (uncorrectable) xx 1a Double bit error (uncorrectable) xx 1b Data bit 41 xx 1c Double bit error (uncorrectable) xx 1d Data bit 40 xx 1e Multiple bit error (uncorrectable) xx 1f Double bit error (uncorrectable) xx 20 Double bit error (uncorrectable) xx 21 Multiple bit error (uncorrectable) xx 22 Data bit 77 xx 23 Double bit error (uncorrectable) xx 24 Data bit 76 xx 25 Double bit error (uncorrectable) xx 26 Double bit error (uncorrectable) xx 27 Data bit 75 xx 28 Data bit 74 xx 29 Double bit error (uncorrectable) xx 2a Double bit error (uncorrectable) xx 2b Data bit 73 xx 2c Double bit error (uncorrectable) xx 2d Data bit 72 xx 2e Multiple bit error (uncorrectable) xx 2f Double bit error (uncorrectable) xx 30 Data bit 70 xx 31 Double bit error (uncorrectable) xx 32 Double bit error (uncorrectable) xx 33 Multiple bit error (uncorrectable)

ARRAY 1 (Continued)

center; 1 1. Syndrome Bit(s) in error _ xx 34 Double bit error (uncorrectable) xx 35 Data bit 71 xx 36 Multiple bit error (uncorrectable) xx 37 Double bit error (uncorrectable) xx 38 Double bit error (uncorrectable) xx 39 Multiple bit error (uncorrectable) xx 3a Multiple bit error (uncorrectable) xx 3b Double bit error (uncorrectable) xx 3c Multiple bit error (uncorrectable) xx 3d Double bit error (uncorrectable) xx 3e Double bit error (uncorrectable) xx 3f Check bit 6 xx 40 Double bit error (uncorrectable) xx 41 Multiple bit error (uncorrectable) xx 42 Multiple bit error (uncorrectable) xx 43 Double bit error (uncorrectable) xx 44 Multiple bit error (uncorrectable) xx 45 Double bit error (uncorrectable) xx 46 Double bit error (uncorrectable) xx 47 Multiple bit error (uncorrectable) xx 48 Multiple bit error (uncorrectable) xx 49 Double bit error (uncorrectable) xx 4a Double bit error (uncorrectable) xx 4b Data bit 67 xx 4c Double bit error (uncorrectable) xx 4d Multiple bit error (uncorrectable) xx 4e Data bit 66 xx 4f Double bit error (uncorrectable) xx 50 Multiple bit error (uncorrectable) xx 51 Double bit error (uncorrectable) xx 52 Double bit error (uncorrectable) xx 53 Data bit 65 xx 54 Double bit error (uncorrectable) xx 55 Data bit 64 xx 56 Data bit 63 xx 57 Double bit error (uncorrectable) xx 58 Double bit error (uncorrectable) xx 59 Data bit 62 xx 5a Data bit 61 xx 5b Double bit error (uncorrectable)

ARRAY 1 (Continued)

center; l l. Syndrome Bit(s) in error _ xx 5c Data bit 60 xx 5d Double bit error (uncorrectable) xx 5e Double bit error (uncorrectable) xx 5f Check bit 5 xx 60 Multiple bit error (uncorrectable) xx 61 Double bit error (uncorrectable) xx 62 Double bit error (uncorrectable) xx 63 Data bit 57 xx 64 Double bit error (uncorrectable) xx 65 Data bit 56 xx 66 Data bit 55 xx 67 Double bit error (uncorrectable) xx 68 Double bit error (uncorrectable) xx 69 Data bit 54 xx 6a Data bit 53 xx 6b Double bit error (uncorrectable) xx 6c Data bit 52 xx 6d Double bit error (uncorrectable) xx 6e Double bit error (uncorrectable) xx 6f Check bit 4 xx 70 Double bit error (uncorrectable) xx 71 Data bit 50 xx 72 Multiple bit error (uncorrectable) xx 73 Double bit error (uncorrectable) xx 74 Data bit 51 xx 75 Double bit error (uncorrectable) xx 76 Double bit error (uncorrectable) xx 77 Check bit 3 xx 78 Multiple bit error (uncorrectable) xx 79 Double bit error (uncorrectable) xx 7a Double bit error (uncorrectable) xx 7b Check bit 2 xx 7c Double bit error (uncorrectable) xx 7d Check bit 1 xx 7e Check bit 0 xx 7f No errors detected

9. Appendix B

MEMORY MODULE MEMORY MAP

REGISTER NAMES

center; l. l. FFFF FFFC status register FFFF FFF4 error address register FFFF FFEC error information register FFFF FFE4 control register FFFF FFC4 interrupt information register

REGISTER FIELDS

llllllll. FFFF FFFC X lcbw<6:0> XXXX 4Mb ms<1:0> board.id<7:0> FFFF FFF4 – XXXX
err.addr<27:24> err.addr<23:16> err.addr<15:08> err.addr<07:00> FFFF FFEC 0sb* 0.syn<6:0> 0mb* 0.info<6:0> 1sb* 1.syn<6:0> 1mb* 1.info<6:0> FFFF FFE4 X control<6:0> X X X X X X X X X X X X X X X X X
diag.cb<6:0> FFFF FFC4 int.vect<7:0> X
int.prio<6:0> id<3:0>ia<11:08> int.addr<07:00>

FIELD DESCRIPTION

lcbw<6:0> - last check bits written to either array (n/a on a partial write)

4Mb - 4 Megabit chips are stuffed in the arrays

ms<1:0> - size encoding of the memory board as follows :
3 - fully stuffed board
2 - half stuffed board
1 - quarter stuffed board
0 - undefined

board.id<7:0> - identifier code for the memory board (01 hex)

err.addr<31:00> - address at which the last EDAC error occurred which has not yet had its interrupt reset (bits 31-28 are not valid - always = 1)

0sb* - array 0 single bit (correctable) error (active low)

0.syn<6:0> - array 0 syndrome

0mb* - array 0 multiple bit (uncorrectable) error (active low)

0.info<6:0>* - error information bit breakdown:

6 - error happened on a read (active low)
5 - error happened on a 16 byte read (active low)
4 - error happened on a 32 byte read (active low)
3:0 - slot address of read requester
or byte selects for partial write (active high)

1sb* - array 1 single bit (correctable) error (active low)

1.syn<6:0> - array 1 syndrome

1mb* - array 1 multiple bit (uncorrectable) error (active low)

1.info<6:0>* - error information bit breakdown :
6 - error happened on a read (active low)
5 - error happened on a 16 byte read (active low)
4 - error happened on a 32 byte read (active low)
3:0 - slot address of read requestor
or byte selects for partial write (active high)

diag.cb<6:0> - diagnostic check bits

control<6:0> - defined as follows:
6 - Red LED* (active low)
5 - Green LED (active high)
4 - enable memory error interrupt generation (active high)
3 - enable EDAC (active high)
2 - enable parity error mode (active high)
1 - enable check bit as data mode (active high)
0 - enable diagnostic mode (active high)

int.vect<7:0> - interrupt vector written by service module

int.prio<6:0> - interrupt priority written by service module

id<3:0> - service module destination address written by service module

ia<11:08>,int.addr<07:00> - service module offset address written by service module

REGISTER READ AND WRITE SIZE

All registers are readable only as a right aligned longword. Control register is aligned byte writable. Diagnostic check bits are aligned byte writable. Interrupt information register is longword writable.

MEMORY ARRAY SIZE

center; l l. 0000 0000 - 007F FFFF 8 MegaByte memory board size 0000 0000 - 00FF FFFF 16 MegaByte memory board size 0000 0000 - 01FF FFFF 32 MegaByte memory board size 0000 0000 - 03FF FFFF 64 MegaByte memory board size 0000 0000 - 07FF FFFF 128 MegaByte memory board size

LEDs

There are 8 LEDs on the front panel of the Memory Module The meaning of the lights from top to bottom is as follows:

- Red LED (controlled by software - see control register above)
- Green LED (controlled by software - see control register above)
- Read (when active, indicates that a read is in progress)
  - 0.Write (when active, indicates that a write is in progress to array 0, i.e. an address with low byte X0 or X8)
  - 1.Write (when active, indicates that a write is in progress to array 1, i.e. an address with low byte X4 or XC)
- Partial Write (when active, indicates that a partial write is in progress.
- Error (when on, indicates that an error has been latched and the error information which belongs to this error has not yet been read. Refer to error information register above)
- Ready (indicates the state of the CSS bus ready signal - the board is ready when on.)