EK-MS730-TD-001

VAX-11/730 Memory System

Technical Description

Prepared by Educational Services
of
Digital Equipment Corporation

1st Edition, May 1982

The information in this document is subject to change without notice. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

Printed in U.S.A.

CONTENTS

TABLES

CHAPTER 1
INTRODUCTION AND OVERVIEW

1.1 SCOPE

This document is a technical description of the VAX-11/730 memory system. This system consists of the M8391 memory controller module and the M8750 memory array module(s). The memory system is briefly described, on a system level, in Chapter 1 of the *CPU Technical Description Manual* (EK-KA-730-TD) (Table 1-1). In this description, the M8391 and M8750 modules are treated as single blocks with a brief description of their basic function within the VAX-11/730 system. The interface of the memory system with other portions of the VAX 11/730 is also described.

This document treats the memory system in two levels of detail. Chapter 1 contains an overall block diagram of the memory system divided into functional areas. The system is described in terms of these areas and the functions they perform.

Chapter 2 further develops the descriptions by providing a detailed discussion of the functional areas defined in Chapter 1. Block diagrams and flow diagrams are used in Chapter 2 to show the logic composition of each area and how it performs its function with respect to the system.

Chapter 3 treats the memory system firmware. Program control of the memory system is implemented by a 512 $\times$ 72 bit control store PROM. A listing of the PROM's contents is contained in the engineering documentation. Chapter 3 illustrates how to use the listings to generate flow diagrams for the memory system routines.

1.2 RELATED DOCUMENTS

The documents listed in Table 1-1 provide additional information related to the VAX-11/730 system.

1.3 MEMORY CONTROLLER SIMPLIFIED BLOCK DIAGRAM DESCRIPTION (Figure 1-1)

The memory controller manages operation of the VAX-11/730 memory system. Upon request, the controller assigns the memory system to the CPU or to a UNIBUS device. Using a translation buffer, the controller translates virtual addresses from the CPU or addresses from UNIBUS devices, into physical addresses that are applied to the M8750 memory array module(s). Up to five array modules may be used.

If the memory reference is a read operation, the data, along with associated ECC (error checking and correction) check bits, are retrieved from the array module and applied to the ECC logic. The ECC logic checks for data errors. If a single bit error is detected, the logic is used to correct the error. The logic can detect, but not correct, multibit errors.

The data passes from the ECC logic to the data rotator. Data retrieved from the arrays is always a 32-bit longword (4 bytes) but is not always in the desired position for the CPU or UNIBUS device (i.e., byte 0 in bit position 0 on the bus). If the data is not in the desired position, the data rotator functions to rearrange the data before it is sent to the CPU or the UNIBUS device.

Table 1-1 Related Documents

Item	Title	Document Number	Contents
1	VAX-11/730 Hardware User's Guide	EK-11730-UG	This document contains hardware operating information and user care information. Included is a description of the system controls and indicators, user maintenance instructions, and an overview of the system hardware including the peripheral equipment.
2	VAX-11/730 Central Processor Unit Technical Description	EK-KA730-TD	This document provides an overview of VAX-11/730 system including summary of system busses. It describes the following CPU functions: control store and microsequencing, instruction processing, process interrupt control, data flow,and the generation of system clocks. It also describes the CPU console.
3	FP730 Floating-Point Accelerator Technical Description	EK-FP730-TD	This document describes the floating-point accelerator option. It also describes the option interface, instructions, and algorithms as well as the FPA microcode.
4	VAX-11/730 Integrated Disk Controller Technical Description	EK-RB730-TD	This document describes IDC as an interface to the RL02 and/or R80 disk units.
5	H7202B Power Supply Technical Description	EK-PS730-TD	This document provides a functional description of the H7202B power supply. It also describes the power supply controls and indicators, operating information, power and power signal distribution, and power supply specification data.
6	VAX Hardware Handbook	EB-17281	This document introduces the VAX hardware elements, including central processor units, intelligent console subsystems, I/O subsystems, MASSBUS and UNIBUS systems, main memory and memory management.
7	VAX Handbook Architecture	EB-19580	This document introduces VAX architecture, addressing modes, and the native mode instruction set.

Table 1-1 Related Documents (Cont)

Item	Title	Document Number	Contents
8	VAX Software Handbook	EB-08126	This document introduces the VAX/VMS virtual memory operating system, its operation, hardware interaction, data structures, features, and capabilities.
9	PDP11 Bus Handbook	EB-17525	This document contains the UNIBUS specification that defines the terminology and specifies the requirements of the UNIBUS. It also contains portions of the LSI-11 specification.
10	Micro 2 User's Guide Reference Manual	AA-H531A-TE	This document introduces the MICRO 2 language. It defines fields and macros and describes assembly and output listings.

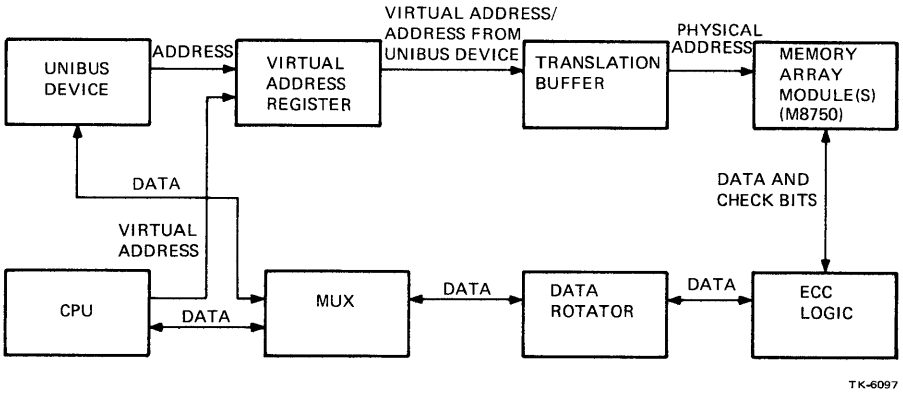


Figure 1-1 VAX-11/730 Memory System Simplified Block Diagram

If the memory reference is a write operation, the write data from the CPU, or the UNIBUS device, is applied to the data rotator. If the array location that is to be written is not longword aligned (the address is not a multiple of 4*), the data rotator formats the data according to the addressed location in the array.

The data is input into the ECC logic where ECC check bits are generated for the new data. The new data, along with the generated check bits, is then written into the memory arrays.

*See Paragraph 1.4.3.1 and Figure 1-3.

1.4 MEMORY CONTROLLER FUNCTIONAL BLOCK DIAGRAM DESCRIPTION (Figure 1-2)

Figure 1-2 is an overall functional block diagram of the M8391 memory controller. The functional subsections shown in the diagram are discussed in Paragraphs 1.4.1 through 1.4.6. The same subsections are developed in more detail in the logic descriptions of Chapter 2.

1.4.1 Arbitrator

The arbitrator regulates activity on the UNIBUS and assigns the memory controller to the CPU if the controller is free (not being used by a UNIBUS device).

The arbitrator receives an NPR (Non Processor Request for the UNIBUS) from a UNIBUS device. If no other UNIBUS device is waiting for the UNIBUS (SACK* false) the arbitrator issues an NPG (non processor grant) to the requesting device which then becomes bus master.†

When the CPU requests the memory, it asserts MEMORY REQ. When the arbitrator receives MEMORY REQ, it returns CPU GRANT to the CPU so long as a UNIBUS device doesn't have control of memory (MSYN false). CPU GRANT indicates to the CPU that it has control of memory.

Note that although the arbitrator does not assign the memory controller to UNIBUS devices, it does monitor requests for memory (MSYN) from the devices.

Both CPU GRANT and MSYN are sent to the microsequencer.

1.4.2 Microsequencer and Control Store PROM

The control store PROM outputs 72-bit microwords that control all operations within the memory controller. The microsequencer formulates the PROM address and thereby selects the next 72-bit microword.

When power is applied, the microsequencer steps through a power-up/initialize routine that checks operating voltages, resets the memory controller circuits, and places the memory controller into an idle state. The microsequencer remains in the idle state until CPU GRANT or MSYN asserts.

When CPU GRANT asserts, the memory controller is assigned to the CPU. The microsequencer looks at CSR dispatch bits from the CPU to determine the type of operation requested (read, write, etc.). The microsequencer uses these bits to dispatch the PROM to the correct starting address.

When MSYN asserts, the memory controller is assigned to a UNIBUS device. The microsequencer looks at the control bits (C <01:00>) from the UNIBUS device to determine the type of operation requested. The microsequencer uses these bits to dispatch the PROM to the correct starting address.

After the microsequencer has dispatched the PROM to the requested operation, it steps the PROM through the selected routine. Twenty-one of the 72 microword control bits are used by the microsequencer for branch testing and to formulate the PROM's next address.

---

*Selection acknowledged.

†The device becomes bus master only if the UNIBUS is free. If some other device is bus master, the requesting device must wait until the UNIBUS is free.

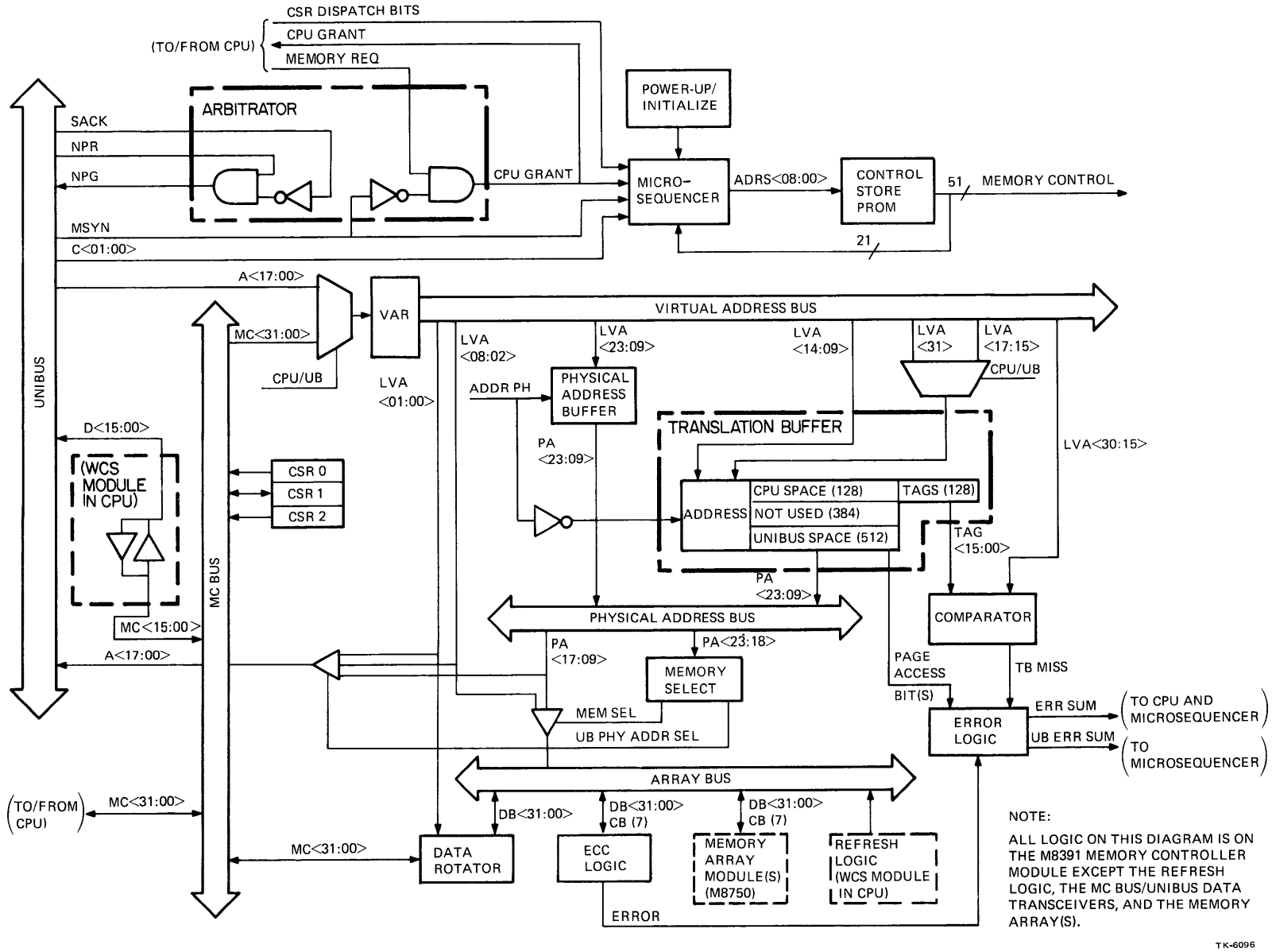


Figure 1-2 VAX-11/730 Memory System Block Diagram

1.4.3 Address Translation

The translation performed to obtain the array's physical address may be of the virtual address obtained from the CPU or of the address obtained from the UNIBUS device. The following discusses each translation separately.

1.4.3.1 CPU Virtual Address Translation – When the CPU has control of the memory controller, an address mux selects the 32-bit virtual address (MC<31:00>) from the memory controller (MC) bus. The mux places the virtual address into the VAR (virtual address register) which outputs the virtual address (LVA<31:00>) onto the virtual address bus.

With four bytes to a longword, LVA <02> becomes the least significant bit addressing longword locations. LVA <01:00> select only the byte location within the longword (Figure 1-3). Thus LVA <01:00> are not used to address the array but are used by the data rotator to rearrange data (shift bytes) as required.

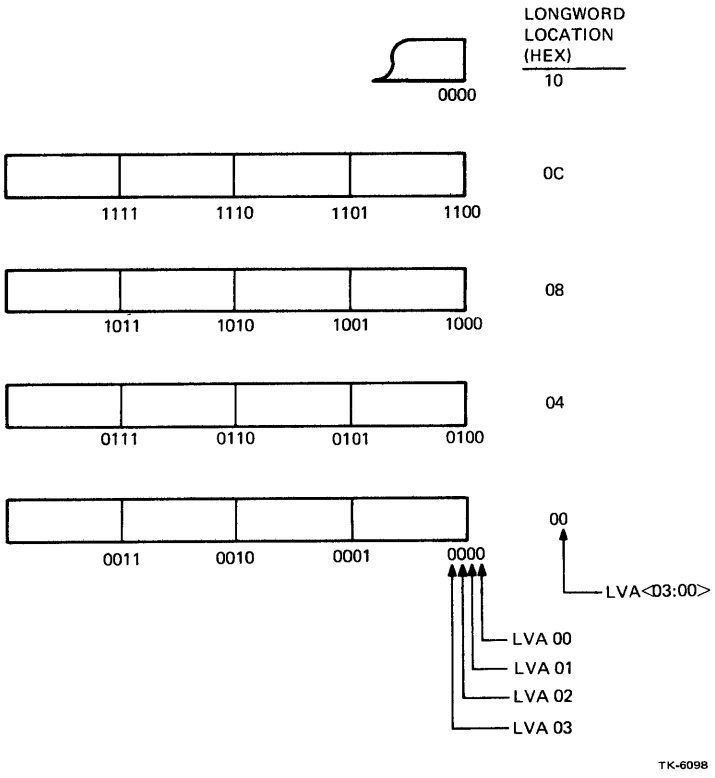


Figure 1-3 Longword Alignment in Memory Arrays

Virtual address bits LVA <08:02> are the page frame offset or index into the page frame. Therefore, these bits are not translated but contribute directly to the physical address formulated on the array bus.

The address translation process uses a translation buffer consisting of 1K of storage. The 1K area is divided into three parts.

1. CPU space – 128 locations used during a CPU/memory operation
2. UNIBUS space – 512 locations used during a UNIBUS/memory operation
3. Space not used – 384 unused locations

Each CPU and UNIBUS location contains an entry consisting of a 15-bit physical address (page frame number or PFN) and page access bit(s). (UNIBUS TB entries have only one access bit. CPU entries have several.)

Each TB entry in CPU space has a 16-bit tag stored in a tag store area of the TB. The tag is stored at the same address as its associated TB entry. The tag is LVA <30:15> of the virtual address used to address the TB when the associated TB entry was written into the TB. Thus the tag is part of the virtual address of the TB entry.

The TB is enabled by the negated state of ADDR PH. LVA <14:09> and LVA <31> form a 7-bit address into the CPU space of the TB. Address bit LVA <31> is selected by a CPU/UB mux. The addressed TB entry and its associated tag are retrieved from the TB.

The tag is sent to a comparator which compares the tag with bits LVA <30:15> from the virtual address bus. If a match is obtained, the TB entry is the one being addressed from the virtual address bus. If they don't match, the entry is not the one being addressed. In this case, the comparator asserts TB MISS to error logic which asserts ERR SUM to the CPU and the microsequencer.

The TB entry's page access bits are coupled to the error logic, which determines if the type of operation requested is allowed on this particular page. If the error logic detects an access violation, it asserts ERR SUM to the CPU and the microsequencer.

The TB entry's PFN (PA<23:09>) is placed on the physical address bus as the physical address of the desired page. The area of physical memory referenced by the CPU may be a memory array module or it may be a UNIBUS address. Physical address bits PA<23:18> specify which area is referenced. The bits are applied to memory select logic which asserts MEM SEL if an array module is the target, or UB PHY ADDR SEL if the UNIBUS is the target.

If an array module is the CPU target, PA<17:09> contribute the page address to the array bus and LVA <08:02> contribute the offset address. Together they select the referenced longword within the memory array.

If the UNIBUS is the CPU target, PA<17:09> combines with LVA <08:02> and LVA <01:00> from the virtual address bus to form an 18-bit UNIBUS address (A <17:00>). The UNIBUS address thus becomes the physical address referencing the CPU target location on the UNIBUS.

1.4.3.2 CPU Physical Address Reference – In certain cases (e.g., during system boot-up) the CPU accesses pages that reside at specific locations in physical memory. In this case, the CPU can make a direct reference to physical memory without the need for address translation. To make a direct physical access, the CPU places the physical address on the MC bus. The physical address is loaded into the VAR and output onto the virtual address bus. LVA <01:00> is used by the data rotator and LVA <08:02> specify the page offset to the array bus just as in CPU virtual address translations. ADDR PH asserts during a physical address reference disabling the TB and enabling the physical address buffer. The physical address buffer transfers LVA <23:09> from the virtual address bus directly to the physical address bus as PA <23:09>. Physical address bits PA <23:09> function to select the location of the page just as they would during a CPU virtual address translation.

1.4.3.3 UNIBUS Address Translation – When the UNIBUS has control of the memory controller, the address mux selects the 18-bit address (A<17:00>) from the UNIBUS. The mux places the address into the VAR which outputs LVA <17:00> onto the virtual address bus.

LVA <01:00> is used by the data rotator and LVA <08:02> specify the page offset to the array bus just as in CPU virtual address translations.

UNIBUS space in the translation buffer is used for a UNIBUS address translation just as CPU space is used for a CPU address translation. With ADDR PH negated, the translation buffer is enabled and LVA <14:09> and LVA <17:15> form a 9-bit address into UNIBUS space. Address bits LVA <17:15> are selected by the CPU/UB mux. The addressed TB entry is retrieved from the translation buffer.

In a UNIBUS translation, LVA <17> is the most significant bit on the virtual address bus. Thus the nine address bits cannot address more than the 512 entries in UNIBUS space. Therefore, no tags are used with the UNIBUS entries.

The entry's page access bit is applied to the error logic to check the validity of the access. If an access violation is found, UB ERR SUM is asserted to the microsequencer.

The physical address bits (PA<23:09>) from the TB entry are placed onto the physical address bus. The bits select the physical location of the page as in a CPU virtual address translation except that in the latter only the memory arrays are referenced.

1.4.4 Refresh Logic

The memory arrays must be periodically refreshed (at least every four ms) in order to retain their stored data. The refresh logic performs this function by refreshing all locations in the arrays every 3.4 ms. Refresh cycles are performed between read array and write array operations.

The refresh logic is located on the WCS module in the CPU. The WCS module and the array module(s) are powered by the battery backup option. Thus if the option is used, the refresh logic will continue to function during power interruptions, thereby saving the data stored in the arrays.

1.4.5 Data Flow

The remaining functional blocks in Figure 1-2 pertain to data flow in and out of memory. They can best be described in terms of a read and a write operation where their function can be discussed for each type of operation. The data read/writes can be bytes (8 bits) or words (16 bits) for CPU/memory or UNIBUS/memory transfers. A CPU/memory transfer could also be a 32-bit longword.

1.4.5.1 Read Operation – The physical address on the array bus selects the desired module and addresses the desired location on the module. The addressed longword and its seven associated check bits are retrieved and placed onto the array bus.

The ECC logic takes the longword and check bits from the array bus and checks the longword for data errors. If a data error is found, ERROR is asserted to the error logic which then asserts ERR SUM or UB ERR SUM depending on whether this is a CPU or a UNIBUS operation. The ECC error logic is used to correct single bit data errors but cannot correct multibit errors. The ECC error logic returns the longword to the array bus for the data rotator.

The data rotator accepts the longword from the array bus and performs any data rearrangement that may be necessary before placing the longword onto the MC bus. The CPU (or UNIBUS device) expects to receive data with the low order bits in the low order positions on the bus. The data will be in this format only if the longword read from the memory array was in the proper position. Referring to Figure 1-3, consider a word read at location 1001. The entire longword at 1000 is read out but the data must be rearranged to place the desired word at the low order position on the bus. The data rotator uses the two least significant bits off the virtual address bus (LVA <01:00> to determine if any data alignment (rotation) is necessary and functions accordingly. The data rotator outputs the aligned data onto the MC bus (MC <31:00>).

The longword is sent to the CPU if the operation is a CPU read of the memory arrays. If the operation is a UNIBUS device read of the arrays, the byte or word is tranferred to the data lines of the UNIBUS (D<15:00>) via transceivers on the WCS module in the CPU, and then to the UNIBUS device. In this case the two higher order bytes on the MC bus (MC<31:16>) are ignored.

If the operation is a CPU read of a UNIBUS device, the read data is taken off the UNIBUS data lines (D<15:00>), transferred to the MC bus via the WCS transceivers, then to the data rotator for possible data alignment, and then to the CPU.

1.4.5.2 Write Operation – In a CPU write operation, write data is placed on the MC bus from the CPU and then applied to the data rotator. The data rotator receives the write data from the MC bus and the two least significant bits of the virtual address (LVA <01:00>) from the virtual address bus. The rotator uses the two bits to determine if the data needs realignment for the addressed location. If realignment is necessary, the rotator formats the data as required.

If the CPU write is to a UNIBUS device, the rotator outputs the data back to the MC bus. From the MC bus the data is placed onto the data lines of the UNIBUS (D<15:00>) via transceivers on the WCS module, for transfer to the device. If the CPU write is to the memory arrays, the rotator outputs the data onto the data lines of the array bus (DB<31:00>).

For a UNIBUS to memory write operation, write data is placed on the MC bus via the UNIBUS data lines and the WCS transceivers, and then applied to the data rotator. From the rotator the data is placed onto the data lines of the array bus.

The ECC logic takes the write data off the array bus and uses it to generate seven ECC check bits. The data is returned to the array bus along with the check bits.

The write data and the associated check bits are taken from the array bus and written into the selected array module in the location addressed from the array bus.

1.4.6 CSR Registers

Three CSRs (control/status registers) are used to input control signals into the memory controller and to report status to the CPU. The CSRs interface with the MC bus.

CSR0 is a read only register containing the ECC check bits. The CPU reads the check bits to analyze data errors.

CSR1 is a read/write register. The CPU writes control bits into CSR1 for maintenance purposes and to regulate operation of the memory controller. Errors relating to a CPU/memory transfer are sensed by the error logic and set error bits in CSR1. When the logic asserts ERR SUM, the CPU reads the CSR1 error bits for error analysis.

CSR2 is a read only register. Errors relating to a UNIBUS/memory transfer are sensed by the error logic and set error bits in CSR2. When the logic asserts UB ERR SUM, the microsequence causes a timeout on the UNIBUS resulting in the CPU reading the CSR2 error bits for error analysis.

1.5 MAINTENANCE FEATURES

Maintenance aids have been designed into the hardware to facilitate checkout and trouble analysis of the memory subsystem. The maintenance logic is described in the functional descriptions of the area with which these features are associated. In addition, these features have been summarized in Appendix C.

CHAPTER 2
FUNCTIONAL DESCRIPTION

2.1 INTRODUCTION

The functional block diagrams in Chapter 2 use logical AND and OR symbols. It does not necessarily follow that a corresponding gate exists on the M8391 logic prints. The assertion of inputs A and B causing the assertion of output C may be represented on a block diagram by a single AND gate, yet the engineering drawing may show that several circuit stages are involved in the ANDing operation.

The signal names used on the functional block diagrams are the names used on the engineering circuit schematics (CS prints). Where other signal names or notes are used, they are enclosed in parentheses.

2.2 ARBITRATOR AND UNIBUS INTERFACE

2.2.1 General

This section describes the arbitrator and the memory controller interface to the UNIBUS.

The arbitrator consists of the CPU GRANT logic, NPG logic, UNIBUS lockout logic, and UNIBUS activity logic. The purpose of the arbitrator is to regulate UNIBUS activity and to arbitrate transfer requests from the CPU and from UNIBUS devices. The arbitrator issues CPU GRANT to the CPU when allowing a CPU access. It also issues NPG to the UNIBUS when selecting a UNIBUS device to be bus master. The device becomes bus master only if the UNIBUS is not busy. The arbitrator allows overlapping of UNIBUS requests in that it can issue an NPG and select a UNIBUS device to become the next bus master while the controller is busy processing a memory transaction with the present bus master. When the transaction is finished, the waiting UNIBUS device becomes bus master and can then request access to memory.

The UNIBUS interface paragraphs describe the signals, data, and address information relating to UNIBUS-to-controller transactions. Interfacing to the UNIBUS involves the use of standard UNIBUS signals. If the reader is unfamiliar with UNIBUS operation, it may be helpful to refer to Part 1 (UNIBUS Specification) of the *PDP-11 Bus Handbook* (Table 1-1).

Figure 2-1 is a block diagram of the arbitrator and UNIBUS Interface, and should be referred to throughout this section.

The following are the six UNIBUS signals.

UBS INTR (interrupt)
UBS NPR (non processor grant)
UBS SACK (selection acknowledged)
UBS BBSY (bus busy)
UBS SSYN (slave sync)
UBS MSYN (master sync)

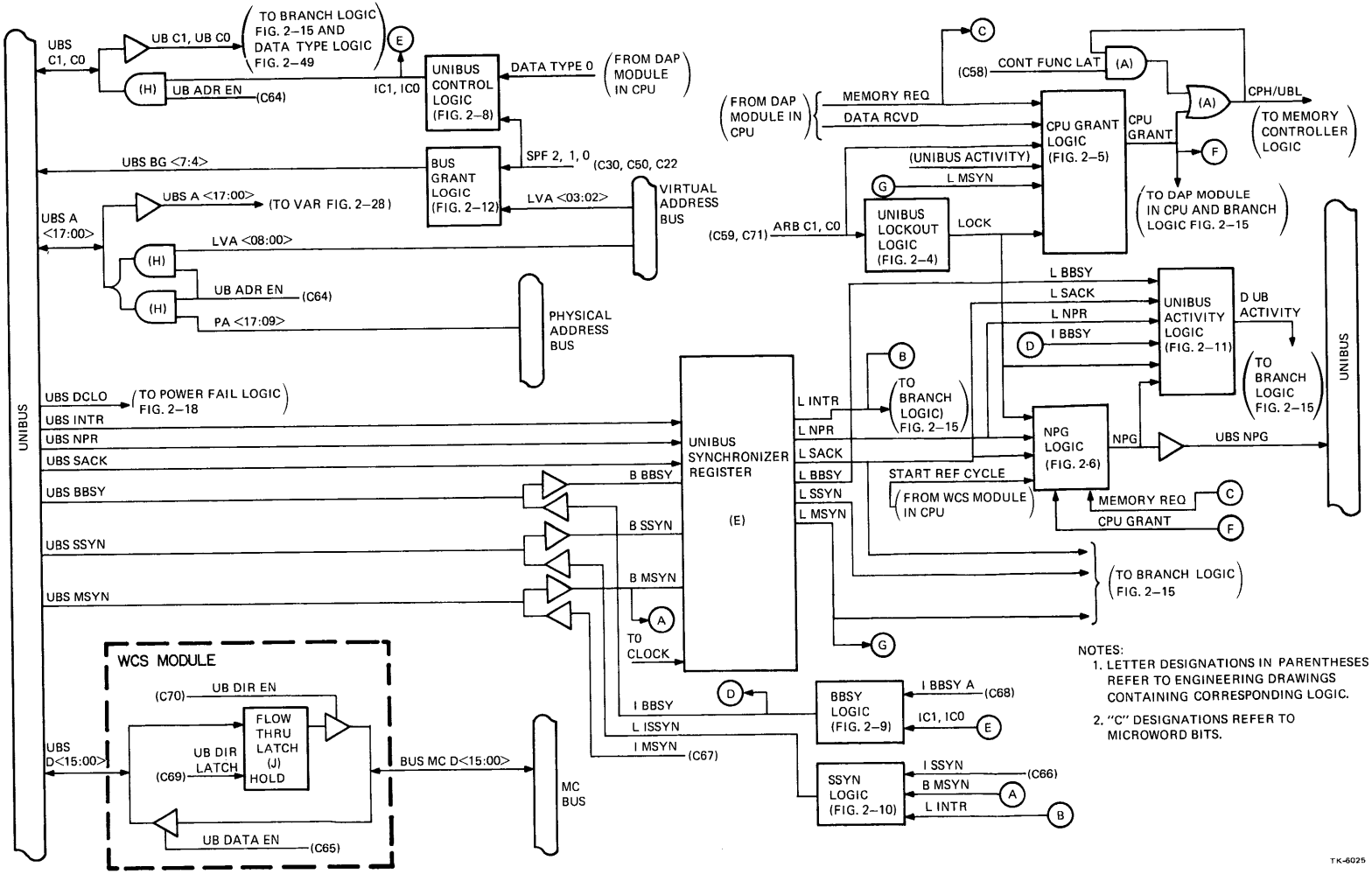


Figure 2-1 Arbitrator and UNIBUS Interface Block Diagram

The UNIBUS signals are synchronized to memory controller time by T0 CLOCK in a two-stage synchronizer register before being applied to the memory controller circuits. Besides being synchronized by T0 CLOCK, the register outputs are latched for the clock period. The output mnemonics are prefixed with L (latched) while three of the register inputs (B BBSY, B SSYN, B MSYN) are prefixed with B (buffered). UNIBUS signals generated by the microsequencer are prefixed with I (issue).

Figures 2-2 and 2-3 are flow diagrams of a CPU-to-memory transaction and a UNIBUS-to-memory transaction, respectively. The CPU-to-memory flow diagram (Figure 2-2) is oriented toward a CPU transaction in which the UNIBUS is the target address.

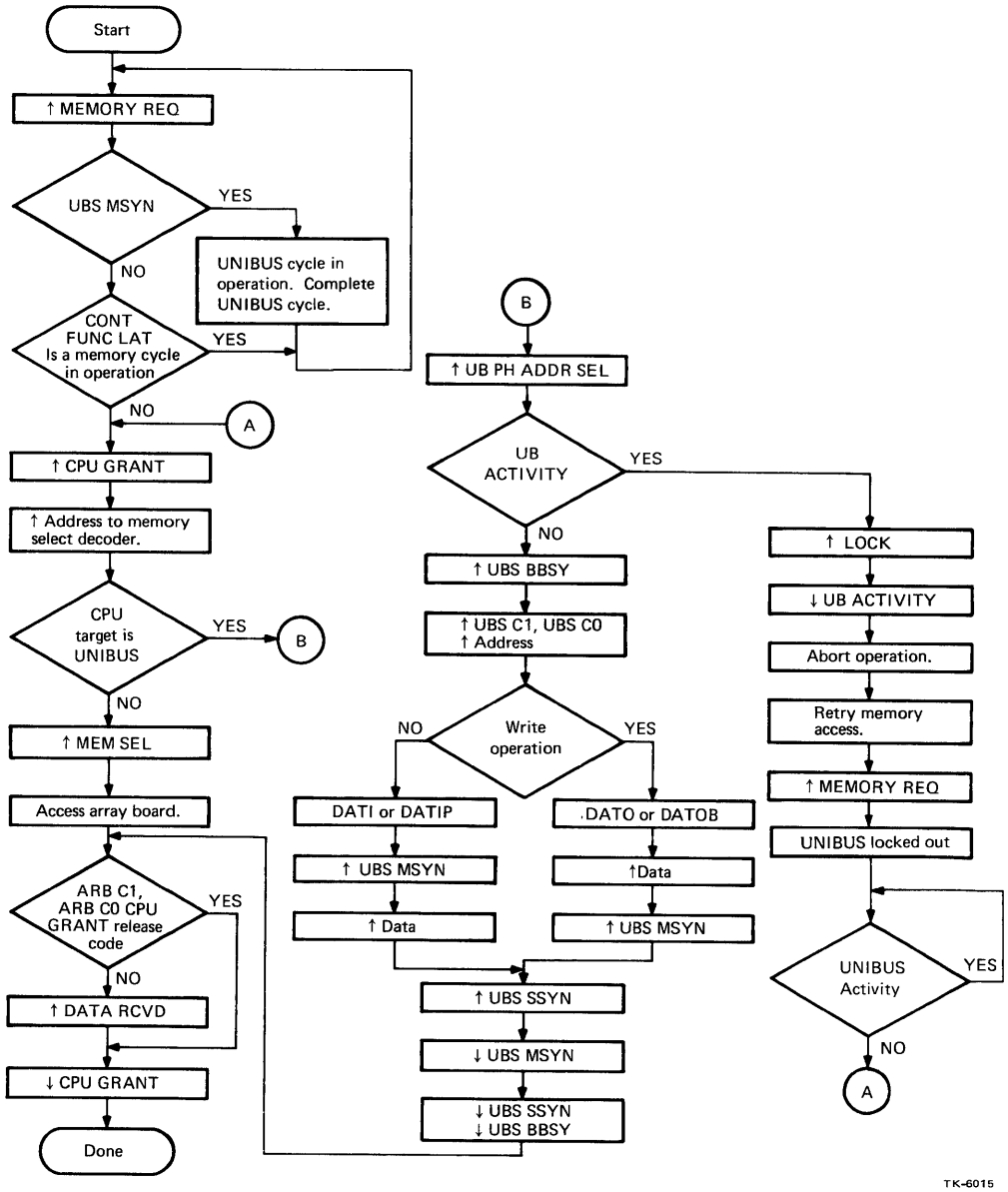


Figure 2-2 CPU/Memory Flow Diagram

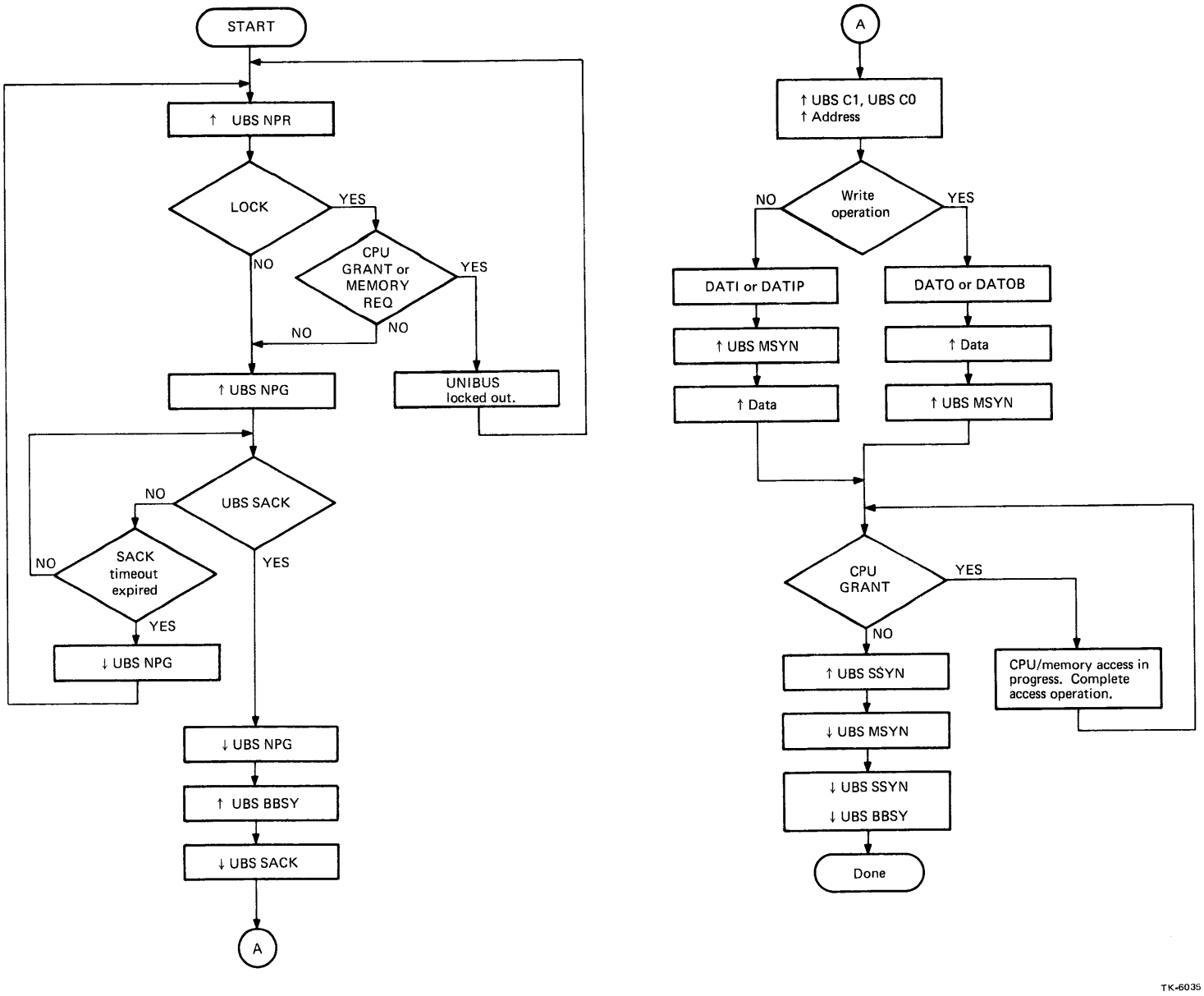


Figure 2-3 UNIBUS/Memory Flow Diagram

2.2.2 CPU/Memory Transaction (Figure 2-2)

The CPU makes a request for memory by asserting MEMORY REQ to the memory controller. A check is then made for the presence of UBS MSYN. The true state of UBS MSYN indicates a UNIBUS-to-memory transaction is in progress and the CPU must wait for the transaction to complete. If UBS MSYN is false, a check is made of CONT FUNC LAT to see if the controller is busy with other operations (e.g., a refresh cycle). CONT FUNC LAT is true if any controller operation is in progress.

If UBS MSYN and CONT FUNC LAT are false, the arbitrator asserts CPU GRANT to the CPU which now assumes control of the memory controller.

The CPU presents the virtual address to the controller where the memory select decoder decodes the memory area to be accessed. If the array boards are the CPU target, MEM SEL is asserted by the controller and the memory access will proceed.

When the memory access has been completed, arbitrator control bits (ARB C1, ARB C0) from the controller microword may specify a CPU GRANT release to negate CPU GRANT. CPU GRANT may also be negated by the receipt of DATA RCVD from the CPU. The negation of CPU GRANT releases the arbitrator for the next memory access.

If the memory select decoder specified the UNIBUS as the CPU target, UB PH ADDR SEL is asserted and the controller checks for any UNIBUS activity. The signal UB ACTIVITY is true if NPR, NPG, SACK, or BBSY are present on the UNIBUS. If there is no UNIBUS activity (UB ACTIVITY false) the controller proceeds to access the UNIBUS address. It issues UBS BBSY indicating the UNIBUS is busy and the CPU is now bus master (via the controller).

The memory controller places the two control bits (UBS C1, UBS C0), specifying the type of operation (read or write), on the UNIBUS along with the address of the UNIBUS device. If the operation is a write to the device (a data out or a data out byte) the controller places the data on the data lines of the UNIBUS and asserts UBS MSYN to the device. When the device receives the data it responds with UBS SSYN whereupon the controller negates UBS MSYN. This is followed by the negation of UBS SSYN by the device and the negation of UBS BBSY by the controller, to complete the UNIBUS access. CPU GRANT is then negated as previously discussed.

If the UBS C1, UBS C0 control bits specified a read operation (data in or data in pause), the memory controller asserts UBS MSYN to the device which responds by placing the data on the UNIBUS and raising UBS SSYN. Otherwise, the read transaction is identical to a write.

In the case where the UNIBUS is the CPU target and the UNIBUS is active (UB ACTIVITY true), the CPU access cannot complete. The memory controller asserts a UNIBUS lockout signal (LOCK) which negates UB ACTIVITY and places the controller into a "UNIBUS lockout pending" state. In this state the arbitrator responds to NPRs in the usual manner only so long as the CPU is not trying to access memory. If the CPU attempts to access memory (MEMORY REQ or CPU GRANT asserts), the arbitrator enters the "UNIBUS lockout" state wherein new NPRs are inhibited.

After the negation of UB ACTIVITY, the memory controller aborts the operation* and looks for the CPU to retry the memory access. The CPU reasserts MEMORY REQ (placing the arbitrator into the "UNIBUS lockout" state) and checks for UNIBUS activity. If there is no UNIBUS activity, CPU GRANT asserts and the CPU access of the UNIBUS will complete. If there is UNIBUS activity, the arbitrator waits for the activity to finish. With the arbitrator being in the UNIBUS lockout state, the UNIBUS is guaranteed to be quiet after the present activity is finished.

---

*UB ACTIVITY sets a BBSY error bit in CSR1 causing ERR SUM to assert. The memory controller aborts the operation by terminating the transfer and sending ERR SUM to the CPU to indicate that the data transfer was not completed.

2.2.3 UNIBUS/Memory Transaction (Figure 2-3)

A UNIBUS device initiates a request for memory by asserting UBS NPR on the UNIBUS. If the UNIBUS lockout signal (LOCK) is false, the arbitrator asserts UBS NPG to the requesting device. If LOCK is true and the CPU is accessing memory (MEMORY REQ or CPU GRANT true), the UNIBUS is locked out. In this case the UNIBUS device must wait until the lock is released to proceed with the memory access.

When UBS NPG is asserted on the UNIBUS, the requesting device normally responds by asserting UBS SACK to the controller. If UBS SACK is not received from the device within a specified timeout period (12.8 to 25.6 μs), the arbitrator negates UBS NPG thereby terminating the device's attempt to access memory.

If UBS SACK is received within the timeout period, the arbitrator negates UBS NPG causing the device to assert UBS BBSY and to negate UBS SACK.* The assertion of UBS BBSY signifies that the UNIBUS is busy and that the device is now bus master.

The device presents the two control bits (UBS C1, UBS C0) specifying the type of operation and the target address to the memory controller. If the device is to write to memory (a data out or a data out byte operation), it places the write data onto the data lines of the UNIBUS and raises UBS MSYN to the controller. When the controller receives MSYN, it checks for the presence of CPU GRANT. The true state of CPU GRANT indicates a CPU/memory transaction is in progress, in which case the transaction must complete and CPU GRANT negated before the UNIBUS access to memory can continue. With CPU GRANT false, the memory controller replies to the device with UBS SSYN whereupon the device negates UBS MSYN. This is followed by the negation of UBS SSYN by the controller and the negation of UBS BBSY by the device to complete the UNIBUS/memory transaction.

If the UBS C1, UBS C0 control bits specified a read operation (data in or data in pause), the device asserts UBS MSYN to the memory controller which responds by placing the data onto the UNIBUS and raising UBS SSYN. Otherwise, the read transaction is identical to a write.

2.2.4 UNIBUS Lockout (Figure 2-4)

The UNIBUS lockout signal (LOCK) is generated by two arbitrator control bits (ARB C1, ARB C0) obtained from the controller microword. When the lockout decoder senses a "set UNIBUS lockout" command, it asserts an output to the UB lockout flip-flop that is then set by T0 CLOCK. When the flip-flop sets, LOCK is asserted to the arbitrator. LOCK is also fed back to the decoder to latch the flip-flop set until the decoder senses a "release UNIBUS lockout" command. When this occurs the decoder output negates and causes LOCK to go false.

The arbitrator bit code relating to UNIBUS lockout is shown in Table 2-1.

*If UBS BBSY is already asserted by another UNIBUS device, the requesting device keeps UBS SACK asserted and waits for the current bus master to negate UBS BBSY.

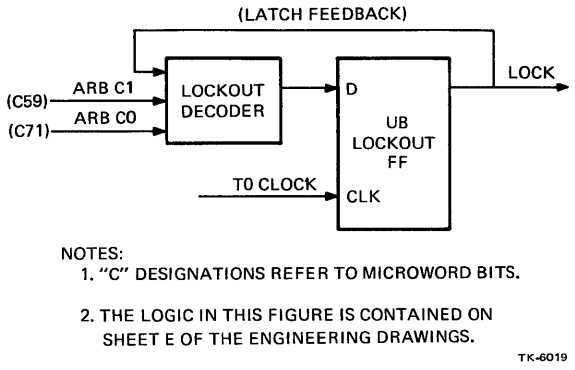


Figure 2-4 UNIBUS Lockout Block Diagram

Table 2-1 UNIBUS Lockout Code

Function	Code ARB C1	ARB C0
No change	0	0
Set UNIBUS lockout	0	1
Clear UNIBUS lockout	1	0
Clear UNIBUS lockout and CPU GRANT	1	1

2.2.5 CPU Grant Logic (Figure 2-5)

When the CPU requests access to memory, it asserts MEMORY REQ to the controller. If the controller is not executing a function (CONT FUNC LAT false) and the controller is not busy executing a UNIBUS request (L MSYN false), CPU GRANT is asserted to the CPU.

Under certain conditions (Paragraph 2.2.2) the microsequencer locks out new UNIBUS requests by asserting LOCK. With LOCK true, the CPU GRANT logic is not effected by L MSYN but is checking for UNIBUS activity by looking at NPG, L SACK, and L BBSY. When these are all false (indicating that the UNIBUS is quiet) CPU GRANT will assert.

If a DATIP operation is being executed, IBBSY is asserted by the microsequencer to interlock the read and write sequences of the operation. That is, IBBSY remains asserted after the read sequence to "hold" the UNIBUS for the write sequence. If the UNIBUS lockout signal (LOCK) is asserted during the read sequence, the CPU GRANT logic will not find the UNIBUS quiet. However, the logic senses that IBBSY is causing the UNIBUS activity and allows a CPU GRANT.

Note in Figure 2-1 that CPU GRANT asserts CPH/UBL (central processor high/UNIBUS low) to the memory controller logic indicating that a CPU/memory operation is in progress. CPH/UBL is latched up, via a feedback gate, by CONT FUNC LAT.

CPU GRANT is latched up via a feedback path such that DATA RCVD from the CPU or an arbitrator "release CPU GRANT" command from the controller microword is required too negate CPU GRANT. The arbitrator code to negate CPU GRANT is shown in Table 2-1.

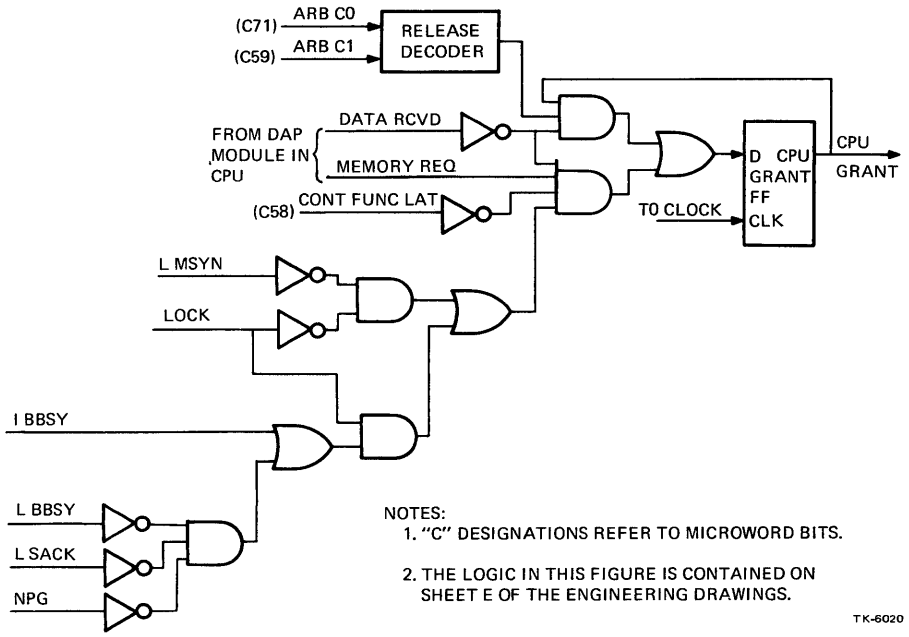


Figure 2-5 CPU Grant Block Diagram

2.2.6 NPG Logic (Figure 2-6)

When a UNIBUS device requests the memory, UBS NPR is asserted to the memory controller from the UNIBUS. UBS NPR is synchronized by T0 CLOCK in the synchronizer register and applied to the NPG logic as L NPR. If the UNIBUS is not locked out (LOCK false) and is not in the process of being locked out by the ARB code (ARB C0 false), the L NPR request asserts NPG which is placed on the UNIBUS as UBS NPG.* If the UNIBUS is locked out (LOCK true), NPG can still be asserted if the CPU does not have control of the memory controller (CPU GRANT false) and is not requesting use of the controller (MEMORY REQ false).

NPG latches itself via feedback gate A and remains true until the UNIBUS device issues UBS SACK. UBS SACK becomes L SACK via the synchronizer and negates NPG.

Timeout circuitry in the NPG logic releases NPG if UBS SACK is not received within a given timeout period. Refresh cycles are used to set the timeout period which can range from approximately 13 to 26 μs. Figure 2-7 is a timing diagram for the timeout logic. Refer to it during the following discussion.

START REF CYC from the WCS board asserts for the duration of each refresh cycle. It is synchronized by T0 CLOCK and becomes REF IN PROG which is used to trigger a timeout pulse generator. The generator output (TIMEOUT) is pulses spaced 12.8 μs apart and occurring just after each refresh cycle. The first TIMEOUT pulse after the assertion of NPG enables the R1 flip-flop to be set by the next T0 CLOCK pulse. When R1 asserts, it latches itself up via a feedback AND gate. The assertion of R1 inhibits NPG feedback gate A but enables timeout gate B which now functions to keep NPG asserted. The next TIMEOUT pulse inhibits timeout gate B thereby breaking the feedback latch causing NPG to negate.

The minimum timeout period is the 12.8 μs between refresh cycles. The maximum timeout period could be up to 25.6 μs depending on where NPG is asserted in the refresh cycle.

*Note that when LOCK is false, NPG can be asserted to a UNIBUS device even though memory is busy with a CPU request (CPU GRANT true).

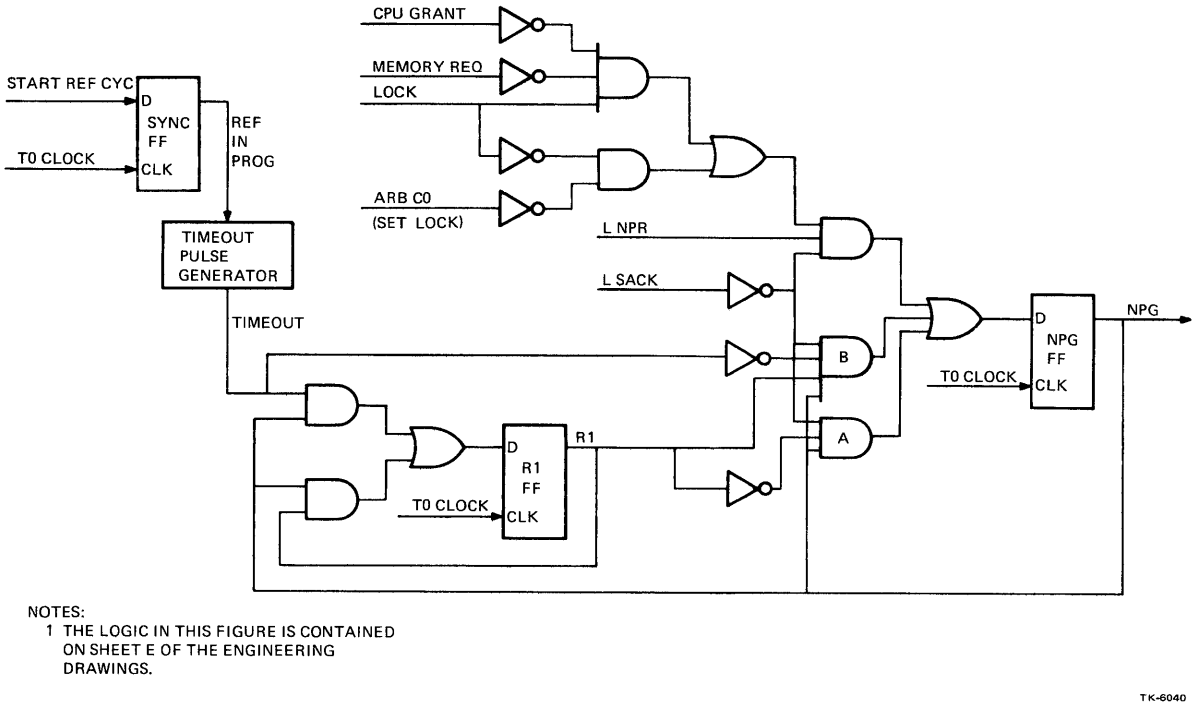


Figure 2-6 NPG Block Diagram

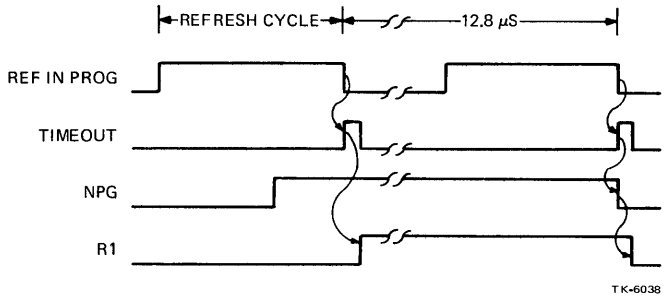


Figure 2-7 NPG Timeout Timing Diagram

2.2.7 UNIBUS Control Logic

UNIBUS control lines C1, C0 specify the type of operation to occur on the UNIBUS. The C1, C0 code is given in Table 2-2.

When a UNIBUS device is bus master, it specifies the type of operation on the UNIBUS control lines. The control signals (UBS C1, UBS C0) are received by the controller and coupled to the controller branch logic as UB C1 and UB C0.

When the CPU is bus master it specifies the type of operation, and the memory controller places the UBS C1, UBS C0 code on the UNIBUS for the UNIBUS slave device. The UNIBUS control logic generates the code from a three-bit special function code (SPF<02:00>μ0) from the controller microword and from DATA TYPE 0 from the CPU (Figure 2-8).

The special function codes pertaining to the types of UNIBUS operations are shown in Table 2-3.

Note that a DATO and DATOB operation have the same special function code. When a data out operation is specified, DATA TYPE 0 from the CPU is checked to determine if the operation is to be a DATO or a DATOB. DATA TYPE 0 is latched up as L DT0 by CONT FUNC LAT for the duration of the operation.

The code generated by the UNIBUS control logic (IC1, IC0) is gated to the UNIBUS by UB ADR EN from the controller microword.

Table 2-2 UNIBUS Control Line Code

Name	Code		Function
	C1	C0	
DATI (data in)	0	0	One word of data transferred from slave to master
DATIP (data in, pause)	0	1	Same as DATI but inhibits restore cycle in destructive read-out devices; must be followed by DATO or DATOB to the same location
DATO (data out)	1	0	One word of data from master to slave
DATOB (data out, byte)	1	1	One byte of data from master to slave

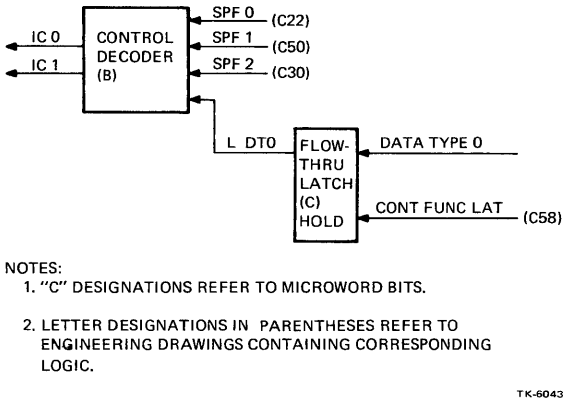


Figure 2-8 UNIBUS Control Block Diagram

Table 2-3 Special Function Codes for UNIBUS Operations

DT0	Microword Special Function Code			Operation	UNIBUS Control Bit Code	
	SPF 2	SPF 1	SPF 0		C1	C0
X	1	0	0	DATI (data in)	0	0
X	1	0	1	DATIP (data in, pause)	0	1
1	1	1	0	DATO (data out)	1	0
0	1	1	0	DATOB (data out, byte)	1	1

X = don't care

2.2.8 BBSY Logic

When a UNIBUS device becomes bus master, it asserts UBS BBSY to the controller which then asserts B BBSY and a synchronized L BBSY.

When the CPU becomes bus master, I BBSY A from the controller microword is raised which asserts I BBSY (Figure 2-9), and then UBS BBSY to the UNIBUS.

When I BBSY is asserted, it is fed back to an AND gate which also receives the IC1, IC0 code from the UNIBUS control logic. When the IC1, IC0 code specifies a DATIP operation the AND gate is enabled latching up I BBSY. The latch is necessary because the CPU master must hold the UNIBUS busy after a DATIP until the following DATO/B routine asserts I BBSY A for the DATO/B operation.*

2.2.9 MSYN Logic (Figure 2-1)

When UBS MSYN is received from a UNIBUS device that is bus master, a synchronized L MSYN is asserted to the controller branch logic. When the CPU is bus master, I MSYN is raised by the microword and placed on the UNIBUS as UBS MSYN.

2.2.10 SSYN Logic

When a UNIBUS device functions as a slave, it asserts UBS SSYN to the controller in response to UBS MSYN. UBS SSYN then asserts B SSYN to the synchronizer which outputs a synchronized L SSYN to the controller branch logic.

If the controller is functioning as the slave, the SSYN logic generates L ISSYN and outputs it to the UNIBUS as UBS SSYN. To generate L ISSYN, the SSYN logic (Figure 2-10) requires the following three inputs.

1. I SSYN from the controller microword
2. Either B MSYN (memory controller is the slave) or L INTR (UNIBUS device interrupt)
3. A negated ERROR input from the ECC logic

When L ISSYN asserts, it latches itself up, so long as B MSYN or L INTR remains true.

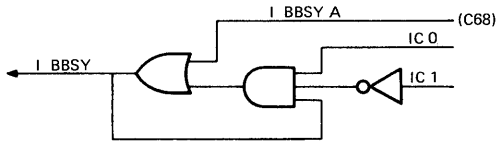
2.2.11 UNIBUS Activity (Figure 2-11)

UNIBUS activity is monitored by ORing the UNIBUS signals (L NPR, NPG, L SACK, L BBSY) and asserting UB ACTIVITY if any of the signals are true.

As discussed in Paragraph 2.2.2, the CPU cannot access the UNIBUS as a target if UNIBUS signals are active. UB ACTIVITY flags the CPU (via CSR1) that there is activity on the UNIBUS. LOCK is asserted by the CPU to lock out the UNIBUS so the CPU access may complete.

When LOCK comes true the UB ACTIVITY flag is negated. Also UB ACTIVITY is inhibited if the UNIBUS activity is due to a CPU access operation (I BBSY true).

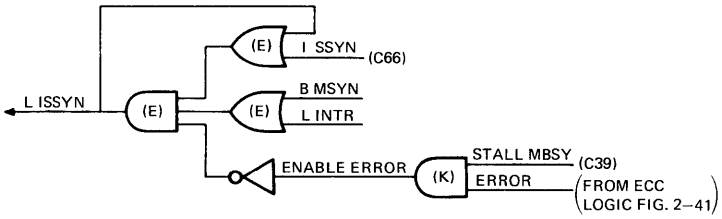
*A DATO or DATOB must follow a DATIP to the same location to maintain UNIBUS protocol.



NOTES:
1. "C" DESIGNATIONS REFER TO MICROWORD BITS.
2. THE LOGIC IN THIS FIGURE IS CONTAINED ON SHEET B OF THE ENGINEERING DRAWINGS.

TK-6042

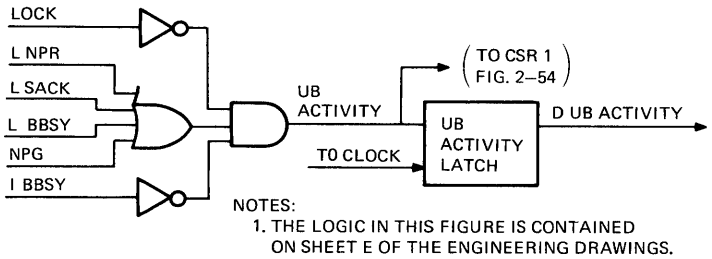
Figure 2-9 BBSY Logic



NOTES:
1. "C" DESIGNATIONS REFER TO MICROWORD BITS.
2. LETTER DESIGNATIONS IN PARENTHESES REFER TO ENGINEERING DRAWINGS CONTAINING CORRESPONDING LOGIC.

TK-6041

Figure 2-10 SSYN Logic



NOTES:
1. THE LOGIC IN THIS FIGURE IS CONTAINED ON SHEET E OF THE ENGINEERING DRAWINGS.

TK-6022

Figure 2-11 UNIBUS Activity Logic

2.2.12 Data Interface (Figure 2-1)

The 16 UNIBUS data bits (UBS D<15:00>) interface to the MC bus via drivers and a flow-thru latch located on the WCS module in the CPU.

UB DATA EN from the controller microword enables the drivers that couple data from the MC bus to the UNIBUS. Data flowing from the UNIBUS to the MC bus is applied to a flow-thru latch. The latch holding signal is UB DIR LATCH and the signal enabling the latch output is UB DIR EN. Both signals are from the controller microword.

The data interfaces to the CPU or the memory controller from the MC bus.

2.2.13 Address Interface (Figure 2-1)

The 18 UNIBUS address bits (UBS A<17:00>) are used by UNIBUS devices to reference locations on the memory array cards and by the CPU to reference UNIBUS locations.

When a UNIBUS device is referencing a memory array module, the 18-bit address is taken off the UNIBUS and passed through a transceiver amplifier to the virtual address register.

When the CPU is referencing a UNIBUS location, the 18-bit address is placed onto the UNIBUS from the memory controller. The 18 address bits are obtained from two sources. The first nine bits (LVA <08:00>) are obtained from the virtual address bus. These bits contain the byte address within the addressed page. The last nine bits (PA <17:09>) are obtained from the translation buffer via the physical address bus. These bits contain the page address.

The address bits are gated to the UNIBUS by UB ADR EN from the controller microword.

2.2.14 Bus Grant Logic

Bus requests (BRs) are placed on the UNIBUS by UNIBUS devices wanting to interrupt the CPU. The BRs go directly to the CPU which initiates a bus grant response (BG) via the memory controller. The priority level of the BG corresponds to the priority level of the BR.

The bus grant logic (Figure 2-12) receives the three-bit special function code (SPF<02:00>) from the controller microword. When all three bits are high, ISSUE BG asserts and enables the bus grant decoder output. The decoder asserts one of four bus grant outputs according to the priority level specified by LVA bits <03:02> from the virtual address bus.

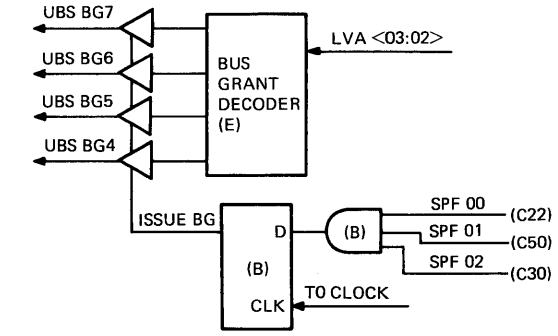


Figure 2-12 Bus Grant Block Diagram

2.2.15 DCLO

UBS DCLO is asserted on the UNIBUS as an indication that dc power has failed. It is applied to power fail logic in the microsequencer which functions to place the memory controller into the power fail microstate (Paragraph 2.3.6).

2.3 MICROSEQUENCER AND CONTROL STORE

2.3.1 General (Figure 2-13)

The memory controller control store consists of nine 512 × 8 PROMs. The microsequencer consists of the addressing logic associated with the PROMs. Each PROM has a nine-bit address input and an eight-bit output. The PROMs are addressed in parallel to effect a 512 × 72 PROM with 512 addressable locations each outputting a unique 72-bit microword.

The 72-bit microword consists of 51 memory control bits, nine "next address" bits and twelve branch control bits (Figure 2-14). The memory control bits regulate all operations within the memory controller. The "next address" bits are used as the base address in formulating the PROM's next nine-bit address. The branch control bits specify which (if any) branch condition(s) are examined to modify the four least significant bits of the base address. Parity generation and checking is not performed on the 72-bit microword.

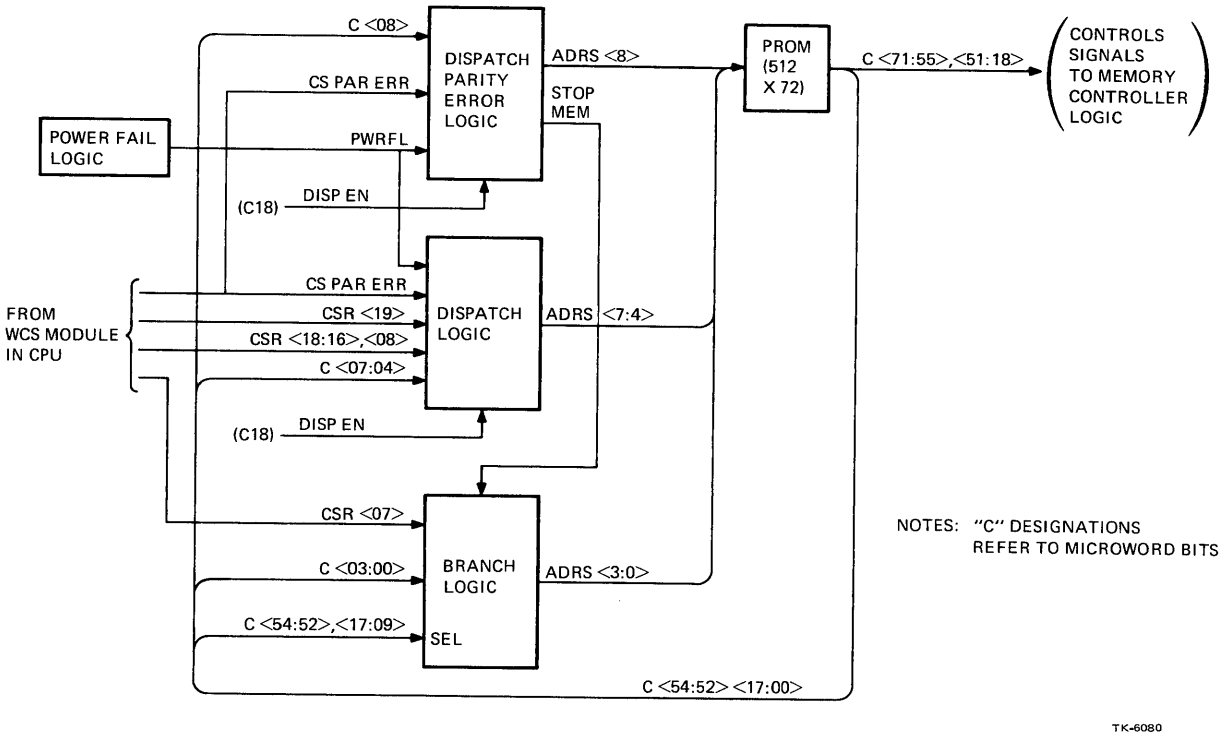


Figure 2-13 Microsequencer/Control Store Simplified Block Diagram

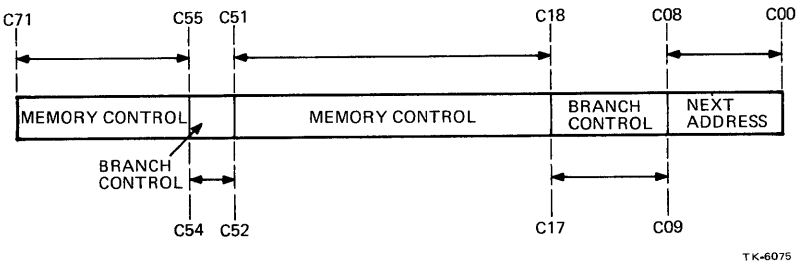


Figure 2-14 Memory Controller Microword

The microsequencer formulates the address presented to the PROM via one of the three functions listed below.

1. Dispatch Parity Error/Power Fail Function – This function monitors memory controller power and CPU control store parity. It also forces the memory controller control store PROM to a power fail routine if power is low or the CPU has a control store parity error during a dispatch function. The dispatch parity error/power fail function overrides the two functions listed below.

2. Dispatch Function – This function obtains memory control store PROM address from CPU control store. Address from CPU dispatches PROM to desired routine.

3. Branch Logic – This function monitors various branch condition signals. It sends control store PROM to specific locations according to state of selected branch condition signal.

Table 2-4 illustrates the composition of the PROM address for the three functions listed above.

2.3.2 PROM (Figure 2-15)

Every 90 ns the PROM is clocked by T0 CLOCK and outputs the 72-bit microword currently being addressed by ADRS<8:0>. The microword bits are identified as C<71:00>. In addition, the 51 memory control bits (C<71:55>, <51:18>) have mnemonics indicating their function within the controller. (The mnemonics are listed on sheet M of the engineering logic prints.) The 12 branch control bits (C<54:52>, C<17:09>) and the nine "next address" bits (C<08:00>) are identified only by their "C" designations.

The nine address lines (ADRS<8:0>) are applied to an LED display where the PROM address can be read for maintenance purposes.

The PROM outputs are always enabled by +3 V from the memory controller power source.

2.3.3 Branch Logic (Figure 2-15)

The four least significant address bits (ADRS<3:0>) are obtained from four branch enable multiplexers. Each multiplexer receives a three-bit select input of branch control bits from the microword. The three bits select one of eight data inputs for the output address bit. One of the data inputs is a next address bit corresponding to the output address bit. Thus, if the current microword does not call for any branching tests, next address bits C<03:00> become address bits ADRS<3:0> respectively. In this case the microword is specifying the next address, not subject to any branch conditions.

One of the data inputs to branch enable multiplexer 3 is CSR <07> from the WCS board in the CPU. This bit is used during the dispatch function and is discussed in Paragraph 2.3.4. The remaining data inputs are various branch condition signals selected by the multiplexer to determine the next PROM address.

Table 2-4 PROM Address Bits

Microsequencer Function	Dispatch Parity Error and Power Fail Function	Dispatch Function				Branch Function			
Address Bits for Control Store PROM	**ADRS 8**	**ADRS 7**	**ADRS 6**	**ADRS 5**	**ADRS 4**	**ADRS 3**	**ADRS 2**	**ADRS 1**	**ADRS 0**
ADRS bits during dispatch parity error or power fail microstate	1	0	0	0	0	0	0	0	0
ADRS bits during dispatch microstate	CO8	CSR 18	CSR 17	CSR 16	CSR 08	CSR 07	C02	C01	C00
ADRS bits during branch microstate	C08	C07	C06	C05	C04	C03 or one of seven branch condition signals	C02 or one of seven branch condition signals	C01 or one of seven branch condition signals	C00 or one of seven branch condition signals

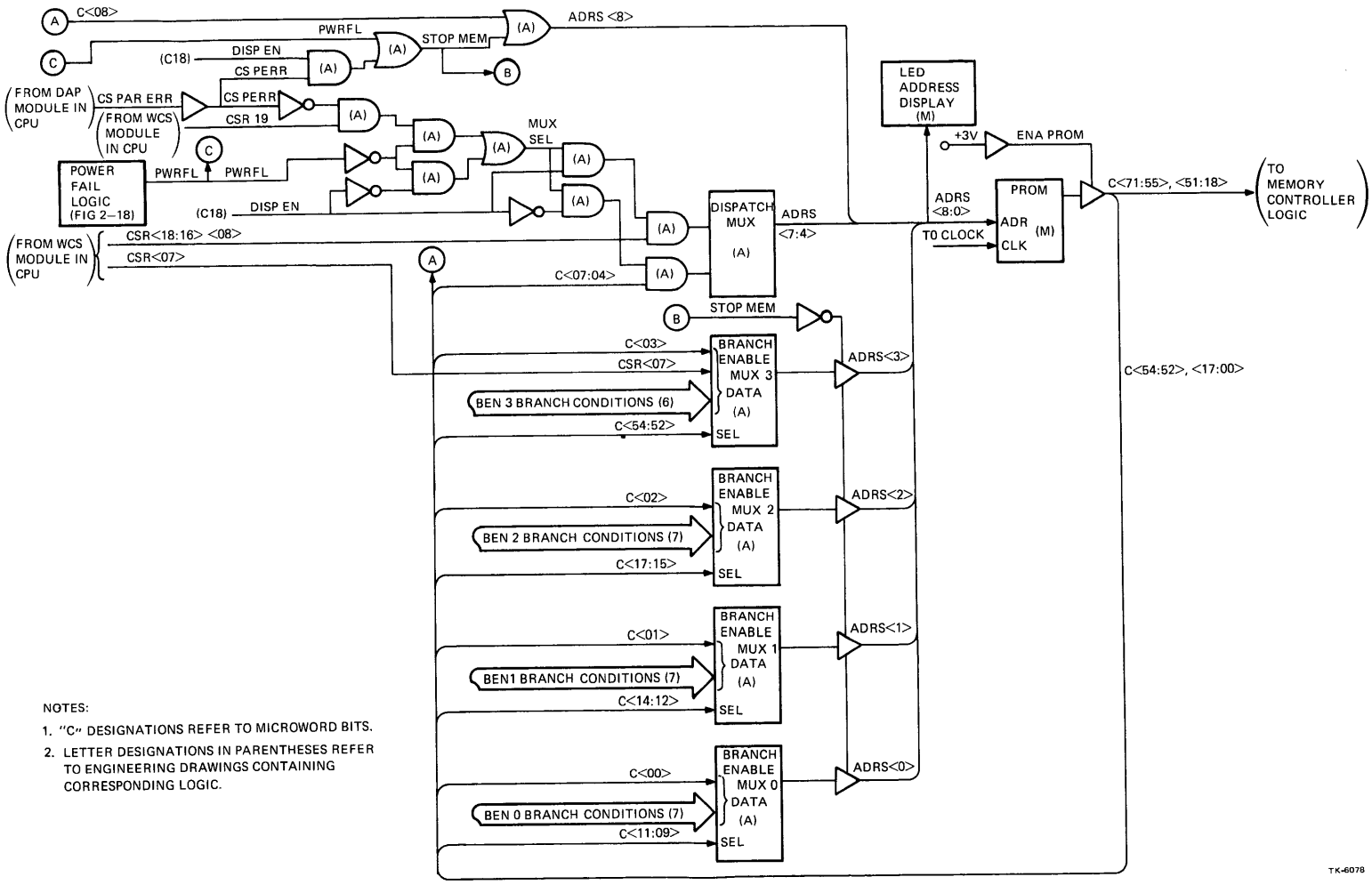


Figure 2-15 Microsequencer/Control Store Block Diagram

2.3.4 Dispatch Function (Figure 2-15)

The dispatch logic consists of a dispatch multiplexer and associated logic. The dispatch multiplexer supplies address bits ADRS<7:4> to the PROM. If the dispatch function is enabled, control store register (CSR) bits from the CPU WCS module are placed on the four address lines (ADRS<7:4>). If the dispatch function is disabled, next address bits from the PROM are placed on the address lines.

Figure 2-16 is a flow diagram of the dispatch function. The following discussion of the dispatch functions relates to Figure 2-16.

With the dispatch function enabled (DISP EN true) the CPU CSR (control store register) bits specify the PROM address. Bit CSR<19> controls the state of the dispatch multiplexer. If it is false, MUX SEL is false and the multiplexer is disabled. With the multiplexer disabled ADRS<7:4> are forced to all zeros causing the PROM to jump to the CPU.ISTREAM.REQ routine. If CSR<19> is true, the dispatch multiplexer is enabled and selects control store bits CSR<18:16>, <08> for address bits ADRS <7:4>. The CSR bits send the PROM to one of 31 dispatch routines as listed in Table 2-5.

If DISP EN is false, the dispatch function is disabled and the dispatch multiplexer selects next address bits C<07:04>. In this case the microword selects its own next address.

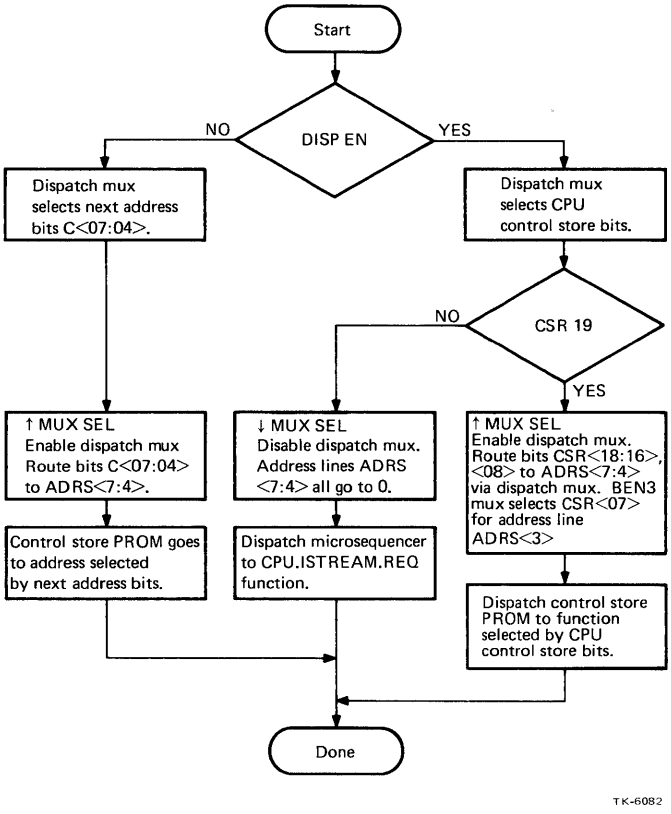


Figure 2-16 Dispatch Function Flow Diagram

Table 2-5 CPU Dispatch Routines

CSR/ADRS Bits					Dispatch Routine
CSR<18> ADRS<7>	CSR<17> ADRS<6>	CSR<16> ADRS<5>	CSR<08> ADRS<4>	CSR<07> ADRS<3>	
0	0	0	0	0	CPU.ISTREAM.REQ
0	0	0	0	1	WRITE.TB
0	0	0	1	0	TEST.V.RCHK
0	0	0	1	1	CPU.READ.PH
0	0	1	0	0	WRITE.PH
0	0	1	0	1	TEST.V.WCHK
0	0	1	1	0	ROTATE.9BITS.RIGHT
0	0	1	1	1	ROTATE.15BITS.LEFT
0	1	0	0	0	CPU.READ.V.NOCHK
0	1	0	0	1	CPU.READ.V.WCHK
0	1	0	1	0	CPU.READ.V.RCHK
0	1	0	1	1	RD.MAINT.VAR.INC
0	1	1	0	0	WRITE.V.NOCHK
0	1	1	0	1	WRITE.V.WCHK
0	1	1	1	0	READ.V.RCHK.IFILL
0	1	1	1	1	WRITE.UBS.MAP
1	0	0	0	0	WRITE.PH.OCTA
1	0	0	0	1	WRITE.TB.STEP
1	0	0	1	0	READ.UBS.MAP
1	0	0	1	1	READ.TB
1	0	1	0	0	READ.MAINT.ADDR
1	0	1	0	1	MAINT.ECC.DAT
1	0	1	1	0	READ.CSR
1	0	1	1	1	WRITE.CSR
1	1	0	0	0	CPU.RD.PH.OCTA
1	1	0	0	1	READ.V.WCHK.LOCK
1	1	0	1	0	OCTA.READ.V.RCHK
1	1	0	1	1	RD.MAINT.BRANCH.CK
1	1	1	0	0	ISSUE.BG
1	1	1	0	1	OCTA.WR.V.WCHK
1	1	1	1	0	QUAD.READ.V.RCHK
1	1	1	1	1	RD.MAINT.UBS

The dispatch function described above is implemented in the logic of Figure 2-15. With DISP EN true, the next address bits are blocked from the multiplexer input and the CSR bits are selected. However, for the multiplexer to be enabled, MUX SEL must be asserted by CSR<19> true. If CSR<19> is false, MUX SEL is false and the multiplexer is disabled. When this occurs, ADDRS<7:4> go to 0 and the CPU.ISTREAM.REQ instruction is executed. If CSR<19> is true, the multiplexer is enabled and the CSR bits from the CPU are selected for ADRS<7:4>.

A fifth control store bit (CSR<07>) is routed to the PROM during the dispatch function as ADRS<3> and contributes toward the selection of a routine from Table 2-5. CSR<07> is routed to the PROM via branch enable multiplexer 3 which selects the CSR<07> data input when in the dispatch microstate.

When DISP EN is false, MUX SEL is true enabling the multiplexer which now selects next address bits C<07:04> for the PROM address.

2.3.5 Power Fail/Parity Error Function (Figures 2-15 and 2-17)

Address bit ADRS<8> is the most significant bit addressing the control store PROM. ADRS<8> is normally a function of next address bit C<08>; however, two types of improper system operation will assert STOP MEM forcing ADRS<8> to a 1. One is a system power failure and the other is a control store parity error in the CPU during a dispatch operation.

Power fail logic monitors system power and asserts PWRFL if power is lost (Paragraph 2.3.6). The assertion of PWRFL asserts STOP MEM.

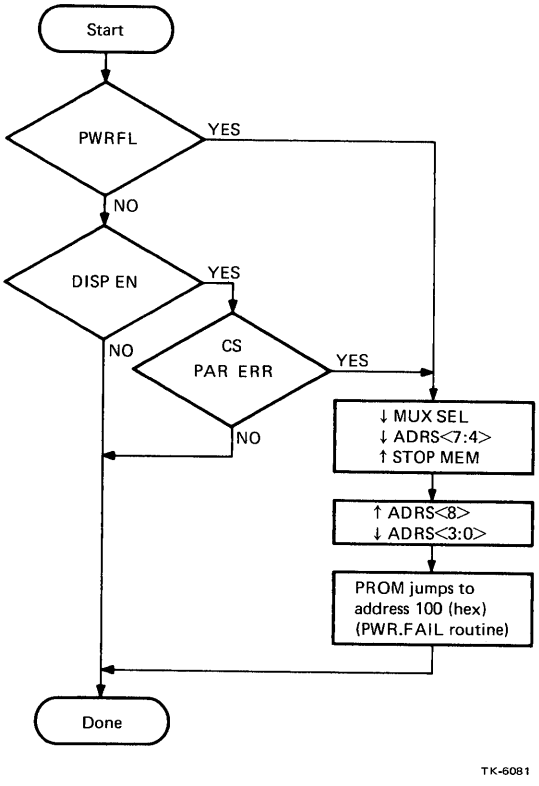


Figure 2-17 Power Fail/Parity Error Flow Diagram

If a control store parity error occurs in the CPU, the CPU asserts CS PAR ERR to the memory controller. CS PAR ERR becomes CS PERR which asserts STOP MEM if a dispatch function is in progress (DISP EN True). During a dispatch operation, the CPU control store register bits are used to address the PROM, hence a CPU CS parity error at this time aborts the dispatch operation.

STOP MEM, in addition to asserting ADRS<8>, inhibits the output of the branch logic causing ADRS<3:0> to go to all zeros.

Although STOP MEM is not applied to the dispatch logic, CS PERR, DISP EN, and PWRFL are inputs to the dispatch logic and cause it to respond in the same manner. If PWRFL is true, or if both DISP EN and CS PERR are true, then MUX SEL is false disabling the dispatch multiplexer. With the dispatch multiplexer disabled, the multiplexer outputs (ADRS<7:4>) all go to 0.

Thus, when a power failure occurs, or a CS parity error occurs during a dispatch operation, ADRS<8:0> goes to 100 (hex) sending the PROM to the power fail routine (PWR.FAIL). The power fail routine resets the memory controller arbitrator and, if system power is normal, returns the memory controller to the idle state.

2.3.6 Power Fail Logic (Figure 2-18)

The power fail logic functions to assert PWRFL whenever memory controller power is below its minimum specification tolerance.

When a system ac power failure occurs, a power-down sequence is initiated wherein CINIT is asserted from the WCS module a few milliseconds before dc power falls below its minimum tolerance level. CINIT asserts INIT u which is synchronized by T0 CLOCK and delayed by one clock pulse to become L INIT DLY. L INIT DLY asserts PWRFL if an operation is not in progress (CONT FUNC LAT false). If an operation is in progress the assertion of PWRFL is delayed until the operation completes. If the operation is a refresh cycle (ALLOW REF true) PWRFL asserts as soon as L INIT DLY comes true.

PWRFL is sent to the memory array logic to disable generation of the timing gates for the array modules (Figure 2-36; Part 1). (A battery backup option powers the WCS module, where the array timing gates are generated during refresh cycles, thereby allowing refresh cycles to continue during a power failure.) PWRFL is latched up via an AND gate so long as L INIT DLY is true. UBS DCLO asserts on the UNIBUS when power drops below its minimum tolerance level thereby maintaining L INIT DLY true during the power interruption.

Thus, when ac power fails, the logic functions to allow the current operation to complete and then sends the control store PROM (in the memory controller) to the power fail routine (PWR.FAIL).

During power-up (when power is applied or returns after an interruption) the memory controller power-up flip-flop receives its dc operating voltage and initially resets due to the exponential delay in the assertion of +5 V POWER UP. With the power-up flip-flop reset, PWRUP FLP and PWRFL are both true. When +5 V POWER UP rises to the level required to condition the power-up flip-flop to set, T0 CLOCK sets the flip-flop negating PWRUP FLP and PWRFL.

Thus, during power-up, UBS DCLO negates; however the power fail logic functions to delay the negation of PWRFL assuring that memory controller power is normal before sending the control store PROM to the idle state.

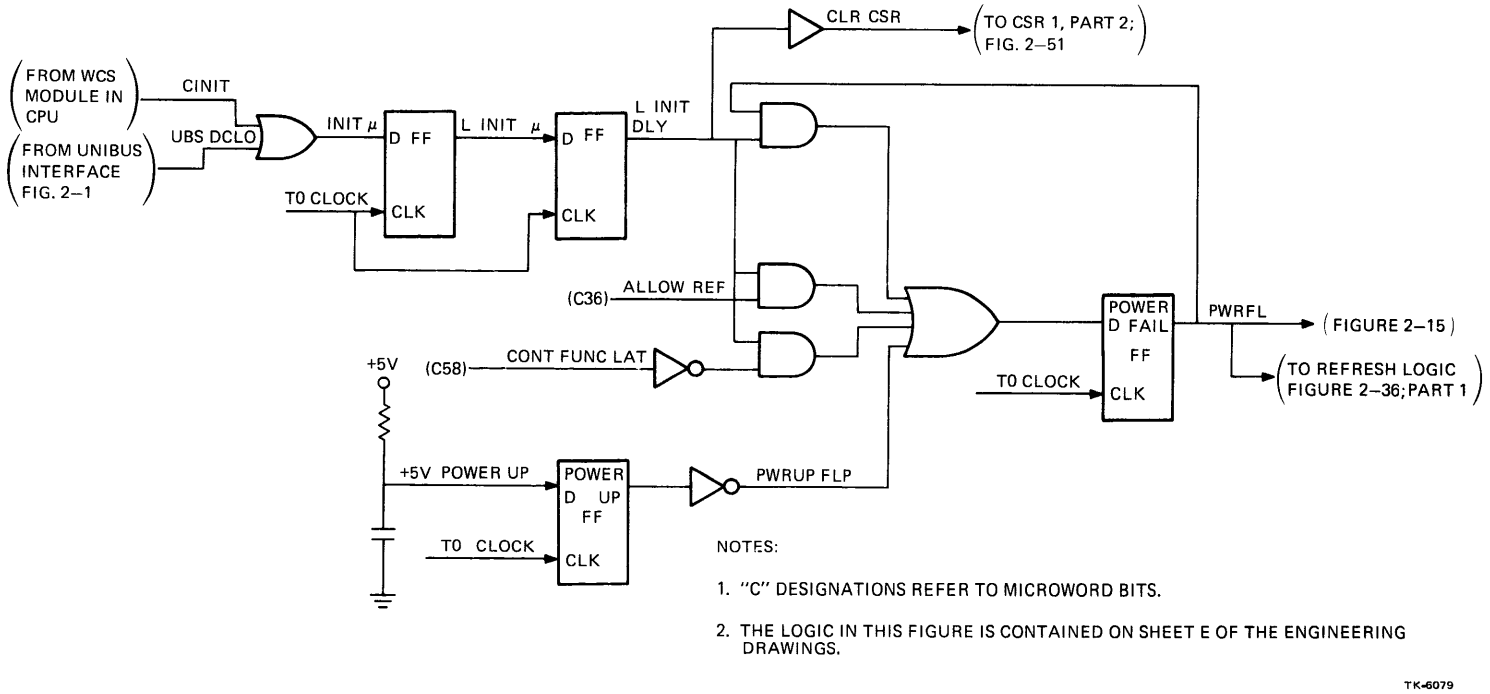


Figure 2-18 Power Fail Logic

2.4 ADDRESS TRANSLATION

2.4.1 General

The translation of virtual addresses from the CPU and addresses from the UNIBUS into physical addresses is accomplished by use of the virtual address register (VAR) and the translation buffer (TB). Figure 2-19 is a simplified block diagram of the translation function.

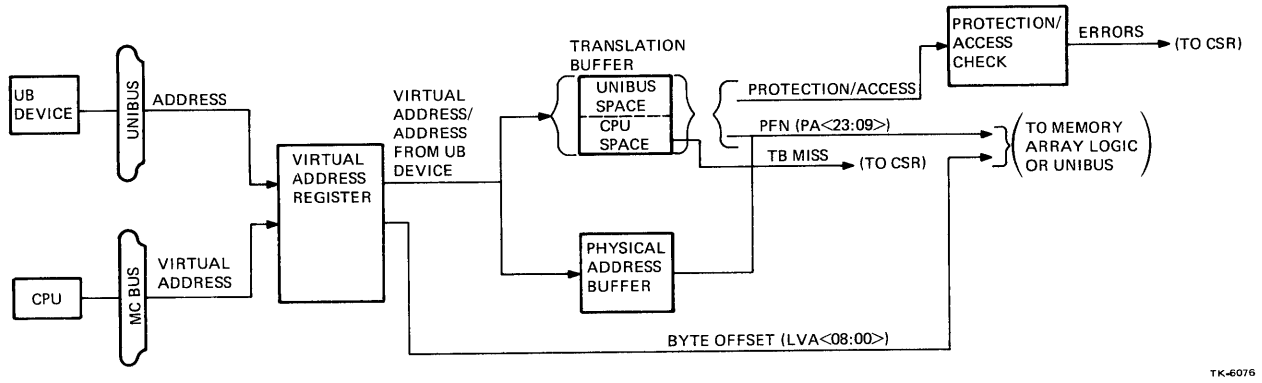


Figure 2-19 Simplified Block Diagram of Address Translation

The virtual address register selects either a virtual address from the MC bus or an address from a UNIBUS device. The address selected by the VAR is divided into two parts. The lower nine bits are the byte offset (index into the page frame) and are sent directly to the array logic (or the UNIBUS*). The higher order bits are used to address the translation buffer and select a TB entry.

The translation buffer is divided into two sections. One portion of the buffer is allocated to CPU entries. This area is addressed when a CPU translation is being executed. The other portion of the buffer is allocated to UNIBUS entries. This area is addressed when a UNIBUS translation is being executed.

A TB entry is made up of the page frame number (PFN) and protection/access bits. The PFN is sent to the array logic (or the UNIBUS*) along with the byte offset as the physical address. The protection/access bits are checked for illegal operational errors and, if any are found, the CPU is flagged and the appropriate error bit is set in the control/status registers.

During a CPU virtual address translation, the TB entry addressed by the VAR output may not be present in the translation buffer. When this occurs, a TB MISS error bit is set in one of the control/status registers (CSRs) and the CPU is flagged that the TB entry associated with the referenced page is not in the translation buffer. The CPU uses the virtual address to get the referenced page and its associated page table entry (PTE), to place the page into the memory arrays, to enter the corresponding TB entry into the translation buffer, and to retry the address translation. A TB miss can only occur during a CPU virtual address translation as all 512 possible UNIBUS addresses map to one of the 512 UNIBUS entries in the translation buffer.

The CPU can place a physical address on the MC bus which would not need address translation. When this is done the address from the VAR is the physical address. A physical address buffer is used to bypass the translation buffer and transfer the physical address directly to the memory array logic or the UNIBUS. The byte offset is still obtained from the VAR.

*When the UNIBUS is the CPU target.

2.4.2 CPU Virtual Address Translation

Figure 2-20 is a flow diagram of a CPU virtual address translation. Figure 2-21 is a block diagram of the logic involved in the translation. Paragraphs 2.4.2.1 through 2.4.2.6 relate to the block diagram to provide a functional explanation of the logic. The flow diagram is used mainly as a supplement to show the steps of an address translation and the sequence in which they occur.

2.4.2.1 Virtual Address (Figure 2-22) – The virtual address applied to the VAR from the MC bus is shown in Figure 2-22A. The address is shown in segments with each segment performing a specific function within the translation logic.

Bits <01:00> select the byte being addressed within the selected longword. This is accomplished in the data rotator and the data rotator control logic.*

Bits <08:02> select the longword being addressed within the selected page. This is accomplished in the memory address mux.*

Bits <31>, <14:09> select the page by addressing the translation buffer and retrieving the associated PFN. The memory logic uses the PFN to select the page.

Bits <30:15> is the tag associated with the TB entry. When the TB entry is loaded into the translation buffer, the tag is stored at the same address in the tag store area. When the entry is addressed, the tag is retrieved and compared with bits <30:15> of the current virtual address. They must match to affect a valid translation.

When the CPU presents a physical address to the memory controller, bits <23:09> are used as the page address.

2.4.2.2 Virtual Address Register/VAR Counter – The true state of CPU GRANT negates UB IDLE causing VAR MUX SEL to negate. When VAR MUX SEL negates, the virtual address register selects the 32-bit virtual address from the MC bus. T0 CLOCK loads the virtual address into the VAR.

VAR output bits LVA <01:00> select the byte location within the addressed longword. They are sent to the data rotator control logic to specify the amount of byte rotation required and to the UNIBUS interface as part of the UNIBUS address when the UNIBUS is the CPU target.

VAR output bits <30:02> are loaded into a VAR counter by VAR LOAD from the controller microword. The counter is used to increment the address to the next longword location when a two-cycle access is being executed. As the least significant bit in the counter is LVA <02>, incrementing the counter increases the virtual address by four to the next longword location. Negating VAR LOAD and asserting VAR MUX SEL increments the counter.

Counter bits LVA <08:02> are monitored by logic within the VAR counter. All seven bits true indicates an address of 128 (decimal) which is the last location on the page. When this occurs the counter asserts PAGE BOUNDARY indicating that the last longword in the page is being addressed.

If the operation is a two-cycle access, the second cycle would be addressing the first longword in a new page. If the two-cycle operation is a write, the WR ACROSS PG ERR bit is set in CSR1, ERR SUM is asserted to the CPU, and the memory controller aborts the operation. Thus the first longword is not written because access to the new page has not been checked and writing the second longword may not be allowed.

---

*When storage area is an array board. If the UNIBUS is the CPU target the bits are transferred to the UNIBUS.

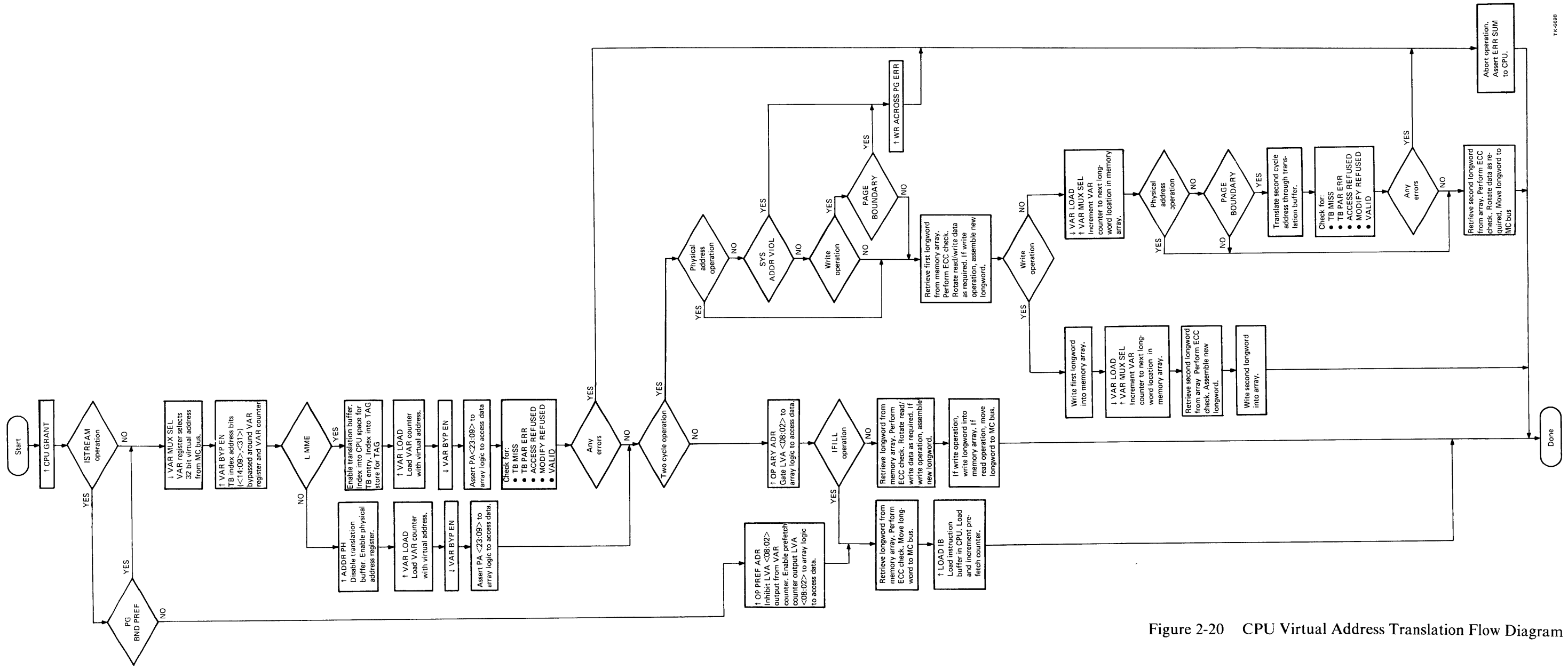


Figure 2-20 CPU Virtual Address Translation Flow Diagram

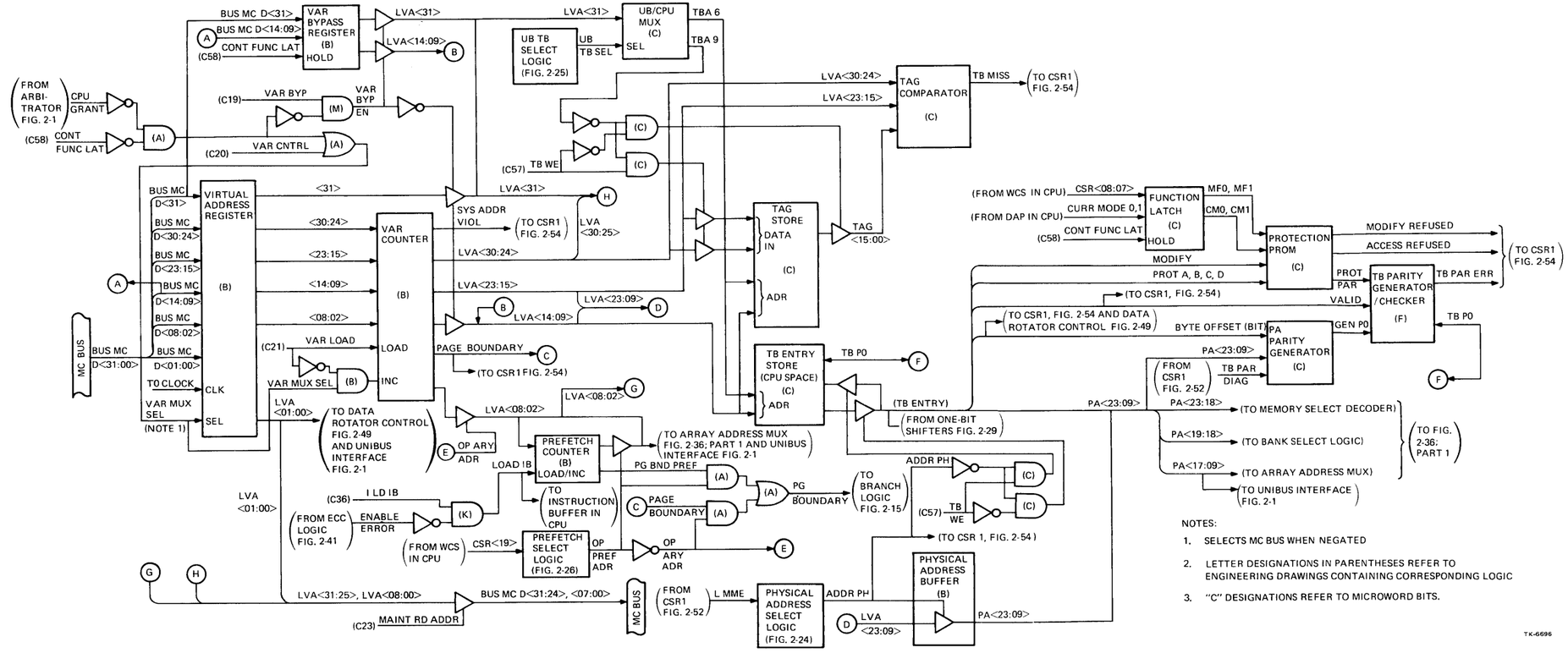


Figure 2-21 CPU Virtual Address Translation Block Diagram

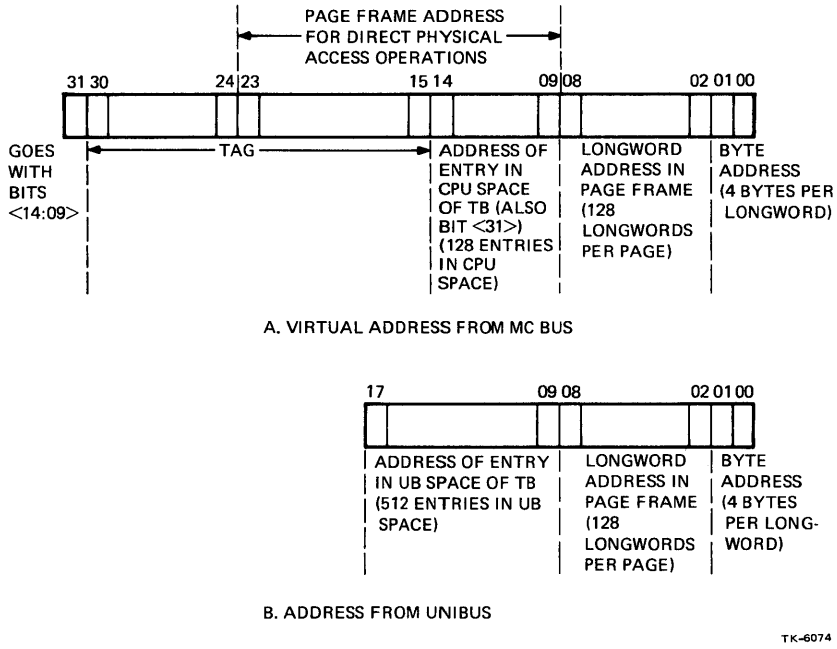


Figure 2-22 CPU/UNIBUS Address Segments

If the two-cycle operation is a read, the operation continues and the new virtual address is checked for access. If access to the new page is not allowed, ERR SUM is asserted and the operation is aborted. A read operation is allowed to continue across a page boundary, while a write operation is not, because a read operation does not destroy old data. Refer to the address translation flow diagram (Figure 2-20) where the sequence for a two-cycle operation is illustrated.

Other logic within the VAR counter monitors bits LVA <29:02>. All 28 bits true asserts SYS ADDR VIOL indicating that the last longword in the system region is being addressed. If a two-cycle operation is being executed, the second cycle would cross a system boundary which is not allowed for any type of operation. Thus, if SYS ADDR VIOL asserts when a two-cycle operation is initiated (either read or write), the WR ACROSS PG ERR bit is set in CSR1, ERR SUM is asserted to the CPU, and the memory controller aborts the operation (Figure 2-20).

Bits LVA <08:02> (byte offset) from the VAR counter are coupled to the array logic and to the UNIBUS interface. Bits LVA <14:09> from the VAR counter and bit LVA <31> from the VAR are coupled to the translation buffer as the index address into the buffer. (Bit LVA <31> is routed to the translation buffer via the UB/CPU mux.) Bits LVA <30:15> from the VAR counter are coupled to the tag store area of the TB (Paragraph 2.4.2.4.1).

2.4.2.3 VAR Bypass – Bit LVA <31> from the VAR and bits LVA <14:09> from the VAR counter address the translation buffer and select the desired TB entry. A VAR bypass register routes the address bits around the VAR and VAR counter and applies them directly to the translation buffer. By applying them directly to the buffer, the TB entry can be retrieved and the parity, access, and protection checks started without waiting for the address bits to work through the VAR and the VAR counter.

VAR BYP EN enables the VAR bypass. VAR BYP is asserted by the memory control store PROM causing VAR BYP EN to come true. VAR BYP EN inhibits LVA <31> coming from the VAR and LVA <14:09> coming from the VAR counter, and substitutes LVA <31>, <14:09> from the VAR bypass register. The VAR bypass register is a flow-thru register, thus the TB index address bits on the MC bus are immediately applied to the translation buffer. The bits are not subject to the delay of being clocked into bit registers. CONT FUNC LAT holds the flow-thru register open.

The address bits are also clocked into the virtual address register by T0 CLOCK and then loaded into the VAR counter by VAR LOAD. Thus when VAR BYP negates and the VAR bypass is inhibited, bit LVA <31> is in the virtual address register and bits LVA <14:09> are in the VAR counter. If the operation requires a second cycle, the counter is incremented to address the next longword location.

2.4.2.4 Translation Buffer

2.4.2.4.1 TB Space – Figure 2-23 illustrates the space allocation within the translation buffer. The TB entries are 23 bits long and are stored in a 1K area. The upper 512 locations are for UNIBUS entries. The lowest 128 locations are for CPU entries. The remaining 384 locations are not used.

The TB entry store area is enabled by the negated state of ADDR PH obtained from the physical address select logic (Figure 2-24). When memory management is enabled by the CPU (L MME true), ADDR PH goes false and enables the TB entry store area. The TB will have either its data in path or its data out path enabled depending on the state of TB WE from the memory control store PROM.

The translation buffer is addressed by bits LVA <14:09> and TBA <9:6>. TBA <9:6> are obtained from a UB/CPU mux. The mux select signal (UB TB SEL) is a function of the three special function bits (SPF <2:0>) from the memory control store PROM (Figure 2-25). UB TB SEL is false for CPU translations thereby causing the mux to couple LVA <31> to the TBA6 address line. TBA <9:7> are grounded for CPU translations (Figure 2-23). Thus for a CPU translation, seven address bits (LVA <14:09, TBA <6>) index into the 128 locations of CPU space for the desired TB entry.

A tag store area consisting of 256 16-bit locations is used to store the tag associated with each TB entry in CPU space (Figure 2-23). The tag is bits LVA <30:15> of the virtual address used to load the entry into the translation buffer. Only 128 of the 256 tag locations are used, corresponding to the 128 TB entry locations. Thus each index address into CPU space locates a TB entry and also its associated tag.

The tag store area is enabled by a negated TBA 9 (ground) from the UB/CPU mux during CPU address translations. TB WE from the control store PROM enables either the LVA <30:15> input path or the TAG <15:00> output path.

When a TB entry is addressed during a translation, its associated tag is coupled into a tag comparator. Bits LVA <30:15> of the virtual address are also applied to the comparator. If LVA <30:15> match the 16 tag bits a TB hit is scored meaning that the TB entry for the virtual address has been loaded into the translation buffer. If the TB entry for the virtual address is not in the translation buffer, a match is not obtained and the comparator asserts TB MISS. TB MISS sets an error bit in CSR1 and asserts ERR SUM to the CPU. The CPU then functions to swap the desired TB entry into the translation buffer and try the translation again.

The TB entry selected by the index address is composed of 23 bits as shown in Table 2-6. The bit functions are described in Paragraphs 2.4.2.4.2 and 2.4.2.4.3.

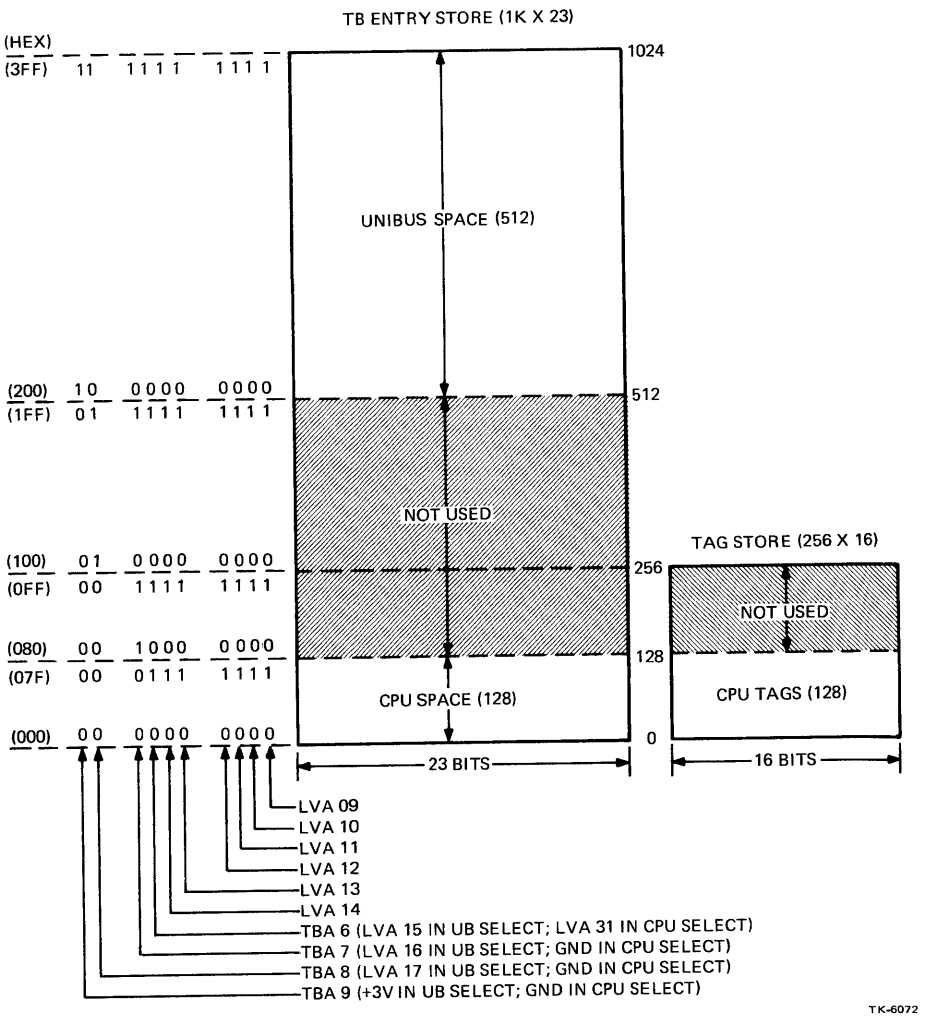


Figure 2-23 Translation Buffer Space Allocation

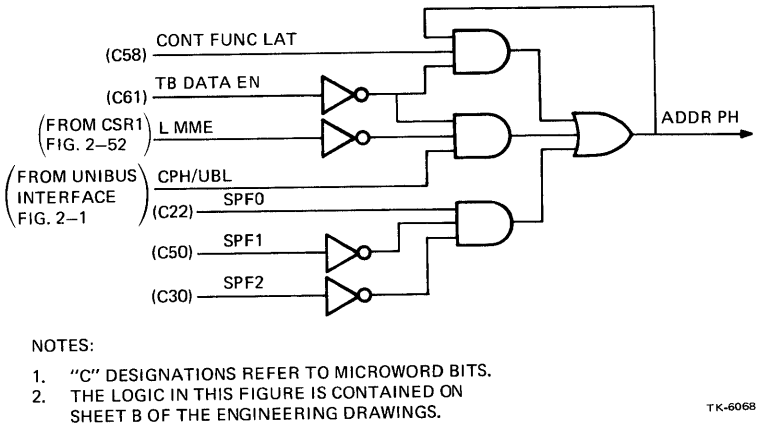


Figure 2-24 Physical Address Select Logic

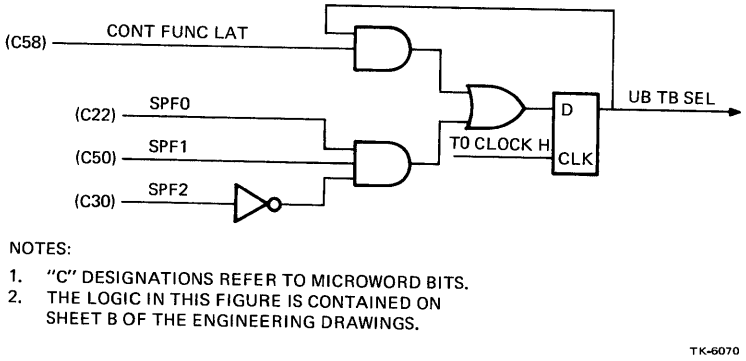


Figure 2-25 UB TB Select Logic

Table 2-6 TB Entry Bits (CPU Space)

Number of Bits	Item	Mnemonic	Function
15	Page frame number (PFN)	PA<23:09>	Selects physical storage location
4	Protection bits	PROT A,B,C,D	Allows no access, read only, or read/write according to operating mode
1	Modify bit	MODIFY	Indicates data in frame has been modified*
1	Valid bit	VALID	Indicates page frame is in the working set
1	Byte offset bit	BYTE OFFSET	Indicates TB data is a genuine entry written by the CPU
1	Parity bit	TB P0	Parity bit for TB entry

*This bit is set before the data in the frame is modified.

2.4.2.4.2 Protection – The protection PROM has a nine-bit address input as listed below.

.PROT A, B, C, D
.MODIFY
.MF <1:0>
.CM <1:0>

The four protection bits are part of the TB entry and specify the type of access allowed (no access, read only, read/write) for the four operating modes (user, supervisor, executive, kernel).

The MODIFY bit is part of the TB entry and indicates that the data in the referenced frame has been modified or is to be modified during the current translation.

MF <1:0> are memory function bits derived from CSR <08:07> respectively received from the CPU. CSR <08:07> specify the type of check to be made by the PROM. The CSR bit code is shown in Table 2-7.

CM <1:0> are current mode bits derived from CURR MODE <1:0> respectively received from the CPU. CURR MODE <1:0> specify the current operation mode of the system. The CURR MODE bit code is shown in Table 2-8.

The protection PROM has three outputs as listed below.

.ACCESS REFUSED
.MODIFY REFUSED
.PROT PAR

ACCESS REFUSED is asserted if the referenced page cannot be accessed in the current operating mode (Table 2-8) and for the type of check specified (Table 2-7). MODIFY REFUSED is asserted if the referenced page is to be written (write operation) and the MODIFY bit is not set. In this case, the CPU sets the MODIFY bit and the write operation is retried. Both ACCESS REFUSED and MODIFY REFUSED set error bits in CSR1 and assert ERR SUM to the CPU.

PROT PAR is a parity bit formed from the four protection bits and the MODIFY bit as a component of parity for the entire TB entry (Paragraph 2.4.2.4.3).

2.4.2.4.3 Parity Checking/Generation – Each TB entry has a parity bit (TB P0) that is checked for parity error when the entry is retrieved. As a TB entry is retrieved, the fifteen PFN bits (PA <23:09>) and the BYTE OFFSET bit are applied to a parity generator which develops a parity component signal (GEN P0) which is applied to the TB parity generator/checker.* Also applied to the parity generator/checker is the VALID bit from the TB entry and PROT PAR from the protection PROM. PROT PAR is a parity component derived from the four protection bits and the MODIFY bit (Paragraph 2.4.2.4.2).

Thus the TB parity generator/checker develops a composite parity signal from all bits in the TB entry. This signal is compared to the parity bit in the entry (TB P0). If they don't match, TB PAR ERR is asserted to CSR1 and ERR SUM is asserted to the CPU.

Each TB entry written into the translation buffer generates a composite parity signal in the same manner. This signal becomes TB P0 and is written into the buffer along with the TB entry.

*Another input bit (TB PAR DIAG) can be asserted for maintenance purposes.

Table 2-7 Memory Function Bit Code

Bits		Type of Check
CSR<08> MF1	CSR<07> MF0	
0	0	No check
0	1	Write check
1	0	Read check
1	1	Not used

Table 2-8 Current Mode Bit Code

Bits		Mode
CURR MODE 1 CM 1	CURR MODE 0 CM 0	
0	0	Kernel
0	1	Executive
1	0	Supervisor
1	1	User

2.4.2.5 Physical Address – During system boot-up, the CPU accesses page frames that reside in specific locations in physical memory. The address the CPU places on the MC bus is the physical address of the page frames, therefore no translation is required and the translation buffer is not used.

The physical address on the MC bus is clocked into the VAR by T0 CLOCK and then loaded into the VAR counter by VAR LOAD. Bits LVA <08:02> from the counter are applied to the array logic (or the UNIBUS) just as during a translation operation. Bits LVA <23:09> from the counter are applied through a physical address buffer and then to the physical address bus as PA <23:09> where they perform the same function as the PFN during an address translation operation.

The physical address buffer is enabled by the assertion of ADDR PH from the physical address select logic (Figure 2-24). ADDR PH asserts during a CPU operation (CPH/UBL true) when memory management is disabled (L MME false). Once asserted ADDR PH latches itself up so long as CONT FUNC LAT is asserted.

The assertion of ADDR PH disables the translation buffer which is not used during a direct physical access to memory. It follows that there are no protection/access checks.

2.4.2.6 Prefetch Counter – CPU.ISTREAM.REQ operations use a prefetch counter to provide the byte offset (LVA <08:02>) to the memory array logic in place of the byte offset from the VAR counter.

A READ.V.RCHK.IFILL instruction must be executed before a CPU.ISTREAM.REQ can be performed. (It is not necessary that READ.V.RCHK.IFILL be the preceding instruction.) During the READ.V.RCHK.IFILL instruction, the controller asserts LOAD IB to the CPU which loads the CPU instruction buffer with the longword read from memory. LOAD IB also loads the byte offset into the prefetch counter and increments the counter to the next longword location in preparation for a CPU.ISTREAM.REQ (prefetch) operation. The CPU.ISTREAM.REQ instruction is issued before the instruction buffer in the CPU is emptied. Thus the virtual address presented to the controller on the MC bus is to the last byte of the original longword location in memory. The byte offset from the prefetch counter, having been incremented by the LOAD IB signal, is addressing the next longword location. The CPU.ISTREAM.REQ transfer executes using the byte offset from the prefetch counter. Thus the next longword access is already in progress when the instruction buffer is emptied.

After the longword has been retrieved, the memory controller issues LOAD IB to the CPU to load the instruction buffer with the longword from the MC bus. The prefetch counter output wraps around and is loaded into itself by the LOAD IB signal. LOAD IB also increments the counter so that the prefetch byte offset now points to the next longword location in preparation for the next CPU.ISTREAM.REQ operation.

When an ISTREAM operation is reading out the last longword in a page, the prefetch counter is at a full count (LVA <08:02> all true). When the counter is loaded and incremented by LOAD IB, it asserts PG BND PREF indicating that the counter has reset to 0 and is prepared to address the first longword in the next page. With PG BND PREF true, the next ISTREAM instruction loads the new page address into the VAR, translates the address through the translation buffer, and performs the protection/access checks for the new page (Figure 2-20).

The prefetch function is enabled by OP PREF ADR from the prefetch select logic. OP PREF ADR gates the output from the prefetch counter onto the byte offset lines (LVA <08:02>) and inhibits the LVA <08:02> lines coming from the VAR counter.

The prefetch select logic (Figure 2-26) monitors CSR <19> from the WCS board in the CPU. When the CPU hardware initiates a CPU.ISTREAM.REQ. operation, bit CSR <19> is negated, causing OP PREF ADR to assert. OP PREF ADR is latched up by CPU GRANT and held true by CONT FUNC LAT.

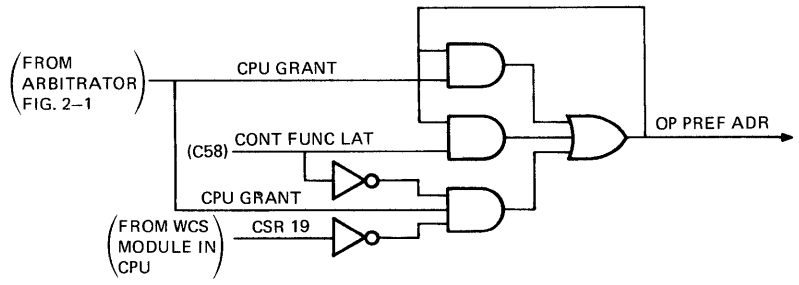


Figure 2-26 Prefetch Select Logic

2.4.3 UNIBUS Address Translation

UNIBUS address translation is similar to CPU virtual address translation but less complex. UNIBUS translations do not involve the VAR bypass, TAG store, physical address buffer, prefetch counter, or protection checks.

Figure 2-27 is a flow diagram of a UNIBUS address translation. Figure 2-28 is a block diagram of the logic involved in the translation. Paragraphs 2.4.3.1 through 2.4.3.3.2 relate to the block diagram in providing a functional explanation of the logic. The flow diagram may be used as a supplement to show the steps of an address translation and the sequence in which they occur.

2.4.3.1 UNIBUS Address – The address applied to the VAR from the UNIBUS is shown in Figure 2-22B. The address is shown in segments with each segment performing a specific function within the translation logic.

Bits <01:00> select the byte being addressed within the selected longword. This is accomplished in the data rotator and the data rotator control logic.*

Bits <08:02> select the longword being addressed within the selected page frame. This is accomplished in the memory address mux.

Bits <17:09> select the page frame by addressing the translation buffer and retrieving the associated PFN. The memory logic uses the PFN to select the page.

2.4.3.2 Virtual Address Register/VAR Counter – The negated state of CPU GRANT asserts UB IDLE causing VAR MUX SEL to assert. When VAR MUX SEL asserts, the virtual address register selects the 18-bit address input from the UNIBUS. T0 CLOCK loads the address into the VAR. Note that UNIBUS activity is not required to assert VAR MUX SEL. Thus, in the idle state, the virtual address register receives the UNIBUS address lines.

The UNIBUS address translation is initiated by the assertion of MSYN (Figure 2-27) which branches the microsequencer out of the idle state into the UNIBUS address translation routine.

VAR output bits LVA <01:00> select the byte location within the addressed longword. They go to the data rotator control logic to specify the amount of byte rotation required.*

VAR output bits <17:02> are loaded into a VAR counter by VAR LOAD from the controller microword. The counter is used to increment the address to the next longword location when a two-cycle access is being executed. As the least significant bit in the counter is LVA <02>, incrementing the counter increases the address by four to the next longword location. Negating VAR LOAD and asserting VAR MUX SEL increments the counter.

Note in Figure 2-27 that if a two-cycle write operation involves writing the last longword of a page and the first longword of the next page, writing the first longword is allowed before the next page is checked for errors (UB ERR SUM). This is in contrast to a CPU write operation where under the same conditions, the write operation is aborted.

Bits LVA <08:02> (byte offset) from the VAR counter are coupled to the array logic. Bits LVA <14:09> from the VAR counter are coupled to the translation buffer as part of the index address into the buffer (Paragraph 2.4.3.3.1). Bits LVA <17:15> from the VAR counter are coupled to the UB/CPU mux where they form the rest of the TB index address.

---

*The data rotator control logic also uses the BYTE OFFSET bit in determining the amount of data rotation.

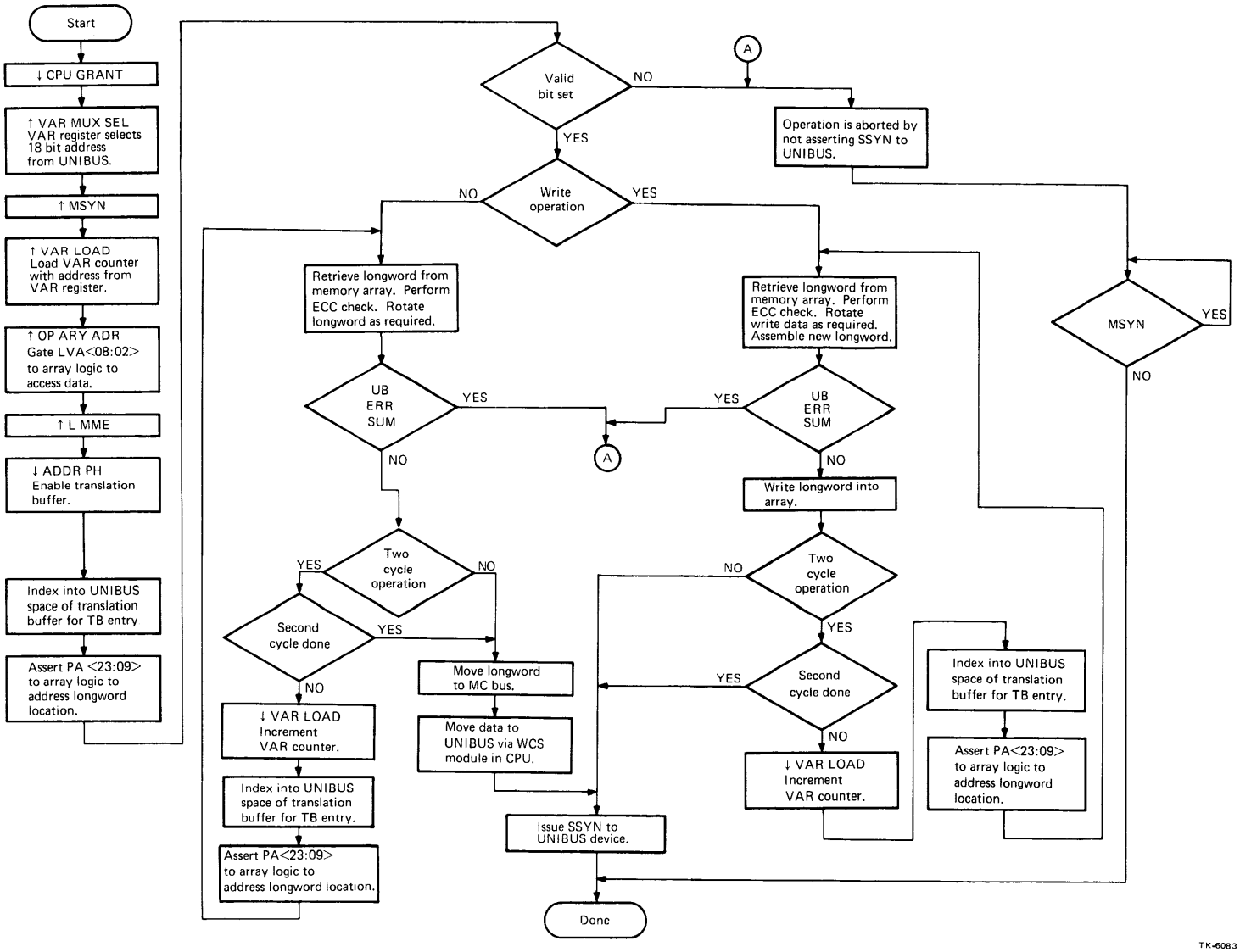


Figure 2-27 UNIBUS Address Translation Flow Diagram

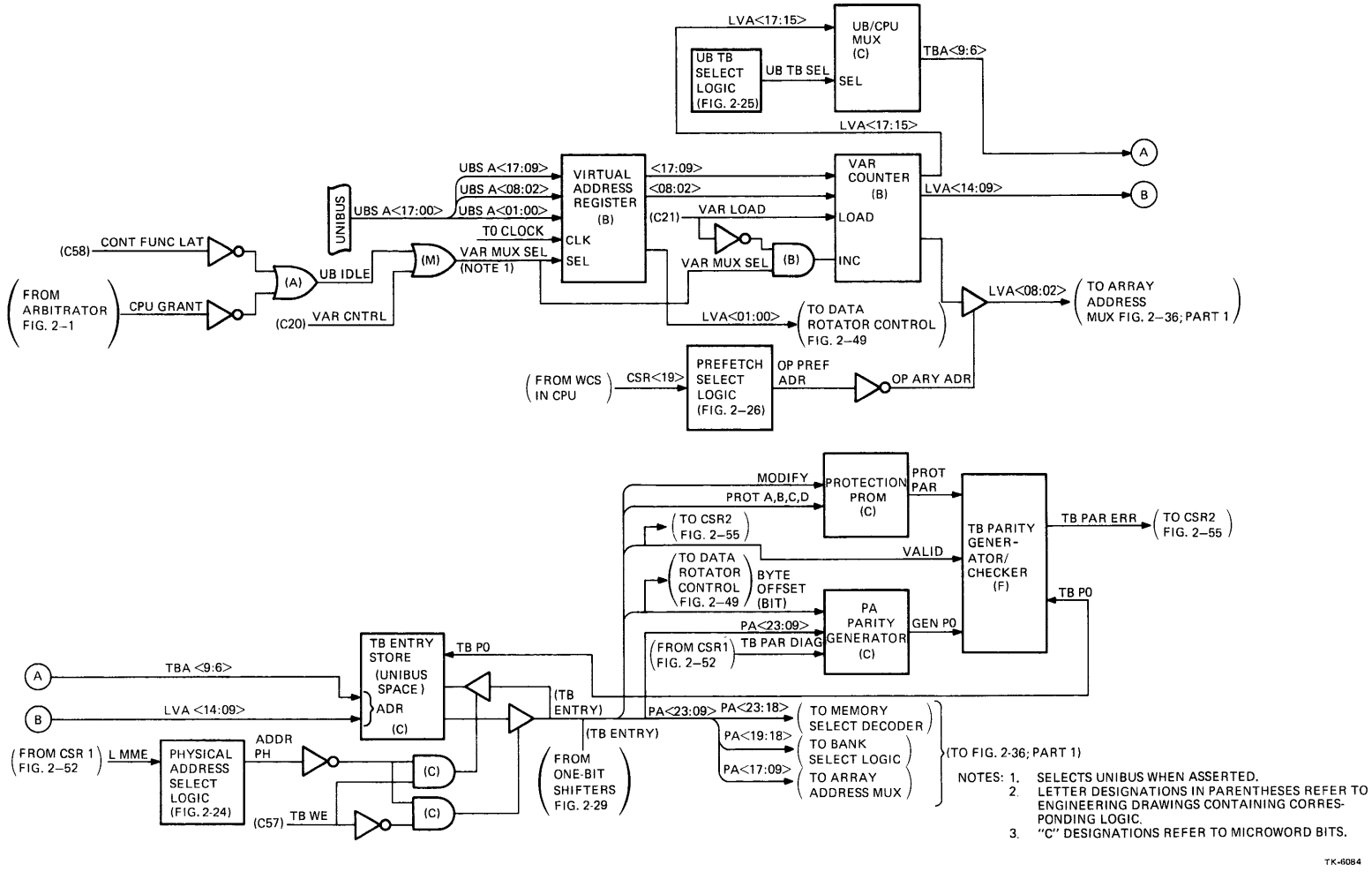


Figure 2-28 UNIBUS Address Translation Block Diagram

2.4.3.3 Translation Buffer

2.4.3.3.1 TB Space – Figure 2-23 illustrates the space allocation within the translation buffer. The TB entries are 23 bits long and are stored in a 1K area. The upper 512 locations are for UNIBUS entries. The lowest 128 locations are for CPU entries. The remaining 384 locations are not used.

The TB entry store area is enabled by the negated state of ADDR PH obtained from the physical address select logic (Figure 2-24). When the memory controller is being used by a UNIBUS device (CPH/UBL false), ADDR PH is false to enable the TB. The TB has its data-out path enabled by the negated state of TB WE from the memory control store PROM.

The translation buffer is addressed by bits LVA <14:09> and TBA <9:6>. TBA <9:6> are obtained from the UB/CPU mux. The mux select signal (UB TB SEL) is a function of the three special function bits (SPF <2:0>) from the memory control store PROM (Figure 2-25). UB TB SEL is true for UNIBUS translations thereby causing the mux to couple LVA <17:15> to the TBA <8:6> address lines respectively (Figure 2-23). Address line TBA9 is always true (+3 V) for UNIBUS translations. Thus for UNIBUS translations, ten address bits (LVA <14:09>, TBA <9:6>) index into the 512 locations of UNIBUS space for the desired TB entry.

The TB entry selected by the index address is composed of 23 bits as shown in Table 2-9. The bit functions are described in Paragraph 2.4.3.3.2.

Table 2-9 TB Entry Bits (UNIBUS Space)

Number of Bits	Item	Mnemonic	Function
15 (PFN)	Page frame number	PA<23:09> storage location	Selects physical
4	Protection bits	PROT A,B,C,D	Not used
1	Modify bit	MODIFY	Not used
1	Valid bit	VALID	Indicates TB data is a genuine entry
1	Byte offset bit	BYTE OFFSET	Indicates UNIBUS reference is to an odd byte location (i.e., 1001, 1003)*
1	Parity bit	TB P0	Parity bit for TB entry

*UNIBUS protocol allows UNIBUS address bits to present only even addresses to the memory controller during a word transfer.

2.4.3.3.2 Parity Checking/Generation – Each TB entry has a parity bit (TB P0) that is checked for parity error when the entry is retrieved. As a TB entry is retrieved, the fifteen PFN bits (PA <23:09>) and the BYTE OFFSET bit are applied to a parity generator which develops a parity component signal (GEN P0) for the TB parity generator/checker.* Also applied to the parity generator/checker is the VALID bit from the TB entry and PROT PAR from the protection PROM. PROT PAR is a parity component derived from the four protection bits and the MODIFY bit. The MODIFY and protection bits serve no function during a UNIBUS address translation except for the generation of the TB entry parity bit.

Thus the TB parity generator/checker develops a composite parity signal from all bits in the TB entry. This signal is compared to the parity bit in the entry (TB P0). If they don't match, TB PAR ERR is asserted to CSR2 and UB ERR SUM is asserted to the memory controller branch logic.

Each TB entry written into the translation buffer generates a composite parity signal in the same manner. This signal becomes TB P0 and is written into the buffer along with the TB entry.

The BYTE OFFSET bit in the TB entry must be set if a UNIBUS reference is made to an odd byte location during a word transfer. In accordance with UNIBUS protocol, byte transfers can be made to odd or even addresses but word transfers can be made only to even addresses. Thus, when a UNIBUS word transfer is to be made to an odd address (odd byte location), the BYTE OFFSET bit is set which effectively adds one to the even address taken from the UNIBUS address lines. BYTE OFFSET is sent to the data rotator control logic as a factor in determining the amount of byte rotation during a UNIBUS transfer.

2.4.4 Writing/Reading the Translation Buffer (Figure 2-29)

2.4.4.1 Writing the Translation Buffer – TB entries to be written into the translation buffer are placed on the MC bus by the CPU. The format of the entry as it appears on the MC bus is shown in Figure 2-30A. Note that the 15-bit PFN is located in bits <14:00>. The PFN as it is used on the physical address bus is located in bits PA <23:09>. In order to write the PFN into the TB so that it can be read out in the correct format, the entry must be shifted nine bits left from its position on the MC bus.

The entry is first fed into the data rotators where it is shifted one byte left and returned to the MC bus. The entry on the MC bus is now as shown in Figure 2-30B.

The TB entry is then divided into two parts and sent through one-bit, bidirectional shifters where it is shifted left one more bit. The shifter outputs are shown in Figure 2-30C where it is seen that the PFN is now in the desired bit location (<23:09>).

One shifter receives the PFN (BUS MC D <22:08>) and outputs onto bits <23:09> of the physical address bus. The other shifter receives the protection/access bits (BUS MC D <07:01>) and outputs onto the respective protection/access signal lines. The output of the two shifters is written into the translation buffer.

The PFN one-bit shifter is enabled by TB DATA EN + MAINT while the direction of data flow is determined by TB DATA DIR RD. Both signals are obtained from the memory control store PROM.

The protection/access one-bit shifter is identical to the PFN shifter except that TB DATA EN is the enabling signal. The shifters have different enabling signals for maintenance purposes.

*Another input bit (TB PAR DIAG) can be asserted for maintenance purposes.

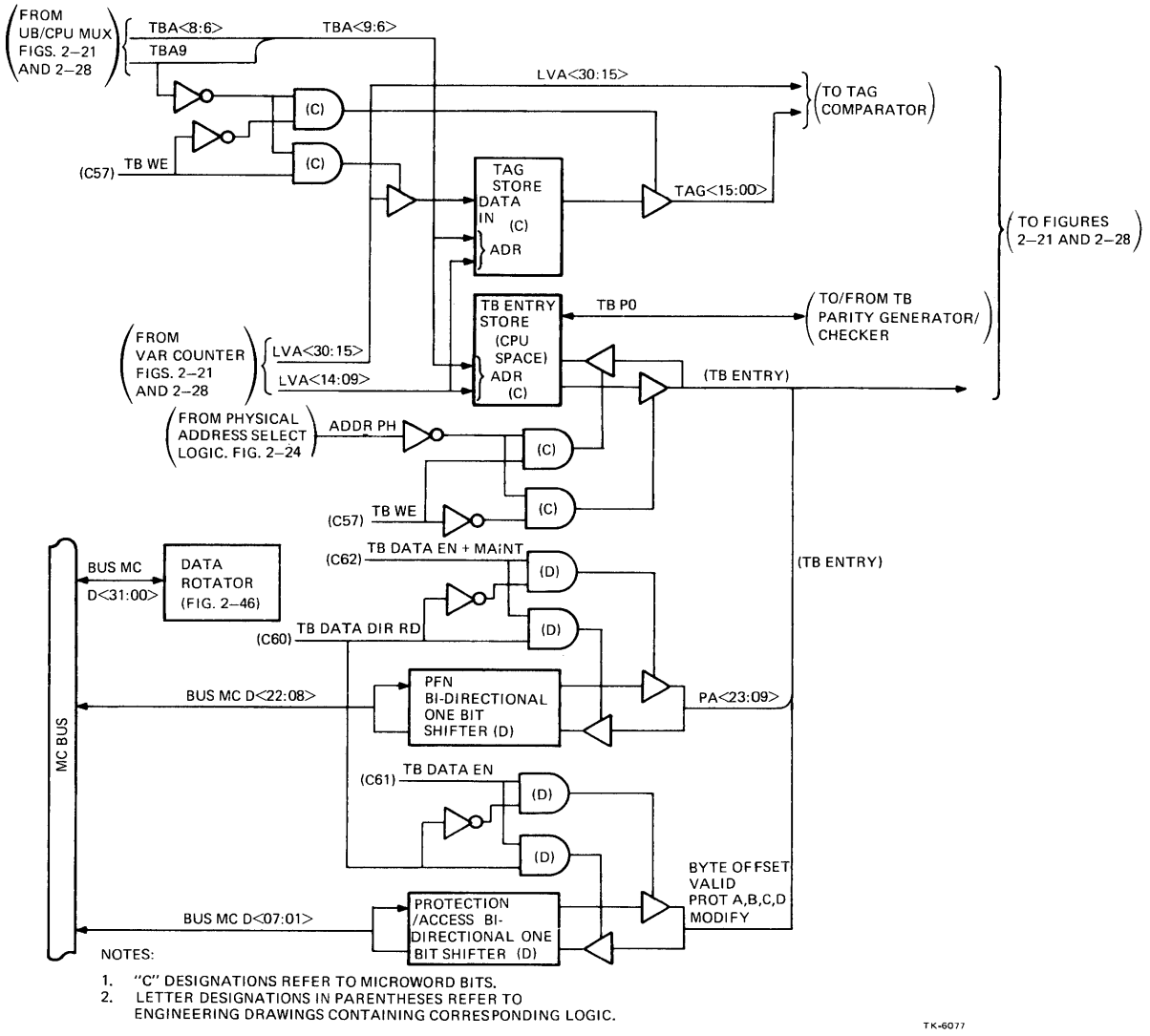


Figure 2-29 Translation Buffer Write/Read Block Diagram

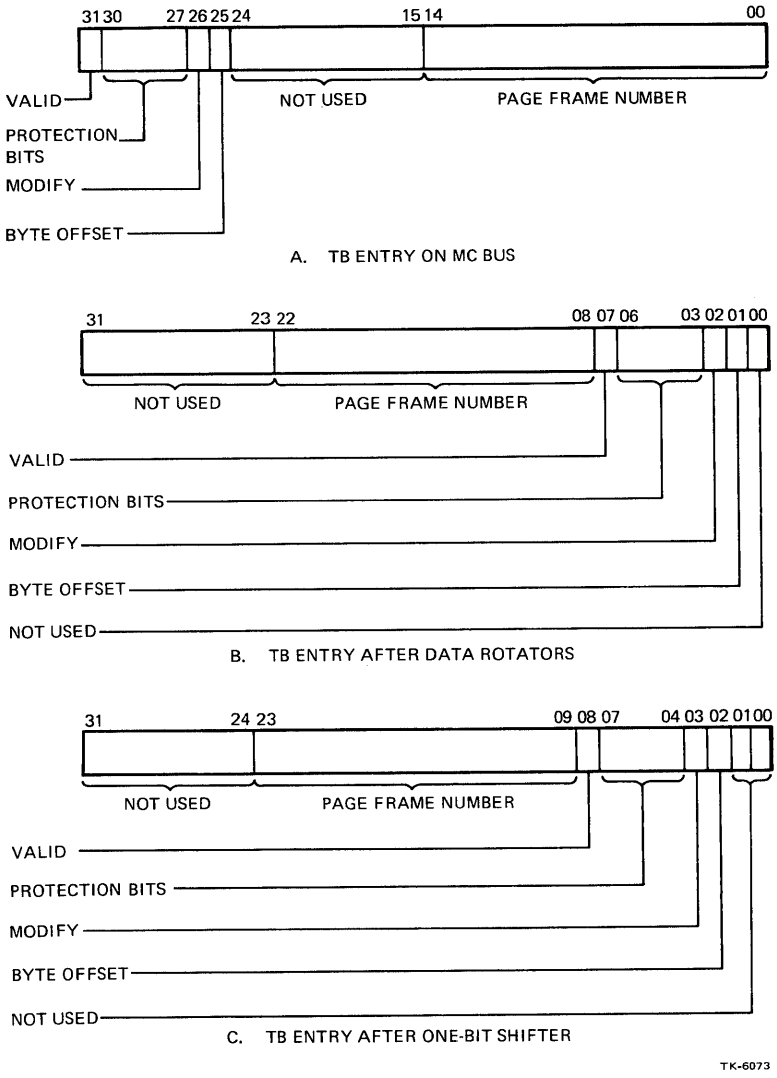


Figure 2-30 Translation Buffer Entry

The output of the two shifters is written into the translation buffer. To write the entry into the translation buffer, the control store PROM asserts TB WE. TB WE true enables the data path into the TB and inhibits the data path out of the TB.

The index address of the entry is the virtual address placed into the virtual address register from the MC bus. The address lines are LVA <14:09> from the VAR counter and TBA <9:6> from the UB/CPU mux.

If the TB entry is being written into CPU space, an associated tag is written into the same index address in the tag store area. The tag is virtual address bits LVA <30:15> from the VAR counter.

The TB entry generates a parity bit which is written into the TB along with the entry. The generation of the parity bit (TB P0) is described in Paragraphs 2.4.2.4.3 and 2.4.3.3.2.

2.4.4.2 Reading the Translation Buffer – At times the CPU will want to read a TB entry (e.g., for maintenance purposes). Reading the translation buffer is accomplished by retrieving the TB entry and asserting TB DATA DIR RD. The true state of TB DATA DIR RD establishes the shifter direction as toward the MC bus. The shifters are enabled by TB DATA EN and TB DATA EN + MAINT from the control store PROM.

The TB entry is shifted one bit right as it passes back through the shifters. This is followed by a one byte shift to the right in the data rotator. Thus the entry appears on the MC bus in the proper format for the CPU (Figure 2-30A).

2.5 PHYSICAL ADDRESS SPACE

2.5.1 General

Figure 2-31 shows the physical space that can be addressed from the memory controller. The physical address is 24 bits long, allowing a total address space of 16 megabytes. The 24 address bits are the PFN from the TB entry (PA<23:09>) and the byte offset from the virtual address bus (LVA<08:00>).

Most of the address space (15 megabytes) is memory space. The memory array cards are located in memory space. The sixteenth megabyte is I/O space and is divided into two parts. The first 3/4 of I/O space is UNIBUS adapter space. This space contains simulated UNIBUS registers that actually do not exist. The registers are simulated by the CPU to the VAX operating system in order to achieve software compatibility between the VAX-11/730 and other VAX-11 systems. When a reference is made to this area of memory, the memory controller asserts the ADAPT REG SEL error bit in CSR1 and then asserts ERR SUM to the CPU. The CPU reads CSR1, finds that UNIBUS adapter space has been referenced, and simulates the proper response to the system software.

The last quarter of I/O space is for UNIBUS addresses.

A memory select decoder on the MCT module receives the six most significant bits of the physical address (PA<23:18>) and outputs UB PH ADDR SEL, UB ADAPTER REG SEL, NXM, or MEM SEL "X", depending on the area of address space referenced (Figure 2-36; Part 1).

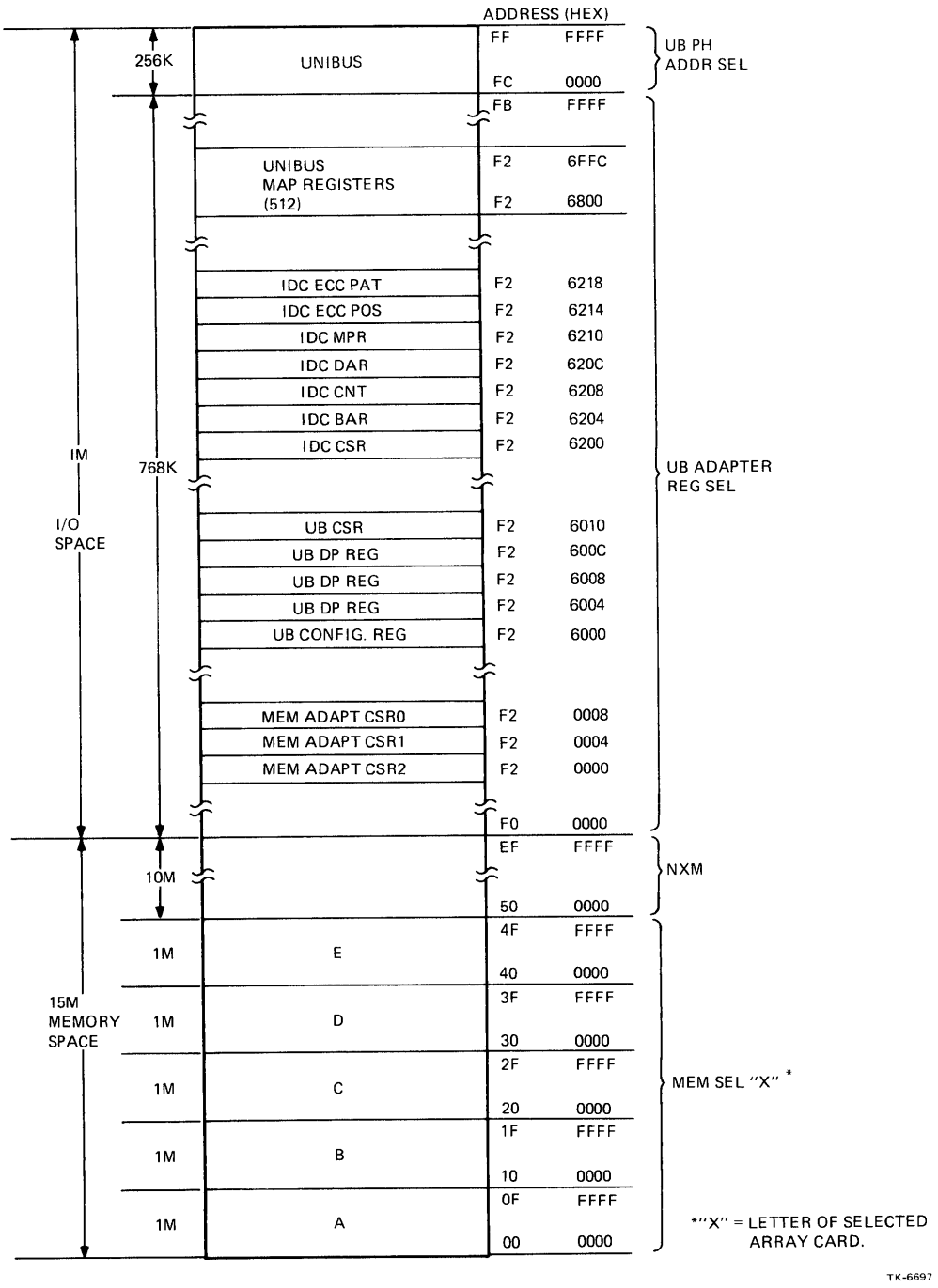


Figure 2-31 Physical Address Space

2.5.2 Memory Space

The 15 megabytes of memory space consists of the memory array modules plus nonexistent memory space. Up to five M8750 array modules may be used. Each module contains one megabyte of memory and must be placed in consecutive positions starting with the lowest address (position A). For example, if three memory cards are used, modules A, B, and C must be the three cards. This is necessary to prevent holes in real memory (i.e., all memory array space must be contiguous).

When a reference is made to memory space where an array card is located, MEM SEL "X" is asserted by the memory decoder ("X" is the particular module referenced). If no memory card is located at the referenced address, NXM is asserted to CSR1 as an error condition. Note that the area of nonexistent memory is ten megabytes plus whatever area of memory array space is not used.

2.5.3 UNIBUS Adapter Space

A memory reference from F0 0000 to FB FFFF causes the memory select decoder to assert UB ADAPTER REG SEL. The memory controller notifies the CPU that a reference has been made to the UNIBUS adapter area and the CPU responds to the reference. The CPU may read the memory controller translation buffer or the CSR hardware registers in order to formulate the response to the reference.

Only aligned longword references are allowed to UNIBUS adapter space.

Three memory adapter registers are simulated at addresses F2 0000, F2 0004, and F2 0008. The composition of the registers is shown in Figure 2-32.

NOTE

The memory adapter registers are designated as CSR0, CSR1, and CSR2. They are simulated by the CPU and are not to be confused with hardware registers CSR0, CSR1, and CSR2 that physically exist in the memory controller (Paragraph 2.9).

A UNIBUS configuration register is simulated at F2 6000. The register reads out 28 (hex) and ignores writes. Figure 2-33A illustrates the composition of the register.

Three UNIBUS data path registers are simulated at F2 6004, F2 6008, and F2 600C. The registers read out zeros and ignore writes (Figure 2-33B).

A UNIBUS control status register is simulated at F2 6010. Figure 2-33C illustrates the composition of the register.

NOTE

Seven software registers are simulated for the RB-730 Disk Subsystem. The seven registers are located in the address range from F2 6200 to F2 6218 inclusive. The registers are shown in Figure 2-31 as IDC (integrated disk controller) registers. Figure 2-31 also shows the specific address of each register. The composition of the registers is shown in Figure 2-34.

There are 512 UNIBUS map registers simulated from F2 6800 to F2 6FFC in 4-step increments.

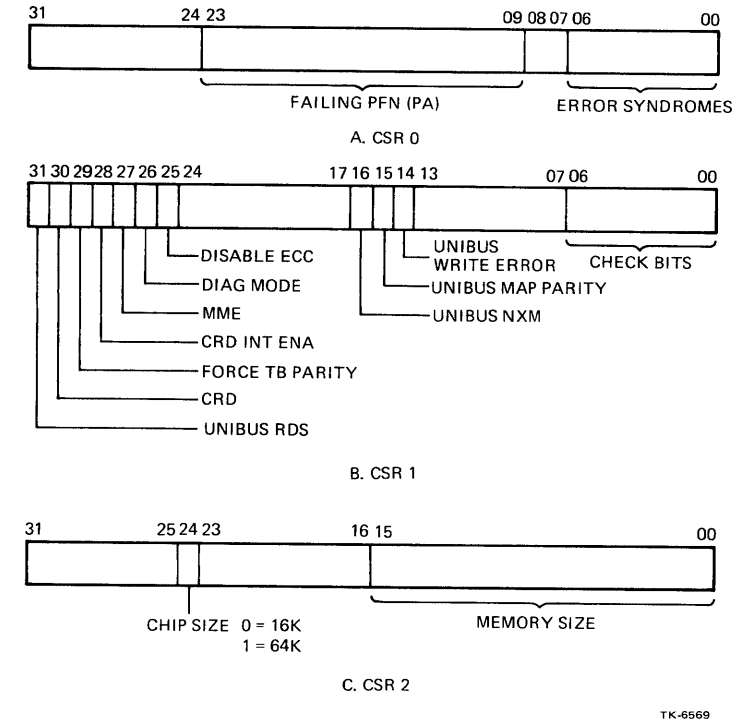


Figure 2-32 Memory Adapter Registers

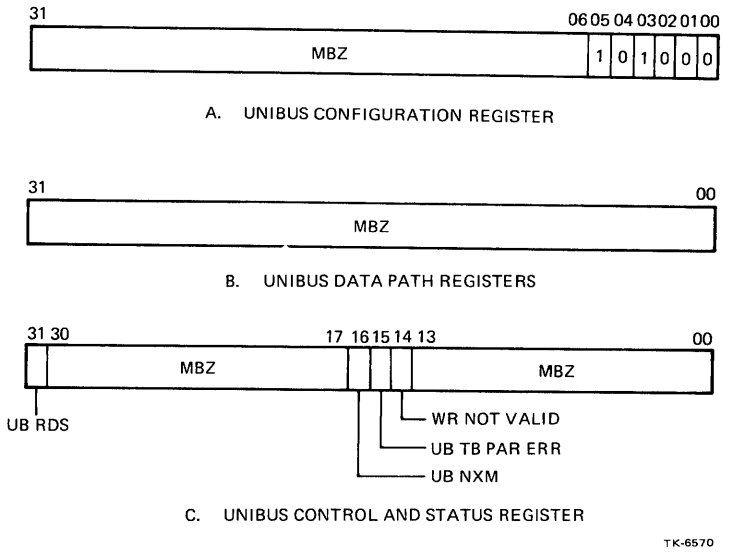
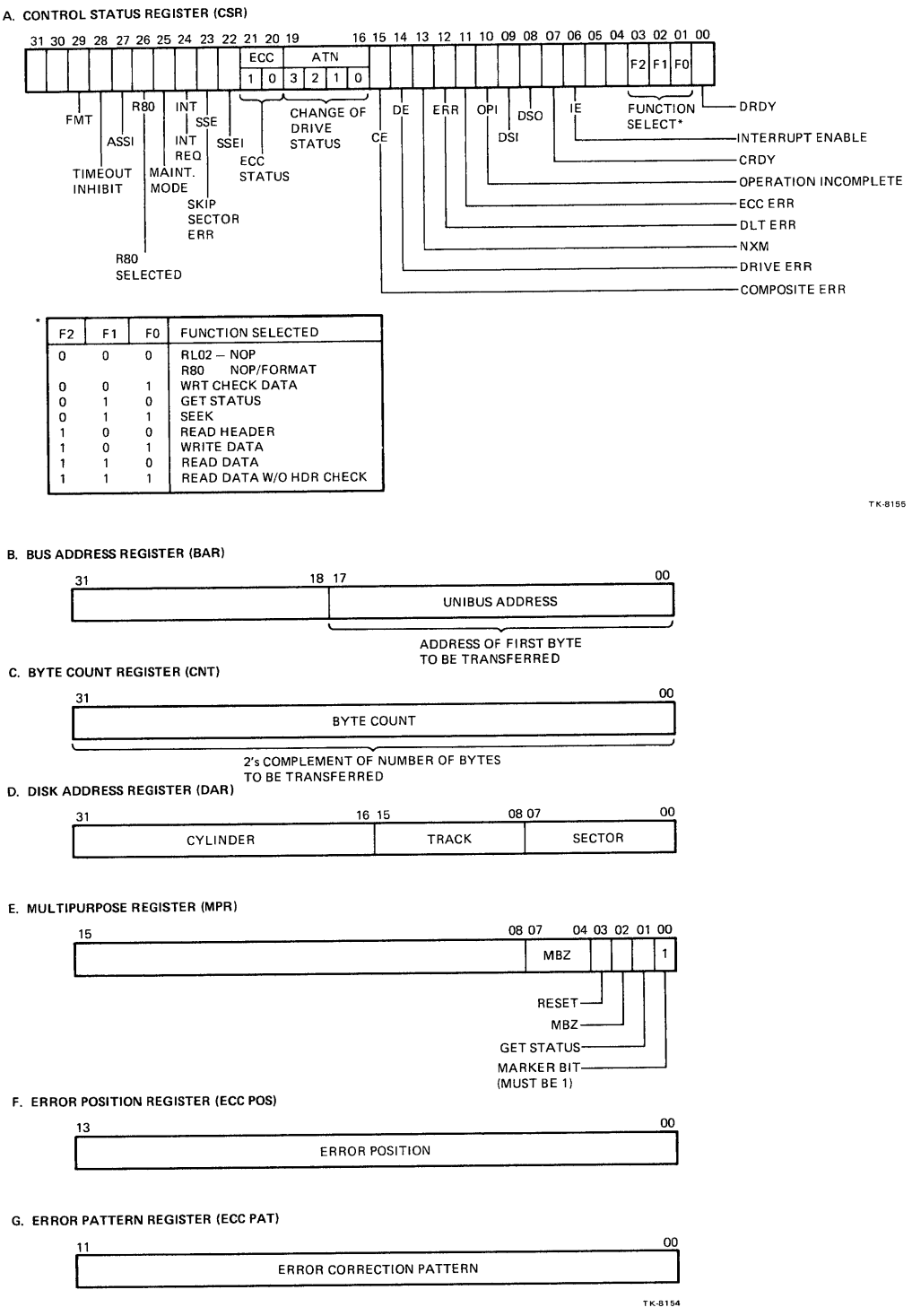


Figure 2-33 UNIBUS Adapter Registers



F2	F1	F0	FUNCTION SELECTED
0	0	0	RL02 – NOP R80 NOP/FORMAT
0	0	1	WRT CHECK DATA
0	1	0	GET STATUS
0	1	1	SEEK
1	0	0	READ HEADER
1	0	1	WRITE DATA
1	1	0	READ DATA
1	1	1	READ DATA W/O HDR CHECK

Figure 2-34 RB-730 Software Registers

2.5.4 UNIBUS Space

Address references from FC 0000 to FF FFFF are to real devices on the UNIBUS. When UNIBUS space is referenced, the memory select decoder asserts UB PH ADDR SEL which couples the translated physical address (PA<17:09>, LVA<08:00>) to the eighteen UNIBUS address lines.

2.6 MEMORY ARRAY READ/WRITE

2.6.1 General

Figure 2-35 illustrates the configuration of the 4164 RAM chips on the M8750 MOS memory array card. There are 156 RAM chips arranged in four banks of 39 chips each. Each RAM chip has a 256 × 256 matrix providing 64K (65,536) one-bit locations and, therefore, 64K 39-bit data locations per bank. The four banks provide the array card with a total of 256K 39-bit data locations. A 39-bit data location is made up of a 32-bit longword and seven check bits.

During a write to memory, input data longwords and check bits are applied to all four banks. Thus each of the 39 input bits is applied to four RAM chips, however, only one of the four banks is enabled at a time.

Eight address bits (A<07:00>) are applied to each of the 156 RAM chips. The address lines are multiplexed to first carry the row address and then the column address. A row address strobe (RAS) and a column address strobe (CAS) load the row and column addresses respectively into the RAMs. CAS is applied to all the RAMs and produces column addressing on all 156 chip arrays. A separate RAS strobe is generated for each bank and only one of these asserts during a memory read/write. The bank that has the row address strobed in thereby becomes the enabled one. A two-bit code (MA<15:14>) from the array bus specifies which bank is to be enabled. RAS TIM from the array bus times the RAS strobes.

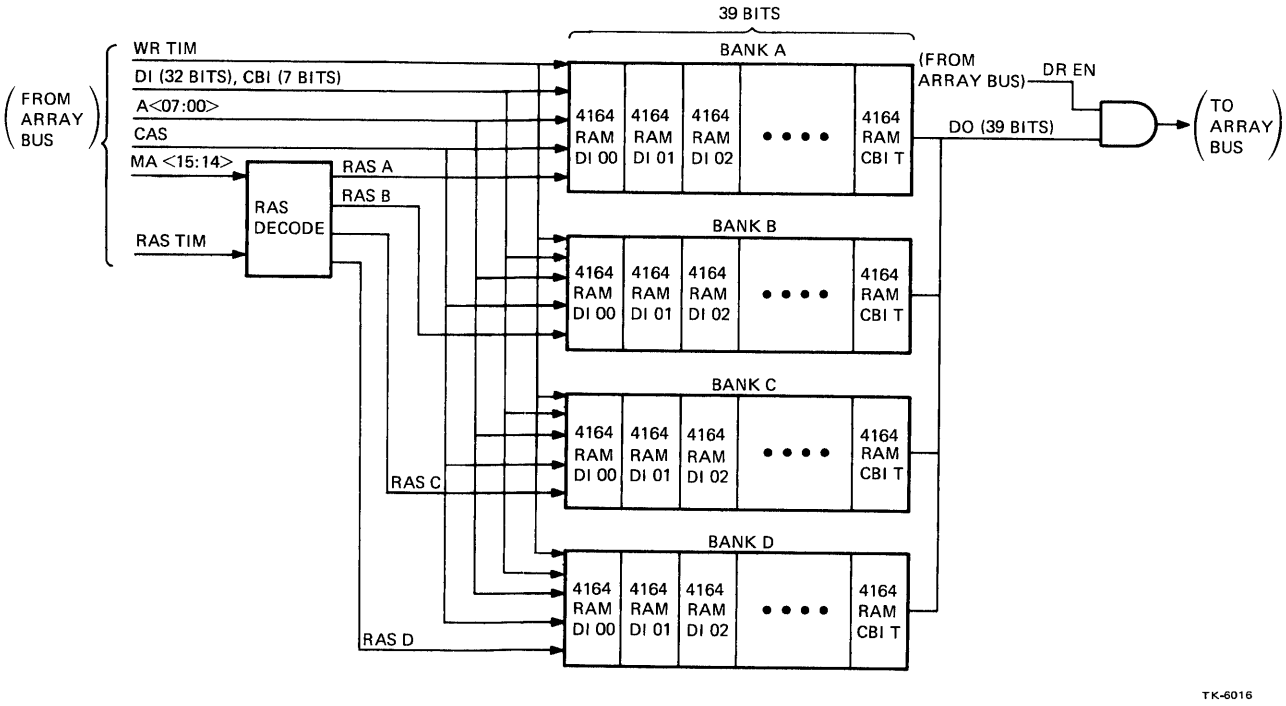


Figure 2-35 RAM Chip Configuration on M8750 Array Board

Data is written into and read out of the array as a full 32-bit longword (plus seven check bits). WR TIM clocks data into the bank selected by RAS during a write. DR EN allows data out to the array bus during a read.

Figure 2-36 is a block diagram of the memory array read/write function. The figure is divided into two parts: part 1 is the array read/write logic on the M8391 memory controller module; part 2 is the M8750 array module itself. Some signal mnemonics on the array module do not match the mnemonics on the memory controller module. Figure 2-37 illustrates the array bus signals and gives the signal mnemonic on the controller side of the bus and on the array side.

2.6.2 Memory Select (Figure 2-36)

The memory select decoder on the MCT module receives six address bits (PA<23:18>) from the physical address bus that specify which storage area is to be addressed. Possible areas are UNIBUS physical addresses (UB PH ADDR SEL), UNIBUS adapter space (UB ADAPTER REG SEL),* or an M8750 array card (MEM SEL A,B,C,D,E). If the memory select decoder senses a hole in memory, or a reference is made to a nonexistent location, NXM is asserted to indicate an error condition.

A fingerprint bit [ARRAY FP 4] from each array card is used by the memory select decoder to indicate if the card is present. Up to five array cards may be used. To prevent holes in memory, the last card used must be in the last slot of contiguous memory.

Each array card connects to a select line on the array bus. If one of the cards is selected by the memory select decoder, the decoder outputs MEM SEL on the corresponding select line. From the array bus, the memory select signal is input to the array card as INT BUS ADD MEM SEL which then becomes MEM SEL. MEM SEL enables all the I/O paths to and from the memory arrays.

2.6.3 Array Addressing (Figure 2-36)

Two bits from the physical address bus (PA<19:18>) select which of the four banks is to be addressed on the selected card. The two bits are placed on the array bus as BUS ARRAY B SEL <01:00> and appear on the array card as INT BUS MA <15:14>. The bits are decoded in the RAS decoder which asserts one of four RAS SEL outputs. The selected output is strobed by INT BUS RAS TIM to generate RAS A, B, C, or D. RAS is applied to the row address latch/decoders of the 39 RAM arrays of the selected bank.

The address of the location within the memory array to be read or written into, is obtained from the memory address mux. The address is in two segments: the memory page address from the physical address bus (PA<17:09>) and the longword address within the page from the virtual address register (LVA<08:02>). The two sets of mux inputs are PA<17>, <15:09> and PA<16>, LVA<08:02>. The mux defaults to the PA<16>, LVA<08:02> input, outputting the eight bits to the array bus as BUS ARRAY A <07:00>. The 8-bit address is input to the selected array card as INT BUS A <07:00> which then becomes A <07:00>.

When a location on the array card is to be accessed, bit 32 of the microcode (RAS EN) is asserted. This causes ARRAY RAS TIM to assert which is passed over to the array card as INT BUS RAS TIM and strobes A<07:00> into the row address latch/decoder of the selected bank. Thus PA <17>, <15:09> is used as the array row address. Forty-five nanoseconds later COLAD (column address) switches the address mux to the PA<16>, LVA <08:02> input. Thirty nanoseconds later CAS TIM asserts and is passed over to the array card as INT BUS CAS TIM. This input asserts CAS (A,B,C,D) which strobes A <07:00> into the column address latch/decoder of all four banks. Thus PA <16>, LVA <08:02> is used as the array column address. Figure 2-38 illustrates the address timing.

*Reference is to pseudo adapters. The VAX-11/730 has no UNIBUS adapters.

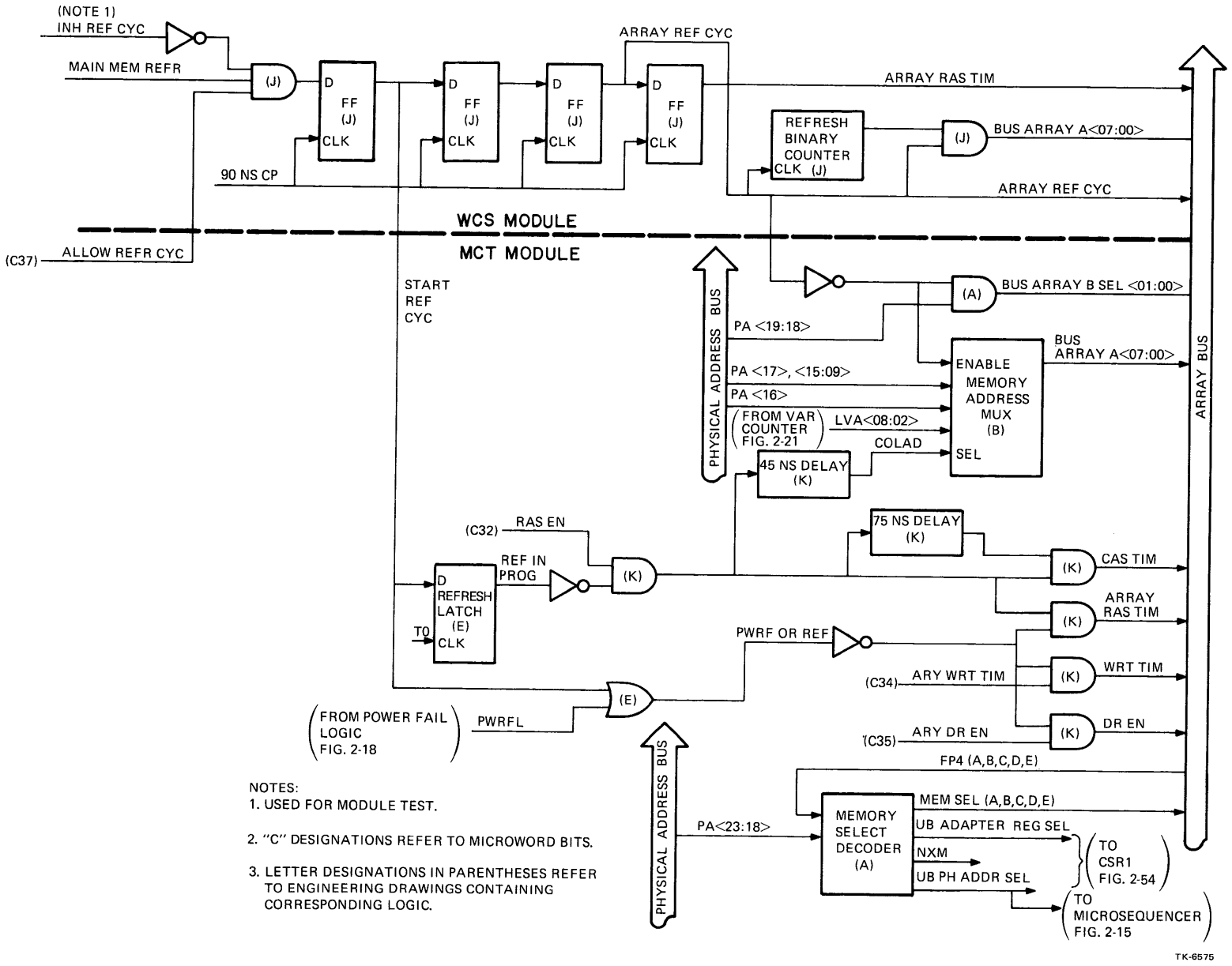


Figure 2-36 Memory Array Read/Write Block Diagram (Sheet 1 of 2)

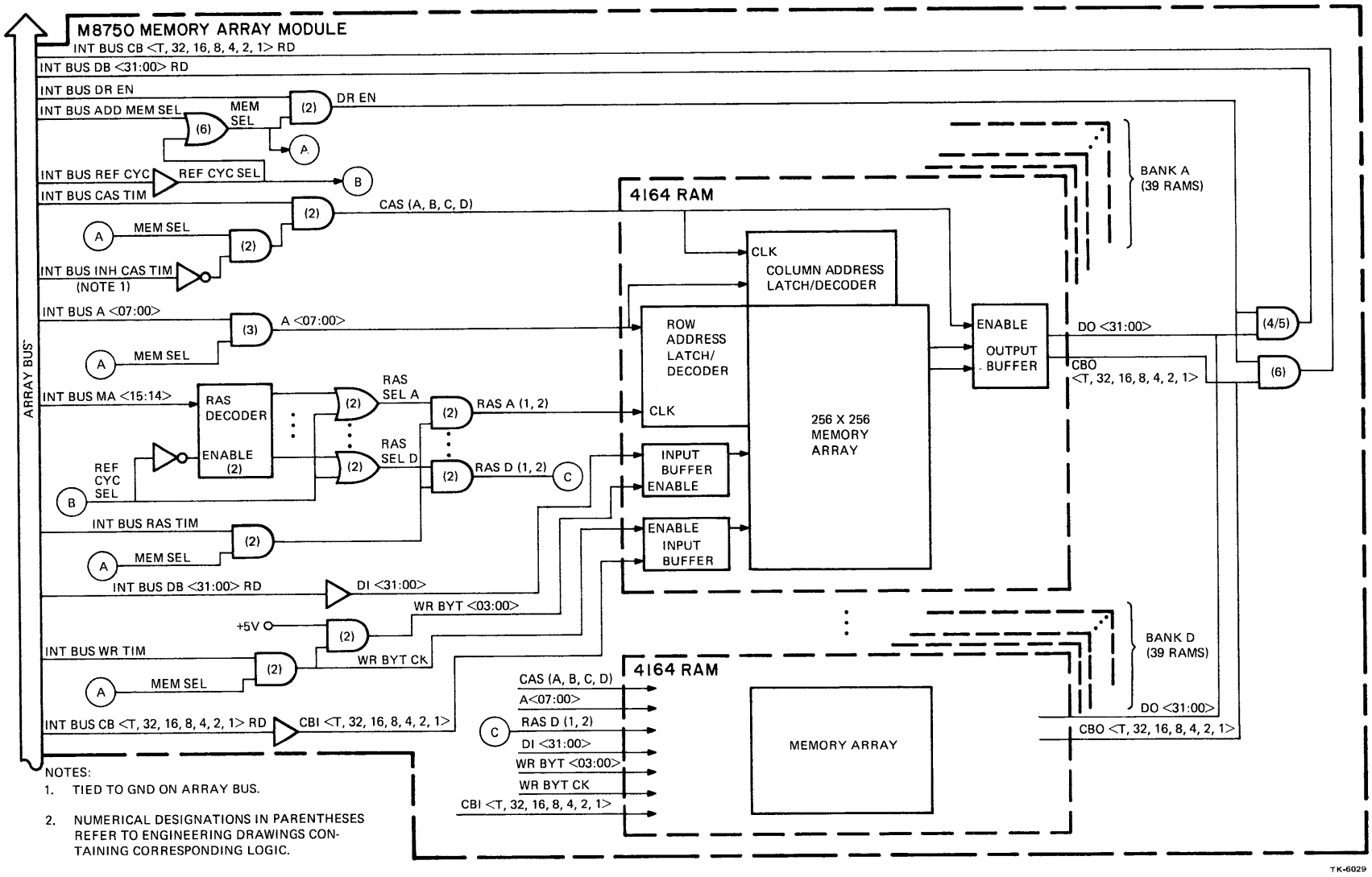


Figure 2-36 Memory Array Read/Write Block Diagram (Sheet 2 of 2)

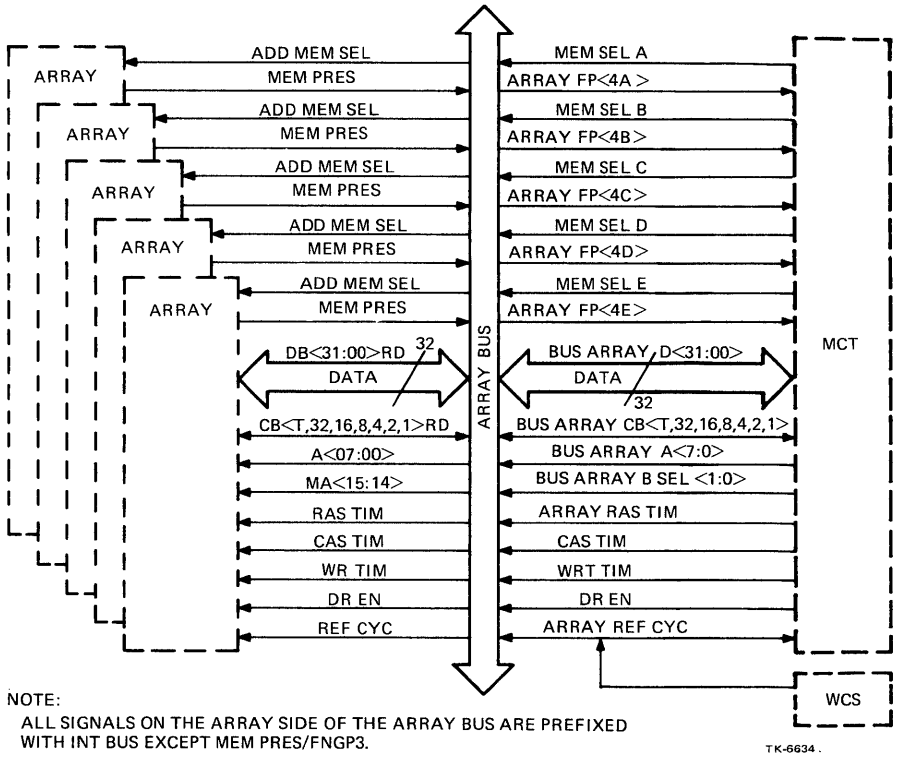


Figure 2-37 Array Bus Signals

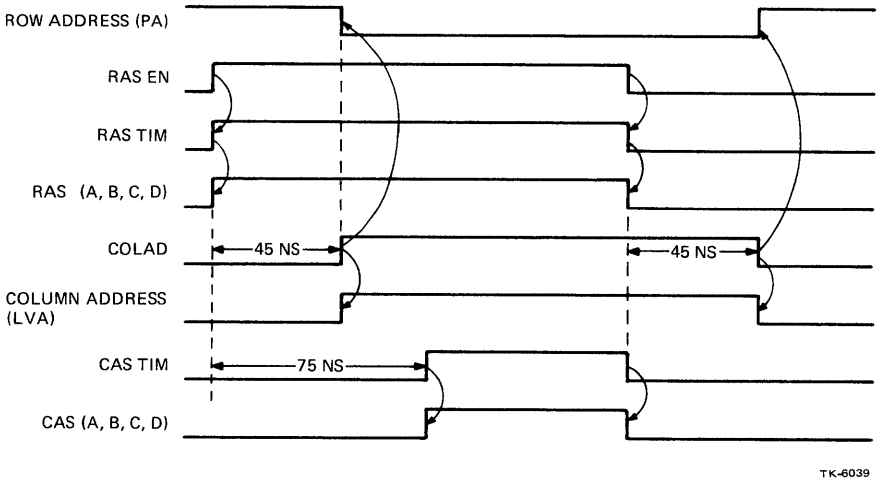


Figure 2-38 Array Address Timing Diagram

2.6.4 Refresh (Figure 2-36)

The 4164 RAM memory arrays must be refreshed at least every 4 ms in order to retain stored data. All four banks on all the array cards are refreshed at the same time. All of the 256 rows in a bank must be addressed and refreshed in no more than 4 ms. There is no column addressing as all the columns are enabled during refresh.

To perform a refresh, the normal array address and timing signals are removed and a new row address and a RAS strobe are obtained from refresh logic on the WCS board. No CAS strobe is generated during a refresh cycle hence the RAM output buffers are not enabled and no output occurs from the RAMs.

The refresh logic is on the WCS module in the CPU. A refresh cycle (to refresh one row) is triggered by MAIN MEM REFR. MAIN MEM REFR is a 90 ns pulse occurring at a 78 kc rate (Figure 2-39). Thus all 256 rows get refreshed in less than 3.4 ms. A refresh cycle takes 720 ns to complete. Refresh cycles are done in between reads and writes to the arrays.

If ALLOW REFR CYC (bit 37 of the memory controller microword) is true, MAIN MEM REFR triggers refresh logic on the WCS board that generates START REF CYC, ARRAY REF CYC, and ARRAY RAS TIM. (Figure 2-40 illustrates the timing of these signals.) START REF CYC asserts REF IN PROG and PWRF OR REF which inhibit read/write timing signals CAS TIM, ARRAY RAS TIM, WRT TIM, and DR EN. ARRAY REF CYC disables the memory address mux, inhibits the bank select gate, and enables the output of the refresh binary counter on the WCS board. The 8-bit binary output is placed on the array bus as BUS ARRAY A<07:00> and is applied to the array cards as the address of the next row to be refreshed. ARRAY REF CYC is also placed on the array bus and thence to all array cards as INT BUS REF CYC. On the array card it becomes REF CYC SEL which asserts MEM SEL thereby selecting each array card for a refresh. REF CYC SEL also enables all four RAS SEL outputs from the RAS decoder thereby selecting all four array banks. INT BUS RAS TIM is asserted by ARRAY RAS TIM from the WCS refresh logic and strobes RAS to the four array banks.

2.6.5 Data In (Figure 2-36; Part 2)

Thirty-two bits of data (INT BUS DB <31:00> RD) and 7 check bits (INT BUS CB <T,32,16,8,4,2,1> RD) are obtained from the array bus for writing into the memory arrays. They are applied to input buffers as DI <31:00> and CBI <T,32,16,8,4,2,1>. ARY WRT TIM (bit 34 of the microword) is placed on the array bus from the MCT board as WRT TIM, and then to the array card as INT BUS WR TIM. It then asserts WRT BYT (3:0)* and WR BYT CK to strobe the array input buffers and input the data into the arrays.

2.6.6 Data Out (Figure 2-36; Part 2)

The assertion of CAS (A,B,C,D) enables the RAM output buffers which couples DO <31:00> and CBO <T,32,16,8,4,2,1> from the addressed location in the memory arrays to output drivers. ARY DR EN (bit 35 of the microword) is placed on the array bus from the MTC board as DR EN and then to the array cards as INT BUS DR EN. It then asserts DR EN which enables the output drivers and places the 32 data bits and 7 check bits on the array bus.

2.6.7 Array Terminator

The address lines to the 4164 RAM array chips are connected through diodes to a terminating network (TERM A). TERM A is at +0.75 V and is obtained from a transistor voltage divider which functions as a +0.75 V voltage source. The voltage source and the diodes prevent any negative excursions on the address lines.

---

*Input data is always written as a 32-bit longword. INT BUS WR BYT (3:0) are all tied to +5 V on the array bus.

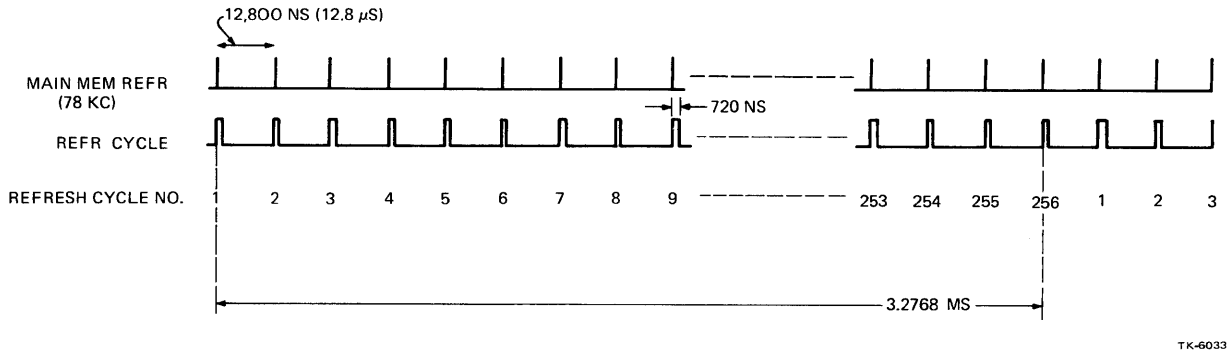


Figure 2-39 Rate of Refresh Cycles

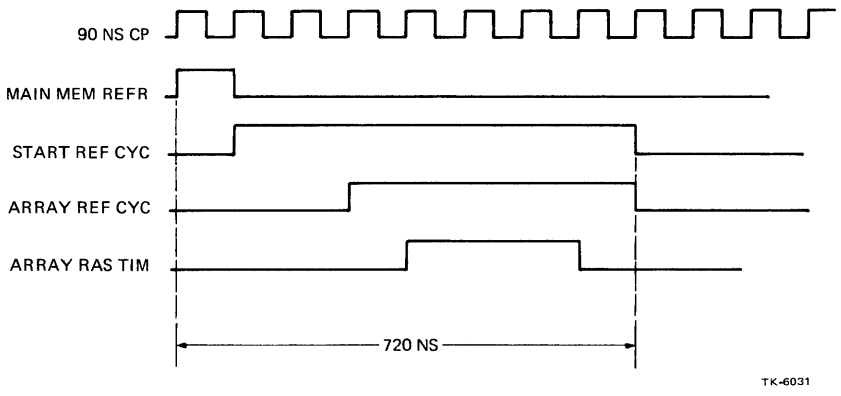


Figure 2-40 Refresh Cycle Timing Diagram

2.7 ECC (ERROR CHECKING/CORRECTION)

2.7.1 Read Array/ECC Check (Figures 2-41 and 2-42)

A read array/ECC check operation is performed in two different instances with the operations being almost identical in both cases. The first case is the reading of data from the memory array out to the UNIBUS or the CPU. When data is to be read out to the UNIBUS or the CPU, the data is read out of memory, an ECC check is performed, and the data is placed on the array bus for transfer to the data rotator. The second case is the writing of data from the CPU or the UNIBUS into the memory array. In this case the memory location is read out and an ECC check is performed*; however this time the data is left in the ECC data out latch (not placed on the array bus) where the write data will modify or replace it. The following discussion applies to both types of reads except where noted.

All data read out of the memory array undergoes an ECC check. If the ECC check shows a single bit error, the data is corrected and the error is reported to the CSR register as a CRD error. If the check indicates that more than one bit is in error, the data error is uncorrectable. The data is still read out to the array bus and the uncorrected error is reported to the CSR register as an RDS error. If a write from the CPU or UNIBUS is being performed, the writing of the new data is aborted and the old data is written back into the memory array.

A 39-bit word (32 data bits and 7 check bits) is read from the M8750 array card and placed onto the array bus. The data bits are latched into the data in latch by LAT DAT IN and the check bits are latched into the check bit latch by LAT CB REG. The output of the data in latch is then applied to the check bit generator where new check bits are generated. In the error checking mode (GENERATE false), the output of the check bit latch is applied to XOR gates where it is compared with the generated check bits. If the generated check bits match the check bits read out of memory, there are no XOR outputs (syndromes) asserted indicating a no error condition. If the check bits do not match, one or more syndromes are asserted. The output from the XOR gates is applied to a syndrome decoder. The decoder asserts ERROR to the CSR register and decodes the syndromes to determine if the error is a single bit error (correctable) and if so, which bit is in error. The decoding of the syndromes is shown in Table 2-10 followed by a discussion of each case.

If no syndromes are generated, the data from the data in latch passes through the correction logic and is latched into the data out latch by LAT OUTPUT BYT (3:0). It is possible to latch up only selected bytes into the data out latch. However, during a read array/ECC check operation, all four LAT OUTPUT BYT signals assert to latch up the full 32 bit longword. If the read array/ECC check operation is performed as part of a CPU or UNIBUS read of the memory, the data in the data out latch is placed onto the array bus by the assertion of OUTPUT BYT (0–3). The four OUTPUT BYT pins on the ECC chips are tied to C31 of the microword. Therefore the assertion of OUTPUT BYT (0–3) always places the full longword in the data out latch onto the array bus. If the read array/ECC check operation is performed as part of a write to memory, the data in the data out latch is not placed onto the array bus at this time.†

*Not necessarily true for an Octa-write.

†In a write to memory, the data in the data out latch will be modified or replaced by new data in an ECC check bit generation/write array operation (Paragraph 2.7.2 and Figure 2-43).

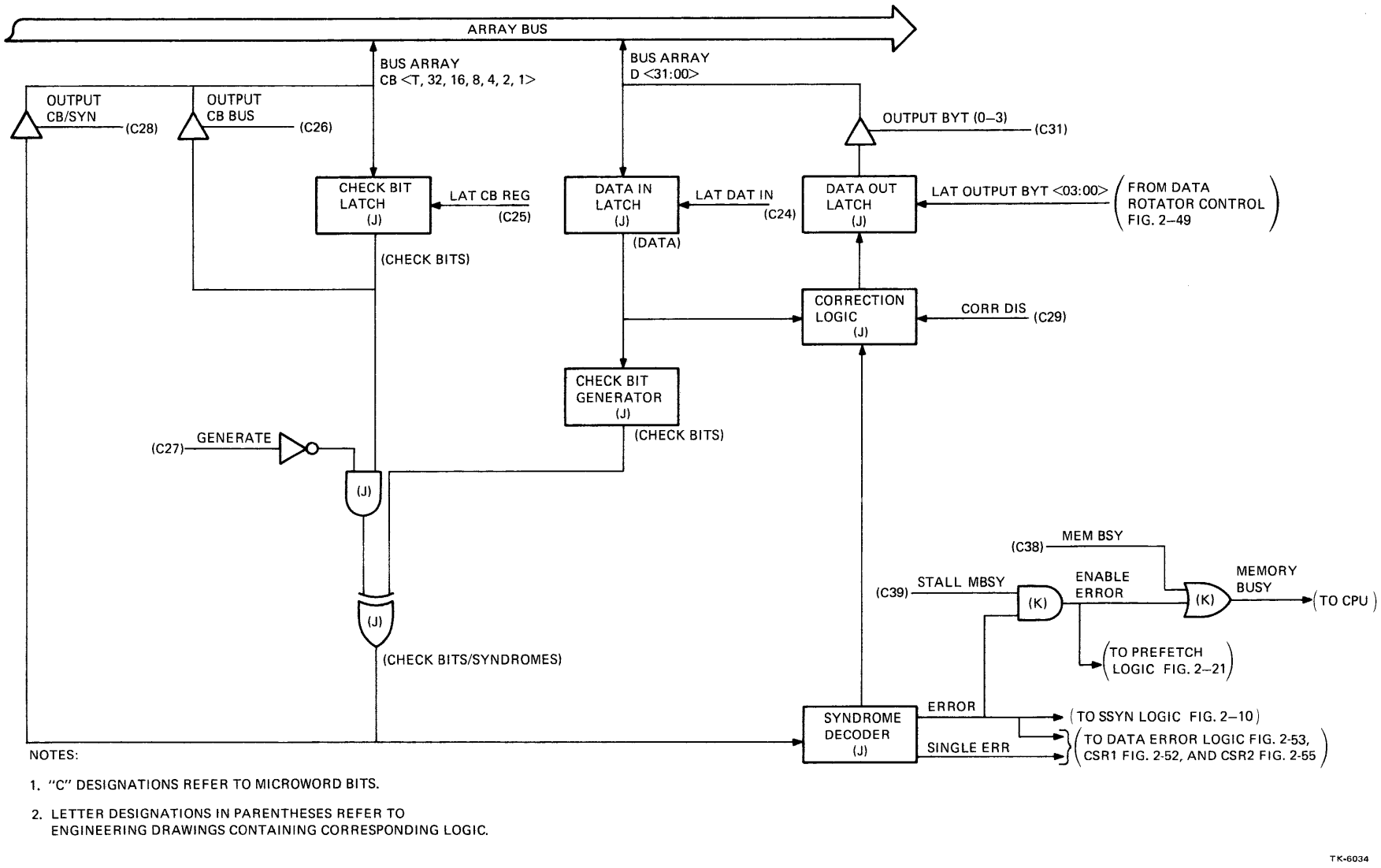


Figure 2-41 ECC Block Diagram

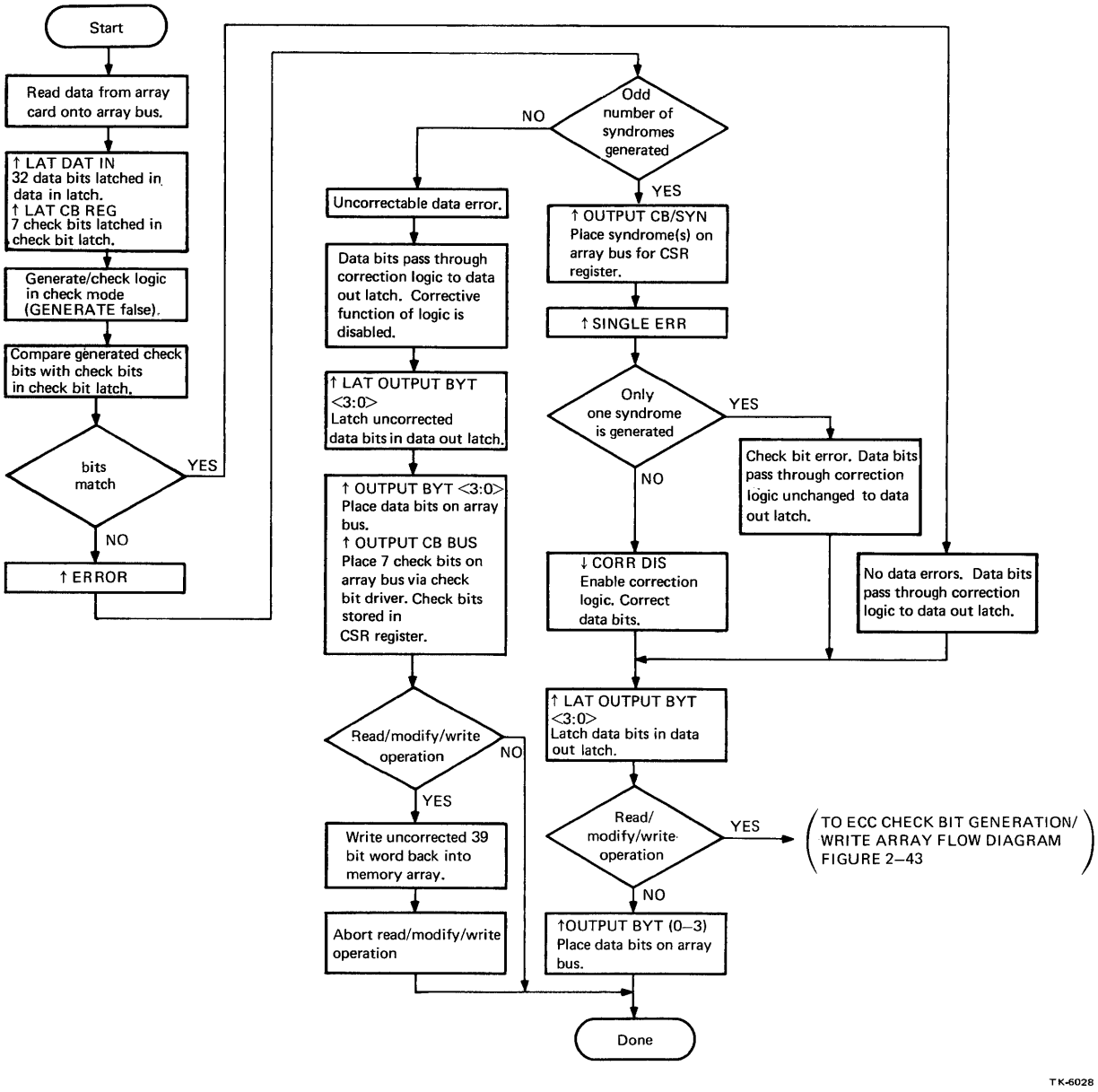


Figure 2-42 Read Array/ECC Check Flow Diagram

Table 2-10 Syndrome Codes

Syndromes Asserted	Meaning	Action
None	No data error	Read data normally.
One syndrome	Check bit error Data bits OK	Read data normally. Report error to CSR. Place syndrome in CSR.
Odd number of syndromes (more than one)	Single bit data error	Correct data bit. Read data normally. Report error to CSR. Place syndromes in CSR.
Even number of syndromes	Multi-bit data error (uncorrectable)	Read data normally. If doing a read/modify/write operation, write data and check bits back into memory array and abort the modify/write portion of the operation. Report error to CSR. Place check bits in CSR.

If an odd number of syndromes are generated, a single-bit data error exists. The syndrome(s) are placed on the array bus via the check bit/syndrome drivers, which are enabled by OUTPUT CB/SYN, and then applied to the CSR register for error analysis. The syndrome decoder asserts ERROR to indicate an error condition and SINGLE ERR to indicate it is a single-bit error and is correctable. The decoder determines which data bit is in error and provides a corrective input to the correction logic. CORR DIS negates to enable the correction logic. The data from the data in latch is coupled to the correction logic where the erroneous bit is corrected. The corrected data is then latched into the data out latch by the assertion of LAT OUTPUT BYT (3:0). The data remains in the data out latch when a write operation is being performed. If a read operation is being performed the data in the data out latch is placed onto the array bus by the assertion of OUTPUT BYT (0–3).

If an even number of syndromes are generated, more than one data bit is in error and the error is uncorrectable. The syndrome decoder asserts ERROR but does not assert SINGLE ERR thus indicating that a multibit uncorrectable error exists. The normally true state of CORR DIS keeps the correction logic disabled. The data from the data in latch passes through the data correction logic to the data out latch and then to the array bus in the normal manner. The check bits in the check bit latch are also returned to the array bus via the check bit drivers. The drivers are enabled by OUTPUT CB BUS from the microword. Now on the array bus are the data bits and check bits originally read out of the memory array. The CSR register is loaded with the check bits on the array bus (instead of the syndromes) for later analysis of the uncorrectable error by the CPU. In addition, if a read/modify/write operation is being performed, the modify/write portion of the operation is aborted and a write to the array is performed. Writing to the array stores the data that gave the uncorrectable error.*

*The uncorrectable error may have been caused by a soft read error, therefore the need to write back into the memory array.

2.7.2 ECC Check Bit Generation/Write Array (Figures 2-41 and 2-43)

An ECC check bit generation/write array operation is performed as part of a CPU or UNIBUS write to memory. As such, a read array/ECC check operation would have just been performed* and the data presently in the addressed memory array location is latched up in the ECC data out latch (Paragraph 2.7.1).

The write data on the array bus from the data rotator is latched into the data in latch by LAT DAT IN. The data then passes from the data in latch to the data out latch via the correction logic. CORR DIS is true disabling the corrective function of the correction logic. The write data is latched into the data out latch by LAT OUTPUT BYT (3:0). From one to four bytes of write data are latched up replacing the corresponding bytes presently in the latch. The new longword thus assembled is the data word for the memory array.

Check bits are now generated for the new longword before it is written into memory. OUTPUT BYT (0–3) asserts placing the longword on the array bus. LAT DATA IN asserts placing the longword into the data in latch. From the data in latch the data wraps around to the data out latch via the disabled correction logic. The four LAT OUTPUT BYT signals assert and latch up the data in the data out latch. The data in latch output is also applied to the check bit generator where seven check bits are generated. In the check bit generating mode (GENERATE true), the check bit generator outputs are coupled to the check bit/syndrome driver.

OUTPUT CB/SYN and OUTPUT BYT (0–3) assert and place the generated check bits and their associated data bits respectively onto the array bus. The 39 bits are then written into the memory array.

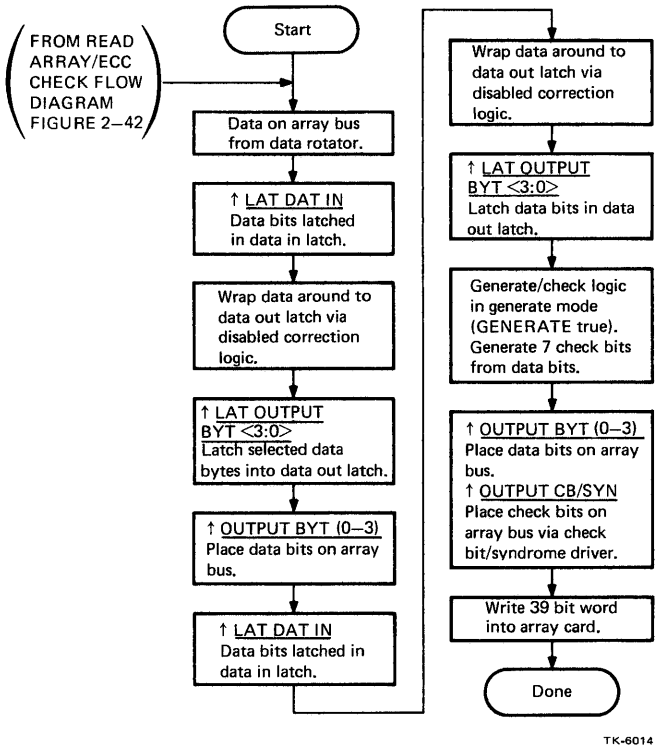


Figure 2-43 ECC Check Bit Generation/Write Array Flow Diagram

*Not necessarily true for an Octa-write.

2.7.3 ECC Chip Configuration (Figure 2-44)

Two DC631 ICs make up the ECC logic. One chip processes the low order word of the data longword and the other the high order word. The two chips are identical. A voltage potential of +3 V or 0 V on pin 23 (HI WORD) sets up the internal logic to configure the chip as either high word or low word. The two chips work in tandem to function as shown in Figure 2-41 and as described in Paragraphs 2.7.1 and 2.7.2. This paragraph does not repeat the descriptions given there, but rather discusses how the two chips function together.

2.7.3.1 ECC Check – The low word (bytes 0,1) is applied to the low word data in latch and then wrapped around through the correction logic to the data out latch. The low word is also applied to the low word check bit generator. The seven check bits from the array bus are applied to the check bit latch and then to XOR gates for comparison with the generated check bits. Partial syndromes are obtained from XORing the check bits with the check bits generated from the low word. The partial syndromes are coupled to the high word chip, via the seven PART SYND lines, where they are used against the upper word. The partial syndromes leave the low word chip via the check bit/syndrome driver, which is always enabled by +3 V, and are applied to the high word check bit latch. The latch is held open by +3 V and is transparent to the partial syndromes which flow through to the high word XOR gates. Here the partial syndromes are applied against the check bits generated by the high word check bit generator. The output of the high word XOR gates are the actual syndromes reflecting the error status of the longword. The syndromes are passed to the syndrome decoder via the high-word/low-word mux. The decoder outputs error status signals and supplies correction data to the correction logic if a single error is detected and the error is in the high word. The syndromes are also placed onto the array bus when OUTPUT CB/SYN asserts. From the array bus the syndromes are applied to the low word syndrome decoder via the high-word/low-word mux in the low word chip. If a single error is detected and the error is in the low word, the decoder supplies correction data to the low word correction logic. The high-word/low-word mux in the low word chip always selects the bus array CB signal input while the mux in the high word chip always selects the syndromes from the high word XOR gates.

2.7.3.2 ECC Check Bit Generation – The low word (bytes 0,1) is applied to the low word check bit generator via the low word data in latch. Check bits generated from the low word (partial check bits) are output to the seven PART SYND lines via the always enabled check bit/syndrome driver. The partial check bits pass through the check bit latch (held in the transparent state by +3 V) to the high word XOR gates. The high word (bytes 2,3) is applied to the high word check bit generator via the high word data in latch. Check bits generated from the high word are mixed with the low word partial check bits in the high word XOR gates. The XOR outputs are the seven check bits for the 32 bit longword. When OUTPUT CB/SYN asserts, the seven check bits are placed onto the array bus.

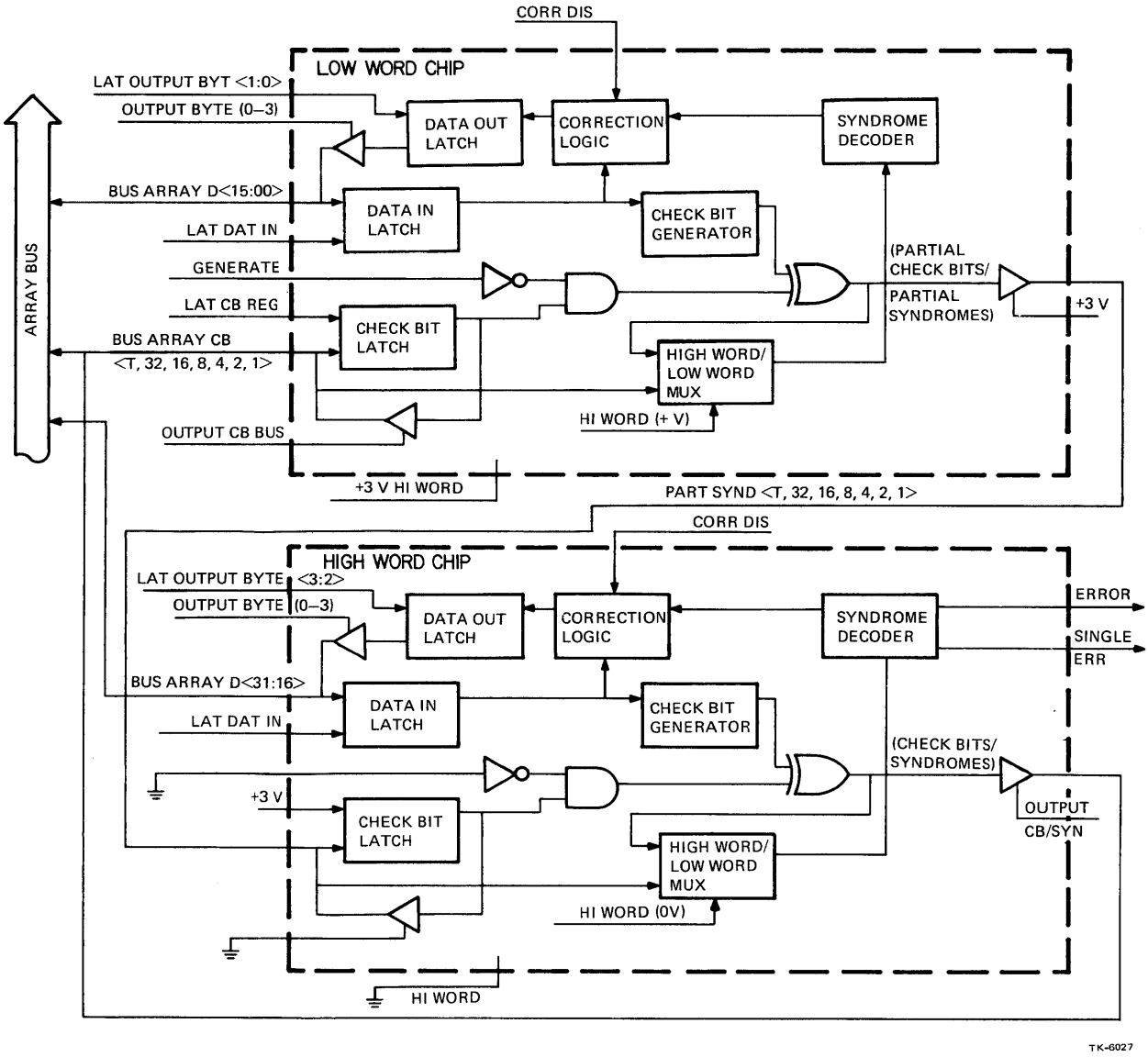


Figure 2-44 ECC Chip (DC631) Configuration

2.8 DATA ROTATOR

2.8.1 General

Data read from memory is sent to the CPU or the UNIBUS via the MC bus. The reading device expects the data to be oriented on the low order lines of the MC bus; however data read out of the memory array may be located in any of the four byte positions of the longword. If the data is not longword aligned (byte 0 in bit position 0), data rotation or shifting is required so the output is properly oriented on the MC bus. Furthermore, if the desired data crosses a longword boundary and is located in two memory locations, two read cycles are required to extract all the data. In this case, after the data is retrieved from memory, both byte selection and rotation is required to select the desired data and format it. The data rotator performs the function of byte selection and rotation.

Figure 2-45 illustrates several examples of data selection and rotation performed by the data rotator during read operations. Part A represents the byte orientation of data on the MC bus. The CPU and UNIBUS devices require data being presented to them to be in this format. Part B represents a longword location on the array card and how the data will appear on the array bus. In part B, two bytes (B0, B1) are read out and transferred to the MC bus. As they are already properly orientated, no rotation is required. The two bytes (X) in locations 1002 and 1003 are also transferred but are ignored by the reading device. Part C represents a longword location where the desired bytes (B0, B_1) are not lined up with the longword boundary. Bytes B0 and B_1 must be rotated (shifted) and placed on the MC bus in the byte 0 and byte 1 positions respectively, as shown in part A. This is done by rotating the data one byte right. In part D, four bytes are read which are not located in the same longword location in memory. Consequently, two read cycles are needed with byte rotation and selection required for both cycles. The first memory cycle reads location 1000, the data is rotated two bytes right (thereby placing bytes B0 and B1 in the byte 0 and byte 1 positions of the MC bus), and bytes B0 and B1 are selected (the bytes in locations 1000 and 1001 are discarded). A second memory cycle reads location 1004 and also rotates the data two bytes right placing bytes B2 and B3 into their proper positions, and selects bytes B2 and B3 while discarding the bytes in locations 1006 and 1007. The selected bytes from both cycles are placed on the MC bus supplying the reading device with the requested data in the proper format.

Figure 2-45 can be used in a reverse manner to illustrate the need for byte rotation during a CPU or UNIBUS write to memory. Part A represents how data to be written into memory is presented to the data rotator from the MC bus. In part B, byte 0 and byte 1 on the MC bus are to be written into byte positions B0 and B1 at locations 1000 and 1001. The data is directly passed through the data rotator as no rotation is required. In part C, byte 0 and byte 1 are to be written into B0 and B_1 at locations 1001 and 1002. The data rotator rotates (shifts) the data one byte left (opposite to the right rotation for a read)* thereby placing byte 0 and byte 1 into the proper position. In part D, four bytes of data are to be written into two memory locations. Two write cycles are performed. In the first cycle the MC bus data is rotated two bytes left and bytes 0 and 1 are selected and written into the last two byte positions of longword location 1000 (locations 1002 and 1003). Bytes 2 and 3 are inhibited (not discarded). (The byte selection for a write cycle is accomplished in the ECC logic, not in the data rotator. (Refer to Paragraph 2.7.2.) The two X bytes at locations 1000 and 1001 are retained and rewritten into their same locations. In the second cycle, bytes 2 and 3, now in the proper position from the cycle 1 rotation, are selected and written into the first two byte positions of longword location 1004 (locations 1004 and 1005). The two X bytes at locations 1006 and 1007 are retained and rewritten into their same locations.

*Note that the number of bytes rotated is the same for both a read and a write.

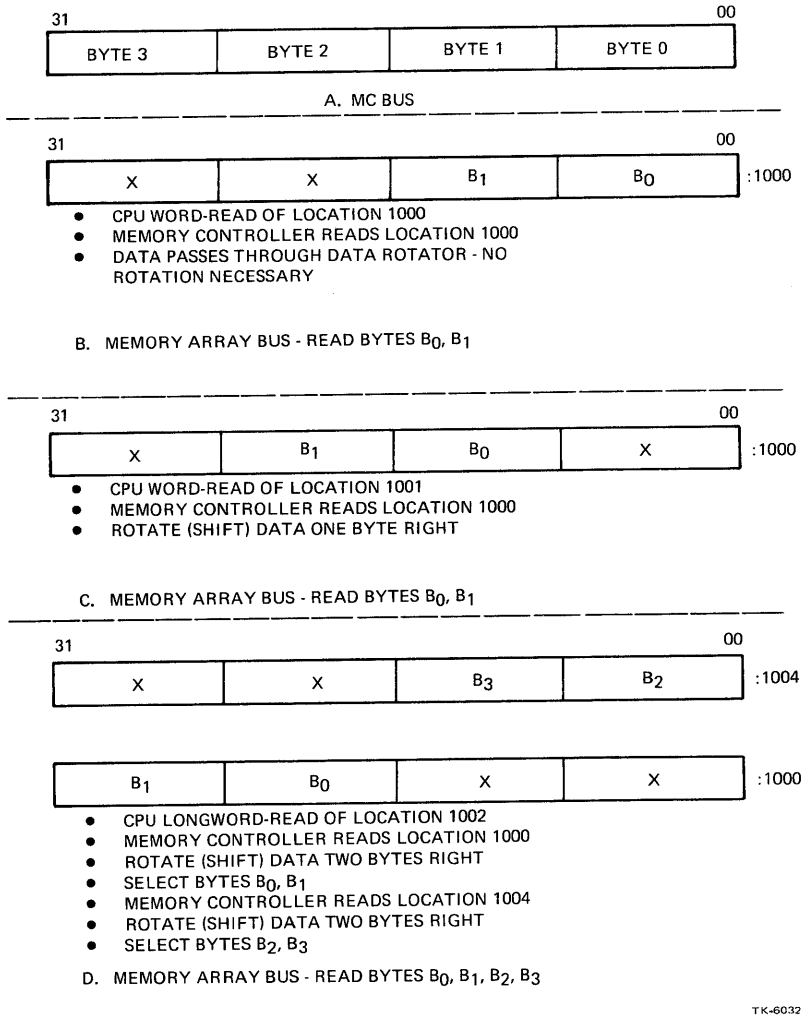


Figure 2-45 Data Rotation for Read Operations

2.8.2 Data Rotation – Read Operation (Figures 2-46 and 2-47)

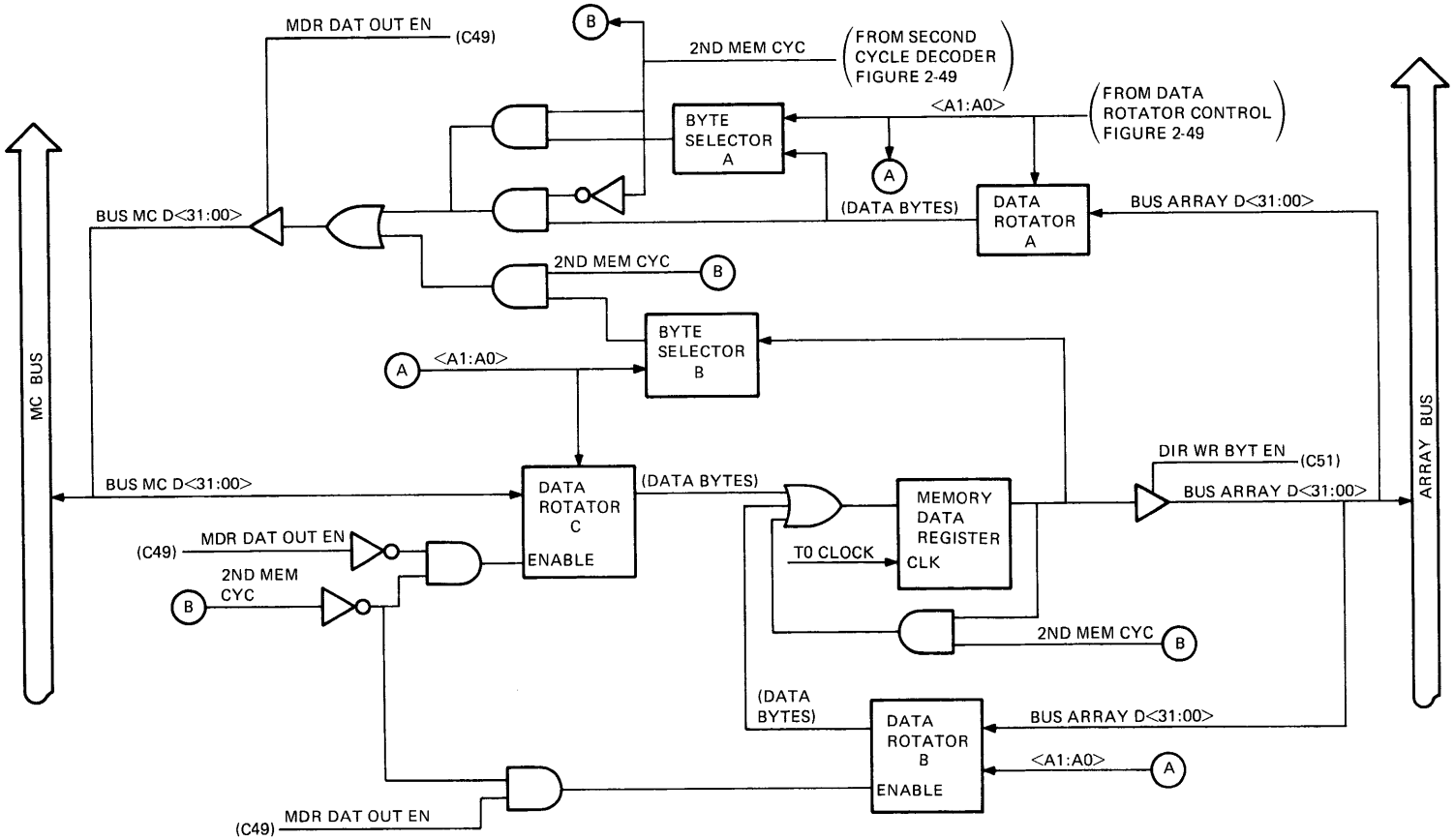
2.8.2.1 One-Cycle Read – The data rotator contains three rotator circuits designated as A, B, and C (Figure 2-46). Data rotators A and B are used for read operations. Data rotator C is used for write operations.

If the data to be read is entirely contained in one memory array location, a one-cycle read is performed using only data rotator A. MDR DAT OUT EN asserts and enables the data rotator A output to the MC bus. A "read array/ECC check" operation is performed to place the data on the array bus. The data on the array bus is input to data rotator A. A0 and A_1 from the data rotator control (Figure 2-49) specify if the data needs rotating and if so, by how much. Data rotator A is a mux with each input byte capable of being switched to any of four output byte positions with A0 and A_1 selecting which position. The mux allows any byte to be moved to the position of any other byte. The data is rotated, if necessary, and output to the MC bus via the MC bus driver.

2.8.2.2 Two-Cycle Read – If the data to be read is contained in two memory array locations, a two-cycle read is performed using data rotators A and B. In a two-cycle read, the first location is read out of the array, the data is rotated, and the desired byte(s) are selected. Then the second location is read, the data is rotated, and the bytes desired from the second read are selected. The desired bytes from the first and second read are placed onto the MC bus for the reading device.

MDR DAT OUT EN asserts to perform three functions: it enables the data rotator A output to the MC bus, it enables data rotator B, and it disables data rotator C. A "read array/ECC check" operation is performed to place the first cycle data on the array bus. The data on the array bus is input to data rotator B. A0 and A_1 from the addressing logic specify if the data needs rotating and if so, by how much. Data rotator B is similar to rotator A. Each input byte is muxed to one of four output byte positions as determined by the A0, A_1 code. The output of rotator B is applied to the memory data register (MDR) via an OR gate. Other OR gate inputs are from data rotator C and the MDR feedback AND gate. These two sources are disabled by MDR DAT OUT EN true and 2ND MEM CYC false respectively thereby leaving data rotator B as the only MDR input source. The first cycle data is latched into the MDR by T0 CLOCK. The data output from the MDR is applied to byte selector B which selects the desired bytes as determined by the A0, A_1 code.

The second cycle starts with the assertion of 2ND MEM CYC which disables data rotator B and enables the MDR feedback path so the MDR output wraps around and becomes the MDR input. T0 CLOCK latches up the same data thereby holding the first cycle data in the MDR for the second cycle. With 2ND MEM CYC true, the selected first cycle bytes are coupled from byte selector B to the MC bus via the MC bus driver. A second "read array/ECC check" operation is performed on the memory array placing the second longword on the array bus. The data longword is input to data rotator A where the required byte rotation is performed. The true state of 2ND MEM CYC routes the rotator output through byte selector A where the desired bytes are selected from the second cycle data and placed onto the MC bus via the MC bus driver.



NOTES:

1. "C" DESIGNATIONS REFER TO MICROWORD BITS.
2. THE LOGIC IN THIS FIGURE IS CONTAINED ON SHEET L OF THE ENGINEERING DRAWINGS.

TK-6568

Figure 2-46 Data Rotator Block Diagram

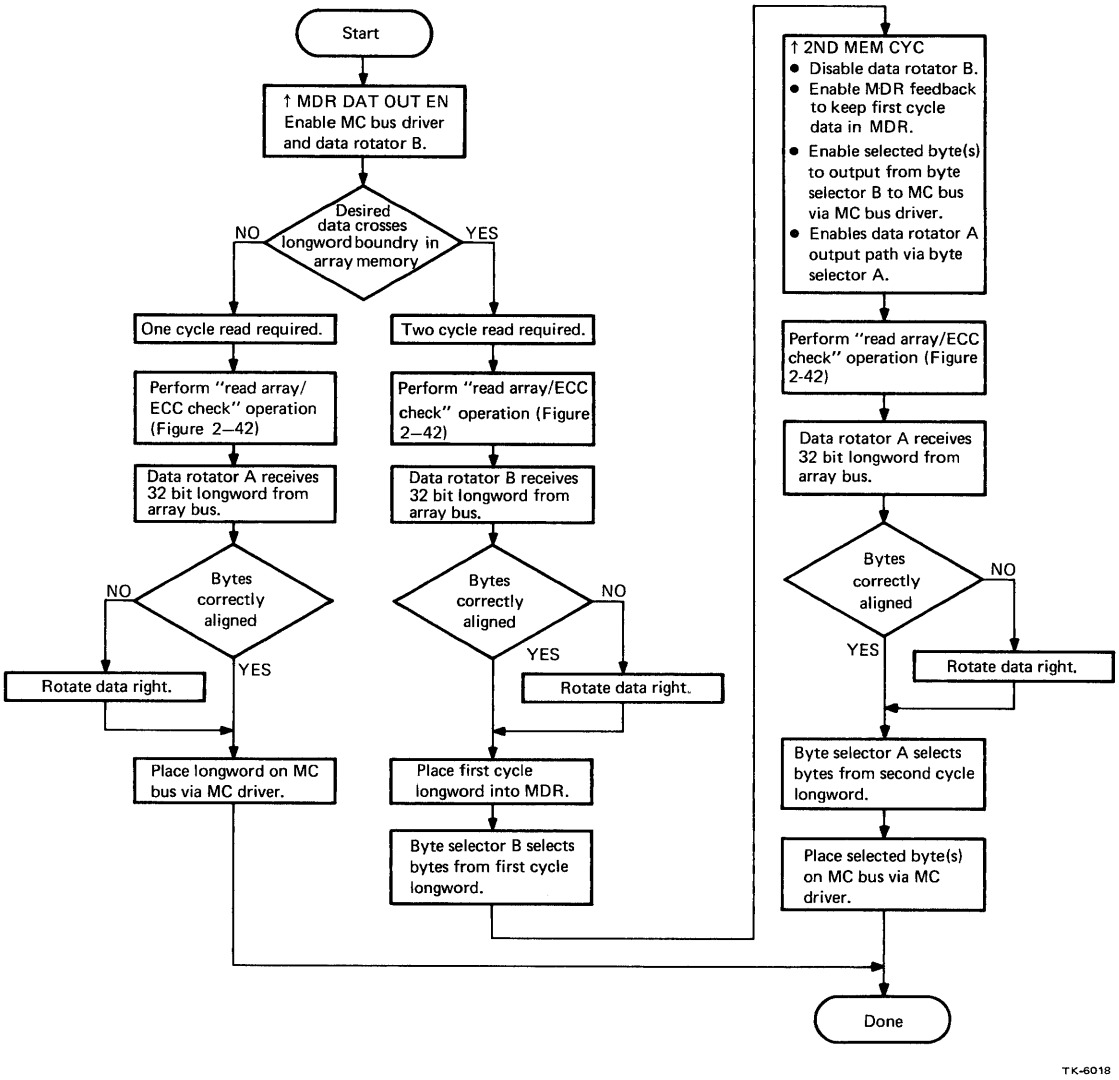


Figure 2-47 Data Rotator Read Flow Diagram

2.8.3 Data Rotation – Write Operation (Figures 2-46 and 2-48)

2.8.3.1 One-Cycle Write – To perform a write operation, MDR DAT OUT EN is negated enabling data rotator C and disabling data rotator B. Data from the CPU or the UNIBUS is placed on the MC bus and input to data rotator C. A0 and A_1 from the addressing logic specify if the data needs rotating and if so, by how much. Data rotator C is a mux and functions identically to data rotators A and B. The input data is rotated (shifted left), if necessary, and then output to the MDR via an OR gate. Data rotator B and the MDR feedback AND gate are disabled by MDR DAT EN false and 2ND MEM CYC false, respectively, thereby leaving data rotator C as the only MDR input source. The data is latched into the MDR by T0 CLOCK. A "read array/ECC check" operation is performed wherein the data in the addressed longword array location is read out, an ECC check performed, and the data latched up in the ECC data out latch (Figure 2-42). DIR WR BYT EN is then asserted to output the MDR data onto the array bus. An "ECC check bit generation/write array" operation is performed during which the new data bytes to be written into the longword array location are latched into the ECC data out latch. Check bits are generated for the new longword now in the data out latch and the longword and the generated check bits are placed onto the array bus and written into the memory array.

2.8.3.2 Two-Cycle Write – If data is to be written into two memory array locations, a two-cycle write is performed. The first cycle is identical to a one-cycle write, and the second cycle is almost identical to the first. During the second cycle, the write data is held in the MDR (no data is rotated) and a second read and write of the memory is performed.

The first cycle of a two-cycle write is as described in Paragraph 2.8.3.1 (One-Cycle Write). Upon completion of the first cycle, 2ND MEM CYC asserts disabling data rotator C and enabling the MDR feedback path so that the MDR output wraps around to the MDR input. T0 CLOCK latches up the input thereby holding the first cycle data in the MDR for the second cycle. A second "read array/ECC check" operation is performed wherein the data in the second longword array location is read out, checked by the ECC logic, and latched up in the ECC data out latch. DIR WR BYT EN asserts placing the MDR output on the array bus once again. Another "ECC check bit generation/write array" operation is performed where the data bytes to be written into the second longword location are latched into the ECC data out latch. New check bits are generated and the longword, together with its associated check bits, are placed onto the array bus and written into the second longword location in the memory array.

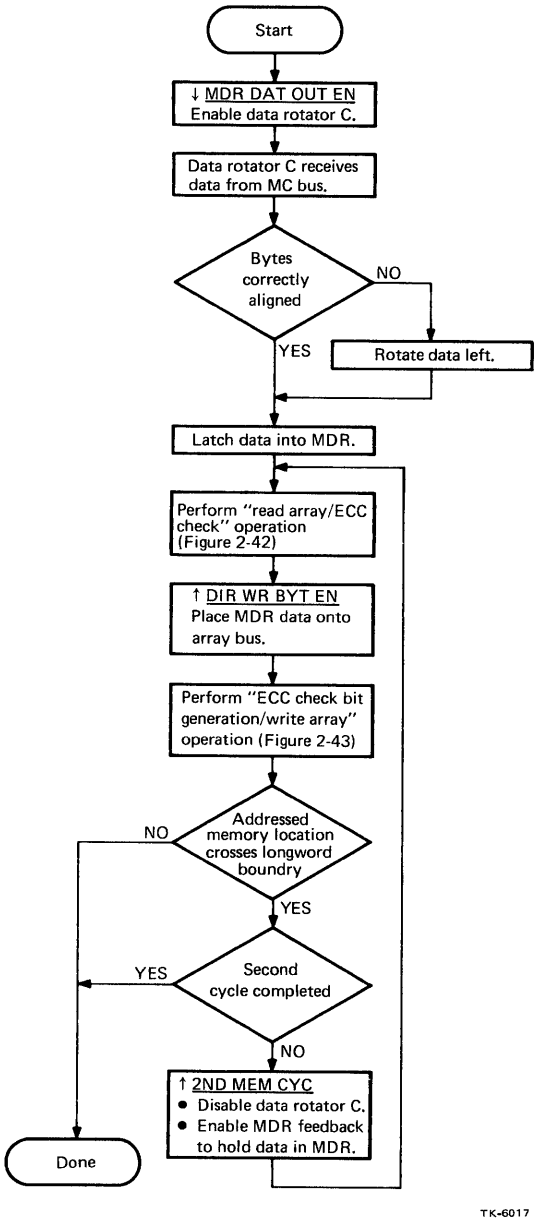


Figure 2-48 Data Rotator Write Flow Diagram

2.8.4 Data Rotation Control and Byte Selection Logic (Figure 2-49)

2.8.4.1 General – The data rotation control and byte selection logic determines how much data rotation is necessary (if any) and generates the A1, A0 control signals for the data rotator. The logic also determines if two memory cycles are required and which bytes are to be replaced in the ECC data out latch during a write operation. To accomplish this, the logic looks at the two least significant bits from the virtual address bus, the type of data transfer (byte, word, or longword), and the source of the reference request (CPU or UNIBUS device). The logic also responds to rotational commands (ROT C<1:0>) and special function commands (SPF<2:0> from the memory controller microword.

2.8.4.2 Data Rotator Control – The data rotator control generates the A1 and A0 control signals for the data rotator. To determine if data rotation is required, the rotator senses the byte location of the reference address within the array via bits LVA<01:00> from the virtual address bus. If a UNIBUS reference is being made (CPH/UBL false) the BYTE OFFSET bit from the translation buffer is also examined. The data rotator control code generated by the logic is shown in Table 2-11.

The direction of shift is right for a read operation and left for a write operation. The direction of shift is implemented within the data rotator logic where different rotators are used for read and write operations (Paragraph 2.8.2.1).

2.8.4.3 Data Type Logic – The data type logic generates an RS<1:0> code specifying the type of data being transferred (byte, word, or longword). The CPU indicates the type of data being transferred in a CPU operation by DATA TYPE <1:0> from the DAP module. A UNIBUS device indicates the type of data being transferred in a UNIBUS operation by the UNIBUS control bits UB C<1:0>. The signals are coupled through a flow-thru latch and are applied to the data type logic as L DT <1:0> and L UB C<1:0> respectively. CPH/UBL sets the logic for a CPU or a UNIBUS operation. The RS<1:0> codes generated for a CPU transfer and a UNIBUS transfer are shown in Tables 2-12 and 2-13 respectively.

2.8.4.4 Data Out Latch Byte Select – The data type RS code is applied to the data out latch byte select logic. The select logic generates four LAT OUTPUT BYT signals for the data out latch in the ECC logic. During a write operation (except for an aligned longword write) only selected bytes are latched into the ECC data out latch (Paragraph 2.7.2). The select logic determines which bytes are to be latched, and outputs the proper LAT OUTPUT BYT signal(s) accordingly. (For an aligned longword write operation or a read operation, all four LAT OUTPUT BYT signals are asserted.) To perform its function, the select logic looks at the amount of data rotation (A1, A0) and the data type (RS1, RS0), and determines if this is a CPU or a UNIBUS device transfer (CPH/UBL) and if this is the second cycle of a two-cycle access (2ND MEM CYC).

Table 2-11 Data Rotator Control Code

A1	A0	Byte Shift
0	0	No rotation
0	1	Shift 1 byte
1	0	Shift 2 bytes
1	1	Shift 3 bytes

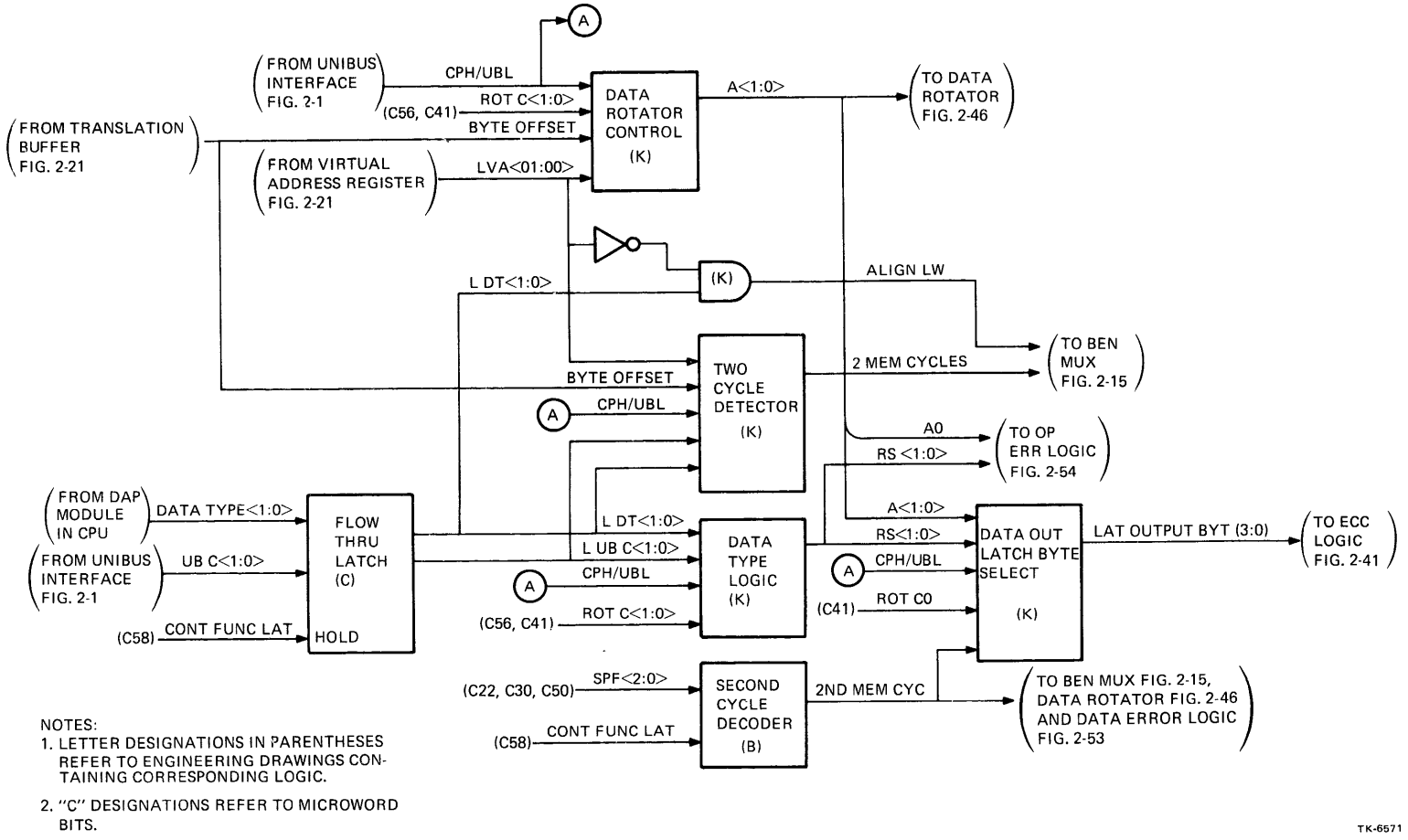


Figure 2-49 Data Rotation Control and Byte Selection Logic

Table 2-12 Data Type Bits vs RS Code for CPU Transfers

Data Type Bits		RS Bits		Data Type
L DT 1	L DT 0	RS 1	RS 0	
0	0	0	0	Byte
0	1	0	1	Word
1	0	1	0	
1	1	1	1	Longword

Table 2-13 UNIBUS Control Bits vs RS Code for UNIBUS Transfers

UNIBUS Control Bits		RS Bits		Data Type
L UB C1	L UB C0	RS1	RS0	
1	1	0	0	Byte
1	0	0	1	Word
0	1	0	1	Word
0	0	0	1	Word

2.8.4.5 Second Cycle Decoder – A second cycle decoder responds to the microword special function code (SPF<2:0> and asserts 2ND MEM CYC to the microsequencer and the data rotator during the second cycle of a two-cycle access.

2.8.4.6 Two-Cycle Detector – A two-cycle detector senses that an operation is going to require two memory array cycles. The detector looks at the byte position of the reference address (LVA<01:00>) and the type of data being referenced. The data type is indicated by L DT<1:0> for a CPU operation and L UB C<1:0> for a UNIBUS operation. CPH/UBL specifies which operation is being executed. Knowing the referenced byte position within the array and the type (length) of data to be transferred, the detector senses if two memory cycles are required and if so, asserts 2 MEM CYCLES to the microsequencer branch logic.

2.8.4.7 ALIGN LW – Another microsequencer branch signal generated in the data rotation control and byte selection logic is ALIGN LW. If the byte position referenced is byte 0 (LVA 01 and LVA 00 both false) and the data type is a longword (D DT1 and L DT0 both true), then the reference is to an aligned longword and the microsequencer branches accordingly.

2.9 ERROR LOGIC AND CSR REGISTERS

2.9.1 General

There are three CSR registers that receive control and test signals from, and report error status to, the CPU. These are CSR0, CSR1, and CSR2.* Figure 2-50 illustrates the three registers and their contents. CSR0 is a read only register that contains the seven syndrome bits, if a CPU to memory transaction encountered a correctable read data error, or the seven check bits if a CPU to memory transaction encountered an uncorrectable error. CSR1 contains control bits and error information on CPU to memory transactions. It also contains some diagnostic check bits. The error bits are read only while the control bits are read/write. CSR2 is a read only register containing error information on UNIBUS to memory transactions.

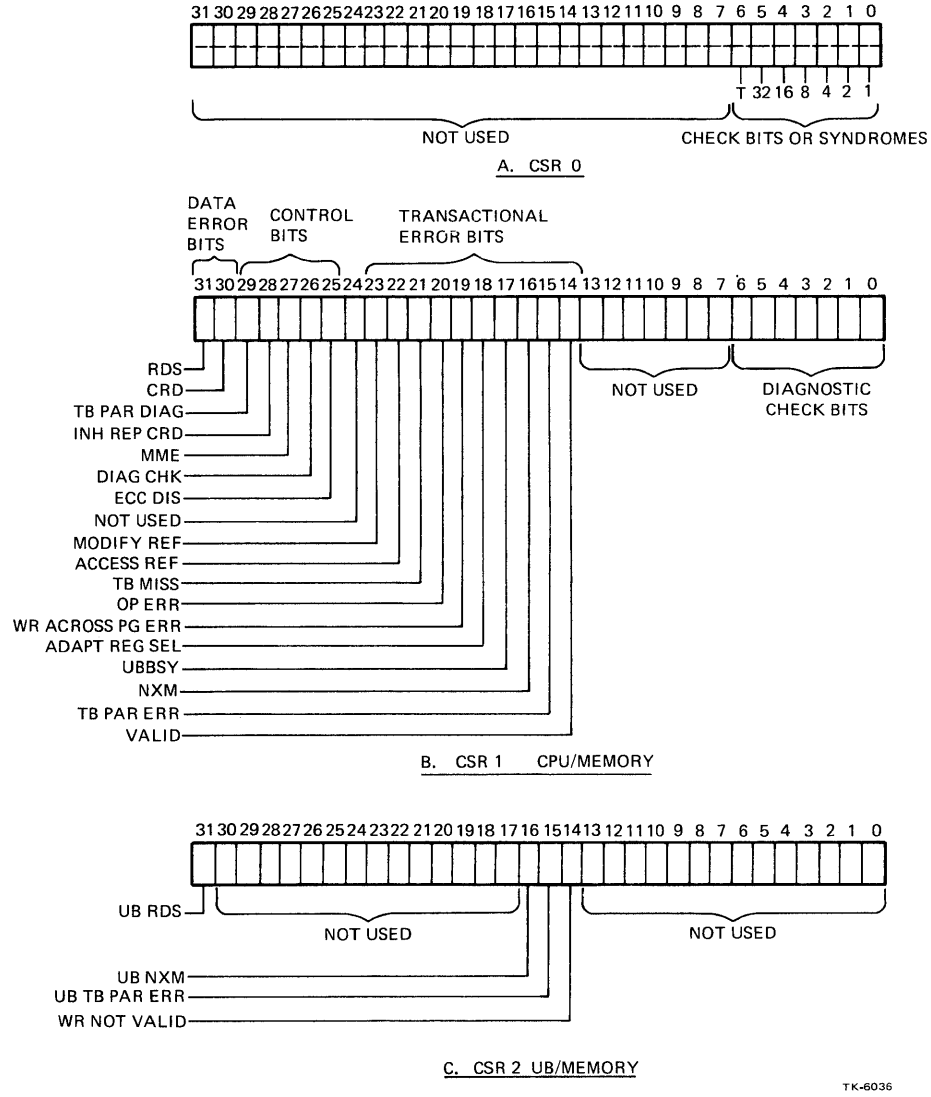


Figure 2-50 Summary of CSR Register Bits

*CSR0, CSR1, and CSR2 referred to here are real hardware registers as opposed to memory adapter registers CSR0, CSR1, and CSR2. The memory adapter registers are simulated by the CPU to the VAX operating system (Paragraph 2.5.3).

Figure 2-51 is a block diagram of the error logic and CSR registers. Refer to it throughout the rest of this section.

A RD CSR decoder and a CSR write decoder generate the read/write control signals for the registers. The decoders receive a two bit code (LVA <03:02>) from the virtual address bus that selects which register is to be addressed. The decoder outputs are enabled at the proper time by RD CSR and WR CSR from the microword.

2.9.2 CSR0 Check Bit/Syndrome Register

CSR0 is a read only register. BUS ARRAY CB<T,32,16,8,4,2,1> are clocked into CSR0 from the array bus by CSR CB/SYN CLK from the microword. BUS ARRAY CB <T,32,16,8,4,2,1> are syndromes if a correctable error occurred during a CPU to memory transaction, or check bits if a non-correctable error occurred. The assertion of RD CSR0 places the syndromes (or check bits) onto bits <06:00> of the MC bus for transfer to the CPU for analysis.

2.9.3 CSR1 ECC Diagnostic Check Bits <06:00>

Bits <06:00> of CSR1 are write only bits that are clocked in from the MC bus by WR CSR1 from the CSR write decoder. The seven bits are diagnostic check bits used to check the ECC logic. They are placed onto the check bit lines of the array bus by CSR CB EN from the microword.

2.9.4 CSR1 CPU/Memory Control Bits <29:25>

Bits <29:25> of CSR1 are read/write bits that are clocked in from the MC bus by WR CSR1 from the CSR write decoder. The five bits are control bits from the CPU used to control operations within the memory controller. The bits are shown in Figure 2-52 and described in Paragraphs 2.9.4.1 through 2.9.4.5.

The bits are read out to the MC bus by RD CSR1 from the RD CSR decoder, and cleared by a power up (UBS DCLO) or system initialization (CINIT).

2.9.4.1 ECC DIS <25> – Bit 25 is the error correction bit. When this bit is set it disables the ECC correction logic. When the correction logic is disabled, the following actions occur.

On a CPU to memory read, the check bits from the array are logged into bits <06:00> of CSR0. This gives a means of directly reading the check bits stored in the memory array.

On a CPU aligned longword write to memory, the check bits generated on the longword will be stored in bits <06:00> of CSR0 as well as in the memory array. This gives a means of diagnosing the ECC logic.

If on a CPU to memory read, a CRD occurs it will be logged in bit <30> of CSR1 and the check bits will be placed in CSR0.

RDS errors will be handled as in normal operating mode.

Unaligned CPU writes to memory and UNIBUS reads or writes to memory will not be allowed.

If both DIAG CHK bit and ECC DIS bit are set, the memory will operate as with just ECC DIS true, except that bits <06:00> of CSR1 will be clocked into CSR0 on every CPU to memory read cycle. This checks the signal path from the output of CSR1 to the input of CSR0 via the array bus.

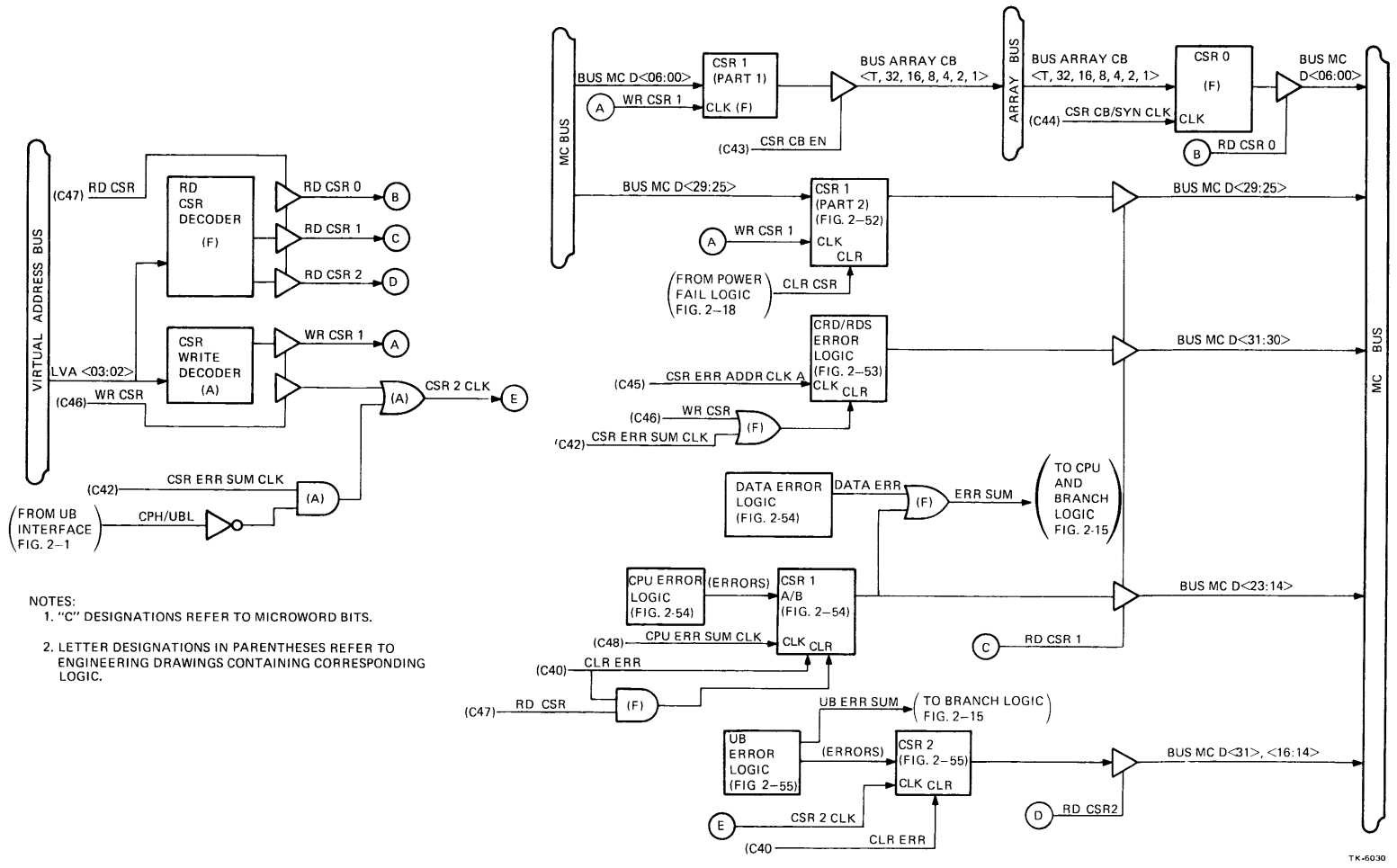


Figure 2-51 Error Logic and CSR Block Diagram

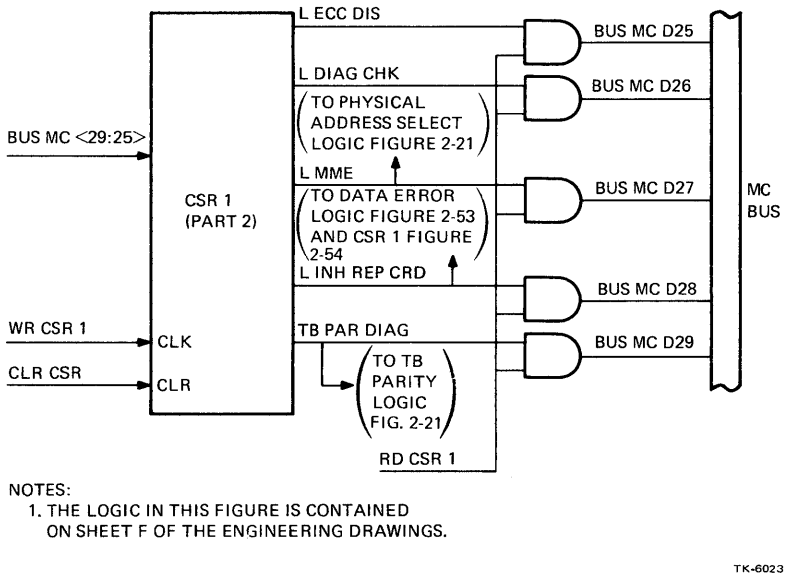


Figure 2-52 CSR1 CPU/Memory Control Bits

2.9.4.2 DIAG CHK <26> – Bit 26 is the diagnostic check bit. When this bit is set, the memory controller is in the diagnostic mode. This allows bits <06:00> of CSR1 to be substituted for the check bits read from the array in order to test the ECC logic. This mode is only recognized for CPU to memory one cycle reads (i.e., not recognized for any write cycles or UNIBUS initiated reads). CRD and RDS error information is logged per normal running mode. This bit together with the DIS ECC bit is also used to control microdiagnostic tests of the memory controller.

2.9.4.3 MME <27> – Bit 27 is the memory management enable bit. This bit enables translations via the translation buffer. If this bit is not set the CPU address will map directly into physical memory. Note that the UNIBUS map is always enabled.

2.9.4.4 INH REP CRD <28> – Bit 28 is the bit that inhibits reporting the occurrence of correctable read data errors (CRDs). When this bit is set, all CRDs will still be corrected by the ECC logic and syndromes will be logged in bits <06:00> of CSR0, however, the CRD bit (CSR1 <30>) will not be set. When the ECC DIS bit is set, INH REP CRD is overridden as all single errors are uncorrectable.

2.9.4.5 TB PAR DIAG <29> – Bit 29 is the translation buffer parity diagnostic bit. When this bit is set it will cause TB PAR ERR on any access to a TB entry in the translation buffer.

2.9.5 CSR1 CPU/Memory Data Error Bits <31:30>

Bits <31:30> of CSR1 are read only, data error bits indicating that a CRD (correctable) or an RDS (uncorrectable) error has been obtained on a CPU to memory transaction. The bits are clocked by CSR ERR ADDR CLK A from the controller microword. The logic asserting the bits is shown in Figure 2-53 and described in Paragraphs 2.9.5.1 and 2.9.5.2. The bits are read out to the MC bus by RD CSR1 from the RD CSR decoder and cleared by either WR CSR or CSR ERR SUM CLK. The assertion of either will clear the CRD/RDS bits except that CSR ERR SUM CLK will not clear the bits during the second cycle of a two-cycle memory access (2ND MEM CYC true). Thus a data error bit set during the first cycle is retained until the end of the second cycle when the CPU will read the CSRs. The CPU then checks to determine which memory cycle contained the error.

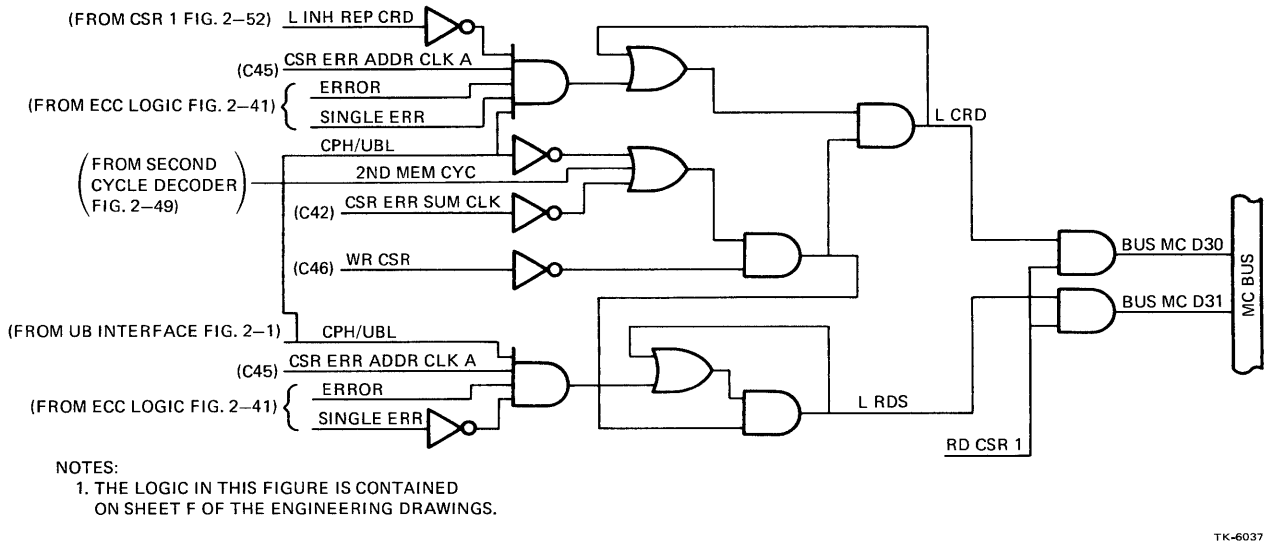


Figure 2-53 CSR1 CPU/Memory Data Error Bits

2.9.5.1 CRD <30> – This bit is set if on any CPU read to memory a single bit data error is detected and corrected. If the CPU has control of memory (CPH/UBL true), errors are being reported (L INH REP CRD false), and there is a correctable error (both ERROR and SINGLE ERR true), the next assertion of CSR ERR ADDR CLK A will clock L CRD set. Once set, L CRD latches itself up via an OR gate feedback path. The assertion of WR CSR releases the latch. The assertion of CSR ERR SUM CLK also releases the latch except during the second cycle of a two-cycle operation.

2.9.5.2 RDS <31> – This bit is set if on any CPU read to memory an uncorrectable data error is encountered. If the CPU has control of memory (CPH/UBL true), and ERROR is true but SINGLE ERR is false, the next assertion of CSR ERR ADDR CLK A will clock L RDS set. Once set, L RDS latches itself up via an OR gate feedback path. The assertion of WR CSR releases the latch. The assertion of CSR ERR SUM CLK also releases the latch except during the second cycle of a two-cycle operation.

2.9.6 CSR1 CPU/Memory Transaction Error Bits <23:14>

Bits <23:14> of CSR1 are read only bits indicating various errors associated with a CPU to memory transaction. The bits are clocked set by CPU ERR SUM CLK from the controller microword. The logic asserting the ten error bits is shown in Figure 2-54 and described in Paragraphs 2.9.6.1 through 2.9.6.10. The bits are read out to the MC bus by RD CSR1 from the RD CSR decoder. Bits <16:15>, <23:20> are cleared by CLR ERR from the microword. Bits <14>, <19:17> are cleared by CLR ERR except during a CPU WR CHK operation (RD CSR asserted).

2.9.6.1 VALID <14> – This bit is set if a CPU access to the translation buffer finds the TB entry VALID bit not set.* This bit (<14>) can be asserted only if memory management is enabled (ADDR PH false).

*The TB entry VALID bit is set when the page referenced by the TB entry is in the working set.

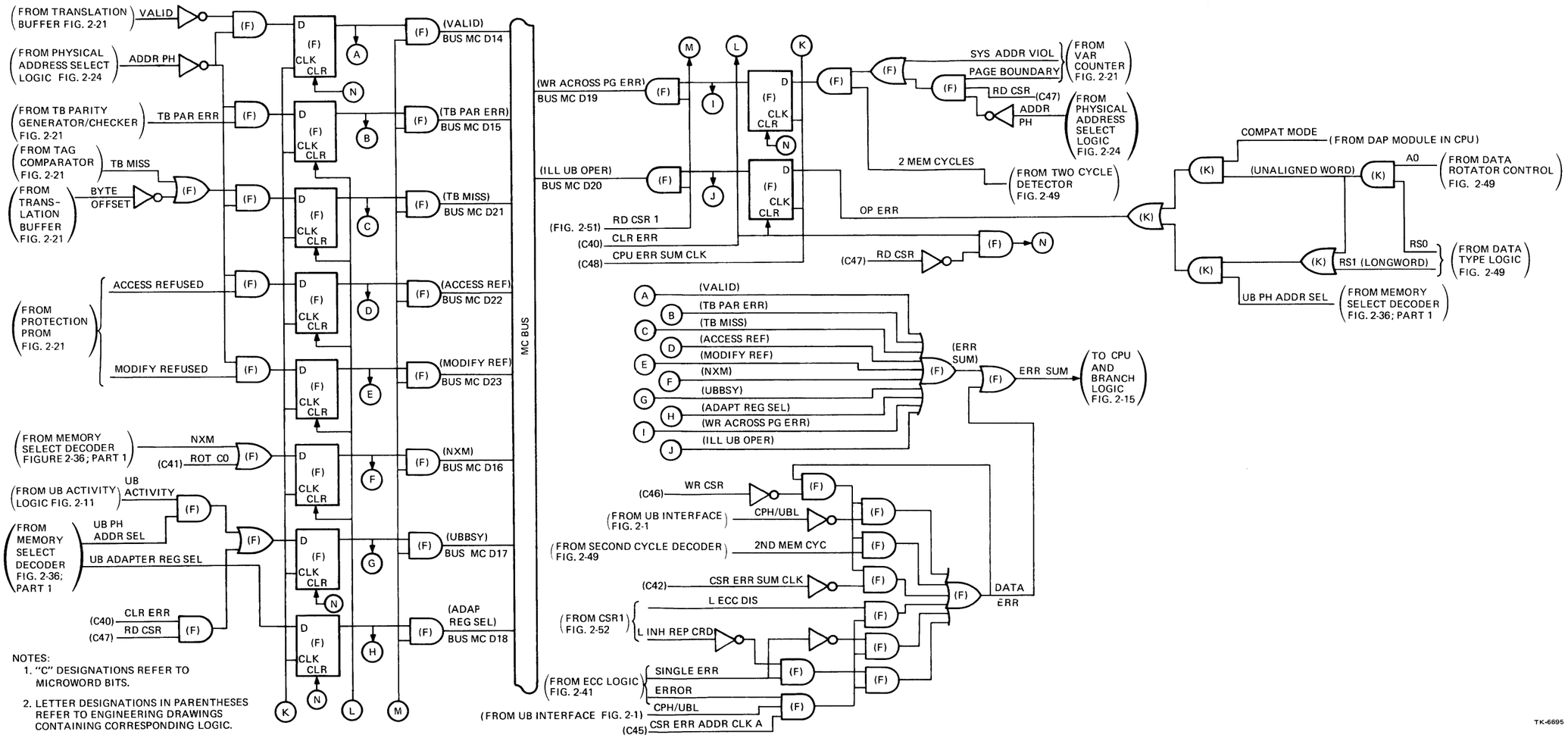


Figure 2-54 CSR1 CPU/Memory Transaction Error Bits and Error Summary Logic

2.9.6.2 TB PAR ERR <15> – This bit is set if a CPU access to the translation buffer results in a parity error. This bit can be asserted only if memory management is enabled (ADDR PH false).

2.9.6.3 NXM <16> – This bit indicates that the memory select decoder referenced a memory address where there is no memory array board (nonexistent memory). The bit is also asserted by ROT C0 from the microword when:

1. A CPU to UNIBUS request resulted in a SSYN timeout on the UNIBUS, or
2. A passive BR release results from the CPU issuing an ISSUE BG command.

2.9.6.4 UBBSY <17> – This bit indicates that a CPU transaction has the UNIBUS as a target but the UNIBUS is busy. The controller aborts the CPU transaction (Paragraph 2.2.2).

The UBBSY bit is also set during the interlocked READ.V.WCHK.LOCK operation (RD CSR true) even if the UNIBUS is not the CPU target. This keeps the UNIBUS locked out during the read/write cycles of the operation. CLR ERR from the microword is used for this purpose.

2.9.6.5 ADAPT REG SEL <18> – This bit indicates that the physical address (target) of a memory reference lies in the 3/4 megabyte UNIBUS adapter space between F0 0000 and FB FFFF (Figure 2-31). The CPU (not the memory controller) handles this reference to memory.

2.9.6.6 WR ACROSS PG ERR <19> – This bit is set if a CPU write (with a WCHK) attempts to write across a page boundary. The memory controller aborts the write cycle. The CPU then tests that the write access would be accepted in both pages, then repeats the write with a NOCHK.

Refer to Figure 2-54. When the data control logic senses that two memory cycles are needed for a transaction (2 MEM CYCLES true), the error logic checks for either of two error conditions. One of the error conditions is a system address violation (SYS ADDR VIOL) which asserts if the second memory reference will cross system boundary space. The other error condition is crossing a page boundary during a CPU WR CHK operation (indicated by RD CSR true) when memory management is enabled (ADDR PH false). Either condition sets the WR ACROSS PG ERR bit.

2.9.6.7 ILL UB OPER <20> – This bit is set by an operation error (OP ERR). OP ERR indicates that the CPU tried to perform a memory operation that is illegal in compatibility mode or tried to perform an illegal UNIBUS reference.

In compatibility mode only bytes and words are transferred (not longwords). However, word transfers must be word aligned in the memory arrays (i.e., reference must be made to byte 0 or byte 2 locations). If a word (RS0 true) is referenced to byte location 1 or 3 (A0 True), OP ERR asserts. An unaligned word reference to the UNIBUS or any longword (RS1 true) reference to the UNIBUS is always illegal and will assert OP ERR.

2.9.6.8 TB MISS <21> – This bit indicates that the translation buffer tag didn't match bits <30:15> of the virtual address in the VAR, or the BYTE OFFSET bit was not set. The BYTE OFFSET bit is used in CPU address translations to indicate that the entry in the TB is a legitimate entry written in by the CPU. In either case, the CPU translates the address, loads the TB, and tries again. If TB PAR ERR is set, this bit is meaningless. This bit will assert only if memory management is enabled (ADDR PH false).

2.9.6.9 ACCESS REF <22> – This bit signifies that the protection PROM has decided that the access requested by the CPU is not allowed. If the PAR ERR bit is set, this bit is meaningless. This bit will assert only if memory management is enabled (ADDR PH false).

2.9.6.10 MODIFY REF <23> – This bit means that the modify bit was not set in a TB entry referenced with a "WCHK" protection request. When an unmodified page is to be written into, it is required that the MODIFY bit be set before the write operation executes. This bit asserts ERR SUM to the CPU which sets the MODIFY bit and retries the "WCHK" operation. This bit will assert only if memory management is enabled (ADDR PH false).

2.9.7 CPU/Memory Error Summary (Figure 2-54)

The ten CPU/memory translation error bits are ORed to assert ERR SUM to the CPU. Upon receiving ERR SUM, the CPU reads CSR1 to determine the type of error.

ERR SUM is also asserted by DATA ERR obtained from data error logic that is similar to the CRD/RDS error logic of Paragraph 2.9.5. Referto Figure 2-54. If this is a CPU/memory transaction (CPH/UBL true), an error is detected in the ECC logic (ERROR true), and the data error logic is clocked by CSR ERR ADDR CLK A, then DATA ERR asserts if one of the following is true.

1. The ECC logic is disabled.
2. The error is not a single bit error (SINGLE ERR false).
3. The error is a single bit error and reporting of single errors is allowed (L INH REP CRD false).

When DATA ERR asserts, it is latched up by one of the following.

1. The memory controller is referencing the UNIBUS (CPH/UBL false).
2. A second memory cycle is in progress.
3. CSR ERR SUM CLK is false and has not cleared the CRD/RDS bits of CSR1.

The DATA ERR latch is released by a write to CSR1 (WR CSR true).

2.9.8 CSR2 UB/Memory Error Bits <31>, <16:14>

CSR2 is a read only register containing four error bits relating to UNIBUS to memory transactions. The remaining 28 bits of the register are not used. The error bits are clocked by CSR2 CLK from the CSR write decoder. The logic asserting the bits is shown in Figure 2-55 and described in Paragraphs 2.9.8.1 through 2.9.8.4. The bits are read out to the MC bus by RD CSR2 from the RD CSR decoder, and cleared by CLR ERR from the controller microword.

The UNIBUS error logic allows only one of the four error bits (the first to occur) to be set. The input logic for each bit contains feedback from the other bits such that, if any of the three bits is already set, the feedback will inhibit the setting of the fourth bit.

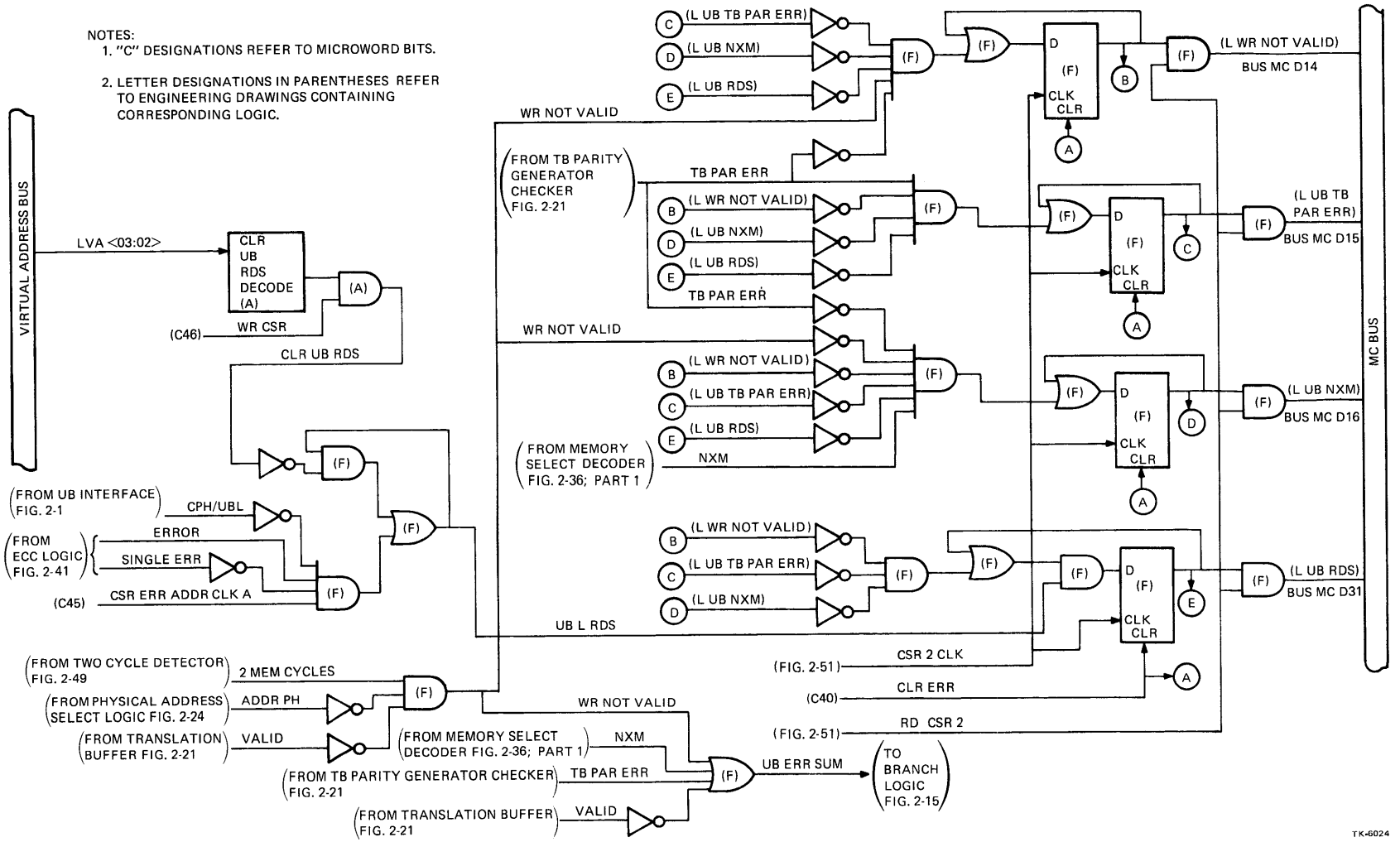


Figure 2-55 CSR2 UB/Memory Error Bits and Error Summary Logic

2.9.8.1 WR NOT VALID <14> – This bit is set if, during a two-cycle operation, an attempt is made to write into a page that does not have a valid entry in the translation buffer.*

Refer to Figure 2-55. If memory management is enabled (ADDR PH false) and two memory cycles are needed for the transaction (2 MEM CYCLES true), the UNIBUS error logic checks the VALID bit from the TB entry. If the bit is not set, there is no TB entry for the referenced page and the WR NOT VALID bit in CSR2 is set.

When WR NOT VALID sets, the write operation is aborted and SSYN is inhibited to the UNIBUS device.

2.9.8.2 UB TB PAR ERR <15> – This bit is set if a UNIBUS/memory transaction results in a TB parity error (TB PAR ERR). The transaction is aborted and SSYN inhibited resulting in a UNIBUS timeout.

2.9.8.3 UB NXM <16> – This bit is set if a UNIBUS request referenced a nonexistent array card.

2.9.8.4 UB RDS <31> – This bit is set if on any UNIBUS read to memory an uncorrectable data error is encountered.

The bit is asserted by UB L RDS obtained from RDS logic shown in Figure 2-55. If the UNIBUS has access to memory (CPH/UBL false) and an uncorrectable error exists (ERROR true and SINGLE ERR false), the next assertion of CSR ERR ADDR CLK A will clock UB L RDS into the UB RDS flip-flop of CSR2.

UB L RDS latches itself up via an OR gate. The latch is released by CLR UB RDS whenever CSR1 is written (WR CSR true) and the correct LVA <03:02> code is obtained from the virtual address bus.

When this bit is set, the memory controller inhibits a SSYN response to the UNIBUS device and the UNIBUS times out. The UNIBUS device sets its own NXM bit.

2.9.9 UNIBUS/Memory Error Summary (Figure 2-55)
A summation of UNIBUS errors (UB ERR SUM) is generated and sent to the microsequencer branch logic. The components making up this signal are:

1. NXM true,
2. TB PAR ERR true,
3. WR NOT VALID true, and
4. VALID false.

When UB ERR SUM asserts, the memory controller does not assert SSYN thereby causing the UNIBUS device to timeout. This in turn causes the device to interrupt the CPU which enters the ISR (interrupt service routine) to find out what the error was (by reading CSR2).

*In a one-cycle operation, if an attempt is made to write into a page that does not have a valid entry in the translation buffer (VALID bit not set), the operation is aborted before CSR2 is clocked and UB ERR SUM is checked (refer to Figure 2-27).

CHAPTER 3
USING THE PROM MICROCODE LISTING

3.1 GENERAL

A 512 × 72 bit PROM on the MCT module outputs a 72-bit microword on each transition of T0 CLOCK. The microword contains MCT control signals and thereby causes specific operations to be performed on the MCT module. The microword also contains next address bits and branching bits to formulate the PROM's next address and thus select the next microword output. Refer to Section 2.3 for a logic description of how the microsequencer functions to select the PROM's next address. This chapter illustrates how to use the PROM listing in the engineering documentation to determine the next address and thereby follow through the MCT routines.

3.2 MICROCODE LISTING

The microcode listing is divided into four parts.

1. **Definitions File**
   Starting with microword bit (field) 71, the name of each field is given, its state when asserted and negated (1 or 0), and its default state.

2. **MACRO Definition File**
   MACROs used within the body of the microcode are defined. The pertinent microword fields and their states are listed for each MACRO.

3. **Microcode Body**
   This section of the listing specifies the PROM address, the microword, the state of the pertinent fields, MACROs used (if any), branch conditions (if any), and notes.

4. **Cross Reference Listings**
   The following three cross reference lists are given.

   a. **Field Names vs Line Numbers**
      Field names are listed alphabetically and the line number given where that field is defined.

   b. **MACRO Names vs Line Numbers**
      MACRO names are listed alphabetically and the line number given where the MACRO is defined.

   c. **Memory Location vs Line Number**
      Memory locations are listed numerically and the line number where they can be found.

3.3 MICROWORD

The hexadecimal characters of the microwords in the listings are segmented into groups of four as shown in the following example.

C3E6, 8EE9, B041, BC01, 01

Figure 3-1 identifies each field in the microword and groups the bits in rows of 16 corresponding to the four hex bits of the microword notation. The nine NAD (next address) bits are the two least significant hex characters plus the least significant bit of the third hex character. The BEN (branch enable) bits for BEN 0, BEN 1, BEN 2, and BEN 3 are also identified.

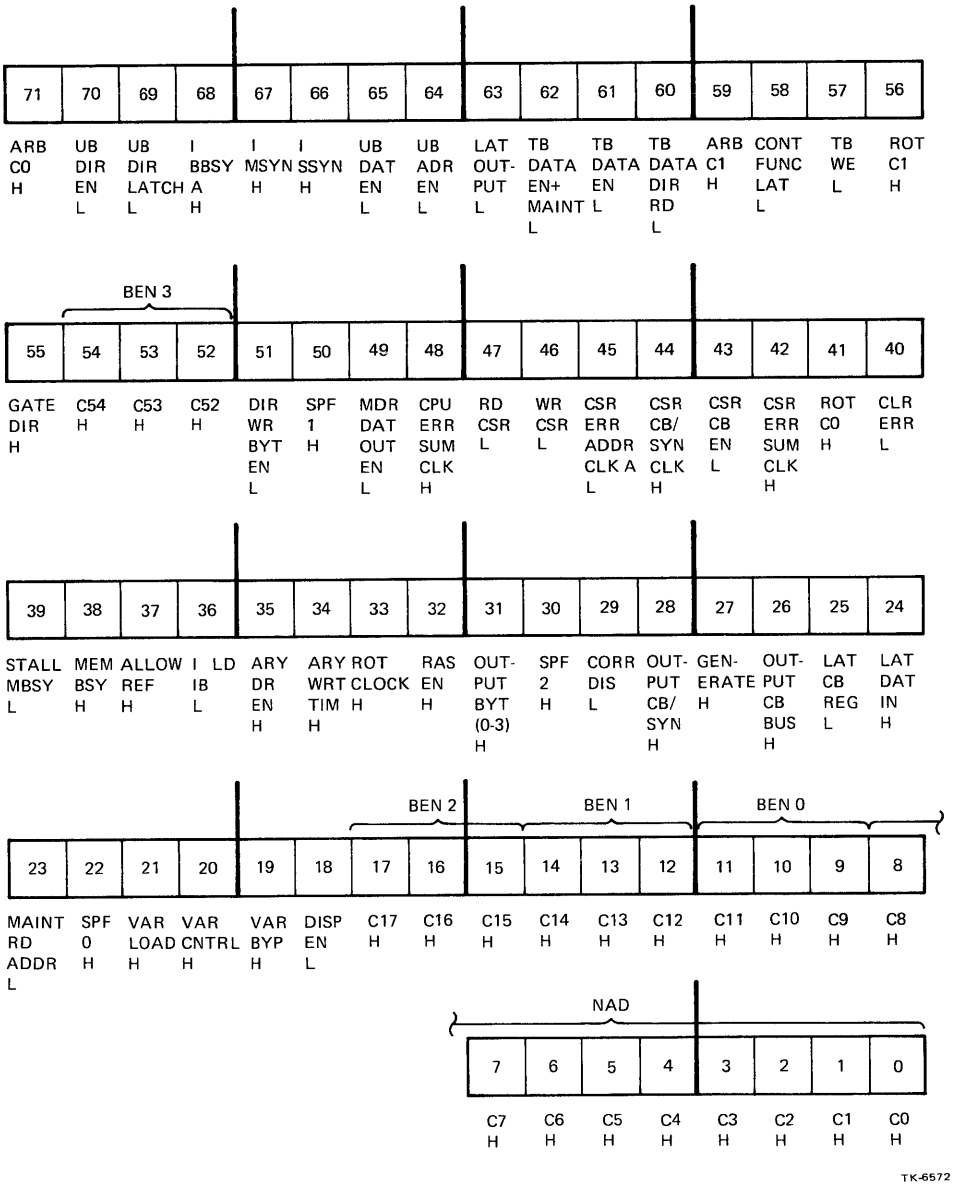


Figure 3-1 Microword Fields

3.4 READING THE MICROCODE LISTING

An exercise follows which involves reading the microcode listing. The exercise steps through the power-up sequence into the idle microstate, then into the dispatch microstate, and finally through a "write to the array" operation. The memory location of each step of the sequence is computed from the microword at the preceding location. Work forms (Figure 3-2) are used to present a suggested approach to reading the microword and interpreting the microword fields.

The MACROs used and the branch conditions (if any) for each microword is provided on the exercise forms. This data would be found in the body of the microcode listing. The listing itself is not used for the exercise due to changes that occur in the PROM routines and the resulting relocation of PROM microwords. The primary purpose of the exercise is not to show the power-up and WRITE.V.WCHK routines, but to illustrate reading of the listing and following an operation through to its conclusion. Using the method shown, the most recent revision of the listing can be read and current routines can be examined.

Figure 3-2 is the form used for the exercise. The nine PROM ADRS bits are given and the sources from which the ADRS bits are obtained. Microword bits C <8:4> are the source for address bits ADRS <8:4> respectively.* The source for address bits ADRS <3:0> is the branch condition selected by branch enable mux 3 through 0 respectively. When the three BEN select bits of any mux are all 0's, the respective "C" bit is selected. The actual NAD is formed from the microword "C" bits and the BEN branches (if any). The actual NAD in hex is then noted.

The procedure used in the exercise is as follows:

1. Record the memory location (hex) and the microword.
2. Record the NAD from the microword.
3. Record the three BEN bits for each of the four branch muxes.
4. Record any branch conditions selected by the BEN bits. (Branch conditions are found in the "definitions file" of the microcode listings).†
5. Record the actual NAD bits selected by branch conditions (if any).
6. Record the remaining NAD bits from the microword NAD.
7. Record the actual NAD in hex and find the next location. (When using the listing, refer to the "memory location vs line number" cross reference listing).

Figure 3-2a through 3-2o trace the PROM microwords through a power-up/write array routine. When power is applied to the VAX-11/730 the PROM is forced to location 100, the starting point of the power-up routine (refer to Paragraphs 2.3.5 and 2.3.6).

Figure 3-3 is a flow diagram of the exercise showing all the locations and branch conditions contained in the exercise sheets. The sheets specify the state of the branch conditions. The flow diagram also shows the location that would be reached if the branch condition were of the opposite state. A brief description is given of what would occur if the alternate branch were taken.

Flow diagrams can be made from the microcode listing for any of the routines contained in the PROM.

---

*Except during the dispatch microstate. Refer to Figure 3-2d.

†Note that branch conditions that are asserted low are followed by L. Otherwise the condition is asserted high.

ADDRESS; U *100*
(HEX)

MICROWORD; *C3E6, 8EE9, B041, BC01, 01*

MACROS; *PWR.FAIL:*
IDLE,
ARB.CLEAR.CPUG,
SET.DATO

NAD (FROM MICROWORD); *101*
(HEX)

BEN0; *000 → C0*

BEN1; *000 → C1*

BEN2; *000 → C2*

BEN3; *000 → C3*

ADRS;	8	7	6	5	4	3	2	1	0
SOURCE;	*C8*	*C7*	*C6*	*C5*	*C4*	*BEN3*	*BEN2*	*BEN1*	*BEN0*
ACTUAL NAD;	*1*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*1*

ACTUAL NAD (HEX); U *101*

TK-6608

Figure 3-2a Microcode Exercise Form

ADDRESS; U 101
(HEX)

MICROWORD; 43E6, 8AE9, B001, BC00, 06

MACROS; IDLE,
NAD/IDLE

NAD (FROM
MICROWORD); 006
(HEX)

BEN0; 000 → C0

BEN1; 000 → C1

BEN2; 000 → C2

BEN3; 000 → C3

ADRS;	8	7	6	5	4	3	2	1	0
SOURCE;	C8	C7	C6	C5	C4	BEN3	BEN2	BEN1	BEN0
ACTUAL NAD;	0	0	0	0	0	0	1	1	0

ACTUAL NAD (HEX); U 006

TK-6608A

Figure 3-2b Microcode Exercise Form

ADDRESS; U *006*
(HEX)

MICROWORD; *43E6, 8AE9, B001, BC7E, 04*

MACROS; *IDLE:*
IDLE,
BEN1/CPUG.L,
BEN0/LMSYN,
NAD/C1

NAD (FROM MICROWORD); *004*
(HEX)

BEN0; *111* → *LMSYN*

BEN1; *111* → *CPUG.L*

BEN2; *000* → *C2*

BEN3; *000* → *C3*

MSYN is not asserted.
CPUG.L has been asserted by the arbitrator.

ADRS;	8	7	6	5	4	3	2	1	0
SOURCE;	*C8*	*C7*	*C6*	*C5*	*C4*	*BEN3*	*BEN2*	*BEN1*	*BEN0*
ACTUAL NAD;	*0*	*0*	*0*	*0*	*0*	*0*	*1*	*0*	*0*

ACTUAL NAD (HEX); U *004*

TK-6608B

Figure 3-2c Microcode Exercise Form

ADDRESS; U 004
(HEX)

MICROWORD; 43E2, 9AE9, 9001, B800, 02

MACROS; C1: START.ADDR.PROM,
NAD/CPU.ISTREAM.REQ

NAD (FROM
MICROWORD); 002
(HEX)

BEN0; 000 → C0

BEN1; 000 → C1

BEN2; 000 → C2

BEN3; 001 → CSR07

C18 (DISP EN L) is asserted indicating that this is the CPU dispatch microstate. The CSR bits from the CPU are used to formulate the NAD.

ADRS;	8	7	6	5	4	3	2	1	0
SOURCE;	C8	CSR18	CSR17	CSR16	CSR08	BEN3	BEN2	BEN1	BEN0
ACTUAL NAD;	0	0	1	1	0	1	0	1	0

ACTUAL NAD (HEX); U 06A

TK-6608C

Figure 3-2d Microcode Exercise Form

ADDRESS; U 06A
(HEX)

MICROWORD; 43E2, 0A69, D301, 9C14, 84

MACROS; WRITE.V.WCHK:
RAS.EN/ON,
BYPBSY,
ROT.CLOCK/CLOCK
RD.CSR/READ,
BEN1/REF.IN.PROG,
BEN0/UB.PH.ADDR.SEL.L,
NAD/WR.UBS.CYC.WCHK

NAD (FROM MICROWORD); 084
(HEX)

BEN0; 010 → UB.PH.ADDR.SEL.L

BEN1; 001 → REF.IN.PROG

BEN2; 000 → C2

BEN3; 000 → C3

A refresh cycle is not in progress.
The translated physical address is not a UNIBUS address.

ADRS;	8	7	6	5	4	3	2	1	0
SOURCE;	C8	C7	C6	C5	C4	BEN3	BEN2	BEN1	BEN0
ACTUAL NAD;	0	1	0	0	0	0	1	0	1

ACTUAL NAD (HEX); U 085

TK-6608D

Figure 3-2e Microcode Exercise Form

ADDRESS; U 085
(HEX)

MICROWORD; 43E2, 0A69, D901, 9C01, C1

MACROS; ;WRITE TO ARRAYS
;=01
WR.ARRY.C4.WCHK:
BYPBSY,
READ.ARRAY,
RD.CSR/READ,
NAD/WR.ARRY.C5.WCHK

NAD (FROM MICROWORD); 1C1
(HEX)

BEN0; 000 → C0

BEN1; 000 → C1

BEN2; 000 → C2

BEN3; 000 → C3

ADRS;	8	7	6	5	4	3	2	1	0
SOURCE;	C8	C7	C6	C5	C4	BEN3	BEN2	BEN1	BEN0
ACTUAL NAD;	1	1	1	0	0	0	0	0	1

ACTUAL NAD (HEX); U 1C1

TK-6608E

Figure 3-2f Microcode Exercise Form

ADDRESS; U *1C1*
(HEX)

MICROWORD; *4362, 0B6D, D902, 9E81, 02*

MACROS; *WR.ARRY.C5.WCHK: BYPBSY, READ.ARRAY, OPEN.ECC, RD.CSR/READ, CSR.ERR.SUM.CLK/CLOCK, CPU.ERR.SUM.CLK/CLOCK, BEN2/ALIGN.LW.L, NAD/WR.ARRY.ALIGN.LW.C6*

NAD (FROM MICROWORD); *102*
(HEX)

BEN0; *000* → *C0*

BEN1; *000* → *C1*

BEN2; *101* → *ALIGN.LW.L*

BEN3; *000* → *C3*

The data to be written is not an aligned longword.

ADRS;	8	7	6	5	4	3	2	1	0
SOURCE;	*C8*	*C7*	*C6*	*C5*	*C4*	*BEN3*	*BEN2*	*BEN1*	*BEN0*
ACTUAL NAD;	*1*	*0*	*0*	*0*	*0*	*0*	*1*	*1*	*0*

ACTUAL NAD (HEX); U *106*

TK-6608F

Figure 3-2g Microcode Exercise Form

ADDRESS; U *106*
(HEX)

MICROWORD; *4362, 8AE9, 5902, 9D01, 15*

MACROS; *WR.ARRY.C6:*
BYPBSY,
GATE.DIR/READ.CPU,
READ.ARRAY,
OPEN.ECC,
STALL.MEM.BUSY/STALL,
BEN2/ERR.SUM.L,
NAD/WR.ARRY.C7

NAD (FROM MICROWORD); *115*
(HEX)

BEN0; *000* → *C0*

BEN1; *000* → *C1*

BEN2; *010* → *ERR.SUM.L*

BEN3; *000* → *C3*

There are no translation errors.

ADRS;	8	7	6	5	4	3	2	1	0
SOURCE;	*C8*	*C7*	*C6*	*C5*	*C4*	*BEN3*	*BEN2*	*BEN1*	*BEN0*
ACTUAL NAD;	*1*	*0*	*0*	*0*	*1*	*0*	*1*	*0*	*1*

ACTUAL NAD (HEX); U *115*

TK-6608G

Figure 3-2h Microcode Exercise Form

ADDRESS; U 115
(HEX)

MICROWORD; 43E2, 8AEB, 1001, 9C66, 7C

MACROS; WR.ARRY.C7:
WRITE.SUBSTITUTE,
STALL.MEM.BUSY/STALL,
VAR.BYPASS.EN/ON,
GATE.DIR/READ.CPU,
BEN1/LCPU.DR,
BEN0/ERROR.L,
NAD/WR.ARRY.WDE.C8

NAD (FROM MICROWORD); 07C
(HEX)

BEN0; 011 → ERROR.L

BEN1; 110 → LCPU.DR

BEN2; 000 → C2

BEN3; 000 → C3

There is no data error
CPU has placed write data on MC bus (L CPU.DR asserted).

ADRS;	8	7	6	5	4	3	2	1	0
SOURCE;	C8	C7	C6	C5	C4	BEN3	BEN2	BEN1	BEN0
ACTUAL NAD;	0	0	1	1	1	1	1	1	1

ACTUAL NAD (HEX); U 07F

TK-6608H

Figure 3-2i Microcode Exercise Form

ADDRESS; U 07F
(HEX)

MICROWORD; 4362, 86EB, 9000, 9401, 1E

MACROS; WR.ARRY.C8: GATE.DIR/READ.CPU,
WRITE.SUBSTITUTE,
SECOND.MEMORY.CYCLE,
ECC.DATA.IN.LATCH/OPEN,
ECC.OUTPUT.LATCH/OPEN,
MDR.TO.ARRAY.EN/ON,
NAD/WR.ARRY.C9

NAD (FROM MICROWORD); 11E
(HEX)

BEN0; 000 → C0

BEN1; 000 → C1

BEN2; 000 → C2

BEN3; 000 → C3

ADRS;	8	7	6	5	4	3	2	1	0
SOURCE;	C8	C7	C6	C5	C4	BEN3	BEN2	BEN1	BEN0
ACTUAL NAD;	1	0	0	0	1	1	1	1	0

ACTUAL NAD (HEX); U 11E

TK-66081

Figure 3-2j Microcode Exercise Form

ADDRESS; U *11E*
(HEX)

MICROWORD; *43E2, 0AEB, 9001, 9411, 1C*

MACROS; *WR.ARRY.C9:*

WRITE.SUBSTITUTE,
BEN1/REF.IN.PROG,
NAD/WR.ARRY.C10

NAD (FROM MICROWORD); *11C*
(HEX)

BEN0; *000* → *C0*

BEN1; *001* → *REF.IN.PROG*

BEN2; *000* → *C2*

BEN3; *000* → *C3*

A refresh cycle is not in progress.

ADRS;	8	7	6	5	4	3	2	1	0
SOURCE;	*C8*	*C7*	*C6*	*C5*	*C4*	*BEN3*	*BEN2*	*BEN1*	*BEN0*
ACTUAL NAD;	*1*	*0*	*0*	*0*	*1*	*1*	*1*	*0*	*0*

ACTUAL NAD (HEX); U *11C*

TK-6608J

Figure 3-2k Microcode Exercise Form

ADDRESS; U 11C
(HEX)

MICROWORD; 43E2, 0AEB, 9498, 9431, ID

MACROS; WR.ARRY.C10:

WRITE.SUBSTITUTE,
ECC.DATA.IN.LATCH/OPEN,
ECC.WRITE,
WR.TIM/ON,
BEN1/LECC.DIS,
NAD/WR.ARRY.C11

NAD (FROM MICROWORD); 11D
(HEX)

BEN0; 000 → C0

BEN1; 011 → LECC.DIS

BEN2; 000 → C2

BEN3; 000 → C3

ECC is not disabled.

ADRS;	8	7	6	5	4	3	2	1	0
SOURCE;	C8	C7	C6	C5	C4	BEN3	BEN2	BEN1	BEN0
ACTUAL NAD;	1	0	0	0	1	1	1	0	1

ACTUAL NAD (HEX); U 11D

TK-6608K

Figure 3-21 Microcode Exercise Form

ADDRESS; U 11D
(HEX)

MICROWORD; 43E2, 0AE9, 9598, 9401, B5

MACROS; WR.ARRY.C11:
WRITE.ARRAY,
ECC.DATA.IN.LATCH/OPEN,
ECC.WRITE,
NAD/WR.ARRY.C12

NAD (FROM MICROWORD); 1B5
(HEX)

BEN0; 000 → C0

BEN1; 000 → C1

BEN2; 000 → C2

BEN3; 000 → C3

ADRS;	8	7	6	5	4	3	2	1	0
SOURCE;	C8	C7	C6	C5	C4	BEN3	BEN2	BEN1	BEN0
ACTUAL NAD;	1	1	0	1	1	0	1	0	1

ACTUAL NAD (HEX); U 1B5

TK-6608L

Figure 3-2m Microcode Exercise Form

ADDRESS; U 1B5
(HEX)

MICROWORD; C3E2, 0AE9, 9599, 9421, 20

MACROS; WR.ARRY.C12:
WRITE.ARRAY,
ECC.WRITE,
ARB.CLEAR.CPUG,
BEN1/TWO.MEM.CYCLES.L,
NAD/WR.ARRY.C13

NAD (FROM MICROWORD); 120
(HEX)

BEN0; 000 → C0

BEN1; 010 → TWO.MEM.CYCLES.L

BEN2; 000 → C2

BEN3; 000 → C3

One memory cycle.

ADRS;	8	7	6	5	4	3	2	1	0
SOURCE;	C8	C7	C6	C5	C4	BEN3	BEN2	BEN1	BEN0
ACTUAL NAD;	1	0	0	1	0	0	0	1	0

ACTUAL NAD (HEX); U 122

TK-6608M

Figure 3-2n Microcode Exercise Form

ADDRESS; U 122
(HEX)

MICROWORD; 43E2, 0AE9, 9501, A400, 06

MACROS; WRITE.ARRAY,
VAR.CONTROL/LOAD.UB,
NAD/IDLE

NAD (FROM
MICROWORD); 006
(HEX)

BEN0; 000 → C0

BEN1; 000 → C1

BEN2; 000 → C2

BEN3; 000 → C3

ADRS;	8	7	6	5	4	3	2	1	0
SOURCE;	C8	C7	C6	C5	C4	BEN3	BEN2	BEN1	BEN0
ACTUAL NAD;	0	0	0	0	0	0	1	1	0

ACTUAL NAD (HEX); U 006

TK-6608N

Figure 3-2o Microcode Exercise Form

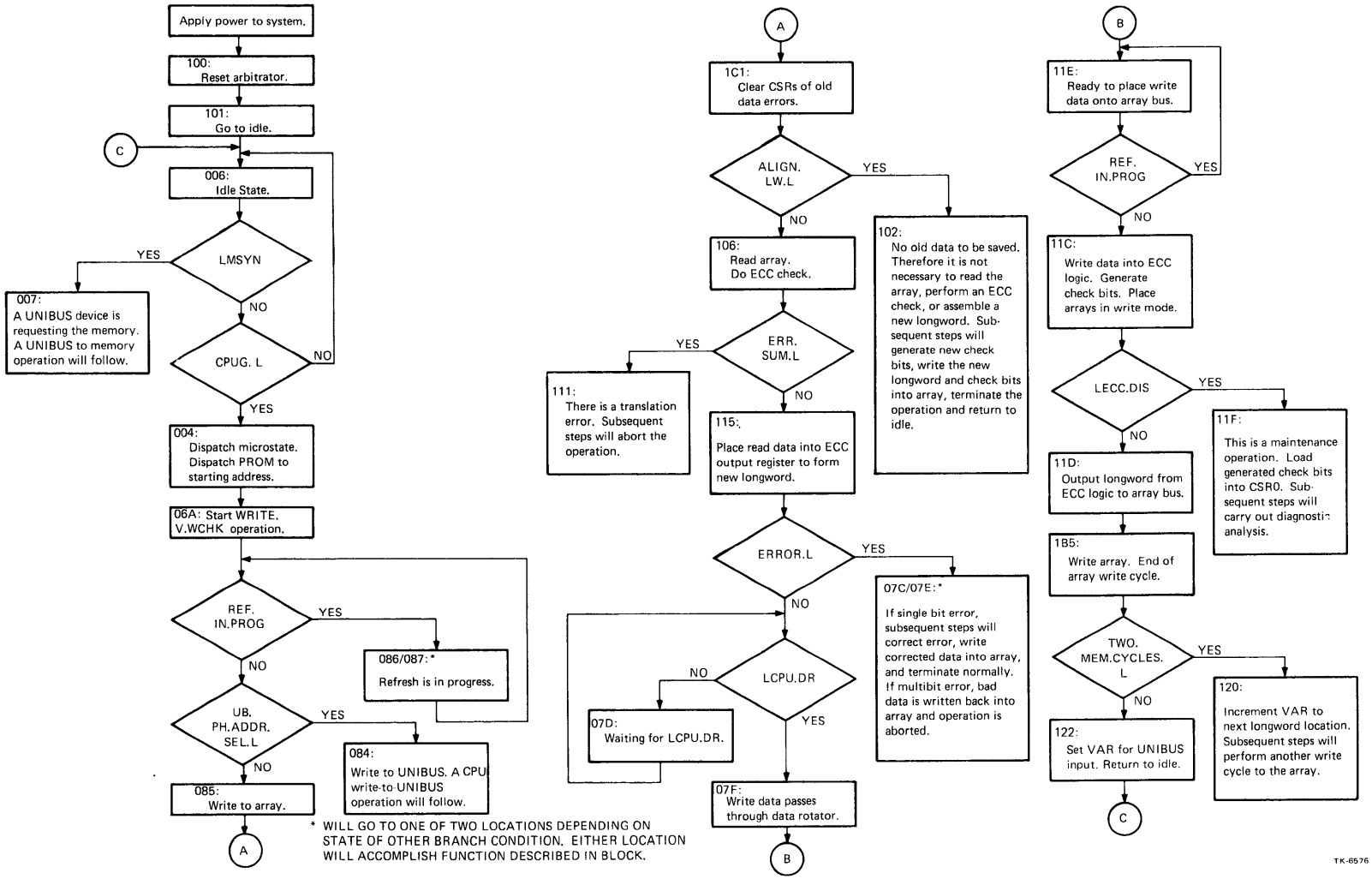


Figure 3-3 Power-Up/Write Array Flow Diagram

APPENDIX A
PROGRAMMED ARRAY LOGIC DEVICES (PAL)

A.1 INTRODUCTION

Programmed array logic devices (PALs) are logic arrays incorporating fusable link technology. PALs are manufactured on a chip using the TTL shottky bipolar process used to make fusable link PROMs. Like PROMs, the PALs may be programmed to give a custom-designed chip unique to a specific requirement.

The basic logic configuration used in PALs is shown in Figure A-1. The circuitry consists of a programmable AND array connecting to a fixed OR array. Note that the AND array shown in the basic logic configuration has only four programmable (fusable-link) inputs and two fixed OR outputs. In the actual PAL circuits used in the VAX-11/730, up to 32 programmable AND inputs and up to 8 fixed OR inputs are used per output.

An unprogrammed PAL has all fuses intact as indicated in Figure A-1 for the basic PAL circuit. A PAL is programmed by first determining the AND inputs to be used and then "blowing" the links for the unused AND inputs to give the desired AND before OR logic configuration. (A standard PROM programming device is used for this operation.) For example, the upper half of Figure A-2 shows the links blown to implement the XOR function AB V AB in the basic PAL logic configuration. This same logic function may also be represented as shown in the lower half of Figure A-2 where an "X" represents the links that are left intact to perform the logical AND. This last method of showing an AND array configuration is the convention used in the PAL plot listings provided in the VAX-11/730 microfiche set.

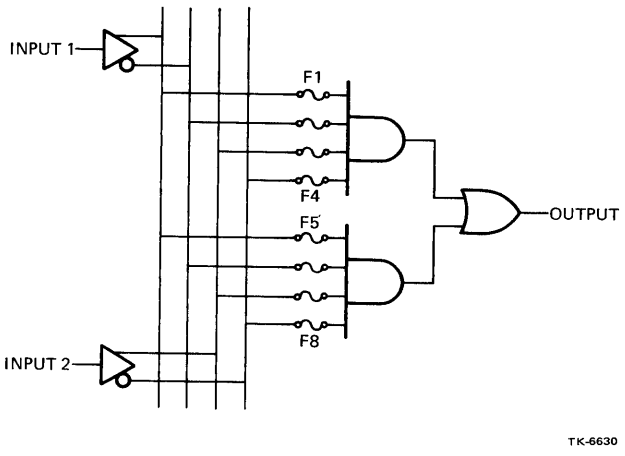


Figure A-1 Basic PAL Logic Configuration

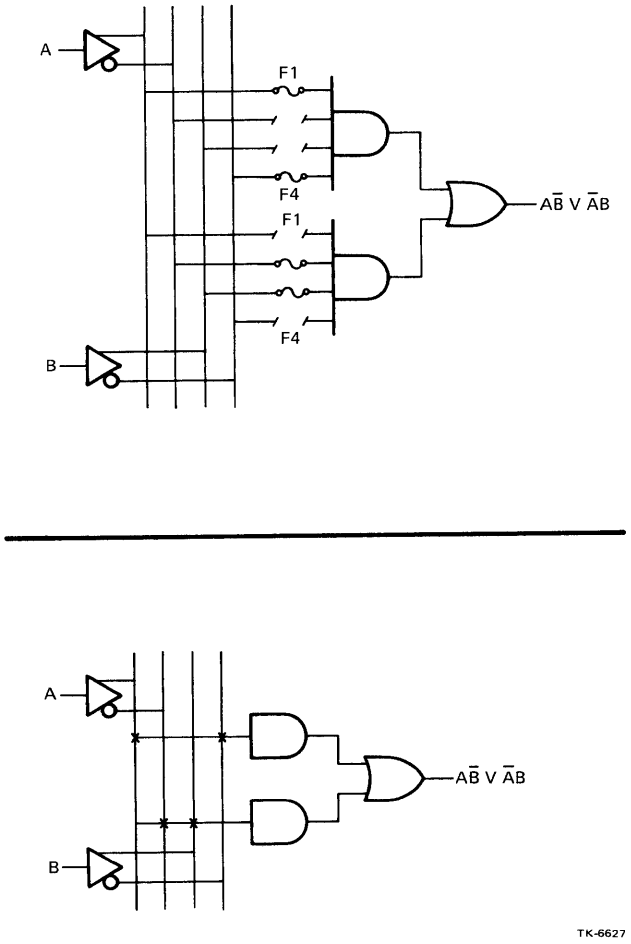


Figure A-2 XOR Logic Function Using PAL Logic

A.2 PAL DEVICE TYPES

The four types of PALs used in the VAX-11/730 are listed in Table A-1. Logic diagrams for each PAL are given at the end of this appendix.

With reference to the logical diagrams, it can be seen that the four PAL devices all use the basic AND before OR logic configuration discussed in Paragraph A.1. However, outputs from the l6L8 gate array chip are inverted and six of the eight outputs feed back to the AND arrays for added capability. In addition, the output inverters for these six outputs may be turned on and off by the AND arrays (programmable I/O). This provides still more logic capability (when the inverter is turned off) in that it allows the corresponding output pin to be used as an input to the AND array just like a designated input pin.

Also note from the logic diagrams that the 16R8 chip has register outputs (D-type flip-flops) and no gate outputs. Again, outputs are fed back to the AND array but not directly by way of the chip's output pins. Instead, the 0 outputs of the flip-flop drive the array. As a result, the output pins cannot be used as inputpins as for a 16L8. The other two PAL types, the 16R6 and the 16R4, have varying combinations of both gate and register outputs on the same chip.

Table A-1 PAL Device Types Used in VAX-11/730

PAL Device Type	Inputs	Outputs	Prog. IO	Register Outputs	Description
16L8	16	8	6	0	AND-OR gate array
16L8	16	8	0	8	AND-OR gate array with registers
16R6	8	8	2	6	AND-OR array with registers
16R4	8	8	4	4	AND-OR array with registers

A.3 PAL SYMBOLOGY

A typical PAL as represented in the VAX-11/730 Engineering Print Set is shown in Figure A-3. Information within the symbol includes the device type, part number, and chip location. For example, the PAL in Figure A-3 is a 16R4 located at E50 with a part number equal to 010K3. The PAL part number distinguishes one programmed PAL from another. Because PALs are programmed for specific applications, it is seldom that more than one PAL will have the same part number.

Inputs to the designated PAL input pins are shown at the left of the PAL symbol. Outputs appear at the right. When an output pin is used as an input pin as discussed in Paragraph A.2, the input signal is entered at the left of the symbol and a dotted line (drawn across the PAL symbol) is used to show the connection to the output pin on the right. Pins having both input and output capability are labeled as I/O on the PAL symbol. Gate outputs not having both input/out capability are labeled with an O. Register outputs are identified by an R. Finally, designated input pins are specified by a D.

A.4 READING THE PAL PLOT LISTING

An example of the PAL plot listing is shown in Figure A-4. The part number and PAL device type (a 16R6 in this case) are at the top of the listing. The input or output associated with each PAL pin is given next. (An NC indicates no connection; VCC indicates the +5 V power source.) A low assertion level for input/output signals on the listing is indicated by a slash (/) immediately preceding the signal name. If there is no slash, the signal is asserted high. It should be remembered that input/output signal names on the listing are sometimes abbreviated and are not exactly the same as in the Engineering Print Set.

The rest of the listing consists of the AND array plots for each output pin. An X represents the fusable links left intact; a dash (-) represents a blown link. More importantly (in order to read the listing), to the right of each line in a plot is the list of signals selected by the intact links that make up the AND inputs. Because these individual AND terms are ORed by the PAL logic, the list of AND terms in the listing (ORed together) result in an easily read Boolean expression that represents the logic function performed. For example, output pin 12 which is a gate output (refer to the 16R6 logic diagram) and the last plot in the listing, has the following input.

```
VCC
START_8085_CYC*10*A14*/RAS
/RAS*STATE
```

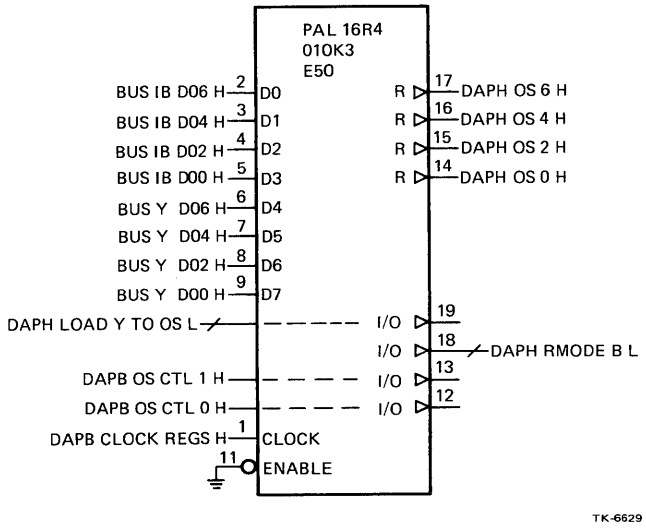


Figure A-3 PAL Symbology (Typical)

The enable level for the gate output inverter (the top line) is connected to VCC, a logical 1. The dashes in the expressions above only represent a space (a blank character) in the signal name. An asterisk (*) between signal names specifies the logical AND operation. Discounting the enable level for the output inverter which in this case is always asserted, this input expression for output pin 12 (/UART_CHIP_SEL) may be read as follows.

$$\text{UART CHIP SEL L} = \text{START 8085 CYC H} \wedge \text{IO H} \wedge \text{A14 H} \wedge \overline{\text{RAS H}}$$
$$\overline{\text{RAS H}} \wedge \text{V STATE H}$$

NOTE: $\wedge$ = AND
V = OR

The PAL circuitry for this output may be represented as shown in the upper half of Figure A-5. The same input using Engineering Print Set conventions is shown in the lower half of Figure A-5.

For a register output, the Boolean expression read from the listing specifies the output signal just as for a gate output. Of course, the output pin is not asserted or negated until the register flip-flop is clocked. Flip-flops are clocked by the positive-going transition of the clock.

When the input statements given in the plot listing are read as AND terms ORed together as just described, it defines the actual PAL circuit and the conditions to make the PAL output go low. (In the example that was given, the PAL output signal was also asserted when it was low.) When a PAL output signal is asserted when at a high level, it is sometimes more convenient to think of the PAL's AND before OR circuit in terms of its equivalent OR before AND configuration. For example, the Boolean equation for register output pin 17 (REFRESH DONE) on the sample listing is as follows when read as AND before OR logic as done previously.

$$\overline{\text{REFRESH DONE L}} = \overline{\text{REQUEST REFER H}} \text{ V } \overline{\text{REFRESH DONE H}} \wedge \overline{\text{REFRESH CYC H}}$$

However, the listing may also be read as OR before AND logic as follows.

REFRESH DONE H = REQUEST REFER L

REFRESH DONE L V REFRESH CYC H

As can be seen, the second expression more clearly indicates that REFRESH DONE is set by the assertion of REQUEST REFR and REFRESH CYC and that it remains set until REQUEST REFR is negated. The AND before OR circuit and the equivalent OR before AND circuit are diagramed in Figure A-6.

To summarize:

1. When the plot listing is read as AND-OR, it specifies the input to give a low PAL output. The output may or may not be asserted low.

2. If the PAL output signal is asserted low, the AND-OR input expression is usually the best way to specify the output.

3. If the PAL output is asserted high, the equivalent OR-AND input expression is usually the best way to specify the output.

A.5 PAL LOGIC DIAGRAMS

The logic diagrams for the 16L8, 16R4, 16R6, and 16R8 PAL devices are shown in Figures A-7 through A-10.

PART NUMBER: 23-004K4-0-0

DEVICE TYPE: PAL16R6

PIN NUMBER = SYMBOL TABLE:

1 = CLOCK	8 = SEL_9600_BAUD	15 = STATE
2 = ALE	9 = RESET	16 =/RAS
3 = REQUEST_REFR	10 = GROUND	17 = REFRESH_DONE
4 = IO	11 = OUT_EN	18 =/START_8085_CYC
5 = A14	12 =/UART_CHIP_SEL	19 =/LONG_CYCLE
6 = 9600_BAUD	13 =/9600_300_BAUD	20 = VCC
7 = 300_BAUD	14 = REFRESH_CYC	

FUSE PLOT: (X = FUSE INTACT , - = FUSE BLOWN)

OUTPIN 19 VCC

```
____ ___X ____ X___ ____ ____ ____ ____  START_8085_CYC*A14
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
```

OUTPIN 18

```
X___ ____ ____ ____ ____ ____ ____ ____  ALE
____ ___X ____ X___ ____ __X_ ____ ____  REFRESH_CYC*START_8085_CYC*A14
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
```

OUTPIN 17

```
____ _X__ ____ ____ ____ ____ ____ ____  /REQUEST_REFR
____ ____ ___X ____ ____ ___X ____ ____  /REFRESH_DONE*/REFRESH_CYC
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
```

```
OUTPIN 16 _____ _____ _____ ___X__ _____ ___X__ _____ _____ /RAS*REFRESH_CYC
          _____ _____ _____ _____X ___X__ _____ _____ _____ RAS*STATE
          _____ _____X __X___ X__X__ _____ _____ _____ _____ START_8085_CYC*/RAS*/IO*A14
          _____ _____ _____ _____ _____ _____ _____ X_____ RESET
          XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
          XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
          XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
          XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

OUTPIN 15 _____ ___X__ _____ _____ _____ _____ _____ _____ /START_8085_CYC
          _____ _____ _____ _____X _____ _____ _____ _____ RAS
          _____ _____ _____ ___X__ _____ _____ _____ _____ /A14
          XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
          XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
          XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
          XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

OUTPIN 14 _____ ____X _____ _____ _____ _____ _____ _____ START_8085_CYC
          _____ _____ _____ ____X _____ ___X__ _____ _____ RAS*REFRESH_CYC
          _____ _____ _____ ____X ___X__ _____ _____ _____ RAS*STATE
          _____ ___X__ _____ _____ _____ _____X _____ _____ /REFRESH_CYC*/REQUEST_REFR
          _____ _____ ___X__ _____ _____ _____X _____ _____ /REFRESH_CYC*REFRESH_DONE
          X_____ _____ _____ ___X__ ____X ____X _____ _____ /REFRESH_CYC*/RAS*ALE*/STATE
          _____ _____ _____ _____ _____ _____ _____ X____ RESET
          XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

OUTPIN 13 _____ ___ _____ ___ _____ X____ _____ X____ _____ SEL_9600_BAUD*9600_BAUD
          _____ ___ _____ _____ _____ _____ X_____X___ _____ /SEL_9600_BAUD*300_BAUD
          XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
          XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
          XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
          XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
          XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
          XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

OUTPIN 12                                                 VCC
          _____ ____X X____ X_X___ _____ _____ _____ _____ START_8085_CYC*IO*A14*/RAS
          _____ _____ _____ ___X__ ___X__ _____ _____ _____ /RAS*STATE
          XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
          XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
          XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
          XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
          XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
```

Figure A-4 PAL Plot Listing

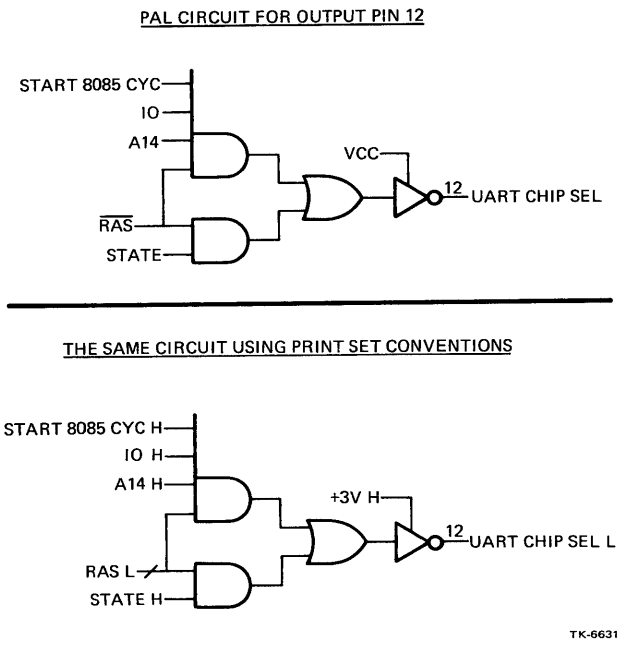


Figure A-5 PAL Circuit for Output Pin 12 of Sample Listing

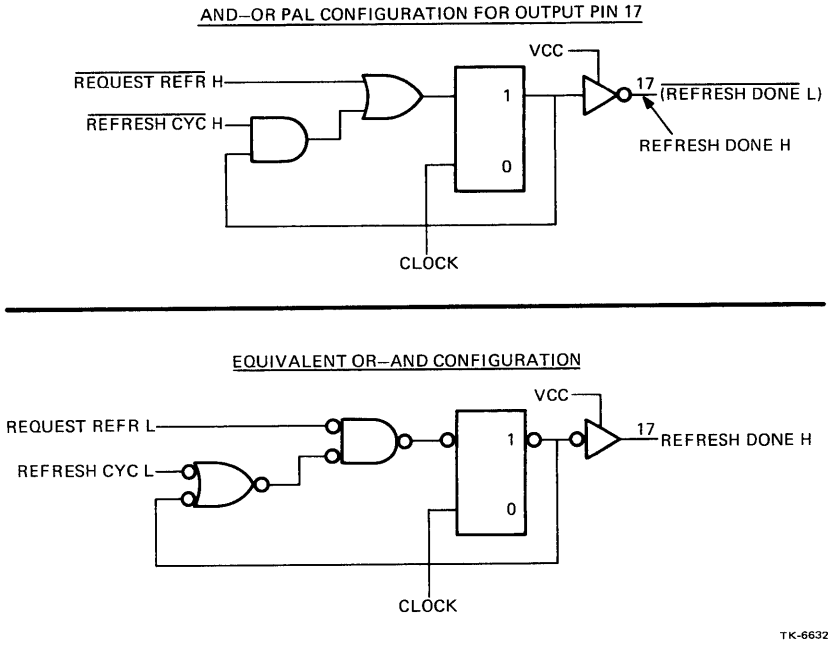


Figure A-6 PAL Circuit and Equivalent Circuit for Output Pin 17 of Sample Listing

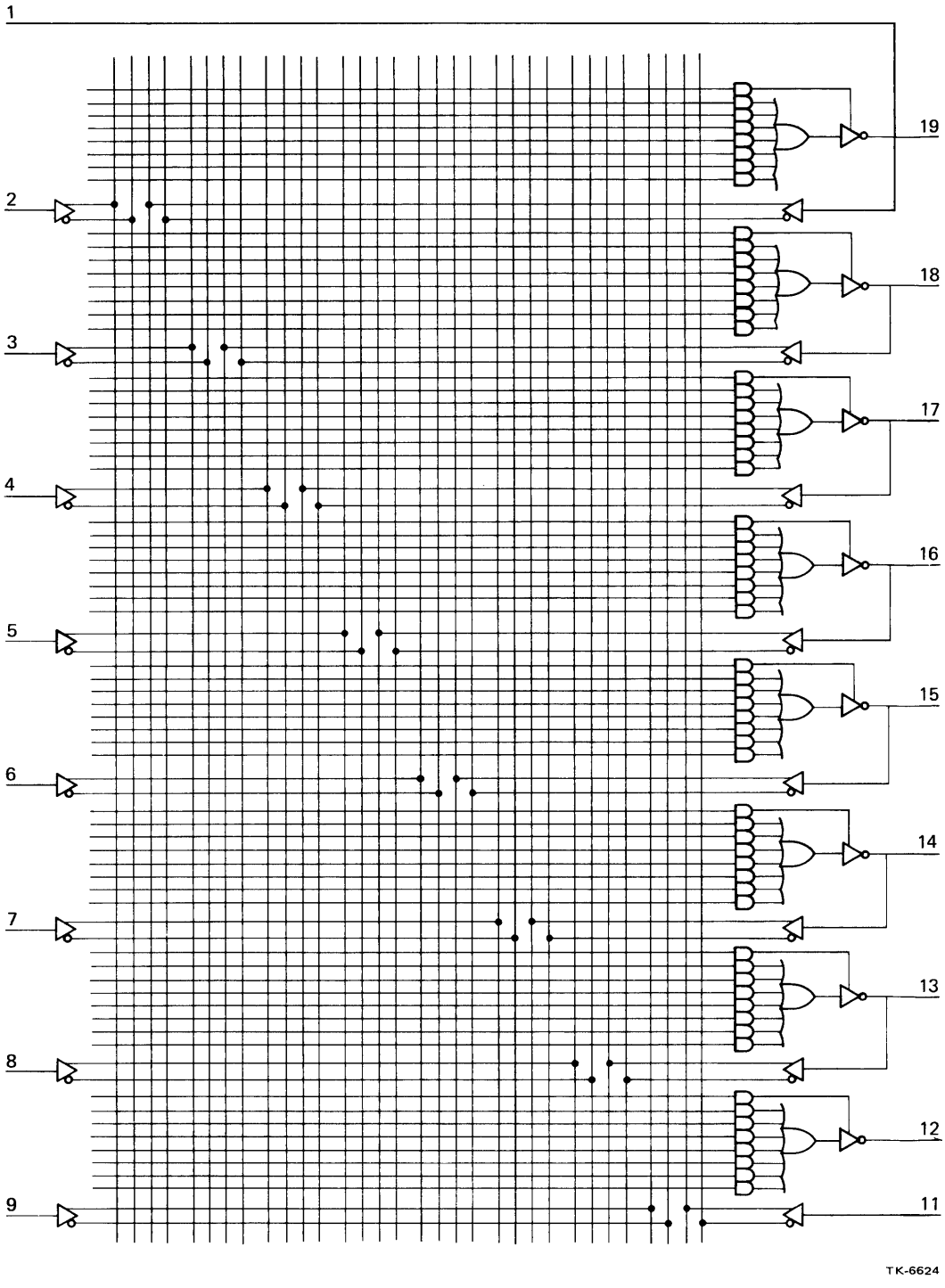


Figure A-7 PAL 16L8 Logic Diagram

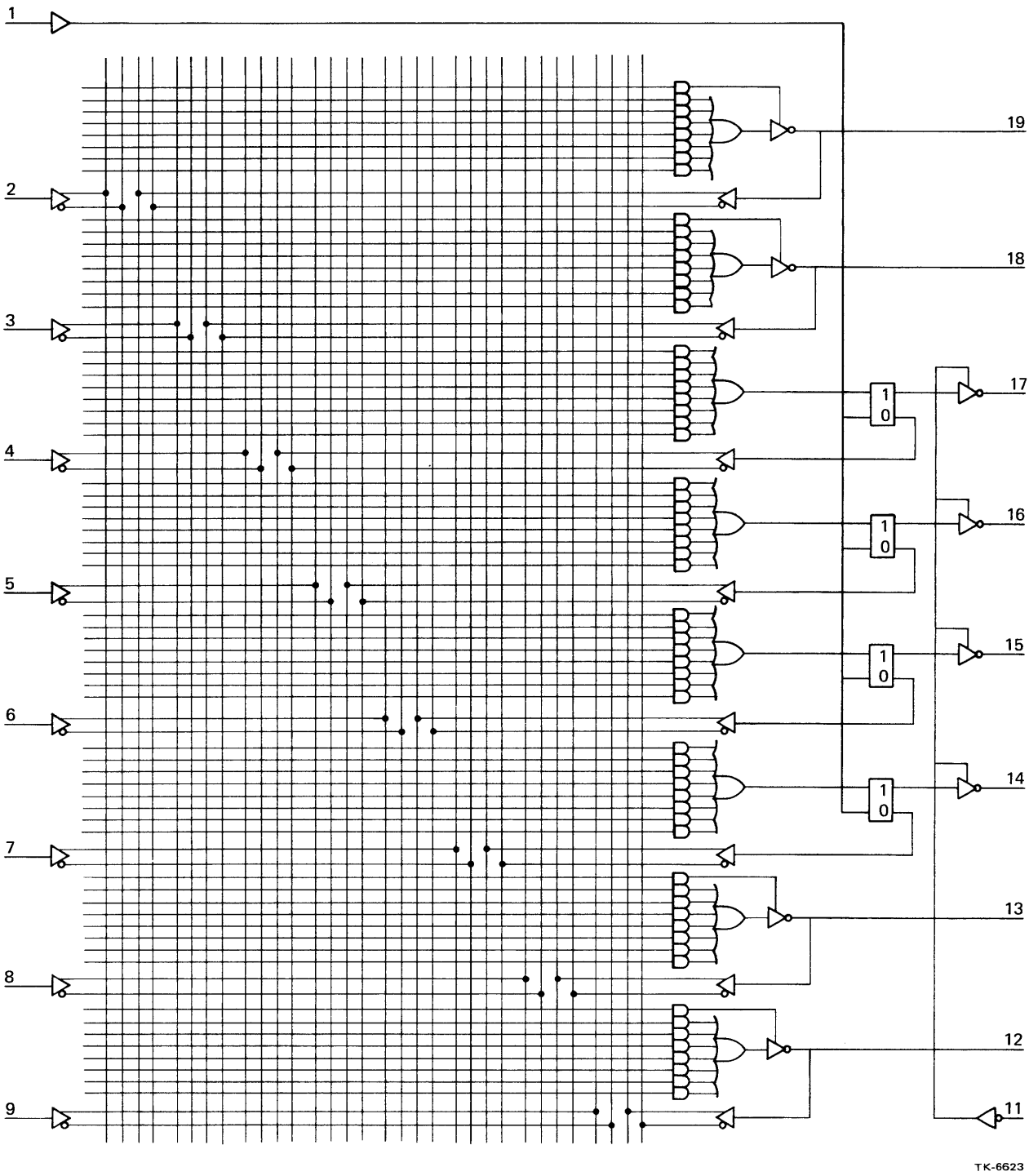


Figure A-8 PAL 16R4 Logic Diagram

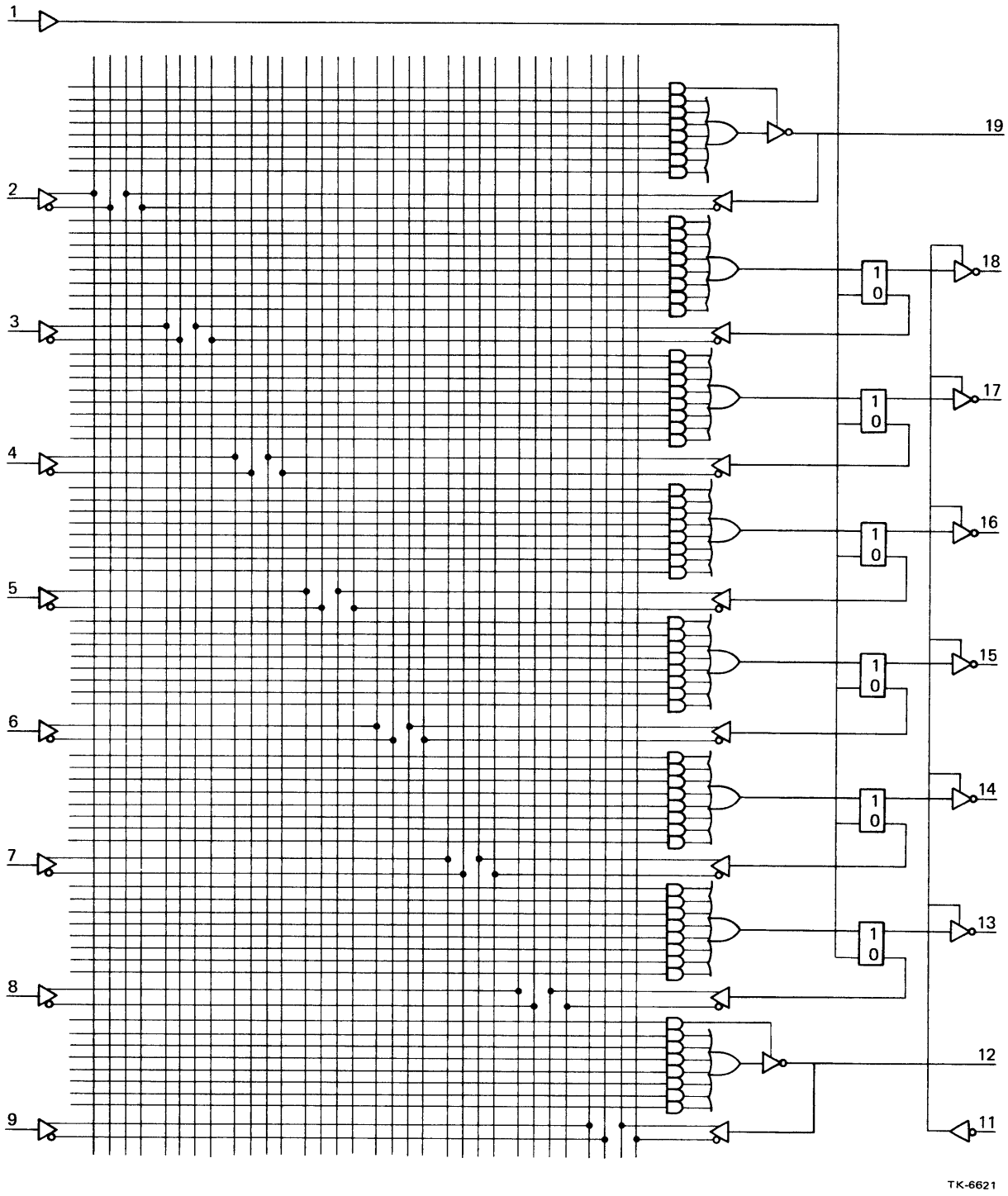


Figure A-9 PAL 16R6 Logic Diagram

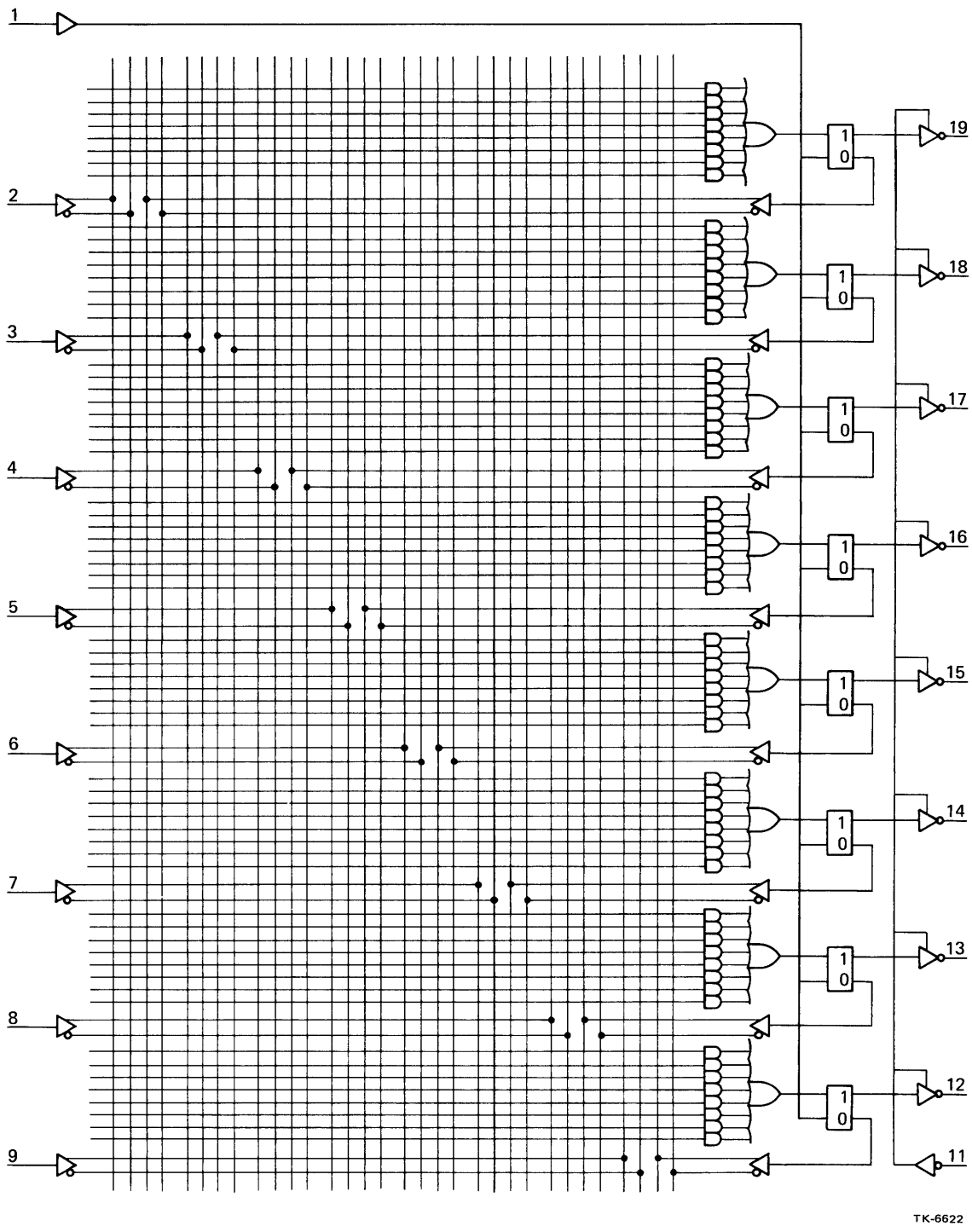


Figure A-10 PAL 16R8 Logic Diagram

APPENDIX B
FLOW DIAGRAM SYMBOLS

B.1 GENERAL

The flow diagram symbols used in this manual are defined in Figure B-1.

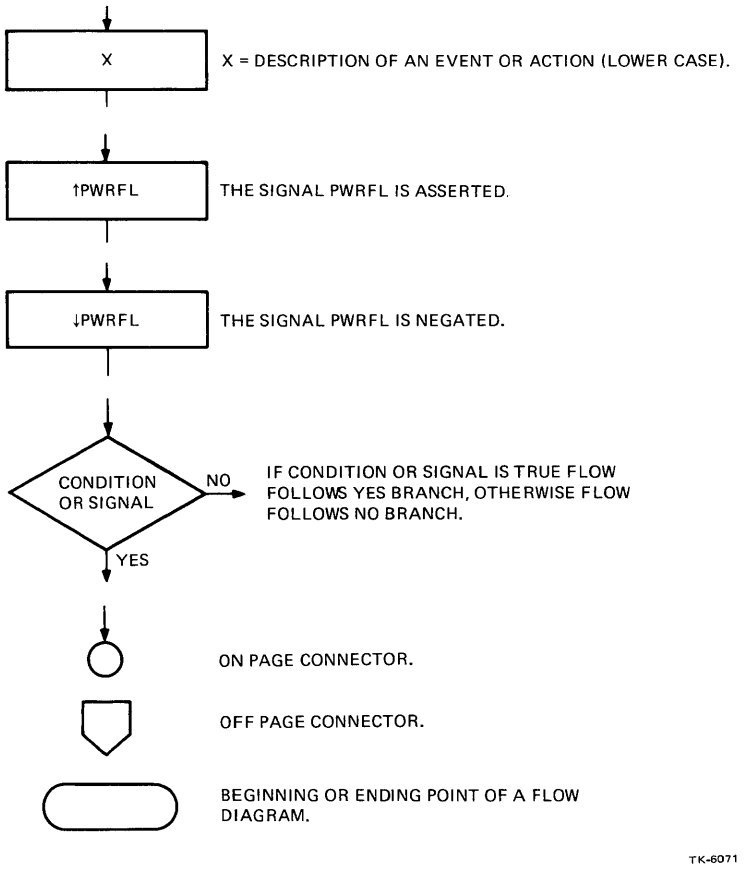


Figure B-1 Flow Diagram Symbols

APPENDIX C
MAINTENANCE FEATURES

C.1 GENERAL

There are five maintenance dispatches provided in the memory controller microcode to aid in diagnosing memory failures. The dispatches start with a simple diagnostic and increase in complexity building upon the successful completion of the previous diagnostic. The UNIBUS maintenance diagnostic (RD.MAINT.UBS) has four operations selected by the value of the DIAG CHK bit and the ECC DIS bit in CSR1. The ECC maintenance diagnostic (MAINT.ECC.DAT) has five operations selected by the value of DIAG CHK, ECC DIS, and LVA00. The maintenance dispatches are summarized in Table C-1. For a description of the diagnostics, how to use them, and how to interpret the results, refer to "Microdiagnostics for VAX-11/730 MCT module" (document number ENKCC).

In addition to the five maintenance dispatches, the memory controller parity logic can be checked via the TB PAR DIAG bit in CSR1. When set, this bit will cause a parity error and assert TB PAR ERR in CSR1 on any CPU access to the translation buffer.

Table C-1 Diagnostic Dispatch Functions

Diagnostic Dispatch	Dispatch Operation	Operation Control Bits			Function
		DIAG CHK	ECC DIS	LVA00	
READ.MAINT.	–	–	–	–	This diagnostic dispatch reads the virtual address register as data.
RD.MAINT. VAR.INC	–	–	–	–	This diagnostic dispatch reads the incremented virtual address register as data.
RD.MAINT. BRANCH.CK	–	–	–	–	This diagnostic dispatch checks the branch conditions DIAG CHK, ECC DIS, and LVA00. The virtual address register is incremented after each successful test and the final value is returned as data.
RD.MAINT. UBS	Address check	0	0	–	The VAR contents are gated onto the UNIBUS address lines. The VAR is loaded from the UNIBUS address lines. The VAR contents are returned as data.

Table C-1 Diagnostic Dispatch Functions (Cont)

Diagnostic Dispatch	Dispatch Operation	Operation Control Bits DIAG CHK	ECC DIS	LVA00	Function
	Data check	0	1	–	The VAR contents are gated onto the MC BUS. The MC BUS is enabled through the data drivers to the UNIBUS data lines. The UNIBUS data lines are enabled to the ECC data in latch. The data in latch is enabled to the MC BUS and returned as data.
	Sync check	1	0	–	This operation checks that BUS MSYN and BUS SSYN can be asserted and negated on the UNIBUS. The VAR is incremented after each successful check and the contents of the VAR is returned as data.
	Control check	1	1	–	This operation checks that IC0, BUS C1, and UNIBUS ACTIVITY can be asserted and negated. The VAR is incremented at each successful pass and is returned as data.
MAINT.ECC.DAT	Data rotator	0	0	0	Data from the CPU is loaded into the MOR and then returned to the CPU. The data rotators are disabled.
	ECC check generator	0	1	0	The CPU sends data to the ECC data in latch. ECC generates check bits on the data. The check bits go into CSR0. The data is returned to the CPU.
	ECC correction	1	0	0	This operation takes data from the CPU into the ECC data in latch. The check bits come from CSR1. The data from the ECC data out latch is returned to the CPU. Data error information is logged into CSR1.
	ECC check bit path	1	1	0	Check bits from CSR1 go into the ECC check bit latch. Check bits are then clocked into CSR0. Data sent by the CPU is stored in ECC latches and returned to the CPU as read data.
	Write data error check	X	X	1	Data from the CPU and check bits from CSR1 go into a memory location physically addressed by the contents of the VAR. (Can load single or double error as desired.)

X = Don't care

Fold Here

DO NOT TEAR – FOLD HERE AND TAPE

digital

No Postage Necessary if Mailed in the United States

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO 33 MAYNARD, MA

POSTAGE WILL BE PAID BY ADDRESSEE

Digital Equipment Corporation
Educational Services/Quality Assurance
12 Crosby Drive, BU/E08
Bedford, MA 01730

VAX-11/730 MEMORY SYSTEM TECHNICAL DESCRIPTION **Reader's Comments**

Your comments and suggestions will help us in our continuous effort to improve the quality and usefulness of our publications.

What is your general reaction to this manual? In your judgment is it complete, accurate, well organized, well written, etc? Is it easy to use? _____

_____

_____

_____

What features are most useful? _____

_____

_____

_____

What faults or errors have you found in the manual? _____

_____

_____

_____

Does this manual satisfy the need you think it was intended to satisfy? _____

Does it satisfy *your* needs? _____ Why? _____

_____

_____

_____

Please send me the current copy of the *Documentation Products Directory,* which contains information on the remainder of DIGITAL's technical documentation.

Name _____ Street _____

Title _____ City _____

Company _____ State/Country _____

Department _____ Zip _____

Additional copies of this document are available from:

Digital Equipment Corporation
Accessories and Supplies Group
P.O. Box CS2008
Nashua, New Hampshire 03061

Attention: Documentation Products
Telephone: 1-800-258-1710

Order No. **EK-MS730-TD-001**

MY