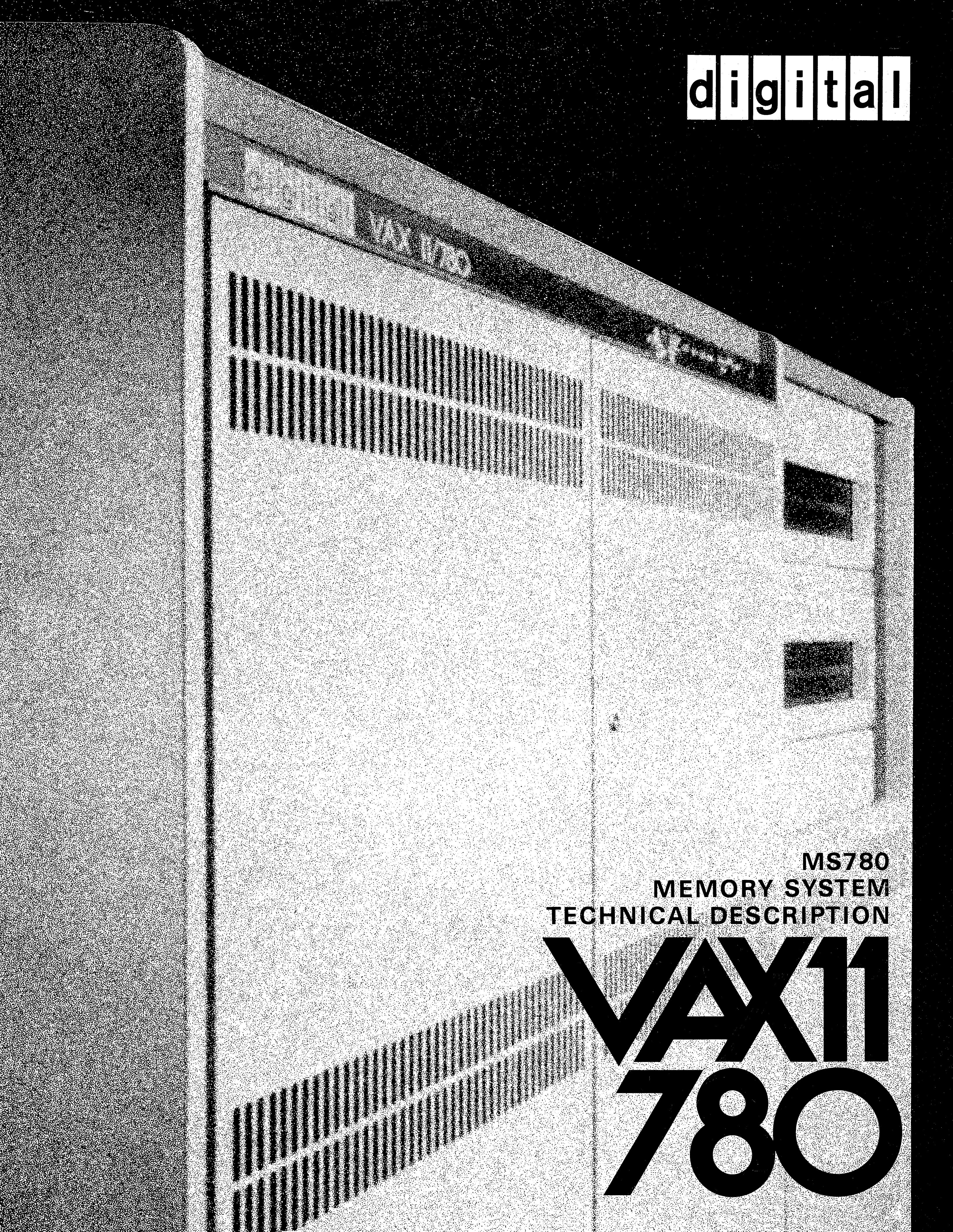


MS780 MEMORY SYSTEM TECHNICAL DESCRIPTION

VAX11 780

EK-MS780-TD-001

MS780 MEMORY SYSTEM TECHNICAL DESCRIPTION

digital equipment corporation • maynard, massachusetts

1st Edition, August 1978

The material in this manual is for informational purposes and is subject to change without notice.

Digital Equipment Corporation assumes no responsibility for any errors which may appear in this manual.

Printed in U.S.A.

This document was set on DIGITAL's DECset-8000 computerized typesetting system.

CONTENTS

CONTENTS (Cont)

Page

FIGURES

TABLES

CHAPTER 1
INTRODUCTION

1.1 MANUAL SCOPE

The MS780 Memory System Technical Description consists of three chapters:

1. Introduction
2. Functional Description
3. Detailed Description

The Introduction gives a brief physical description of the memory system and describes its basic operation. Also included is a brief description of error checking and correction (ECC), the metal oxide semiconductor (MOS) storage chips, and synchronous backplane interconnect (SBI) protocol. The Functional Description provides a description of the information decoding sequence and memory cycle sequence, with special emphasis on address decoding. Simplified timing diagrams are also included. The Detailed Description discusses individual logic sections. Detailed timing diagrams are included to illustrate signal timing relationships.

1.2 RELATED DOCUMENTS

Table 1-1 is a list of related hardware manuals and their availability.

Table 1-1 Related Hardware Manuals

Title	Document Number	Notes
Translation Buffer, Cache, SBI Control Technical Description	EK-MM780-TD-001	In Microfiche Library.** Available on hard copy.*
KA780 Central Processor Technical Description	EK-KA780-TD-PRE	In Microfiche Library.**
DW780 Unibus Adaptor Technical Description	EK-DW780-TD-PRE	In Microfiche Library.** Available on hard copy.*
REP05/REP06 Disk Subsystem Technical Description	EK-REP06-TD-PRE	In Microfiche Library.**
KC780 Console Interface Board Technical Description	EK-KC780-TD-001	In Microfiche Library.** Available on hard copy.*

*This document can be ordered from:

Digital Equipment Corporation
444 Whitney Street
Northboro, MA 01532
Attn: Communication Services (NR2/M15)
Attn: Customer Services Section

**For information concerning microfiche libraries, contact:

Digital Equipment Corporation
Micropublishing Group, PK3-2/T12
129 Parker Street
Maynard, MA 01754

1.3 OVERVIEW

The MS780 main memory is a MOS random access memory (RAM) that is designed to interface with the SBI of a VAX 11/780 system. The memory subsystem consists of a controller and one to sixteen array boards that utilize either 4K or 16K N-channel MOS IC storage elements. Each array board can thus contain 64K or 256K bytes of memory, giving the system a capacity of either 1 or 4 megabytes, depending on the size of storage chips used.

The memory is capable of random access read or write operations to a single 32-bit longword or extended 64-bit quadword. Write operations can also be executed on byte boundaries within a longword or quadword. Memory features also include an error checking and correcting scheme that can detect all double-bit errors and detect and correct all single-bit errors. The error detection and correction algorithm requires an entire quadword of data; thus, during any type of read or write operation, an entire quadword of data is fetched from the array. Eight check bits are stored with each quadword and accessed with the data to determine its integrity.

Figure 1-1 illustrates a simplified block diagram of the memory subsystem. As shown in this figure, the controller consists of three extended hex boards interconnected by two internal buses, namely the file information bus and the MOS data bus. The letters in the lower left corner of each board in the diagram indicate the engineering print sheets. The logic of each board is briefly described in the following paragraphs.

SBI Interface Module – The M8214 SBI interface module contains all SBI related logic. This includes SBI transceivers, address decode logic, address correction logic for the interleaved and non-interleaved cases, function decode logic, confirmation and fault logic, and arbitration logic. The MSB board also includes a command file to buffer up to four information transfers from the SBI.

Control and Timing Module – The M8213 control and timing module controls all internal memory activity by generating control and timing signals for the data path module and array modules. Decoded control and command signals from the SBI interface board are used for cycle decode, I/O data multiplexer control, ECC control (error checking and correction), and array timing. The MCN logic also includes a refresh counter, logic for array size correction due to chip size, the address register, three configuration registers, and a bootstrap ROM.

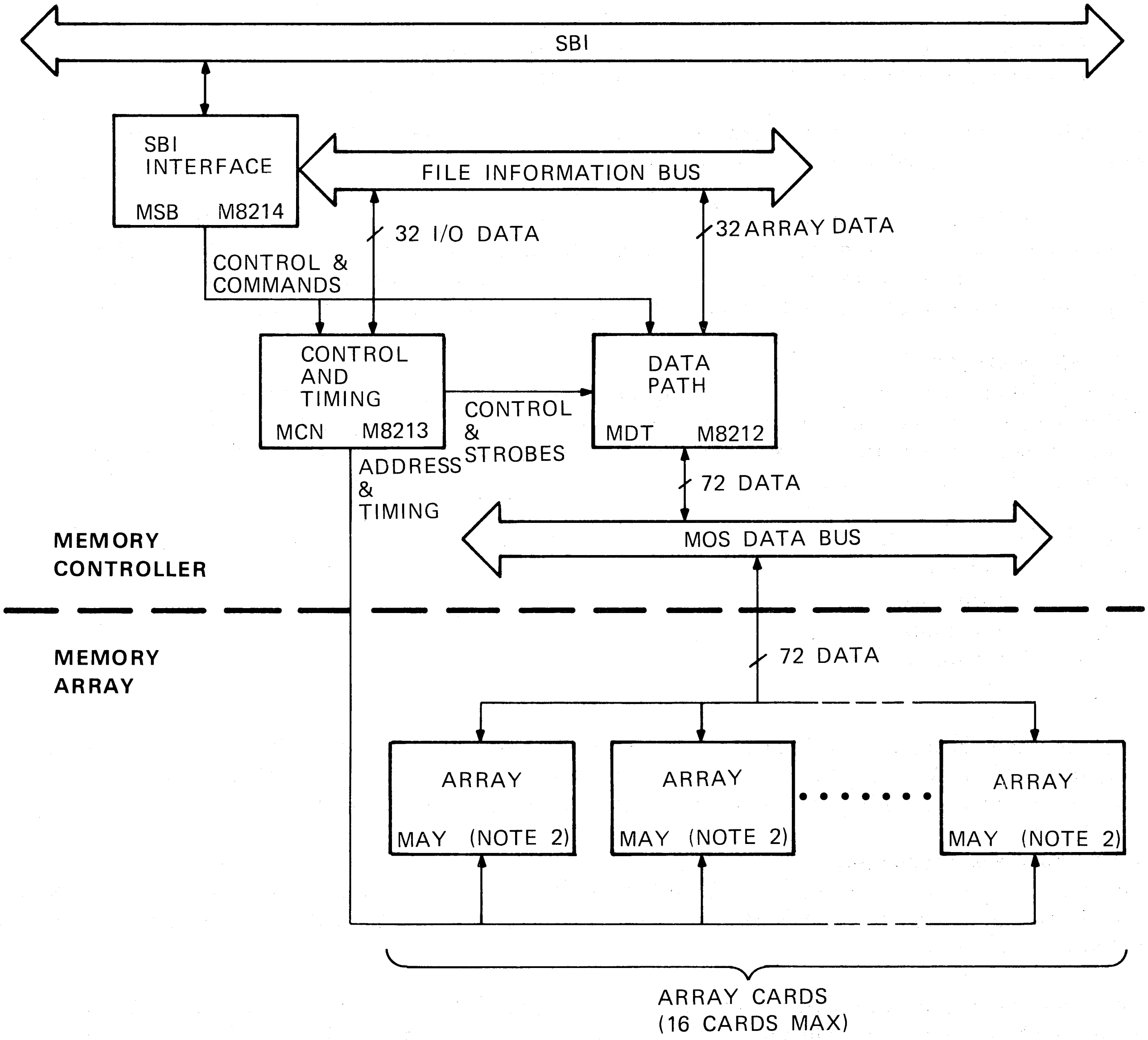
Data Paths – The M8212 data path module contains the central data path between the array boards and the SBI interface board, and all associated logic. This includes data transmit and receive latches, byte control, and all error detection and correction logic.

Array Modules – The memory controller is capable of interfacing two types of array cards as described at the bottom of Figure 1-1: an 8K version (8K × 72) and a 32K version (32K × 72). The M8211 array module is the 8K version and the M8210 is the 32K version. Both versions contain N-channel MOS ICs and their associated drive circuitry. All array modules receive address and control signals from the control and timing module (MCN) and use the MOS data bus as a data path.

Internal Buses – The memory controller contains two tristate internal buses. Each module interconnecting the buses thus contains tristate drivers and receivers.

The file information bus is used to transfer data from the command file on the SBI interface module to the control and timing module and the data path module. Likewise, it transfers array data from the data path, or I/O data from the control and timing module, to the SBI transceivers on the SBI interface module.

The MOS data bus provides a data path between the array modules and the data path module. All transfers consist of one quadword of data (64 bits) and eight corresponding check bits to accommodate the ECC logic on the data path module.



NOTES:

1. EACH MEMORY LOCATION CONTAINS 72 BITS: 8 BYTES OF DATA, AND 8 CHECK BITS FOR ERROR CHECKING AND CORRECTION.

2. EACH ARRAY BOARD CONTAINS 8K MEMORY LOCATIONS IF 4K CHIPS ARE USED, OR 32K LOCATIONS IF 16K CHIPS ARE USED. THE ARRAY MODULE WITH 4K CHIPS IS THE M8211; THE ARRAY MODULE WITH 16K CHIPS IS THE M8210.

TK-0184

Figure 1-1 Memory Subsystem Simplified Block Diagram

1.4 BASIC MEMORY OPERATIONS

The MS780 is capable of six operations:

Read Masked	Write Masked
Extended Read	Extended Write Masked
Interlock Read Masked	Interlock Write Masked

Basically, a write masked is executed to transfer one to four bytes of data to memory. A read masked, however, is only capable of transferring four bytes of data from memory.

An extended read is executed to transfer eight bytes of data (two longwords) from memory to a requesting nexus. An extended write masked, on the other hand, provides a byte selectable transfer of up to eight bytes to memory. Interlock read masked and interlock write masked perform the same function as read masked and write masked but also provide process synchronization (Paragraph 1.5.1).

1.5 SBI PROTOCOL

The MS780 memory system is connected directly to the SBI and thus conforms to SBI protocol. The SBI provides a checked parallel information transfer path that is synchronous with a common system clock. In each clock period or cycle, interconnect arbitration, information transfer, and transfer confirmation may occur in parallel.

The 84 lines of the SBI are divided into the following functional groups:

1. Arbitration
2. Information
3. Confirmation
4. Interrupt
5. Control

Each of these groups is explained in the following paragraphs, along with an explanation of SBI synchronization. A description of the interrupt lines, however, has been omitted because they are not accessed by memory. Likewise, other non-memory related SBI information has been omitted.

1.5.1 SBI Synchronization

Six control group lines are clock signals and are used as a universal time base for all nexus connected to the SBI (including memory). All SBI clock signals are generated on the CPU clock module and provide a 200 ns clock period.

The clock signals, in conjunction with the standard nexus clock logic, provide the derived clocks within each attached nexus to synchronize SBI activity. Two clock signals (TPH and TPL) derive the basic time states. The remaining four (P CLK H, P CLK L, PD CLK H, and PD CLK L) are phased clocks and help compensate for the clock distribution skew due to cable, backplane, and driver/receiver propagation delays.

1.5.1.1 Derived Time States – The derived clocks within each nexus define four, 50 ns (nominal) time states in one clock period. The time states (T0, T1, T2, and T3) determine the transmit and receive times on the SBI, with T0 representing the start of a particular clock period or SBI cycle. Figure 1-2 illustrates the phase and timing relationships required to generate the individual derived time states.

1.5.1.2 Basic Transmit/Receive Timing – Information is received or transmitted over the SBI at the same instant during any SBI cycle. Immediately prior to T0 a transmitting nexus enables its transmitters. At T0 the transmitters are clocked and the content is enabled to the information path of the SBI. In the case of receive data, nexus receiver latches are opened at T2 and latched at T3. Note that the information may be considered undefined at the outputs of the bus receivers between T2 and T3; only after T3 is the information considered defined. All checking, decoding, and subsequent decision making is then based on these latched signals.

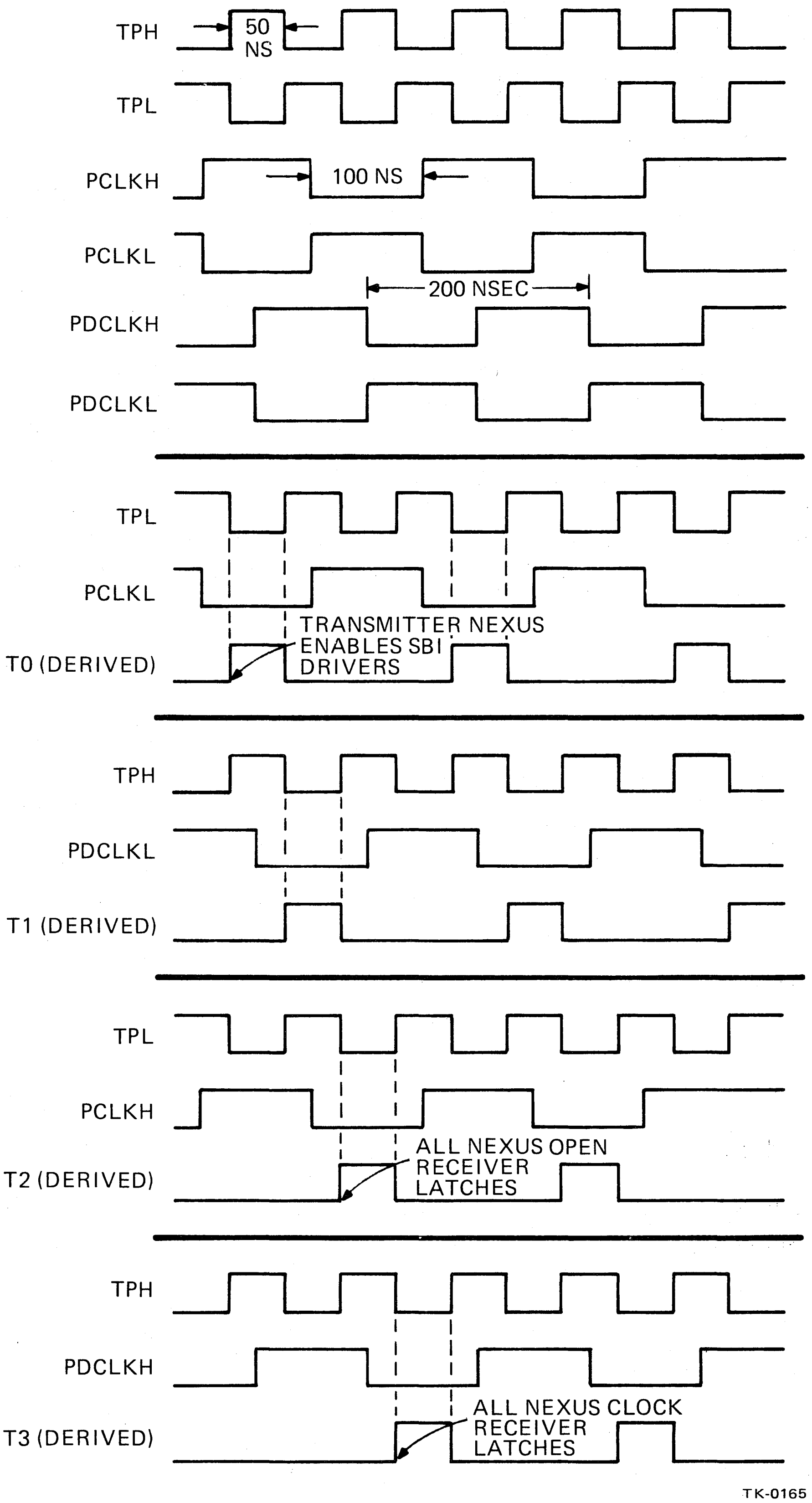


Figure 1-2 SBI Time and Phase Relationships

1.5.2 Arbitration Group

The arbitration lines [Transfer Request TR (15:00)] allow up to 15 nexus to arbitrate for the information lines (information transfer group). One arbitration line is assigned to each nexus to establish the fixed priority access. Priority is set in an ascending order: TR15 to TR00. The priority level (TR line) assignment of each nexus is selectable at system build time.

The 15 highest priority nexus are assigned TR15 through TR01. The lowest priority level is reserved for the CPU (the 16th nexus) and requires no actual TR signal line. The highest priority level, TR00, is reserved as a hold signal for those nexus that require more than one successive SBI cycle.

Arbitration on the SBI is considered decentralized; that is, each nexus contains its own arbitration logic. A nexus requests control of the information path by asserting its assigned TR line at T0 of an SBI cycle. At T3 of the same SBI cycle, the nexus examines (arbitrates) the state of all higher priority TR lines. If no higher TR lines are asserted, the requesting nexus assumes control of the information path at T0 of the following SBI cycle. At this T0 time state, the nexus negates its TR line, and asserts command/address or data information on B(31:00). In addition, if a write type exchange is specified, the nexus asserts T00 to retain control of adjacent SBI cycles.

If higher priority TR lines are asserted, the requesting nexus cannot gain control of the information path. The nexus keeps its TR line asserted and again examines the state of higher priority lines at T3 of the next SBI cycle. As before, if no higher TR lines are asserted, the nexus assumes information path control at T0.

1.5.3 Information Transfer Group

The three types of SBI information transfer formats utilized for memory operations are command/address, read data, and write data. (Interrupt summary read and interrupt summary response are not used by memory.) All three formats are divided into five fields. The fields are: parity, tag, identity, mask, and information (Figure 1-3). Each information group field is described in detail in the following subsections.

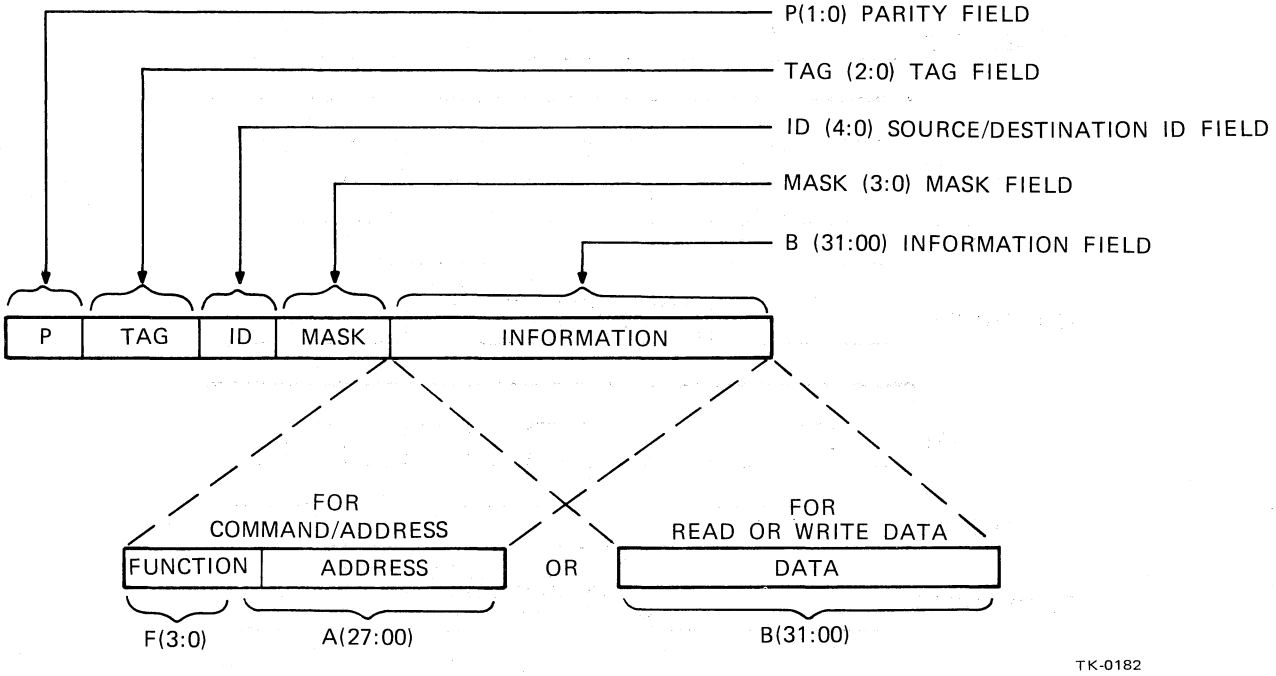


Figure 1-3 SBI Information Group Fields

1.5.3.1 Parity Field – The parity field (P1:0) provides even parity for detecting single bit errors in the information group. A transmitting nexus generates P0 as parity for TAG(2:0), ID(4:0), and M(3:0). The P1 parity bit is generated for B(31:00). P0 and P1 are generated such that the sum of all logic one bits in the checked field, including the parity bit, is even. With no SBI transmissions, the information transfer path assumes an all zeros state; thus, P(1:0) should always carry even parity. Any transmission with odd parity is considered an error.

1.5.3.2 Tag Field – The tag field [TAG(2:0)] is asserted by a transmitting nexus to indicate the information format being transmitted on the information lines. The tag field determines the contents of the B field. The following subsections describe each information type, tag code, and associated field content.

Command/Address Tag – A tag field content of 011 specifies that the content of B(31:00) is a command/address word. ID(4:0) asserted at this time is a unique code identifying the logical source (commander) of the command. As shown in Figure 1-3, B(31:00) is divided into a function field and an address field to specify the command and its associated address.

The ID field code of a command/address represents the logical source and the address field specifies the logical command destination. For a read type command, the receiving nexus will reassert this ID when transmitting the read data.

The 28 bits of the address field define a 268,435,456 longword address space (1,073,741,824 bytes), which is divided into two sections. Addresses 0–$7FFFFFF_{16}$ (A27=0) are reserved for primary-memory. Addresses 8000000_{16}–$FFFFFFF_{16}$ (A27=1) are reserved for device control and I/O registers.

Read Data Tag – A tag field content of 000 specifies that B(31:00) contains data requested by a previous read type command. In this case, ID(4:0) is a unique code that was received with the read command and identifies the logical destination of the requested data. The retrieved data may be one of three types: read data, corrected read data, or read data substitute, where the particular type is identified by M(3:0).

Read data is the normal expected error-free data having M(3:0) = 0000. Corrected read data is data in which an error was detected and subsequently corrected by the ECC logic of the device transmitting the read data. In this case, the mask field flags the corrected data with M(3:0) = 0001. Read data substitute represents data in which an error was detected but the ECC logic was unable to correct it. In this case, B(31:00) will contain the substitute data in the form of uncorrected data. The mask field flags the uncorrected data with M(3:0) = 0010.

Write Data Tag – A tag field content of 101 specifies that B(31:00) contains the write data for the location specified in the address field of the previous write command. The write data will be asserted on B(31:00) in the SBI cycle immediately following the command/address cycle. The mask field specifies bytes within B(31:00) for the operation.

1.5.3.3 Identifier Field – The ID field [ID(4:0)] contains a code that identifies the logical source or logical destination of the information contained in B(31:00). In a command/address or write data format it identifies the source. In a read data format it identifies the destination.

ID codes are assigned to nexus that are capable of issuing a command/address. Because of this, memory is not assigned an ID code. Every nexus, other than memory, is assigned an ID code that corresponds to the TR line which it operates. For example, a nexus assigned TR 05 would also be assigned an ID code of 5.

1.5.3.4 Mask Field – The four-bit mask field has two interpretations (Figure 1-4). For the first interpretation, the mask is encoded to specify particular data bytes of an addressed memory location for the memory operation. As shown at the top of Figure 1-4, each bit in the mask field of a command/address or write data corresponds to a particular data byte.

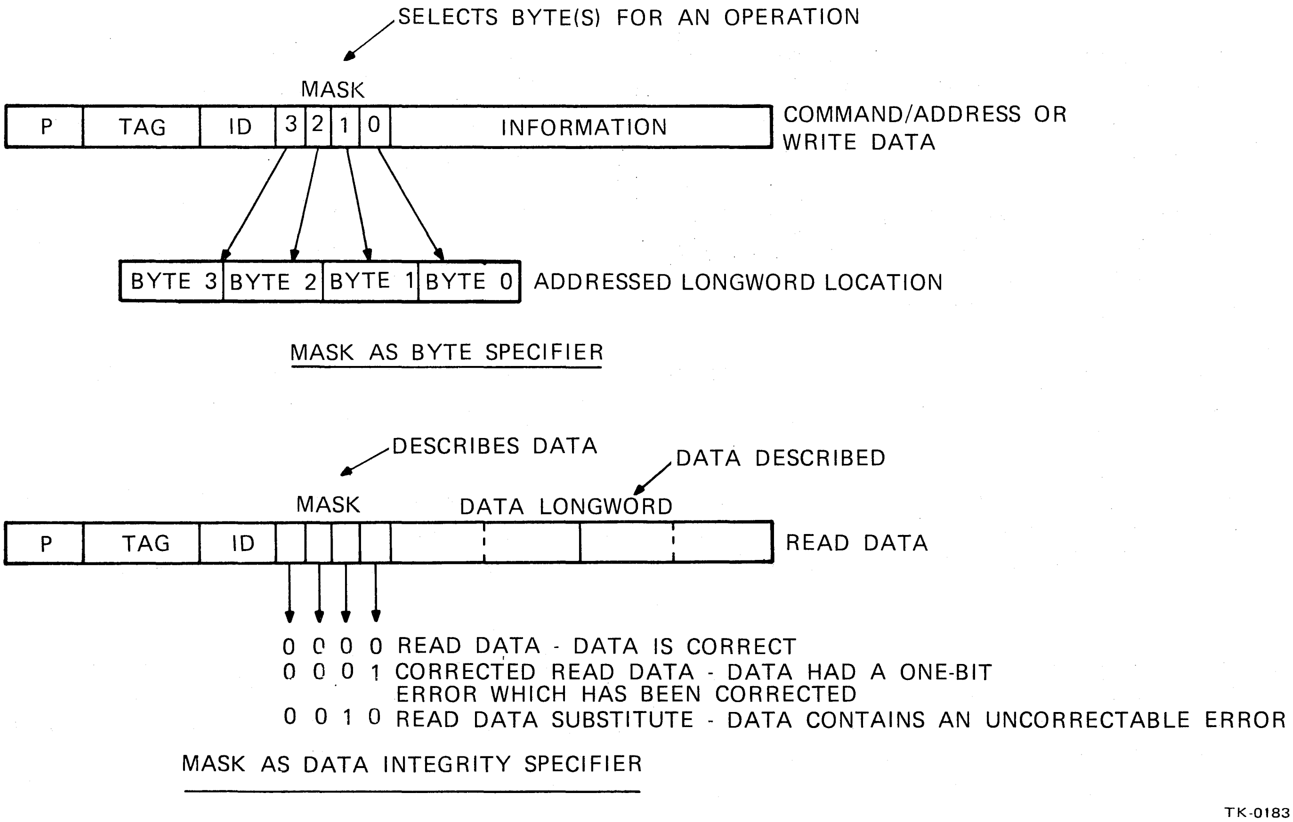

SELECTS BYTE(S) FOR AN OPERATION
MASK
P
TAG
ID
3
2
1
0
INFORMATION
COMMAND/ADDRESS OR WRITE DATA
BYTE 3
BYTE 2
BYTE 1
BYTE 0
ADDRESSED LONGWORD LOCATION
MASK AS BYTE SPECIFIER
DESCRIBES DATA
DATA DESCRIBED
MASK
DATA LONGWORD
P
TAG
ID
READ DATA
0 0 0 0 READ DATA - DATA IS CORRECT
0 0 0 1 CORRECTED READ DATA - DATA HAD A ONE-BIT ERROR WHICH HAS BEEN CORRECTED
0 0 1 0 READ DATA SUBSTITUTE - DATA CONTAINS AN UNCORRECTABLE ERROR
MASK AS DATA INTEGRITY SPECIFIER
TK-0183

Figure 1-4 Mask Field Interpretation

As mentioned in Paragraph 1.5.3.2, the second interpretation is used when TAG(2:0) = 000 (read data). In this case the mask bits specify the integrity of the data of the information field. The integrity of the data is specified in one of three categories as summarized in Figure 1-4: correct, corrected, or uncorrectable.

1.5.3.5 Information Field – As shown in Figure 1-3, the information field consists of two format types: data or command/address. The information field of a command/address format is divided into two subfields: a 4-bit function and a 28-bit address. The 4-bit function specifies the operation to be performed. Table 1-2 lists the function codes for memory operations. The 28-bit physical address selects a memory location (longword address) for the selected operation. Byte selection for the operation is specified in the mask field.

The information field of a read data or write data format contains one longword of data for transfer to or from a selected address. During a read data operation, the complete longword is transmitted even though the mask field may have selected fewer bytes. The receiving nexus ignores the unrequested bytes. Similarly, during a write data operation, an entire longword is received and bytes not specified by the mask field are ignored.

Table 1-2 Function Bits of a Command/Address

Function Bits				Memory
3	**2**	**1**	**0**	**Operations**
0	0	0	1	Read Masked
0	0	1	0	Write Masked
0	1	0	0	Interlock Read Masked
0	1	1	1	Interlock Write Masked
1	0	0	0	Extended Read
1	0	1	1	Extended Write Masked

1.5.4 Response Group

The three response lines are divided into two fields: confirmation [CNF(1:0)], and fault (FAULT). CNF(1:0) informs the transmitter whether the information was correctly received, or if the receiver can process the command. FAULT is a cumulative error indication of protocol or information path malfunction and is asserted with the same timing as the confirmation field.

Either field is transmitted two cycles after each information transfer. Confirmation is delayed to allow the information path signals to propagate, be checked, and decoded by all receivers, and to be generated by the responder. During each cycle, every nexus in the system receives, latches, and makes decisions on the information transfer signals. Except for multiple bit transmission errors or nexus malfunction, only one of the nexus receiving the information path signals will recognize an address or ID code. This nexus then asserts the appropriate response in CNF.

Any (or all) nexus may assert FAULT after detecting a protocol or information path failure. However, a nexus asserting FAULT may not assert CNF(1:0).

Table 1-3 lists the confirmation codes and their interpretation.

Table 1-3 Confirmation Codes

CNF Code		Mnemonic	Indication
1	**0**		
0	0	N/R	No response to selection (the unasserted state)
0	1	ACK	Positive acknowledgement to the transfer
1	0	BSY	Response to command/address transfer only: successful selection of a nexus which is presently unable to execute the command
1	1	ERR	Response to command/address transfer only: successful selection of a nexus which cannot execute the command.

1.5.5 Control Group

The control group functions synchronize system activites and provide specialized system communications. The clock functions provide SBI activity synchronization and are described in Paragraph 1.5.1. The Interlock line is one of the system communication functions and is described in Paragraph 1.6. The remaining control lines are described in the following paragraphs.

DEAD – The DEAD signal indicates a dc power failure in the clock circuits or bus terminating networks. Nexus will not assert any SBI signal while DEAD is asserted. Thus, nexus prevent invalid data from being received while the bus is in an unstable state.

The assertion of the power supply DC LO to the clock circuits or terminating networks causes the assertion of DEAD. DEAD is asserted asynchronously to the SBI clock and occurs at least 2 μs before DC LO is negated. The negation of DC LO negates DEAD.

FAIL – FAIL can be asserted by any nexus whose existence in the system is necessary for proper restart after a power failure. The FAIL signal notifies the CPU that a system restart operation cannot be initiated. A nexus asserts the Fail (FAIL) signal asynchronously to the SBI clock when the power supply AC LO signal is asserted to that nexus. The assertion of FAIL inhibits the CPU from initiating a power up service routine. FAIL is negated asynchronous to the SBI clock when all nexus that are required for the power up operation have detected the negation of AC LO. The CPU samples the FAIL line following the power down routine (assertion of FAIL) to determine if the power up routine should be initiated.

UNJAM – The UNJAM signal restores (initializes) the system to a known, well-defined state. The UNJAM signal is asserted only by the CPU through a console function, and is detected by all nexus connected to the SBI. The duration of the UNJAM pulse is 16 SBI cycles; it is negated at T0.

For the assertion of UNJAM, the CPU asserts TR00 for 16 SBI cycles. The CPU continues to assert TR00 for the duration of UNJAM and for a minimum of 15 SBI cycles after the negation of UNJAM. This use of TR00 ensures that the SBI is inactive preceding, during, and after the UNJAM operation.

If asserted, UNJAM is received by every nexus at T3 and a restore sequence is begun. Any current operation of short duration is not aborted if that operation might leave the nexus in an undefined state. Nexus do not perform operations using the SBI during the assertion of UNJAM. In addition, the nexus is in an idle state, with respect to SBI activity, at the conclusion of the UNJAM pulse.

While UNJAM is asserted, nexus cannot assert FAULT. However, a CPU asserting FAULT prior to UNJAM will continue to do so to preserve the content of the nexus FAULT status registers. The restore sequence (UNJAM asserted) should not cause a nexus to pass through any states that will assert any SBI lines. All read commands issued before the UNJAM are cancelled.

In the event of a power failure during UNJAM, some nexus will assert FAIL and/or DEAD. The restore sequence should cause the nexus to clear any existing ALERT status bits and subsequently negate ALERT.

ALERT (part of the Interrupt Group) – A nexus asserts ALERT when any of its ALERT status bits are set. The bits are set during the following events:

1. during power failure at the nexus when the assertion of power supply AC LO is recognized;
2. during the restoration of power when the negation of AC LO is recognized;
3. when other environmental conditions, such as overtemperature, are detected.

Each nexus maintains bits in its status register to indicate conditions that cause assertion of ALERT.

The ALERT line is the logical OR of the ALERT status bits and is asserted synchronously to the SBI clock. ALERT status bits are cleared when written as logic one; when written as logic zero, they are not changed. These status bits are also cleared when the UNJAM signal is received.

A nexus asserting ALERT continues to assert ALERT until:

1. all ALERT status bits are cleared (written with a logic one),
2. UNJAM signal is received,
3. nexus loses dc power.

The negation of ALERT is synchronous to the SBI clock and occurs within two cycles of the write data transmission used to clear the ALERT condition.

1.5.6 SBI Summary

Table 1-4 summarizes the signal fields associated with each functional group.

Table 1-4 SBI Field Summary

Field	Description
Arbitration Group	
Arbitration Field [TR(15:00)]	Establishes a fixed priority among nexus for access and control of the information transfer path.
Information Transfer Group	
Information Field [B(31:00)]	Bidirectional lines that transfer data, command/address, and interrupt information between nexus.
Mask Field [M(3:0)]	Primary function: encoded to indicate a particular byte within the 32-bit information field [B(31:00)]. Secondary function: in conjunction with the tag field indicates a particular type of read data.
Identifier Field [ID(4:0)]	Identifies the logical source or destination of information contained in B(31:00)
Tag Field [TAG(4:0)]	Defines the transmit or receive information types and the interpretation of the content of the ID and information fields.
Function Field [F(3:0)]	Specifies the command code, in conjunction with the tag field.
Parity Field [P(1:0)]	Provides even parity for all information transfer path fields.
Response Group	
Confirmation Field [CNF(1:0)]	Asserted by a receiving nexus to specify one of four response types and indicate its capability to respond to the transmitter's request.
Fault Field (FAULT)	A cumulative error line that indicates one of several errors on the SBI.

Table 1-4 SBI Field Summary (Cont)

Field	Description
Interrupt Request Group	
Request Field* [REQ(7:4)]	Allows a nexus to request an interrupt to service a condition requiring CPU intervention. Each request line represents a level of nexus request priority.
Alert Field (ALERT)	A cumulative status line that allows those nexus not equipped with an interrupt mechanism to indicate a change in its power or operating conditions.
Control Group	
Clock Field (CLOCK)	Six control lines that provide the clock signals necessary to synchronize SBI activity.
Fail Field (FAIL)	A single, line from nexus required to initiate a system bootstrap operation.
Dead Field (DEAD)	A single line to the CPU to indicate an impending clock circuit or bus terminating network power failure.
Unjam Field (UNJAM)	A single line from the CPU to attached nexus that restores the nexus to a known state.
Interlock Field (INTLK)	A single line that provides coordination among nexus responding to ensure exclusive access to shared data structures.

*Not used by memory.

1.6 MEMORY CYCLES; GENERAL DESCRIPTION

To perform a memory cycle, a requesting nexus arbitrates for control of the SBI's information lines. Having gained control of the bus, the nexus then transmits a command/address format onto the bus. All subsystems monitor the bus by checking for parity and decoding the tag. A decoded command/address tag also initiates an address and function decode. If the decoded address corresponds to the memory subsystem, and memory is not busy, a memory cycle is initiated (providing no faults are detected). If, however, the memory is presently executing a cycle, the command information is stored in the command file until the present cycle is complete. Either way, memory notifies the requesting nexus that the message has been received by asserting a response on the confirmation lines.

Figure 1-5 illustrates the command sequence for each memory operation. Each sequence is calibrated in SBI cycles. Note that more than one nexus may use the SBI lines in any given cycle, provided there is only one nexus utilizing each group of lines. That is, during one SBI cycle, while one nexus is transferring a command/address or data, a second nexus may use the arbitration lines while a third nexus transmits a confirmation on the response lines. The sequence of each operation, however, must be preserved.

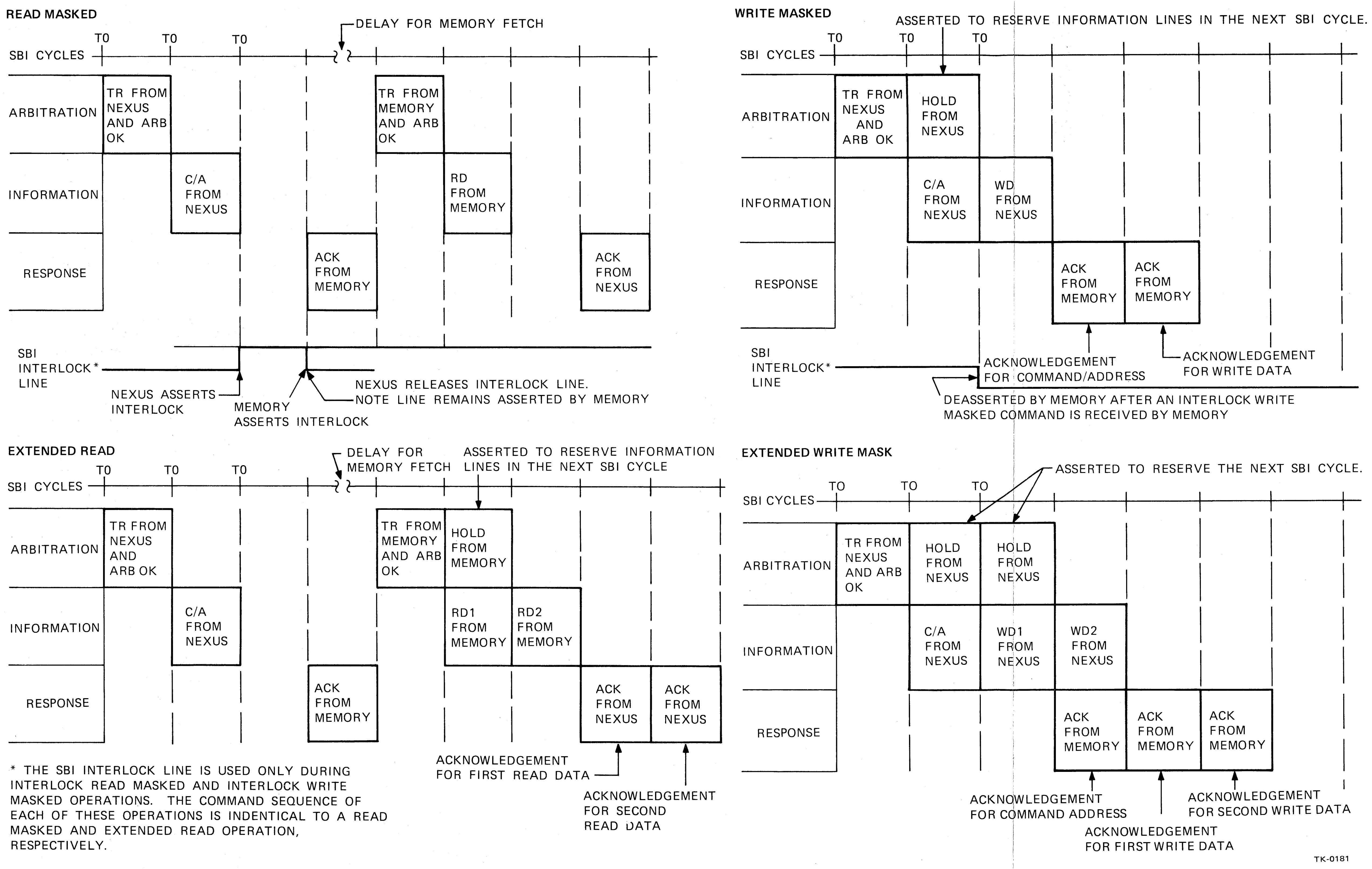

READ MASKED
DELAY FOR MEMORY FETCH
T0
T0
T0
SBI CYCLES
ARBITRATION
TR FROM NEXUS AND ARB OK
TR FROM MEMORY AND ARB OK
INFORMATION
C/A FROM NEXUS
RD FROM MEMORY
RESPONSE
ACK FROM MEMORY
ACK FROM NEXUS
SBI INTERLOCK* LINE
NEXUS ASSERTS INTERLOCK
MEMORY ASSERTS INTERLOCK
NEXUS RELEASES INTERLOCK LINE.
NOTE LINE REMAINS ASSERTED BY MEMORY
WRITE MASKED
ASSERTED TO RESERVE INFORMATION LINES IN THE NEXT SBI CYCLE.
T0
T0
T0
SBI CYCLES
ARBITRATION
TR FROM NEXUS AND ARB OK
HOLD FROM NEXUS
INFORMATION
C/A FROM NEXUS
WD FROM NEXUS
RESPONSE
ACK FROM MEMORY
ACK FROM MEMORY
SBI INTERLOCK* LINE
ACKNOWLEDGEMENT FOR COMMAND/ADDRESS
ACKNOWLEDGEMENT FOR WRITE DATA
DEASSERTED BY MEMORY AFTER AN INTERLOCK WRITE MASKED COMMAND IS RECEIVED BY MEMORY
EXTENDED READ
DELAY FOR MEMORY FETCH
ASSERTED TO RESERVE INFORMATION LINES IN THE NEXT SBI CYCLE
T0
T0
T0
SBI CYCLES
ARBITRATION
TR FROM NEXUS AND ARB OK
TR FROM MEMORY AND ARB OK
HOLD FROM MEMORY
INFORMATION
C/A FROM NEXUS
RD1 FROM MEMORY
RD2 FROM MEMORY
RESPONSE
ACK FROM MEMORY
ACK FROM NEXUS
ACK FROM NEXUS
ACKNOWLEDGEMENT FOR FIRST READ DATA
ACKNOWLEDGEMENT FOR SECOND READ DATA
* THE SBI INTERLOCK LINE IS USED ONLY DURING INTERLOCK READ MASKED AND INTERLOCK WRITE MASKED OPERATIONS. THE COMMAND SEQUENCE OF EACH OF THESE OPERATIONS IS INDENTICAL TO A READ MASKED AND EXTENDED READ OPERATION, RESPECTIVELY.
EXTENDED WRITE MASK
ASSERTED TO RESERVE THE NEXT SBI CYCLE.
T0
T0
T0
SBI CYCLES
ARBITRATION
TR FROM NEXUS AND ARB OK
HOLD FROM NEXUS
HOLD FROM NEXUS
INFORMATION
C/A FROM NEXUS
WD1 FROM NEXUS
WD2 FROM NEXUS
RESPONSE
ACK FROM MEMORY
ACK FROM MEMORY
ACK FROM MEMORY
ACKNOWLEDGEMENT FOR COMMAND ADDRESS
ACKNOWLEDGEMENT FOR SECOND WRITE DATA
ACKNOWLEDGEMENT FOR FIRST WRITE DATA
TK-0181

Figure 1-5 Command Sequences

The interlock operations, interlock read masked and interlock write masked, are not shown because their sequences are identical to read masked and write masked, respectively. During interlock read masked, however, the SBI interlock line is asserted by the commander for one cycle when the command is issued. Before the cycle is ended, the responding memory controller also asserts the interlock line. The interlock line, therefore, continues to be asserted even though it is dropped by the commander. With the interlock line set, memory responds with busy (BSY) to any interlock read commands. The interlock line remains asserted until interlock write masked is received. With this, memory releases the interlock line and executes the interlock write masked (the sequence of interlock write masked is identical to write masked).

For the read masked and extended read operations, memory arbitrates for control of the SBI information lines while it is fetching the requested data from its array cards.

A transmitting nexus can reserve the information lines of the succeeding SBI cycle by asserting HOLD on the arbitration lines. HOLD can be asserted only by a nexus that has asserted a transfer request (TR) and gained control of the SBI information lines (Paragraph 1.5.2).

1.7 ACCESS AND CYCLE TIMES

Table 1-5 lists the minimum cycle times and read access times for each memory operation in SBI cycles and microseconds. These values are quoted with the assumption that the command file is empty (i.e., queuing not required). Values are given for each operation with and without error correction. Definitions are also included specifying time references.

Table 1-5 Access and Cycle Times

Memory Operation	Access Time (in Bus Cycles)		Cycle Time (in Bus Cycles)	
	No Error	**With Error**	**No Error**	**With Error**
Read Masked	3 (0.6 μs)	4 (0.8 μs)	3 (0.6 μs)	6 (1.3 μs)
Extended Read	4 (0.8 μs)	5 (1.0 μs)	4 (0.8 μs)	6 (1.3 μs)
Write Masked	–	–	7 (1.4 μs)	7 (1.4 μs)
Extended Write Masked	–	–	7 (1.4 μs)	–
	Duration from leading edge of memory GO, to the leading edge of valid data at the input of the SBI drivers in the memory controller.		Duration between two consecutive memory GO pulses during a successful read or write.	

1.8 ERROR CHECKING AND CORRECTION (ECC)

In general, odd or even parity is used as a means of detecting an error in a given data word. As a simple example, odd parity is developed by adding the number of ones in a data word. If the total is even, a one is stored as an extra bit (parity bit) along with the data. When the data word is read from memory, the number of ones in the data word are again added and a new parity bit developed. This new parity bit is then compared with its corresponding stored parity bit. The agreement or disagreement between the two bits indicates a no-error or error condition in the data word. This scheme, however, makes it possible to detect only an odd number of errors in the data word.

The ECC scheme used in the memory subsystem is capable of detecting a single- or double-bit error. It is also capable of correcting all single-bit errors. This is accomplished by storing eight parity bits, called check bits, along with the 64 data bits in each memory location. Each check bit is generated by parity checking selected groups of data bits in the given data quadword. When parity is again checked during a read, an incorrect bit will be detected by the parity checking logic and will develop a unique 8-bit syndrome that identifies the bit in error. Error correction logic may thus correct the bit in error. There are 72 unique syndromes pointing to individual bits in the coded quadword.

1.9 REFRESH

The storage device used in the MS780 is a dynamic MOS cell in which a data bit is represented by a charge. This charge can be discharged over a period of time resulting in loss of data. Because of this, MOS storage cells must be recharged through a memory operation called refresh.

The discharge time of a MOS cell is approximately 2 ms. This means that for a 4K MOS device organized internally as a 64 × 64 matrix, a row refresh is required approximately every 32 μs. Note also that a 16K MOS device requires a refresh cycle approximately every 16 μs because it is organized internally as a 128 × 128 matrix. The duration of a refresh cycle is approximately equal to 1 memory cycle (500 ns). During the refresh cycle, the 64 cells in each row are read and rewritten internally to restore the full charge. Refresh is performed on all chips of a given row on all array boards simultaneously.

Because of these requirements refresh is performed periodically in the MS780. For 4K array boards, a refresh cycle is performed approximately every 28 μs to ensure reliability over temperature and voltage margins. For 16K array boards, a refresh cycle is performed approximately every 14 μs.

CHAPTER 2
FUNCTIONAL DESCRIPTION

2.1 INTRODUCTION
The execution of a memory operation can be divided into two sequences because of the design architecture employed. The two sequences are the information decoding sequence and the memory cycle sequence.

Basically, during an information decoding sequence, the memory interface latches information from the SBI (as all nexus on the SBI) and decodes it. The validity of memory-destined information is determined and the request information is stored in the command file for a memory cycle sequence. Various flags and indicators are also stored with the request as a result of the decoding.

The memory cycle sequence is an internal memory operation in which a valid memory request is executed. During this sequence the information stored during the information decoding sequence, is accessed from the command file and the command is executed.

The following paragraphs provide a general description of both sequences using the command file as a midpoint. Flow diagrams of the sequences are provided as a summary (Paragraphs 2.5 and 2.6).

2.2 INFORMATION DECODING SEQUENCE; GENERAL
To initiate a memory cycle, a requesting nexus arbitrates and gains control of the SBI information lines. Having gained control, the nexus (commander) then transmits a command/address format onto the SBI.

Information decoding in the memory controller, as in any nexus, begins at T3 of every SBI cycle. At this instant any signals on the SBI information lines are latched through the memory's SBI transceivers. Between T3 and T1 of the next SBI cycle, the information is parity checked and the tag is decoded. If the decoded tag indicates a command/address format, the address and function are also decoded. At T2, following decoding, the proper confirmation (also fault and interlock bits, if applicable) is set for transmission onto the SBI at the next T0 (Figure 2-1).

The address decode selects the memory array, configuration registers, or bootstrap ROM for the memory operation. The function decode selects the type of memory operation and determines if the command file has enough space to accept the full command string (i.e., the command/address and any subsequent write data formats). If the file has space, the information field of the command/address format is stored along with the decoded address until the memory controller becomes available to execute the memory cycle. If the file is full or does not have enough space to accept the full command string, a busy confirmation is transmitted. In this case, subsequent entries into the file continue to be inhibited until information is passed from the file to the file information bus for a memory cycle execution. Note space for write data in the file is checked when the command/address is received and the function is decoded.

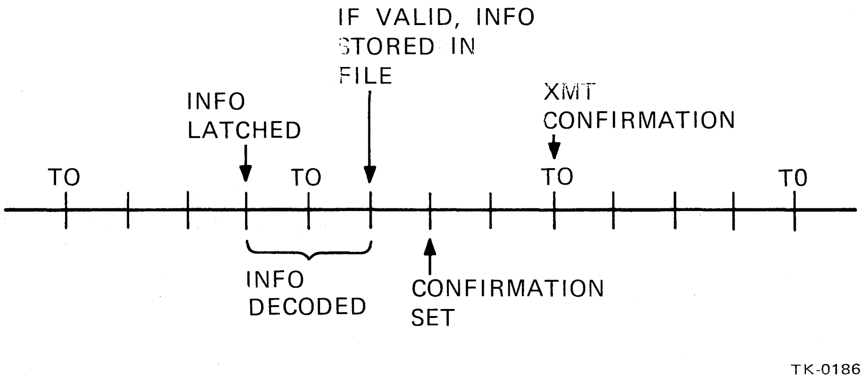


Figure 2-1 Typical Information Decode Timing in the Memory Controller

2.2.1 Parity Check

A parity bit is generated by the SBI transceivers for every four bits of an information format received. The parity check is performed on these bits and the two parity bits received.

A parity error in a received command/address format aborts the memory cycle immediately. In this case, the bad information is not reserved in the command file. For the command/address of a write cycle, the following write data format(s) are also ignored.

In the case of a parity error in a write data format, the bad data is reserved in the command file along with an indicator to abort the write cycle. Thus, the write cycle will be aborted when the information is accessed from the file during a memory cycle execution.

2.2.2 Tag Decode

The tag of received information is decoded to determine the format of the information. Memory accepts only two types of tags: write data and command/address. All other tags result in a response of FAULT or NO RESPONSE.

If the tag is determined to be command/address, the address and function of the information field are decoded. In this case the destination is decoded from a portion of the address received to select either the bootstrap ROM, configuration registers, or memory array for an operation. The selected destination is used along with the decoded function to determine function validity and also is input to the command file for cycle decode during a memory cycle sequence.

If the tag is determined to be write data, the condition of the EXPECT WRITE DATA flag is examined. If set, this flag indicates the write data is for memory and the information is stored in the command file. If the flag is not set, the information is ignored (Paragraph 2.2.2.1).

2.2.2.1 EXPECT WRITE DATA – Write data formats contain no addressing information. Thus, when the function of a command/address is decoded to be write masked, interlock write masked, or extended write masked, the EXPECT WRITE DATA flag is set. This indicates the next latched SBI information is expected to be write data for memory. If the EXPECT WRITE DATA flag is set and the tag of received data is not write data, memory aborts the write cycle by storing a WRITE ABORT indicator with the data in the command file. The EXPECT WRITE DATA flag is then cleared and when the data is accessed from the file for a memory cycle sequence, a write sequence fault occurs.

If the EXPECT WRITE DATA flag is not set and the tag of a latched format is decoded to be write data, a NO RESPONSE confirmation is asserted by the response logic and the information is ignored.

The EXPECT WRITE DATA flag is set for one decrement by the function decode of a write masked or interlock write masked operation because in each case one write data format is expected. Similarly, an extended write masked operation sets the flag for two decrements because two write data formats are expected. Each time a write data tag is decoded, the EXPECT WRITE DATA flag is decremented.

2.2.3 Function Decode

The function is decoded during the information decoding sequence for three reasons:

1. to determine function validity,
2. to determine if the file has enough space to accept all of the operation's formats,
3. for input to the command file for cycle decode at the start of a memory cycle sequence.

Part of the address decode selects one of three destinations (array, configuration register, or ROM) for the address and command received. Function validity is dependent on the destination selected. Table 2-1 lists the selectable functions (operations) for each destination.

Table 2-1 Function Validity

	Destination		
Memory Operation	**Bootstrap ROM**	**Configuration Registers**	**Memory Array**
Read Masked	Valid	Valid	Valid
Extended Read	Invalid	Invalid	Valid
Write Masked	Invalid	Valid (if mask = 1111)	Valid
Extended Write Masked	Invalid	Invalid	Valid
Interlock Read Masked	Valid	Valid	Valid
Interlock Write Masked	Invalid	Valid (if mask = 1111)	Valid

As mentioned previously, function decode may also set the EXPECT WRITE DATA flag (Paragraph 2.2.2.1). If the function is decoded to be write masked or interlock write masked, the EXPECT WRITE DATA flag is set for one decrement. If the function is decoded to be extended write masked, the flag is set for two decrements (two write data formats are expected).

When an interlock write masked function is decoded a flag is set to clear the interlock flip-flop. This flag is dropped and the interlock flip-flop is cleared when a write data tag is decoded (Paragraph 2.2.2).

If the function is decoded to be a read command, the ID of the received information is reserved for transmission with requested data.

2.2.4 Address Decode

During address decode a portion of the address is decoded to determine the destination of the address and command. Once selected, the validity of the address and command (function) may be checked. The decoded destination is also stored in the command file to be used for cycle decoding during a memory cycle sequence.

The destination decode may select the memory array for a data transfer. Likewise, it may select an I/O transfer with the memory's configuration registers or bootstrap ROM. If the address destination is determined to be none of the three (i.e., not for memory), a NO RESPONSE confirmation is asserted and the decoding sequence is terminated. Bit 27 of the received address is set for a reference to I/O space, and not set for a reference to main memory space (Figure 2-2).

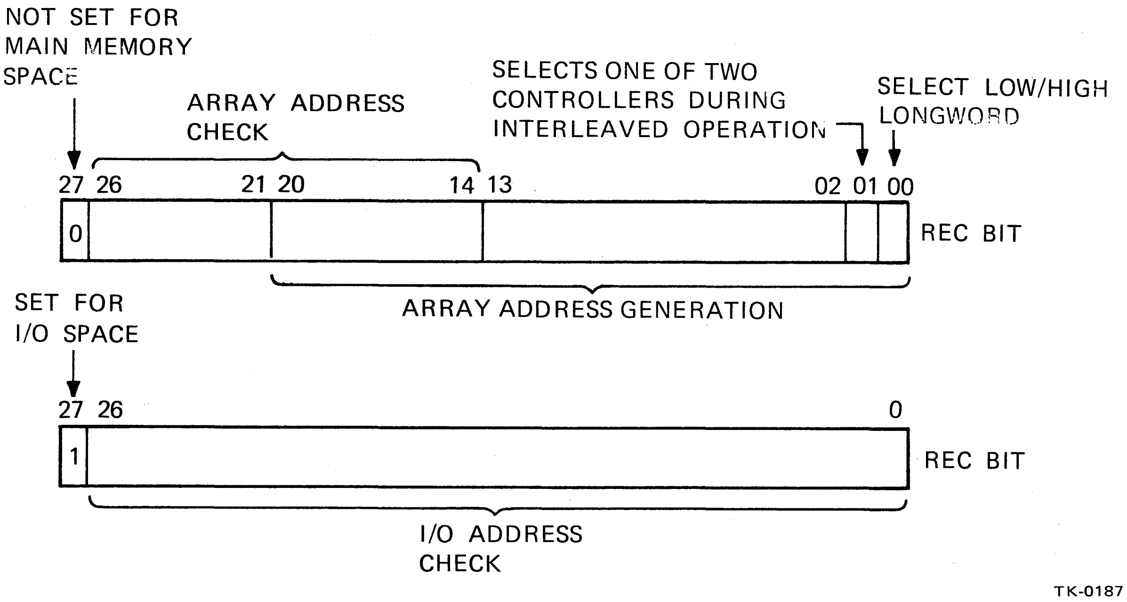


Figure 2-2 Received Address Formats

2.2.4.1 Array Address – If the reference is to main memory, part of the address is used to determine if the reference is to the controller's array, and part is corrected and transferred to the command file for use during the memory cycle sequence (Figure 2-2). The address correction logic enables the controller to accept 4K chip modules (M8211) or 16K chip modules (M8210). Figure 2-3 illustrates the array address check. Basically, if the address lies between the upper and lower boundaries of the controller's array, the address is valid.

The lower boundary check is accomplished by comparing the controller's starting address (stored in configuration register B) to the received address. Likewise, the upper boundary check is accomplished by comparing the received address to the controller's upper boundary. The upper address boundary is the sum of the controller's starting address and the array size for that controller. If the received address is equal to or greater than the controller's starting address and less than the controller's upper boundary address, the reference is to that controller's array.

The array size for a controller, however, is dependent on the size of the chips being used and whether or not interleaving is used. Figure 2-4 shows the array size correction for these conditions. The array size before correction [MEM SZ (3:0)] is encoded from the number of array cards plugged into the controller's backplane. The first correction is determined by chip size and the second is determined by the interleave condition. Thus, the actual array size of the controller is given as COR ARY SZ (6:0).

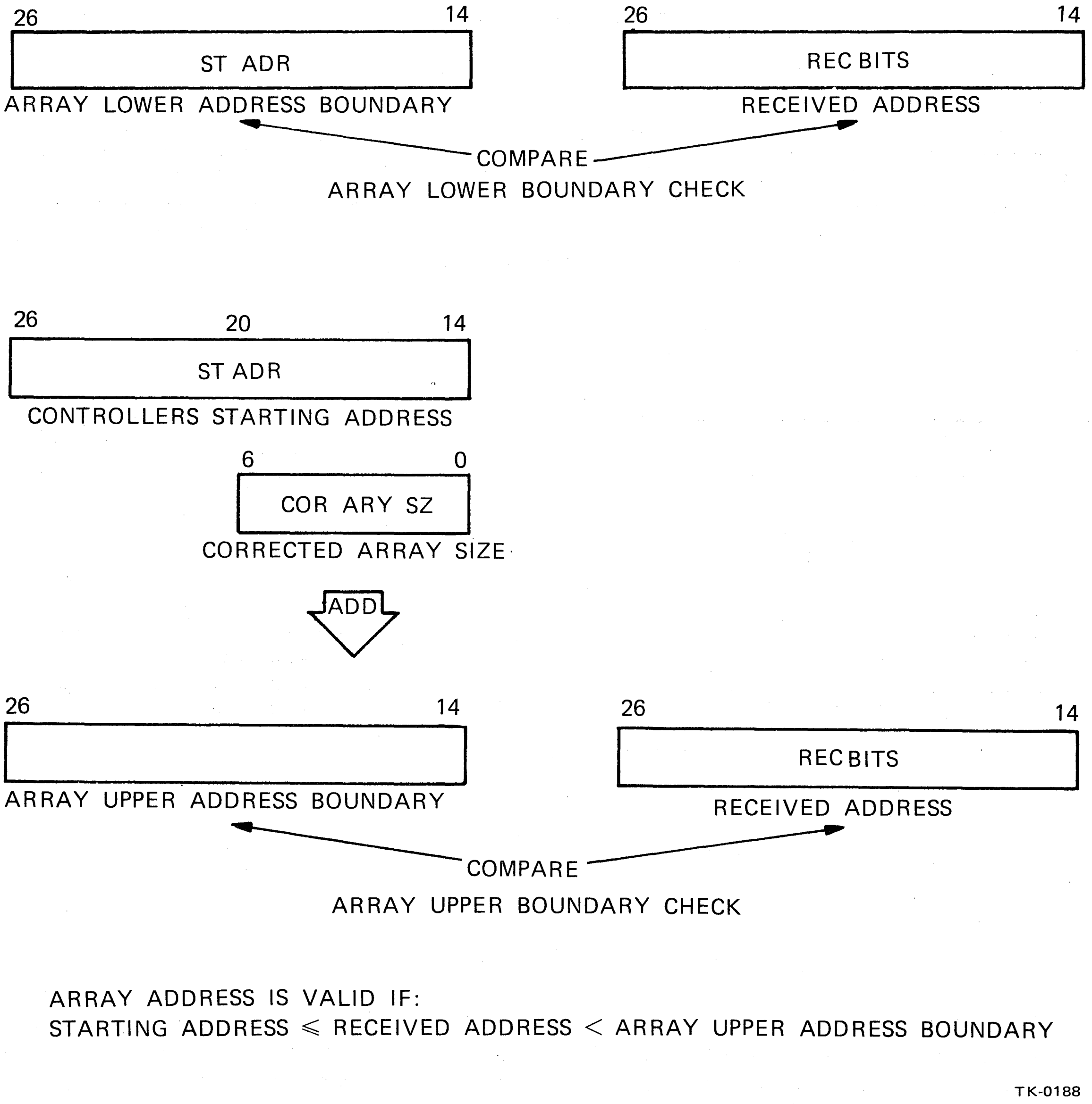

26
14
ST ADR
ARRAY LOWER ADDRESS BOUNDARY
26
14
REC BITS
RECEIVED ADDRESS
COMPARE
ARRAY LOWER BOUNDARY CHECK
26
20
14
ST ADR
CONTROLLERS STARTING ADDRESS
6
0
COR ARY SZ
CORRECTED ARRAY SIZE
ADD
26
14
ARRAY UPPER ADDRESS BOUNDARY
26
14
REC BITS
RECEIVED ADDRESS
COMPARE
ARRAY UPPER BOUNDARY CHECK
ARRAY ADDRESS IS VALID IF:
STARTING ADDRESS ⩽ RECEIVED ADDRESS < ARRAY UPPER ADDRESS BOUNDARY
TK-0188

Figure 2-3 Controller's Array Address Check

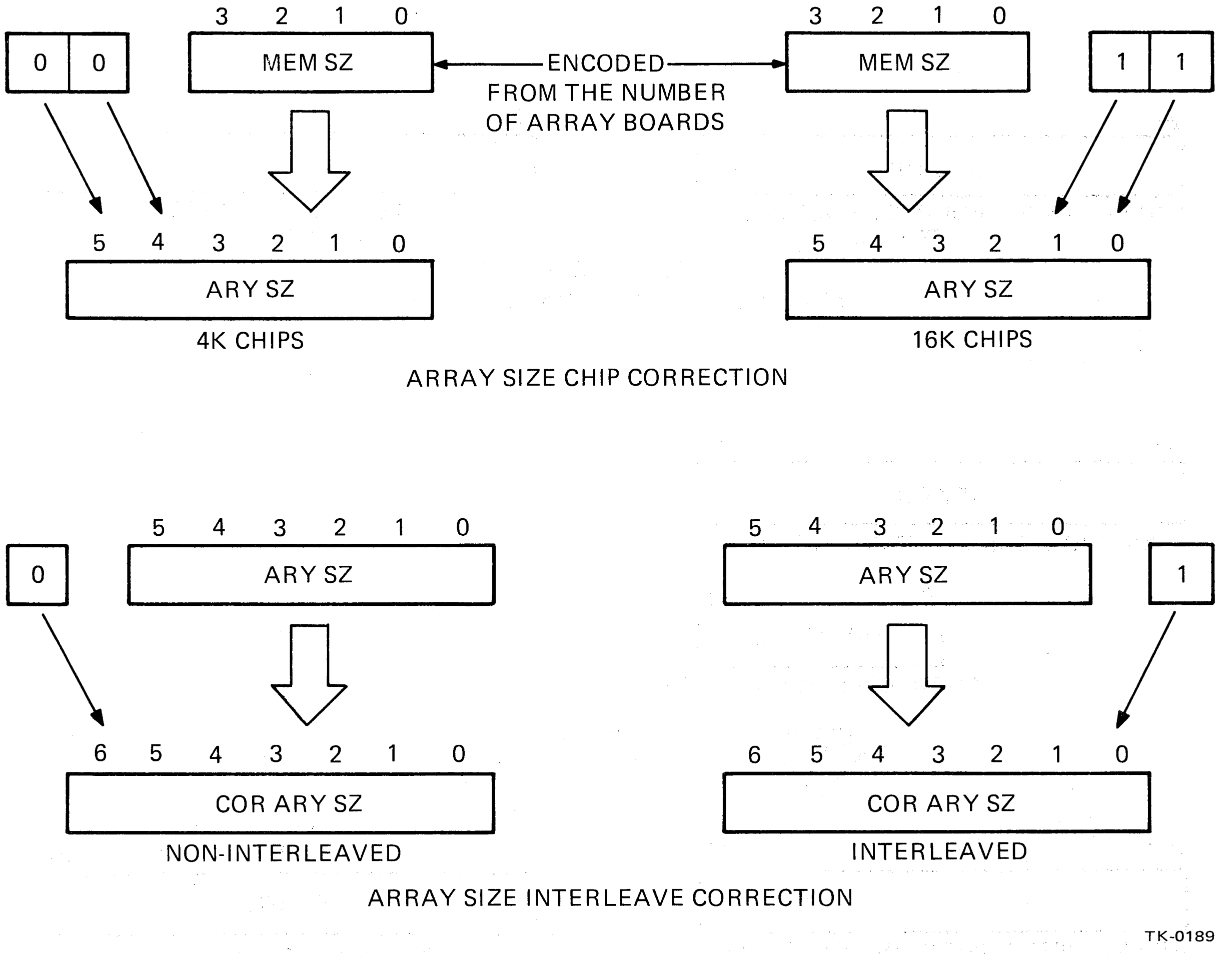
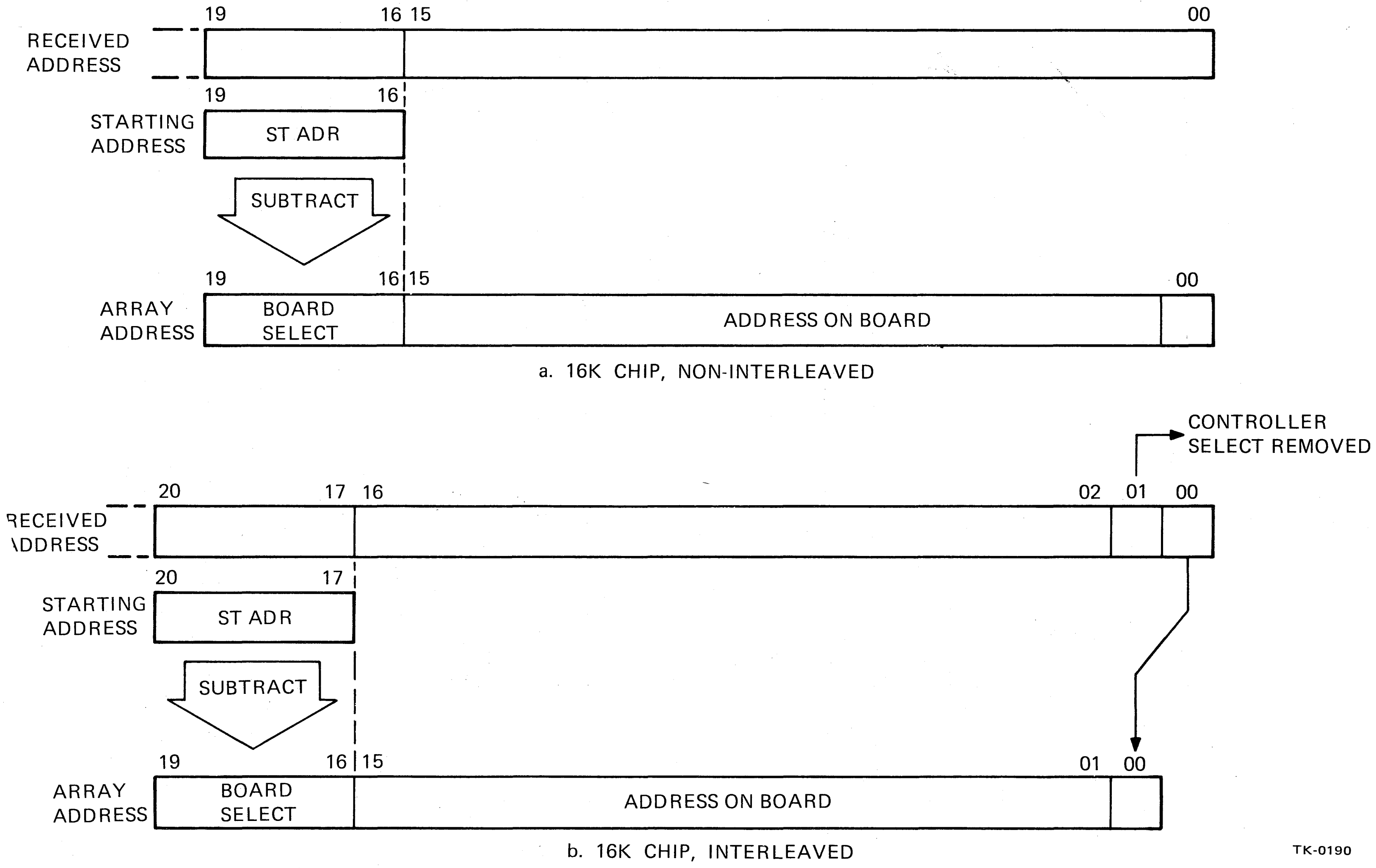


Figure 2-4 Array Size Correction

2.2.4.2 Array Address Generation – As shown in Figure 2-4, REC bit (20:00) of the latched address is used to generate a main memory reference address. The hardware design provides for this address translation so that memory can accommodate varied chip size and interleave conditions. These conditions dictate which of the 21 REC bits are used for the array address generation.

Sections A and B of Figure 2-5 illustrate the array address generation for memory with 16K chips in the interleaved and non-interleaved modes. In both modes a 20-bit address is required. For the interleaved mode, however, REC bit 01 is used as the deciding bit for controller selection (selection of odd or even controller during interleave operation). Thus, in this case, REC bit 20 is also used.

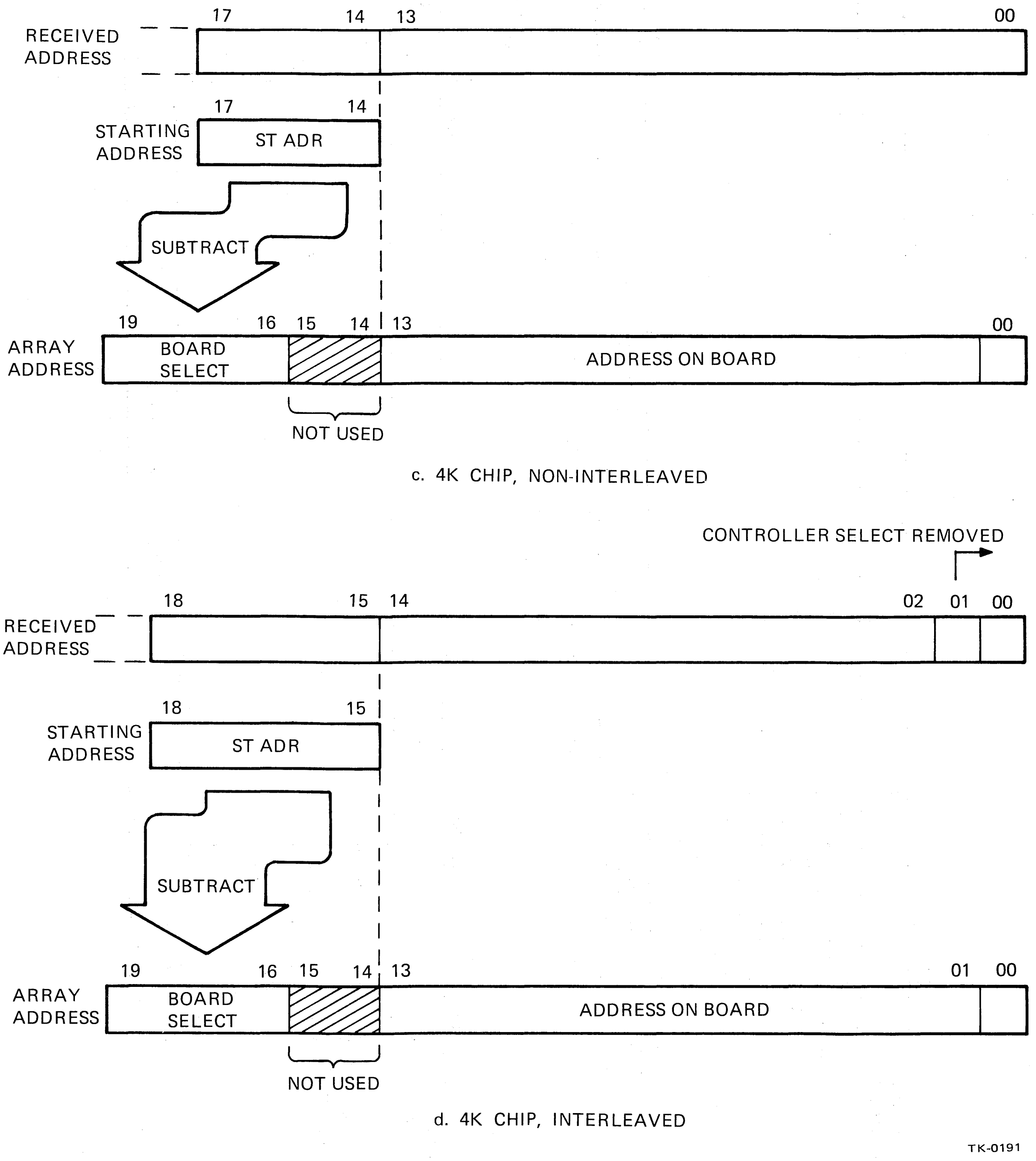
Similarly, sections B and C of Figure 2-5 illustrate the array address generation for memory with 4K chips in the interleaved and non-interleaved modes. Only 18 bits of the received address [REC bit (17:00)] are used for the non-interleaved mode. For the interleaved mode, however, 19 bits [REC bit (18:00)] are used. In this mode REC bit 01 is used for controller select. Note in either mode, for memory with 4K chips, REC bits 14 and 15 are ignored.



a. 16K CHIP, NON-INTERLEAVED

b. 16K CHIP, INTERLEAVED

Figure 2-5 Array Address Generation
(Sheet 1 of 2)



c. 4K CHIP, NON-INTERLEAVED

d. 4K CHIP, INTERLEAVED

Figure 2-5 Array Address Generation (Sheet 2 of 2)

2.2.4.3 I/O Address – If the reference is to I/O space, an I/O address check is performed. The I/O address check is performed on a received address just as an array address check to determine address validity. A memory I/O address will prove invalid as an array address, but valid as an I/O address. For a valid I/O address, the address translation logic is inhibited and the address is transferred unaffected over the REC lines to the file.

The I/O addresses for memory are shown in Figure 2-6. The hex representation is the byte address a seen from the console. Only 28 of the 30 bits, however, are transmitted over the SBI (longword addresses). Bit 27 is asserted in each case to designate device control space. Note the addresses are assignable in so far as the TR level is assignable (i.e., address bits 14–11 define the TR level of the memory controller.)

2.3 COMMAND FILE

The memory controller contains a file to store incoming command/address or write data information. The file is capable of storing up to four successive information formats until the memory subsystem is available to utilize the information. The file's performance is similar to a first in, first out (FIFO) silo. Associated logic provides buffering and queuing.

2.3.1 File Control

The file control logic monitors the amount of space available in the command file and determines if there is sufficient space to store a memory request. This is implemented with a write-file pointer and a read-file pointer in conjunction with a difference decoder that determines the amount of space in the file. A comparator in the logic is used to compare this amount to the encoded amount of space required to store the entire command string (from function decode). If enough space exists, all formats for the function are stored as they are received, providing they are valid.

Although they are input to the file, invalid command/address formats (bad parity, invalid function, or invalid address) are not stored in the file. This is accomplished by not advancing the write pointer. Thus the next format input to the file is written over the invalid command/address, thereby eliminating it.

2.3.2 File Operation

At T1 of every SBI cycle, if the latched data is memory destined and valid, the write pointer of the command file is advanced leaving it in the file for a memory cycle sequence. If the information is not memory destined, or is in error, the write pointer remains unaltered and the file location is rewritten with new data at the next T1.

A memory cycle sequence is also initiated at T1 if there is data in the file and the memory is not busy executing another memory cycle. The first transfer from the file is always command/address information that advances the read pointer. If the decoded function (also taken from the file) indicates a write operation, write data formats are removed from the file at subsequent T1s advancing the read pointer appropriately. When all formats for the operation have been removed from the file, a memory bus signal is asserted and a memory cycle sequence is executed. The memory busy signal prevents initiation of another memory cycle sequence (removal of more information from the file) u current one has been completed.

CONFIGURATION REGISTER A

27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00		
1	0	0	0	0	0	0	0	0	0	0	0	0	—	—	—	—	0	0	0	0	0	0	0	0	0	0	0	0	0
2		0				0				—				—				0				0				0			

FOR BYTE ADDRESS

CONFIGURATION REGISTER B

27																											0		
1	0	0	0	0	0	0	0	0	0	0	0	0	—	—	—	—	0	0	0	0	0	0	0	0	0	0	1	0	0
2		0				0				—				—				0				0				4			

CONFIGURATION REGISTER C

27																											0		
1	0	0	0	0	0	0	0	0	0	0	0	0	—	—	—	—	0	0	0	0	0	0	0	0	0	1	0	0	0
2		0				0				—				—				0				0				8			

BOOTSTRAP ROM

27																	10										0		
1	0	0	0	0	0	0	0	0	0	0	0	0	—	—	—	—	1	X	X	X	X	X	X	X	X	X	X	0	0
2		0				0				—				—				X				X				X			

NOTES:

1. BIT 27 IS SET TO DESIGNATE DEVICE CONTROL ADDRESS SPACE.
2. THE ASSIGNED TR LEVEL OF THE CONTROLLER DEFINES I/O ADDRESS BITS 14-11.
3. X = ALLOWABLE ADDRESS SPACE FOR ROM BOOTSTRAP.

TK-0178

Figure 2-6 I/O Register Addresses

2.4 MEMORY CYCLE SEQUENCE; GENERAL

The memory cycle sequence is the execution of an internal memory operation in which a valid memory request is honored. As previously mentioned and in Figure 2-7, if the controller is not busy at T1 and the command file contains information, a memory cycle sequence is initiated. The signal MEMGO is generated when the last information format of the command string has been removed from the file. A memory busy signal is also generated at this time to eliminate the initiation of another memory cycle sequence before the current one is complete (Paragraph 2.3.2). The asssertion of MEMGO occurs between T1 and T2 of the same SBI cycle for read masked, interlock read masked, and extended read commands (no write data formats must be taken from the file). Write masked and interlock write masked commands delay MEMGO for one SBI cycle in order to fetch the write data format from the file. Likewise, an extended write masked command delays MEMGO for two SBI cycles.

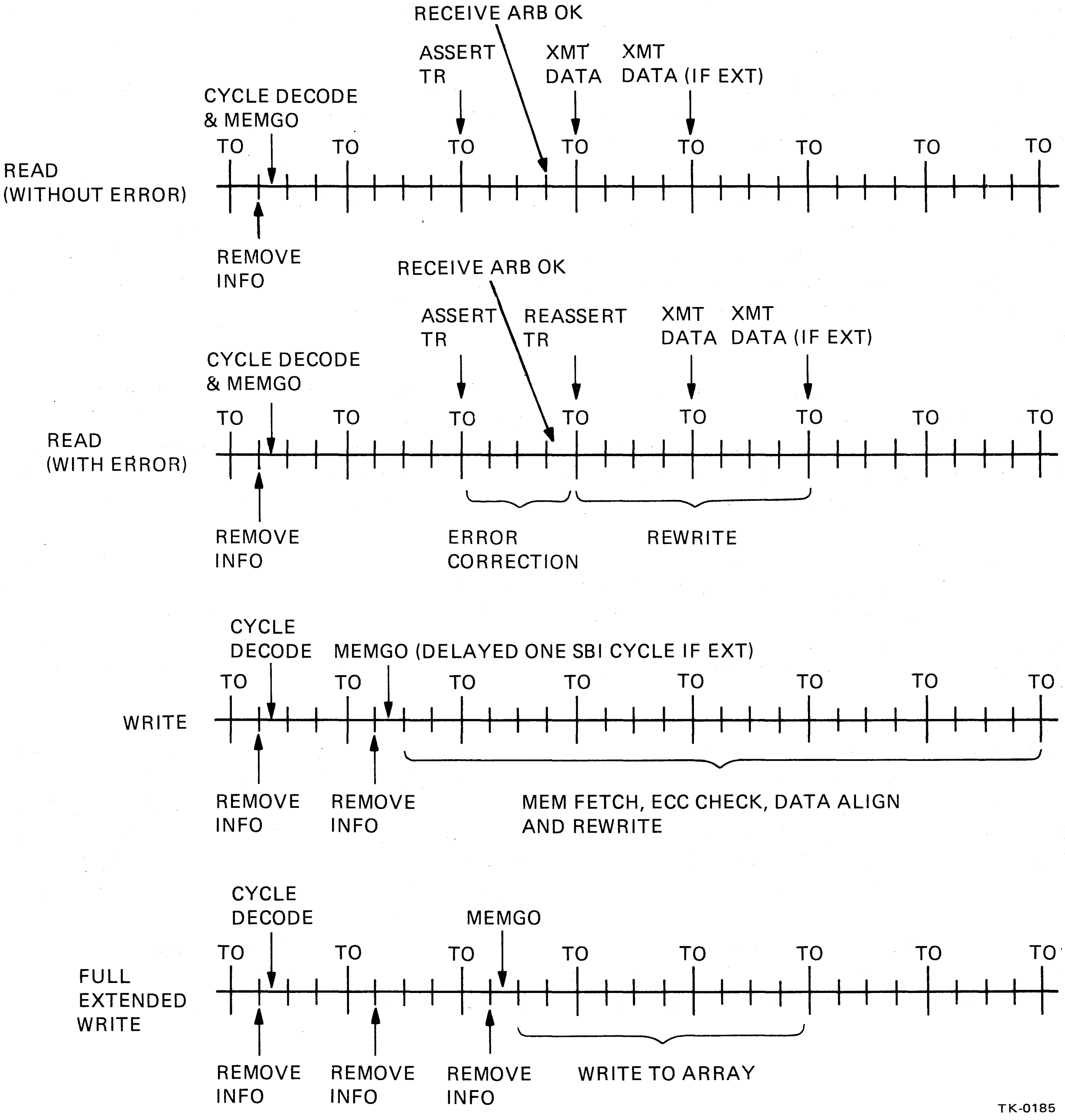


Figure 2-7 Memory Cycle Sequence

The type of cycle is determined when the decoded function and decoded destination are input to the cycle decode logic from the file. The function and destination are taken from the file whenever a command/address is removed. The cycle decode logic generates control signals to properly channel address and data during the memory cycle sequence. The address is transferred to the address register via the file information bus as soon as the command/address is removed from the file.

2.4.1 Read Cycles

As all memory cycles, a memory read cycle begins with MEMGO. For all read commands, memory data must be transferred onto the SBI. Therefore at T0 of the second SBI cycle after cycle decode, memory asserts its TR line to arbitrate for the bus in anticipation that the read data will be ready at T0 of the third SBI cycle. If, however, during error checking a correctable error is detected, the data is delayed one cycle for correction and memory must re-arbitrate for the bus. For all extended read cycles, HOLD is also asserted at T0 of the third SBI cycle to reserve the information lines for two longword transfers.

If the read is to a memory I/O register, the addressed register enables the requested data to the I/O multiplexer and onto the file bus. At the proper time the data can then be transmitted onto the SBI in a read data format through the SBI transceivers.

If the read is to a memory location in the array, the address from the address register is transferred to the array. The entire quadword containing the requested data is then placed on the MOS data bus (along with its check code) and latched for error checking and correction. A tag is generated as a result of the error check and transferred onto the file bus to be transmitted with the requested data. The tag indicates the integrity of the read data to be transmitted (read data, corrected read data, or read data substitute).

If a correctable error is not detected, the data on the MOS data bus is then transferred to the file bus as selected by the mask decode. For extended read operations the lower longword is transferred first. Once on the file bus, the data is available for transmission onto the SBI. Transmission occurs when memory gains control of the SBI information lines. At this time the memory busy signal is dropped and an ID check for multiple transmitters on the SBI is executed for each read data transmitted.

In the case of a correctable error, the error is corrected in the ECC logic. The corrected data is then placed onto the MOS data bus for transfer to the file bus for transmission. Information transmission is executed just as any read data transmission. The corrected data and new check code, however, are also transferred to the array via the MOS data bus for a rewrite. The memory busy signal is cleared only after all requested data has been transmitted and the corrected data has been rewritten into the array.

2.4.2 Write Cycles

As with all memory read cycles, a memory write cycle begins with MEMGO. The write cycle, however, may be aborted during the memory cycle sequence if a WRITE ABORT indicator had been set and stored in the file with the write data during the information decoding sequence. The WRITE ABORT indicator is set during the information decoding sequence if the EXPECT WRITE DATA flag is set and the information latched is not write data, or a parity error occurs in the received write data. In this case an ERROR confirmation would have been transmitted at the appropriate time (Paragraph 2.2).

The only I/O write cycle is a write to the configuration registers. The addressed register latches the data from the file bus as it is removed from the file. An ACKNOWLEDGE confirmation is transmitted and the memory busy signal is dropped at the appropriate time.

In the case of a write to the array, data is not required to be fetched from the array for error correction if and only if the entire quadword is to be written (i.e., the operation is an extended write masked and all masks = 1111). In this case MEMGO is set and each data longword is transferred from the file bus onto the MOS data bus as they are removed from the file. The entire quadword is then latched by the ECC logic for check code generation. When the check code is generated, it is placed onto the MOS data bus and transferred with its corresponding data to the array. Otherwise, for both extended and non-extended writes to the array, MEMGO is set and a read cycle is started. The read cycle is executed to perform error correction on the portion of the selected quadword that will not be replaced (Paragraph 1.6).

For an uncorrectable error, the bad data and check code are merely rewritten into the array, as is. Although the new information is not written, the error will be detected when the data is read from the array during a read cycle. In this case the memory busy flip-flop is cleared as soon as the rewrite is complete.

For a correctable error, the data is corrected and rewritten as if no error had occurred. That is, the new data is aligned with the data read from the array and then is rewritten into the array. The memory busy flip-flop is cleared when the write is complete.

2.5 INFORMATION DECODING SUMMARY

Figures 2-8 through 2-12 summarize the information decoding sequence discussed in Paragraph 2.2. Point A of Figure 2-8 indicates the entry point into the sequence. Point B indicates the sequence termination. Note the sequences for the decoding of command/address and write data formats are provided in the figures listed in Figure 2-8.

As shown in Figure 2-8, at point B the response and interlock bits are enabled for transmission and the information decoding sequence is re-initiated. In addition, if the command file contains information and memory is not busy, a memory cycle sequence is also started (Paragraph 2.6).

As mentioned previously, part of the command/address decode selects one of three destinations (array, CNFG register, or ROM) for the address and command received. This destination decode is shown in Figure 2-9. Each destination selects a unique branch for function decode in the information decoding sequence. Note that if the address destination is determined to be none of the three (i.e., not for memory), a no-response confirmation is asserted for transmission and the decoding sequence is terminated.

2.6 MEMORY CYCLE SUMMARY

As shown previously in Figure 2-8, an internal memory cycle may begin at T1 only if the command file contains information and memory is not busy. Figures 2-13 through 2-16 summarize the memory cycle sequence described in Paragraph 2.4.

When a command/address format is removed from the file at T1 to begin a memory cycle sequence, the read file counter is advanced. The removal of a command/address also triggers cycle decoding. The cycle decode generates control signals that determine the correct memory cycle branch. Each branch is outlined in the figures listed in Figure 2-13. Note all branches converge at point E.

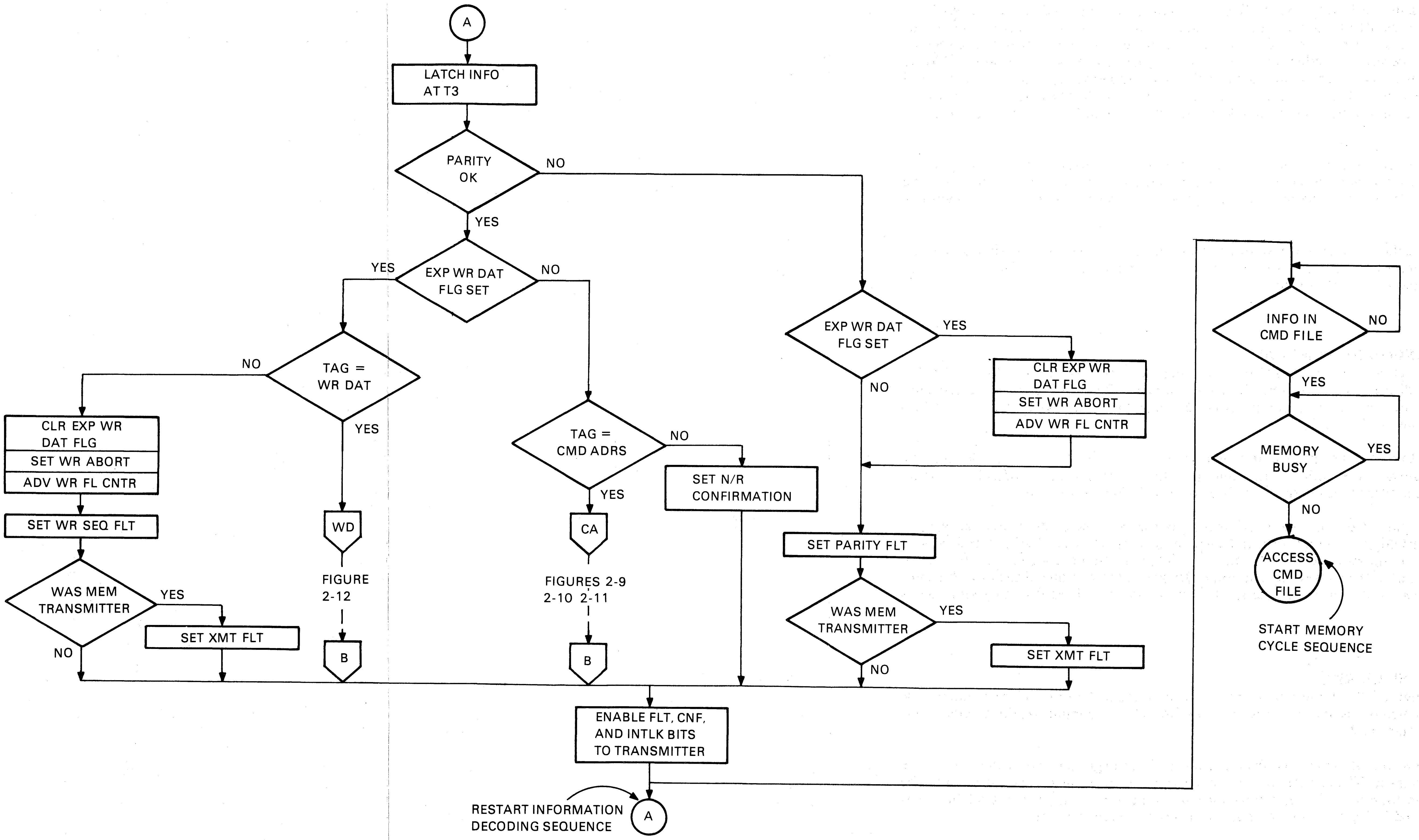


Figure 2-8 Information Decoding Sequence

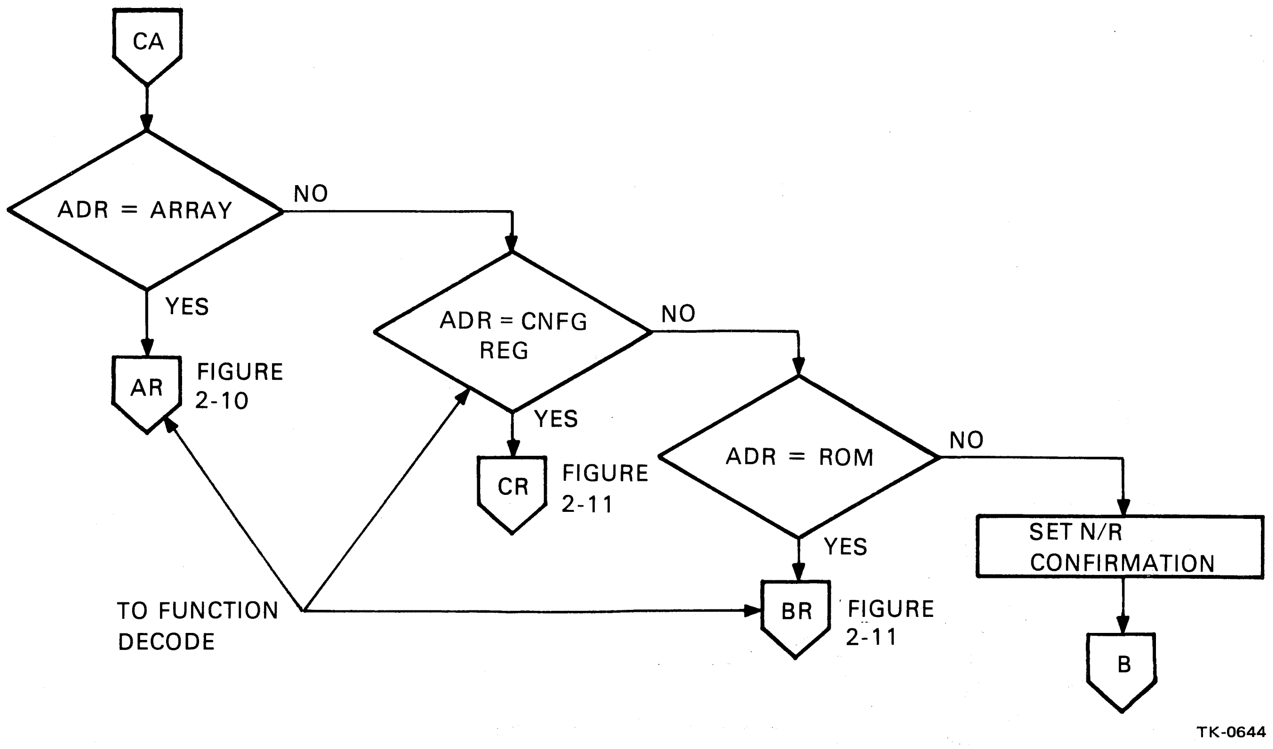


Figure 2-9 Command/Address Destination Decode

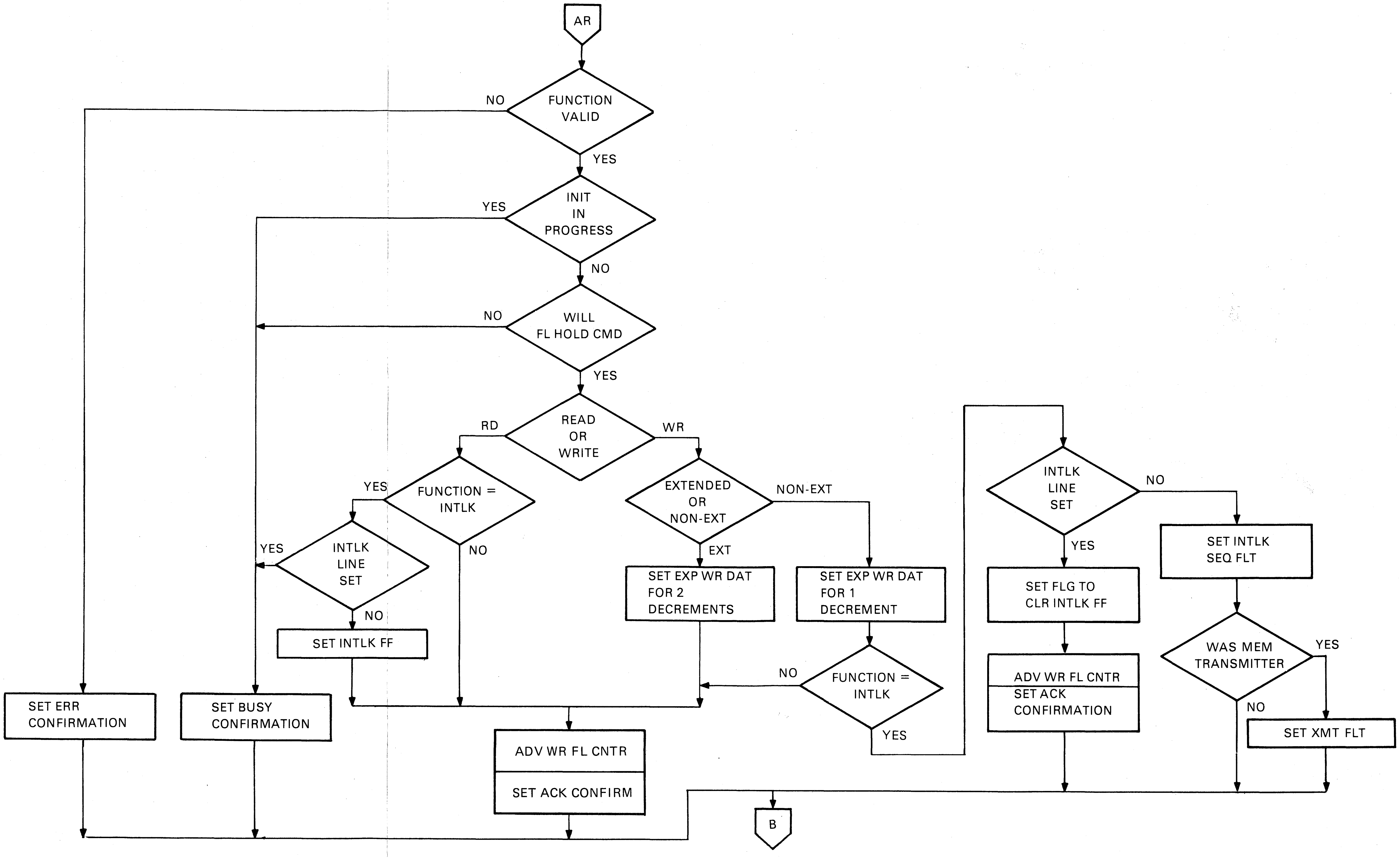


Figure 2-10 Array Function Decode

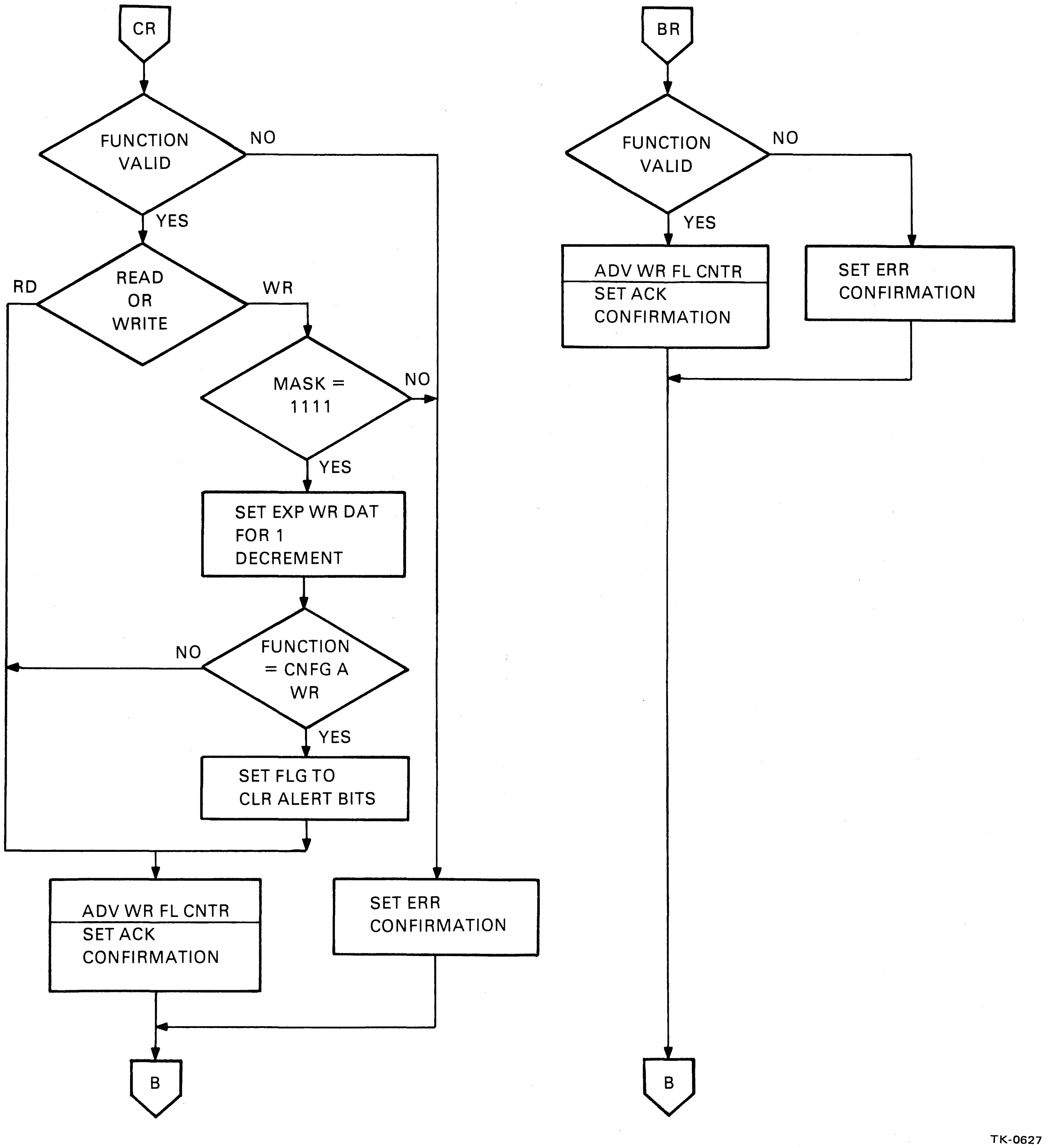
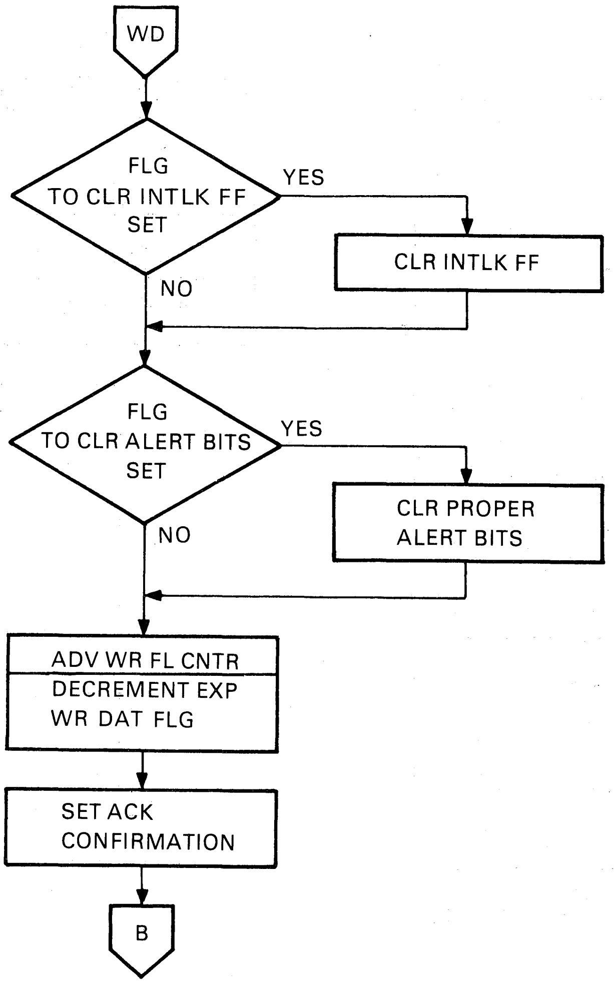


Figure 2-11 Configuration Registers and ROM Function Decode



TK-0628

Figure 2-12 Write Data Information Decode Sequence

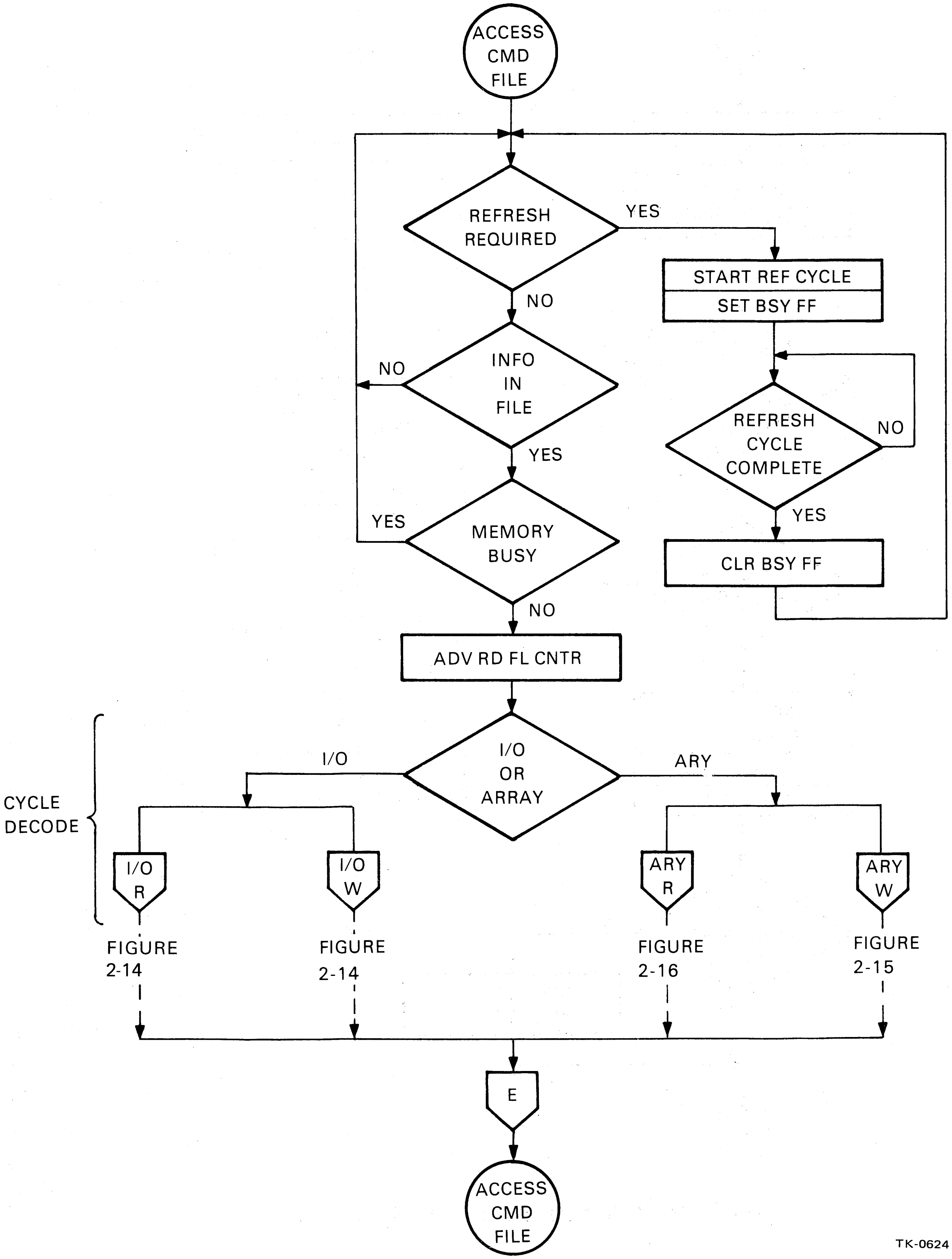


Figure 2-13 Memory Cycle Sequence

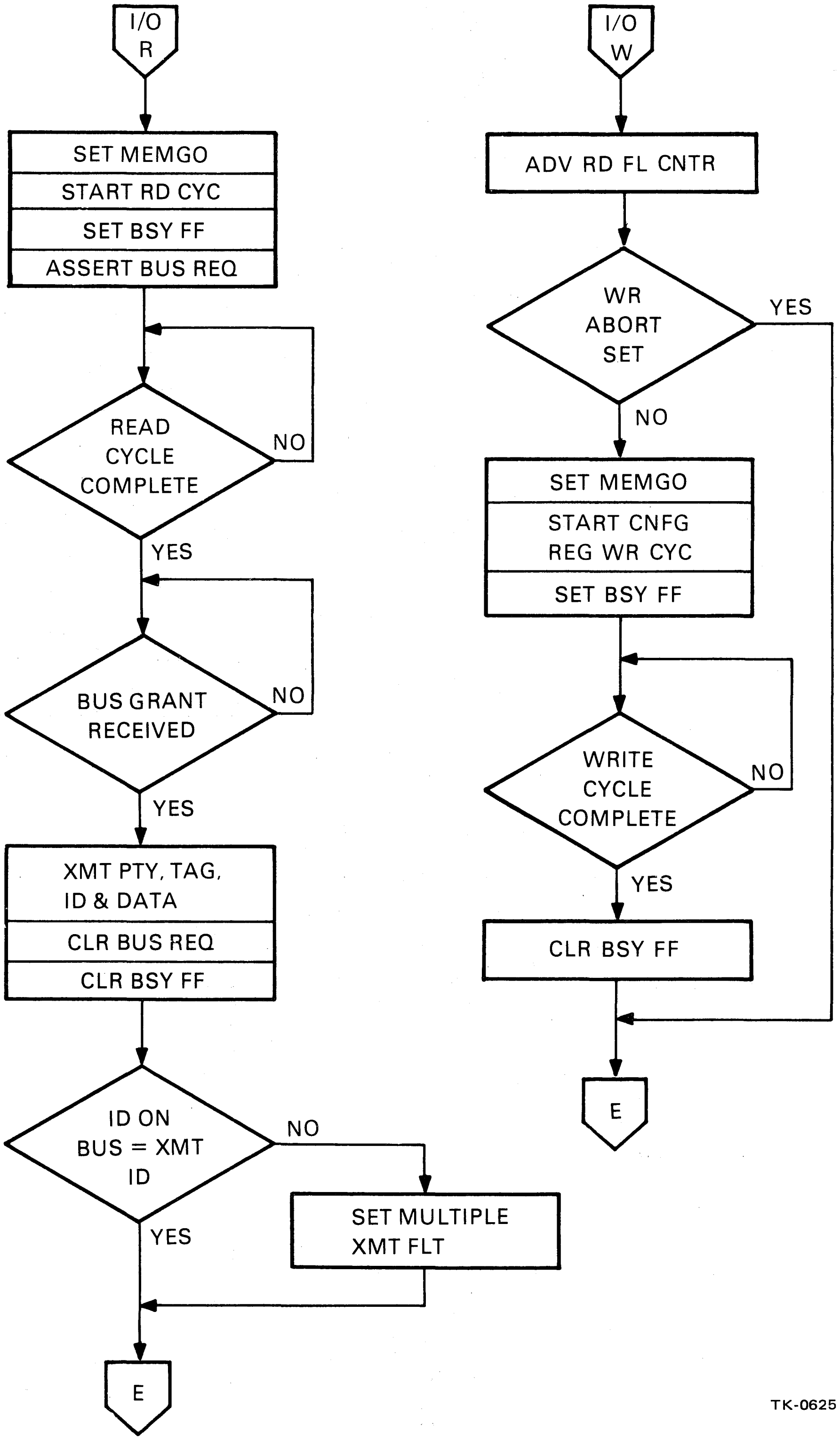
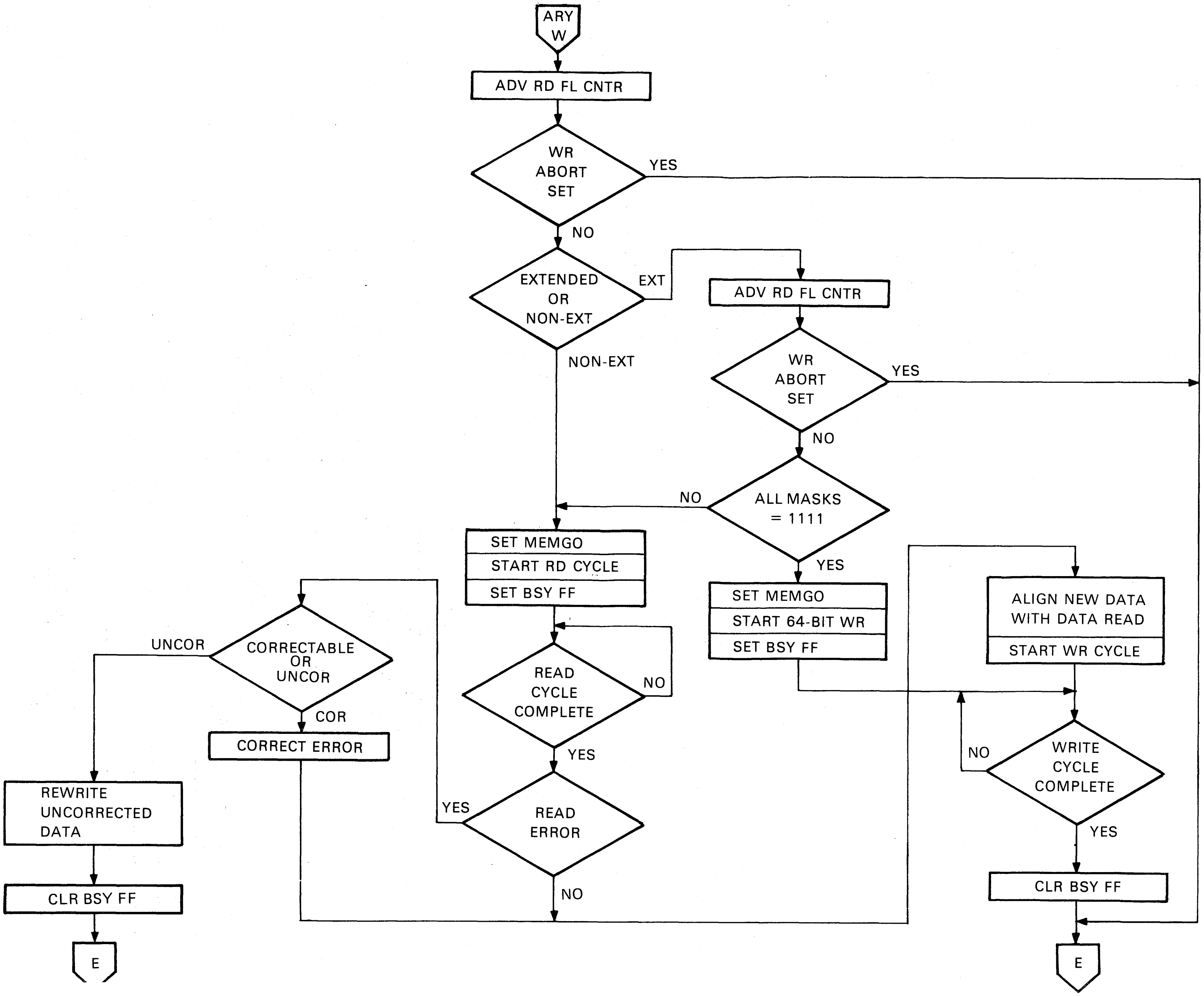


Figure 2-14 I/O Read and Write Cycle Sequences



TK-0620

Figure 2-15 Array Write Cycle Sequence

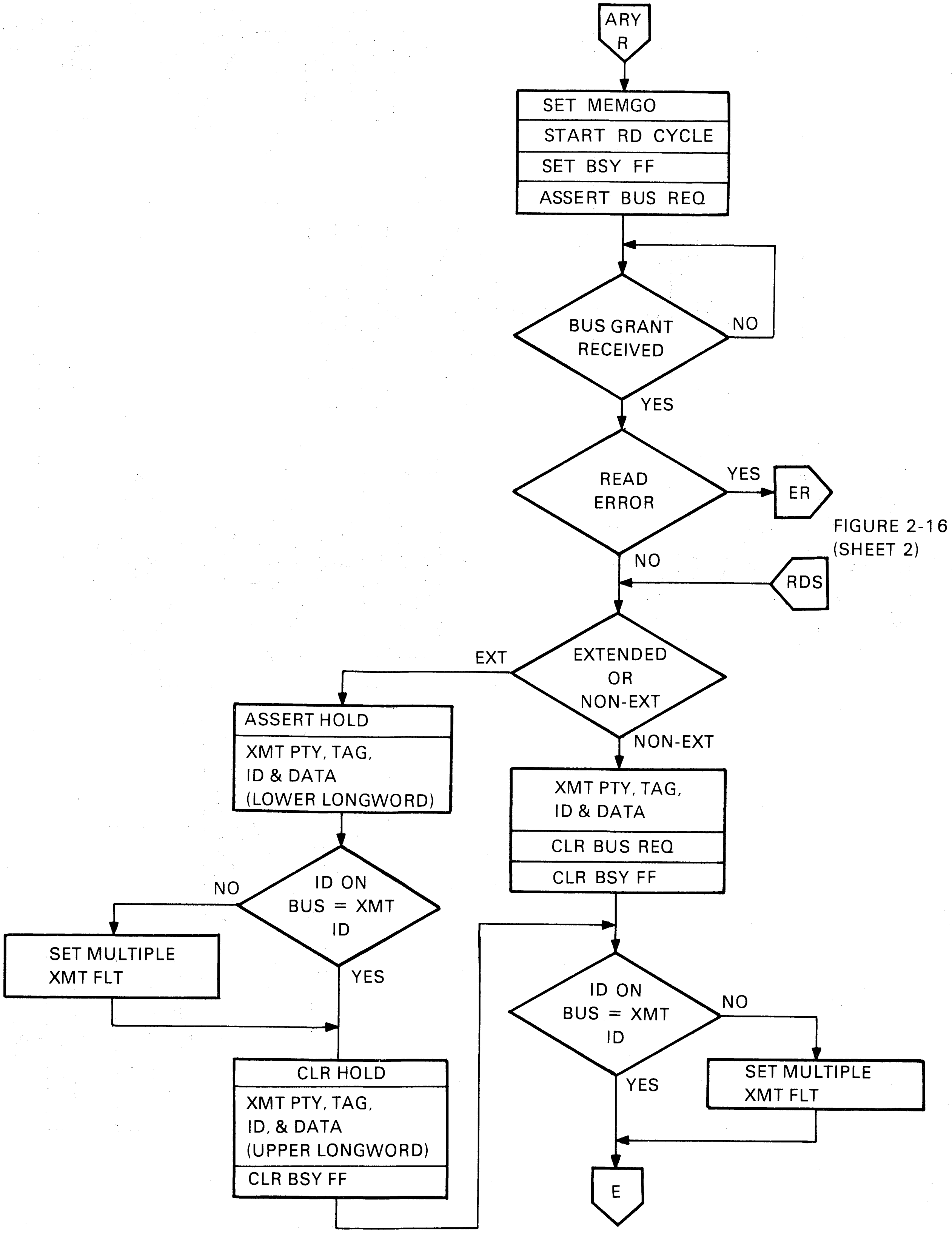


Figure 2-16 Array Read Cycle Sequence (Sheet 1 of 2)

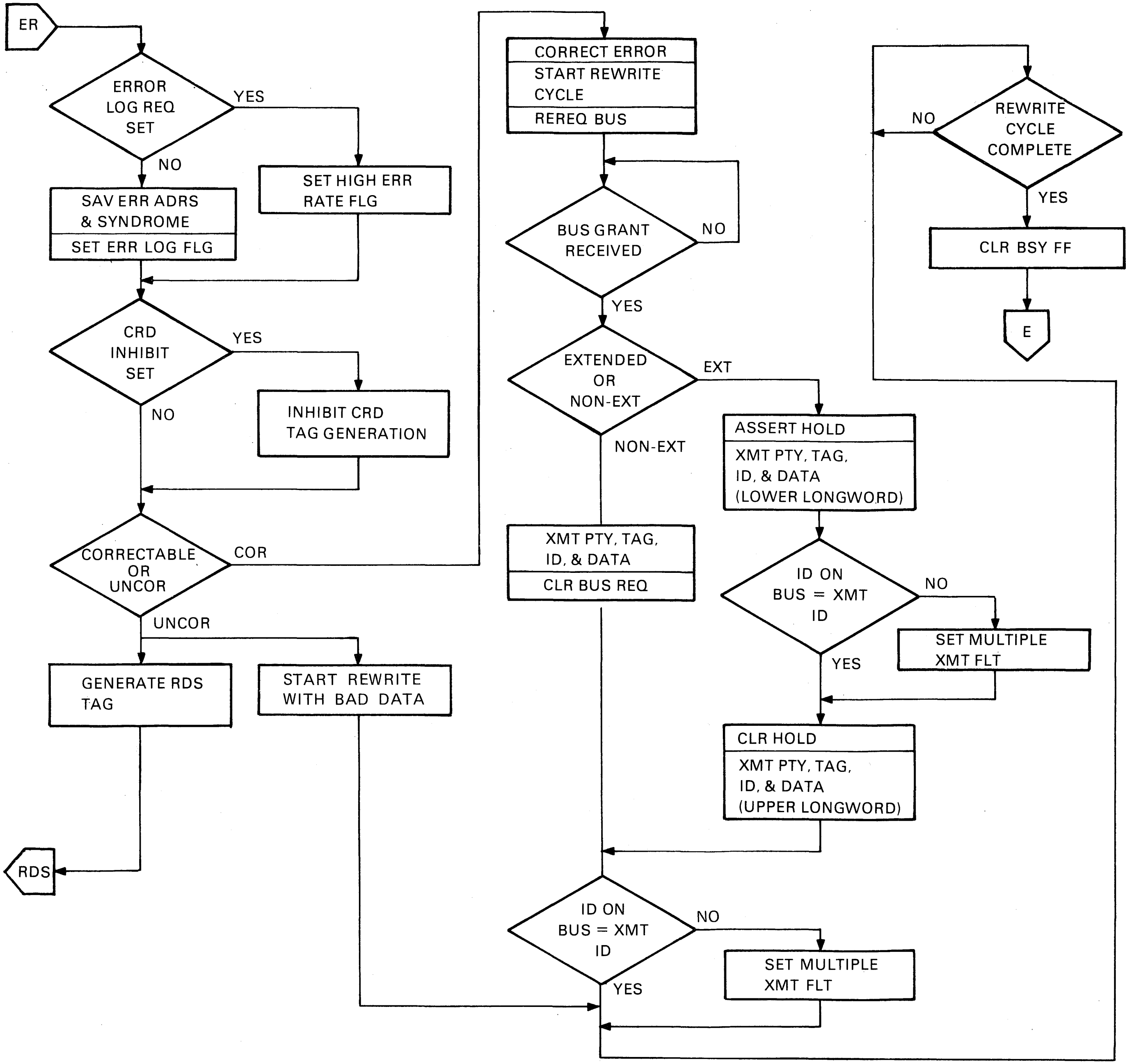


Figure 2-16 Array Read Cycle Sequence (Sheet 2 of 2)

CHAPTER 3
DETAILED DESCRIPTION

3.1 INTRODUCTION

The following paragraphs describe the logic of the MS780 in detail. A simplified block diagram of the logic is provided in Figure 3-1. The major sections of discussion are address logic, command file logic, data path logic, I/O data logic, and the array board logic. In addition to these sections, various other logic circuits are discussed. Summary paragraphs explaining the signal-to-signal timing relationships are also provided.

3.2 ADDRESS LOGIC

Once placed on the file information bus, the address is available for a memory cycle sequence. MCNB ADR 0 BUS H is generated when a command/address format is removed from the command file and placed on the file bus. This signal causes the address register to latch the address from the bus (Figure 3-2). Bit 27, which is set to indicate I/O, is also latched from the bus and sent to cycle decode logic as MCND CMD IO H.

A portion of the latched address (MCNE MEM ADR (15:00) is selected through the address multiplexer and sent to the array as MCNE ARY ADR (13:01) as long as a refresh or initialization sequence is not required.

For a refresh cycle or during initialization the address multiplexer is selected to provide MCNF REF ADR (14:01) as a board address. These signals are generated by a refresh address counter that is incremented by a dedicated refresh clock.

A portion of the output of the address register [MCNE MEM ADR (9:0)] is also sent to the ROM bootstrap. These bits select a ROM location when the ROM is enabled (MCNA ROM EN L is generated).

The array board organization and logic is described in Paragraph 3.14.1.

3.3 COMMAND FILE LOGIC

As mentioned previously, the command file is a storage buffer for incoming SBI commands and data. Figure 3-3 is a simplified diagram of the logic associated with the command file. The following paragraphs describe this logic, in addition to the file control logic.

As seen in Figure 3-3, the command file logic links the SBI with the file information bus. At T2 of every SBI cycle, SBI transceivers latch information from the SBI and present it to the command file. This SBI information is likewise presented to decode logic to determine its validity. The result of the decode enables the write-file control logic appropriately. The file control logic is explained in Paragraph 3.3.1.

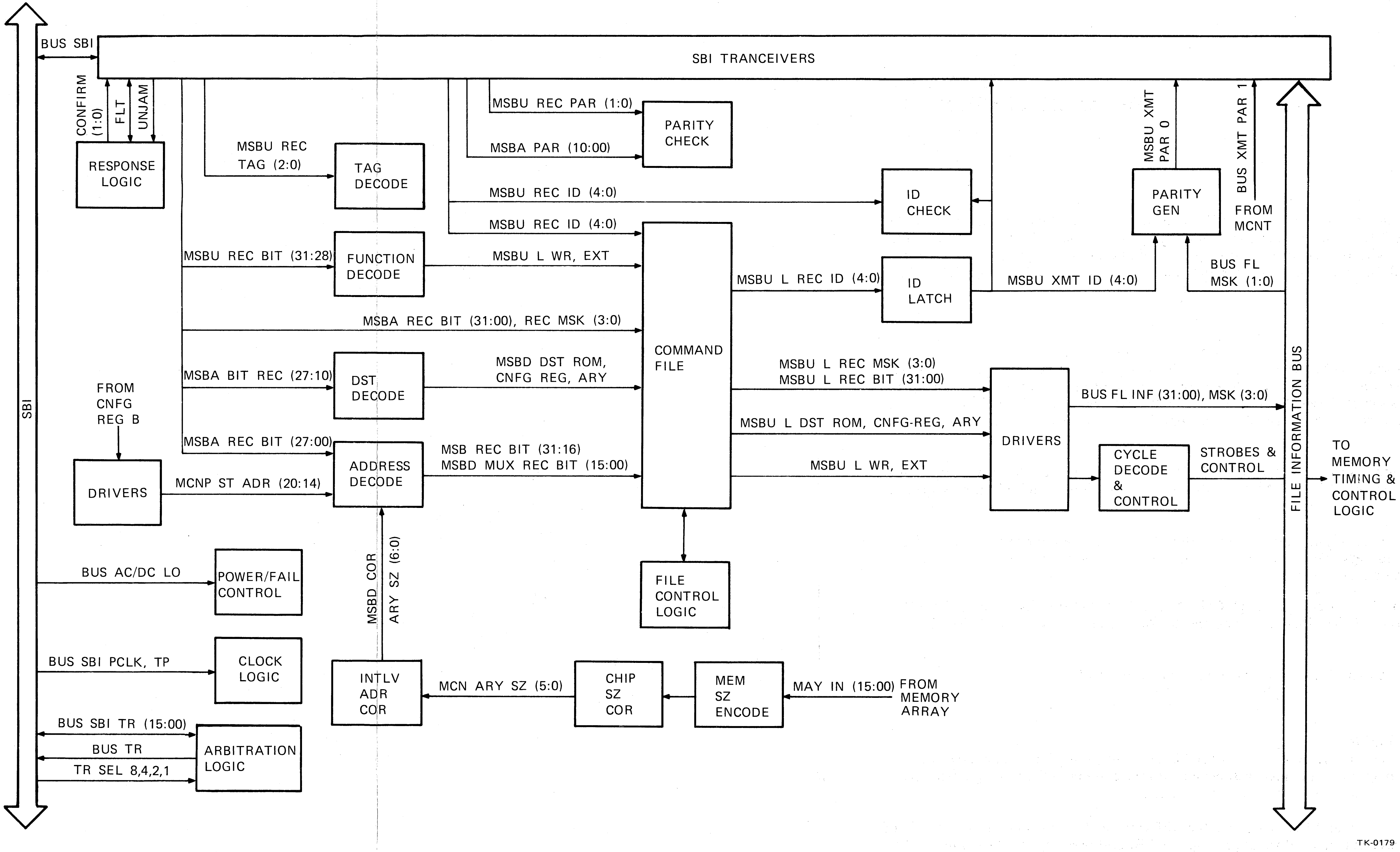


Figure 3-1 Simplified Block Diagram of the MS780 (Sheet 1 of 2)

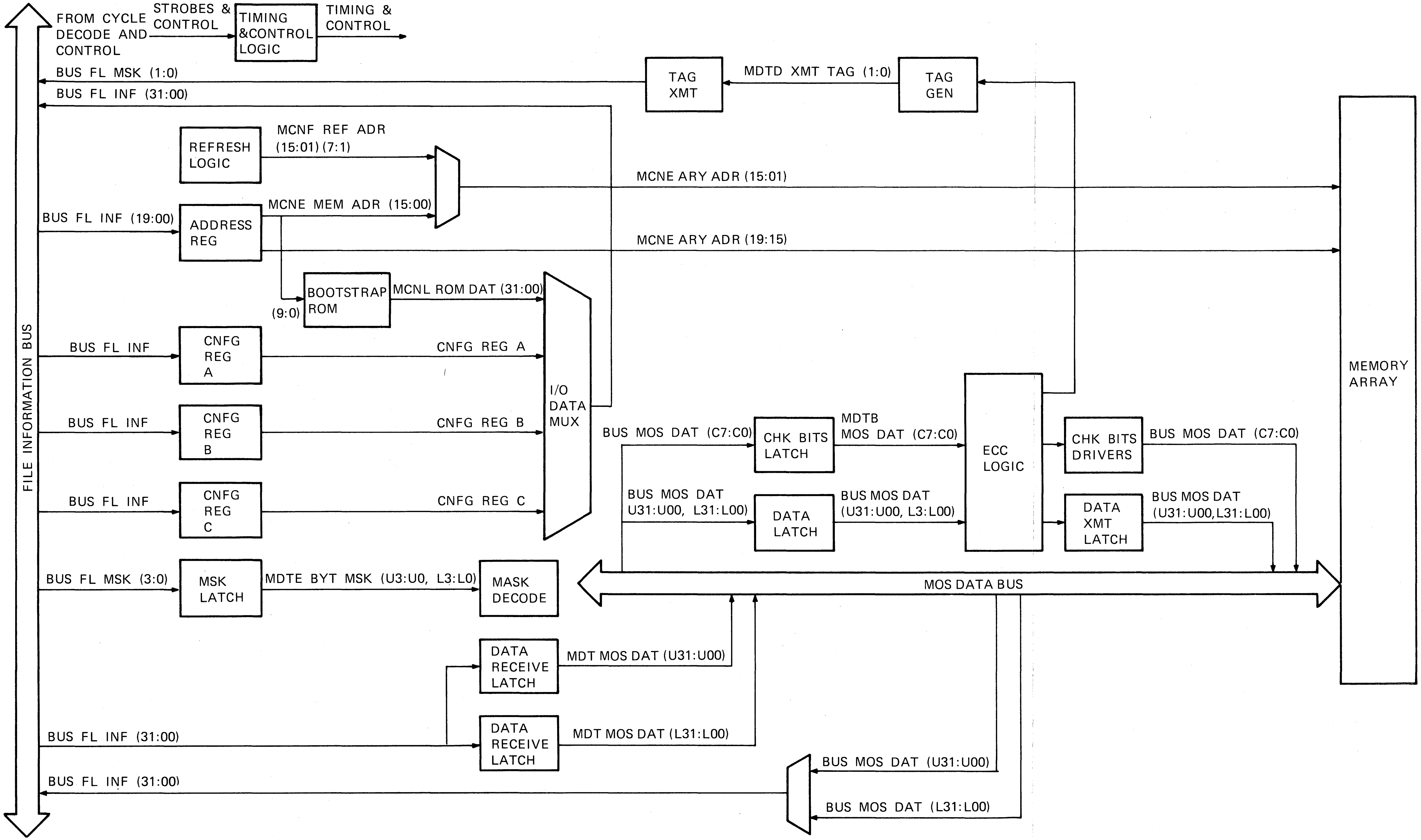


Figure 3-1 Simplified Block Diagram of the MS780 (Sheet 2 of 2)

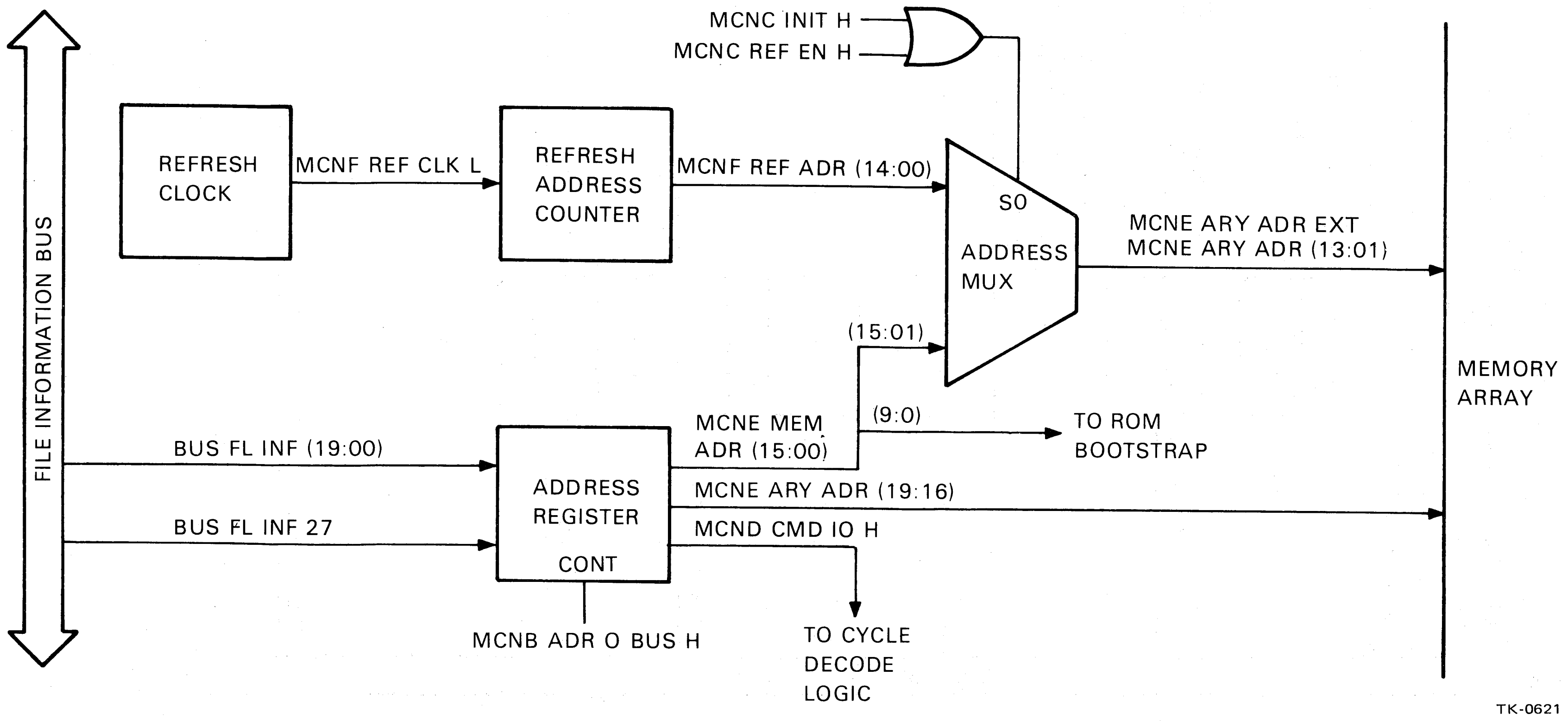


Figure 3-2 Address Logic

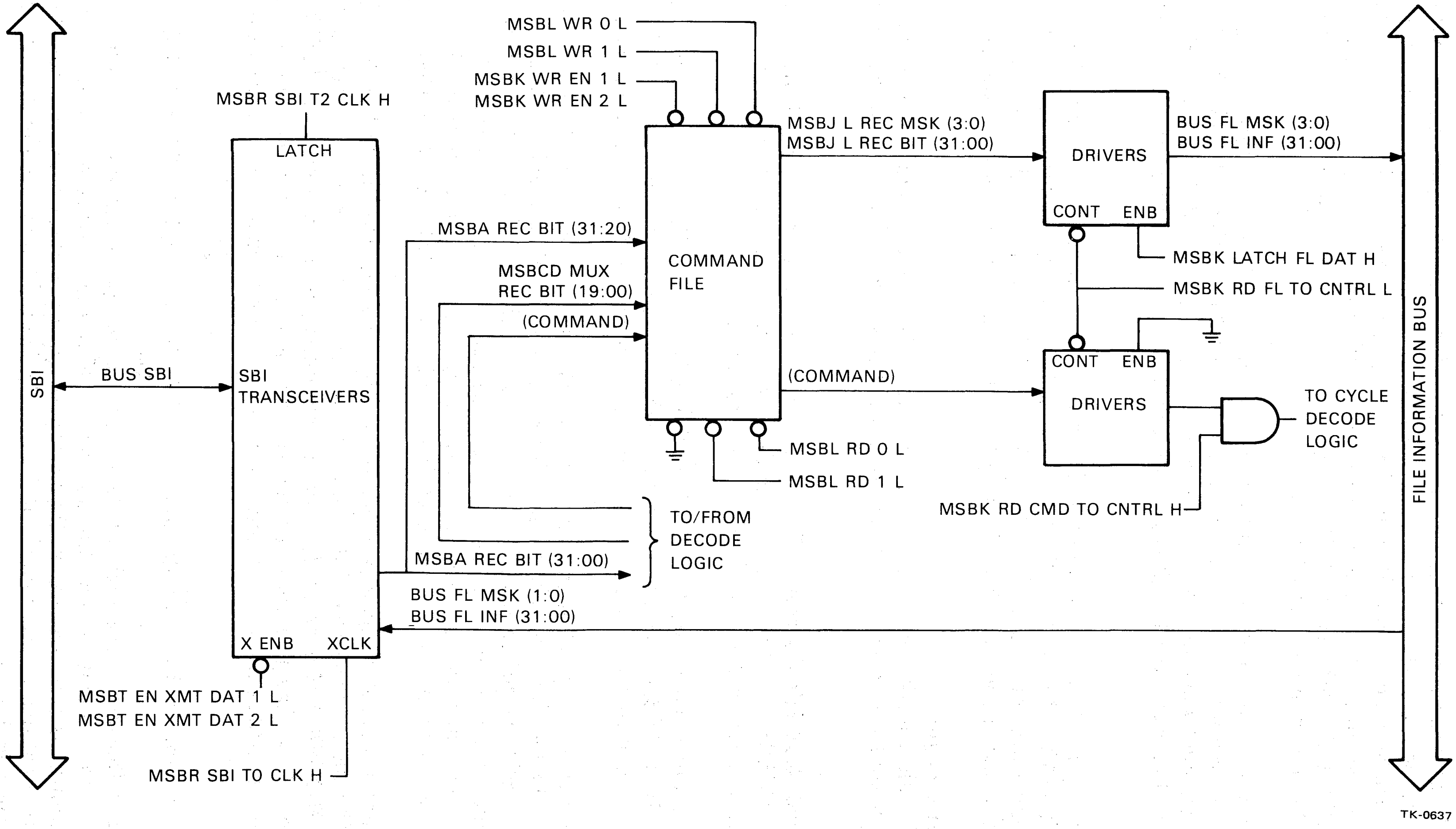


Figure 3-3 Command File Information Path

Information is removed from the file when MSBK RD FL TO CNTRL L is generated. This signal is asserted at SBI T1 providing memory is not busy and the file contains information. MSBK RD CMD TO CNTL H is generated if the information removed from the file includes a command (not write data). This signal enables the command to the cycle decode logic. MSBK RD FL TO CNTRL L is also generated if the file is empty and a valid memory command or data is received. For a valid command in this case, MSBK RD CMD TO CNTRL H is delayed 15 ns to allow the command to propagate through the file before enabling cycle decode.

Information removed from the file is input to drivers on the file information bus. Once on the bus, the information is available for a memory cycle sequence. With MSBK RD FL TO CNTRL L asserted, these drivers are enabled when MSBK LATCH FL DAT H is generated. MSBK LATCH FL DAT H is generated 30 ns after each T0. This allows the file output to become stable before being placed on the file bus.

When read data is placed on the file information bus at the end of a memory read cycle sequence, MSBT EN XMT DAT 1 L and MSBT EN XMT DAT 2 L are generated to enable the SBI transmitters. These signals are asserted when MSBT ARB OK L is generated indicating the SBI is available (i.e., memory's transfer request has been honored) for the transmission of the read data.

3.3.1 File Control Logic

The file control logic consists of a 2-bit write counter, a 2-bit read counter, a file overflow flag, a difference decoder, and a room-in-file comparator. All of this logic is found on MSBL of the engineering print set.

3.3.1.1 Write and Read File Counters – The output of the write-file counter (MSBL WR 1:0 L) and read-file counter (MSBL RD 1:0 L) select one of four command file locations in binary form. The operation of each counter is described in the following paragraphs.

The write-file counter is advanced by the generation of MSBK ADV WR CNT H. This signal is generated at SBI T0 if MSBH VAL DAT H is asserted and there is room in the file. MSBH VAL DAT H is asserted as a result of the decoding of a valid memory destined information format.

The read-file counter is advanced by the generation of MSBK CLK FL TO CNTRL H. This signal is generated at T1 if the file contains information (MSBL RD-WR 0 H asserted) and memory is not busy (MCND MEM T BSY H deasserted). This signal is also generated if the file is empty and a valid memory command or data is received. For this case the generation of this signal is delayed for 15 ns. The delay allows the command or data to propagate through file before the read-file counter is advanced. (The write-file counter is advanced at T0 if MSBH VAL DAT H is asserted.)

The read-file counter logic also includes an overflow flag. This flag (MSBL OV H) is set at SBI T3 if the file is full. The file is considered full if the write-file counter is one location behind the read-file counter (MSBL RD WR 1 H generated) when the write-file counter is advanced (MSBK ADV WR CNT H). The overflow flag prohibits further advancement of the write-file counter. This flag is dropped when the read-file counter is advanced (an information format is read from the file creating a vacancy).

During initialization the read-file counter and write-file counter are set to 00 by MSBF INIT 1 L.

3.3.1.2 Difference Decoder – The logic of the difference decoder performs a 2's complement arithmetic operation on the contents of the read- and write- file counters. During the operation the content of the read-file counter is subtracted from the content of the write-file counter. The result specifies the number of empty locations in the command file. This encoded number is input to the room-in-file comparator (Paragraph 3.3.1.3).

Since a result of 00 can indicate that the file is completely empty or completely full, an overflow flag is included in the file control logic. (Paragraph 3.3.1.1.)

3.3.1.3 Room-In-File Comparator – The room-in-file comparator determines if there is enough space in the command file for an incoming command string. This logic performs a comparison between the amount of empty file locations and the encoded number of locations required to store the command string. If the file contains enough locations, MSBL RM IN FL H is generated. This signal must be asserted to advance the write-file counter. The failure to generate MSBL RM IN FL H results in a busy or error confirmation.

3.4 DATA PATH LOGIC

The data transfer logic provides a data path between the file information bus and the MOS data bus. Before studying the logic implementation, the reader should understand the basic data flow on the MOS tristate data bus. The three general cases of data flow are illustrated in Figure 3-4 and explained in the following text. This figure is also an introduction into the ECC strategy.

As seen in this figure, during a read masked, interlock read masked, or extended read operation:

1. The data quadword and check bits of the addressed location are placed on the MOS data bus by the array board output data buffers.

2. This data and check bits are then latched by the ECC logic for error checking. Correctable errors will be corrected.

3. Once ECC is complete, the data is placed back onto the MOS data bus.

4. The requested data is then available for transfer to the file information bus. (If a correctable error was detected during ECC, a rewrite is initiated and the corrected data is also transferred back to the array.)

For a write masked, interlock write masked, or extended write masked that is not a full write, a read operation is executed before the write operation:

1. The data quadword and check bits of the addressed location are placed on the MOS Data bus by the array board output data buffers.

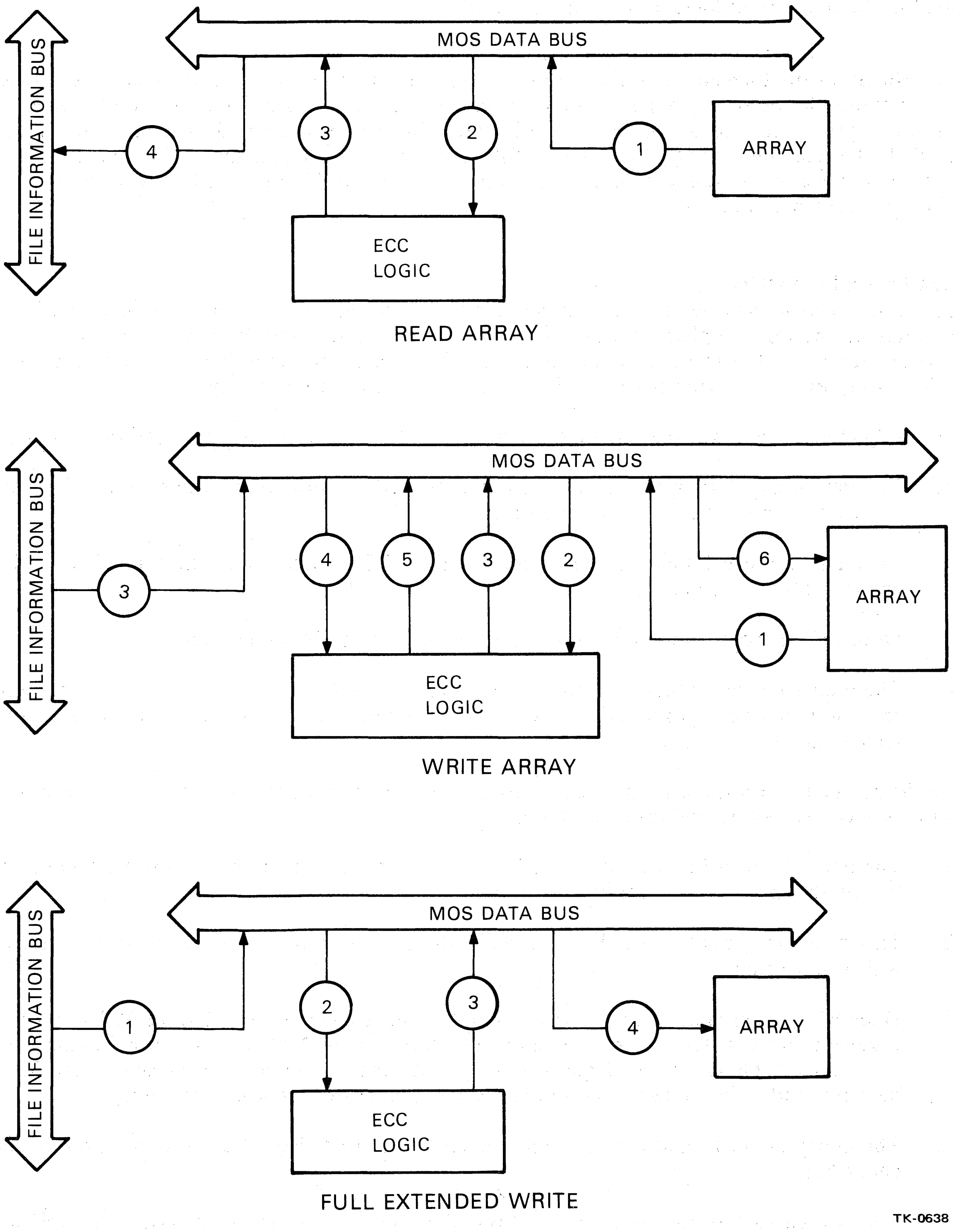
2. This data and check bits are then latched by the ECC for error checking. Correctable errors will be corrected.

3. Once ECC is complete, the bytes that will not be replaced are returned to the MOS data bus. Likewise, the new data bytes (write data) are placed on the MOS data bus for alignment. (If an uncorrectable error was detected during ECC, the new data is not placed on the MOS data bus and only the bad data and check bits are written back into the array.)

4. The newly aligned quadword is then latched by the ECC logic for the generation of new check bits.

5. The new check bits are placed on the MOS data bus for transfer to the array with the new data.

6. The new data and check bits are transferred to the array.



READ ARRAY

WRITE ARRAY

FULL EXTENDED WRITE

Figure 3-4 Basic Data Flow on the MOS Data Bus

For the case of a full extended write masked (the entire quadword rewritten), a read for error checking is not performed before the write operation:

1. The new data quadword is placed on the MOS data bus.
2. This data is then latched by the ECC logic for the generation of check bits.
3. The check bits are placed on the MOS data bus for transfer to the array with the new data.
4. The new data and check bits are transferred to the array.

3.4.1 Data Transfer Logic

Figure 3-5 illustrates the logic associated with the transfer of data between the file information bus and the MOS data bus.

During a write operation, the appropriate write data latch is enabled by signals generated in the mask logic (Paragraph 3.4.3) to latch data from the file bus. Only the selected bytes of the proper longword are placed on the MOS data bus to be aligned with data previously fetched from the array and error checked by the ECC logic. Once aligned with the new data, the modified quadword is latched by the ECC logic. This time the ECC logic generates check bits for the modified data (Paragraph 3.4.2). Once the check bits have been generated, they are placed on the MOS data bus for transfer to the array with the new quadword.

For a read operation, the entire quadword containing the requested data is initially placed on the MOS data bus along with its check bits by the array logic. The ECC logic then latches and error checks the data. Any correctable errors are corrected at this time. The data is then once again placed on the MOS data bus. In this way the requested longword is available for selection by the data transmit multiplexer.

The data transmit multiplexer provides selection of the upper or lower longword from the MOS data bus for transfer onto the file information bus. RD DAT EN L is generated as a result of cycle decode to enable the multiplexer. The condition of RD LO SEL L selects the upper or lower longword.

3.4.2 Error Check and Correction Logic (ECC)

Figure 3-6 illustrates the logic associated with ECC. The following paragraphs describe the ECC logic operation for read and write operations.

3.4.2.1 ECC During Reads – During a read operation to main memory, the memory quadword addressed is fetched from the array and placed on the MOS data bus along with its check bits. The data and check bits are then latched by the array receive data latch and the array receive check bit latch for error checking. This is accomplished by the assertion of MCND CHK BIT DAT CK H and MCND ARY DAT CK H.

The data portion is presented directly to the error check and syndrome logic. Also input to this logic are the latched check bits via the check bits multiplexer. (The check bits multiplexer provides the capability of selecting check bits from configuration register B for diagnostic purposes.) The resultant error syndrome (SYN 7:0) is presented to the error detect logic, the syndrome decode logic, and the tag generator logic.

The tag generator logic generates a tag to indicate whether or not the data is correct, and if not correct, whether it is correctable. The resultant tag is transferred to the mask logic for transmission with the requested data. In this way, it defines the data as read data, corrected read data, or read data substitute.

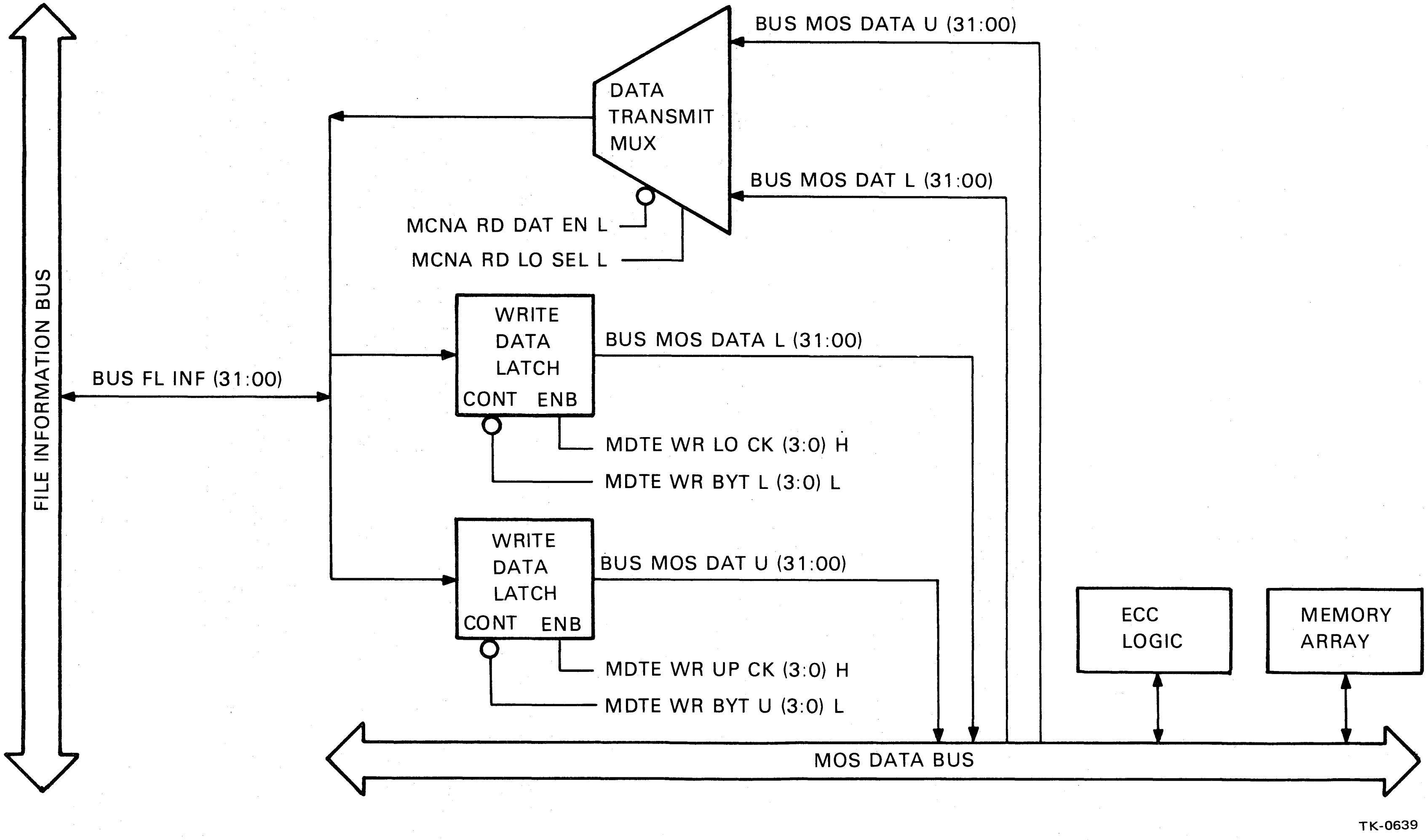


Figure 3-5 Data Transfer Logic

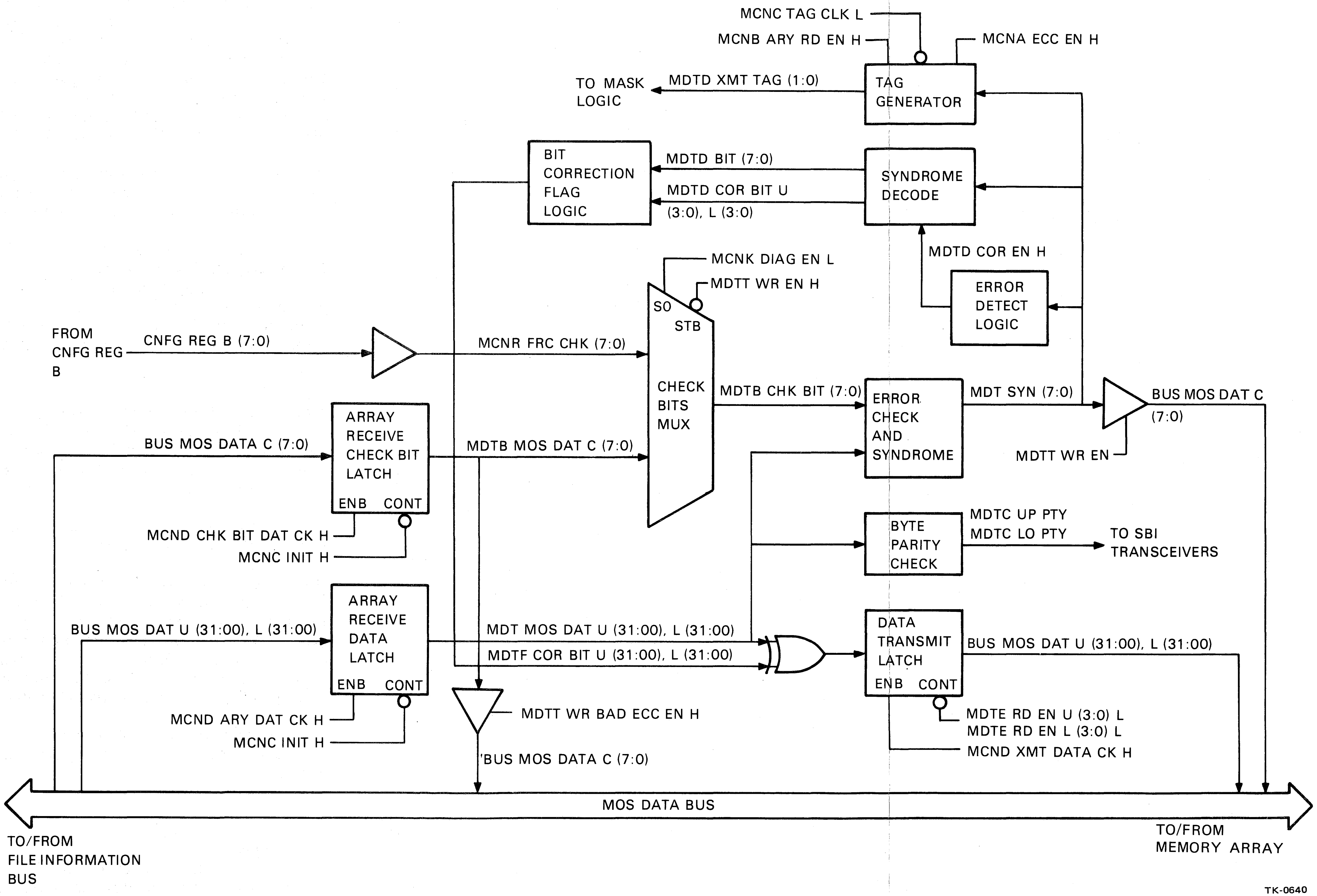


Figure 3-6 Error Check and Correction Logic (ECC)

The error detect logic is capable of determining if a single or double bit error exists in the quadword. If a single bit error exists (i.e., correctable error), MDTD COR EN H is generated. This signal enables the syndrome decode logic, which identifies the bit and byte in error. The bit correction flag logic employs eight bit lines and eight byte lines, which produce an error flag for the appropriate bit in error. The appropriate bit flag is then input to exclusive-OR gates along with the original latched quadword. The exclusive OR function changes only the bit indicated by the bit correction flag logic. The corrected quadword is then placed back onto the MOS data bus via the data transmit latch, for transfer to the file information bus. Note that the original check bits remain on the bus.

If an uncorrectable error is detected by the error detect logic, the syndrome decode logic and data transmit latch are not enabled. For this case, the data and check bits on the MOS data bus remain unaffected for transfer to the file information bus. This also occurs if no error is detected.

To conform to SBI protocol, a parity bit is generated for transmission with the requested longword. This is accomplished by parity generators in the ECC logic. The quadword containing the requested data is presented to these parity generators when it is latched from the MOS data bus by the array receive data latch for ECC. Two parity bits are generated; one for the upper longword and one for the lower longword. The appropriate parity bit (upper or lower) is then selected for transmission with the corresponding longword by the signal MCNA RD LO SEL L.

3.4.2.2 ECC During Writes – During a write operation to main memory, the memory quadword involved is fetched from the array and placed on the MOS data bus along with its check bits. The data and check bits are then latched by the ECC logic for error checking just as during a read operation (Paragraph 3.4.2.1). When ECC is complete, only the bytes that will not be replaced are returned to the MOS data bus via the data transmit latch for alignment with the new data.

If an uncorrectable error is detected during ECC, the data and check bits on the MOS data bus remain unaffected for a rewrite to the array. That is, the new data bytes are not placed on the MOS data bus.

If an uncorrectable error is not detected, the new data byte or bytes are transmitted onto the MOS data bus for alignment when the error check is complete. This is accomplished by the data receive latch of the data transfer logic (Paragraph 3.4.1). Once aligned, the new quadword is latched by the array receive data latch for the generation of new check bits. The new check bits are generated by the error check and syndrome logic. This logic presents the new check bits to MOS bus drivers. MDTT WR EN is generated to transmit these check bits onto the MOS data bus. With the new data quadword and corresponding check bits on the MOS data bus, a write to the array is executed.

Note as mentioned previously, the ECC logic does not perform error checking in the case of a full extended write (no read cycle is performed). For this case, the data bus latch and check bit latch are not enabled.

3.4.3 Mask Logic

When a command is removed from the file and placed on the file information bus, the mask is decoded. This decode generates signals to select the appropriate data bytes of the addressed longword for the ECC and data transfer logic. For extended write masked operations, the mask of the first write data format is also decoded to select the bytes of the second write data format.

The mask decode logic is illustrated in Figure 3-7. As seen in this figure, the mask is latched from the file information bus as MDTE BYT MSK U(3:0) H and MDTE BYT MSK L(3:0) H. These signals are input to various drives for the data transfer logic and a multiplexer for the ECC logic.

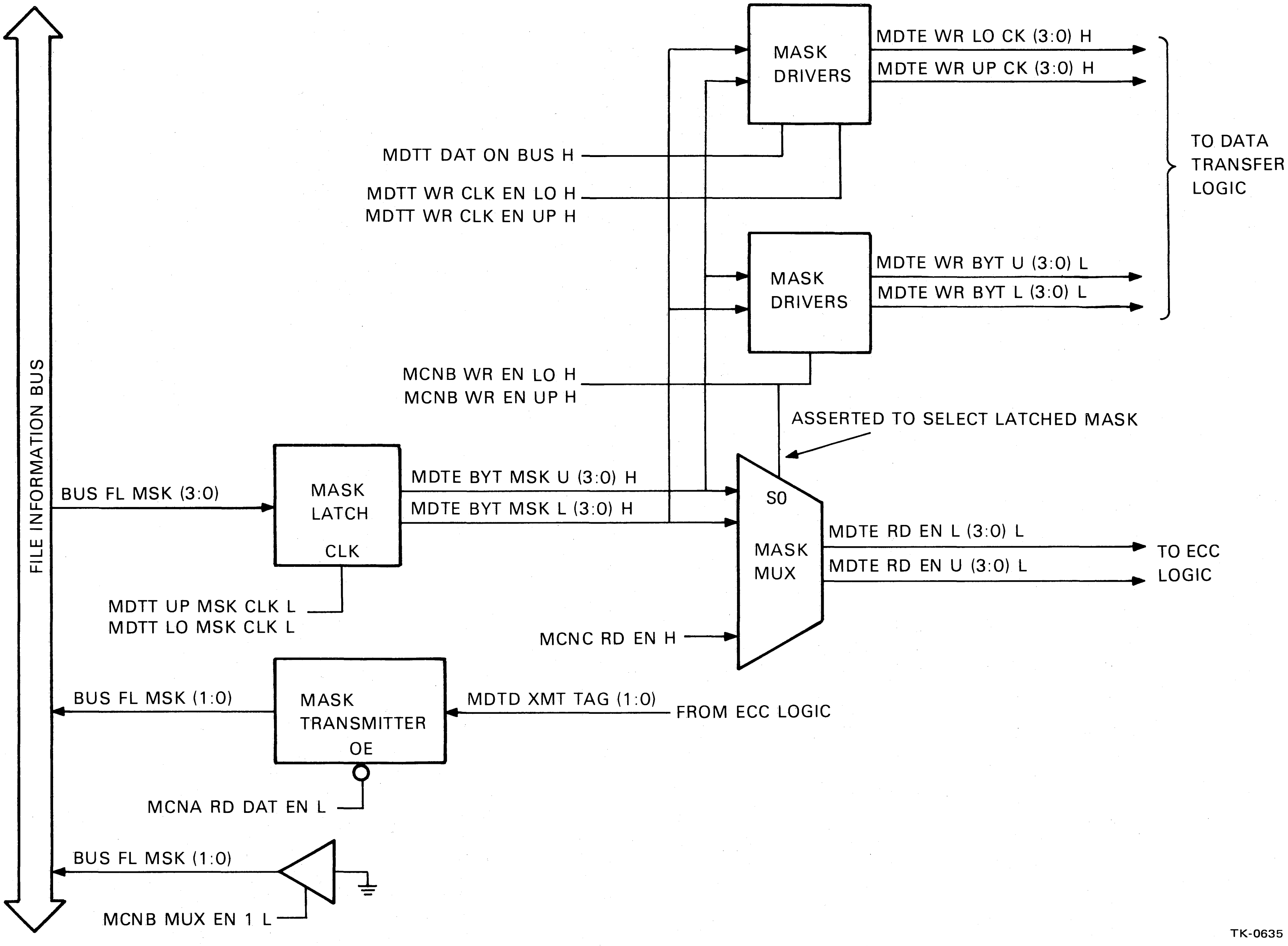


Figure 3-7 Mask Logic

3.4.3.1 Mask Multiplexer – The mask multiplexer can be selected to supply the ECC logic with the latched mask or a mask of all ones. During a write masked operation, MCNB WR EN LO H or MCNB WR EN UP H is generated to select a latched mask for the generation of MDTE RD EN L(3:0) L or MDTE RD EN U(3:0) L. These signals are generated to select bytes of the appropriate longword (high or low) for transmission onto the MOS data bus from the ECC logic. The data is placed on the MOS bus at this time for alignment with the new data. In the case of an extended write masked, both MDTE BYT MSK U(3:0) H and MDTE BYT MSK L(3:0) H are selected to transmit bytes from the entire quadword onto the MOS data bus.

During a read masked, interlock read masked, or extended read cycle, all bytes of the selected longword are required for transmission onto the MOS data bus after ECC. For this reason, MCNC RD EN H (equivalent to a mask of all ones) is selected by the mask multiplexer when neither type of write is being executed.

3.4.3.2 Mask Drivers – MCNB WR EN LO H and MCNB WR EN UP H not only select a mask for the ECC logic, as previously described, but also control the generation of MDTE WR BYT U(3:0) L and MDTE WR BYT L(3:0) L in the data transfer logic. These signals, along with MDTE WR LO CK(3:0) H and MDTE WR UP CK(3:0) H are used to enable the data receive latches in the data transfer logic (Paragraph 3.4.1). The latches are enabled to transfer the appropriate data bytes from the file information bus to the MOS data bus during writes.

The generation of MDTE WR LO CK(3:0) H and MDTE WR UP CK(3:0) L is similar to that of MDTE RD EN L(3:0) L and MDTE RD EN U(3:0) L described previously. During a write masked, either MDTT WR CLK EN UP H or MDTT WR CLK EN LO H is asserted to generate MDTE WR LO CK(3:0) H or MDTE WR UP CK(3:0) L. For extended writes, both signals are asserted.

If the mask latched from the file information bus is equal to all ones, MDTE FULL WR EN H is generated. This signal is used to generate MCND FULL WR H during extended write masked operations. For these operations, MCND FULL WR H generates MCNB WR EN LO H and MCNB WR EN UP H to transfer the new quadword from the ECC logic onto the MOS bus. (The new quadword was latched by the ECC logic, in this case, to generate check bits.)

3.4.3.3 Masks for Transmissions – During a read operation, a mask is generated by the ECC logic to specify the integrity of the requested data. This mask is input to transmitters on the file information bus for transfer to SBI transceivers.

The mask transmitters are shown in Figure 3-7. Note the ECC logic generates only the two low-order mask bits, MDTD XMT TAG (1:0). (These bits should not be confused with the tag of an SBI format.) These mask bits are transmitted onto the file bus when MCNA RD DAT EN L is generated. This signal also enables the data transmit multiplexer to transfer read data from the MOS data bus to the file information bus (Paragraph 3.4.1). The two high-order mask bits are supplied by the SBI transceivers during transmission. These two high-order bits are low for any type of read data and so are not deependent on ECC results.

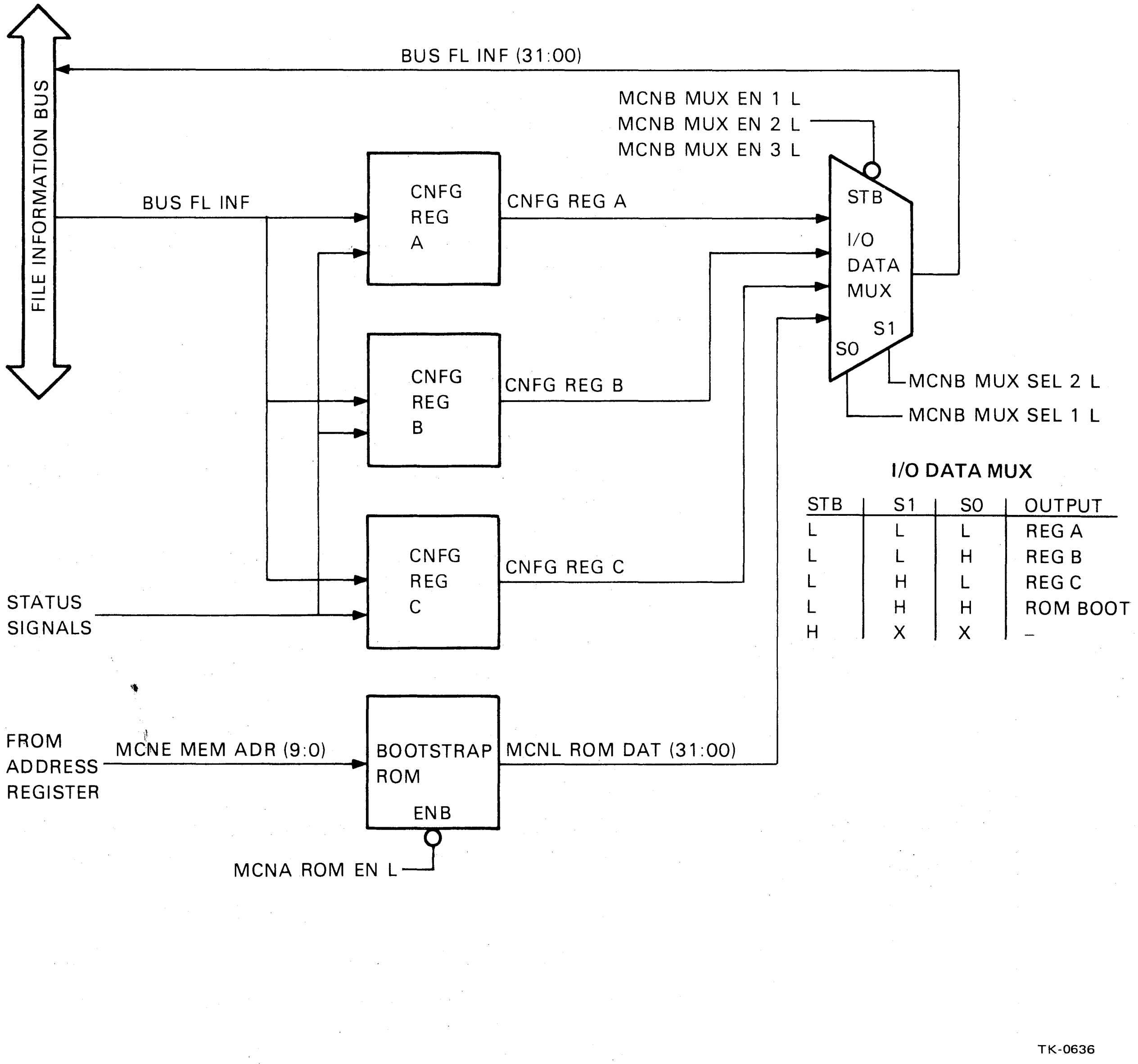
A mask of all zeros is transmitted with any read data requested from the configuration registers or bootstrap ROM. For this case, MCNB MUX EN 1 L is generated to enable file information bus drivers (Figure 3-7). This signal also enables the I/O data multiplexer (Paragraph 3.5).

3.5 I/O DATA LOGIC

The I/O data logic, illustrated in Figure 3-8, is located on the MCN board. As seen in the figure, this logic includes the three configuration registers, the bootstrap ROM, and the I/O data multiplexer.

The contents of the ROM and registers are visible over the file information bus by the proper selection of the I/O data multiplexer. (The function table for the multiplexer is also shown in Figure 3-8.) The configuration registers contain status information for the operating system and diagnostic software. The ROM contains informationn for booting the system.

Each of the configuration registers is described in the following paragraphs. An illustration of each register is included to summarize its contents. A description of the logic associated with the ROM is also included.



I/O DATA MUX

STB	S1	SO	OUTPUT
L	L	L	REG A
L	L	H	REG B
L	H	L	REG C
L	H	H	ROM BOOT
H	X	X	–

Figure 3-8 I/O Data Logic

3.5.1 Configuration Register A

Figure 3-9 illustrates the contents of configuration register A. Each bit is described below in Table 3- 1.

Table 3-1 Configuration Register A

Bit	Function	Description
31:26	SBI Fault Flags	Bit 31, bus parity fault – When set indicates an SBI information path parity error has occurred. May be asserted in one or more other nexus simultaneously. This bit is read-only. Bit 30, write sequence error – This bit is set when a write masked, extended write masked, or interlock write masked command is not immediately followed by a write data format. This bit is read-only. Bit 29, not used (= 0) Bit 28, interlock sequence fault – Set when an interlock write masked command is received, but interlock has not been set by an interlock read masked command. This bit is read-only. Bit 27; multiple transmitters – This bit is set when multiple transmitters are detected on the SBI. (Detection is accomplished by comparing the received ID with the transmitted ID one cycle after transmission.) This bit is read-only. Bit 26, transmit fault – Set if memory was transmitter when a fault occurred. This bit is read/write 1 to clear.
23,22	Power UP/DOWN	Bit 23, power up – If set, indicates a power-up sequence is being executed. This bit is read/write 1 to clear. Bit 22, power down – If set, indicates a power-down sequence is being executed. This bit is read/write 1 to clear.
14:9	Memory Array Size	These bits indicate the array size as shown below. Note bits 13 and 14 are not used for 4K chip applications.

Bits				**Array Size (bytes)**
12	**11**	**10**	**09**	
0	0	0	0	64K
0	0	0	1	128K
0	0	1	0	192K
0	0	1	1	256K
0	1	0	0	320K
0	1	0	1	384K
0	1	1	0	448K

Table 3-1 Configuration Register A (Cont)

<table>
<tr><th>Bit</th><th>Function</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th colspan="4">Bits</th><th>Array Size (bytes)</th></tr>
<tr><th>12</th><th>11</th><th>10</th><th>09</th><th></th></tr>
<tr><td>0</td><td>1</td><td>1</td><td>1</td><td>512K</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>0</td><td>576K</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>1</td><td>640K</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>0</td><td>704K</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>1</td><td>768K</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>0</td><td>832K</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>1</td><td>896K</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>0</td><td>960K</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>1</td><td>1024K</td></tr>
</table>
These bits are encoded from the number of array boards plugged into the backplane and are read-only. (Refer to configuration register in the engineering print set for 16K array size.)
</td></tr>
<tr><td>08</td><td>Interleave Enable</td><td>This bit must be set in order to write into bits (02:00) of this register. This bit is write-only.</td></tr>
<tr><td>04,03</td><td>Memory Type</td><td>These bits indicate the type of memory chips used in the array as shown below:
<table>
<tr><th colspan="2">Bits</th><th>Memory Type</th></tr>
<tr><th>04</th><th>03</th><th></th></tr>
<tr><td>0</td><td>0</td><td>Error Condition, no array boards are plugged into the backplane.</td></tr>
<tr><td>0</td><td>1</td><td>4K MOS</td></tr>
<tr><td>1</td><td>0</td><td>16K MOS</td></tr>
<tr><td>1</td><td>1</td><td>Error Condition, both 4K and 16K chip array boards are plugged into the backplane.</td></tr>
</table>
</td></tr>
<tr><td>02:00</td><td>Interleave Status</td><td>These bits indicate the interleave status as shown below:
<table>
<tr><th colspan="3">Bits</th><th>Interleave Status</th></tr>
<tr><th>02</th><th>01</th><th>00</th><th></th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>Non-Interleaved</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>2-way Interleaved</td></tr>
</table>
These bits are read/write. (Read-only if bit 08 is set.)
</td></tr>
</table>

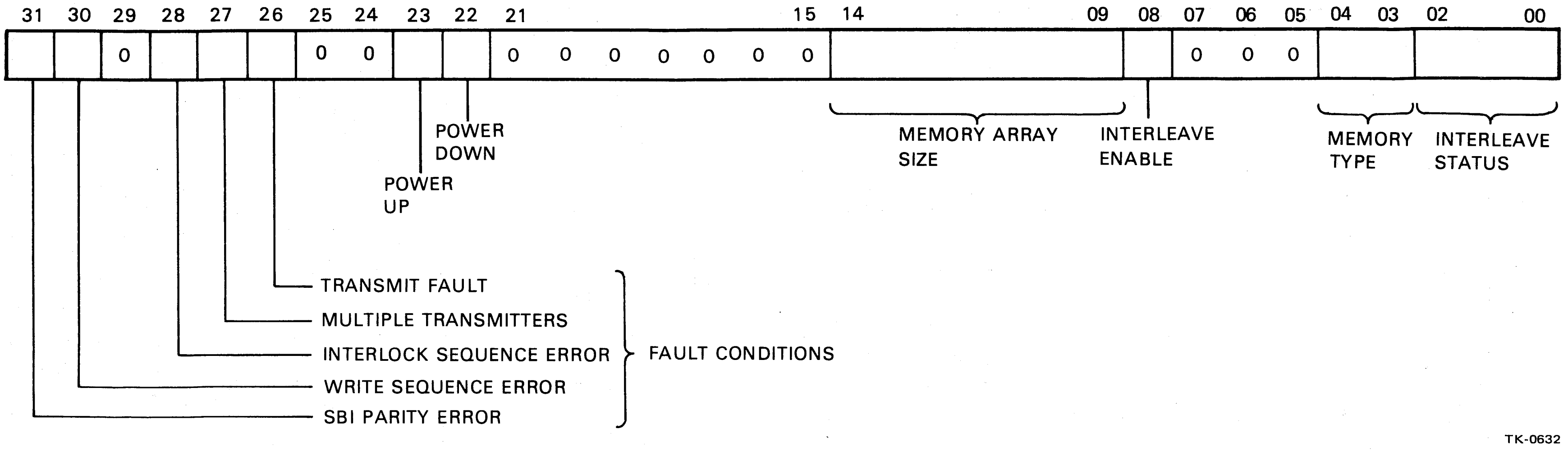


Figure 3-9 Configuration Register A

3.5.2 Configuration Register B

Figure 3-10 illustrates the contents of configuration register B. Table 3-2 describes each bit.

Table 3-2 Configuration Register B

<table>
<tr><th>Bit</th><th>Function</th><th>Description</th></tr>
<tr><td>31,30</td><td>File Output Pointer</td><td>These bits indicate the next file location to be read. These bits are read-only.</td></tr>
<tr><td>29,28</td><td>File Input Pointer</td><td>These bits indicate the next available file location into which command/address or data information can be written. These bits are read-only.</td></tr>
<tr><td>27:15</td><td>Controller's Starting Address</td><td>These bits indicate the controller's starting address in 64K byte increments. These bits are read/write but can only be written when bit 14 is set.</td></tr>
<tr><td>14</td><td>Starting Address Write Enable</td><td>This bit must be set to write bits (27:15). This bit is write-only.</td></tr>
<tr><td>13,12</td><td>Initialization</td><td>These bits contain recovery mode information necessary to determine whether or not memory recovered from battery backup. The status is interpreted as follows:
<table>
<tr><th colspan="2">Bits</th><th>Status</th></tr>
<tr><th>13</th><th>12</th><th></th></tr>
<tr><td>0</td><td>0</td><td>Initialization cycle in progress. (That is, a known data pattern and check code is being written through all memory locations. A command issued to the array at this time will receive BSY.)</td></tr>
<tr><td>1</td><td>0</td><td>(Not valid)</td></tr>
<tr><td>0</td><td>1</td><td>Memory contains valid data in this state.</td></tr>
<tr><td>1</td><td>1</td><td>Initialization is complete. No data was preserved.</td></tr>
</table>
These bits are read-only.</td></tr>
<tr><td>09</td><td>Force Error</td><td>When set, this bit enables the ECC substitute bits (07:00 of this register) as replacements for the actual check bits during ECC computation. This bit is read/write and is for diagnostic use only.</td></tr>
<tr><td>08</td><td>ECC Bypass</td><td>When set, this bit eliminates the ECC function. Under this condition, the data read from memory is placed on the SBI exactly as is. In addition, CRD and RDS tags are not used. The error log, however, continues to operate normally as described in Table 3-3. This bit is read/write.</td></tr>
<tr><td>07:00</td><td>ECC Substitute</td><td>When bit 09 of this register is set, these bits are used as a substitute for the check bits read from memory.
Substitute ECC bits can only be used on the data of SBI addresses with bits 03 and 12 set. The ECC substitute bits are read/write and are for diagnostic use only.</td></tr>
</table>

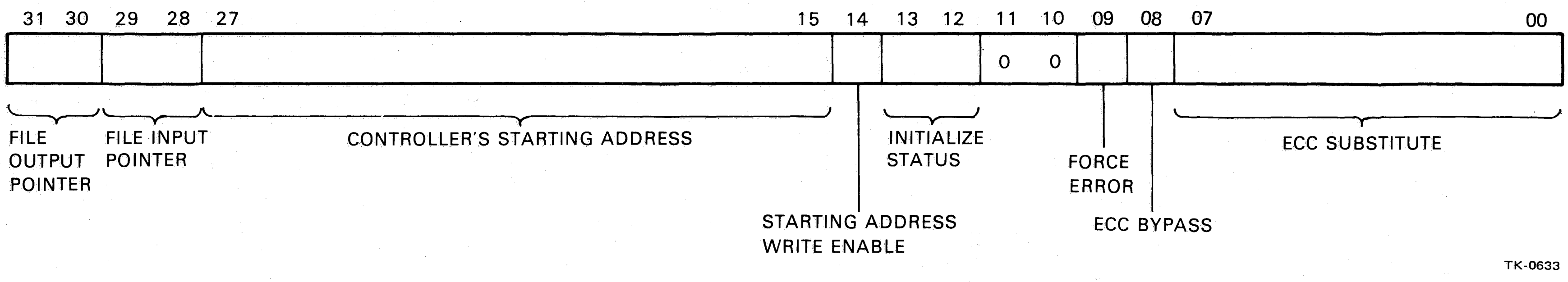


Figure 3-10 Configuration Register B

3.5.3 Configuration Register C

Figure 3-11 illustrates the contents of configuration register C, which contains all ECC error information. Table 3-3 describes each bit.

Table 3-3 Configuration Register C

<table>
<tr><th>Bit</th><th>Function</th><th>Description</th></tr>
<tr><td>30</td><td>Inhibit CRD</td><td>When set, any CRD tags are prevented from being transmitted to the commander. This bit is read/write.</td></tr>
<tr><td>29</td><td>High Error Rate</td><td>When set, this bit indicates a second error has occurred before the first error was able to be serviced by the operating system. This bit is read/write 1 to clear.</td></tr>
<tr><td>28</td><td>Error Log Request</td><td>When set, this bit indicates an error has occurred. This bit is set during the controller's response to an SBI read cycle when an error occurs. Its primary purpose is to indicate which memory controller transmitted an error message in systems with multiple memory controllers. When set, any subsequent CRD transmissions to the SBI commander are inhibited. This bit is read/write 1 to clear.</td></tr>
<tr><td>27:08</td><td>Error Address</td><td>These bits indicate the SBI longword address at which the first read error has occurred. Any subsequent error addresses are not saved until the first error is serviced. The bit assignment of the error address is as follows:
<table>
<tr><th></th><th>4K Chips</th><th>16K Chips</th></tr>
<tr><td>Array Board</td><td>27:24</td><td>27:24</td></tr>
<tr><td>Array Bank</td><td>21</td><td>23</td></tr>
<tr><td>Chip Address</td><td>20:09</td><td>22:09</td></tr>
<tr><td>Up/Low Longword</td><td>8</td><td>8</td></tr>
</table>
These bits are read-only.</td></tr>
<tr><td>07:00</td><td>Error Syndrome</td><td>These bits are loaded with an error syndrome when an error is detected during the response to an SBI read command providing a syndrome has not already been loaded. [The error syndrome points to the bit(s) in error.] The syndrome is retained until the error service routine has serviced the error.
With a syndrome occupying these bits, subsequent syndromes are not retained. The error(s), however, are indicated by bit 29 of this register.
These bits are read-only.</td></tr>
</table>

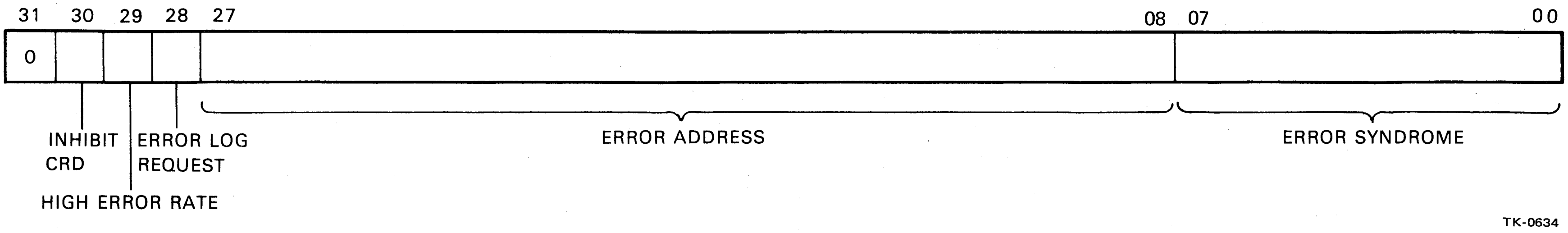


Figure 3-11 Configuration Register C

3.5.4 Bootstrap ROM

The bootstrap ROM is a 4 kilobyte programmable read only memory in the memory controller. The ROM is organized as a 1K × 32 matrix and is assigned a 4K byte I/O address space.

The bootstrap ROM is enabled during initialization when MCNA ROM EN L is generated. This signal is generated as a result of cycle decode. A 10-bit address (MCNE MEM ADR 09:00) from the address register selects each ROM location.

3.6 START MEMORY CYCLE LOGIC

All memory cycles are initiated with the generation of MSBK MEM GO L. This signal is generated when the last information format of a valid memory command string is removed from the file. The logic associated with the generation of this signal is shown in Figure 3-12.

Each time an information format is removed from the file MSBK CLK FL TO CNTRL H is generated. As seen in Figure 3-12, MSBK MEM GO L is asserted to start a memory cycle if the format removed is a read command (MSBJ L CMD H and MSBU L RD H are asserted).

When a write data format is removed from the file during a write masked or interlock write masked operation, MSBK EN DAT L is asserted. This signal generates MSBK MEM GO L providing the write abort bit is not set (MSBJ L WR ABT H is not asserted).

When an extended write masked command is removed from the file, MSBK ADR ON BUS H, MSBU L WR H, and MSBU L EXT H are asserted. These signals set and preset flip-flops 1 and 2 of Figure 3-12, respectively. At the following T0 a write data format is removed from the file and MSBK EN DAT L is generated. For this case, however, flip-flop 2's output is preset to low, so MSBK MEM GO L is not generated. At T2 MSBK EN DAT L is negated clocking flip-flop 1. This provides a high input to flip-flop 2. When the second data format is removed from the file at the following T0, flip-flop 2 is clocked. MSBK EN DAT L is also generated. These conditions generate MSBK MEM GO L providing the write abort bit is not set.

3.6.1 Refresh During Single Step

As mentioned previously, a refresh cycle must be performed periodically to preserve the integrity of the data stored in the MOS array. In accordance with this, a refresh cycle is performed approximately every 28 μs (for 4K chips). The refresh request is asserted by MCNE REF CLK H, which is generated by a dedicated clock at the appropriate time. Under normal operation, MCNC REF GO L is then asserted between T0 and T1 of the following SBI cycle providing memory is not busy.

Refresh cycles must also be executed during single-step operation. During single-step, the SBI clock is temporarily stopped. With this, the time between any two time states (T0-T1, 1-T2, T2-T3, T3-T0) can vary from 50 ns to an indefinitely long period of time. While the SBI clock is stopped, however, refresh cycles must continue to be executed periodically. This is accomplished by the logic shown in Figure 3-13.

As seen in Figure 3-13, with MSBF DC LO DEAD H asserted at T3, a one-shot is fired to generate MCNC SBI CLKS STOP H,L. MCNC SBI CLKS STOP H permits the initiation of a refresh cycle by MCNE REF CLK H without the generation of the T0 time state. (T0 is required for a refresh during normal operation, as previously described.)

3.7 ARBITRATION LOGIC

For any valid memory operation the requested data must be transmitted onto the SBI. To gain control of the information lines for the transmission(s) of the data, memory asserts a transfer request (TR). The logic associated with the assertion of a transfer request is shown in Figure 3-14.

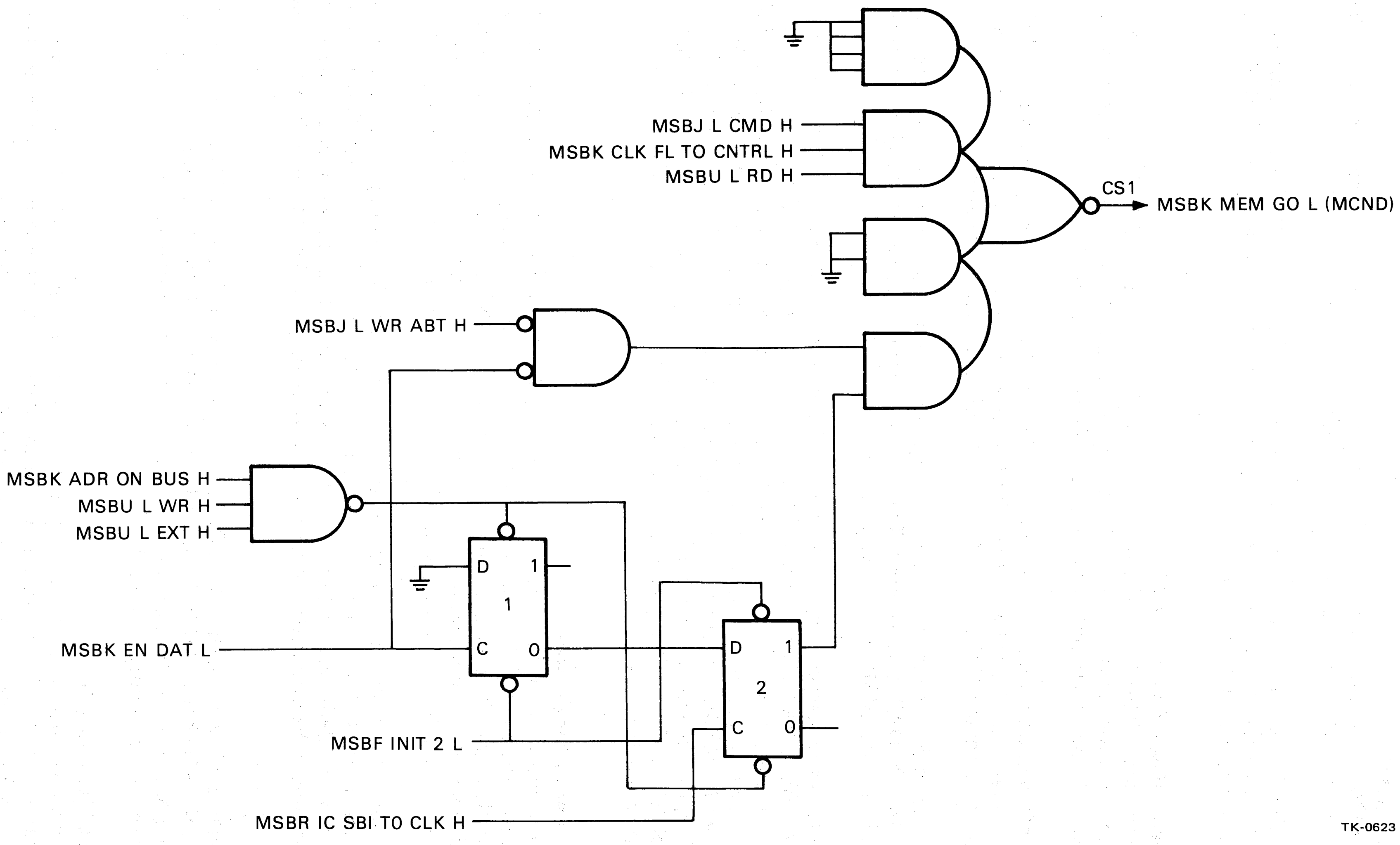


Figure 3-12 Start Memory Cycle Logic

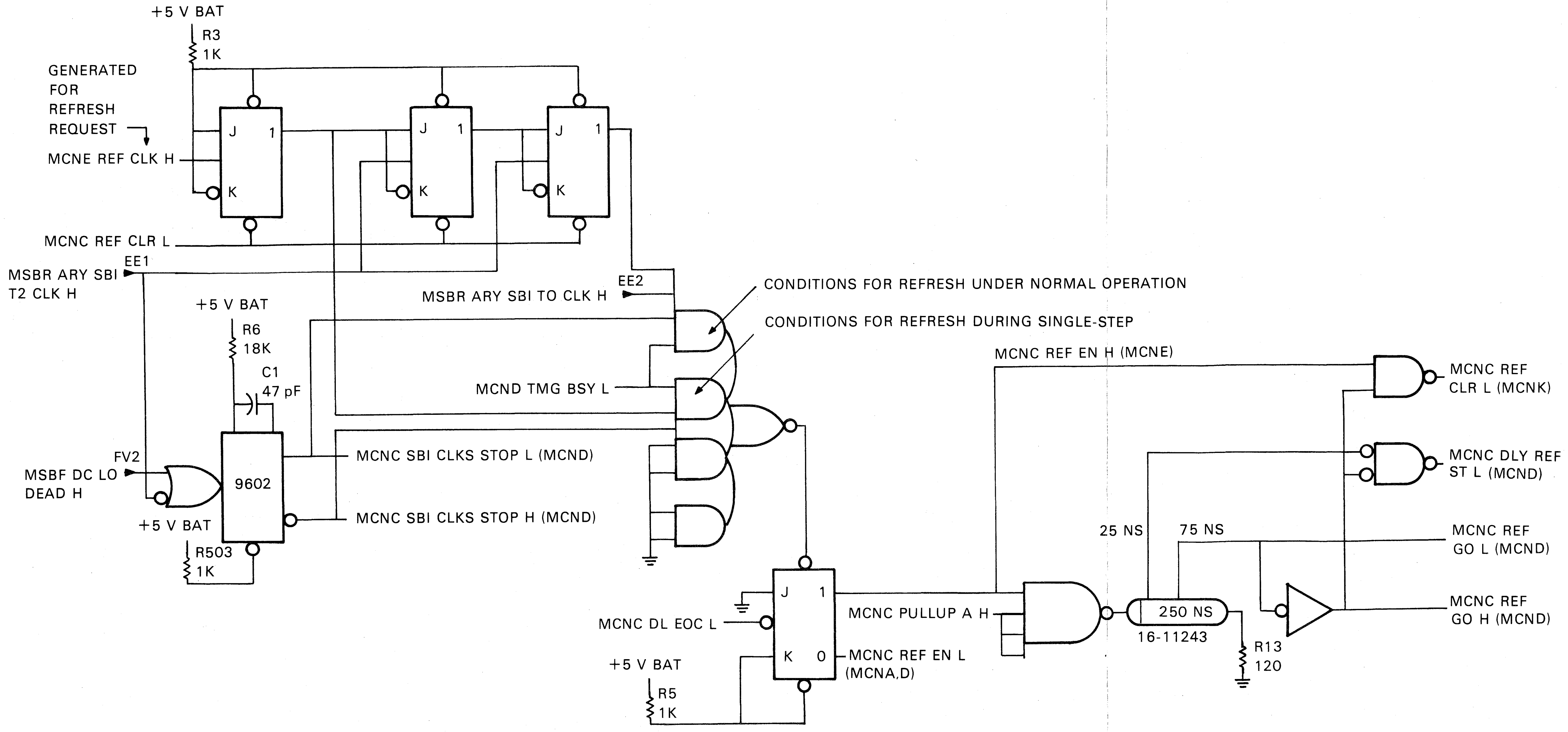


Figure 3-13 Refresh During Single-Step

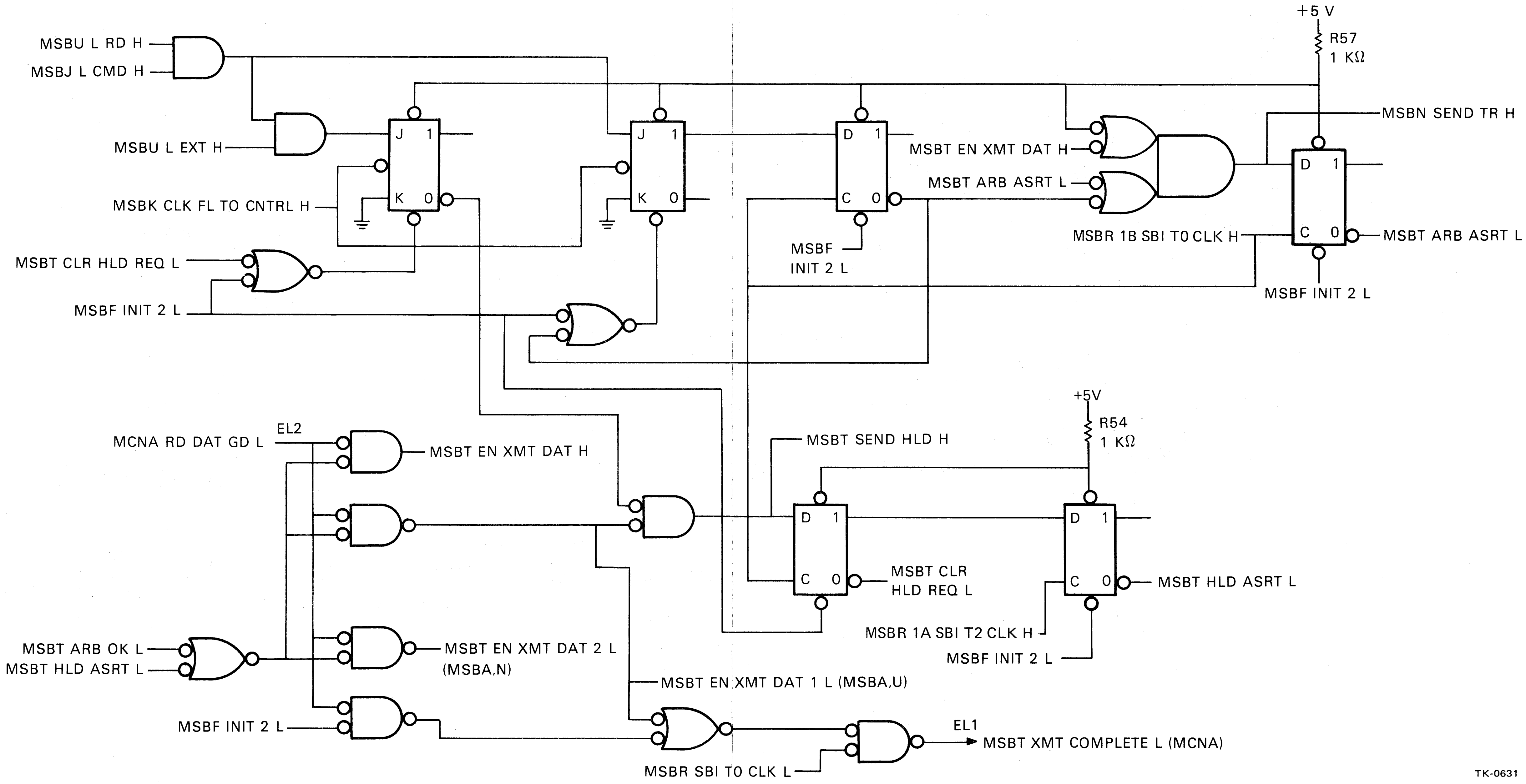


Figure 3-14 Arbitration for Read Data Logic

When a read masked or interlock read masked command is removed from the file, MSBU L RD H and MSBU L CMD H are generated. Likewise, MSBK CLK FL TO CNTRL H is asserted. This results in the generation of MSBN SEND TR H at the following T0 (MSBT EN XMT DAT H is also low). One cycle later, MSBT ARB ASRT L is generated.

The generation of MSBN SEND TR H also enables the assertion of the appropriate TR (MSBT BUS TR L) at the following T0. If no other higher priority TR lines are asserted, MSBT ARB OK L is generated at the next T3 to indicate the bus is available. This signal enables MSBT EN XMT DAT H for the transfer of the requested data (MCNA RD DAT GO L indicates the data is ready). MSBT XMT COMPLETE is asserted at the next T0.

If the operation is an extended read, MSBT SEND HLD H is asserted when MSBT EN XMT DAT H is generated. MSBT SEND HLD H enables the assertion of the highest priority arbitration level. At the following T0, HOLD is transmitted and MSBT CLR HLD REQ L is asserted to negate MSBT SEND HLD H. MSBT HLD ASRT L is generated at T2 to continue to assert MSBT EN XMT DAT H for the transfer of the second longword. MSBT XMT COMPLETE L is also generated for the second time at the following T0.

3.8 MEMORY CONTROL INITIALIZATION LOGIC

The logic associated with memory control initialization is shown in Figure 3-15. As seen throughout the engineering print set, the memory system is initialized by the generation of MSBF INIT (3:1) L. These signals are generated when MSBP REC UNJAM H or MSBF PWR NOT GD L is asserted.

MSBP REC UNJAM H is generated at T0 following the latching of UNJAM from the SBI. UNJAM (BUS SBI UNJAM L) can be latched from the SBI at T2 of any SBI cycle. UNJAM is asserted on the SBI only by the CPU through a console function (Paragraph 1.3.5.).

MSBF PWR NOT GD L is asserted at T3 of an SBI cycle if DEAD is asserted on the SBI (BUS SBI DEAD L) or during power-up (BUS DC LO H). This signal clears the fault flags of Configuration register A.

MSBF DC LO DEAD H is asserted coincidentally with MSBF PWR NOT GD L. MSBF DC LO DEAD H generates MCNH BACK UP MODE H and MCNH BACK UP MODE L. These signals are generated to stabilize the logic during the transition of power-on to battery backup.

3.9 POWER UP/DOWN LOGIC

The logic associated with the power-up and power-down indicators of configuration register A (bits 22 and 23) is illustrated in Figure 3-16. This logic is capable of detecting whether a power-up or power-down sequence is occurring.

The logic of Figure 3-16 was designed on the basis that BUS AC LO H and BUS DC LO H are generated during a power-up and power-down sequence. The order in which these signals are generated determines which sequence is being executed. BUS AC LO H is generated before BUS DC LO H during a power-down sequence. BUS DC LO H is generated before BUS AC LO H during a power-up sequence.

As seen in this figure, the two J-K flip-flops are preset during initialization. Without the generation of BUS AC LO H or BUS DC LO H, the D flip-flops remain set. During a power-down sequence, MSBP REC AC LO L is generated to preset the first D flip-flop. This presents a high to the J input of the power-down flip-flop at T0. At the following T3, MSBP PWR DWN H is generated. Likewise, during a power-up sequence, MSBF DC LO L is generated to preset both D flip-flops. With MSBP REC AC LO L not yet generated, the first D flip-flop presents a high to the J input of the power-up flip-flop at T0. At the following T3, MSBP PWR UP H is generated.

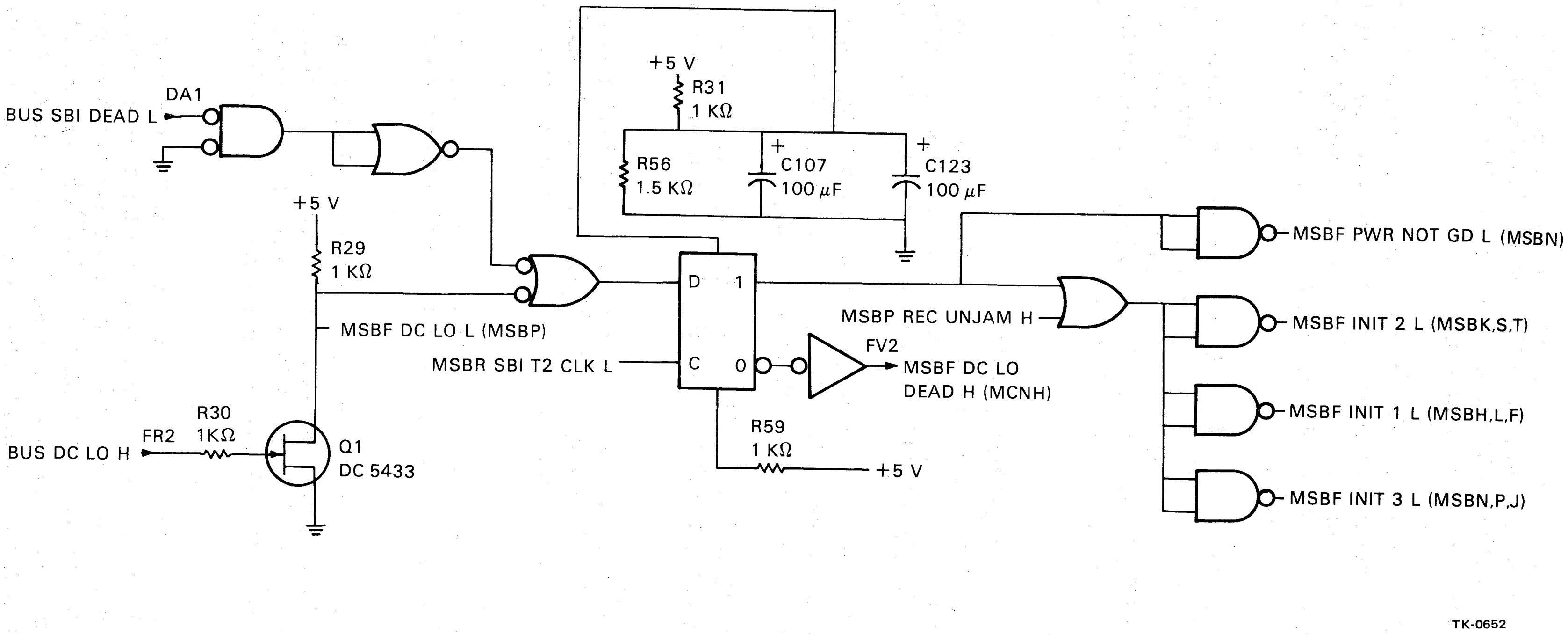
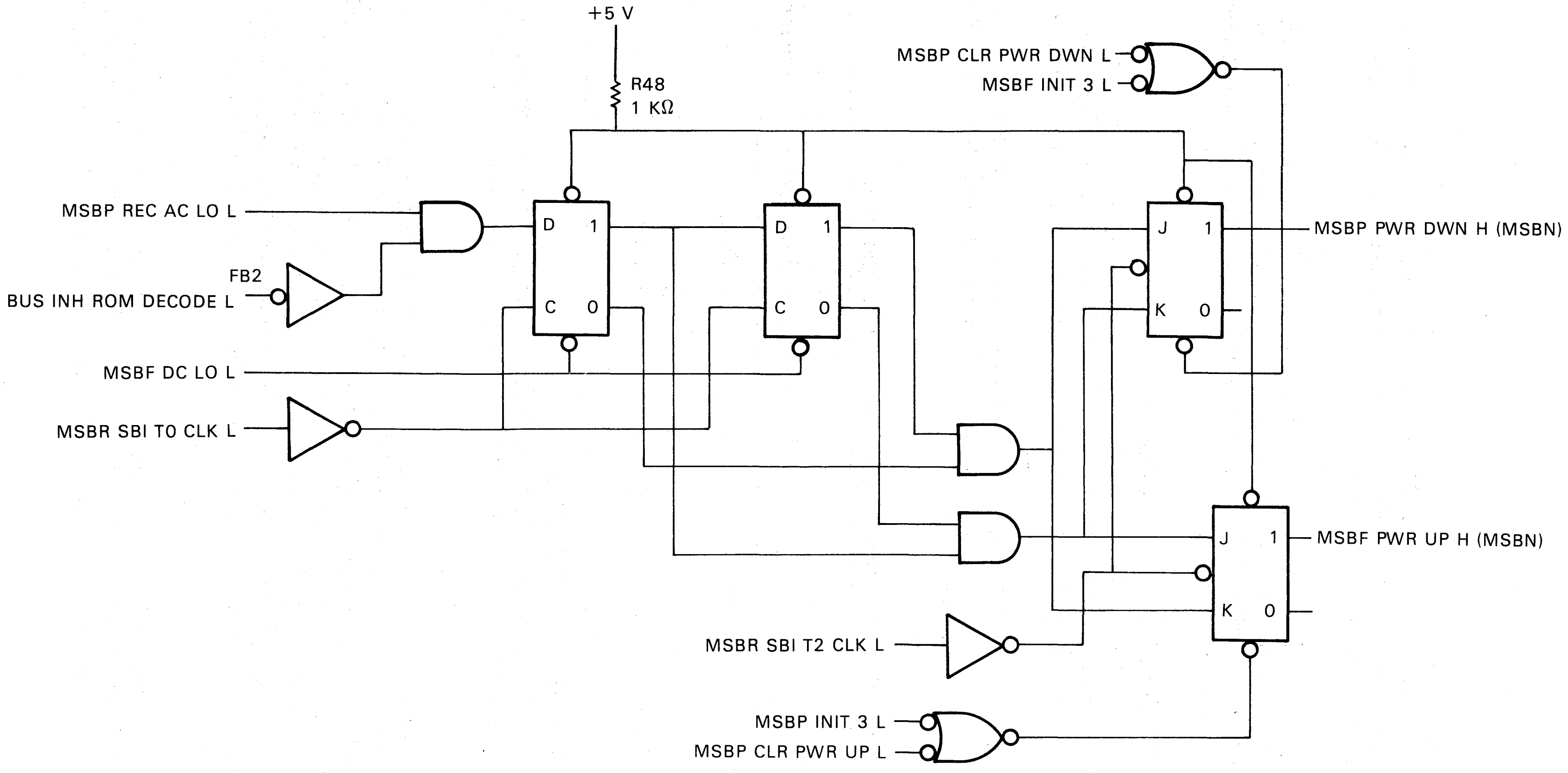


Figure 3-15 Initialization Logic



NOTE:
THE IRD JUMPER (INH ROM DECODE L) MUST BE INSERTED ON THE MSB BOARD FOR THE GENERATION OF MSBP PWR DWN H OR MSBP PWR UP H.

TK-0651

Figure 3-16 Power Up/Down Logic

3.10 ALERT AND FAIL LOGIC

Memory controllers that contain a bootstrap ROM are system critical and must assert FAIL when an AC LO occurs. A dedicated jumper on the MSB board must be inserted for this to occur. This jumper connects MSBP EN FAIL L to BUS SBI FAIL L.

Memory controllers that do not contain a bootstrap ROM assert ALERT when an AC LO or DC LO occurs. The proper power-up or power-down status bit in configuration register A is also set (Paragraph 3.5.1). Note that the IRD (Inhibit ROM Decode) jumper must be inserted. Controllers without a bootstrap ROM do not assert FAIL.

3.11 STARTING ADDRESS SELECTION LOGIC

The starting address of the MS780 controller is selected via two jumpers on the MCN board. The conditions of these jumpers are encoded to generate MCNP ST ADR (21:18) H. These signals are used for address decoding (Paragraph 2.2.4.1). The jumper interpretation is given in Table 3-4.

As described in Paragraph 3.5.1, the power-up and power-down signals can be cleared by writing a one into the respective bit positions of configuration register A. This generates MSBP CLR PWR UP L or MSBP CLR PWR DWN L to clear the signal.

Table 3-4 Starting Address Jumper Configuration

Controller Starting Address	O	4 Megabytes	8 Megabytes	12 Megabytes
BUS STADR JMPR 00 L (Pin AU2)	Open	Gnd	Open	Gnd
BUS STADR JMPR 01 L (Pin AT2)	Open	Open	Gnd	Gnd

3.12 MEMORY CYCLE TIMING SUMMARY

This paragraph summarizes the typical timing relationships of various signals. The write masked and extended read cycles are discussed as examples of typical timing. The array timing is discussed in Paragraph 3.13.4.

Figure 3-17 illustrates the typical timing of a write masked cycle. To begin, assume the file is empty. Once the valid memory command and address are decoded, MSBK ADV WR CNT H is generated to advance the write-file counter. Assuming memory is not busy (MCND MEM T BSY L high), MSBK CLK FL TO CNTRL H is generated and the command and address are removed from the file. MSBL RD-WR 0 L is generated to indicate the file is again empty. With MSBK LATCH FL DAT H going low, the address and command are placed on the file bus and MSBK ADR ON BUS L is asserted. This indicates a command is included in the information and MSBK RD CMD TO CNTRL L is generated to enable it to cycle decode logic.

The write data is loaded and removed from the file in a similar manner as seen in the diagram. This time, however, MSBK DAT ON BUS L is generated instead of ADR ON BUS. With the cycle decode indicating the operation is a write masked, MSBK MEM GO L is asserted at this time to start the memory cycle.

As illustrated in Figure 3-18, the command and address of a valid extended read request are removed from the file in a similar manner. With the generation of MSBK ADR ON BUS L, MSBK RD CMD TO CNTRL is asserted to enable cycle decoding. Likewise, being a read command, no additional information must be removed from the file and MSBK MEM GO L is generated.

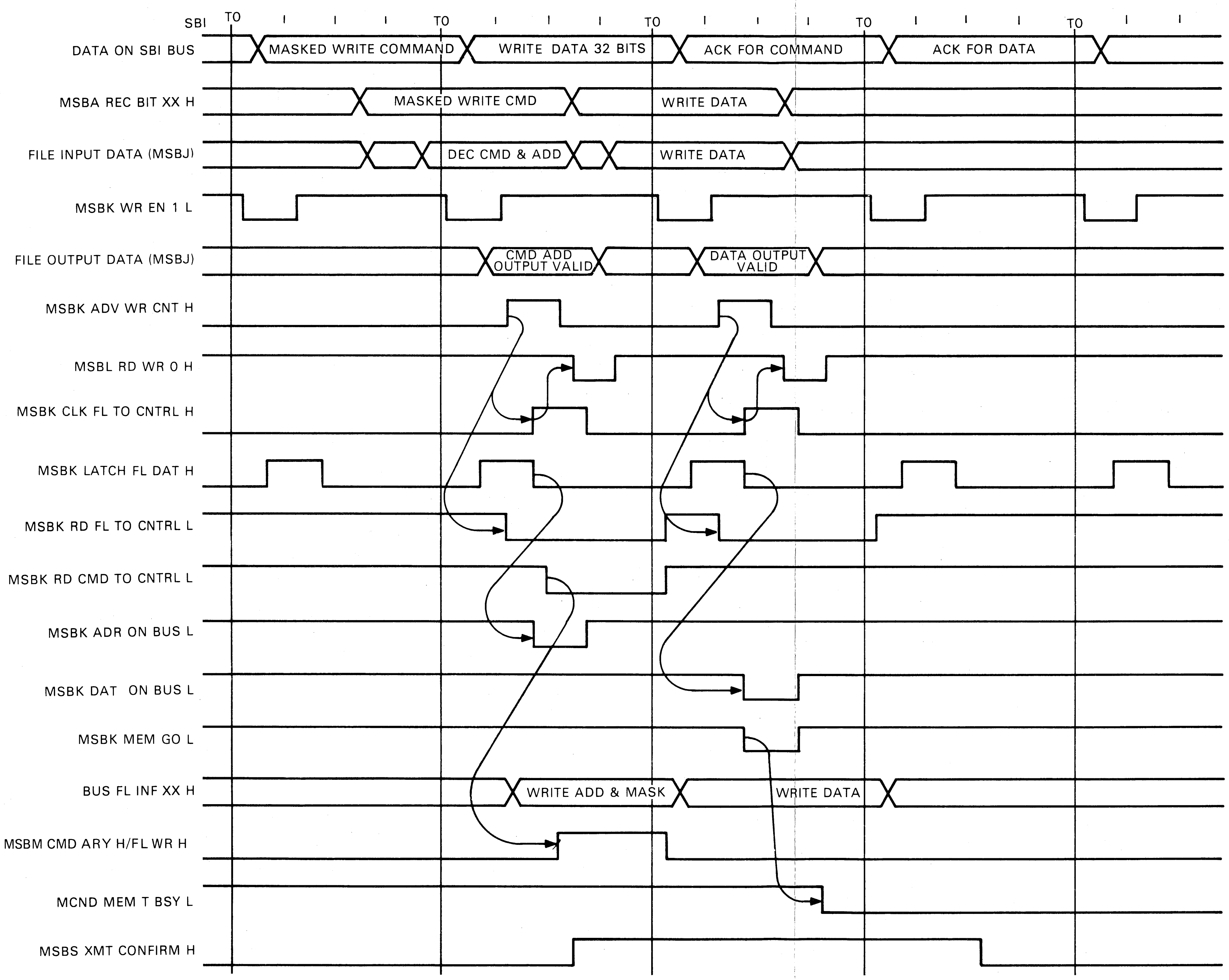


Figure 3-17 Write Masked Timing

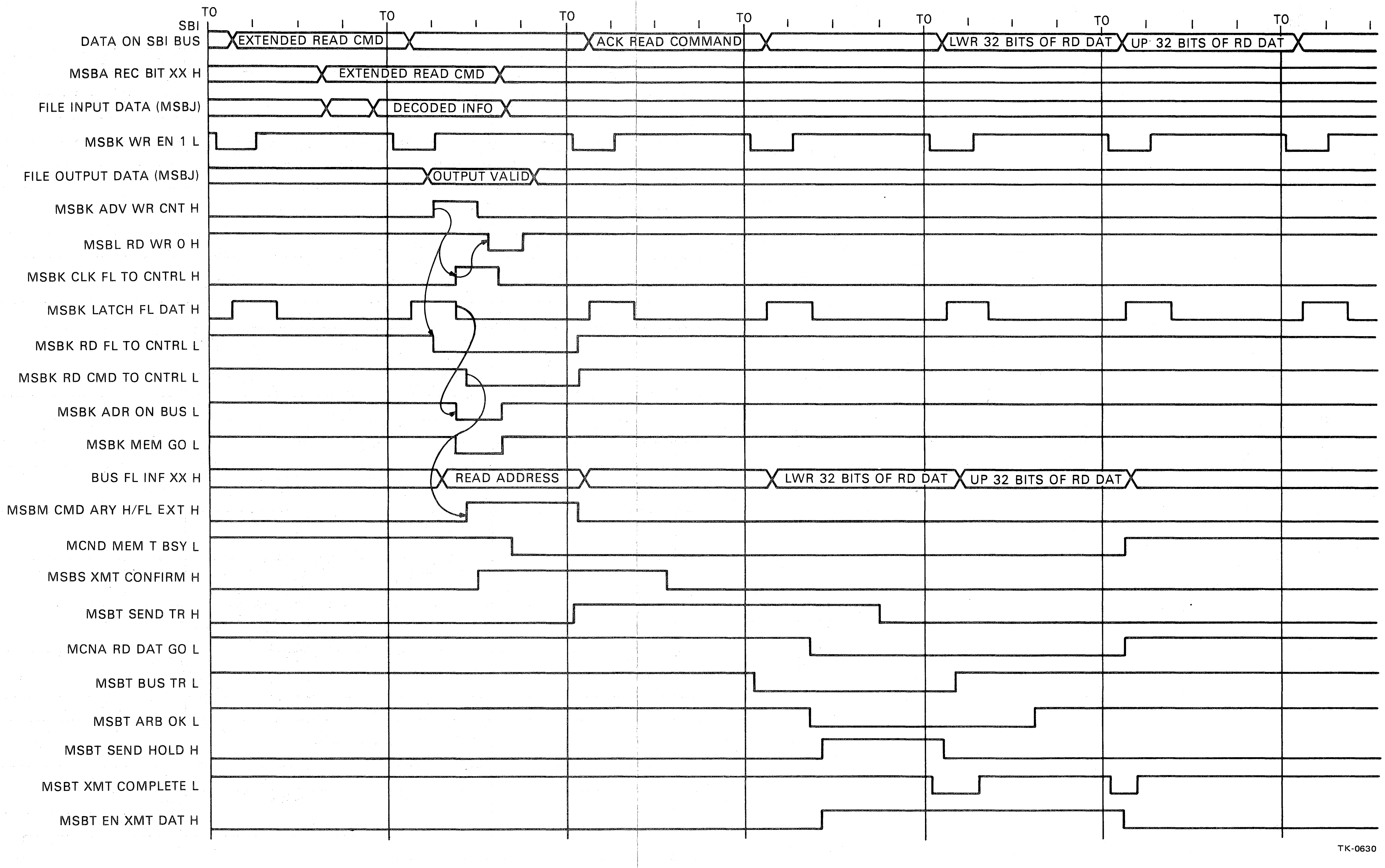


Figure 3-18 Extended Read Timing

At T0 of the following cycle, MSBT SEND TR H is asserted to arbitrate for the transmission of the requested data. Assuming no other nexus with higher priority has asserted its TR, MSBT ARB OK L is received at T2. Coincidentally, MSBT SEND HOLD H is also generated to assert HOLD on the SBI at the next T0. (HOLD is asserted because two data transmissions will be necessary.) At T0 MSBT XMT COMPLETE L is generated to indicate the transmission of the first longword. With MSBT EN XMT DAT H still asserted at the following T0, XMT COMPLETE is asserted again for the transfer of the second longword. (Refer to Paragraph 3.7 for a description of the logic.)

3.13 ARRAY BOARD DESCRIPTION

3.13.1 Array Board Organization

The 8K memory locations on each array board are divided into two groups called banks. Figure 3-19 illustrates the array board organization. As seen in this figure, array address bits (15:01) are routed to both banks on the selected array board. One bit, the bank select bit, is used to enable the appropriate bank for reading or writing. The remaining bits are used to select one of 4K locations in the bank. The particular bits used for these functions are shown at the bottom of Figure 3-19. Note that selection of the high or low longword is provided by the controller and determined by REC BIT 00.

3.13.2 Memory Chip Internal Organization

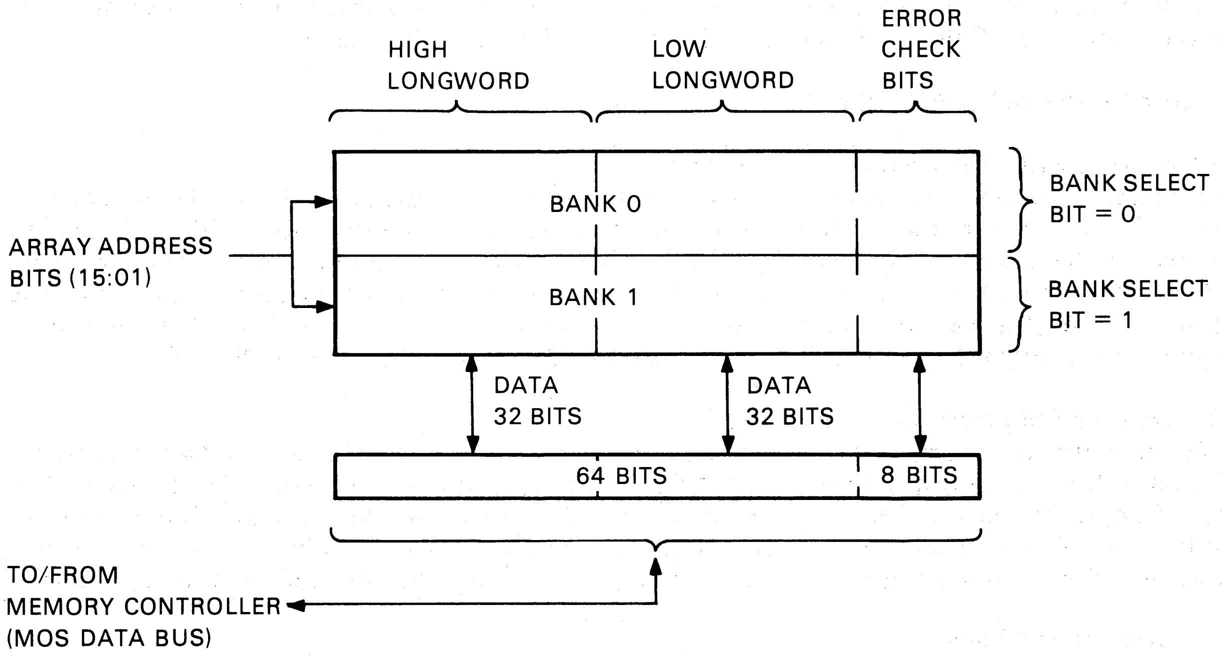
Each 4K memory chip is organized internally as a 64 × 64 matrix with 64 rows and 64 columns. Six address bits select 1 of 64 rows and six select 1 of 64 columns. In this way a single cell or bit location is selected. To optimize the number of pins required for address selection, the row and column addresses are multiplexed. When the Row Address Strobe (RAS) input is enabled, the 6-bit address selects a row. Likewise, when the Column Address Strobe (CAS) input is enabled, the 6-bit address selects a column.

3.13.3 Array Board Logic

Figure 3-20 is a simplified diagram of the 4K chip array board logic. For a read or write to the array, the address in the controller is presented to array board drivers. The output of these address drivers is connected to the row/column multiplexer. The row/column multiplexer provides selection of the row or column bits for input to the MOS chips. The row and column addresses are multiplexed to reduce the number of address inputs on the chips. This makes it possible to address one of 4K locations of the chip using only six pin inputs. With select signal MAYB MUX SEL high, address bits (12:07) are selected to generate MAYB MUX A(5:0). In like manner, MAYB MUX SEL is low to select address bits (06:01). MUX SEL is set by the controller at the appropriate times. (Refer to Paragraph 3.13.4 for timing details.)

As mentioned previously, a dedicated address bit (bit 13 for 4K chips) selects bank 1 or 0 for the operation. MAYA ENRT H and MAYA ENLT H are generated to enable the address bits to bank 1 of the selected array board. Likewise, MAYA ENRB H and MAYA ENLB H are generated to enable the address bits to bank 0. Both pairs of signals are generated during refresh and initialization cycles.

It may be difficult to understand the address signal designation. As an aid to reading the engineering print set, Figure 3-21 explains the signal nomenclature. Note some signal designations are assigned according to the physical location of the associated chips on the board.

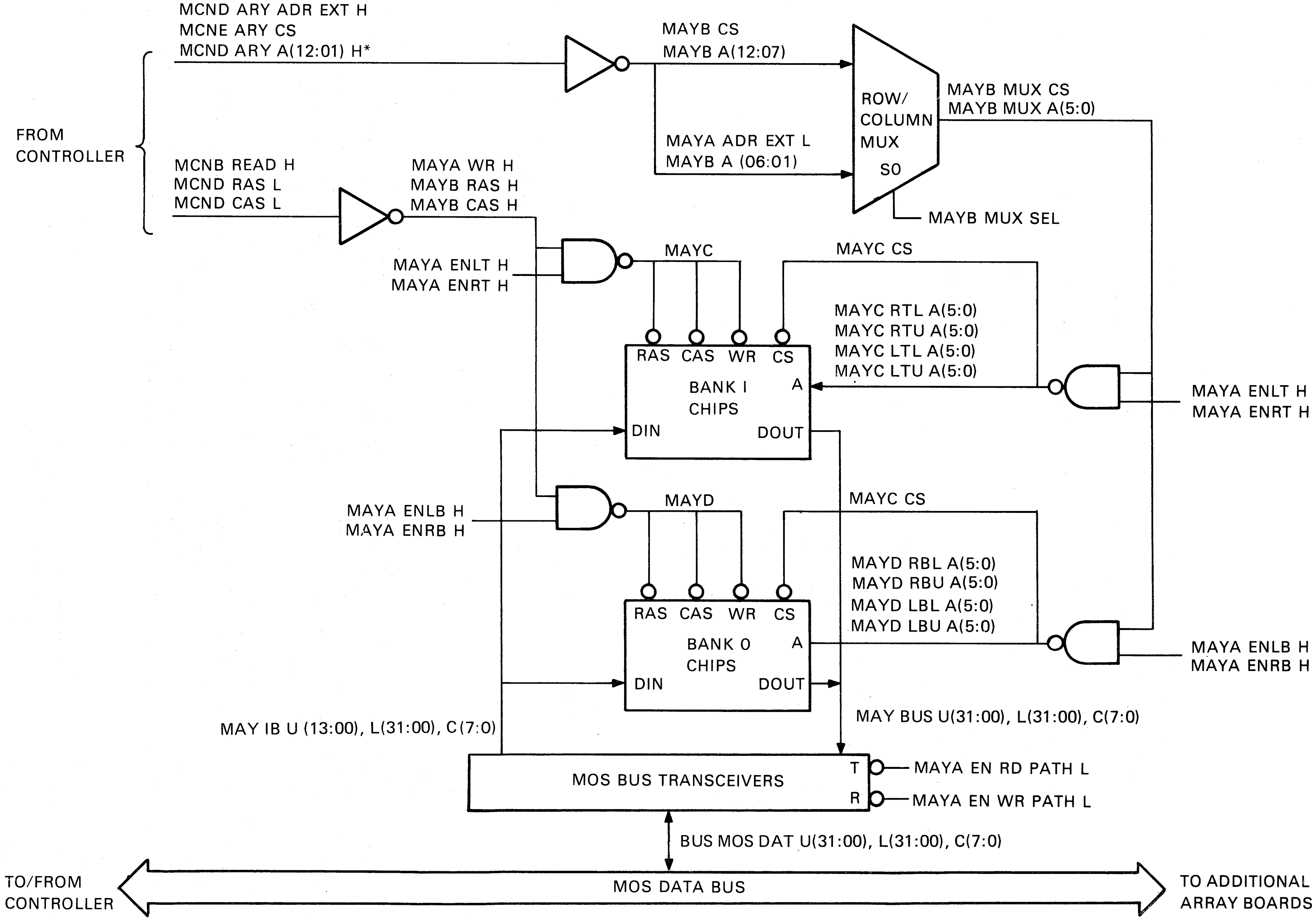


CHIP SIZE	BANK SELECT	BANK ADDRESS
4K	REC BIT 13	REC BIT 12:01
16K	REC BIT 15	REC BIT 14:01

NOTE:
IF 4K CHIPS ARE USED, REC BITS 15 AND 14 ARE IGNORED.

TK-0641

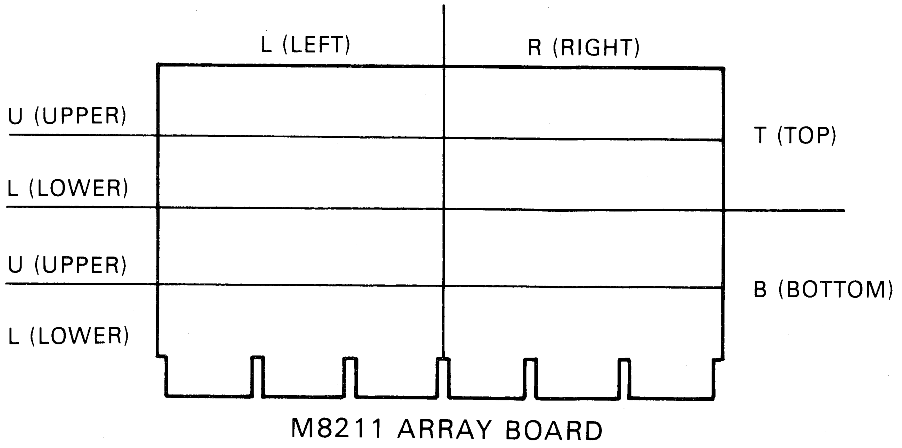
Figure 3-19 Array Board Organization



*A(15:13) ARE USED FOR 16K CHIP APPLICATION ONLY

TK-0642

Figure 3-20 Array Address and Data Logic



MNEMONIC	MEANING
LTL	LEFT, TOP, LOWER
LTU	LEFT, TOP, UPPER
RTL	RIGHT, TOP, LOWER
RTU	RIGHT, TOP, UPPER
LBL	LEFT, BOTTOM, LOWER
LBU	LEFT, BOTTOM, UPPER
RBL	RIGHT, BOTTOM, LOWER
RBU	RIGHT, BOTTOM, UPPER

NOTE:
TOP AND BOTTOM ALSO REFERS TO TOP AND BOTTOM BANK. RIGHT AND LEFT ALSO REFERS TO RIGHT AND LEFT LONGWORD.

TK-0622

Figure 3-21 Array Signal Nomenclature

The results of cycle decode are input to drivers on the array boards. These signals (MCNB READ H and MCND ARY ADR EXT H) generate MAYA EN WR PATH L or MAYA EN RD PATH L to enable the MOS bus transceivers appropriately (Figure 3-20). For reads, data from the MOS bus is presented to the data inputs of the array chips as MAY IB U(31:00), L(31:00). Similarly for writes, the data outputs of the MOS chips [BUS U(31:00), L(31:00)] are enabled to the MOS data bus.

Each MOS chip contains control inputs. The chip select input (CS) is used to enable the appropriate chips of the selected bank during a read or write cycle. The CS input is disabled during a refresh cycle to inhibit the input and output circuitry of each chip. The row address select (RAS), column address select (CAS), and write enable signals are likewise only enabled to the MOS chips of the selected bank. These signals are generated by the controller at the appropriate times. (Refer to Paragraph 3.13.4 for timing.)

3.13.4 Array Initialization Cycle

Each array location must be addressed and loaded with an initial value immediately following power-up to ensure valid ECC codes. This is referred to as an initialization cycle and is automatically accomplished by logic on the MCN board.

As seen in Figure 3-22, MCNK INZ CLR L is generated to start an initialization cycle when memory is powered up. Note this does not include recovery from battery backup. During a power-up, the pulse MCNK INZ CLR L is generated as soon as the +5 V battery backup source is produced. The duration of INZ CLR is dictated by the time it takes to charge the capacitors shown in Figure 3-22. The generation of MCNK INNZ CLR L causes the assertion of MCNK INZ CYC EN L. As a result, MCNC INIT H is sent to the arrays to enable all boards. This is done because all array boards must be written simultaneously. Likewise, MCNC WR H is generated to begin the write. During the write, each memory location is loaded with all 1s.

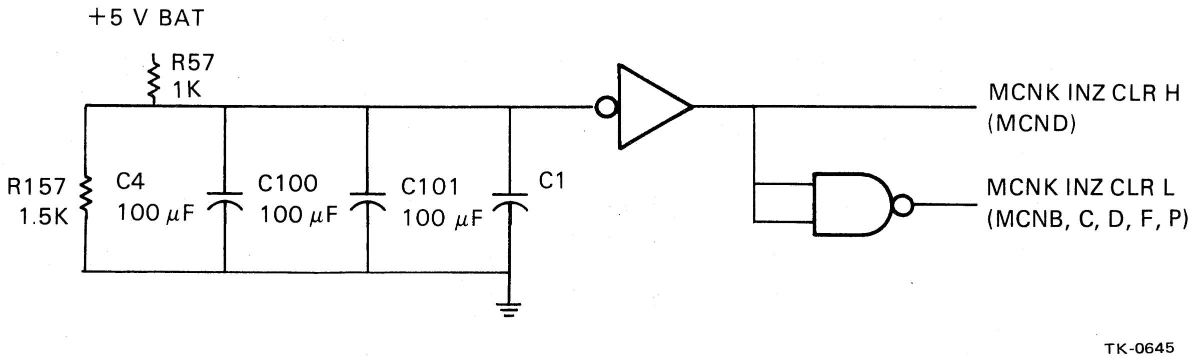


Figure 3-22 Start Initialization Cycle Logic

Each array location is addressed during initialization by the refresh address counter. The counter is initially cleared by MCNK INZ CLR L and then incremented each time MCNF REF CLK L is asserted to generate each array address. (MCNF REF CLK L is the oscillating output of the refresh clock.) When the counter has addressed every location, MCNF INZ GO L goes high. This disables the counter and generates MCNF INZ EOC L at the following T1 to end the cycle.

NOTE
The memory controller is capable of executing I/O read requests during an initialization cycle.

3.13.5 Array Timing Summary

This paragraph summarizes the typical timing relationships of various array signals. The write masked and extended read cycles are discussed as examples of typical timing.

Figure 3-23 illustrates the typical array timing that occurs during a write masked cycle. As mentioned previously, the memory cycle begins with the assertion of MSBJ MEM GO L. MEM GO sets off a timing chain in the array that is asynchronous to the SBI clock. With the row address at the input of the array chips, MCND RAS L is generated about 100 ns after MEM GO to latch the row address bits.

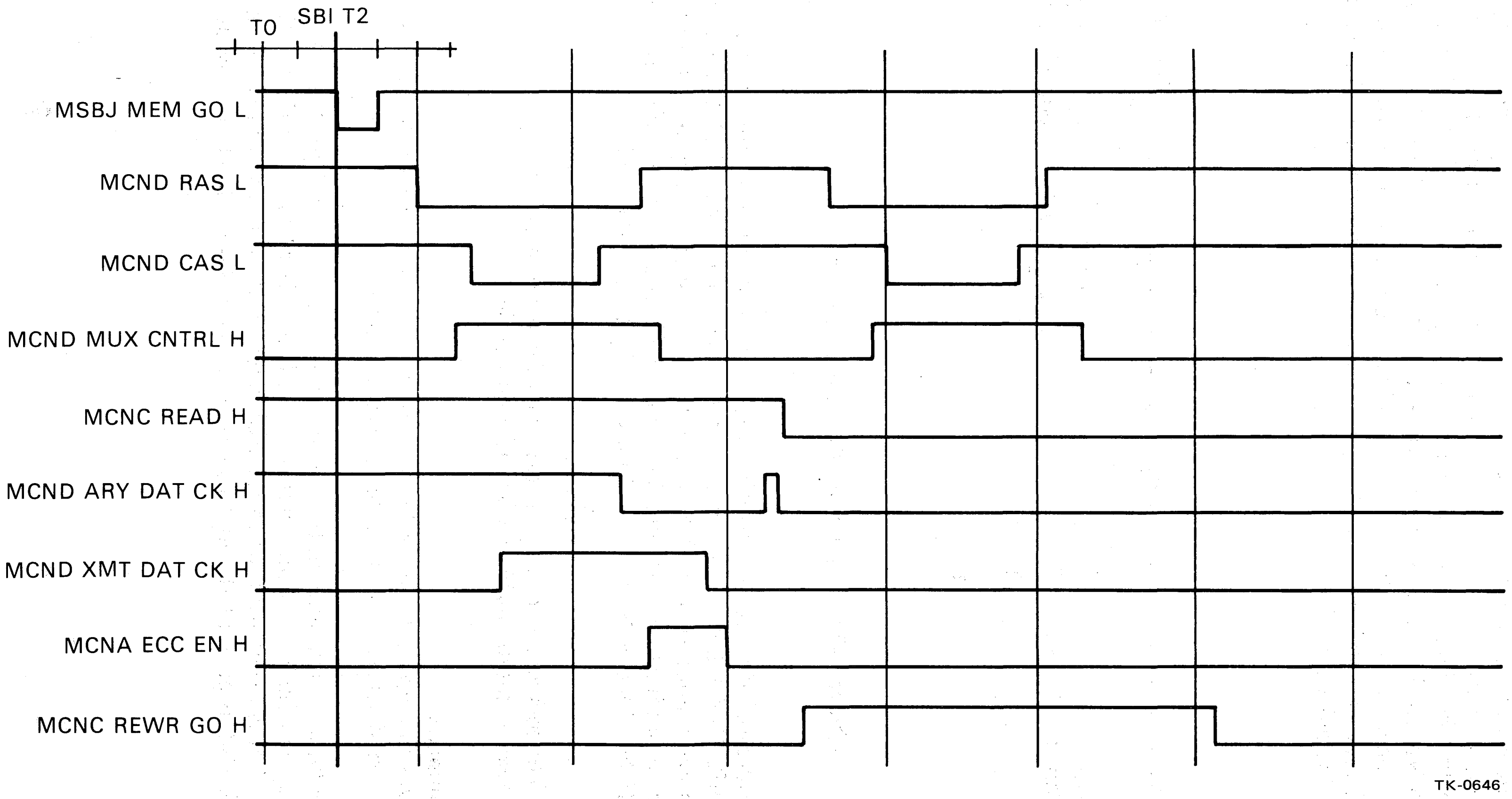


Figure 3-23 Array Timing During a Write Masked

(See Paragraph 3.13.1 for a description of the array organization.) MCND MUX CNTRL H is then asserted to select the column address bits for input to the chips. With MUX CNTRL high, MCND CAS L is asserted to latch the column address bits. This enables the data and check bits of the addressed location onto the MOS bus because MCNC READ H is asserted.

With MCND ARY DAT CK H asserted, the data and check bits on the MOS bus are immediately latched by the ECC logic for an error check. The data is then returned to the MOS bus about 100 ns later by the generation of MCND XMT DAT CK H. MCND ARY DAT CK H is negated after 270 ns. MCNA ECC EN H is then generated to enable the error status to the configuration register.

After a short delay of 25 ns, MCND ARY DAT CK H is asserted again to latch the new data for the generation of check bits. The new check bits are then placed on the MOS bus alongside the new data. With the new data and check bits ready for a write to the array, the addressing sequence is repeated. This time, however, MCNC READ H is low to enable a write when MCND CAS L is asserted.

The typical array timing that occurs during an extended read cycle is illustrated in Figure 3-24. As seen in this figure, the addressing sequence is similar to that of the write masked previously described.

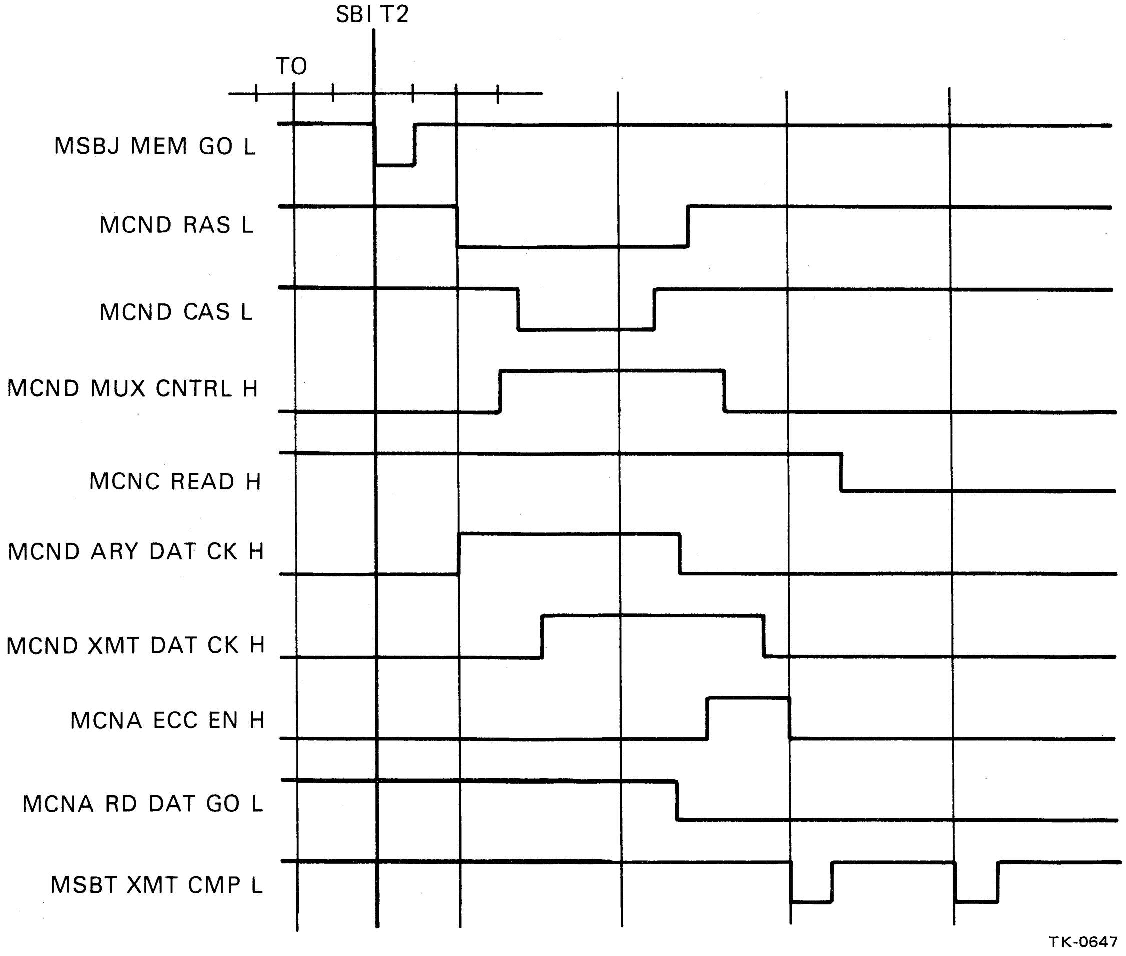


Figure 3-24 Array Timing During an Extended Read

MCND RAS L is generated about 100 ns following the assertion of MSBJ MEM GO L to latch the row address bits. MCND MUX CNTRL H is then asserted to enable the column address bits to the MOS chips. These bits are then latched by the generation of MCND CAS L. With MCNC READ H high, the contents of the addressed location are enabled onto the MOS bus.

Once on the MOS bus, with MCND ARY DAT CK H high, the data and check bits are immediately latched by the ECC logic. An error check (and possible correction) is performed on the data that is then returned to the MOS bus. MCNA ECC EN H is asserted to enable the generation of an appropriate mask for transmission with the requested data. This mask indicates the results of the error check (Paragraph 3.4).

Just before the assertion of MCNA ECC EN H, MCNA RD DATA GO L is asserted. This signal indicates the data has been error checked and is ready for transmission. At the following T0 the low longword is transmitted and MSBT XMT CMP L is asserted. MSBT XMT CMP L is again asserted one cycle later for the transmission of the upper longword.

MS780
MEMORY SYSTEM
TECHNICAL DESCRIPTION
EK-MS780-TD-001

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 *Technical Documentation Catalog*, 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
444 Whitney Street
Northboro, Ma 01532
Attention: Communications Services (NR2/M15)
Customer Services Section

Order No. EK-MS780-TD-001

— — — — — — — — — — — Fold Here — — — — — — — — — — —

— — — — — — — — Do Not Tear - Fold Here and Staple — — — — — — — —

FIRST CLASS
PERMIT NO. 33
MAYNARD, MASS.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

Postage will be paid by:

Digital Equipment Corporation
Technical Documentation Department
Maynard, Massachusetts 01754