VAX-11/780 Unibus Adaptor

Technical Description

EK-DW780-TD.001

EK-DW780-TD-001

VAX-11/780
DW780 Unibus Adaptor
Technical Description

digital equipment corporation • maynard, massachusetts

First Edition, May 1978

Printed in U.S.A.

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

CONTENTS

CONTENTS (Cont)

CONTENTS (Cont)

CONTENTS (Cont)

CONTENTS (Cont)

FIGURES

FIGURES (Cont)

FIGURES (Cont)

FIGURES (Cont)

FIGURES (Cont)

TABLES

TABLES (Cont)

CHAPTER 1
INTRODUCTION

1.1 SCOPE

This manual provides a comprehensive description of the VAX-11/780 Unibus Adaptor (UBA, DW780) on three levels: general, functional, and logical. Descriptions of the Unibus and synchronous backplane interconnect (SBI) are included. The manual will serve as a resource for appropriate branch and support level courses of the field service and manufacturing training programs and as a field reference. Table 1-1 lists related hardware documentation.

Table 1-1 Related Hardware Manuals

Title	Document	Notes
VAX-11 KA780 Central Processor Technical Description	EK-KA780-TD-PRE	In Microfiche Library
VAX-11 MS780 Memory System Technical Description	EK-MS780-TD-PRE	In Microfiche Library
PDP-11 Peripherals Handbook	EB 05961	Available in hard copy*
VAX 11/780 Architecture Handbook	EB 07466	Available in hard copy*

*These documents can be ordered from:

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

For information concerning microfiche libraries, contact:

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

1.2 THE UNIBUS ADAPTOR

The Unibus Adaptor (UBA) connects the Unibus to the synchronous backplane interconnect (SBI) in the VAX-11/780 system. Figure 1-1 shows the relation of the UBA to the VAX-11/780 system. The UBA design allows for a maximum of four UBAs per VAX-11/780 system. The UBA supports both direct memory access (DMA) and non-DMA Unibus peripherals.

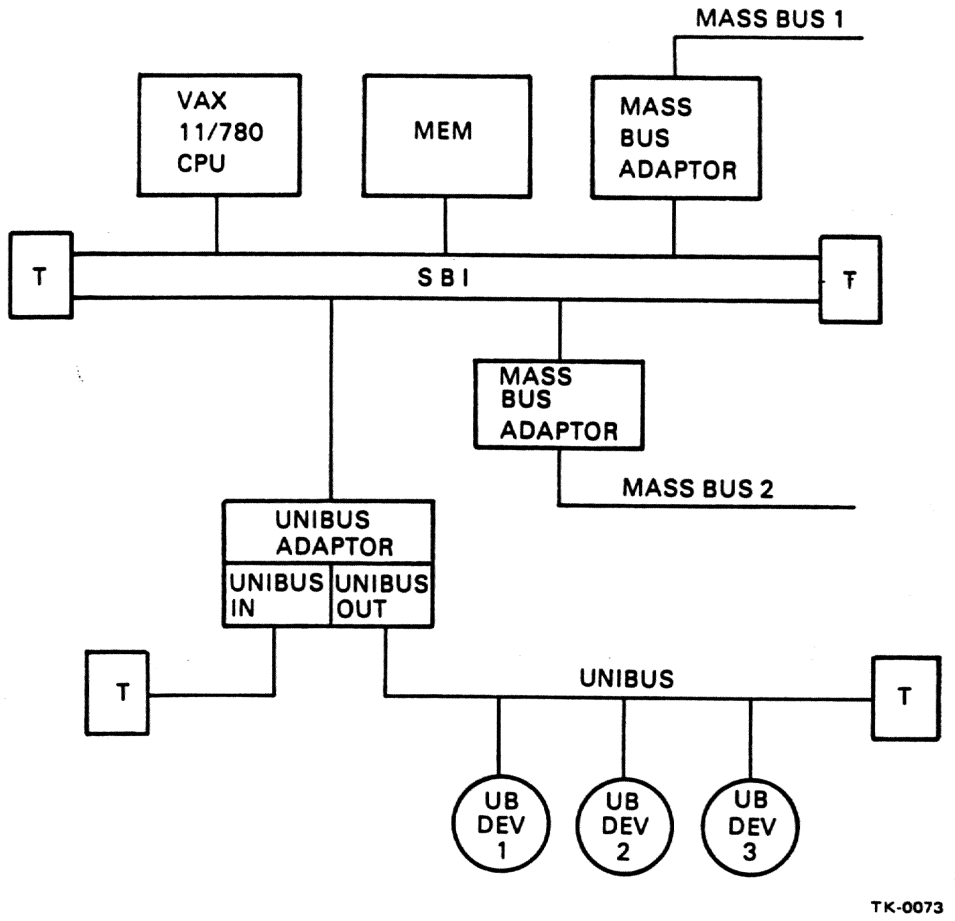


Figure 1-1 VAX-11/780 System Configuration with the UBA

1.3 SYNCHRONOUS BACKPLANE INTERCONNECT

The SBI is a bidirectional information path and communication protocol for data exchanges between the central processor (CPU), memory, and adaptors of the VAX-11/780 system. The SBI provides checked, parallel information exchanges synchronous with a common system clock.

The communications protocol allows the information path to be time multiplexed such that an arbitrary number of data exchanges may be in progress simultaneously. That is, in each clock period (or cycle) interconnect arbitration, information transfer, and transfer confirmation may occur in parallel.

SBI signals are received by data latches. All checking and subsequent decision making is based on these latched signals. Error checking logic detects single bit failures in the information path. However, multiple SBI system failures are not necessarily detected.

1.3.1 SBI Summary

Table 1-2 lists the basic SBI characteristics.

Table 1-2 Basic SBI Characteristics

Characteristic	Definition
Number of signal lines	84 including check bits
Data path width	32 bits
Physical address space	28 bits (268, 435, 456 longwords, 1,073,741,824 bytes)
SBI cycle time	200 ns
Arbitration logic	Distributed throughout the system
Maximum SBI length	3 meters
Interface chips	DEC 8646 DC 101
SBI cable properties	
Characteristic impedance	75 ± 7 ohms
Propagation delay	1.1 ± 0.1 ns/ft
Backplane properties	
Characteristic impedance	75 ohms
Propagation delay	2.0 ns/ft

1.3.2 SBI Unit Definitions

The following terms are defined for SBI-specific units and operations:

a. Nexus – a physical connection to the SBI capable of any or all of the functions described in items b–e, below.

b. Commander – a nexus that transmits command and address information.

c. Responder – a nexus that recognizes and responds to command and address information directed to it.

d. Transmitter – a nexus that drives the signal lines.

e. Receiver – a nexus that samples and examines the signal lines.

1.3.3 Definition Examples

The definition of a nexus changes with its function. For example, a CPU that issues a read type command can be considered one of three nexus types depending on the point in the information exchange.

When the CPU issues the read command to a device, it is a commander, since it is issuing command/address information. At the same time, it is a transmitter since it is driving the signal lines. When the device (responder) returns the requested data, the CPU is considered a receiver since it examines the signal lines.

In the case of a memory read exchange, the memory is the responder since it recognizes and responds to command/address information. Also, since it examines the signal lines, it is a receiver. When the memory returns the requested data by driving the signal lines, it is a transmitter.

1.3.4 Data Types

Four data types are addressable via the SBI, as follows.

- Byte – A byte is eight contiguous bits starting on an addressable byte boundary. The bits are numbered from the right, 0 through 7. A byte is specified by its address, A.
- Word – A word is two contiguous bytes starting on an arbitrary byte boundary. The bits are numbered from the right, 0 through 15. A word is specified by its address, A, the address of the byte containing bit 0.
- Longword – A longword is four contiguous bytes, starting on an arbitrary byte boundary. The bits are numbered from the right, 0 through 31. A longword is specified by its address A, the address of the byte containing bit 0.
- Quadword – A quadword is eight contiguous bytes starting on an arbitrary byte boundary. The bits are numbered from the right, 0 through 63. A quadword is specified by its address, A, the address of the byte containing bit 0.

SBI addresses refer directly to longwords.

1.3.5 Basic Functional Structure

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

a. Arbitration
b. Information
c. Confirmation
d. Interrupt
e. Control

1.3.5.1 Arbitration Group – The arbitration group sets nexus priority access to the SBI. It determines which nexus of those requesting access to the SBI in a particular cycle will perform an information transfer in the following cycle.

1.3.5.2 Information Transfer Group – The information transfer group exchanges command/addresses, mask, data, and interrupt summary information and interprocessor information. Each exchange consists of one, two, or three information transfers.

The mask field consists of four bits, each of which corresponds to 1 byte in the 32-bit wide data path. When the commander asserts one or more mask bits with command/address, the corresponding byte or bytes in the data field will be asserted during the data transmission that follows. SBI addresses refer directly to longwords. The byte mask is used to access a byte or bytes within a longword.

For write type commands, the commander uses two or three successive SBI cycles. The number of successive cycles required depends on whether one or two data longwords are to be written in the exchange. In a write masked transfer, the commander transmits the command/address (and mask) in the first cycle, and data in the second cycle. In an extended write masked transfer, the commander transmits command/address (and the mask for the first data longword) in the first cycle, data longword one (and the mask for the second data word) in the second cycle, and data longword two in the third cycle, as shown in Figure 1-2.

Read type commands are also initiated with a command/address transmitted from the commander. However, since data emanates from the responder, the requested data may be delayed by the characteristic access time of the responder. In a read masked transfer, one data longword will be transmitted, with some or all of the four bytes enabled, in accordance with the mask received from the commander. The responder asserts two data longwords (unmasked) on successive SBI cycles, in answer to an extended read command.

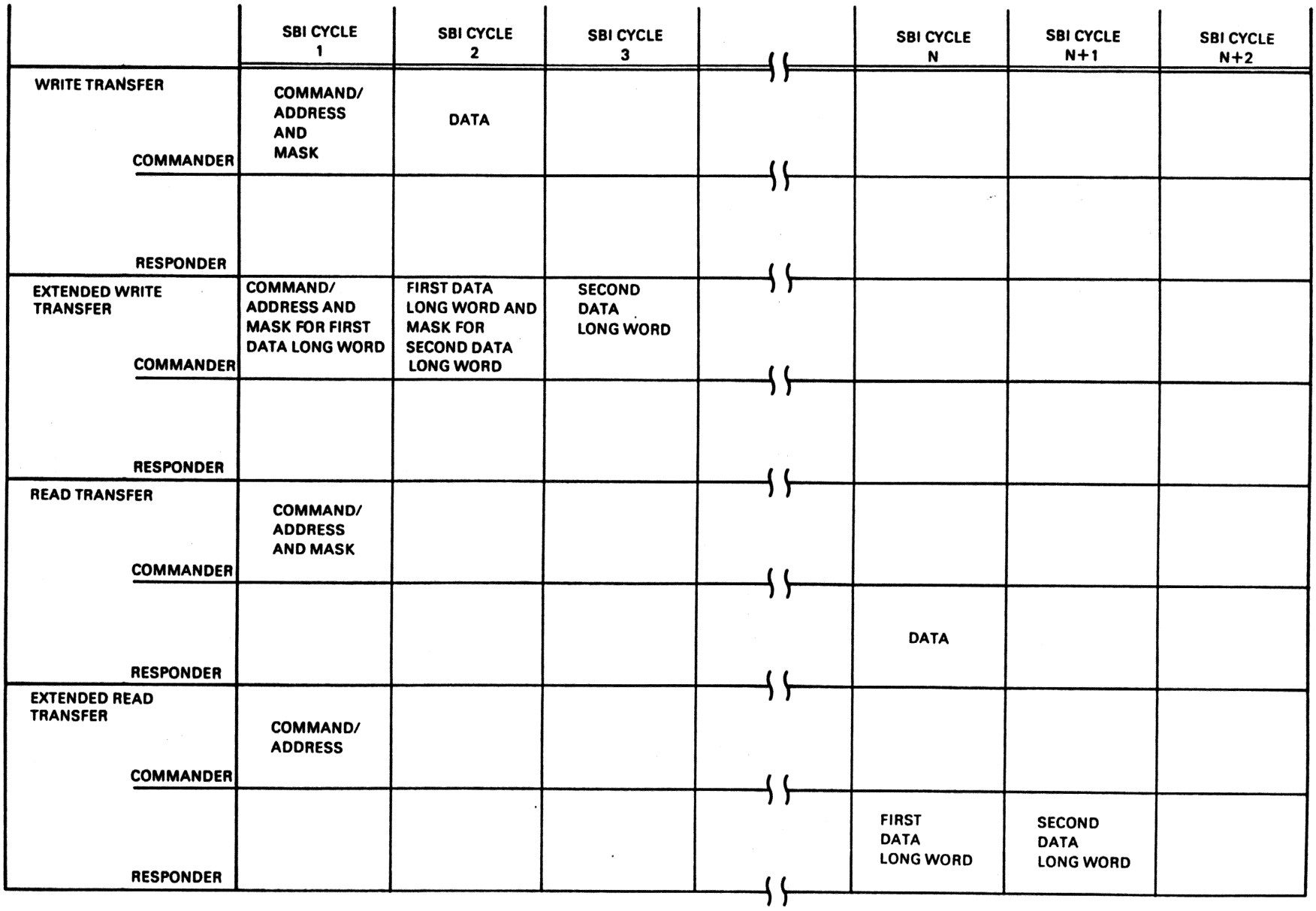


Figure 1-2 Data Transfer Sequences on the SBI

The interlock read masked and interlock write masked functions are similar to the read masked and write masked functions, except that they are always paired by the commander and access the same memory location. The responder will answer any intervening interlock read commands with a BSY confirmation.

An interrupt summary exchange takes place in response to a device-generated interrupt to the CPU. The CPU initiates the exchange with an interrupt summary read transfer. The exchange is completed two cycles later with an interrupt summary response transfer containing the interrupt vector from the interrupting SBI nexus.

1.3.5.3 Response Group – The response group provides a transfer and error information path between nexus. Confirmation informs the transmitter as to whether the information transfer was correctly received and, in the case of a command/address transfer, whether the receiver could process the command. The error path is a cumulative error indicator of protocol or information path malfunction.

Each command/address or information transfer is confirmed by the responder (or receiver) two cycles after transmission by the commander. During a write type exchange, command/address and data transfers are confirmed by the responder. During a read type exchange, the command/address transfer is confirmed by the responder, and the reception of read data is confirmed by the commander.

Interrupt summary transfers are not confirmed except when an information transfer parity error is detected, in which case an SBI Fault signal is asserted.

1.3.5.4 Interrupt Request Group – The interrupt request group provides a path for nexus to interrupt the CPU at one of the four request levels in order to service a condition requiring processor intervention. In addition, the group provides a path for those nexus that do not have the normal interrupt mechanism to interrupt the CPU when changes in power or operating conditions occur.

1.3.5.5 Control Group – The control group provides a path to synchronize system activity and provides specialized system communication. The group includes the system clock, which provides the universal time base for any nexus connected to the SBI. The group also provides initialization, power fail, and restart functions for the system. In addition, an interlock is provided for coordinating between memories to ensure access to shared structures.

1.4 UNIBUS SUMMARY

The Unified Bus (Unibus) connects PDP-11 peripheral devices to the UBA. Address, data, and control information are transferred over the 56 lines of the Unibus. The data path is 16-bits wide, and the address field contains 18 bits. The communication process is identical for every device on the Unibus; a master-slave relationship controls the communication between any two devices on the Unibus. Each control signal issued by a master device must be acknowledged by a slave device to complete the transfer. The UBA must arbitrate for control of the bus, like any other Unibus device.

The Unibus permits an addressing structure in which control, status, and data registers for peripheral devices are directly addressed as memory locations. Therefore, all operations on these registers are performed by normal memory reference instructions.

1.5 UBA OVERVIEW

The UBA provides for access to Unibus device registers from the SBI, Unibus access to random SBI memory addresses, high speed Unibus device access to consecutive increasing memory addresses within a memory page, and Unibus interrupt fielding.

The Unibus address space is assigned as part of the SBI I/O address space. The UBA translates SBI command/addresses to Unibus command/addresses, thereby giving the software the ability to read and write Unibus device registers.

Unibus transfers to Unibus memory addresses are mapped, by the UBA, to SBI addresses on a page by page basis, thereby allowing Unibus data transfers to discontiguous pages of SBI memory.

The SBI uses a 28-bit longword addressing scheme and a 32-bit wide data path, while the Unibus uses 18 address bits and a 16-bit wide data path. The SBI cycle is synchronous, and the number of nexus on the SBI is limited to 16; while Unibus functions are asynchronous, and the maximum number of devices on the bus is comparatively large and may vary, depending on the specific configuration used.

Table 1-3 shows the action taken by the UBA in response to data transfer operations initiated by the software.

Table 1-3 UBA Functions Performed on a Software-Initiated Transfer

SBI Function	UBA Function	Unibus Function
Read Masked	SBI-Unibus read	DATI
Write Masked	SBI-Unibus write	DATO/B
Interlock Read-Interlock Write	SBI-Unibus read-modify-write	DATIP-DATO/B
Read Masked	SBI-UBA register read	
Write Masked	SBI-UBA register write	

Table 1-4 shows UBA functions performed when the UBA initiates transfers to SBI memory.

Table 1-4 UBA Initiated Transfers to SBI Memory

UBA Function	SBI Function
BDP to SBI Write	(Extended) Write Masked
Purge	(Extended) Write Masked
Prefetch	Extended Read

The UBA uses the buffered data paths (BDPs) and the direct data path (DDP) (Paragraph 1.5.1) to translate Unibus device initiated (DMA) transfers into SBI functions, as shown in Table 1-5.

Table 1-5 UBA Function Performed in Response to DMA Transfer on the Unibus

Unibus Function	UBA Function	SBI Function
		DMA-Unibus device initiates transfer
DATI	Unibus-SBI read through DDP	Read-Masked
DATIP	Unibus-SBI read through DDP to be followed by DATO/B to same location	Interlock Read Masked
DATO/B	Unibus-SBI write through DDP	Write Masked
DATI	Unibus-SBI read through BDP	Extended Read or Read Masked
DATO/B	Unibus-SBI write through BDP	(Extended) Write Masked
DATI	Unibus-SBI read with byte offset*through BDP	(Extended) Read (Masked)
DATO/B	Unibus-SBI read with byte offset*through BDP	Extended Write Masked

*The byte offset capability of the UBA enables Unibus devices that can only transfer to or from even byte boundaries to perform DMA transfers to SBI memory space beginning and ending at odd byte locations.

Table 1-6 shows the action of the UBA in response to interrupts generated on the Unibus and on the UBA itself.

Table 1-6 UBA Interrupt Functions

Unibus Function	UBA Function	SBI Function
Interrupt dialogue	Unibus I/O device interrupts CPU through the UBA	Interrupt dialogue
	UBA internal event causes interrupt to CPU	Interrupt dialogue

In addition, the UBA monitors the power status signals on the SBI and the Unibus and performs Unibus and UBA power up and power down sequences and the UBA initialization sequence, as appropriate.

1.5.1 Data Paths

The UBA can transfer data between a Unibus device and SBI memory through any one of 16 data paths on a DMA transfer.

The UBA provides a DDP to allow Unibus transfers to random SBI addresses. Each Unibus transfer through the DDP is mapped directly to an SBI transfer, thereby allowing only one word of information to be transferred during an SBI cycle.

The UBA provides 15 BDPs, each of which allows a block transfer device on the Unibus (a device that transfers to consecutive increasing addresses) access to the SBI in a more efficient manner than that offered by the DDP. The BDPs store data for Unibus devices such that four Unibus transfers are performed for each SBI transfer, thereby making more efficient use of the SBI and memory. Using the BDPs, the UBA can support high speed DMA block transfer devices such as the RK06 disk subsystem and the DMC-11.

1.5.2 Address Mapping

During a DMA transfer, when a Unibus device asserts a memory address on the Unibus, the UBA maps that address to a location within SBI address space. The UBA divides the Unibus memory address space into 496 pages. Each page contains 512 bytes and the lower nine Unibus address bits [UA(8:0)] define the bytes within the page.

The upper nine address bits select one of the 496 (10) map registers contained on the UBA. Figure 1-3 shows the relation of the two address spaces and the function of the map registers in the Unibus address to SBI address translation process.

The UBA map registers are used to map any Unibus DMA transfer to any SBI address on a page by page basis. A Unibus page address is determined by Unibus address bits (17:09) and it points to the map register corresponding to that page. The map register selected contains the 21-bit SBI page address which this transfer will access. The low-order Unibus address bits and the Unibus control bits, C1 and C0, identify both the byte or word in physical memory that is to be accessed and the SBI function to be performed.

The map registers are accessible to the software and are loaded by the software prior to starting the Unibus transfer.

1.5.3 SBI Address Space Controlled by the UBA

The SBI address space controlled by the UBA falls into two categories: the registers internal to the UBA and Unibus address space, as shown in Figure 1-4.

Included in the category of internal registers are the map registers and several registers used for controling UBA processes and alerting the CPU to specific conditions on the UBA that require attention. Address decoders on the UBA determine whether an address asserted on the SBI is within the space controlled by the UBA, and. if so, whether it is an internal register address or a Unibus address.

When the software initiates a data transfer to a Unibus I/O device register, it asserts a 28-bit address and mask on the SBI. The UBA will recognize this address as a location within the Unibus address space and pass the corresponding 18-bit Unibus address on to the Unibus address lines.

1.5.4 Interrupt Functions

The UBA fields the interrupt requests generated by the Unibus peripheral devices. Bus requests (BRs) from the Unibus can be passed on to the VAX-11/780.

The UBA contains the logic necessary to issue bus grants (BGs) and to accept the interrupt vector from the Unibus device. In addition, the UBA logic can issue a non-processor grant (NPG) in response to a non-processor request (NPR).

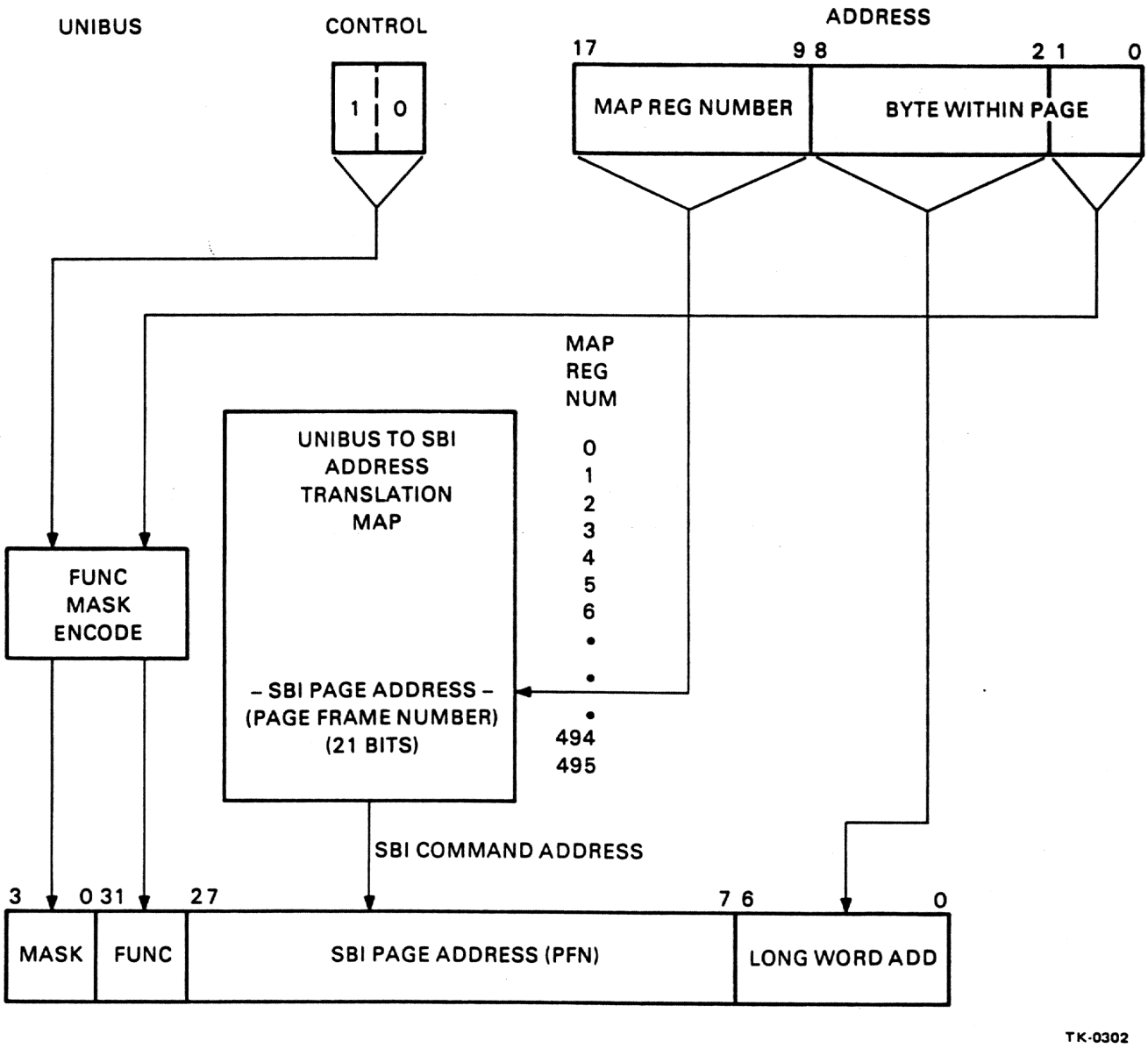


Figure 1-3 Unibus to SBI Address Translation

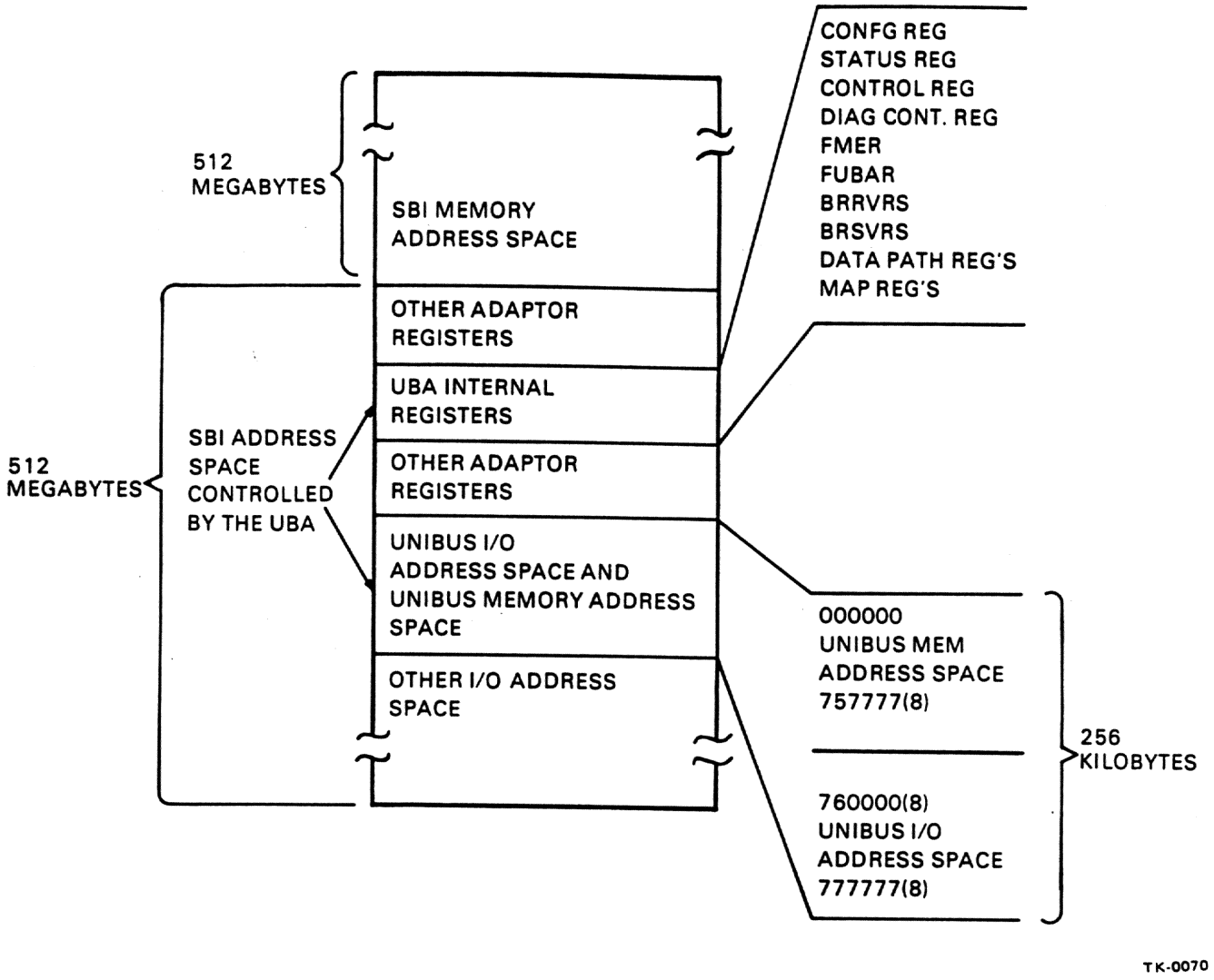


Figure 1-4 SBI Address Space

1.6 BASIC HARDWARE DESCRIPTION

The UBA consists of a six slot backplane (four slots for the UBA modules, one for the UBA terminator, and one for the peripheral Unibus) and the following modules as shown in Figure 1-5.

Slot 1	M8270	USI	Extended hex module (SBI interface)
Slot 2	M8271	UCB	Extended hex module (Control board)
Slot 3	M8272	UMD	Extended hex module (Maps and data paths)
Slot 4	M8273	UAI	Extended hex module (Unibus address and interrupt)
Slot 5	M9042	UPC	Double height by 4-1/2 inch module (Unibus paddle card)
Slot 6	M9044	UBT	Double height by 4-1/2 inch module (UBA Unibus terminator)

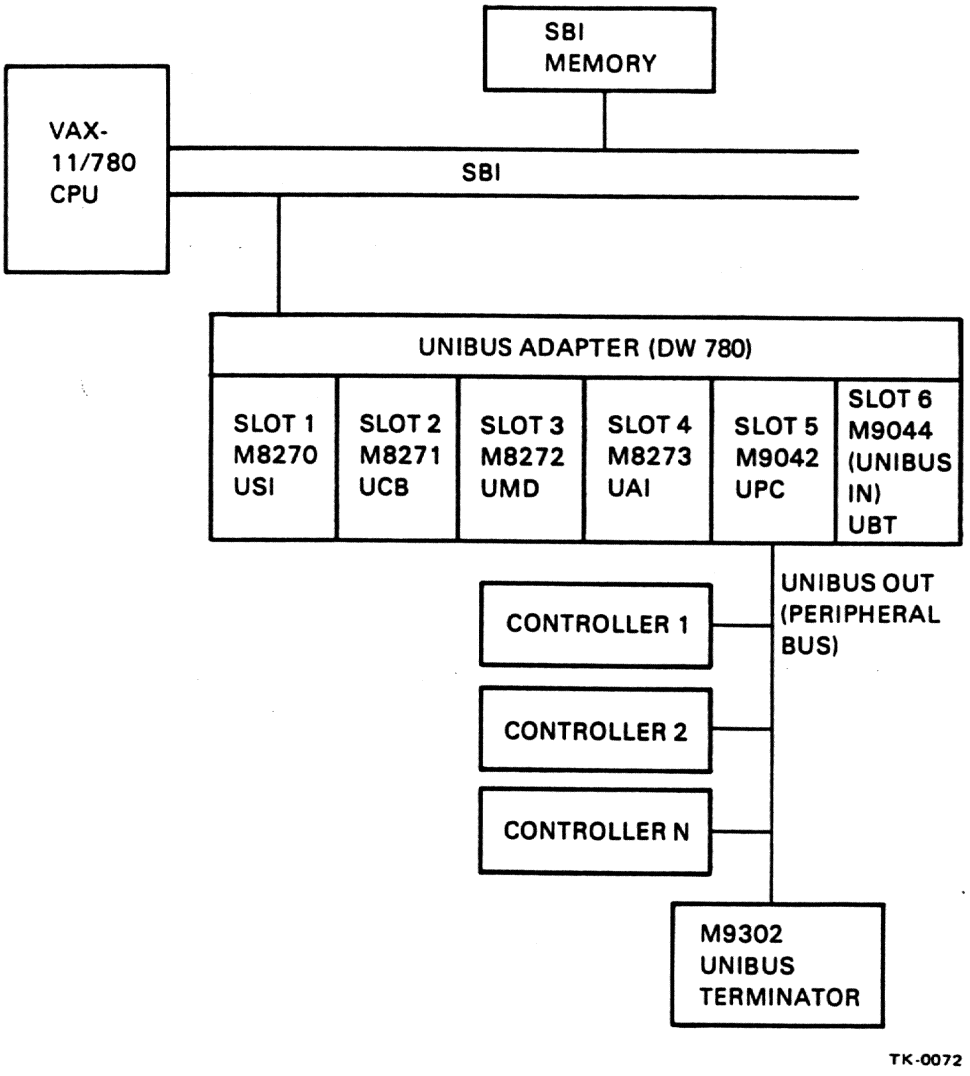


Figure 1-5 Unibus Subsystem Configuration

The Unibus peripheral string connects with slot 5 via the Unibus paddle card (UPC) with three BC05L cables. The Unibus connects with the first peripheral Unibus backplane via the three BC05L cables and an M9014 module (BC05L to Unibus connector). The far end of the Unibus is terminated with the M9302 Unibus terminator with SACK turnaround logic. A description of the four UBA modules follows.

1.6.1 UMD Module (UBA Map and Data Path)

The sixteen data paths and the associated random access memories (RAMs), shifters, and multiplexers make up one half of the UMD module. The four bus request receive vector registers (BRRVRs) and the four buffer select verification registers (BRSVRs) are located in the buffered data path RAMs.

This module also includes the 496 map registers, the failed map entry register (FMER), the 4 × 32 bit data file that stores incoming SBI longwords, and the Unibus data transceivers.

1.6.2 UCB Module (UBA Control Board)

The UCB module contains the microsequencer for the UBA. The control store ROM space is 44 (10) bits wide and 512 (10) words deep. The microroutines blasted into the ROMs control the initialization sequence, SBI attention, Unibus attention, prefetch attention, and DMA attention processes performed by the UBA. The logic on the module includes decoder circuits for the ROM fields and branch logic that enables the microprocessor to respond to various signals from the SBI, the Unibus, and the UBA internal registers. This module also contains the SBI transmit logic.

1.6.3 UAI Module (Unibus Address and Interrupt)

The UAI module functions as the interface between the UBA and the Unibus. It includes bus transceivers, interrupt fielding logic, and control and arbitration logic for the Unibus. In addition, the UAI module contains the control and status registers, the failed Unibus address register (FUBAR), timeout logic, Unibus attention select logic, and power fail and initialization logic for the Unibus and the UBA.

1.6.4 USI Module (UBA SBI Interface)

The USI module interfaces between the UBA and the SBI. It contains the SBI arbitration logic, bus transceivers, SBI parity logic, and decoders and encoders for the various SBI signal groups. The USI module also includes the logic for the configuration register and the diagnostic control register.

Figure 1-6 shows the functional relationship of the four UBA modules.

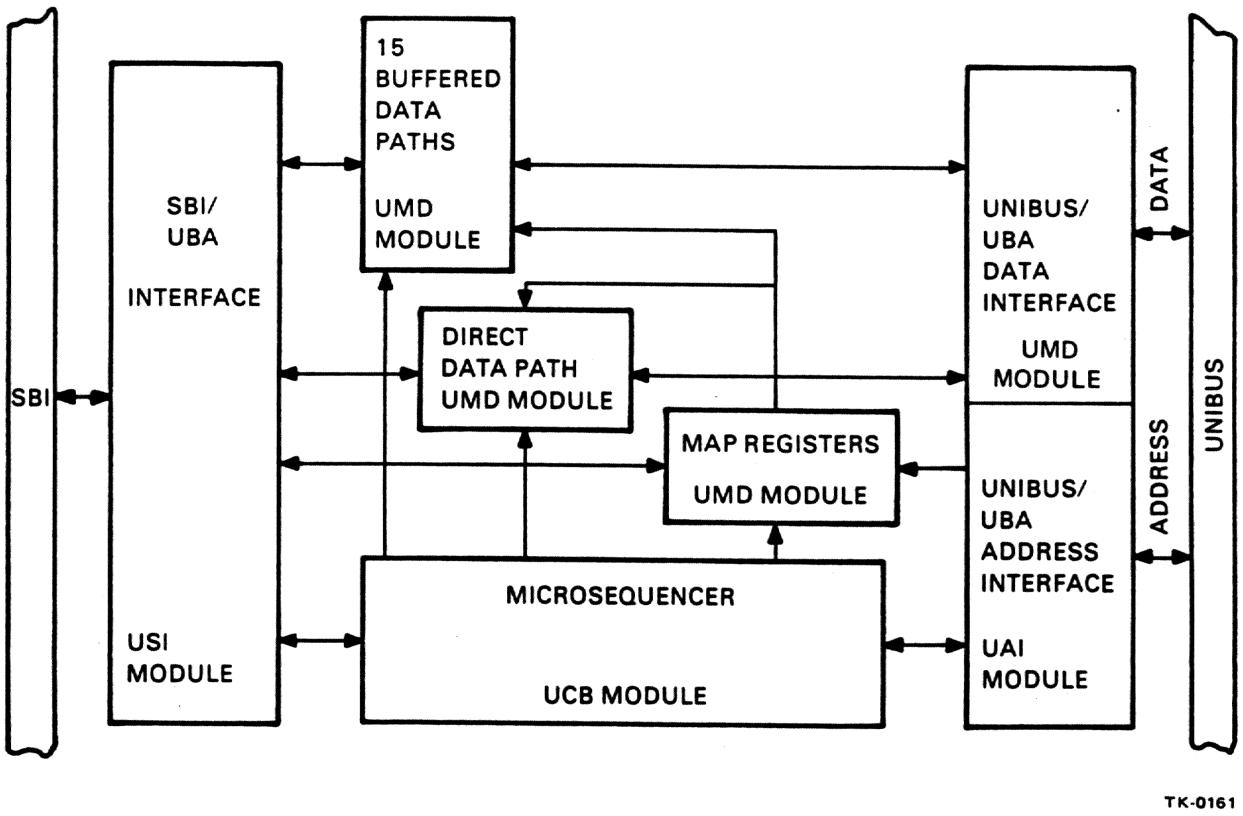


Figure 1-6 UBA, Simplified Block Diagram

1.7 DATA TRANSFER RATES

The UBA can transfer data between the Unibus and the SBI through the DDP at approximately 400K Unibus words per second. However, using the buffered data paths, the data transfer rate can approach 750K Unibus words per second. Using both the DDP and the BDPs, the transfer rate falls between 400K and 750K words per second.

The throughput of the UBA is affected by:

1. SBI and memory usage.
2. The mix of BDP and DDP usage.
3. The frequency of interrupt servicing on the Unibus.
4. The frequency of UBA access by the program.

1.8 POWER REQUIREMENTS

Power supply:

+5 V @ 30 A
-5 V @ 1 A

Power Dissipation, 155 W

1.9 RELIABILITY AND DIAGNOSTIC FEATURES

Certain reliability and diagnostic capabilities are built into the UBA hardware. The following paragraphs summarize these capabilities.

1.9.1 UBA Reliability Features

- The Unibus to SBI address translation map RAM is protected by parity. Errors detected by the control logic will abort the Unibus transfer in progress and report the error.
- The buffered data path RAM is protected by parity. Errors detected by the control logic will abort the transfer and report the error.
- SBI faults, errors, and timeouts are detected, saved by the UBA, and reported through the SBI fault detection logic.
- Unibus timeouts, due to nonexistent memory or a hardware problem on the Unibus, are detected by the UBA and reported.
- The UBA monitors the Unibus data parity indicators (PA, PB) when accessing a Unibus device register or Unibus memory. Unibus register parity errors are reported in the same manner as uncorrectable SBI memory errors.
- The UBA provides timeouts so that problems in a device on the Unibus will not tie up the SBI.
- Power integrity indicators between the Unibus and SBI are isolated so that a power failure on the Unibus will not affect the performance of the SBI. A power problem on the SBI, however, will be indicated on the Unibus. Any change in power status of the Unibus or the UBA will be reported through the UBA interrupt mechanism.
- The UBA stores Unibus device interrupt vectors for use in case of an SBI data transmission failure during the process of fetching the interrupt vector. In this case, the vector is not lost and can be retrieved by the software.

1.9.2 UBA Diagnostic Features

- The UBA provides for complete wraparound data transfer from the SBI, through to the Unibus, and back to the SBI.
- The map registers can be read as well as written. This feature allows the software to test the integrity of the map RAM.
- Four read/write registers [BRSVR (0–3)] are provided so that the data path RAM can be tested easily.
- A diagnostic control register is provided to allow the software to defeat several UBA functions. Thus, the UBA can be tested to ensure that it gives the correct response in various failure modes. For example, the map RAM parity checkers, the data path RAM parity checkers, and interrupt responses from the Unibus can be tested.
- One bit in the diagnostic control register indicates whether or not the UBA microsequencer is in the idle state. It also tells whether the UBA has executed its initialization routine correctly.
- The UBA provides for data path testing at increasing levels of complexity. The least complex transfers are read and write transfers to the configuration and diagnostic control registers. The most complex is a wraparound transfer through the Unibus with byte offset.

CHAPTER 2
FUNCTIONAL DESCRIPTION

2.1 GENERAL

The first two parts of this functional description deal with the SBI and the Unibus, since a knowledge of both of them is fundamental to an understanding of the UBA. The following text provides explanations of the functions and major circuits of the UBA at the block diagram level. Addresses are given in hexadecimal notation and supplemented with octal where appropriate. The reader should note, however, that since the software code references VAX-11/780 system locations in hexadecimal, the Unibus I/O register addresses and vectors listed in the *PDP-11 Peripherals Handbook* must be converted from octal.

2.2 SYNCHRONOUS BACKPLANE INTERCONNECT DESCRIPTION

2.2.1 SBI Function

The SBI is the backplane of the VAX-11/780 system; it interconnects the CPU with the memory system and all adapters in the system. The following paragraphs describe all interconnect lines and their associated communication protocol.

2.2.2 Interconnect Synchronization

Six control group lines are clock signals and are used as a universal time base for all nexus connected to the SBI. 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 an attached nexus to synchronize SBI activity. Two clock signals (TPH and TPL) produce the basic nexus time states. The remaining four (PCLKH, PCLKL, PDCLKH, and PDCLKL) are phased clocks and help compensate for the clock distribution skew due to cable, backplane, and driver/receiver propagation delays.

2.2.2.1 Derived Time Status – The derived clocks (within the nexus) define four, 50 ns (nominal) time states in one clock period. The time states (T0, T1, T2, and T3) determine the transmit, propagate, and receive times on the SBI, with T0 representing the start of a particular clock period. Figure 2-1 illustrates the phase and timing relationships required to generate the individual derived time states. Note that T0 internal to the CPU (CPT 0) is not the same as SBI T0. CPT 0 corresponds to SBI T1. All nexus need a minimum of T0 and T2 for SBI transmit and receive functions.

2.2.2.2 Transmit Data – Information to be transmitted is asserted on the SBI at T0. Immediately prior to T0 a transmitting nexus enables its transmit enable inputs to the SBI transceivers. Figure 2-2 is a basic block diagram for one SBI information path line.

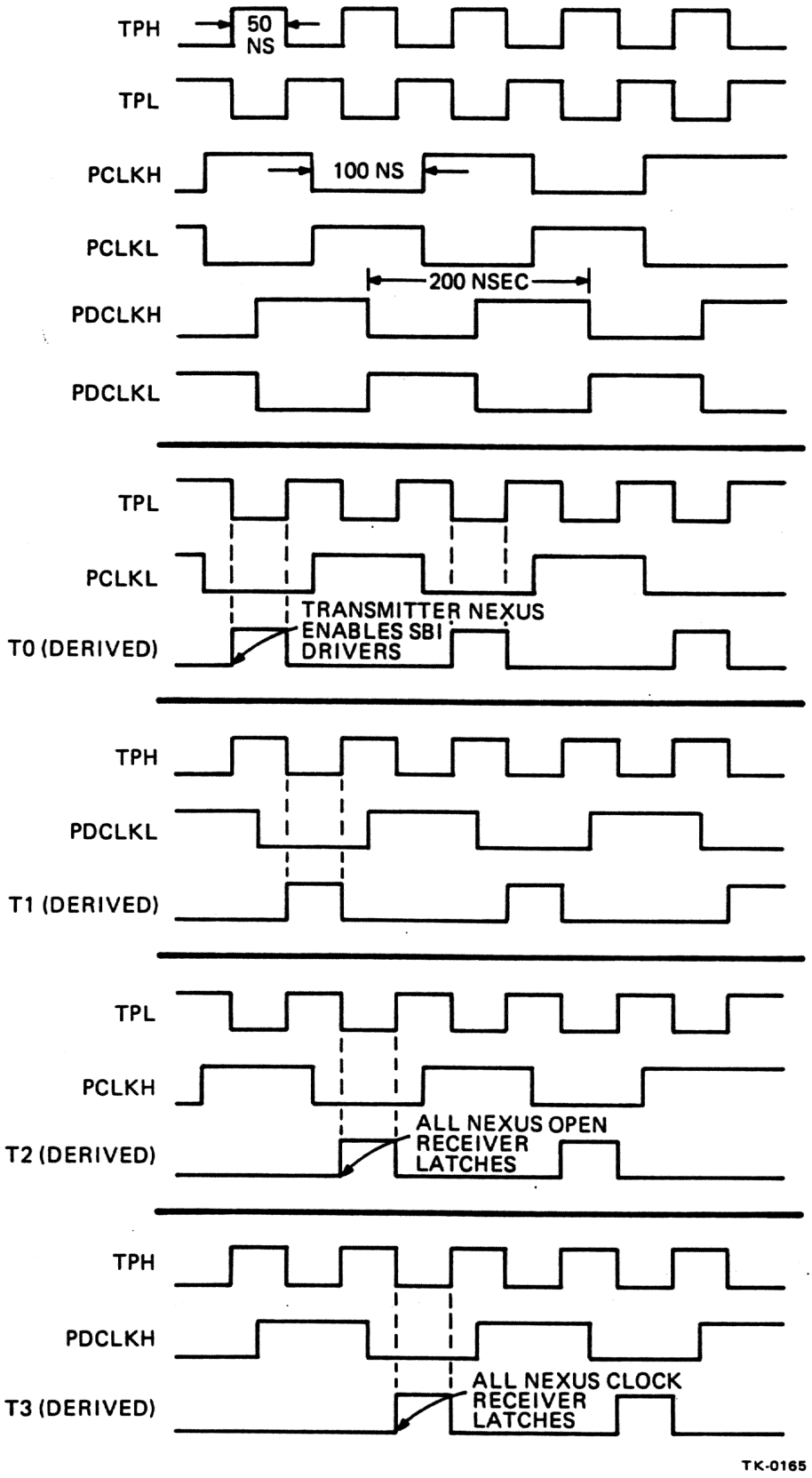


Figure 2-1 SBI Time and Phase Relationships

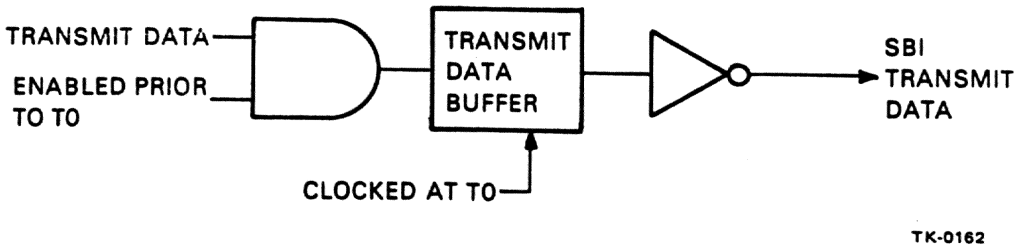


Figure 2-2 Transmit Data Path

2.2.2.3 Receive Data – In the case of receive data, the nexus receiver latches are opened at T2 and latched at T3. Figure 2-3 shows the basic one-line receiver latch logic. Note that the information may be considered undefined between T2 and T3; only after T3 is information considered valid. Nexus checking, decoding, and subsequent decision making are then based on these latched signals.

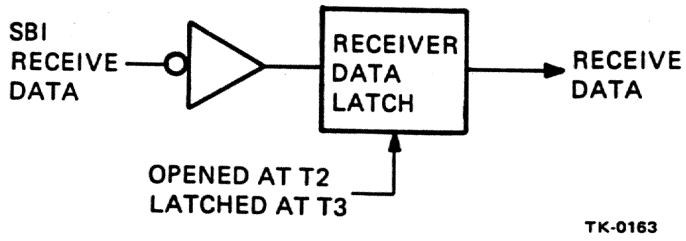


Figure 2-3 Receive Data Path

2.2.2.4 Single Time States – In single time states, the time between any T0-T1, T1-T2, T2-T3, T3-T0 may vary from 50 ns (nominal) to an indefinitely long period of time. SBI operation and protocol will proceed normally. Nexus that implement the SBI timeout functions (Paragraph 2.2.6.4) do so by counting SBI cycles.

2.2.3 SBI Summary

Table 2-1 summarizes the signal field associated with each functional group; Figure 2-4 shows the SBI configuration. The following paragraphs provide detailed descriptions of the individual group field layouts and functions.

2.2.4 Arbitration Group Functions and Assignments

The arbitration lines [Transfer Request, TR (15:00)] allow up to 16 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 increases from TR15 to TR00, where TR00 is the highest. The lowest priority level is reserved for the CPU, and it requires no actual TR signal line. The other 15 nexus are assigned TR15 through TR01.

Table 2-1 SBI Field Summary

Field	Description
Arbitration Group	
Arbitration Field [TR (15:00)]	Establishes a fixed priority among nexus for access to 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 (3:00)D]	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 (2:0)]	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. This field is part of the 32-bit information field.
Parity Field [P (1:0)]	Provides even parity for all information transfer path fields.
Response Group	
Confirmation Field [CNF (1:0)]	Encoded 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 to the CPU that indicates one of several errors stored in the transmitting nexus fault register, and the associated SBI cycle in which the error occurred.

Table 2-1 SBI Field Summary (Cont)

Field	Description
Interrupt Request Group	
Request Field [REQ (7:0)]	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 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 the restart nexus to provide a restart signal to the CPU to initiate a system restart operation.
Dead Field (DEAD)	A single line to the CPU to indicate an impending clock circuit or SBI terminating network power failure.
Unjam Field (UNJAM)	A single line from the CPU to attached nexus that initiates a restore operation.
Interlock Field (INTLK)	A single line that provides coordination among nexus responding to certain read/write commands to ensure exclusive access to shared data structures.

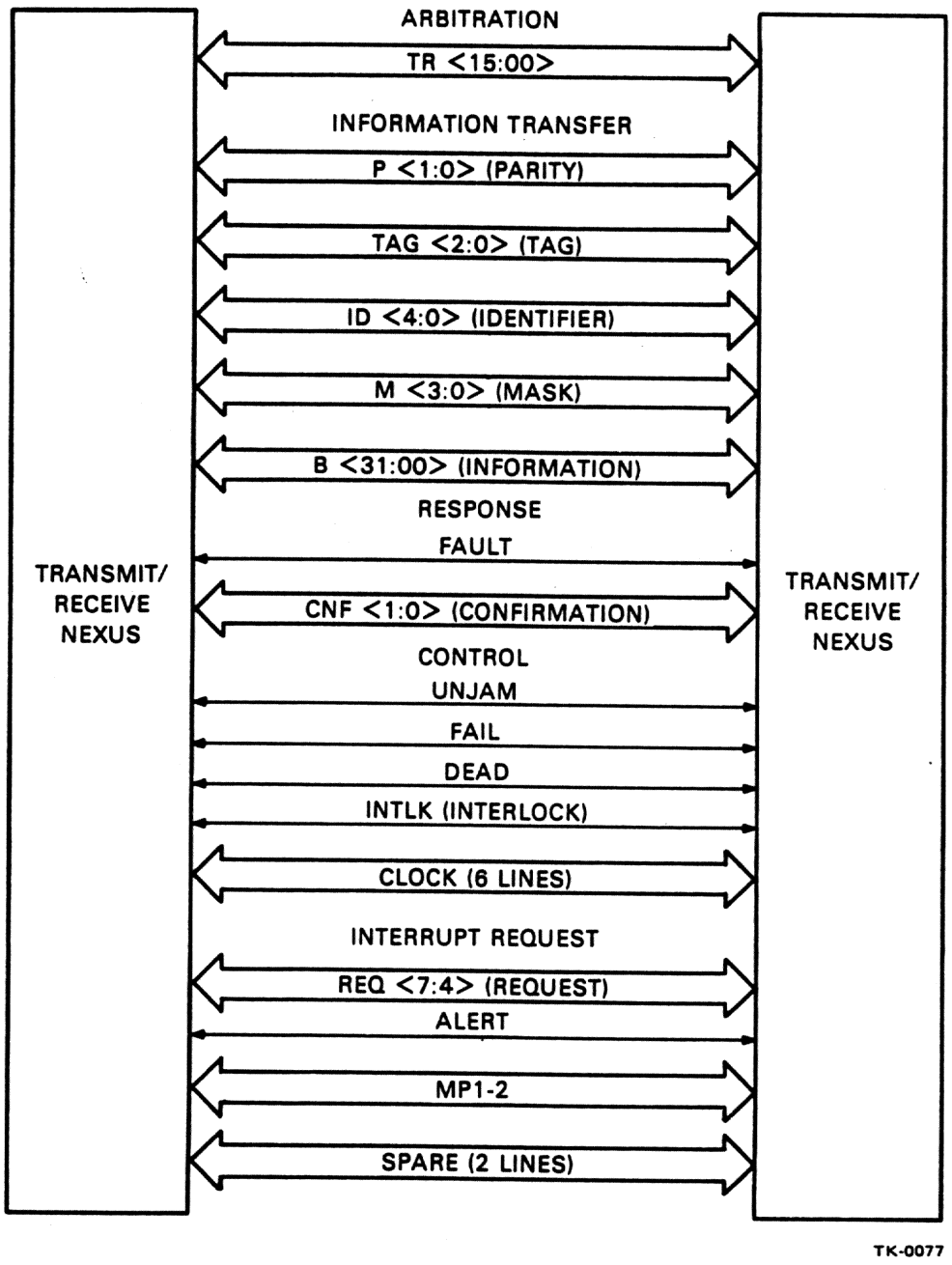


Figure 2-4 SBI Configuration

The highest priority level, TR00, is reserved for those nexus that require more than one successive SBI cycle. TR00 may only be used by nexus that require:

a. Two or three adjacent cycles for a write type exchange.
b. Two adjacent cycles for an extended read exchange.
c. Adjacent cycles for interrupt summary read exchanges or restore operations.

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 TR00 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.

2.2.5 Information Transfer Group Description

Each information group field is described in detail in the following paragraphs. However, the information field [B (31:00)] is described in the context of the other information group fields.

2.2.5.1 Parity Field – The parity field [P (1:0)] provides even parity for detecting single bit errors in the information group (Figure 2-5).

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.

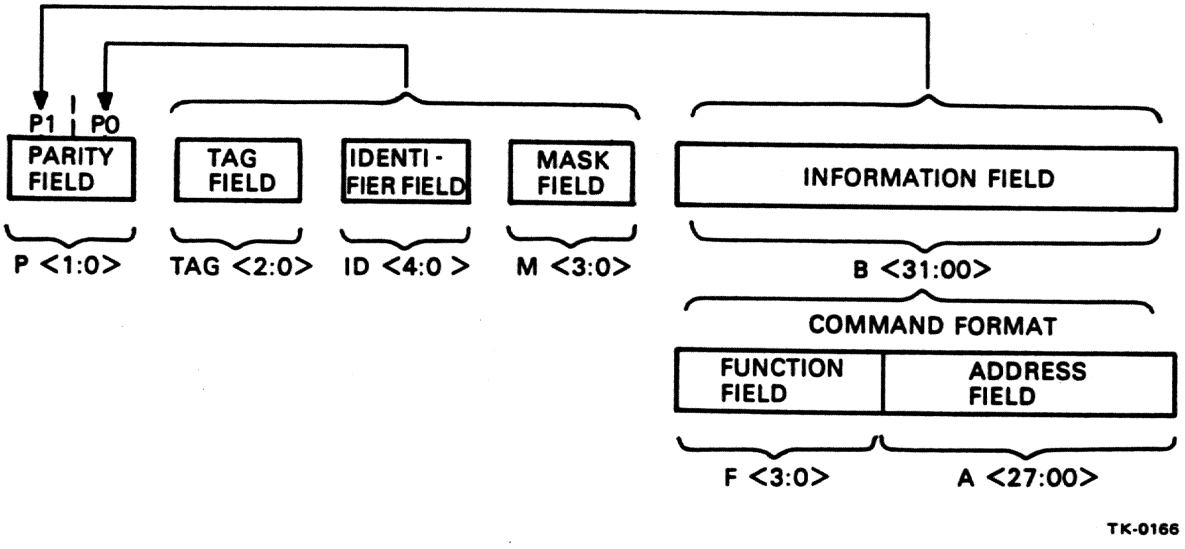


Figure 2-5 Parity Field Configuration

2.2.5.2 Tag Field Formats – The tag field [TAG (2:0)] is asserted by a transmitting nexus to indicate the information type being transmitted on the information lines. The tag field determines the interrelation of the ID and B fields. In addition, the tag field, in conjunction with the mask field, further defines special read and write data conditions. The following paragraphs describe each information type, tag code, and associated field content.

2.2.5.2.1 Command/Address Tag – A tag field content of 011 indicates that the content of B (31:00) is a command/address word, and that ID (4:0) is a unique code identifying the logical source (commander) of that command. As shown in Figure 2-6, B (31:00) is divided into a function field and an address field to specify the command and its associated address.

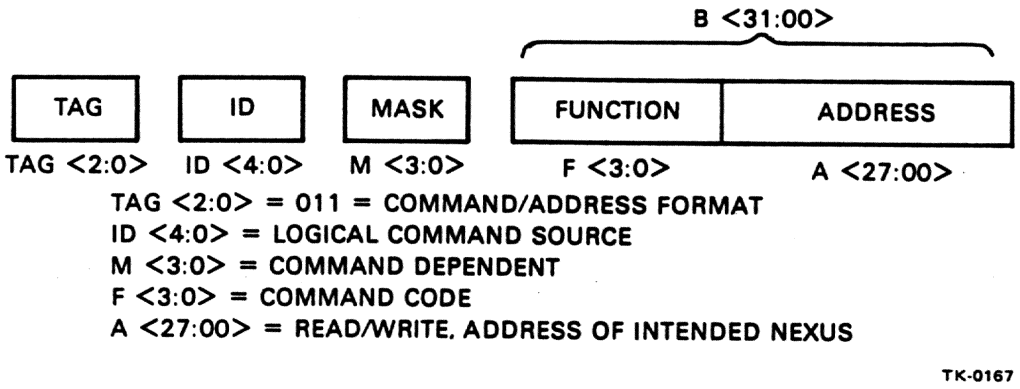


Figure 2-6 Command/Address Format

The ID field code represents the logical source in a write type command; the address field specifies the logical command destination. For a read type command, the ID field represents the logical destination of the data at the location specified in the address field.

The 28 bits of the SBI address field define a 268, 435, 456 longword address space, which is divided into two sections. Addresses $0\text{–}7FFFFFF_{16}$ are reserved for primary memory. Addresses $8000000_{16}\text{–}FFFFFFF_{16}$ are reserved for device control registers. Generally, primary memory begins at address 0; the address space is dense and consists only of storage elements. The control address space is sparse with address assignments based on device type. Each nexus is assigned a 2048, 32-bit longword address space for control. The addresses assigned are determined by the TR number as shown in Figure 2-7.

2.2.5.2.2 Read Data Tag – A tag field content of 000 indicates 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 normally expected error-free data having M (3:0) = 0000 (Figure 2-8). Note that this tag code is also the idle state of the tag field and that ID code 0 is reserved. No devices will respond when the tag is 000 and the ID code is 0.

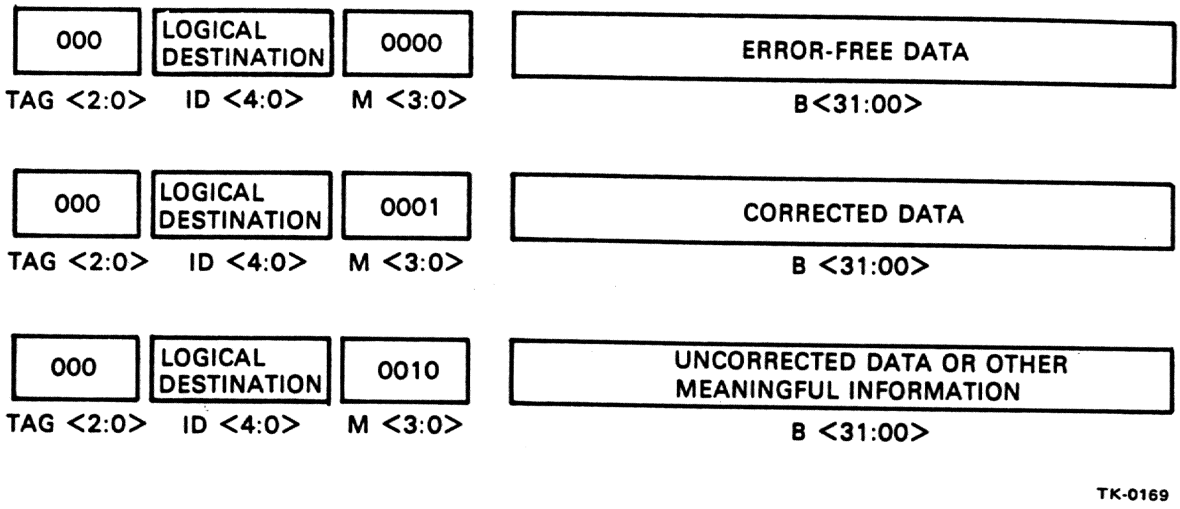


Figure 2-7 Control Address Space Assignment

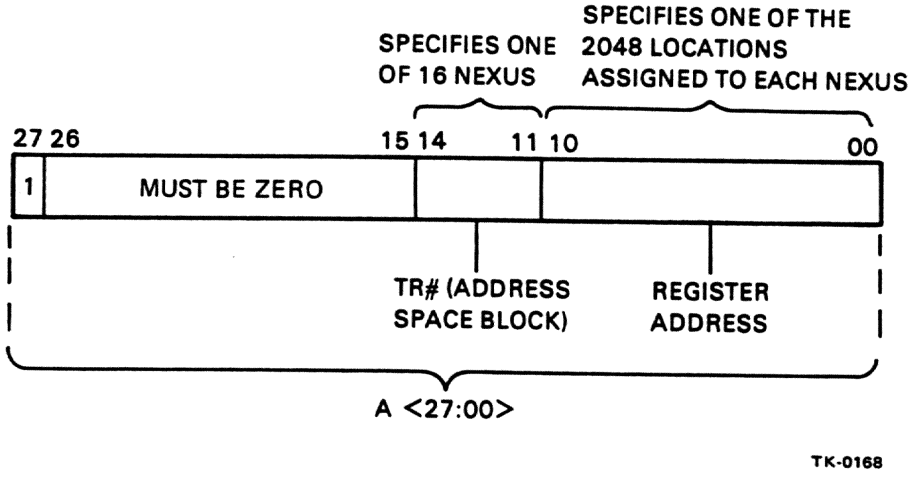


Figure 2-8 Read Data Formats

Corrected read data is data in which an error was detected and subsequently corrected by the error correction code logic (ECC) 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 not corrected. In this case, B (31:00) will contain the substitute data in the form of uncorrected data or other meaningful information. The mask field flags the uncorrected data with M (3:0) = 0010. As with the other read data types, the ID field identifies the read commander.

2.2.5.2.3 Write Data Tag – A tag field content of 101 indicates that B (31:00) contains the write data for the location specified in the address field of the previous write command (Figure 2-9). The write data will be asserted on B (31:00) in the SBI cycle immediately following the command/address cycle. Certain command codes use M (3:0) to specify particular bytes within B (31:00) (Paragraph 2.2.5.4).

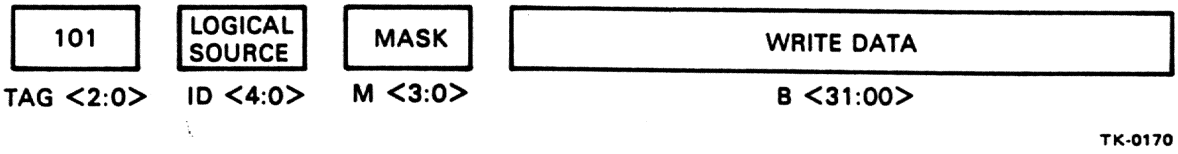


Figure 2-9 Write Data Formats

2.2.5.2.4 Interrupt Summary Tag – A tag field content of 110 defines B (31:00) as the interrupt level mask for an interrupt summary read command. The level mask [B (07:04)] is used to indicate the interrupt level being serviced as the result of an interrupt request. In this case, the ID field identifies the commander, which is usually a CPU. Although unused, M (3:0) must be transmitted as zero.

The interrupt sequence consists of two exchanges:

a. The first exchange indicates the interrupt level being serviced.

b. The second exchange is the response, where the device requesting the interrupt identifies itself.

The interrupt summary read and response formats are illustrated in Figure 2-10. Note that the interrupt summary response encodes TAG (2:0) = 000.

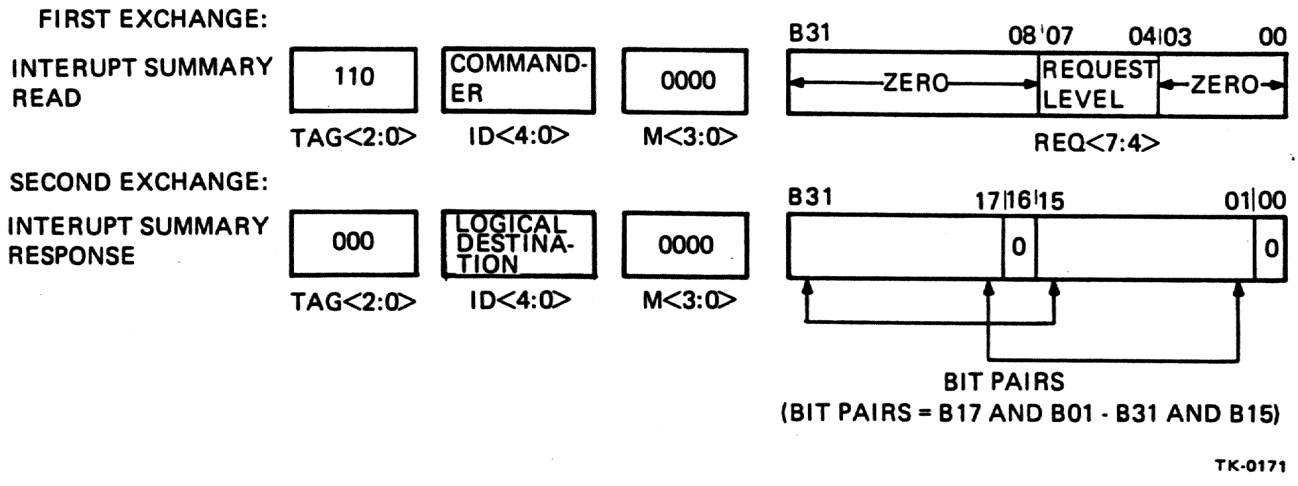


Figure 2-10 Interrupt Summary Formats

2.2.5.2.5 Reserved Tag Codes - TAG (2:0) - Tag code 111 is reserved for diagnostic purposes. Tag codes 001, 010, and 100 are unused and reserved for future definition.

2.2.5.3 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). ID codes are assigned only to commander and responder nexus (i.e., those that issue and recognize command/address information). Each nexus is assigned an ID code that corresponds to the TR line which it operates. For example, a nexus assigned TR05 would also be assigned ID code = 5.

More than one ID code may be assigned to a nexus. However, that nexus must be capable of responding to read type commands for which the read data returns in an order different from the order in which the commands were given. For write masked and extended write masked commands, the mask is transmitted in the cycle preceding the cycle for the data to which the mask applies.

Nexus using more than one code take the first code from the standard ID code assignment (0–15). The second code is taken from the range 17–30 (i.e., first ID code + 16).

Certain ID codes are reserved: ID = 16, unit processors; ID = 31, diagnostic purposes. ID = 0 is reserved so that the idle state of the SBI (read data, destination ID = 0) will not cause a nexus selection. Note that even though a nexus is not selected, all nexus are checking for correct SBI parity.

2.2.5.4 Mask Field - The mask field [M (3:0)] has two interpretations: primary and secondary. In the primary interpretation, M (3:0) is encoded to specify operations on any or all bytes appearing on B (31:00). The mask is used with the read masked, write masked, interlock read masked, interlock write masked, and extended write masked commands. As shown in Figure 2-11, each bit in the mask field corresponds to a particular byte on B (31:00).

As mentioned in Paragraph 2.2.5.2.2, the secondary interpretation is used when TAG (2:0) = 000 (read data). This interpretation defines the data types as specified in Table 2-2. All other mask field codes (0011–1111) are reserved and are interpreted as read data substitutes by receiving nexus.

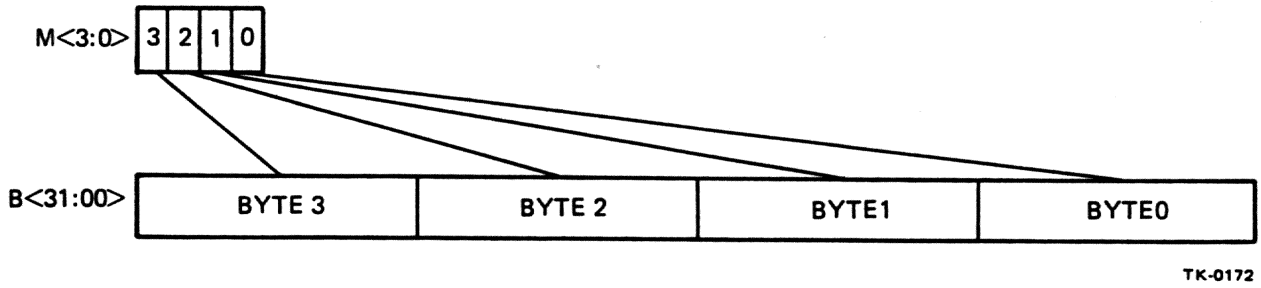


Figure 2-11 Mask Field Format

Table 2-2 Read Data Types

M (3:0)	Data Type
0000	Read Data
0001	Corrected Read Data
0010	Read Data Substitute

2.2.6 Response Group Description

The three response lines are divided into two fields: Confirmation [CNF (1:0)], and Fault (FAULT). CNF (1:0) informs the transmitter as to whether or not the information was received correctly and if the receiver can process the command. FAULT is a cumulative error indication of protocol or information path malfunction; it is asserted with the same timing as the confirmation field.

Either field is transmitted to the receiver 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 allow confirmation generation 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, one (or more) 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.

2.2.6.1 Confirmation Codes – Table 2-3 lists the confirmation codes and their interpretation.

Table 2-3 Confirmation Code Definitions

CNF Code	Definitions
00, No Response (N/R)	The unasserted state; it indicates no response to a commander's selection.
01, Acknowledge (ACK)	The positive acknowledgment to any transfer.
10, Busy (BSY)	The response to a command/address transfer that indicates successful selection of a nexus which is presently unable to execute the command.
11, Error (ERR)	The response to a command/address transfer that indicates selection of a nexus which cannot execute the command.

A BSY (10) or ERR (11) response to transfers other than command/address transfers will be considered as no response from the responder.

2.2.6.2 Response Handling – The transmitting nexus samples the CNF and FAULT lines at T3 of the third cycle following transmission. ACK is the expected confirmation response (i.e., command will be executed or information has been correctly received).

Should a command/address transfer receive a BSY confirmation, the commander will repeat the transmission (after a nominal waiting period) until it is accepted.

An N/R confirmation should be treated like BSY, except that its occurrence may be flagged in a status bit. ERR confirmation is the result of a programming error and should abort the command and invoke the appropriate recovery routine.

Some nexus may be unable to determine within two SBI cycles whether a function will be completed successfully. For these cases, the nexus presumes success and responds with ACK confirmation. If it is later determined that a read type function cannot be completed, a read data transfer of all zeros is transmitted and an interrupt requested. If a write type request cannot be completed, the command is aborted and an interrupt requested. In either case, the cause of the interrupt is indicated in a configuration or status register.

2.2.6.3 Successive Cycle Confirmation – Since write masked, extended write masked, and extended read operations (Paragraph 2.2.8) consist of successive transfers, acknowledgment is more complex.

a. If the command/address transfer is confirmed with N/R or BSY, then no notice will be taken of the data transfer confirmation and the entire sequence will be repeated.

b. If the command/address transfer receives ERR, the sequence is aborted and recovery routines are invoked.

c. If ACK is not received as confirmation for a write data command, the command is repeated.

d. Transmissions of read data are confirmed with ACK by the receiver of that data. The read data transmitter may ignore this confirmation, since the burden of retry is on the commander.

2.2.6.4 SBI Sequence Timeouts – All commanders implement two timeout functions: interface sequence timeout and read data timeout. Both timeouts are specified as 102.4 μs (or 512 SBI cycles).

The interface sequence timeout determines the maximum time allowed to complete an interface sequence. The sequence interval is defined as the time from:

a. when SBI arbitration is initiated, until ACK is received for a command/address transfer that specifies read, or

b. when SBI arbitration is initiated, until ACK is received for a command/address transfer that specifies write, and ACK is also received for each transmission of write data, or

c. when SBI arbitration is initiated, and an ERR confirmation is received for any command/address transfer.

The read data timeout is defined as the time from when an interface sequence that specifies a read command is completed to the time that the specified read data is returned to the commander. In the case of an extended read function, both longwords must be retrieved prior to timeout (102.4 μs).

If the last command/address transfer prior to an interface sequence timeout receives an N/R confirmation, it is recorded in a status bit. Certain nexus may terminate their requests for SBI control due to an unusual occurrence in those nexus. When this occurs, both timeouts are cancelled (e.g., when a nexus detects a data late error).

When a timeout occurs, the commander provides the actual address or reconstructed address for which the timeout occurred. In addition, the commander records the type of timeout received (i.e., interface sequence or read data). Either timeout will terminate a command transmission retry.

2.2.6.5 Fault Detection – Each nexus is equipped with a 32-bit configuration and fault status register (register 0). The fault status portion of this register contains flags that cause the assertion of the FAULT line. The fault status portion is described in Figure 2-12.

A nexus detecting one of the fault conditions will assert the FAULT signal for one cycle. FAULT then causes each nexus on the system to lock its respective configuration register. The fault status bits thus latched refer to the cycle during which the fault occurred. The CPU examines the FAULT signal and latches the signal on the leading edge of FAULT. The CPU then continues to assert FAULT until the software has examined the fault bits of all nexus and has specified the negation of FAULT. Figure 2-13 shows the timing involved.

Figure 2-14 illustrates the confirmation and fault decision flow for all responses and error conditions.

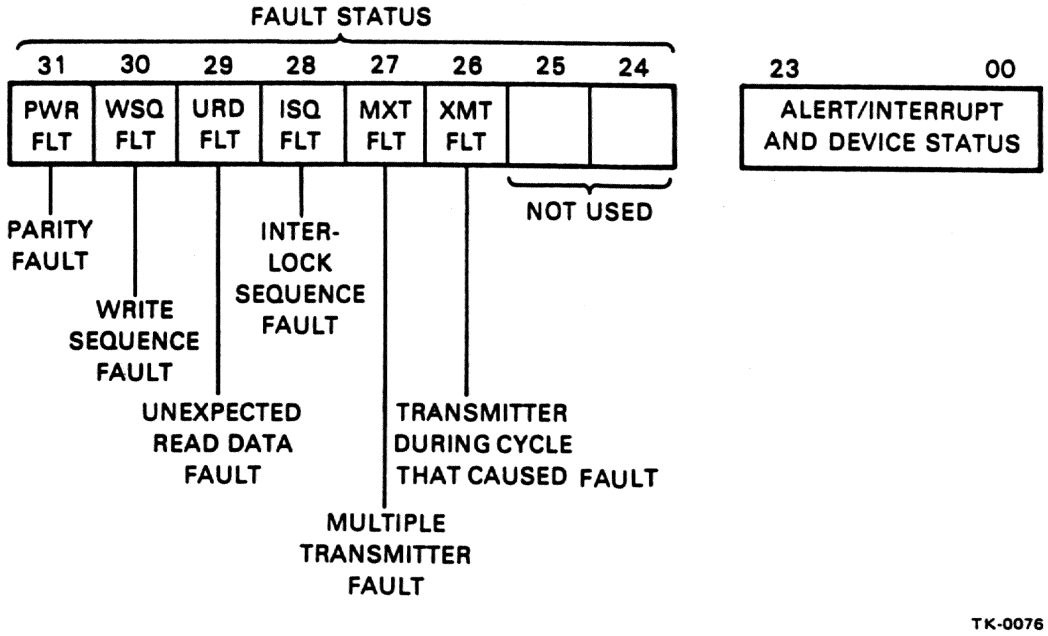


Figure 2-12 Fault Status Flags

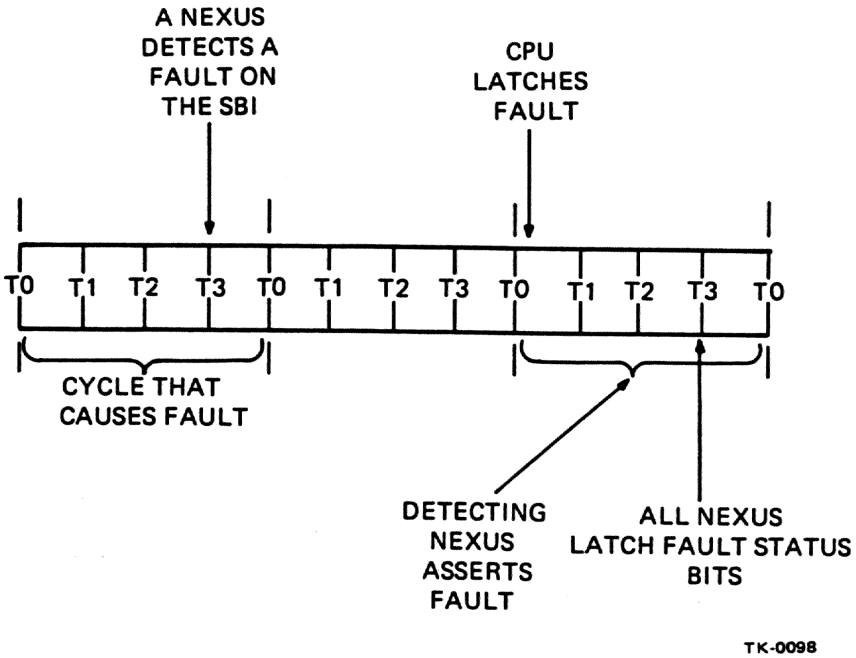
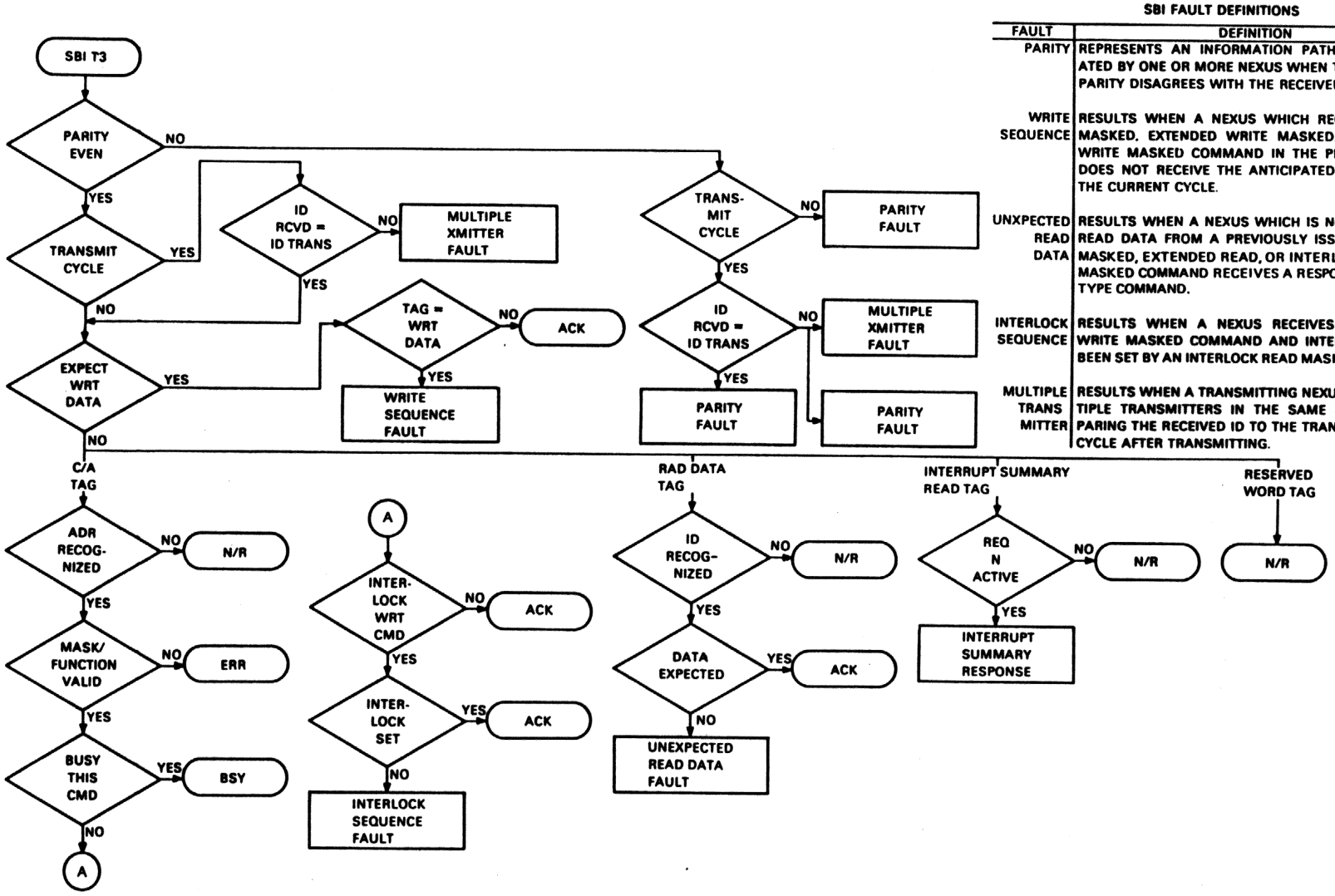


Figure 2-13 Fault Timing



SBI FAULT DEFINITIONS

FAULT	DEFINITION
PARITY	REPRESENTS AN INFORMATION PATH ERROR GENERATED BY ONE OR MORE NEXUS WHEN THE CALCULATED PARITY DISAGREES WITH THE RECEIVED PARITY.
WRITE SEQUENCE	RESULTS WHEN A NEXUS WHICH RECEIVED A WRITE MASKED, EXTENDED WRITE MASKED, OR INTERLOCK WRITE MASKED COMMAND IN THE PRECEDING CYCLE DOES NOT RECEIVE THE ANTICIPATED WRITE DATA IN THE CURRENT CYCLE.
UNXPECTED READ DATA	RESULTS WHEN A NEXUS WHICH IS NOT WAITING FOR READ DATA FROM A PREVIOUSLY ISSUED READ MASKED, EXTENDED READ, OR INTERLOCK READ MASKED COMMAND RECEIVES A RESPONSE TO A READ TYPE COMMAND.
INTERLOCK SEQUENCE	RESULTS WHEN A NEXUS RECEIVES AN INTERLOCK WRITE MASKED COMMAND AND INTERLOCK HAS NOT BEEN SET BY AN INTERLOCK READ MASKED COMMAND.
MULTIPLE TRANS MITTER	RESULTS WHEN A TRANSMITTING NEXUS DETECTS MULTIPLE TRANSMITTERS IN THE SAME CYCLE BY COMPARING THE RECEIVED ID TO THE TRANSMITTED ID ONE CYCLE AFTER TRANSMITTING.

Figure 2-14 Confirmation and Fault Decision Flow

2.2.7 Interrupt Request Group Description

The interrupt request group consists of four request lines [REQ (7:4)] and an alert (ALERT) line. Request lines are assigned to some of the nexus and represent assigned CPU interrupt levels. The lines used by nexus request that the CPU service a condition requiring processor intervention. The request lines are priority encoded in an ascending order of REQ4–REQ7. A requesting nexus asserts its request lines (or line) asynchronously with respect to the SBI clock to request an interrupt. Any of the REQ lines may be asserted simultaneously by more than one nexus, and any combination of REQ lines may be asserted by the collection of requesting nexus.

The ALERT signal is asserted by nexus that do not implement interrupt request lines. Its purpose is to indicate to the CPU a change in the nexus power condition or operating environment. Nexus that implement the REQ lines report such changes by requesting an interrupt.

2.2.7.1 Interrupt Operation – When a nexus requires an interrupt, it asserts its REQ line on the SBI. At a time judged appropriate, the CPU will recognize the interrupt request and issue an interrupt summary read command [TAG (2:0) = 110]. The command will have a single bit set in its interrupt level mask [B (7:4)] which corresponds to the REQ line being serviced. For example, B 04 set to a logic one indicates that the REQ 4 level is being serviced. Note that the remaining information path fields [i.e., B (31:08), ID (03:00), and M (3:0)] are transmitted as zero.

Nexus receiving the interrupt summary read command without error and asserting the REQ line specified in the interrupt level mask will assert a 2-bit code in B (31:00). This code, which identifies the requesting nexus, is asserted with the timing of CNF (1:0). However, the responding nexus does not assert any CNF, TR, ID, or TAG line. Nexus that detect incorrect parity will assert FAULT.

As shown in Figure 2-15, the asserted bits are in corresponding positions in the upper and lower 16 bits of B (31:00). The bit pair uniquely identifies the nexus among those using the particular REQ line. Only 15 bit pairs in the information field are used (i.e., B31 and B15 through B17 and B01). Since only pairs of bits are asserted, parity remains correct regardless of the number of responding nexus. The two bits assserted by the requesting nexus are equal to the nexus TR number and the nexus TR number plus 16.

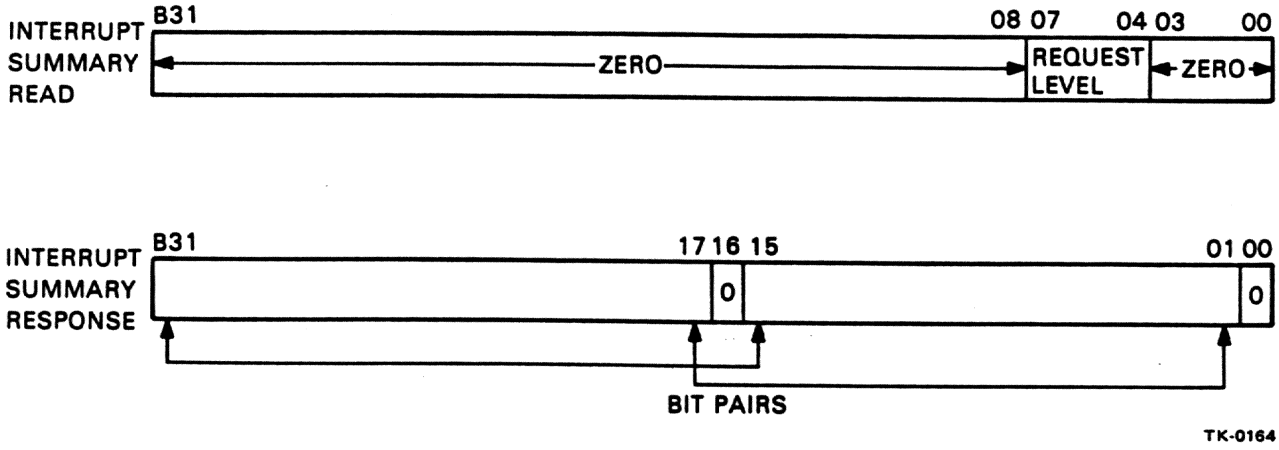


Figure 2-15 Request Level and Nexus Identification

While holding control of the SBI with TR00, the CPU waits two cycles after the interrupt summary read command is transmitted before latching B (31:00) into an internal register. By encoding the REQ level and the bit pair received from responding nexus, the CPU generates a vector unique to that level and nexus. The vector, in turn, is used to invoke the nexus service routine. The service routine will take explicit action by writing a device register to clear the interrupt condition. Clearing the interrupt causes the nexus to negate the REQ line, provided that the nexus does not have any other outstanding interrupts at this level. The negation of REQ occurs within two cycles of the write data transmission.

Normally, the CPU will service requests in a descending order of REQ7–REQ4. Similarly, nexus are identified in descending order beginning with the nexus that asserts bits B31 and B15 and ending with the nexus that asserts bits B17 and B1. If multiple nexus are requesting interrupts on the same REQ line, multiple interrupt summary read commands are issued until all nexus have been serviced and the REQ line is no longer asserted.

Figure 2-16 is a functional timing chart for the interrupt operation.

2.2.7.2 Status Register Alert Flags – As shown in Figure 2-17, each nexus maintains bits in its configuration register to indicate conditions that cause assertion of Alert (or the appropriate REQ line if implemented). Power Down an1 Power Up status bits are provided, but additional Alert status bits are present if other conditions, such as over temperature, are detectable.

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. Note that the UNJAM signal does not clear these status bits.

2.2.7.3 Alert Flag Operation – A nexus asserts ALERT or an interrupt request when any of its Alert status bits are set. The bits are set during the following events:

a. during power failure at the nexus when the assertion of power supply AC LO is recognized
b. during the restoration of power when the negation of AC LO is recognized
c. when other environmental conditions such as overtemperature are detected.

The Alert status bits are only set on the transition of the event that caused them to set.

The Power Down status bit is set when there is a transition of the nexus AC LO from the negated to the asserted state. Setting the Power Down status bit clears the Power Up status bit; likewise, setting the Power Up bit clears the Power Down bit. The overtemperature bit is set when there is a transition from the normal to the overtemperature state.

A nexus asserting ALERT, or asserting an interrupt request due to an Alert status bit set, continues to assert ALERT or the request until:

a. All Alert status bits are cleared (written with a logic one).
b. UNJAM signal is received.
c. Nexus loses dc power.

The negation of Alert (or REQ) is synchronous to the SBI clock and occurs within two cycles of the write data transmission used to clear the Alert condition.

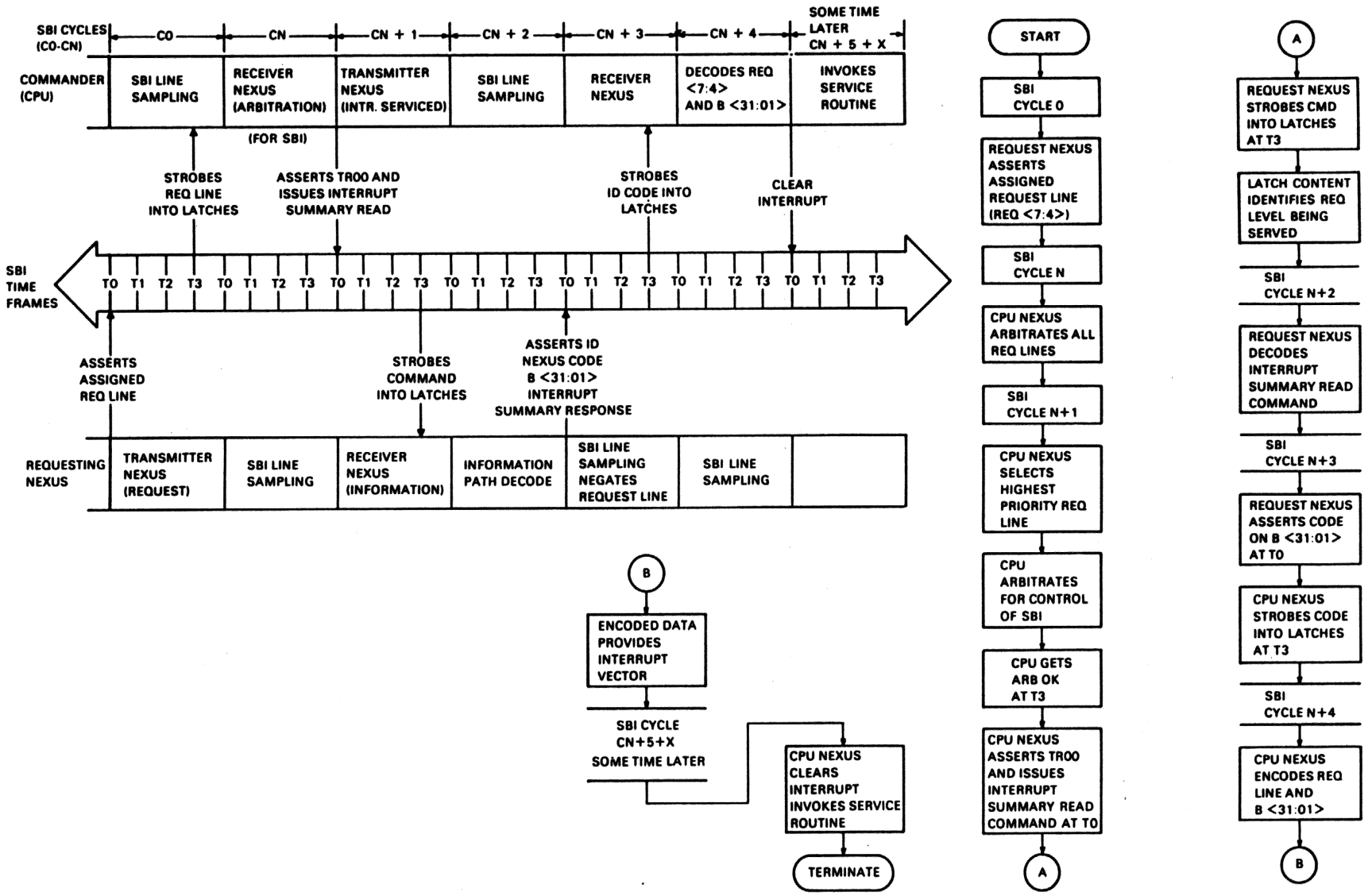


Figure 2-16 Interrupt Operation Timing and Flow

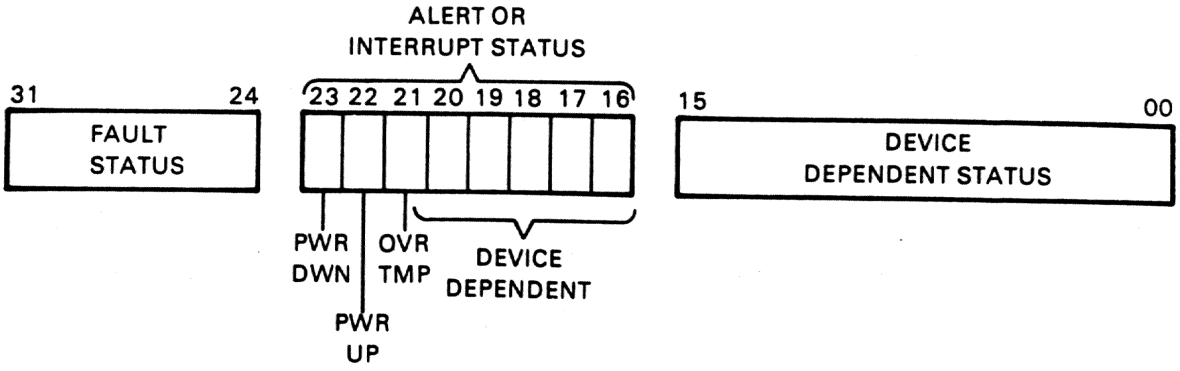
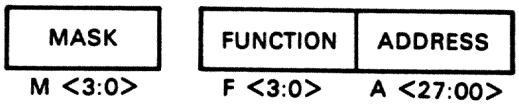


Figure 2-17 Alert Status Bits

2.2.8 Command Code Description

The operations executed over the SBI are specified in command/address (C/A) formaat (Paragraph 2.2.5.2.1) using the mask, function, and address fields. Figure 2-18 summarizes the C/A formats and lists the command codes. Several function codes are unused and reserved for future use. All nexus must respond to the reserved codes with an N/R confirmation.



MASK USE	FUNCTION CODE	FUNCTION DEFINITION
IGNORED	0000	RESERVED
USED	0001	READ MASKED
USED	0010	WRITE MASKED
IGNORED	0011	RESERVED
USED	0100	INTERLOCK READ MASKED
IGNORED	0101	RESERVED
IGNORED	0110	RESERVED
USED	0111	INTERLOCK WRITE MASKED
IGNORED	1000	EXTENDED READ
IGNORED	1001	RESERVED
IGNORED	1010	RESERVED
USED	1011	EXTENDED WRITE MASKED
IGNORED	1100	RESERVED
IGNORED	1101	RESERVED
IGNORED	1110	RESERVED
IGNORED	1111	RESERVED

TK-0083

Figure 2-18 SBI Command Codes

2.2.8.1 Read Masked Function – The read masked function is specified in Figure 2-19.

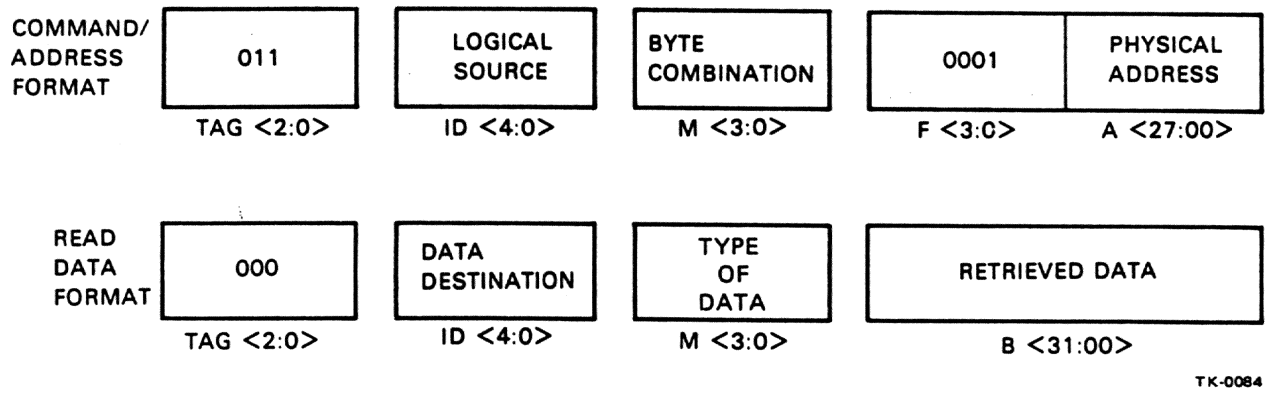


Figure 2-19 Read Masked Function Format

Prior to issuing the command, the commander arbitrates for SBI control. When the commander gains control of the SBI, it asserts the information transfer lines at T0. At T3 of the same cycle each nexus strobes the C/A information into its receiver latches for decoding. The C/A format presented on the SBI instructs the nexus selected by the address field, SA (27:00), to retrieve the addressed data word, and transfer it to the logical destination specified in the ID field. The addressed nexus will respond to the C/A transfer with ACK (assuming no errors) two SBI cycles after the assertion of C/A.

The addressed data is retrieved in a time frame that is dependent on the nexus response time. Following the response delay, the responding nexus must arbitrate for control of the SBI. After ARB OK is true for the responder, the information fields are asserted on the SBI at T0. TAG (2:0) is coded as 000, specifying the read data format, and ID (4:0) is coded to identify the commander. The read data is asserted on B (31:00) and received by the commander as read data [M (3:0) = 0000], or as corrected read data [M (3:0) = 0001]. In the case of uncorrectable read data, the responder transmits read data substitute [M (3:0) = 0010].

After the assertion of read data, the commander latches the content of B (31:00) at T3 of the same SBI cycle. At T0, two cycles later, the commander confirms the successful transfer by asserting ACK.

Figure 2-20 is a functional timing chart for the read masked operation.

2.2.8.2 Extended Read Function – The extended read function is similar to the read masked function in operation. The function format is shown in Figure 2-21.

The mask field and bit SA00 of the received C/A word are ignored. However, the mask field must be transmitted as zero.

In an extended read, 64 bits (two data longwords) are always transmitted, and thus require two contiguous SBI data transfer cycles. In this case, F (3:0) instructs the nexus selected by SA (27:00) to retrieve the addressed 64-bit data and transfer it to the commander (specified in the ID field) as in the read masked function.

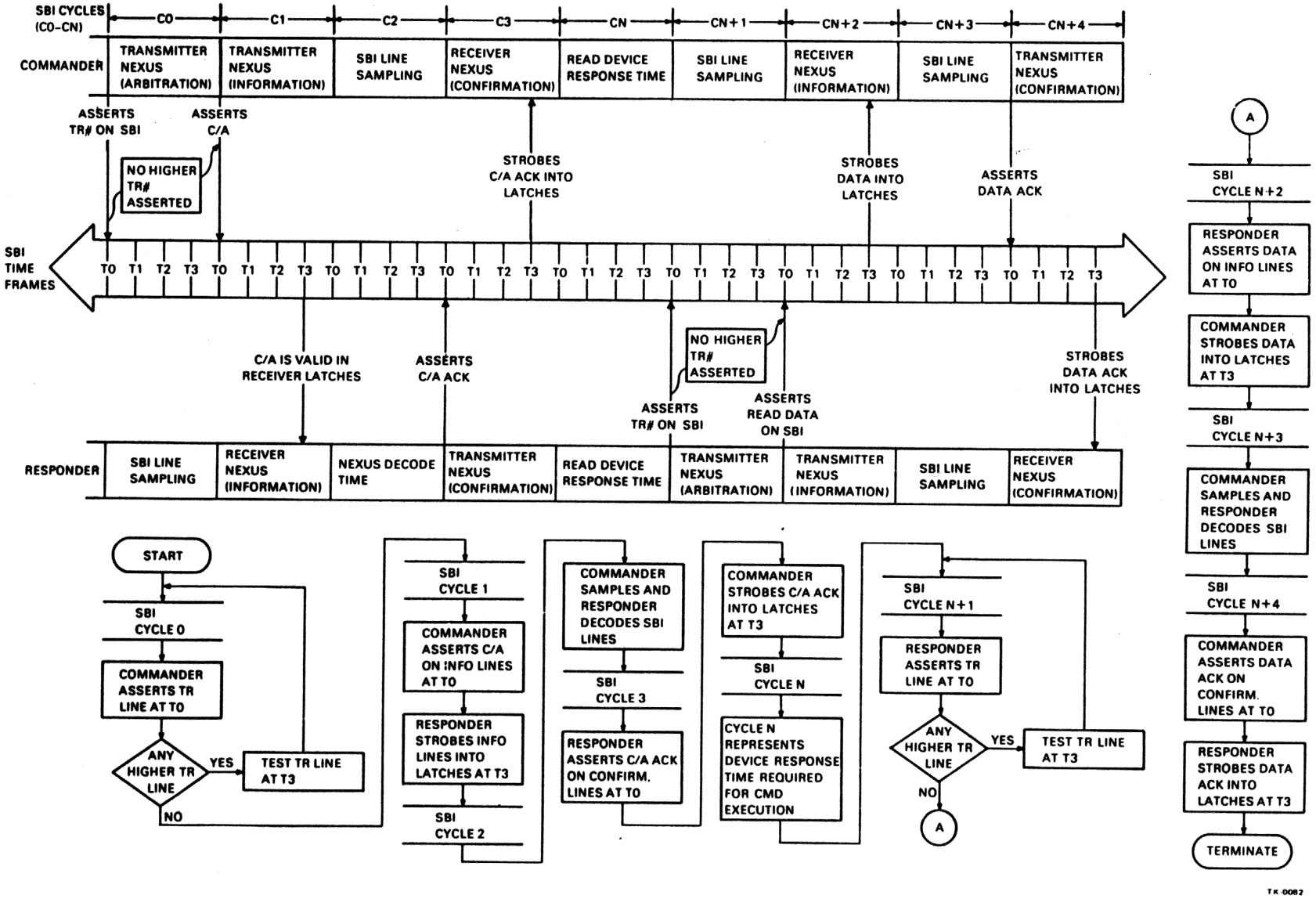


Figure 2-20 Read Masked Timing Chart

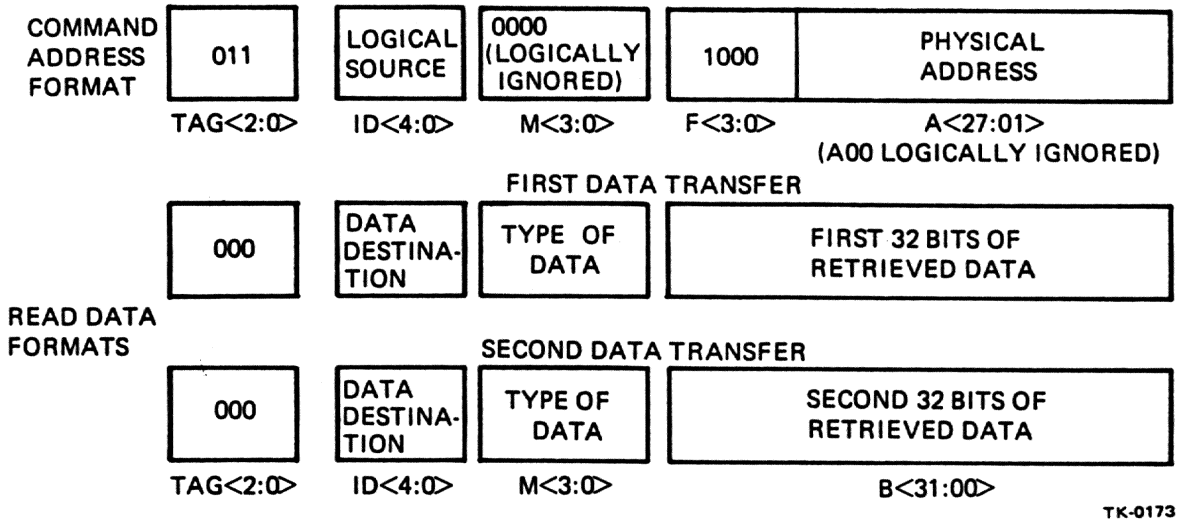


Figure 2-21 Extended Read Function Format

When the commander gains control of the SBI, it asserts the C/A information at T0. At T3 of the same cycle, each nexus strobes the C/A information into its receiver latches for decoding. The addressed nexus confirms the C/A transfer by returning ACK two cycles after the assertion of C/A. Following the response delay and arbitration, the responder asserts the first 32-bit data longword on B (31:00) (SA00=0). The other information fields are coded as in the read masked operation. The second data longword (SA00=1) is asserted on B (31:00) at T0 of the succeeding cycle. The mask field describing the data type will be asserted with each read data longword.

The commander latches B (31:00) (first data longword) at T3 of the cycle when it was transmitted. At T3 of the next cycle, the commander again latches B (31:00) (second data longword). Then, at T0 of the following cycle, the commander confirms the first data transfer with ACK. The commander confirms the second data transfer with ACK at T0 of the cycle after that.

Figure 2-22 is a functional timing chart showing the extended read operation.

2.2.8.3 Write Masked Function – The write masked function format is shown in Figure 2-23. F (3:0) instructs the selected nexus to modify the bytes specified by M (3:0) in that storage element addressed by SA (27:00) using data transmitted in the succeeding cycle.

When the commander gains control of the SBI, it asserts the C/A information at T0. The commander also asserts TR00 at T0 to retain control during the succeeding SBI cycle. At T3 of the same cycle, each nexus strobes the C/A information into its receiver latches for decoding. At T0 of the succeeding cycle, the commander asserts data on B (31:00), and at T3 of the same cycle the selected nexus strobes the data into its receiver latches. TAG (2:0), which accompanies the data, is coded 101 (write data format). The successful C/A transfer is confirmed by the receiving nexus with ACK at T0 of the succeeding cycle. The successful data transfer is confirmed by ACK at T0 one cycle later.

Figure 2-24 is a functional timing chart for the write masked operation.

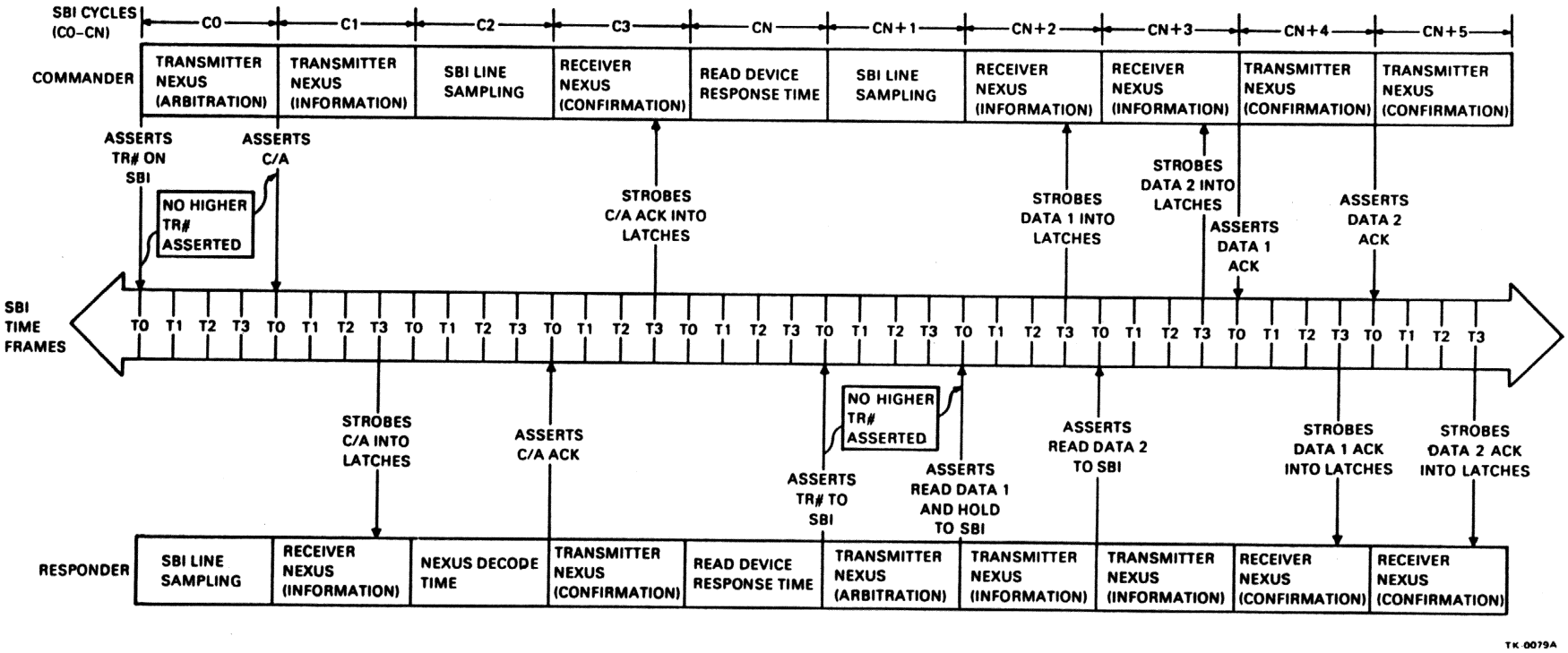


Figure 2-22 Extended Read Timing Chart and Flow (Sheet 1 of 2)

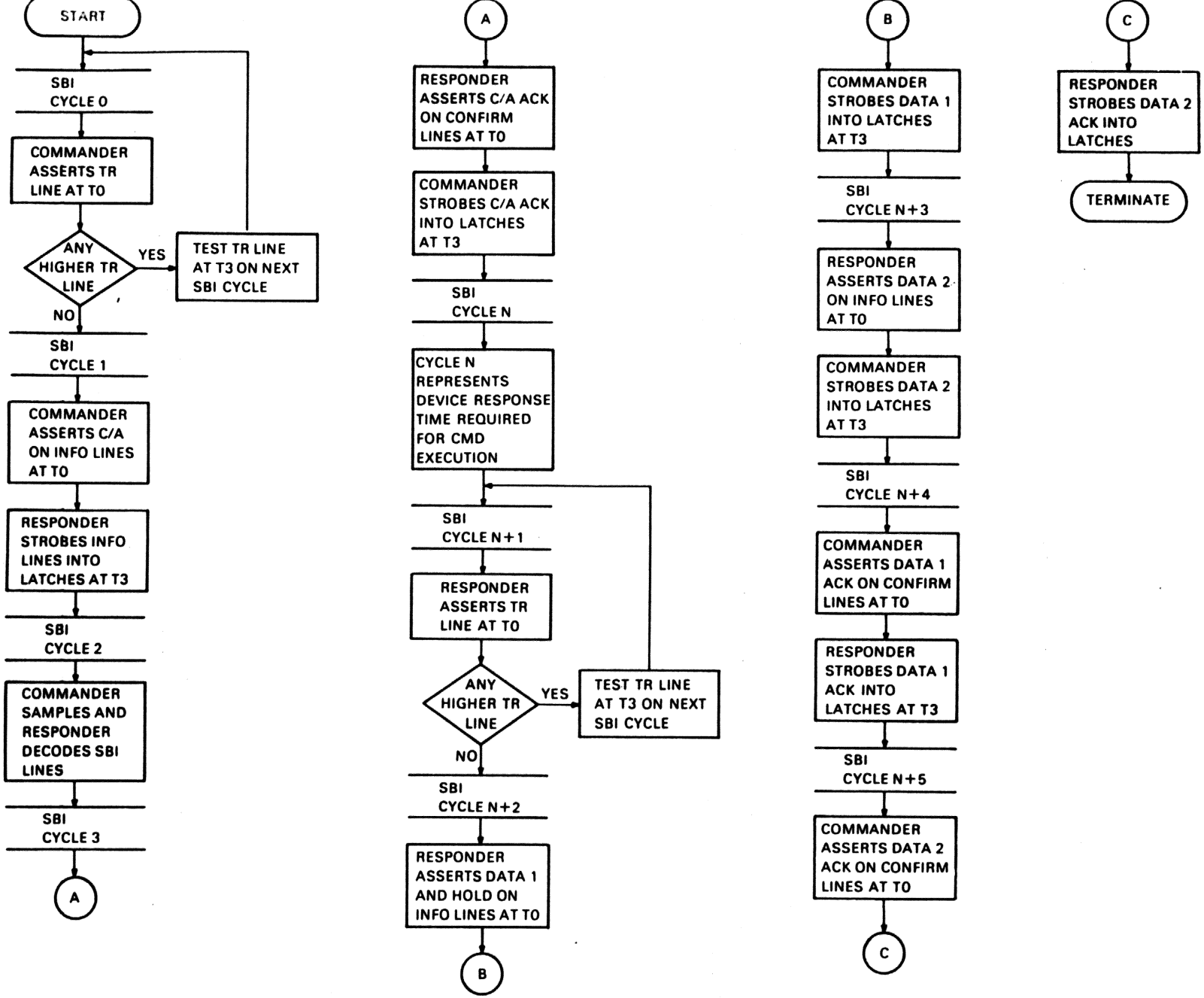


Figure 2-22 Extended Read Timing Chart and Flow (Sheet 2 of 2)

TK-00798

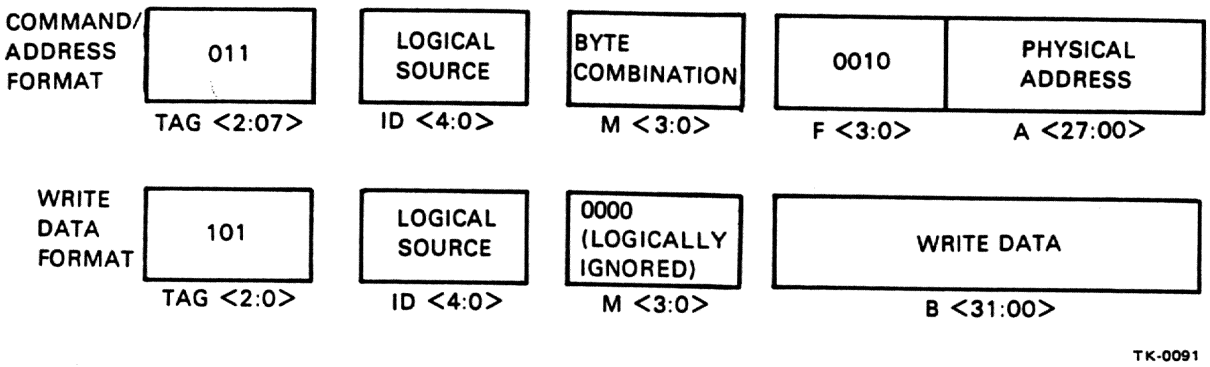


Figure 2-23 Write Masked Function Format

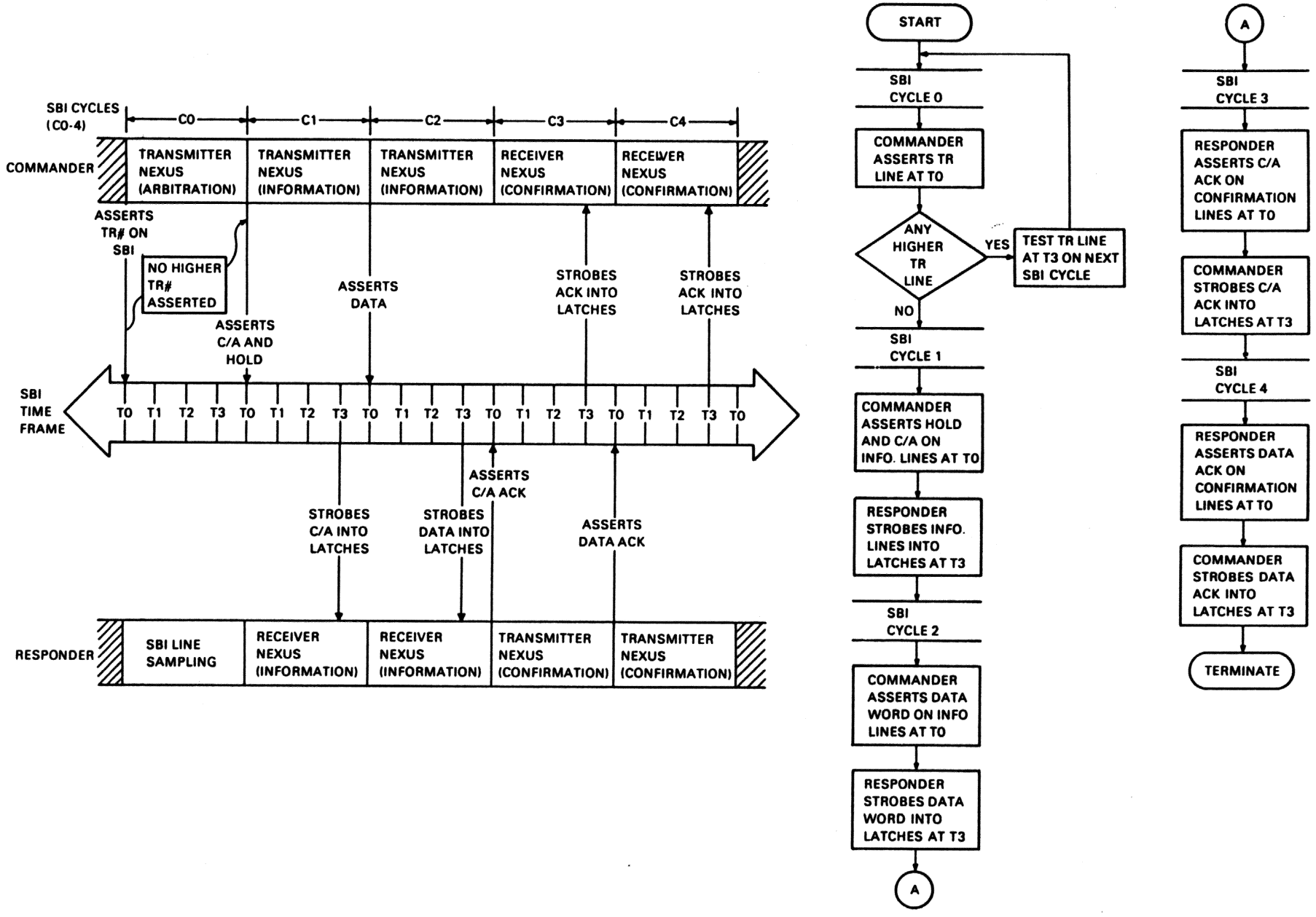


Figure 2-24 Write Masked Timing Chart and Flow

2.2.8.4 Extended Write Masked Function – The extended write masked function format is illustrated in Figure 2-25. F (3:0) is coded 1011 to specify the extended write masked function. In the extended write masked transfer, the number of bits written depends on the mask, but two SBI data transfer cycles are always required. When the commander gains control of the SBI it asserts the C/A information at T0. The commander also asserts TR00 to retain control during the succeeding SBI cycle. At T3 of the same cycle, each nexus strobes the C/A information into its latches for decoding. The mask that accompanies the C/A indicates the bytes to be written in the first data longword, corresponding to SA00 = 0.

At T0 of the succeeding cycle, the commander asserts data on B (31:00) and codes TAG (2:0) as 101 (write data format). At T3 of the same cycle the receiver nexus strobes the data into its latches. In addition, the commander holds TR00 asserted to retain SBI control for the second data longword transfer. Note that the mask that accompanies the first data longword indicates the bytes to be written in the second data longword. At the end of this cycle the commander negates TR00.

At T0 of the succeeding cycle, the second data longword is asserted on B (31:00), and TAG (2:0) is coded 101. At the same time (T0) the receiver nexus confirms the C/A transfer with ACK, if there is no error. At T3 of the same cycle, the receiver nexus strobes the data into its latches. The mask that accompanies the second data longword is ignored by the receiver nexus. During the two succeeding cycles, the receiver nexus confirms the two data transfers with an ACK in each cycle.

Figure 2-26 is a functional timing chart for the extended write masked operation.

2.2.8.5 Interlock Function Description – The interlock function is used to provide coordination between memory nexus to ensure exclusive access to shared data structures. When an interlock sequence is addressed to the UBA, it indicates that a Data-In-Pause/Data-Out (DATIP/DATO) sequence is required on the Unibus. The UBA will initiate an interlock sequence when a Unibus device has initiated a DATIP/DATO sequence. The interlock functions operate like the read and write functions with the additional responsibility of setting and clearing the receiver nexus interlock flip-flop. This flip-flop controls the assertion/negation of the receiver's interlock line. However, not all nexus implement the interlock function. Those nexus that do not will respond to the interlock read and write masked functions exactly as they do to read and write masked functions.

All memory nexus implement the interlock functions and will cooperate through the use of this signal. The interlock line is asserted by the commander nexus which issued the interlock read masked function for that SBI cycle following the C/A transfer. The interlock flip-flop is set in the receiving nexus. When the memory nexus confirms the interlock read function, it asserts the interlock signal in the same cycle as ACK. With interlock asserted, the nexus responds with a BSY confirmation to subsequent interlock read masked commands.

2.2.8.5.1 Interlock Read Masked Function Operation – The interlock read masked function format is the same as that shown in Figure 2-19 except that F (3:0) is coded 0100. F (3:0) causes the nexus selected by SA (27:00) to retrieve and transfer the addressed data exactly as in the read masked operation (Paragraph 2.2.8.1). In addition, this function causes the selected nexus to set its interlock flip-flop. With the interlock flip-flop set, the nexus will assert the SBI interlock line at T0 of the ACK confirmation cycle.

The interlock flip-flop is cleared on receipt of an interlock write masked function. Interlock read masked and interlock write masked functions are always paired by commanders. If the flip-flop remains set for more than 102.4 μs, the responding nexus assumes that the commander has had a catastrophic error. In this case, the nexus will clear the flip-flop at T0 of the next cycle.

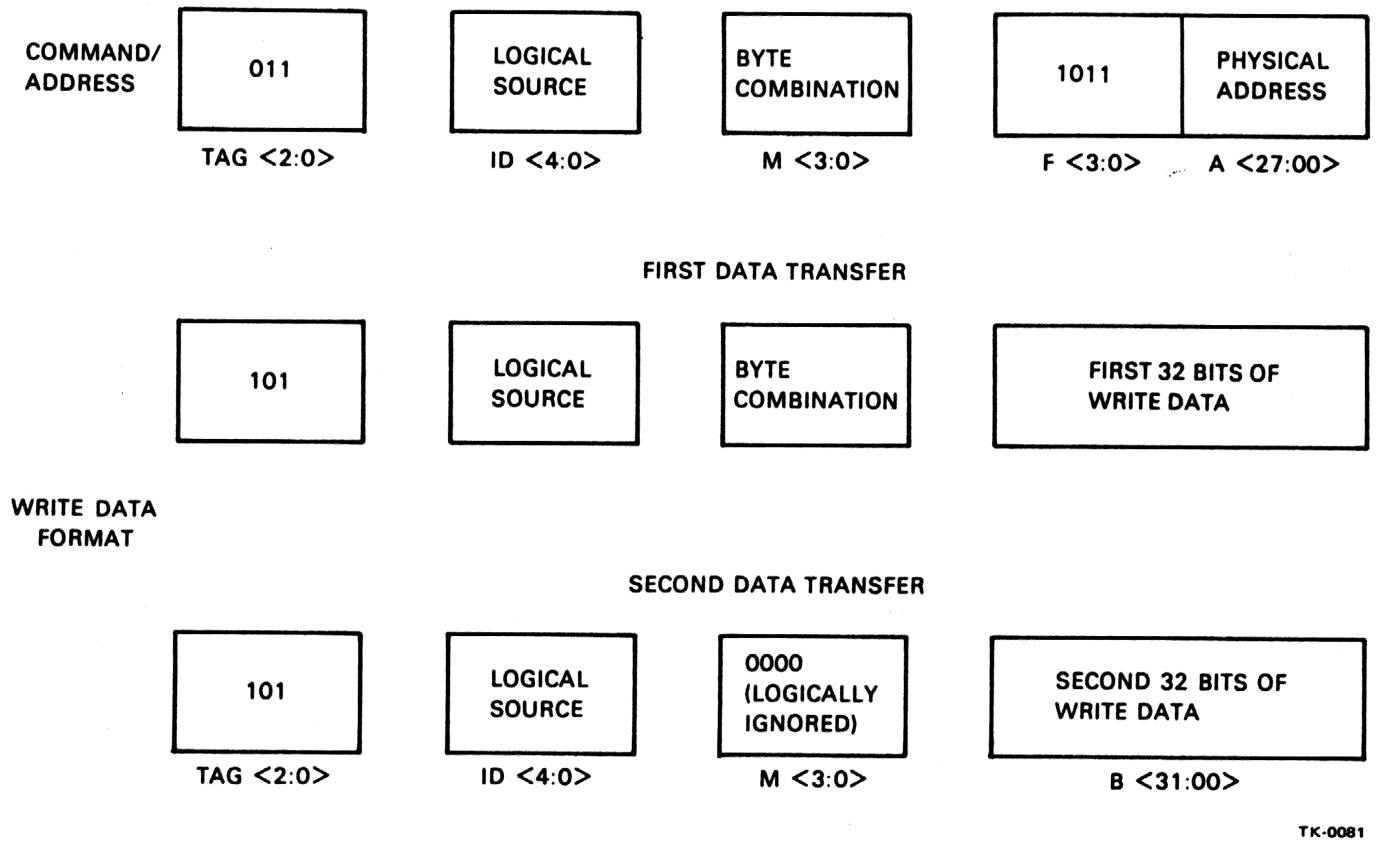


Figure 2-25 Extended Write Masked Function Format

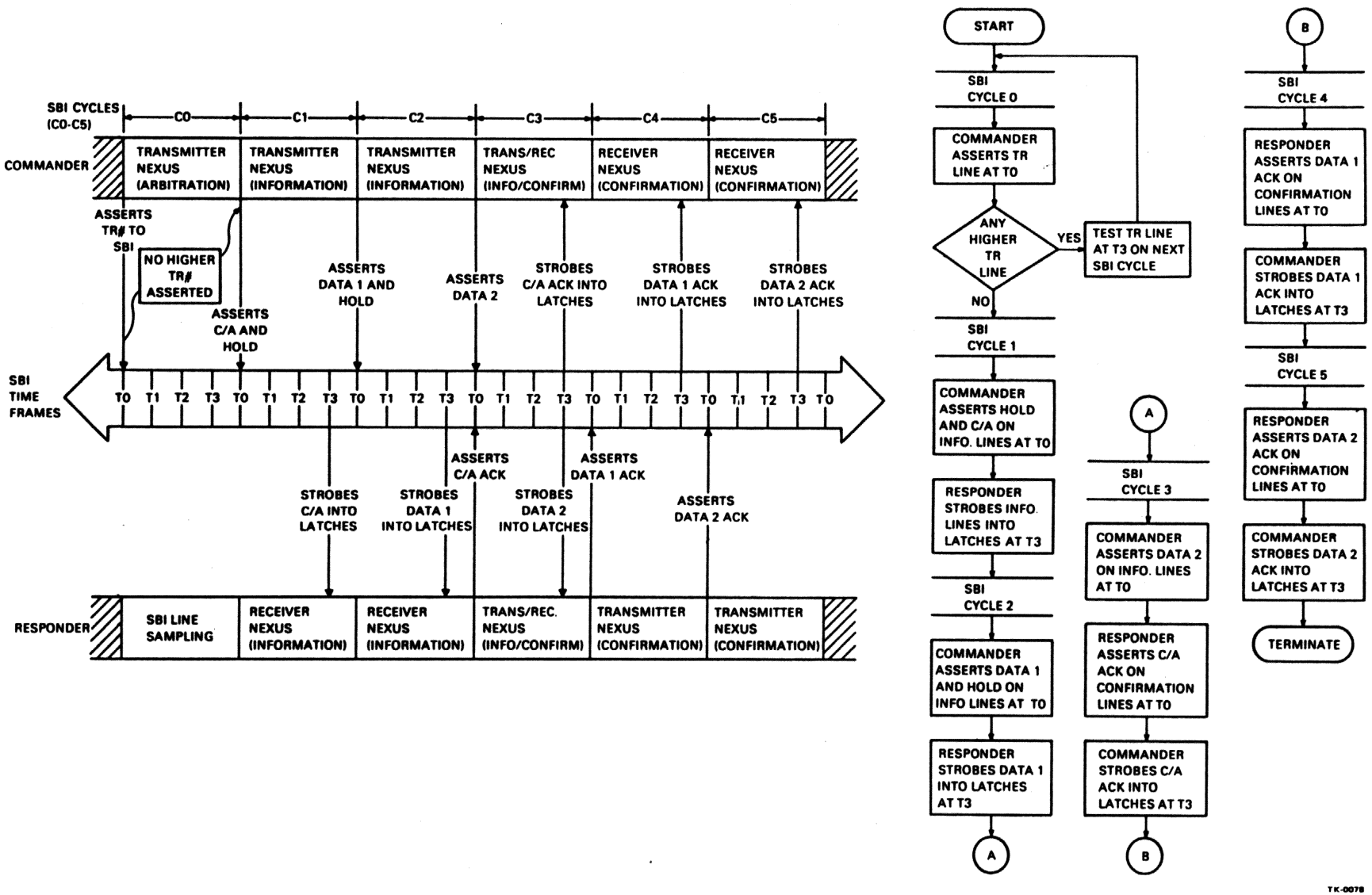


Figure 2-26 Extended Write Masked Timing Chart and Flow

2.2.8.5.2 Interlock Write Masked Function Operation – The interlock write masked function format is the same as that illustrated in Figure 2-23 except that F (3:0) is coded 0111, specifying the interlock write masked function. F (3:0) instructs the nexus selected by SA (27:00) to modify the bytes specified by M (3:0) in the addressed storage element using data transmitted in the succeeding cycle with TAG (2:0) = 101. In addition, the write data clears the interlock flip-flop set by the previous interlock read masked function.

2.2.9 Control Group

The control group functions synchronize system activities and provide specialized system communications. The clock functions provide SBI activity synchronization and are described in Paragraph 2.2.2. The interlock control, also one of the system commmunication functions, is described in Paragraph 2.2.8.5. The remaining control lines are described in the following paragraphs.

2.2.9.1 DEAD Function – The DEAD signal indicates an impending power failure to 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 SBI 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 the clock becomes inoperative. With power restart, the clock will be operational for at least 2 μs before DC LO is negated. The negation of DC LO negates DEAD.

2.2.9.2 Fail Function – A nexus enables the fail (FAIL) function asynchronously to the SBI clock when the power supply AC LO signal is asserted on that nexus. The assertion of FAIL inhibits the CPU from initiating a power up service routine. FAIL is negated asynchronously with respect 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.

2.2.9.3 Unjam Function – The unjam function restores (initializes) the system to a known, well-defined state. The UNJAM signal is asserted only by the CPU or console, and is detected by all nexus connected to the SBI. The console asserts UNJAM only when a console key (or sequence) is selected. The duration of the UNJAM pulse is a minimum of 15 SBI cycles and is negated at T0.

When the console intends to assert UNJAM, the CPU will assert TR00 for a minimum of 15 SBI cycles. The CPU will continue 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.

Each nexus receives UNJAM at T3 time and begins a restore sequence. Any current operation of short duration will not be aborted if that operation might leave the nexus in an undefined state. Nexus will not perform operations using the SBI during the assertion of UNJAM. In addition, the nexus must be in an idle state, with respect to SBI activity, at the conclusion of the UNJAM pulse.

While UNJAM is asserted, nexus will not assert FAULT. However, a nexus asserting FAULT prior to UNJAM must continue to do so to preserve the content of the nexus configuration/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 canceled.

In the event of a power failure during UNJAM, some nexus will assert FAIL and/or DEAD. The restore sequence should cause the nexus negate ALERT or interrupt requests, but should not clear any device status bits.

2.3 UNIBUS SUMMARY DESCRIPTION

2.3.1 General

In the VAX-11/780 system the Unibus connects PDP-11 devices to the UBA. Address, data, and control information are passed over the 56 lines of the Unibus. The form of communication is identical for all devices on the Unibus including the UBA.

The following summary description takes into account the presence of the UBA, since the UBA, together with the Unibus terminator (UBT), performs the PDP-11 CPU Unibus maintenance functions (i.e., arbitration, interrupt vectoring, power fail/restart, and initialization). For example, the UBA enables the system to accept device interrupts and transfer the requests from the Unibus to the SBI. However, UBA and SBI operations between the VAX-11/780 CPU and Unibus are transparent to the Unibus devices.

2.3.2 Unibus Communications/Control

A master/slave relationship defines all communications between devices on the Unibus. The device in control of the bus is considered the master; the device being addressed is the slave. Communication on the bus is interlocked; that is, each control signal issued by the master device must be acknowledged by a corresponding response from the slave to complete the transfer.

2.3.2.1 Bus Request Levels – Each device uses one of two level types for requesting bus control: nonprocessor requests (NPRs) and bus requests (BRs). The NPR is used when a device requests a direct memory or device access data transfer (i.e., a transfer not requiring processor intervention). Normally, NPR transfers are made between a mass storage device (e.g., disk drive) and memory. Two bus lines are associated with the NPR priority level. The device issues its request on the NPR line; the UBA responds by issuing a grant on the nonprocessor grant (NPG) lines.

The BR level is used when a device interrupts the VAX-11/780 CPU to request service. The device may require the CPU to initiate a transfer, or it may need to inform the CPU that an error condition exists. Two lines are associated with each BR level. The bus request is issued on a BR line (BR7–BR4); the bus grant is issued on the corresponding Bus Grant (BG) line (BG7–BG4).

2.3.2.2 Priority Structure and Chaining – When a device requests use of the bus, the handling of that request depends on the location of the device in a two-dimensional device priority structure. Priority is controlled by the priority arbitration logic of the UBA.

The device priority structure consists of five priority levels: NPR, BR7–BR4. Bus requests from devices can be made on any one of the request lines. The NPR has highest priority; BR7 is the next highest priority, and BR4 is the lowest. The priority arbitration logic is structured so that if two devices on different BR levels issue simultaneous requests, the priority arbitration logic grants the bus to the device with the highest priority. However, the lower priority device keeps its request up and will gain bus control when the higher priority device finishes with the bus (providing that no other higher priority device issues a BR).

Since there are only five priority levels, more than one device may be connected to a specific request level. If more than one device makes a request at the same level, the device closest (electrically) to the UBA has highest priority. The grant for each BR level is connected to all devices on that level in a daisy-chain arrangement (chaining). When a corresponding BG is issued it goes to the device closest to the UBA. If that device did not make the request it permits the BG to pass to the next closest device. When the BG reaches the device making the request, that device captures the grant and prevents it from passing on to any subsequent device in the chain. Functionally, NPG chaining is similar to BG chaining.

2.3.2.3 Device Register Organization – The actual transfer of data and status information over the Unibus is accomplished between status, control, and data buffer registers located within the peripheral devices and their control units. All device registers are assigned addresses similar to memory addresses. Thus, all instructions that address memory locations can become I/O instructions.

Control and status functions are assigned to the individual bits within the corresponding addressable registers. Since the register content can be controlled, setting and clearing register bits can control service operations. Internal device status may be loaded into the appropriate register and retrieved when a program instruction addresses that register. Depending on the function, register bits may be read/write, read only, or write only. The number of addressable registers in a device (and control unit) varies, depending on the device's function.

Device registers are assigned addressees within the bus address I/O allocation from $760000_{(8)}$–$777777_{(8)}$ [physical byte addresses $201XE000_{(16)}$–$201XFFFF_{(16)}$, where X may be 3, 7, B or $F_{(16)}$, depending on the UBA in question].

2.3.2.4 Unibus Line Definitions – The Unibus consists of 56 signal lines that may be divided into three function groups: bus control, data transfer, and miscellaneous signals. The 13 lines of the bus control group comprise those signals required to gain bus control through an NPR/BR or for a priority arbitration to select the next bus master while the current bus master is still in control of the bus. The 40 bidirectional lines of the data transfer group are those signals required during data transfers to or from a slave device. The miscellaneous group are the initialization and power fail signals required on the bus. Table 2-4 describes the bus signals within each group. Figure 2-27 shows the Unibus configuration.

2.3.2.5 Unibus Operation Sequencing – The following paragraphs provide a summary description of the Unibus operation sequences. The timing diagrams provided in the descriptions do not reflect actual timing, only general signal relationships. If additional detail is required (e.g., skew times, logic structures, etc.), refer to the appropriate PDP-11 device maintenance manual or Chapter 5 of the *PDP-11 Peripherals Handbook.*

2.3.2.5.1 Priority Arbitration – The priority arbitration operation involves the signal sequence that selects the next bus master. The operation does not actually transfer bus control but merely selects the next bus master.

Table 2-4 Unibus Signal Descriptions

Signal Line	Description
Data Transfer Group	
Address Lines [SA (17:00)]	These lines are used by the master device to select the slave (actually a unique memory or device register address). SA (17:01) specifies a unique 16-bit word; SA00 specifies a byte within the word.
Data Lines [D (15:00)]	These lines transfer information between master and slave.
Control (C1, C0)	These signals are coded by the master device to control the slave in one of the four possible data transfer operations specified below. Note that the transfer direction is always designated with respect to the master device.

Table 2-4 Unibus Signal Description (Cont)

Signal Line	Description
	Data Transfer Designation Description
CI CO 0 0	Data In (DATI): a data word or byte transferred into the master from the slave.
0 1	Data In Pause (DATIP): similar to DATI except that it is always followed by a DATO/B to the same location.
1 0	Data Out (DATO): a data word is transferred out of the master to the slave.
1 1	Data Out Byte (DATOB): identical to DATO except a byte is transferred instead of a full word.
Parity A–B (PA, PB)	These signals transfer Unibus parity information. PA is currently unused and not asserted. PB, when true, indicates a device parity error.
Master Synchronization (MSYN)	MSYN is asserted by the master to indicate to the slave that valid address and control information (and data on a DATO or DATOB) is present on the bus.
Slave Synchronization (SSYN)	SSYN is asserted by the slave. On a DATO it indicates that the slave has latched the write data. On a DATI/P it indicates that the slave has asserted read data on the Unibus.
Interrupt (INTR)	This signal is asserted by an interrupting device, after it becomes bus master, to inform the UBA that an interrupt is to be performed, and that the interrupt vector is present on the D lines. INTR is negated upon receipt of the assertion of SSYN by the UBA at the end of the transaction. INTR may be asserted only by a device that obtained bus mastership under the authority of a BG signal.
Priority Arbitration Group	
Bus Request (BR7–BR4)	These signals are used by peripheral devices to request control of the bus for an interrupt operation.
Bus Grant (BG7–BG4)	These signals form the CPU and UBA response to a bus request. Only one of the four will be asserted at any time.

Table 2-4 Unibus Signal Description (Cont)

Signal Line	Description
Priority Arbitration Group (Cont)	
Nonprocessor Request (NPR)	This is a bus request from a device for a transfer not requiring CPU intervention (i.e., DMA).
Nonprocessor Grant (NPG)	This is the grant in response to an NPR.
Selection Acknowledge (SACK)	SACK is asserted by a bus-requesting device after having received a grant. Bus control passes to this device when the current bus master completes its operation.
Bus Busy (BBSY)	BBSY indicates that the data lines of the bus are in use. It is asserted by the Unibus master.
Initialization Group	
Initialize (INIT)	This signal is asserted by the terminator board (UBT) when DC LO is asserted on the Unibus, and it stays asserted for 10 ms following the negation of DC LO.
AC Line Low (AC LO)	This is an anticipatory signal that warns of an impending power failure. AC LO initiates the power fail trap sequence and may also be issued in peripheral devices to terminate operations in preparation for power loss.
DC Line Low (DC LO)	This signal is available from each system power supply and remains clear as long as all dc voltages are within the specified limits. If an out-of-voltage condition occurs, DC LO is asserted.

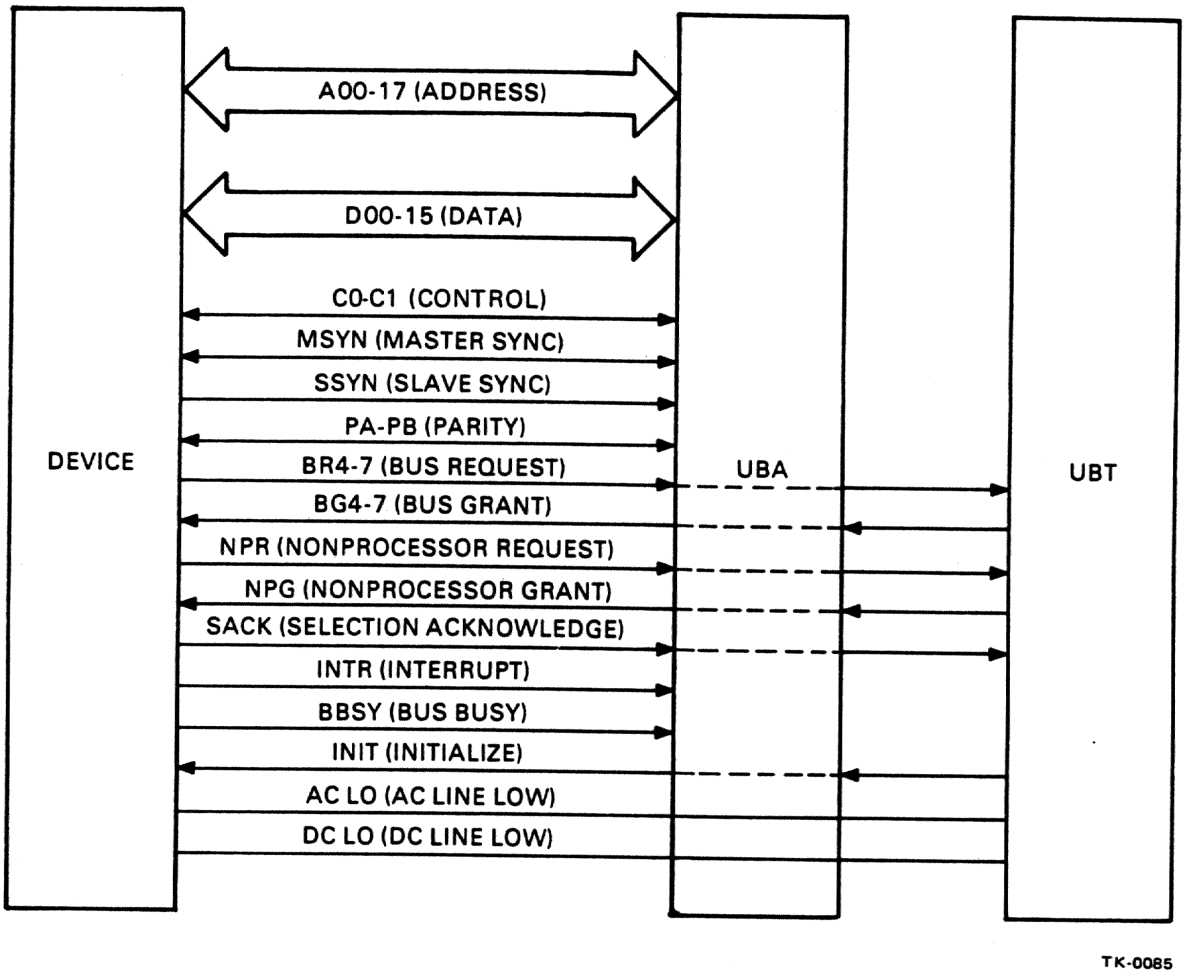


Figure 2-27 Unibus Configuration

The device requiring service asserts its BR (or NPR) line. In practice, the UBA may be receiving several simultaneous BR signals. These signals enter the Unibus arbitration logic of the UBA. If enabled by software, the UBA then conducts a dialogue with the CPU and asserts the corresponding BG (or NPG) line. The grant is propagated through each device on the asserted BG line. The first device on the line having BR asserted acknowledges the grant by asserting SACK, blocks the grant from following devices, and clears its BR. The UBA responds to SACK by clearing BG. If SACK is not asserted by the requesting device, BG is ORed with the other grant lines and returned as a SACK signal to clear BG.

The device will keep SACK asserted until the current bus master relinquishes bus control by clearing its BBSY. SACK asserted prevents other devices from gaining bus control. Once the current bus master has relinquished the bus and negated BBSY, the requesting device asserts BBSY and negates SACK, becoming the new bus master. Priority arbitration can be performed at the same time as a data transaction or the servicing of an interrupt. While one device is using the bus, the arbitration logic is free to monitor other requests and issue an appropriate grant.

Figure 2-28 provides a functional timing diagram for the priority transfer sequence.

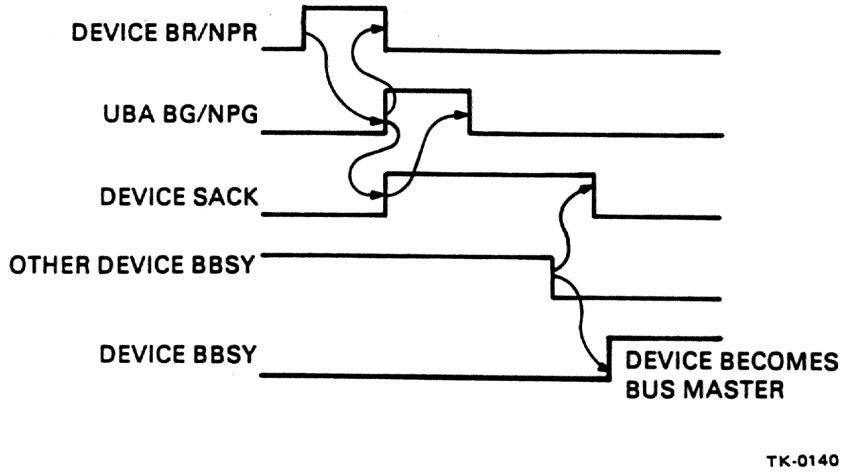


Figure 2-28 Priority Transfer Sequence

2.3.2.5.2 Data Transfer – Once the device has become bus master through an NPR or BR sequence (asserting BBSY and clearing SACK), it places the appropriate address and control information on the bus. The master will place data on the bus at this time, as well, if it is performing a write transfer (DATO/B). No data will be asserted this time if the master is performing a read (DATI/DATIP). The address asserted selects the desired slave device I/O register or memory location, and the control lines, C1 and C0, select the transfer type. Note that the memory will be the slave device in any transfer to or from memory. Following a delay to allow for bus line settling and address decoding, the master asserts MSYN. MSYN is a strobe signal for the address and control lines and is always cleared before these lines are changed. On a write transfer, when the slave acknowledges MSYN with SSYN, the slave is indicating that the address and control signals have been decoded and that the data has been latched. Figure 2-29 shows the timing involved.

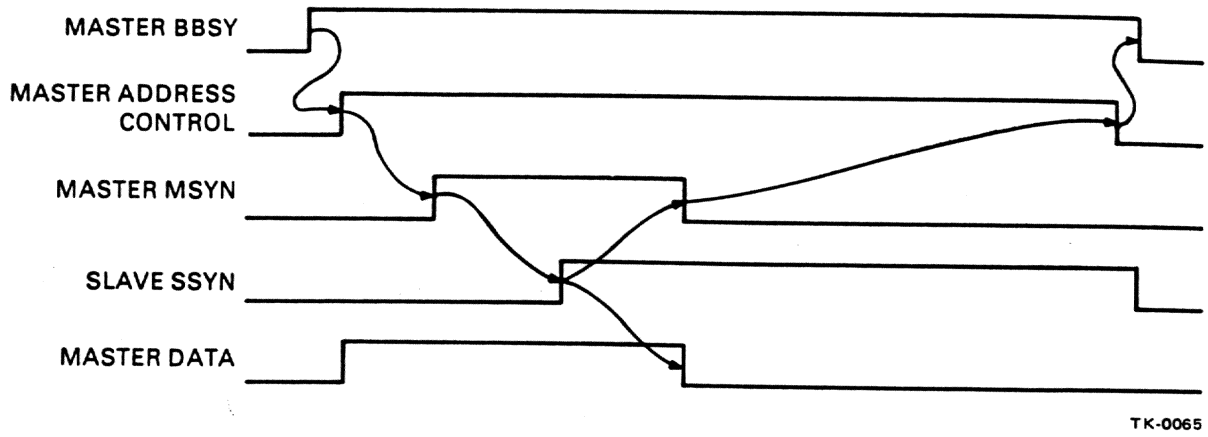


Figure 2-29 Typical DATO Transaction, Timing Diagram

On a read transfer, the slave acknowledges MSYN with SSYN when it has decoded the address and control signals, retrieved the requested data, and asserted the data on the Unibus. The master responds to SSYN by clearing MSYN. Following a specific time delay the master clears the address and control lines. With the clearing of MSYN the slave responds by clearing SSYN.

On a DATI transfer

1. The master asserts only address and control information.
2. The slave gates data onto the bus and asserts SSYN.
3. The master allows a time delay (for skew between SSYN and the data) before clearing MSYN and strobing data.
4. The slave clears the data and clears SSYN.

Figure 2-30 is a timing diagram showing the DATI transaction sequence.

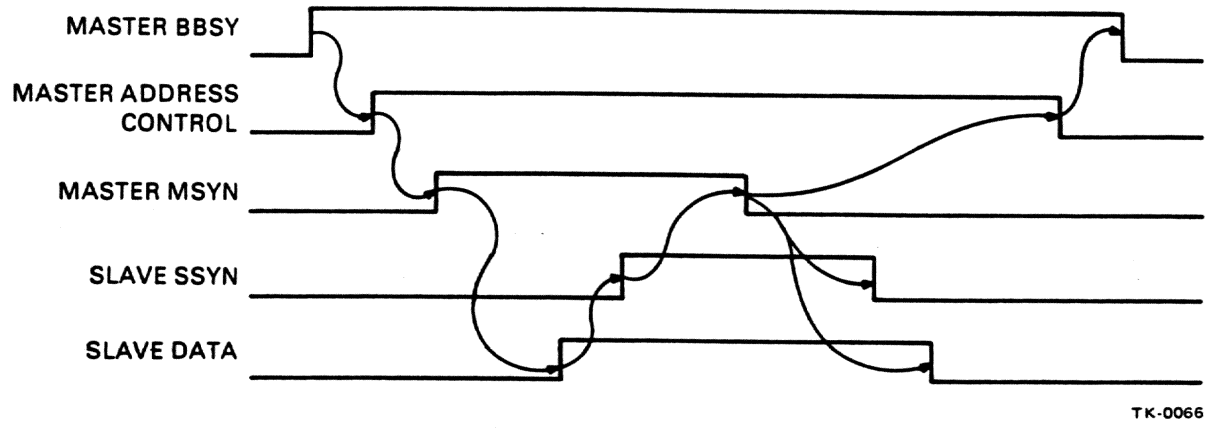


Figure 2-30 Typical DATI Transaction, Timing Diagram

2.3.2.5.3 DATIP-DATO/B Sequence – The UBA and some Unibus I/O devices are capable of performing the DATIP/DATO sequence. This sequence constitutes a read-modify-write process in which the master accesses the same location on both the read and the write transfers. Note that the DATIP operation must be followed by a DATO or a DATOB.

2.3.2.5.4 Interrupt Vector Handling – Unibus interrupts are handled in one of two modes depending on the Unibus configuration. If the VAX-11/780 CPU is servicing Unibus interrupts, the UBA accepts the BR. It then translates the BR to an SBI request and passes the interrupt vector to the software as part of the Unibus interrupt service routine. This transaction is discussed completely in Paragraphs 3.5.4 and 3.5.5.

If the CPU is not servicing interrupts, interrupt requests will be ignored.

Figure 2-31 shows the timing for the interrupt sequence.

2.3.2.5.5 Register/Vector/Priority Assignments – Table 2-5 lists the device register and vector addresses and priority levels assigned to two representative Unibus devices. The assignments follow the standard PDP-11 configuration rules for all options. The physical byte addresses are given for the Unibus connected to UBA0.

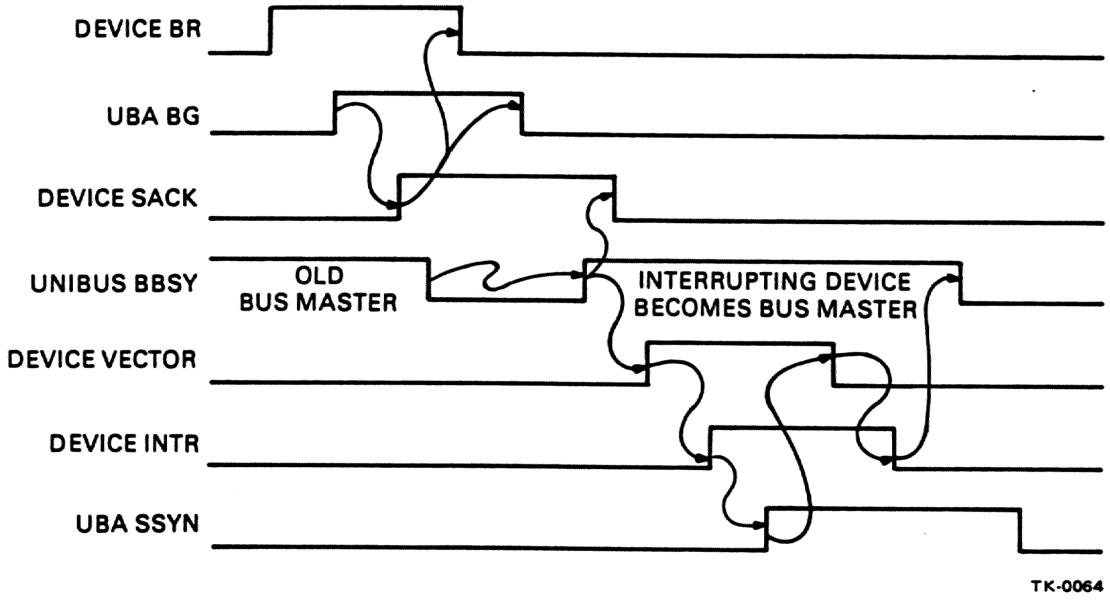


Figure 2-31 Typical Interrupt Transaction, Timing Diagram

Table 2-5 Representative Unibus Device Register, Vector and Priority Assignments

Equipment Option	VAX-11/780 System Physical Byte Address (hex)	Unibus Address (octal)	Register Mnemonic (hex)	Vector Address Assignment (octal)	Interrupt Priority Level
CR11				98 230	BR6
	2013FE70	777160	CRS		
	2013FE72	777162	CRB1		
	2013FE74	777164	CRB2		
LP11				80 200	BR4
	2013FF4C	777514	LPS		
	2013FF4E	777516	LPB		

2.4 UBA BLOCK DIAGRAM

The block diagram given in Figure 2-32 shows the relationships of the major circuits on the UBA and their location on the UBA modules. Notice that the SBI and the SBI interface circuits are shown on the left. The Unibus and the Unibus interface circuits are shown on the right. Data, mask, address, and request signals are routed between UBA modules via the tristate buses listed below.

Bus IB (31:00)
Bus T (31:00)
Bus AD (15:00)
Bus DOUT (15:00)
Bus REQ MSK (3:0)
Bus MSK SB (3:0)

Variations on this diagram are distributed throughout Chapters 2 and 3 of this manual so that the circuits under discussion can be located within the framework of the UBA as a whole.

2.5 UNIBUS SUBSYSTEM ADDRESS SPACE

Each UBA contains two address spaces within the SBI I/O address space. One address space is located within the nexus register address space and contains all registers on the UBA. The other address space is the Unibus address space assigned to each of the UBAs.

2.5.1 Accessing UBA Register Address Space

Each nexus register address space occupies 16 pages of SBI I/O address space and is determined by the TR number of the device as specified in the SBI specification. This TR number is determined by jumpers on the nexus backplane.

The SBI C/A format for accessing UBA registers is shown in Table 2-6.

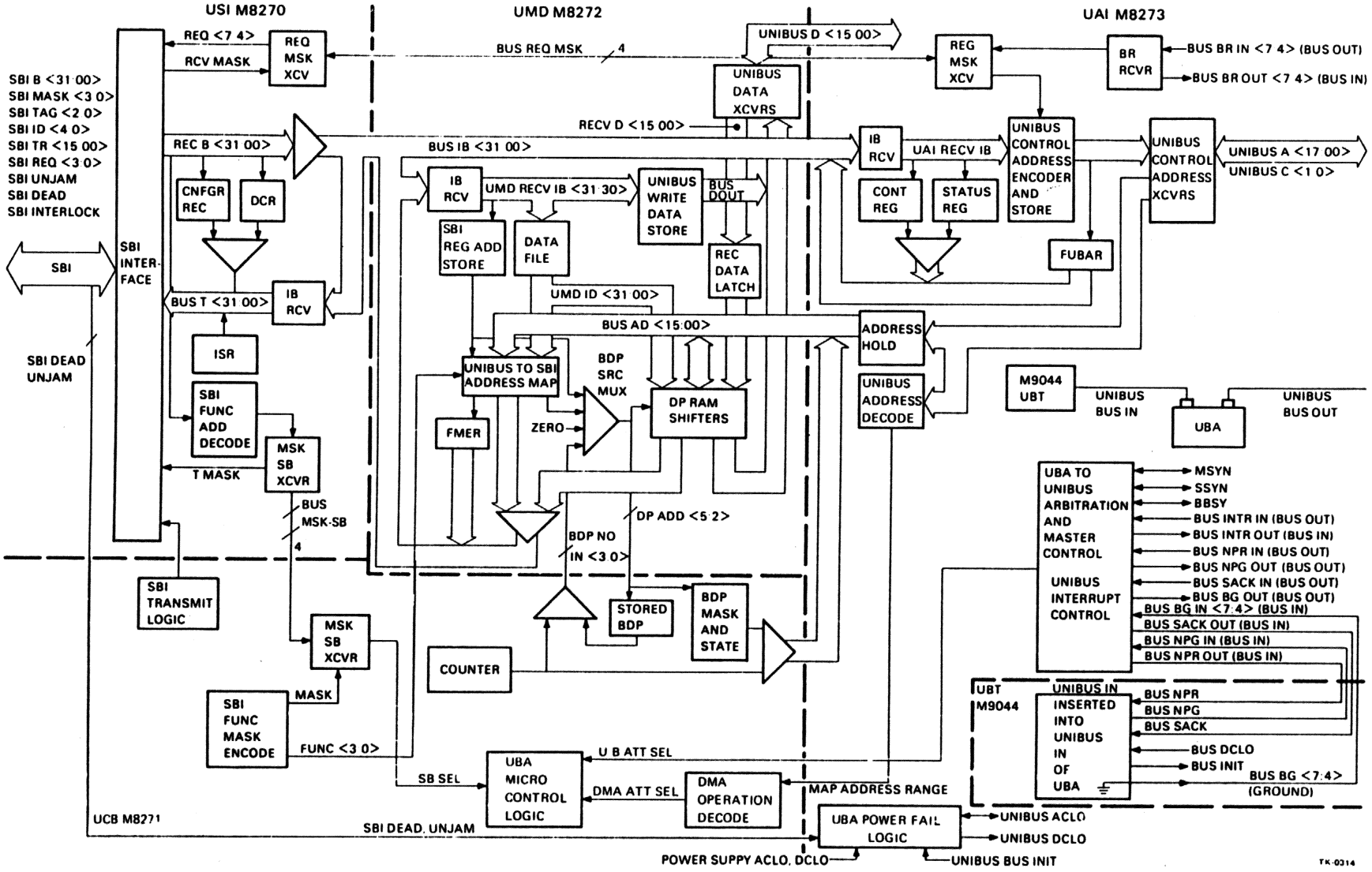
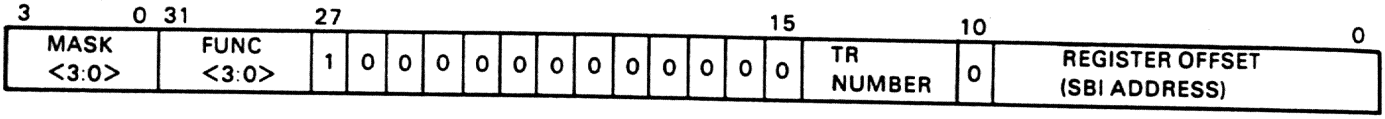


Figure 2-32 UBA Block Diagram

Table 2-6 UBA Address Space and C/A Format

SBI C/A Format for UBA Register Access



TK-032d

Base Addresses

TR Num Base 10	Base Address (Physical hex)	SBI Address (hex)	TR Num Base 10	Base Address (Physical hex)	SBI Address (hex)
0	20000000	8000000	8	20010000	8004000
1	20002000	8000800	9	20012000	8004800
2	20004000	8001000	10	20014000	8005000
3	20006000	8001800	11	20016000	8005800
4	20008000	8002000	12	20018000	8006000
5	2000A000	8002800	13	2001A000	8006800
6	2000C000	8003000	14	2001C000	8007000
7	2000E000	8003800	15	2001E000	8007800

Register Offsets

UBA Reg	Byte Address (Physical hex)	SBI Address (hex)	UBA Reg	Byte Address (Physical hex)	SBI Address (hex)
CNFGR	000	000	.	.	.
UBACR	004	001	.	.	.
UBASR	008	002	DPR 14	078	01E
DCR	00C	003	DPR 15	07C	01F
FMER	010	004	Reserved	080	020
FUBAR	014	005	.	.	.
FMER	018	006	.	.	.
FUBAR	01C	007	Reserved	7EC	1FF
BRSVR 0	020	008	MR 0	800	200
BRSVR 1	024	009	MR 1	804	201
BRSVR 2	028	00A	.	.	.
BRSVR 3	02C	00B	.	.	.
BRRVR 4	030	00C	MR 494	EB8	3EE
BRRVR 5	034	00D	MR 495	EBC	3EF
BRRVR 6	038	00E	Reserved	EC0	3F0
BRRVR 7	03C	00F	.	.	.
DPR 0	040	010	Reserved	EFC	3FF
DPR 1	044	011			

The offset within the UBA address space is also shown for each of the UBA registers with respect to the physical address space. Note that the UBA registers are longword aligned and can only be accessed as longwords. Since the SBI address space is longword aligned and the physical address space is byte aligned, the relation between an SBI address (SA) and a physical address (PA) is as follows:

SA (27:00) = PA (29:02), or SA = PA/4 (shift twice)

The UBA registers are described in Paragraph 2.9.

2.5.2 Accessing the Unibus Address Space

The UBA provides for SBI access to all Unibus device registers and Unibus memory. During such a transfer, the UBA becomes the highest priority Unibus NPR device. A portion of the SBI I/O address space is designated as the Unibus address space.

The Unibus address space for each UBA occupies 512 pages of the SBI I/O address space, 496 pages of Unibus memory address space (reserved), and 16 pages of Unibus I/O address space.

SBI transfers to Unibus addresses for which the map registers have been enabled will be mapped (wrapped around) into the SBI address space.

2.5.2.1 SBI to Unibus Address and Function Translation – Table 2-7 shows the relation of SBI mask and function combinations to Unibus control and address combinations. Note that extended transfers cannot be made to either the Unibus address space or the UBA registers.

Only the function byte mask combinations shown will be valid. All other function byte mask combinations addressed to the Unibus address space will be given an ERR confirmation. The Unibus address space will respond only to word or byte SBI references.

Figure 2-33 shows the SBI command/address format for accessing the Unibus address space for UBAs 0 through 3. Each SBI address (longword address) covers two 16 bit Unibus addresses (word addresses). In addition to the SBI address being decoded, the SBI function and byte mask is decoded to determine the word or byte to be accessed.

When a Unibus device register is accessed from the SBI, the Unibus address is translated from the SBI address and function, as follows. If SBI address bits SA27 and SA18 are true, and bits SA (26:19) are false, the UBA selected by SA (17,16) will pass the lower 16 SBI address bits through to the Unibus as Unibus address bits UA (17:00). The UBA generates Unibus address bits UA (1,0) and control bits C (1,0) by decoding the SBI mask and function bits. Table 2-8 shows the relationship of the Unibus space controlled by UBA0 to the SBI address space.

2.5.2.2 SBI to Unibus Transfer: Data Route on an SBI Write – When an SBI commander nexus initiates a write transfer to a Unibus device, the following sequence occurs.

The commander asserts the mask and function codes as parts of the C/A word. On the next SBI cycle it asserts the data to be written (one or two bytes). The C/A information is accepted on the UBA transceivers and gated to the internal bus, BUS IB (31:00) H. The UBA decodes the mask and function bits to produce address bits 1,0 on the Unibus. Unibus address bit 1 selects either the high or low 16-bit word of the data gated into the 2:1 multiplexer shown in Figure 2-34.

The UBA arbitrates for control of the Unibus, and when the UBA becomes master, the UBA asserts the translated Unibus and the data to be written, together with master sync, beginning a DATO or a DATOB cycle on the Unibus. The Unibus device addressed then latches the data and responds with slave sync, completing the transfer. Figure 2-35 shows the relation of this circuit to the UBA block diagram.

Table 2-7 SBI Function – Mask Translation to Unibus Control – Address

SBI	Unibus		
Function (3:0)	**Mask 3 2 1 0**	**Control C(1:0)**	**Address UA(1:0)**
Read Mask	0 0 0 1	DATI	0 0
	0 0 1 1	DATI	0 0
	0 0 1 0	DATI	0 0
	0 1 0 0	DATI	1 0
	1 1 0 0	DATI	1 0
	1 0 0 0	DATI	1 0
Write Mask	0 0 0 1	DATOB	0 0
	0 0 1 0	DATOB	0 1
	0 1 0 0	DATOB	1 0
	1 0 0 0	DATOB	1 1
	0 0 1 1	DATO	0 0
	1 1 0 0	DATO	1 0
Interlock Read Mask (Sets Interlock flip-flop for DATIP-DATO Sequence)	0 0 0 1	DATIP	0 0
	0 0 1 0	DATIP	0 0
	0 1 0 0	DATIP	1 0
	1 0 0 0	DATIP	1 0
	0 0 1 1	DATIP	0 0
	1 1 0 0	DATIP	1 0
Interlock Write Mask	0 0 0 1	DATOB	0 0
	0 0 1 0	DATOB	0 1
	0 1 0 0	DATOB	1 0
	1 0 0 0	DATOB	1 1
	0 0 1 1	DATO	0 0
	1 1 0 0	DATO	1 0

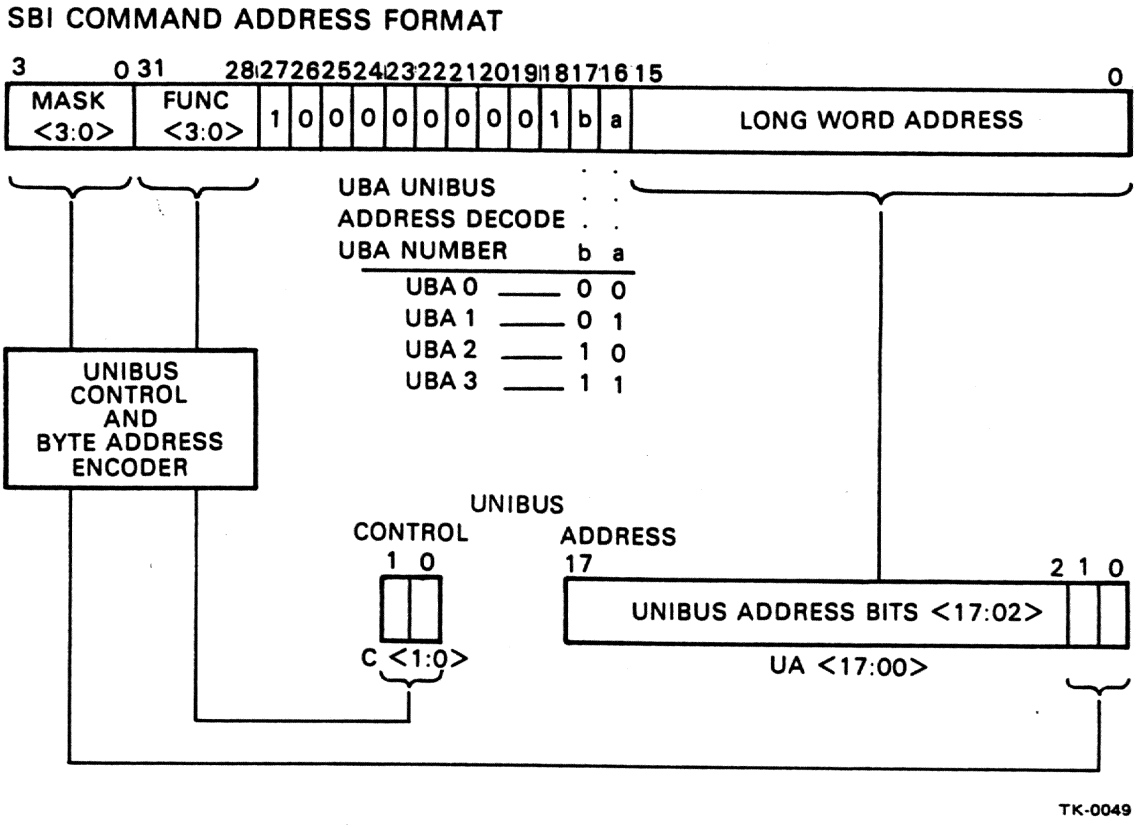


Figure 2-33 SBI to Unibus Control Address Translation

Table 2-8 Unibus and SBI Address Spaces

System Address Space (not to scale)	SBI Longword Address (Hex)	System Physical Byte Address (Hex)	Unibus Address Space (Hex)	Unibus Address Space (Octal)	
Memory Address Space	8040000	20100002 20100000	00002 00000	000002 000000	Unibus Memory Address Space Reserved For Expansion (496 pages) 1 page = 512 bytes
	8040001	20100006 20100004	00006 00004	000006 000004	
	8040002	2010000A 20100008	0000A 00008	000012 000010	
	8040003	2010000E 2010000C	0000E 0000C	000016 000014	
	8040004	20100012 20100010	00012 00010	000022 000020	
Other Adaptor Registers	.	. .	. .	. .	
	.	. .	. .	. .	
	.	. .	. .	. .	
	804F7FD	2013DFF6 2013DFF4	3DFF6 3DFF4	757766 757764	
	804F7FE	2013DFFA 2013DFF8	3DFFA 3DFF8	757772 757770	
	804F7FF	2013DFFE 2013DFFC	3DFFE 3DFFC	757776 757774	
Unibus Adaptor Registers	804F800	2013E002 2013E000	3E002 3E000	760002 760000	Unibus I/O Address Space
	804F801	2013E006 2013E004	3E006 3E004	760006 760004	
Other Adaptor Registers	804F802	2013E00A 2013E008	3E00A 3E008	760012 760010	
	.	. .	. .	. .	
	.	. .	. .	. .	
	.	. .	. .	. .	
Unibus I/O Address Space	804F804	2013E013 2013E012 2013E011 2013E010	3E013 3E012 3E011 3E010	760023 760022 760021 760020	example for DATOB
	.	. .	. .	. .	
	.	. .	. .	. .	
	.	. .	. .	. .	Upper 4K (10) 16 bit words (16 pages)
	.	. .	. .	. .	
Other I/O Address Space	804FFFD	2013FFF6 2013FFF4	3FFF6 3FFF4	777766 777764	
	804FFFE	2013FFFA 2013FFF8	3FFFA 3FFF8	777772 777770	
	804FFFF	2013FFFE 2013FFFC	3FFFE 3FFFC	777776 777774	

Note: These addresses refer to UBA ϕ.

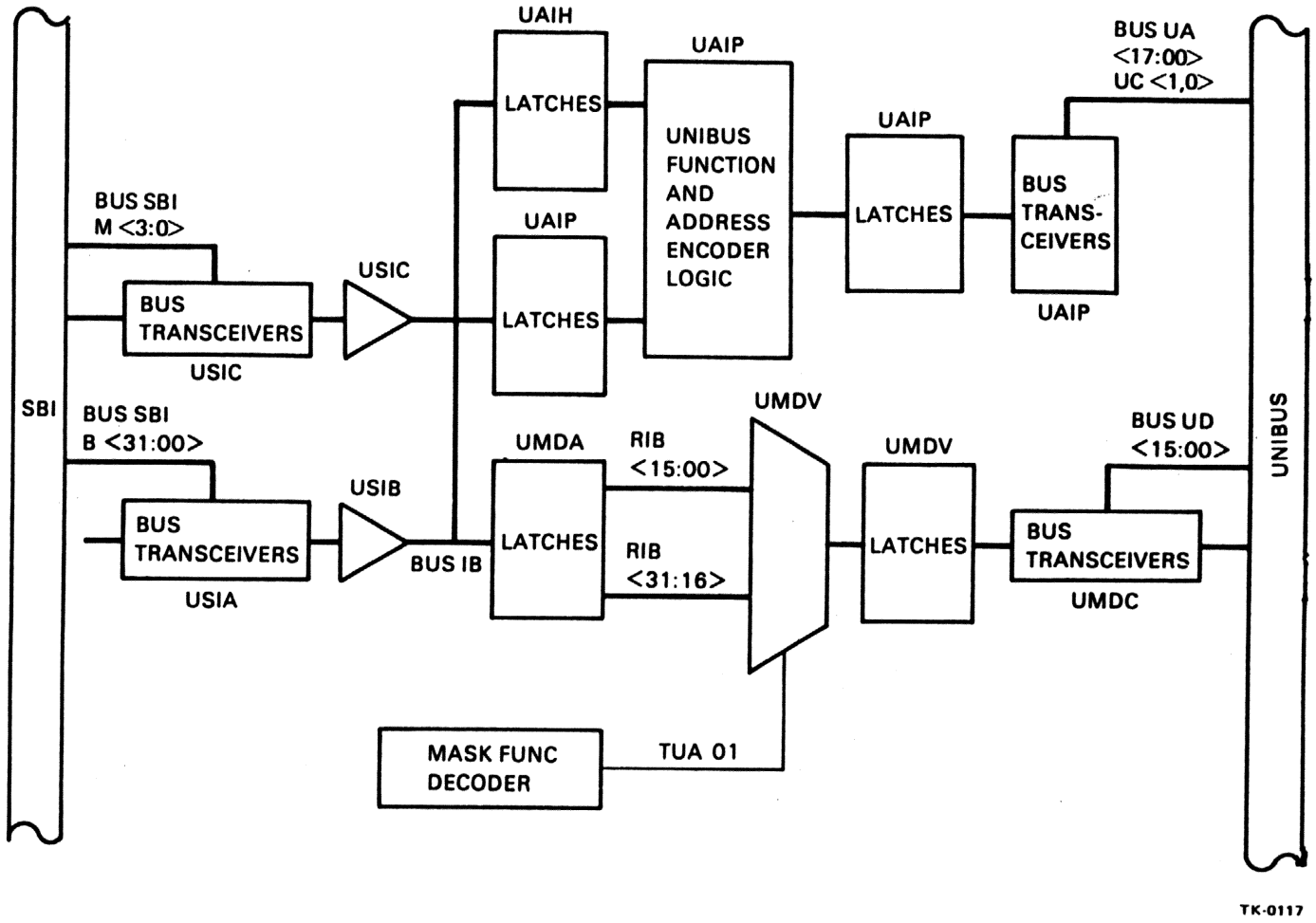


Figure 2-34 Data Route for Write Transfer to Unibus Initiated on the SBI, Block Diagram

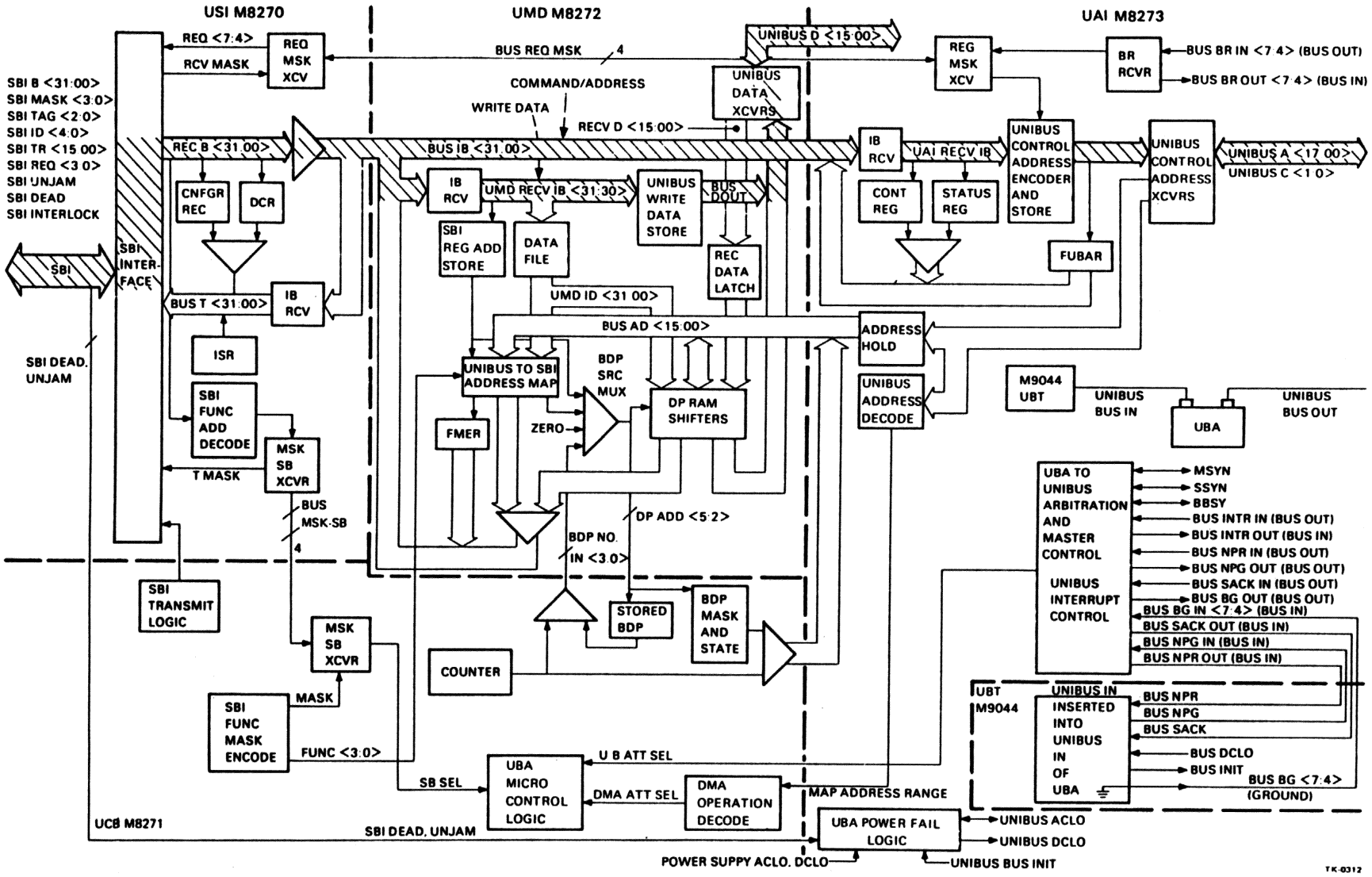


Figure 2-35 Data Route for Write Transfer to a Unibus Device/UBA, Block Diagram

2.5.2.3 SBI to Unibus Transfer: Data Route on an SBI Read – When an SBI commander nexus initiates a read transfer to read data from a Unibus device, the UBA accepts the C/A information and arbitrates for the Unibus in order to begin a DATI sequence. The address information is handled on a read transfer as on the write transfer previously described. When the UBA gains control of the Unibus, it asserts the address of the device register or location to be read, together with the control signals C0 and C1 and master sync. When the Unibus device responds by asserting the data and slave sync on the Unibus, the UBA latches the Unibus data and then selects the data on its 4:1 data path multiplexer, as shown in Figure 2-36. The state of Unibus address bit 1 causes the UBA to gate the data in the high or the low portion of the longword through the data path multiplexer to the internal bus [BUS IB (31:00) H]. The remaining two bytes will contain zeros. Then, when the UBA gains control of the SBI, the microsequencer enables the read data on the SBI, completing the SBI transfer. Figure 2-37 shows the data route for a read transfer to the Unibus in relation to the UBA as a whole.

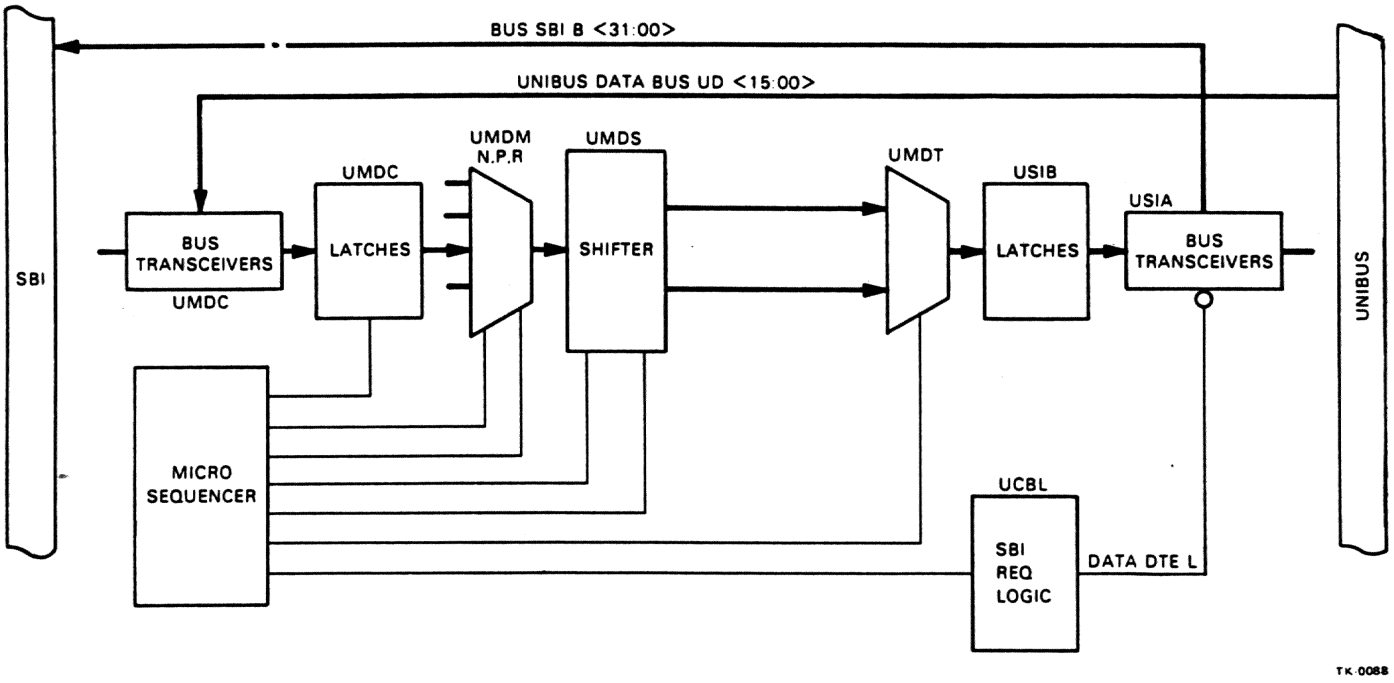


Figure 2-36 Data Route for Read Transfer to the Unibus Initiated on the SBI, Block Diagram

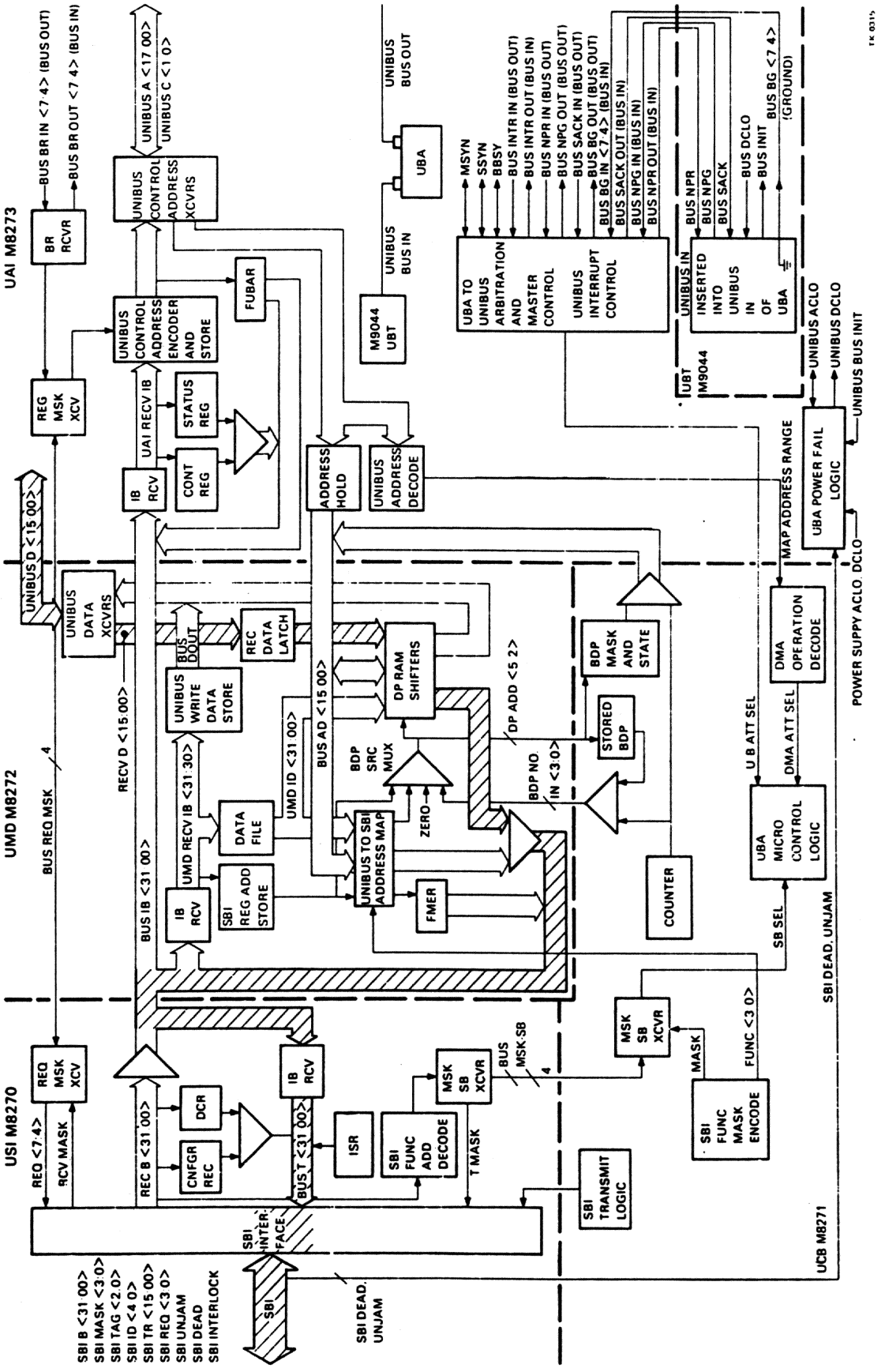


Figure 2-37 Data Route for Read Transfer to the Unibus/UBA Block Diagram

2.5.2.4 SBI to Unibus Transfer Failures – If during a read sequence to the Unibus address space, data is received from the Unibus device with Unibus PB asserted (Unibus Memory Parity Error), then the data will be sent to the SBI as a read data substitute.

If, for some reason, an access is made to the Unibus address space and the transfer is not completed on the Unibus (e.g., non-existent device), the following will occur.

1. An all zeros word will be sent as a read data for a read transfer.

2. The Unibus address bits (17:02) will be stored in the failed Unibus address register (FUBAR).

3. The bit indicating the cause of failure (UBA Select Time Out or SSYN Time Out) will be set in the UBA status register. Note that in the case of a write transfer to the Unibus, the error bit is set either 13 or 50 μs after the command was issued and acknowledged by the UBA, (13 μs for SSYN timeout, 50 μs for select Time Out). Therefore, it will not be immediately known to the software. If the software has set the SUEFIE (SBI to UNIBUS Error Interrupt Enable), the setting of UB Select Time Out or SSYN Time Out will initiate an adaptor interrupt request.

2.6 UNIBUS ACCESS TO THE SBI ADDRESS SPACE

The UBA provides for DMA transfers to SBI memory. The UBA contains map registers to map Unibus memory page addresses to SBI page addresses. It contains the data transfer paths required to transfer data between the Unibus and SBI.

2.6.1 Unibus to SBI Address Translation

The UBA translates Unibus memory addresses to SBI addresses through a Unibus to SBI address translation map. Each Unibus address is mapped to an SBI address in three sections:

1. SBI page address (1 page equals 512 bytes)
2. Longword within an SBI page (1 longword equals 4 bytes)
3. Word or byte within a longword.

NOTE

To avoid confusion between Unibus and SBI address bits, Unibus address bits will be shown as UA (bit num) and SBI address bits will be shown as SA (bit num).

As shown in Figure 2-38, the Unibus to SBI page map translates Unibus memory page addresses to any SBI page address. The map allows the transfer of data to discontiguous pages of SBI memory. The map translates the nine Unibus page address bits [UA (17:09)] to the 21 SBI page address bits [SA (27:07)].

There are 496 map registers provided to map the entire Unibus memory address space at once, thereby reducing the problem of register allocation. Each map register corresponds to the Unibus page that is to be mapped. The map registers are available to VAX-11 software as part of the SBI I/O address space.

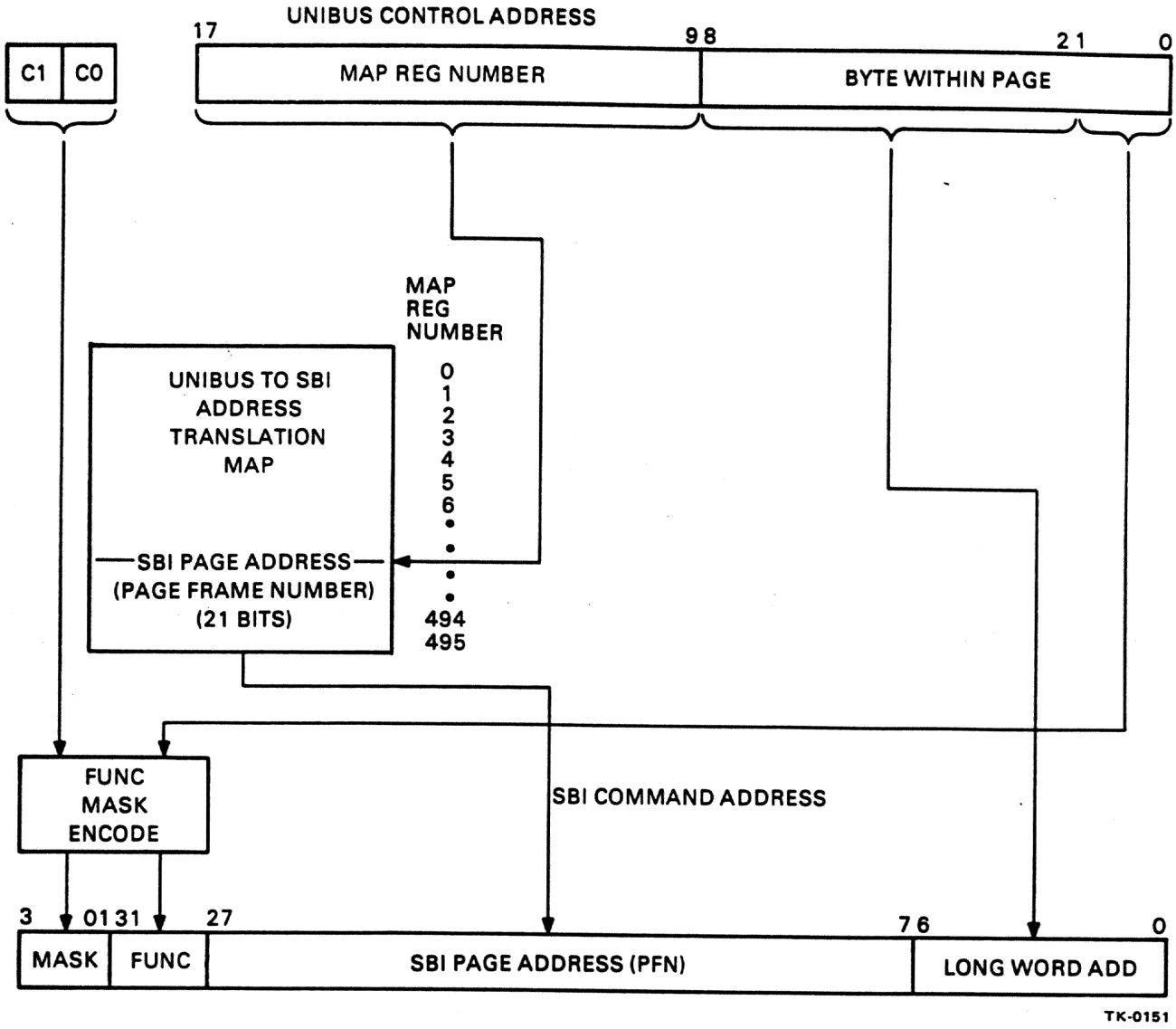


Figure 2-38 Unibus to SBI Address Translation

Each of the map registers contains the following fields:

- The SBI page to which the transfer will be mapped.
- The data path number to be used for the transfer.
- A byte offset bit which, when set, will increase the SBI memory address of a Unibus transfer by one byte to enable byte-aligned transfers from Unibus word transfer devices. Note that the byte offset is recognized only on transfers through the BDPs and to SBI memory space.
- A valid bit to indicate that the software has set up the map register.

If, during a DMA transfer, the Unibus address points to an invalid map register or a map register that has a parity error within the high-order 16 bits, the Unibus transfer will be aborted (SSYN Timeout in the Unibus device) and the bit indicating the problem will be set in the UBA status register (IVMR or MRPF). Note that for this implementation, the low-order 16 bits of the map register are accessed only when an SBI transfer is required, and only at that time is parity checked on the low 16 bits of the map register.

The map registers are discussed in detail in Paragraph 2.9.10.

Unibus address bits UA (08:02) determine the longword within a page and are seen by the SBI as address bits SA (06:00). These seven bits are concatenated with the mapped page address to form the 28 bit SBI address.

The two low-order Unibus address bits [UA (01:00)] and the two control bits [C (1:0)] determine the SBI function and byte mask [F (3:0), M (3:0)].

2.6.2 Data Transfer Paths

Data is transferred between the Unibus and the SBI through one of the 16 data paths of the UBA:

1. The DDP translates each Unibus data transaction (DATI, DATIP, DATO, DATOB) directly to an SBI function for each Unibus word (or byte) transfer.

2. Each of the 15 buffered data paths (BDP 1–15) accumulates data and transfers the data as words or bytes to or from the Unibus device. The BDPs perform quadword transfers to SBI memory addresses and longword transfers to SBI I/O addresses. The BDPs will respond to Unibus DATI, DATO, and DATOB functions, but will not respond to the DATIP function.

The data path to be used by a particular device is assigned by the VAX-11 software when setting up the map registers. The data paths are numbered from DP0 through DP15. DP0 is the direct data path (DDP) and DP1 through DP15 are the buffered data paths BDP1 through BDP15, respectively. One or more transferring Unibus devices can be assigned to DP0. No more than one transferring Unibus device, however, can be assigned to any one of the BDPs at any time.

The data paths are transparent to devices on the Unibus. They will perform transfers as if transferring directly to memory.

Microsequencer attention is always required on DMA transfers. Figure 2-39 is a block diagram of the data path logic. Figure 2-40 shows the relation of the data path logic to the UBA as a whole.

2.6.2.1 Direct Data Path – The direct data path (DP0) translates each of the Unibus data transfer functions, DATI, DATO, and DATOB, to a unique SBI function (read masked or write masked) as shown in Table 2-9. The DDP transfers words or bytes directly between the Unibus and SBI address space. In addition, the DDP allows a Unibus device to interlock its operation with the system by translating a DATIP-DATO/B Unibus sequence to an interlock read masked-interlock write masked sequence on the SBI, thereby setting and clearing the memory interlock. Figure 2-41 shows the functions of the direct data path in a simplified block diagram.

Table 2-9 Mapping Transfer Through the DDP, Unibus Control – Address to SBI Function – Mask Encoder

Unibus		SBI	
C(1:0)	UA (1:0)	FUNC (3:0) Mask	3 2 1 0
DATI	0 0 1 0	Read Masked	0 0 1 1 1 1 0 0
DATO	0 0 1 0	Write Masked	0 0 1 1 1 1 0 0
DATOB	0 0 0 1 1 0 1 1	Write Masked	0 0 0 1 0 0 1 0 0 1 0 0 1 0 0 0
DATIP	0 0 1 0	Interlock Read Masked	0 0 1 1 1 1 0 0
Followed by			
DATO	0 0 1 0	Interlock Write Masked	0 0 1 1 1 1 0 0
or DATOB	0 0 0 1 1 0 1 1	Interlock Write Masked	0 0 0 1 0 0 1 0 0 1 0 0 1 0 0 0

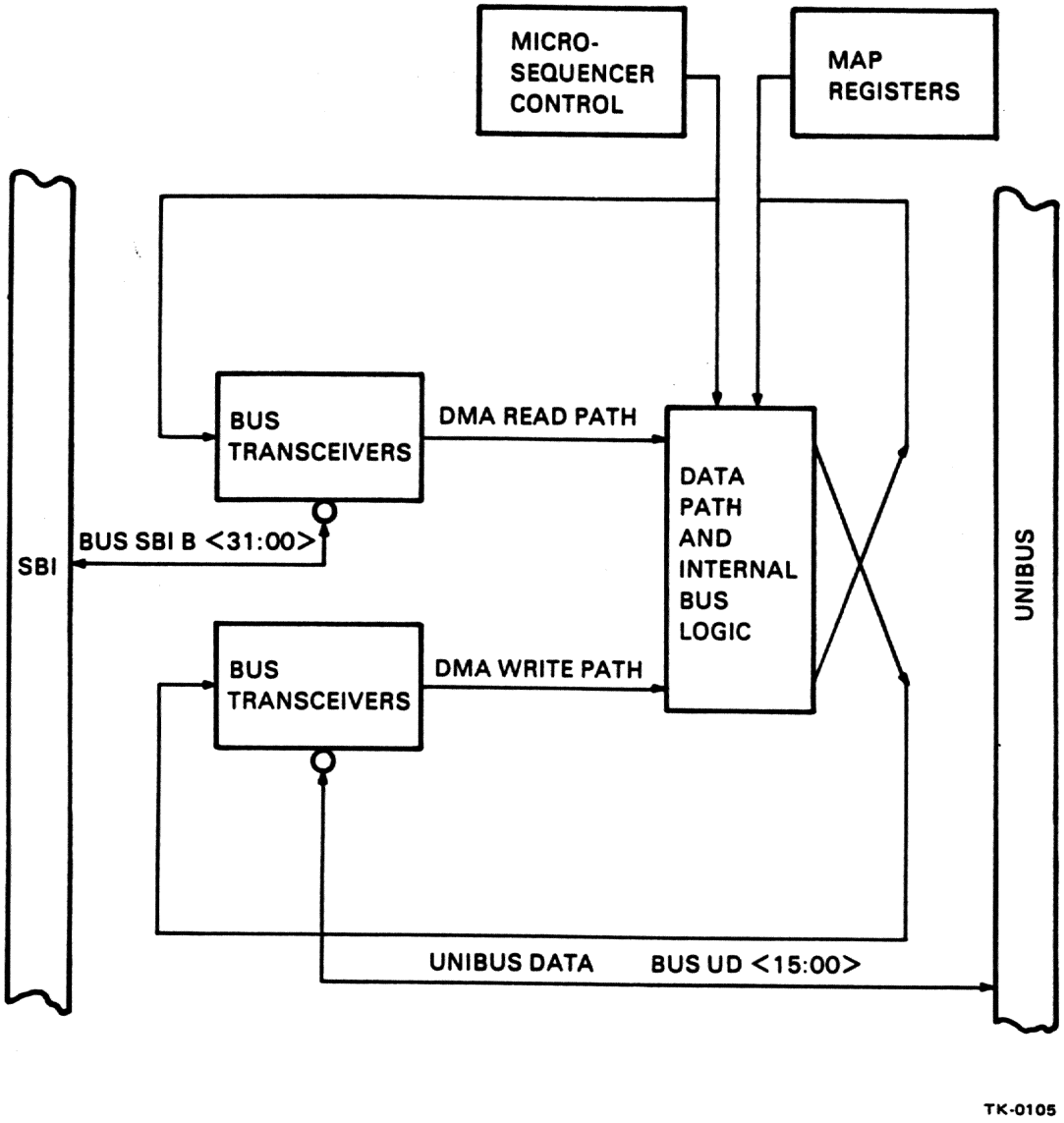


Figure 2-39 Data Path Circuitry, Simplified Block Diagram

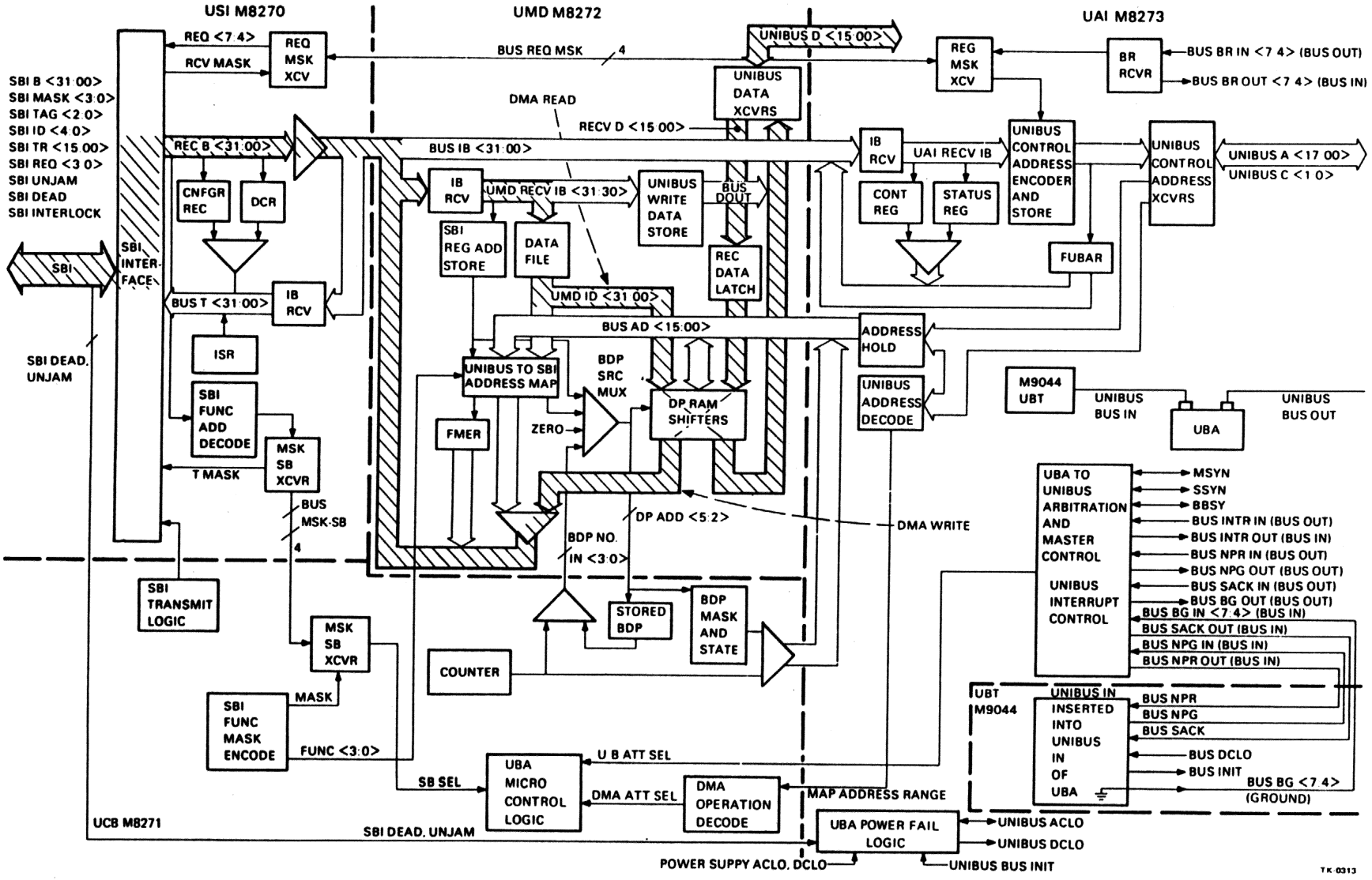


Figure 2-40 Data Paths for Unibus DMA Access to the SBI/UBA Block Diagram

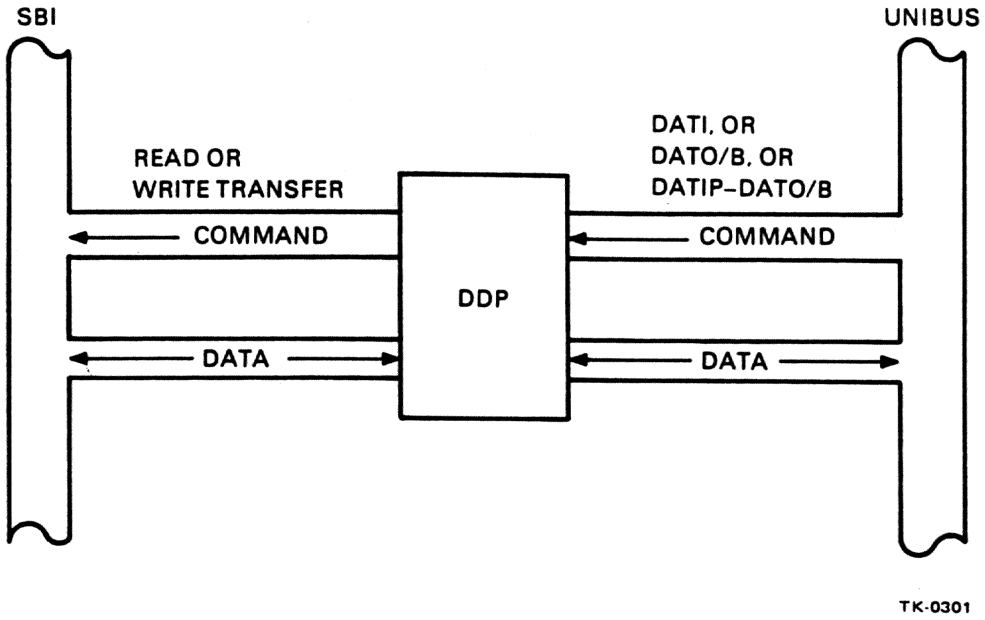


Figure 2-41 Unibus Transfers to the SBI/DDP

Each Unibus word (or byte) transfer is translated by the UBA to an SBI transfer. The Unibus transfer is not completed until the SBI transfer has been completed. The SBI address, function, and byte mask are mapped directly from the Unibus address and control lines, and the state of an internal interlock flip-flop in the case of a DATIP-DATO sequence.

- Usage of the DDP:

  The DDP can be assigned to more than one transferring Unibus device.

  The DDP must be used by any device wanting to execute an interlock sequence (DATIP-DATO/B) to the SBI.

  The DDP must be used by devices not transferring to consecutive increasing addresses or devices that mix read and write functions.

  The maximum throughput via the DDP is approximately 400K words per second.

  The DDP is the simplest data path, for programming, since the map registers are the only UBA registers required to be accessed when initiating a Unibus device transfer.

  Figure 2-42 shows the data path logic in detailed block diagram form.

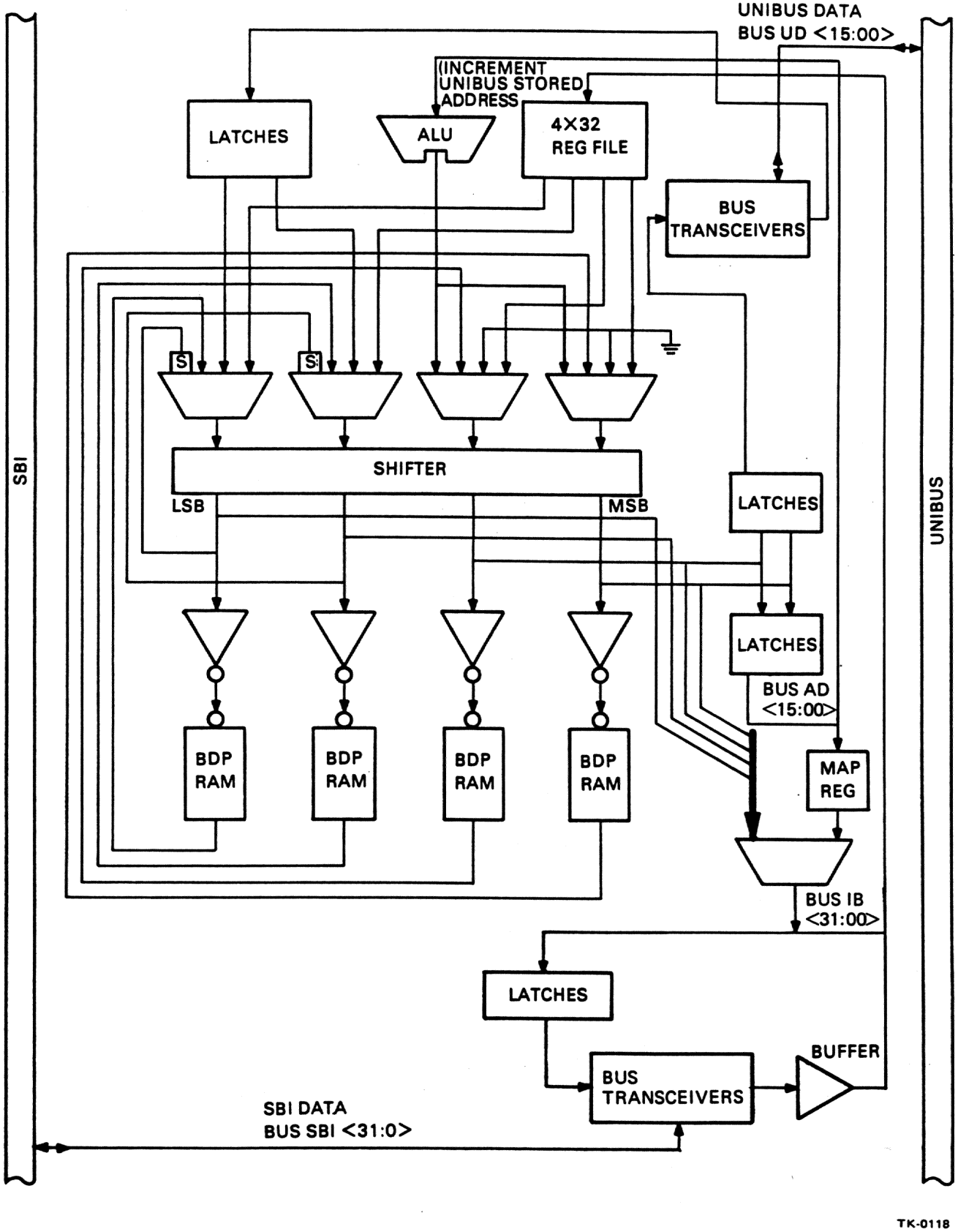


Figure 2-42 Data Path Logic, Block Diagram

- DMA Write through the DDP – When a Unibus DMA device gains control of the Unibus and asserts address, control and data signals, and then Master Sync on the Unibus to initiate a DATO or DATOB transfer, the UBA latches the information and signals the microsequencer that its attention is needed. The upper nine address bits asserted on the Unibus select a map register that the VAX-11/780 software has previously loaded with a data path number (in this case that of the DDP) and an SBI address page number.

  At this point in the transfer, the UBA translates the DATO or DATOB command into the SBI write masked function; arbitrates for control of the SBI; and then after gaining control, asserts command/address on the SBI. Meanwhile, the microsequencer has fed the Unibus data through the 4:1 data path multiplexer shown in Figure 2-42 and into the data path shifter, as one or two bytes in a 32-bit wide data path. The write data is then fed from the shifter to a 2:1 multiplexer, where it is selected by the microsequencer and placed on the internal bus [IB (31:00) H]. The microsequencer then gates the write data to the SBI B field transceivers where it is enabled on the SBI on the next cycle.

- DMA Read through the DDP – When the UBA latches address and control information from the Unibus in response to a DATI initiated by an I/O device, it calls for attention from the microsequencer. The UBA then translates the DATI command into an SBI read masked function. The upper nine Unibus address bits select a map register, which maps the transfer to a specific SBI page address, and which specifies a data path number. Then, after gaining control of the SBI, the UBA asserts a command/address word on the SBI. The microsequencer waits for the returned data. Several cycles later, when the device addressed responds with the requested data (two bytes) on the SBI B lines, the UBA latches the data at time T3 in the SBI cycle. The UBA gates the read data (16 bits in a 32-bit field) from the receive read data logic to one of four locations in the 4 × 32 bit data file. The microsequencer moves the data from the 4 × 32 bit data file through the 4:1 data path multiplexer and to the shifter. The shifter will either gate the 16 data bits straight through or shift them two byte positions, depending on the state of Unibus address bit UA1. The microsequencer then enables the 16-bit word on the Unibus together with Slave Sync, completing the read transfer operation.

2.6.2.2 Buffered Data Paths – The UBA provides 15 buffered data paths (BDPs) for data transfers initiated by fast DMA block transfer devices such as the RK06. The buffered data paths can store up to eight data bytes (one SBI quadword aligned memory image) and thus enable Unibus devices to make more efficient use of the SBI than would be possible with the DDP alone.

Data is transferred between the Unibus and a BDP as words or bytes. It is transferred between the BDP and SBI memory as longwords or quadwords, and between the BDP and SBI I/O registers as longwords only. Several data paths may be in use at the same time, each in connection with a separate Unibus device transfer; and the UBA control circuitry ensures that no conflict as to use of the Unibus and the SBI will arise. However, the VAX-11 software must ensure that no more than one active block transfer is assigned to a specific buffered data path at any time.

Five constraints on use of the BDPs follow.

1. The data transfer must be a block type transfer (a block can be greater than or equal to one byte).

2. The VAX-11/780 software must initiate a BDP maintenance purge following each block transfer. The purge operation is a function unique to the UBA. It clears the BDP of any remaining bytes of data and initializes the state of the BDP. On Unibus to SBI memory write operations the data will be transferred to the appropriate locations in SBI memory. On Unibus to SBI memory read operations, the unneeded data will be discarded.

3. All transfers within a block must be to consecutive, increasing word addresses for DATI or DATO functions or to consecutive increasing byte addresses for the DATOB function.

4. All transfers within a block must be of the same function type – read from memory or write to memory (either DATI or DATO/B).

5. The DATIP Unibus function will not be recognized by the BDPs.

The BDPs perform six types of transfer functions between the Unibus/UBA and the SBI, as follows. Figure 2-43 shows these functions in a simplified block diagram.

1. Unibus to BDP Write – The data from the Unibus is stored in the appropriate byte locations in the BDP. However, unless the transferring Unibus device addresses the last word or byte of an SBI quadword, no transfer on the SBI will occur, and the data will remain stored in the BDP.

2. BDP to SBI Write – When the transferring Unibus device writes data to the last word or byte of an SBI quadword, the UBA arbitrates for control of the SBI and then initiates a write masked or an extended write masked transfer on the SBI, depending on the location of the bytes to be written within the quadword.

3. Purge – Once a BDP has been used for a read or write operation, it cannot be used subsequently for a different block transfer until it has been purged by software. In order to initiate a purge operation, the software writes a one to the BNE bit of the associated Data Path Register (DPR). If the BDP contains write data from the Unibus, the UBA performs the purge by executing a write masked or an extended write masked transfer to the appropriate SBI location. If the BDP contains read data from the SBI, the UBA performs the purge by clearing the data. At the completion of the purge operation, the UBA clears the BNE bit.

4. Unibus to BDP Read – When a Unibus device initiates a DMA read transfer to SBI address space through a buffered data path, the UBA tests the state of the BDP. If the BDP contains data the UBA transfers the data to the Unibus, together with Slave Sync, completing the transfer.

5. BDP to SBI Read – If the BDP accessed on a DMA read transfer from a Unibus device does not contain data (i.e., after it has been purged or initialized), the UBA responds by initiating a read masked or an extended read transfer on the SBI, depending on the location of the word addressed within the quadword. If the word addressed begins on byte 0, 1, 2, or 3, the UBA performs an extended read transfer on the SBI. When the responder returns the quadword addressed, the UBA first stores the data in two locations in the data file, then transfers the word addressed to the Unibus, and then stores the quadword data in the BDP. If the word addressed begins on byte 4, 5, 6, or 7, the UBA performs a read masked transfer on the SBI. When the responder returns the longword data, the UBA stores the longword in the data file, then transfers the word addressed to the Unibus, and then stores the longword data in the BDP.

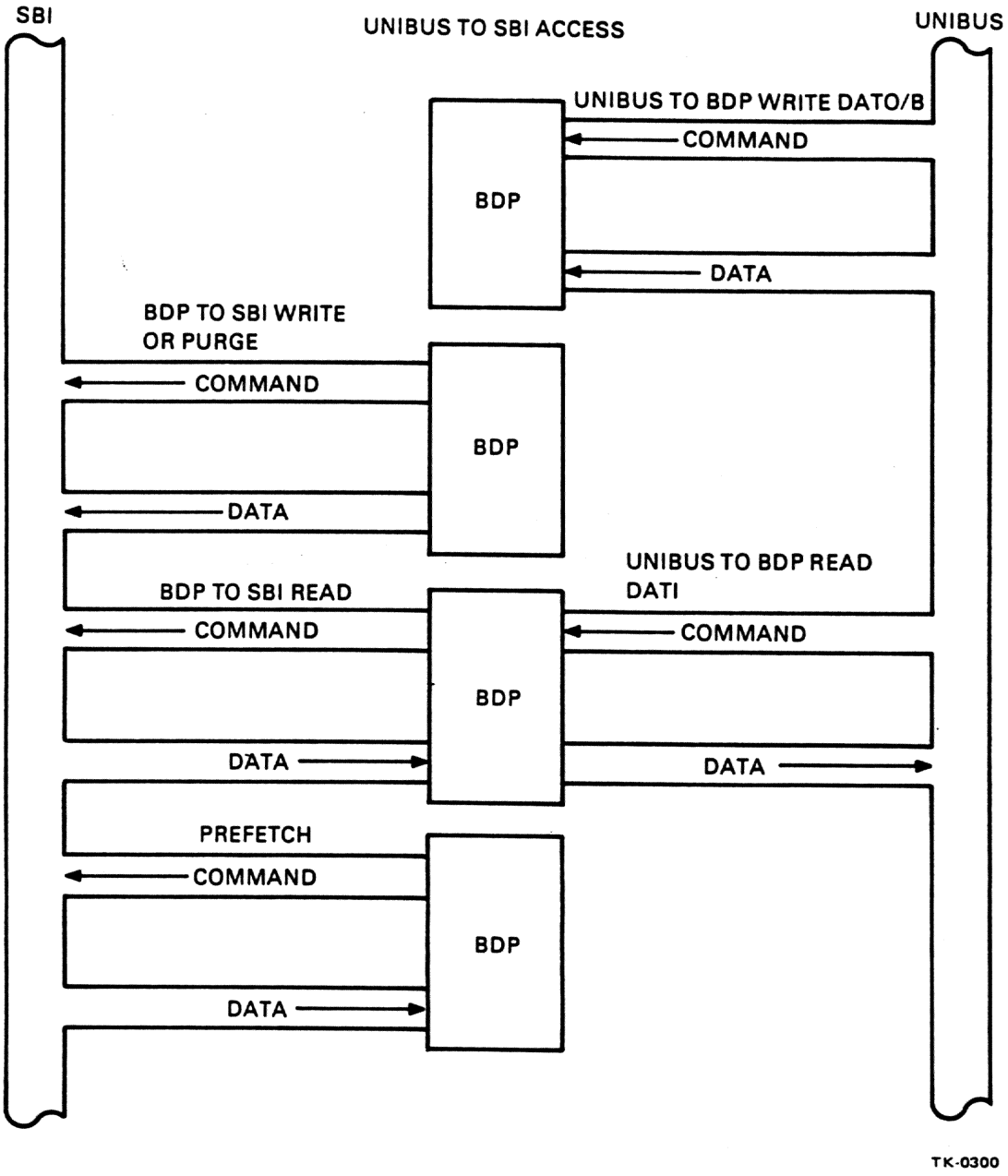


Figure 2-43 Unibus and UBA Transfers to the SBI via the BDPs

6. BDP to SBI Prefetch - When the Unibus device reads the last word in a quadword group, the UBA tests the state of the BDP and then either gates the stored data to the Unibus or performs a read masked transfer on the SBI to retrieve the data before gating it to the Unibus. The UBA will then prefetch the next quadword from memory through an extended read transfer on the SBI. When the UBA receives the data, it holds it in the data file and then moves it to the buffered data path, for use on the next DATI within the same block transfer.

   Since the prefetch allows the possibility that the UBA may cross a page boundary into non-existent memory, resulting in a 100 μs timeout, it is recommended that the software allocate an additional map register following a block. This map register must be invalidated. When the prefetch crosses this page boundary to the invalid map register, the prefetch will be aborted immediately, thereby eliminating the 100 μs timeout. The UBA does not record any UBA or SBI errors that may occur during the prefetch operation since this is an anticipatory function based on the next probable address. If an error does occur, then the prefetch will be aborted and the BDP will not be filled with data. If the Unibus device accesses the same BDP again, the BDP to SBI read will be initiated and any errors that occur will be logged at this time.

The SBI memory allows block transfers to and from memory locations beginning at odd byte locations in the memory address space. However, some Unibus devices are word aligned, and thus unable to assert address bit A0.

The byte offset bit in each of the map registers of the UBA enables word-aligned DMA devices to perform odd byte aligned transfers to or from SBI memory through a buffered data path. The software must set the byte offset bit of the appropriate map register when this feature is to be used.

The byte offset feature can be used on both DMA read transfers and DMA write transfers, and effectively increases the SBI memory address translated from the Unibus address by one byte location. On a DMA write transfer with the byte offset function, when the UBA stores the Unibus data in the assigned buffered data path, it effectively shifts the data to the left by one byte position. When the buffer is full on a block transfer involving seven or more bytes, a maximum of seven data bytes will be enabled on the SBI on the first extended write masked operation within the block transfer. The UBA mask logic will set the appropriate mask bits to indicate that byte 0 (and possibly other bytes) is not to be written. If the transfer requires more than one SBI cycle, the buffered data path stores the last byte transferred from the Unibus in order to send it as the first byte in the next SBI cycle servicing that DMA transfer. Following transfers may invlove all eight bytes. Figure 2-44 shows the relative positions of data in the Unibus and SBI address spaces.

When a Unibus device performs a DMA read transfer through an empty buff ered data path with byte offset to the first location in a memory quadword, the UBA initiates an extended read sequence on the SBI, ignoring the first byte that the memor y puts on the bus. The second and third bytes are gated to the Unibus device as the first word of the transfer. The entire quadword is then stored in the BDP for future access. On subsequent read transfers from the Unibus device to the BDP, the UBA shifts the data from the BDP and passes it to the Unibus.

The BDPs allow DMA Unibus devices to transfer data to and from SBI I/O registers as longwords. In a transfer of this type the software first loads the corresponding map register with the page address of an SBI I/O device. The UBA transfers the data between the Unibus and the BDP as words or bytes and between the SBI and the BDP as longwords. During transfers to SBI I/O address space, the UBA ignores the byte offset bit in the map register.

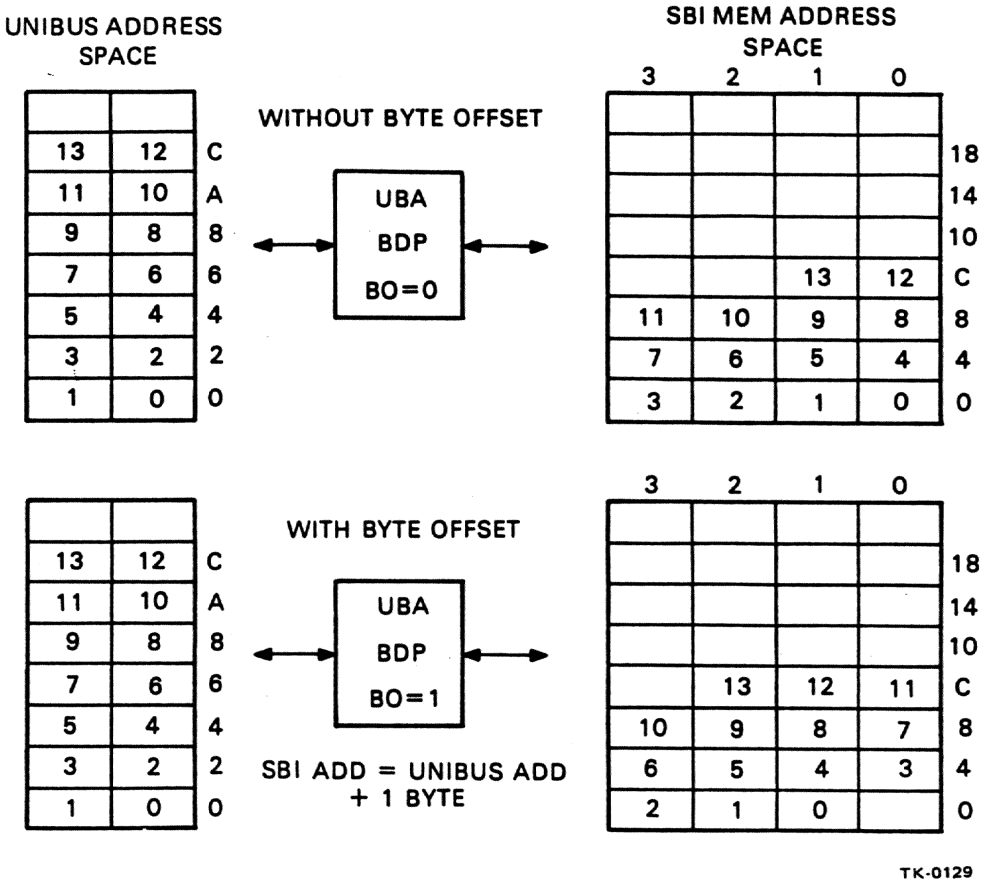


Figure 2-44 Relative Positions of Data in Unibus and SBI Address Spaces

2.7 INTERRUPTS

SBI interrupts can be generated from two sources within the Unibus subsystem: either from the Unibus or from the UBA. The UBA is assigned one interrupt sublevel, which corresponds to its TR number. Interrupts from the UBA itself occur at one assigned request level only. This level is determined by backplane jumpers. Interrupts from the Unibus can occur at any one of the four request levels, as determined by the Unibus BR lines.

When an interrupt request is generated from either the UBA or the Unibus, the VAX-11/780 CPU responds to the request signal with an interrupt summary read command. The UBA decodes the command and replies with an interrupt summary response, sending its request sublevel in a read data format. The combination of request level and UBA request sublevel produces one of the possible 64 SBI interrupt vectors. The four vectors assigned to the UBA point to UBA interrupt service routines corresponding to the four interrupt request levels. The selected UBA service routine must then read and test the BRRVR corresponding to the level of the interrupt:

BRRVR 7 for request Level 7
BRRVR 6 for request Level 6
BRRVR 5 for request Level 5
BRRVR 4 for request Level 4

From the contents of the BRRVRs, the UBA service routine can determine whether the interrupt was generated from within the UBA, from a Unibus device, or from both. The UBA service routine can then service the interrupt, branching on the contents of the BRRVR, as follows.

Bit 31	Bits (15:00)	
0	0	No service required.
0	V	Unibus device service as indicated by vector V.
1	0	UBA service required. Read configuration register and status register to determine the service required.
1	V	Unibus device and UBA service required.

1. Save vector V.
2. Read UBA configuration register and status register.
3. Perform UBA service as indicated by configuration and status register.
4. Index into Unibus device service routine with vector V.

V is the vector field of the BRRVR received from the Unibus device. Zero is the null vector indicating that a vector was not received from the Unibus device.

2.7.1 Interrupts from the Unibus

The UBA will translate the Unibus BR interrupts to SBI request interrupts providing that the interrupt fielder switch (IFS) bit and the BR interrupt enable (BRIE) bit of the UBA control register are set. The assertion of the SBI request lines will initiate an interrupt summary read/interrupt summary response exchange. The UBA service routine then performs a read transfer to the BRRVR corresponding to the level of the interrupt. On receiving the read BRRVR command, the UBA will test for the following conditions.

1. The Unibus BR line corresponding to the BRRVR number is asserted.
2. The BRRVR does not contain an already valid vector.
3. Unibus AC LO is not asserted.

If all of the preceding conditions are met, then the UBA will issue the Unibus bus grant and complete the Unibus interrupt transaction. The BRRVR is loaded with the interrupt vector by the successful completion of the interrupt transaction. The device vector received during the transaction will be sent as read data. If a UBA interrupt is active, then the vector will be sent as a negative quantity (bit 31 sent as a one).

The BRRVR is cleared by the successful completion of the SBI read data cycle. Otherwise, the vector is saved and the BRRVR remains full.

If conditions 1, 2, 3 are not met then the contents of the BRRVR (either the stored vector, from a previously failing SBI read data cycle, or zero) will be sent as read data. If a UBA interrupt is active, then bit 31 will be sent as a one.

If for some reason, the Unibus has initiated an interrupt transaction and then fails to complete it (i.e., passive release), the interrupt vector will not be loaded into the BRRVR. The following mechanism will allow the Unibus interrupt service routine to terminate gracefully.

The idle state of the BRRVR is zero. If, when reading the BRRVR, the UBA interrupt service routine receives the zero vector, it will log an error (if desired) and return from the service routine.

Once successfully loaded, the BRRVR will maintain the interrupt vector until the ACK confirmation to the BRRVR read data has been received, or an adaptor Init sequence is initiated. If the ACK confirmation is not received for the read data, then the BRRVR full bit will not be cleared.

When the software initiates subsequent read transfers to that BRRVR, the UBA will return the stored vector as read data. Receipt of an ACK confirmation for the read data will enable the UBA to clear the BRRVR.

2.7.2 Interrupts from the UBA

If a condition on the UBA warranting an interrupt occurs setting a bit in the configuration register on the status register, and the corresponding interrupt enable bit in the control register is set, the following sequence takes place.

1. The UBA asserts its assigned request line.

2. When the interrupt summary read command, corresponding to the above request level, is seen by the UBA, the request sublevel assigned to the UBA is sent to the CPU as an interrupt summary response.

3. With this information, request level and request sublevel, the CPU can dispatch to the UBA service routine, which will then read the BRRVR corresponding to the level of interrupt. The BRRVR will contain a negative value (bit 31 set).

4. The UBA service routine will detect the negative value and branch to a routine that will read the configuration and status registers to determine the service required.

The request line will remain asserted until all conditions (bits of the UBA configuration and status registers) have been cleared by the software.

2.8 UBA POWER FAIL AND INITIALIZATION

The UBA power fail and initialization logic asserts and responds to signals on both the SBI and the Unibus. It can deal with power fail conditions on the SBI, system power down, system power up, Unibus power down, Unibus power up, and Unibus and UBA initialization, as shown in Figure 2-45.

2.8.1 System Power Up

The Unibus remains in a powered down state as long as the UBA is in a powered down state (power supply DC LO is asserted). During system power up, the UBA will initiate the Unibus power up sequence, providing that the Unibus has power. Once the Unibus power up sequence has been completed, the Unibus initialization complete bit of the UBA status register is set and an interrupt request is initiated to the CPU. If the Unibus power is not on at the time that the system is powered up, the power up sequence will not continue on the Unibus until the Unibus power has been turned on. The UBA power up sequence will completely initialize all registers and functions of the UBA. The negation of power supply AC LO will set the Adaptor Power Up bit in the configuration register and initiate an interrupt request.

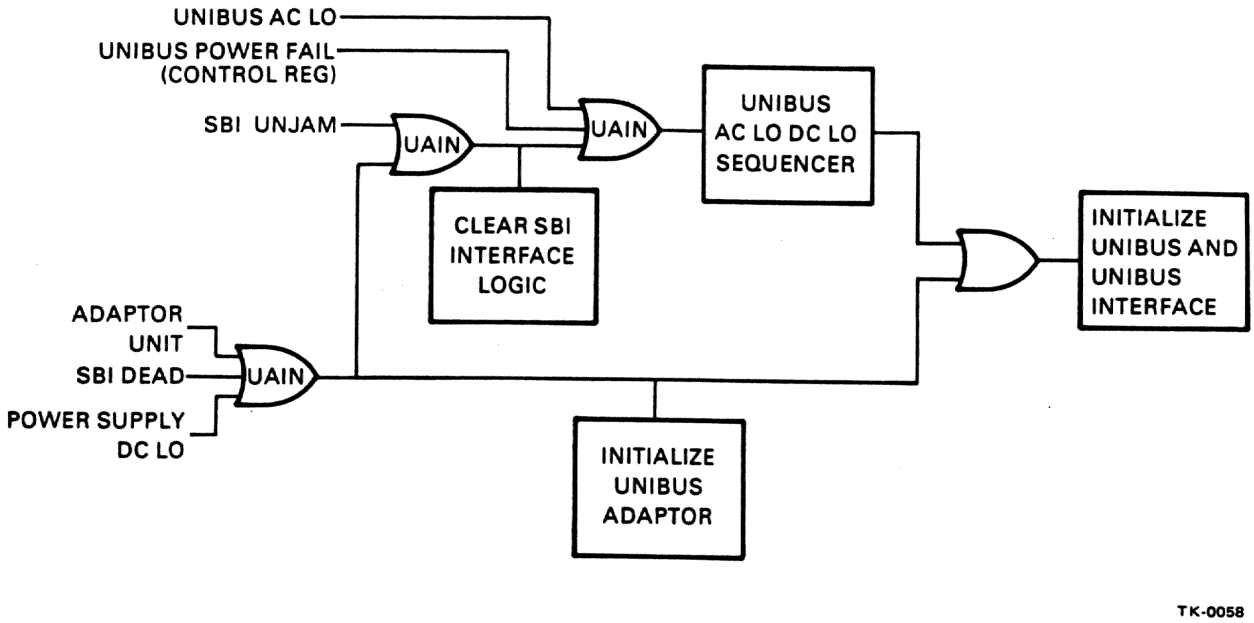


Figure 2-45 Power Fail and Initialization Logic Block Diagram

2.8.2 SBI Power Fail

The UBA will initiate a Unibus power fail sequence whenever an SBI power failure is detected (SBI DEAD asserted). SBI DEAD acts as power supply DC LO within the UBA. The Unibus will remain powered down as long as SBI DEAD is asserted. The UBA will initiate the UBA and Unibus power up sequence when SBI DEAD is released.

2.8.3 UBA Power Fail

The UBA will initiate a UBA and Unibus power fail sequence when the UBA power supply signal SUPPLY DC LO is asserted. The UBA will initiate a UBA and Unibus power up sequence subsequently, when SUPPLY AC LO and SUPPLY DC LO are negated.

2.8.4 Unibus Power Fail

An impending power loss (assertion of Unibus AC LO) on the Unibus will initiate a Unibus power fail sequence. The Unibus power down (UBPON) bit of the status register will be set and the UBA will initiate an interrupt request (providing that the CNFIE of the control register is set). The Unibus will remain in a powered down state until Unibus power has been restored, at which time a Unibus power up sequence is initiated. The Unibus Initialization Complete bit of the status register will be set on a successful power up sequence, and the UBPDN bit will be cleared. The Unibus power fail lines will not affect the state of the SBI power fail lines.

2.8.5 Programmed Unibus Power Fail

The software can induce a power fail sequence on the Unibus by first setting then clearing the Unibus power fail (UPF) bit of the control register. The UBA will initiate a power fail sequence when the UPF bit is set. Once it has been initiated, the power fail sequence will continue to completion independent of the state of the UPF bit. On completion of the power down sequence, the UBA will initiate a power up sequence if, or when, the UPF bit is cleared, providing that power is normal for both the Unibus and UBA.

Setting the AD INIT bit will also initiate a power fail and initialization sequence on the Unibus as well as completely initialize all registers and functions of the UBA.

2.8.6 SBI UNJAM

The assertion of SBI UNJAM will initiate the Unibus power fail and initailization sequence. It will also clear all interrupt enable bits of the UBA control register, and it will initalize the UBA SBI logic so that the UBA is available for an SBI Command.

2.9 SBI ADDRESSABLE UBA REGISTERS

The UBA registers occupy eight pages of the SBI I/O address space. These registers fall into four categories: map registers, (buffered) data path registers, interrupt vector registers, and control and status registers. The UBA registers are all 32-bit registers and can only be written as longwords. These registers, however, will respond to byte or word read commands. These registers will also respond to the interlock-read-interlock-write sequence but will not affect the interlock on the SBI.

The address of the UBA configuration register is located at the base address of the UBA as determined by the backplane TR jumpers (Table 2-6). The addresses of all other UBA registers are relative to the UBA configuration register by an offset. The offset shown is a physical address (PA) (byte) offset in hex (16). The following sections discuss the function and contents of each of the UBA registers.

2.9.1 Configuration Register [CNFGR, PA Offset 000(16)]

The configuration register contains the SBI fault bits, the UBA and Unibus environmental status bits, and the UBA code. This register is required by the SBI specification. Figure 2-46 shows the bit configuration.

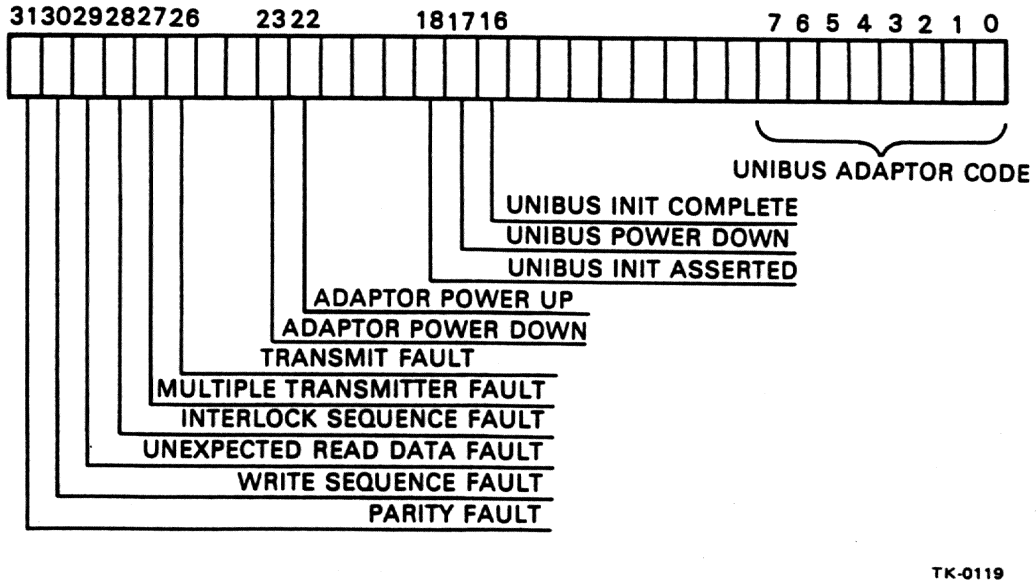


Figure 2-46 Configuration Register, Bit Configuration

The contents of the configuration register are as follows.

Bits (31:27), SBI Faults
These bits are set when the UBA detects specific fault conditions on the SBI. These bits cannot be set once FAULT has been asserted. The negation of FAULT and the disappearance of the fault conditions clear the bits. The setting of any of the bits (31:27) will cause the UBA to assert the FAULT signal for one cycle on the SBI during the confirmation time for the associated transfer.

Bit 31, Parity Fault (PAR FLT)
PAR FLT is set when the UBA detects an SBI parity error.

Bit 30, Write Sequence Fault (WSQ FLT)
WSQ FLT is set when the UBA receives a write masked or interlock write masked command and does not receive the expected write data in the following cycle.

Bit 29, Unexpected Read Data Fault (URD FLT)
URD FLT is set when the UBA receives data for which a read masked, extended read, or interlock read masked command has not been issued.

Bit 28, Interlock Sequence Fault (ISQ FLT)
ISQ FLT is set when an interlock write masked command to Unibus address space is received by the UBA without a previous interlock read masked command.

Bit 27, Multiple Transmitter Fault (MXT FLT)
MXT FLT is set when the UBA is transmitting on the SBI and the ID bits transmitted by the UBA do not match those latched from the SBI. The lack of correspondence indicates a multiple transmitter condition.

Bit 26, Transmit Fault (XMT FLT)
XMT FLT is set if the UBA was the transmitter during a detected fault condition. When the software subsequently reads the configuration and status registers of each of the nexus on the SBI in order to identify the source of the fault, the UBA will be identified as that source if bit 26 is set.

Bits 25, 24
Reserved and zero

Bits 23, 22, 18, 17, 16, Unibus Subsystem Environmental Status Bits
If any of these bits is set and the Configuration Interrupt Enable bit (CNFIE) of the control register is also set, then the UBA will initiate an SBI interrupt request at the level assigned to the UBA.

Bit 23, Adaptor Power Down (AD PDN)
This bit is set when the UBA power supply asserts AC LO. It is cleared by writing a one to the bit location or when the Adaptor Power Up bit is set.

Bit 22, Adaptor Power Up (AD PUP)
This bit is set by the negation of power supply AC LO. It is cleared by writing a one to the bit location or by the setting of the Adaptor Power Down bit.

Bits (21:19)
Reserved and Zero

Bit 18, Unibus Init Asserted (UB INIT)
The assertion of Unibus INIT will set this bit. It is cleared by the setting of the Unibus Initialization Complete bit (UBIC) or by the writing of a one to this bit location.

Bit 17, Unibus Power Down (UB PDN)
This bit is set when Unibus AC LO is asserted. It indicates that the Unibus has initiated a power down sequence. The setting of the UBIC bit or writing a one to this location will clear UB PDN.

Bit 16, Unibus Initialization Complete (UBIC)
This bit is set by a successful completion of a power up sequence on the Unibus. It is the last of the status bits to be set during a UBA initialization sequence, and it can be interpreted to mean that the UBA and the Unibus are ready. The assertion of Unibus AC LO or Unibus INIT, or the writing of a one to this bit location will clear UBIC.

Bits (7:0), Adaptor Code
These bits define the code assigned to the UBA. Table 2-10 shows the bit assignment.

Table 2-10 Adaptor Code Bit Assignment

Bit Number	7	6	5	4	3	2	1	0
Value UBA Number	0	0	1	0	1	0	Vb Vb	Va Va
0							0	0
1							0	1
2							1	0
3							1	1

Adaptor code bits 1 and 0 are determined by backplane jumpers and indicate the starting address of the Unibus address space associated with each UBA, as shown in Table 2-11.

Table 2-11 Selectable Unibus Starting Addresses

UBA Number (Vb, Va)	Starting Address of the Unibus Address Space, base 16 (physical byte address)
0	20100000 (16)
1	20140000 (16)
2	20180000 (16)
3	201C0000 (16)

2.9.2 Control Register [UACR, PA Offset 004(16)]
The UBA control register enables the software to control operations both on the UBA and on the Unibus. All bits used on the register are read/write bits. The Adaptor INIT bit is set by writing a one to the bit location; it is self-clearing. The other bits can be set by writing a one and cleared by writing a zero to the bit locations. Figure 2-47 shows the control register bit configuration.

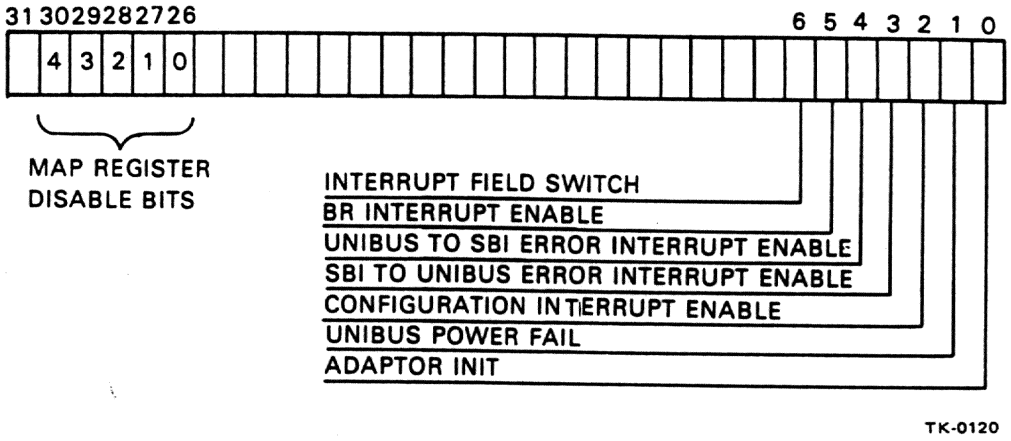


Figure 2-47 Control Register, Bit Configuration

The contents of the control register are as follows.

Bit 31
Reserved and zero.

Bits (30:26) map register disable (4:0) (MRD)
This field of five read/write bits disables map registers in groups of sixteen, according to the binary value contained in the field. The MRD bits prevent the UBA from responding to a Unibus address that points to a disabled map register. The software will load this field with a binary value equal to the number of 4K word units of memory attached to the Unibus, as shown in Table 2-12.

Table 2-12 Map Register Disable Bit Functions

MRD (4:0)	Amount of Unibus Memory (words)	Map Registers Disabled
00000	0K	none
00001	4K	0 to 15 (10)
00010	8K	0 to 31 (10)
00011	12K	0 to 47 (10)
.	.	.
.	.	.
.	.	.
11110	120K	0 to 480 (10)
11111	124K	0 to 495 (10) (all)

DMA transfers to addresses pointing to disabled map registers are not recognized by the UBA. No error bits are set and no SBI transfers are initiated. However, SBI access to disabled map registers is permitted.

The MRD field is initialized as zero, with all map registers enabled. Note, however, that in the initialized state the map registers are all invalid. False Unibus transfers are prevented in this way.

Bits (25:07)
Reserved and zero.

Bit 6, Interrupt Field Switch (IFS)
This bit determines whether interrupts from a Unibus device on the Unibus Out side of the UBA will be fielded by the VAX-11/780 CPU or passed to the Unibus In side of the UBA. If the IFS is set (1), then the interrupt will be passed to the CPU, if the BRIE bit of the control register is set. If the IFS is cleared (0), then the interrupt will be passed to the Unibus In side of the UBA, and it is not seen by the SBI.

The power up state of the IFS bit is 0. The bit is also cleared by the Adaptor INIT and SBI DEAD signals.

Bit 5, Bus Request Interrupt Enable (BRIE)
When this bit is set it allows the UBA to pass interrupts from the Unibus to the CPU, providing that the IFS is set. The power up state of the BRIE bit is 0. The bit is also cleared by the Adaptor INIT, SBI UNJAM, and SBI DEAD signals.

Bit 4, Unibus to SBI Error Field Interrupt Enable (USEFIE)
The USEFIE bit enables an interrupt request to the CPU whenever any of the following status register bits are set on a DMA transfer.

RDTO (Read Data Time Out)
RDS (Read Data Substitute)
CXTER (Command Transmit Error)
CXTMO (Command Transmit Time Out)
DPPE (Data Path Parity Error)
IVMR (Invalid Map Register)
MRPF (Map Register Parity Failure)

The power up state of the USEFIE bit is 0. SBI UNJAM and Adaptor INIT will clear USEFIE.

Bit 3, SBI to Unibus Error Field Interrupt Enable (SUEFIE)
If this bit is set, the UBA will generate interrupt requests to the CPU when one of the two bits in the SBI to Unibus data transfer error field of the status register is set.

UBSTO (Unibus Select Time Out)
UBSSYNTO (Unibus Slave Sync Time Out)

The power up state of the SUEFIE bit is 0. SBI UNJAM, SBI DEAD, and Adaptor INIT will clear the SUEFIE bit.

Bit 2, Configuration Interrupt Enable (CNFIE)
If this bit is set, the UBA will initiate an interrupt request to the CPU whenever any of the environmental status bits of the configuration register is set.

AD PDN (Adaptor Power Down)
AD PUP (Adaptor Power Up)
UB INIT (Unibus Init Asserted)
UB PDN (Unibus Power Down)
UBIC (Unibus Initialization Complete)

The power up state of CNFIE is set (1). CNFIE is cleared by Adaptor INIT, SBI UNJAM, and SBI DEAD.

Bit 1, Unibus Power Fail (UPF)
When the UPF is set, it initiates a power fail sequence on the Unibus, asserting AC LO, DC LO, and INIT in their correct sequence. The software uses this bit to initialize the Unibus. The Unibus will remain powered down as long as UPF is set. The clearing of the UPF bit will initiate a Unibus power up sequence if or when the Unibus power down sequence has finished and Unibus power is ok. Thus, the software can initialize the Unibus by setting and then clearing the UPF bit.

Bit 0, Adaptor INIT (ADINIT)
When this bit is set it will completely initialize the UBA and the Unibus. The map registers, the data path registers, the status register, and the control register will be cleared. The UBA will start the initialization routine in the microsequencer, and it will generate a power fail sequence on the Unibus. The UBA initialization sequence takes only 500 μs to complete, while the Unibus power fail sequence requires approximately 25 ms.

Only the configuration register and the diagnostic control register can be read during the adaptor initialization sequence. Only the configuration register, the diagnostic control register, the control register, and the status register can be written during the adapter initialization sequence.

Once the sequence has been completed, all UBA registers can be accessed. However, the Unibus cannot be accessed until the Unibus initialization sequence has been completed as well. The software can test for this condition by reading the UBIC bit of the configuration register, or by setting the CNFIE bit of the control register and looking for the interrupt generated by the setting of the UBIC bit. Note, however, that the assertion of either Unibus INIT or Unibus power down will also initiate an interrupt (UBINIT). The Adaptor INIT bit can be set by writing a one to the bit location; it is self-clearing.

2.9.3 Status Register [USAR, PA Offset 008(16)]
The UBA status register contains program status and error information. Bits (27:24) are read only bits that are set and cleared by the operations within the UBA. Bits (10:00) can be read and cleared by the software. Writing ones to the appropriate bit locations will clear the bits. Specific conditions which occur on the UBA will set these bits. Writing a zero has no effect on any of the bits. Figure 2-48 shows the status register bit configuration.

The contents of the status register are listed below.

Bits (31:28)
Reserved and zero.

Bits (24:24), BR Receive Vector Register Full
These bits indicate the state of the SBI addressable BRRVRs. Each bit is set when the interrupt vector is loaded into the corresponding BRRVR during a Unibus interrupt transaction, providing that the SBI processor is fielding Unibus device interrupts.

Each bit is cleared by the successful completion of a read data transmission following a read BRRVR command. The software will see these bits set only after a read data failure has occurred during the execution of a read BRRVR command and the Unibus interrupt vector has been saved by the UBA. These bits are cleared only by a subsequent read to the corresponding BRRVR or by an adaptor initialization sequence.

Bit 27 = BRRVR 7 Full
Bit 26 = BRRVR 6 Full
Bit 25 = BRRVR 5 Full
Bit 24 = BRRVR 4 Full

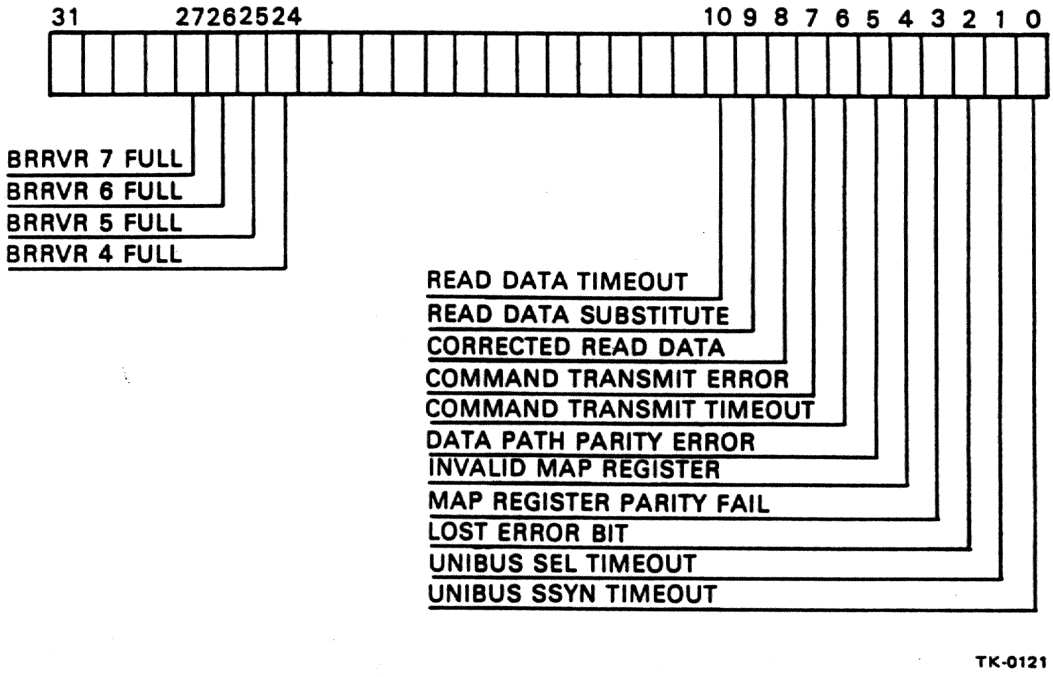


Figure 2-48 Status Register, Bit Configuration

Bits (23:11)
Reserved and zero.

The remaining eleven bits identify specific data transfer errors. They are read and write – one – to clear bits.

Bit 10, Read Data Time Out (RDTO)
The UBA sets the RDTO bit when the following conditions are all true. A Unibus device has initiated a DMA read transfer. The UBA has successfully transmitted a read command on the SBI. The SBI memory has not returned the requested data within 100 μs, and the Unibus device has not timed out. Note that the normal Unibus time out is 10 μs, and that after 10 μs, the Unibus device will set its non-existent memory bit. Thus, the RDTO bit will be set on the UBA status register only if the Unibus device timeout function is inoperative, or takes more than 100 μs. This bit is not set for a BDP to SBI prefetch.

Bit 9, Read Data Substitute (RDS)
This bit is set if a read data substitute is received in response to a Unibus to SBI read command (DMA read transfer). No data will be sent to the Unibus device, and when the device timeout occurs it will set the non-existent memory bit on the device.

Bit 8, Corrected Read Data (CRD)
The UBA sets this bit when it receives corrected read data in response to an SBI read command during a DMA read transfer.

Bit 7, Command Transmit Error (CXTER)
The UBA sets this bit when it receives an error confirmation in response to an SBI command transmission during a Unibus to SBI access, a BDP to SBI read, a BDP to SBI write, or a purge operation. This bit is not set for a BDP to SBI prefetch.

Bit 6, Command Transmit Timeout (CXTMO)
The UBA sets this bit when it fails to complete an SBI command transfer within 100 μs for any of the following operations.

- A BDP to SBI Write
- A BDP purge operation
- A BDP to SBI read operation for which the Unibus device has not timed out

This bit is not set for a timeout for a BDP to SBI prefetch.

Bit 5, Data Path Parity Error (DPPE)
This bit is set when a parity error in a buffered data path occurs during either a Unibus to BDP read, BDP to SBI write, or a BDP purge operation.

Bit 4, Invalid Map Register (IVMR)
The UBA sets this bit during a Unibus DMA transfer or purge operation when the Unibus address points to a map register that has not been validated by the software and has not been disabled by the MRD bits.

Bit 3, Map Register Parity Failure (MRPF)
This bit is set with the occurrence of a map register parity error during one of the following operations.

- A Unibus access in which the Unibus address points to a map register that has a parity error in the upper 16 bits, providing that the map register has not been disabled by the MRD bits.
- Mapping a Unibus address to an SBI address during a direct data path to SBI operation or a BDP to SBI read operation (but not during a prefetch).
- Mapping an address from a buffered data path to an SBI address during a purge operation or a BDP to SBI write.

Seven of the above bits (RDTO, RDS, CXTER, CXTMO, DPPE, IVMR, and MRPF) form an error locking field. If any of these bits is set, the field is locked, thereby preventing the setting of other bits within this field, until the bit indicating the error is cleared. The failed map entry register (FMER) is also locked and unlocked with this field. The setting of any of these bits will cause the UBA to initiate an interrupt request if the interrupt enable bit for the Unibus to SBI data transfer error field (USEFIE) in the control register is set.

Bit 2, Lost Error Bit (LEB)
The UBA sets this bit if the locking error field is locked and another error within this field occurs. The lost error bit does not initiate an interrupt request.

Bit 1, Unibus Select Time Out (UBSTO)
The UBA sets this bit if it cannot gain access to the Unibus within 50 μs in the execution of a software initiated transfer (SBI to Unibus transfer). When UBSTO is set it indicates that the UBA has issued NPR on the Unibus but has not become bus master. This condition indicates the presence of a hardware problem on the Unibus. The Unibus may be inoperative, or one device may be holding it for extended periods. Note that if the Unibus does become inoperative, it may be possible to clear the problem with the assertion of UNJAM on the SBI, the setting and clearing of the Unibus Power Fail bit (control register bit 1) or the setting of Adaptor INIT (control register bit 0).

Bit 0, Unibus Slave Sync Time Out (UBSSYNTO)
This bit is set when an SBI to Unibus transfer (software initiated transfer) times out during the data transfer cycle on the Unibus. The timeout occurs after 12.8 μs. UBSSYNTO indicates a transfer failure resulting when a non-existent memory or device on the Unibus is addressed.

The above two bits, UBSTO and UBSSYNTO, form an SBI to Unibus transfer error locking field. They are set by the occurrence of the conditions mentioned and cleared by writing a one to the bit location. The setting of either bit will cause the UBA to make an interrupt request on the SBI if the SBI to Unibus error interrupt enable bit (SUEFIE) in the control register is set. The setting of either UBSTO or UBSSYNTO will lock the failed Unibus address register (FUBAR), thus storing the high 16 bits of the Unibus address identified with the failure. The FUBAR will remain locked until the UBSTO and UBSSYNTO bits are cleared.

2.9.4 Diagnostic Control Register [DCR, PAA Offset 00C(16)]

The diagnostic control register (DCR) provides control and status bits that aid in the testing and diagnosis of the UBA. The bits of this register, when set, will defeat certain vital functions of the UBA. The DCR, therefore, is not intended for use during normal system operation. Figure 2-49 shows the bit configuration of the DCR.

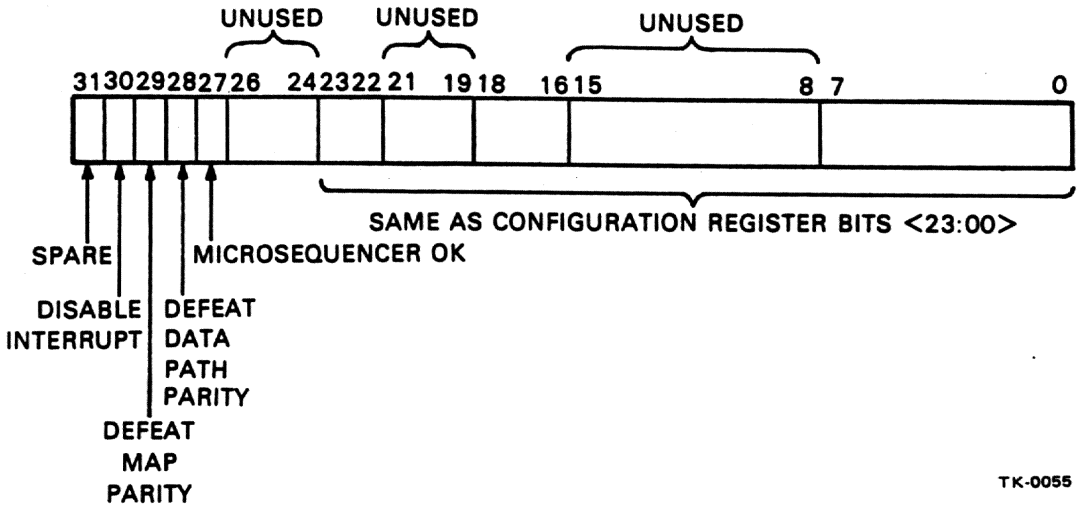


Figure 2-49 Diagnostic Control Register, Bit Configuration

The contents of the diagnostic control register are as follows.

Bit 31, Spare

This read/write bit has no effect on any UBA operation. It can be set by writing a one and cleared by writing a zero to the bit location. SBI DEAD, Adaptor INIT, and a power up sequence on the UBA will clear this bit.

Bit 30, Disable Interrupt (DINTR)

When it is set, this bit will prevent the UBA from recognizing interrupts on the Unibus. It is useful in testing the response of the UBA to the passive release condition during a Unibus interrupt transaction. This bit is set by writing a one and cleared by writing a zero to the bit location. SBI DEAD, Adaptor INIT, and the power up sequence on the UBA will also clear DINTR.

Bit 29, Defeat Map Parity (DMP)
When it is set, this read/write bit will inhibit the parity bits of the map registers from entering the map register parity checkers. The map register parity generator/checkers generate and check parity on eight bit quantities. Each parity field (eight data bits and one parity bit) is implemented so that the total number of ones in the field is odd.

For example, if bits (07:00) of the map register equal zero or contain an even number of ones then the parity bit equals one. However, if the DMP bit is set, then the parity bit is disabled and the parity checkers will see all zeros. This results in a map register parity failure. Then if the DMP bit is cleared, the parity checkers will see correct parity. Note, however, that if bits (07:00) of the map register contain an odd number of ones, the generated parity bit will be zero. The state of the DMP bit, therefore, will have no effect on the parity result in this case.

When the integrity of the parity generator/checkers is to be tested, the map register must contain data such that at least one of the bytes contains an even number of ones. The DMP bit, when set, will disable the parity bit, and the map register parity failure can be detected during a DMA transfer. SBI DEAD, Adaptor INIT, and the power up sequence on the adaptor will clear the DMP bit.

Bit 28, Defeat Data Path Parity (DDPP)
The DDPP bit functions in the same manner as the DMP bit. When it is set, the DDPP bit will inhibit the parity bits of the data path RAM from entering the parity checkers. The data path parity generator/checkers generate and check parity on eight bit data units. Each parity field (eight data bits and one parity bit) is implemented so that the total number of ones in the field is odd. When the integrity of the parity generator/checkers is to be tested through use of the DDPP bit, the total number of ones in at least one of the bytes of data must be even. With the parity bit disabled by the DDPP bit, a data path parity failure will result during a Unibus to BDP read, a BDP to SBI write, or a purge. SBI DEAD, Adaptor INIT, and the power up sequence on the UBA will clear the DDPP bit.

Bit 27, Microsequencer OK (MIC OK)
The MIC OK bit is a read only bit which indicates that the UBA microsequencer is in the idle state. The microsequencer will enter the idle state after it has completed the initialization sequence or once it has completed a UBA function.

The MIC OK bit can be used by the diagnostic to determine whether or not the microsequencer has completed a successful power up sequence and whether or not it is caught up in any loops. Note that SBI DEAD, UBA power supply DC LO, and Adaptor INIT force the microsequencer into the initialization routine. Once the routine has been completed and the microsequencer has entered the idle state, MIC OK will be true (1).

Bits (26:24)
Reserved and zero.

Bits (23:00)
These bits are the same as bits (23:00) of the configuration register.

2.9.5 Failed Map Entry Register [FMER, PA Offset 010, 018(16)]
The FMER contains the map register number used for either a DMA transfer or a purge operation that has resulted in the setting of one of the following error bits of the status register: IVMR, MRPF, DPPE, CXTMO, CXTER, RDS, RDTO. This register is locked and unlocked with the Unibus to SBI data transfer error field of the status register. The FMER is a read only register. Attempts to write to the FMER will result in an SBI error confirmation. When the FMER is not locked, its contents are invalid.

The software can read the FMER to obtain the map register number associated with the failure. It can then read the contents of the failing map register to determine the number of the data path that failed.

Figure 2-50 shows the bit configuration for the FMER.

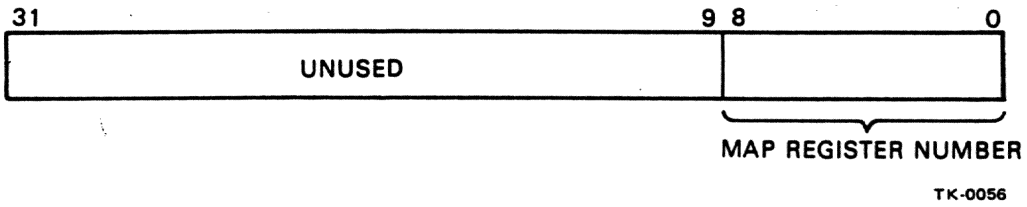


Figure 2-50 Failed Map Register, Bit Configuration

The contents of the FMER are listed below.

Bit (31:09)
Reserved and zero.

Bits (08:00), Map Register Number (MRN)
These bits contain the binary value of the number of the map register that was in use at the time of a failure. Bits (08:00) correspond to bits (17:09) of the Unibus address.

2.9.6 Failed Unibus Address Register [FUBAR, PA Offset 014, 01C(16)]

The FUBAR contains the upper 16 bits of the Unibus address translated from an SBI address during a previous software initiated data transfer. The occurrence of either of two errors indicated in the status register will lock the FUBAR: Unibus Select Time Out (UBSTO) and Unibus Slave Sync Time Out (UBSSYNTO). When the error bit is cleared, the register will be unlocked.

The FUBAR is a read only register. Attempting to write to the register will result in an error confirmation. No signals or conditions will clear the register. Figure 2-51 shows the bit configuration of the FUBAR. The contents of the FUBAR are listed below.

Bits (31:16)
Reserved and zero

Bits (15:00) Failed Unibus Address Bits (17:02)

2.9.7 Buffer Selection Verification Registers 0–3 [BRSVR 0–3, PA Offset 020–02C(16)]

These four read/write do-nothing registers are provided to give the diagnostic software a means of accessing and testing the integrity of the data path RAM. Four locations in the data path RAM have been assigned to these registers. Writing and reading the BRSVRs has no effect on the behavior of the UBA. The BRSVR bit configuration is shown in Figure 2-52.

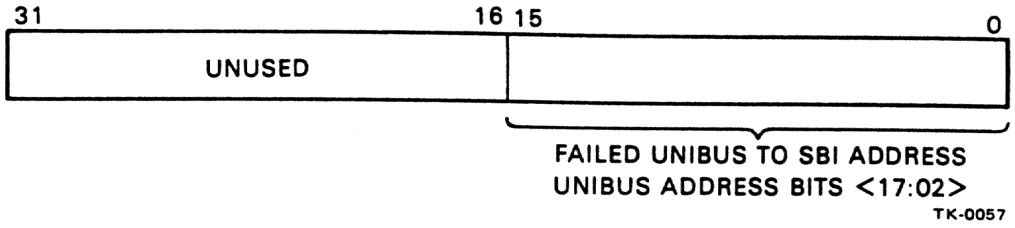


Figure 2-51 Failed Unibus Address Register, Bit Configuration

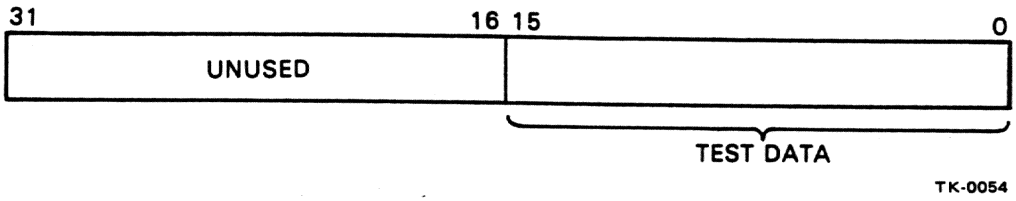


Figure 2-52 Buffer Selection Verification Register
Bit Configuration

2.9.8 BR Receive Vector Registers 4–7 [BRRVR 4–7, PA Offset 030–03C(16)]
The UBA contains four BRRVRs: BRRVR 7, BRRVR 6, BRRVR 5, and BRRVR 4. Each BRRVR corresponds to a Unibus interrupt bus request level: 7, 6, 5, 4. Each BRRVR is a read only register and will contain the interrupt vector of the Unibus device interrupting at the corresponding BR level. Each BRRVR is read by the software as a part of the UBA interrupt service routine. Note that the UBA interrupt service routine is the routine to which the VAX-11/780 CPU will transfer control once it has determined that the UBA or the Unibus has issued an interrupt request to the SBI.

If the IFS and BRIE bits on the control register are set so that Unibus interrupt requests are passed to the SBI, then the CPU responds with an interrupt summary read command. The UBA sends its request sublevel as an interrupt summary response. The software then invokes the UBA interrupt service routine, initiating a read transfer to the appropriate BRRVR. The UBA will assert the contents of the BRRVR on the SBI as read data if the corresponding BRRVR full bit in the status register is set. If the BRRVR full bit is not set, the read BRRVR command causes the UBA to fetch the interrupt vector from the interrupting Unibus device. The interrupt vector is loaded into the BRRVR only at the successful completion of the Unibus interrupt transaction. The UBA will then send the contents of the BRRVR to the SBI as read data. The BRRVR used is cleared only when the UBA receives an ACK confirmation for the read data. Following this exchange, the UBA interrupt service routine will use the contents of the BRRVR to branch to the appropriate Unibus device service routine.

There are five types of abnormal completion conditions that may occur during a Unibus to SBI interrupt sequence.

1. If the software attempts to read a BRRVR for which a BR interrupt line is not asserted, and the BRRVR is not full, the zero vector (all zeros data) will be sent as read data.

2. If the BR line causes an interrupt sequence to begin on the SBI but is released before the interrupt summary read transfer (passive release), then the interrupt summary response from the UBA will be zero.

3. If the BR line asserted by the interrupting Unibus device is released after the interrupt summary read transfer but before the read BRRVR (passive release), then zero will be sent as read data for the read BRRVR command.

4. If the vector has been received from the interrupting device, but an ACK confirmation is not received following the interrupt summary response (read data transmission), then the BRRVR will not be cleared, and the BRRVR full bit will remain set. Subsequent read commands to the full BRRVR will cause the UBA to send the stored vector, but the BRRVR will remain full until the UBA receives an ACK confirmation for the read data. Note that the BRRVR full bits always reflect the state of the BRRVRs.

5. If the IFS bit in the control register is cleared and the software reads a BRRVR, then the zero vector will be sent as read data.

The contents of the BRRVR are also used by the software to determine whether or not the UBA itself has an interrupt pending. Bit 31 of the BRRVR is the adaptor interrupt request indicator. Although the bit is present in all four BRRVRs, it will be active only in the BRRVR corresponding to the interrupt request level that has been assigned to the UBA. If bit 31 is set when the software reads the BRRVR, then an adaptor interrupt request is pending.

Figure 2-53 shows the BRRVR bit configuration.

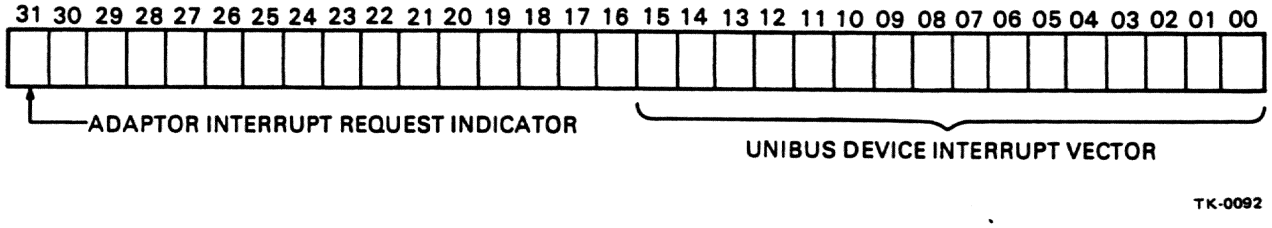


Figure 2-53 BR Receiver Vector Register, Bit Configuration

The contents of the four BRRVRs are listed below.

Bit 31, Adaptor Interrupt Request Indicator

0 = No UBA interrupt pending
1 = UBA interrupt pending

Bits (30:16)
Reserved and zero.

Bits (15:00), Device Interrupt Vector Field
These bits contain the device interrupt vector loaded by the UBA during the Unibus interrupt transaction.

2.9.9 Data Path Registers 0–15 [DPR 0–15, PA Offsets 40–7C(16)]

The UBA contains 16 data path registers (DPR 0 to DPR 15), each of which corresponds to one of the 16 data paths. The DPRs contain status information relative to the buffered data paths and provide the means for purging and initializing the BDPs at the completion of a Unibus block transfer for DP1:DP15. DPR0 corresponds to the DDP and is, therefore, always zero.

Figure 2-54 shows the data path register bit configuration.

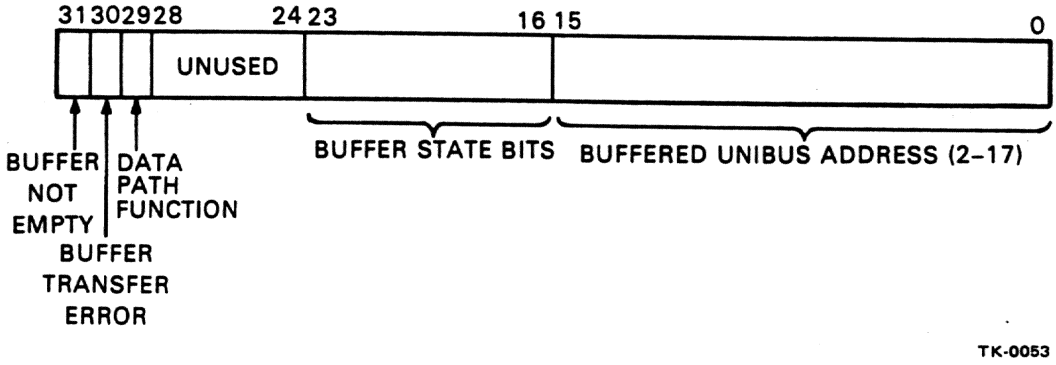


Figure 2-54 Data Path Register, Bit Configuration

The DPR bit functions are as follows:

Bit 31, Buffer Not Empty (BNE)
Each DPR contains a data path status bit called buffer not empty.

1 = Buffer not empty.
0 = Buffer empty.

The BNE bit reflects the state of the associated BDP. If this bit is set (1), the BDP contains valid data. If clear, then the BDP does not contain valid data. The UBA uses the bit to determine the proper action for DMA transfers via the BDP. If bit 31 is set as a DATI transfer begins, the data in the BDP will be asserted on the Unibus. If bit 31 is clear on a DATI, the UBA will initiate a read transfer to SBI memory, gate the addressed data to the Unibus, and then load the read data into the BDP, thereby setting bit 31.

For a DMA write transfer via the associated BDP, the BNE bit is set each time Unibus data is loaded into the BDP. The bit is then cleared when the contents of the BDP are transferred to SBI memory.

The software will write a one to the BNE bit to initiate a purge operation at the completion of a DMA transfer using the corresponding buffered data path. The UBA executes purge operations as follows.

1. Write Transfers to Memory – If any bytes of data remain in the corresponding BDP (BNE is set), the UBA will transfer this data to the SBI location addressed. The UBA will then initialize the BDP and clear the BNE bit. If no data remains to be transferred (BNE is cleared), the purge operation will be treated as a no-op (it is a legal do-nothing function).

2. Read Transfers to Memory – If any bytes of data remain in the BDP, the UBA will initialize the BDP by clearing the BNE bit.

In addition, the following considerations apply to the purge operation.

- For purge operations in which data is transferred to memory, the SBI transfer takes about 2 μs. The UBA will not respond to data path register read transfers during this period (busy confirmation), thereby preventing a race condition when testing for BNE bit.

- A purge operation to data path register 0 (direct data path) is treated by the UBA as a no-op.

Bit 30, Buffer Transfer Error (BTE)
This is a read-write-one-to-clear bit. The UBA sets the BTE bit if a failure occurs during a BDP to SBI write or a purge, or for a buffer parity failure during a Unibus to BDP read access. If bit 30 is set, any additional DMA transfers via the BDP will be aborted until the bit is cleared by the software. Note that if a parity error on the Unibus occurs during a DMA read, the Unibus signal PB will be asserted, giving the Unibus device the opportunity to abort its own DMA transfer. If the device does not abort its own transfer, the UBA will abort the transfer on the next access. The purge operation does not clear the BTE bit. The software clears this bit by writing a one to the bit location.

Bit 29, Data Path Function (DPF)
The DPF is a read only bit. This bit indicates the function of the DMA transfer using this data path.

0 = DMA Read
1 = DMA Write

Bits (23:16), Buffer State (BS)
These eight read only bits indicate the state of each of the eight byte buffers of the associated BDP during a DMA write transfer. They are included in the data path register for diagnostic purposes only. The UBA generates the SBI mask bits from the BS bits during a BDP to SBI write transfer or purge operation. The bits are set as each byte is written from the Unibus. The bits are cleared during the SBI write operation.

0 = Empty
1 = Full

Bits (15:00), Buffered Unibus Address (BUBA)
This portion of each DPR contains the upper 16 bits of the Unibus address, UA (17:02), asserted during a Unibus to BDP write transfer using the associated BDP. If the transfer through the associated BDP is in the byte offset mode, and the last Unibus transfer has spilled over into the next quadword, then these bits contain UA (17:02) + 1. Equation:

BUBA (15:00) = Upper 16 bits of UA (17:00) + Byte Offset

This is the Unibus address from which the SBI address will be mapped during a purge operation.

2.9.10 Map Registers 0–495 [MR 0–495(10), PA Offsets 800–FBC(16)]

The UBA contains 496 map registers, one for each Unibus memory page address (a page = 512 bytes).

Register	Offset
MR0	800
MR1	804
MR2	808
MR3	80C
•	•
•	•
•	•
MR494	FB8
MR495	FBC

When a DMA transfer begins, the upper nine address bits asserted by the Unibus device select a map register. The UBA tests whether the map register has been validated by the software, steers the transfer through one of the 16 data paths, determines whether or not the transfer will take place in the byte offset mode if a BDP has been selected, and maps the Unibus page address to an SBI page address.

The map registers are numbered sequentially from 0 through 495(10). There is a 1-1 correspondence between each map register and Unibus memory page address [i.e., MR0 corresponds to Unibus memory page 0, MR1 to Unibus memory page 1,.... MR495(10) to Unibus memory page 495(10)]. Each map register contains the information required to effect the data transfer of the Unibus device addressing that page:

1. The fact that the software has loaded (or not loaded) the map register. (Map register valid.)
2. The number of the data path to be used by the transfer and, if a BDP is used, whether it is in byte offset mode.
3. The SBI page to which the transfer will be mapped.

Figure 2-38 shows the relation of the map register bits (20:00) to the Unibus and SBI address spaces. Figure 2-55 shows the map register bit configuration.

NOTE

In the interest of brevity, for the map register description, "this Unibus page" refers to "the Unibus memory page corresponding to this map register."

The contents of a map register are as follows:

Bit (31) – Map register valid (MRV).
0 = Not valid – initialized state.
1 = Valid.

The MRV is set by the software to indicate that the contents of this map register are valid. The MRV is tested each time that "this Unibus page" is accessed. If the bit is set (1), the transfer continues. If the bit is not set, the Unibus transfer is aborted (nonexistent memory error in the Unibus device) and the invalid map register bit is set in the UBA status register, providing that the map register has not been disabled by the MRD bits of the control register.

The MRV can be set and cleared by the software.

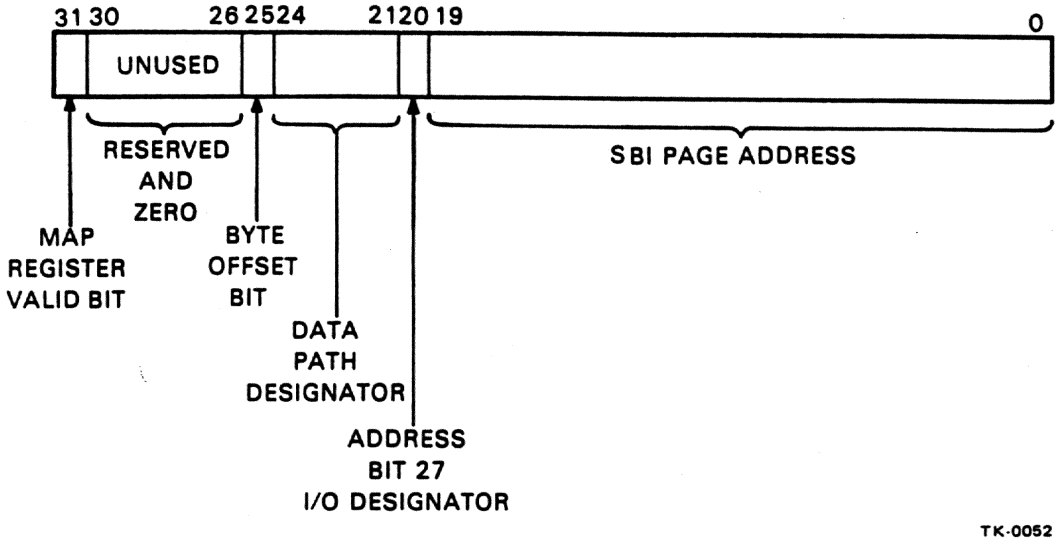


Figure 2-55 Map Register, Bit Configuration

Bits (30:26), Reserved read write bits

Bit 25, Byte Offset Bit (BO)
This is a read/write bit. If set, and "this Unibus page" is using one of the BDPs, and the transfer is to an SBI memory address, then the UBA will perform a byte offset operation on the current Unibus data transfer. The software can interpret this operation as increasing the physical SBI memory address, mapped from the Unibus address, by one byte. This allows word-aligned Unibus devices to transfer to odd byte memory addresses.

Unibus transfers via the DDP or to SBI I/O addresses will ignore the byte offset bit.

This bit is cleared on initialization.

Bits (24:21), Data Path Designator Bits (DPDB)

0000 = Direct Data Path (DDP)
0001 = Buffered data path 1
.
.
.
1111 = Buffered data path 15

The DPDBs are read/write bits that are set and cleared by the software to designate the data path that "this Unibus page" will be using.

The software can assign more than one Unibus transfer to the DDP. The software must ensure that no more than one active Unibus transfer is assigned to any BDP.

The DPDBs are cleared on initialization.

Bits (20:00) SBI Page Address [SPA (27:07), also known as Page Frame Number, PFN].
The SPA bits contain the SBI page address to which "this Unibus page" will be mapped. These bits perform the Unibus to SBI page address translation. When an SBI transfer is initiated the contents of SPA (27:07) are concatenated with Unibus address bits UA (08:02) to form the 28 bit SBI address, as shown in Figure 2-35.

2.10 SBI INTERFACE

The UBA contains the bus transceivers for the SBI lines, decoders for some of the SBI fields, parity check logic, timing logic, SBI arbitration logic, and the diagnostic control and configuration registers. Figure 2-56 shows SBI field configurations corresponding to the various SBI functions that the UBA performs.

2.10.1 UBA Timing

The USI module receives three pairs of SBI clock pulses and from them derives signals that divide the 200 ns SBI cycle into four 50 ns parts separated by T0, T1, T2, and T3. These derived clock signals are distributed throughout the UBA to enable latches and clock registers, thus synchronizing UBA functions with the SBI timing.

2.10.2 Arbitration and ID Functions

At system build time, the UBA is assigned a nexus TR line (from 1 to 15) and a corresponding 5 bit ID code. Note that the fifth ID code bit will be zero in all cases. The arbitration logic, most of which is contained on a single chip, performs all of the functions necessary to enable the UBA to arbitrate for the SBI.

The microsequencer will enable SEND TR H when the UBA must gain control of the SBI to assert command/address or to send read data. The arbitration circuit, in turn, asserts BUS TR L at T0 of an SBI cycle. At T3 of the same SBI cycle, the arbitrator examines the states of all higher priority TR lines, as explained in the SBI functional description, Paragraph 2.2.

If no higher TR line is asserted, the arbitrator asserts ARB OK L, enabling the microroutine to gate the information to be transmitted to the SBI B lines at T0 of the following SBI cycle.

If a higher priority TR line is asserted, the UBA cannot immediately gain control of the information path, but it will keep its TR line asserted until it does gain control. In addition, if the UBA is to perform a data transfer requiring two or three consecutive SBI cycles, the microsequencer will assert SEND HOLD H, enabling the arbitrator to assert BUS SBI TR O L, so that no other nexus can assume control of the information path on the next cycle.

The UBA asserts its own ID code on the SBI with command/address and write data. It also asserts the ID stored from the commanding nexus with read data and the interrupt summary response, as shown in Figure 2-57.

2.10.3 Mask Field, Function Bits and Unibus Signals A1, A0, C1, C0

When the UBA initiates a transfer on the SBI in response to a DMA transfer on the Unibus, it also asserts mask and function bits in accordance with the SBI protocol outlined in Paragraph 2.2.5.4 (Table 2-9). When the UBA becomes the Unibus master in response to a software initiated transfer, it encodes UA1, UA0, C1, and C0 to correspond to the SBI mask and function bit pattern, as shown in Table 2-6. Figure 2-58 shows the mask and function encoder/decoder logic in block diagram form.

	PARITY	TAG	ID	MASK	FUNC	BYTE 3	BYTE 2	BYTE 1	BYTE 0	
READ DATA FORMAT	P1 P0	000	ID	0000						DATA BITS
CORRECTED READ DATA FORMAT	P1 P0	000	ID	0001						CORRECTED DATA BITS
READ DATA SUBSTITUTE FORMAT	P1 P0	000	ID	0010						SUBSTITUTE DATA BITS
INTERRUPT SUMMARY RESPONSE FORMAT	P1 P0	000	ID	0000						BIT PAIR
C/A FORMAT FOR READ MASKED	P1 P0	011	ID	XXXX	0001					ADDRESS BITS
C/A FORMAT FOR WRITE MASKED	P1 P0	011	ID	XXXX	0010					
C/A FORMAT FOR INTERLOCK READ MASKED	P1 P0	011	ID	XXXX	0100					
C/A FORMAT FOR INERLOCK WRITE MASKED	P1 P0	011	ID	XXXX	0111					
C/A FORMAT FOR EXTENDED READ	P1 P0	011	ID	----	1000					
C/A FORMAT FOR EXTENDED WRITE MASKED	P1 P0	011	ID	XXXX	1011					
WRITE DATA FORMAT	P1 P0	101	ID	XXXX		BYTE 3	BYTE 2	BYTE 1	BYTE 0	
INTERRUPT SUMMARY READ FORMAT	P1 P0	110	ID	0000		00000000	00000000	00000000	0000	

REQ <7:4>

TK 0112

Figure 2-56 SBI Field Configuration

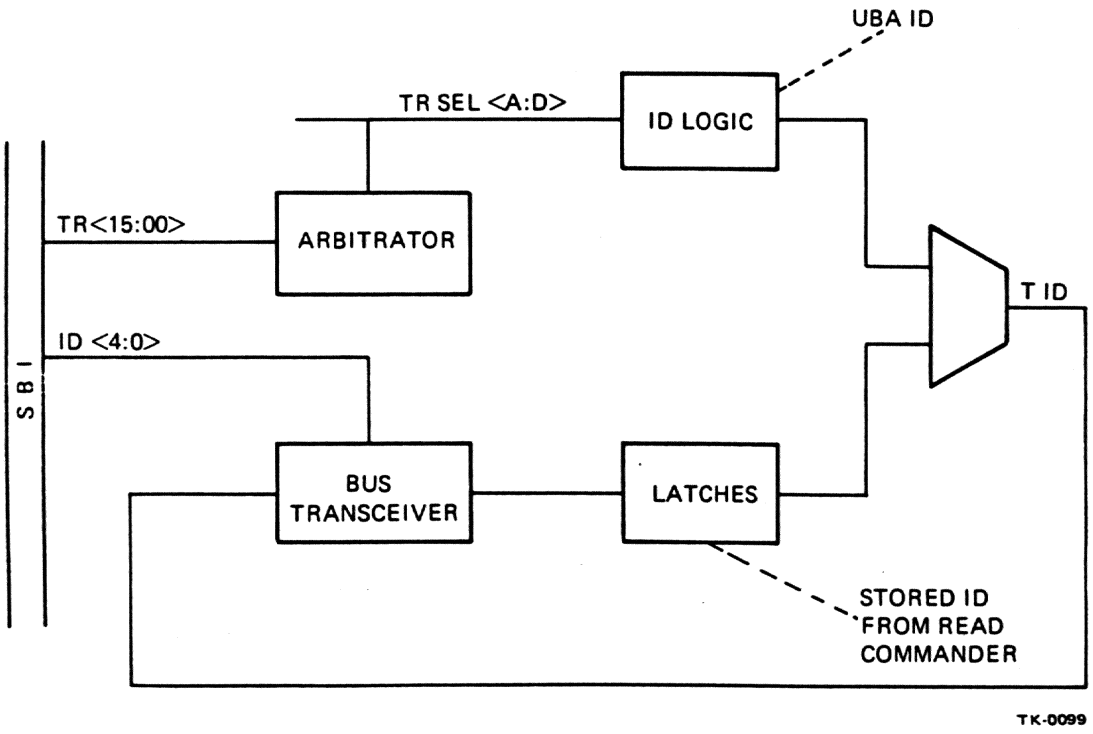


Figure 2-57 Arbitration and ID Logic, Block Diagram

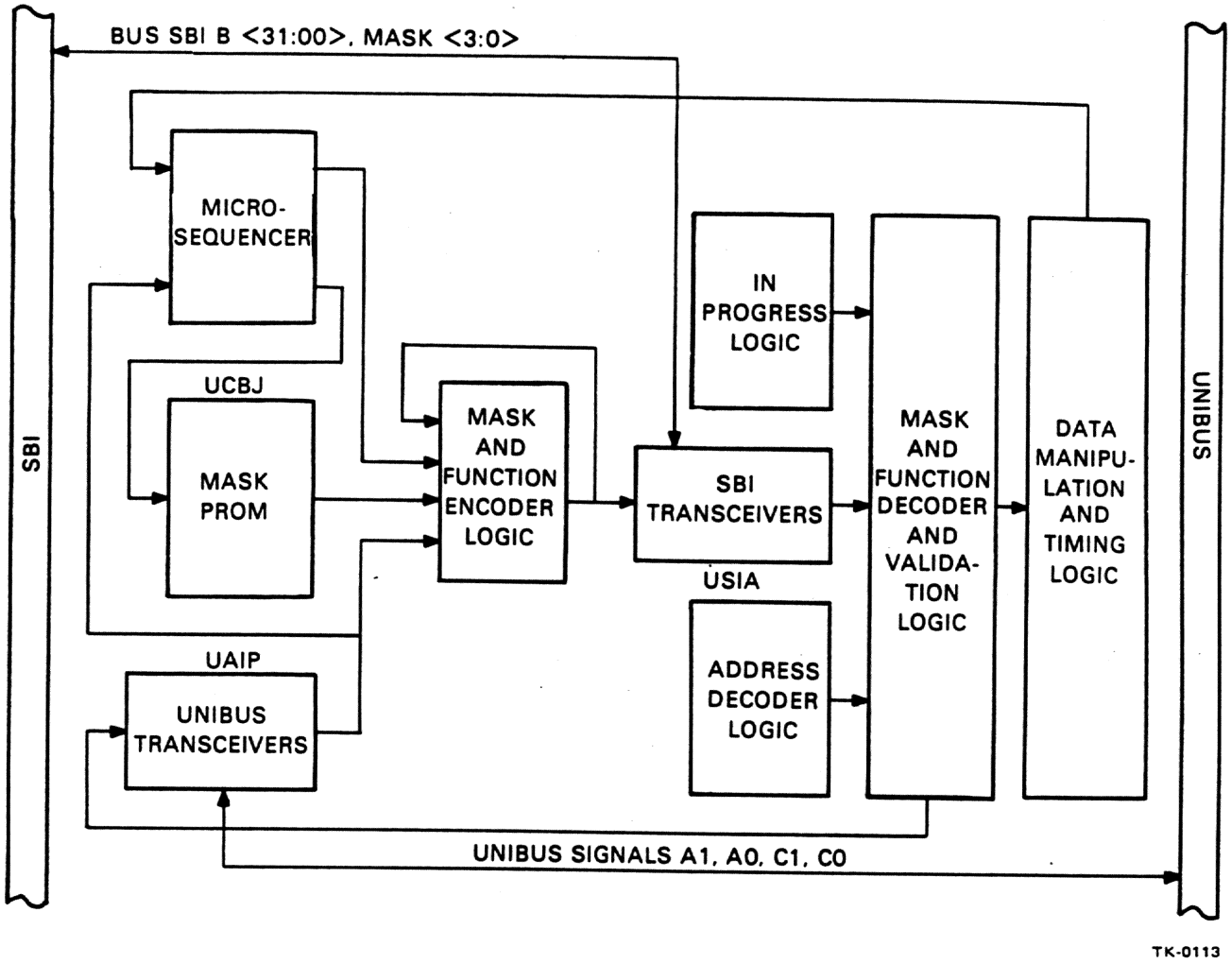


Figure 2-58 Mask and Function Logic, Block Diagram

The function field defines three types of read transfers and three types of write transfers. It is transmitted on the SBI as a part of the command/address format only. When the UBA receives command/address information, the function decoder interprets the function bit pattern. The decoder then enables the appropriate data manipulation logic, and, if necessary, signals the microsequencer that it should begin a microroutine.

When the UBA becomes the SBI commander and asserts the function code on bits (31:28) of the SBI B field as a part of command/address, the microsequencer will in all cases be executing a routine in response to signals on the SBI (purge), the Unibus, or a condition within the UBA needing attention (prefetch). As Figure 2-58 shows, the function encoder logic samples signals from the microsequencer, the mask PROM, and Unibus address and control lines.

2.10.4 Interrupt Request Field

The UBA transmits the four bus request signals in one direction only, from the Unibus or the UBA to the SBI. When the software sets bits 5 and 6 in the control register, enabling Unibus interrupts, the UBA will gate the bus request signals from the Unibus through to the SBI. Likewise, when the software sets bit 2 or bit 3 in the control register, the setting of certain bits in the configuration and status register will cause the UBA to assert its assigned request signal on the SBI. Figure 2-59 shows the request logic in block diagram form.

Table 2-13 lists the UBA signals that will initiate the request.

2.10.5 Confirmation Field

The UBA will assert a 2-bit confirmation code on the SBI two cycles following the reception of any information addressed to it (command/address, read data, or write data) from the SBI. The UBA confirmation logic compares the information received from the SBI to the present state of the SBI receive logic, as determined by the In Progress flip-flops. The UBA then asserts the appropriate confirmation signals. The state of the In Progress flip-flops is changed by the initiation and completion of UBA functions received from the SBI, as shown in Figure 2-60.

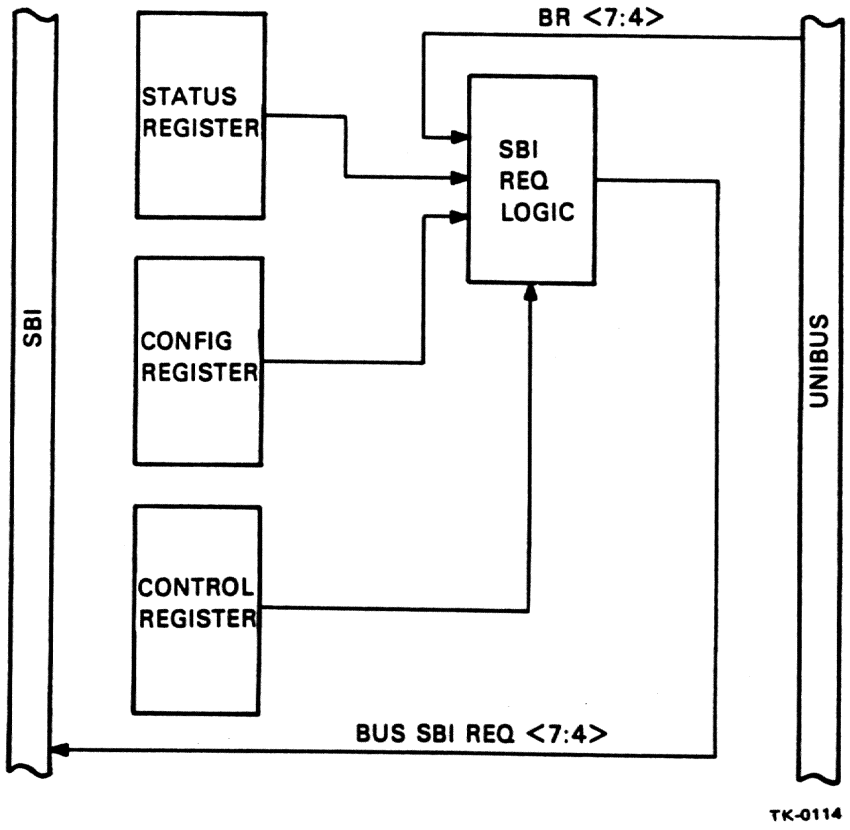


Figure 2-59 Request Logic, Functional Block Diagram

Table 2-13 UBA Signals that Will Initiate SBI Interrupt Requests

Status Register Signal	Bit	Configuration Register Signal	Bit
Read Data Time Out	10	Adaptor Power Down	23
Read Data Substitute	9	Adaptor Power Up	22
Command Transmit Error	7	Unibus Init	18
Command Transmit Time Out	6	Unibus Power Down	17
Data Path Parity Error	5	Unibus Init Done	16
Invalid Map Register	4		
Map Register Parity Failure	3		
Unibus Select Time Out	1		
Slave Sync Time Out	0		

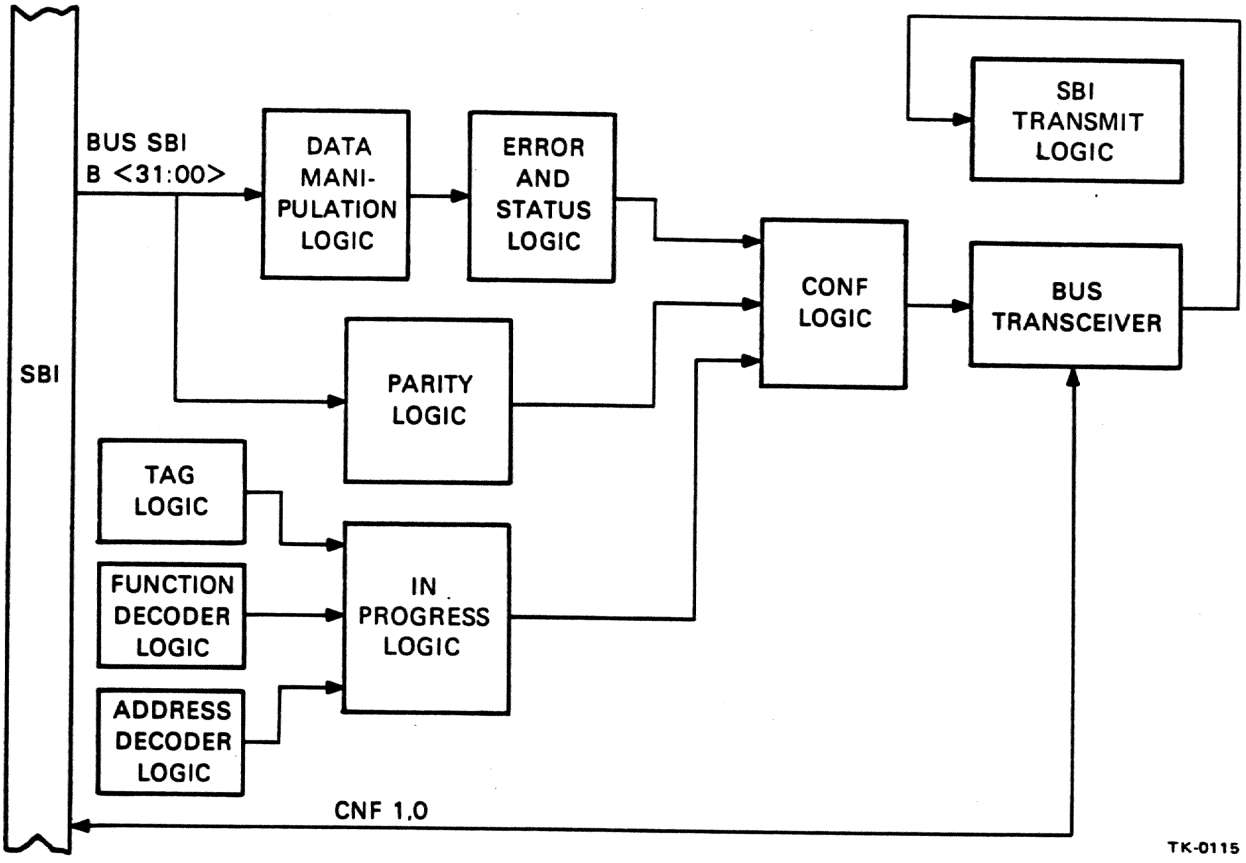


Figure 2-60 Confirmation Logic, Functional Block Diagram

In addition, when the UBA transmits data on the SBI, the microroutine in progress monitors the confirmation lines to ensure that the information transmitted has been received properly. An ACK confirmation will enable the microroutine to proceed as normal, while a BSY, or No Response Confirmation will cause the microsequencer to retransmit the data or take other appropriate action. An error confirmation will abort the SBI transmission and set the appropriate status bits.

2.10.6 Tag Field

The three tag signals are always transmitted when information is transferred on the SBI. The tag specifies whether the information on the B lines of the SBI is command/address, read data (or interrupt summary response), write data, interrupt summary read, or data for diagnostic use. When the UBA latches data from the SBI, it also latches and decodes the tag. The four signals produced by the tag decoder enable the appropriate logic, which in turn handles the information latched from the B field of the SBI.

When the UBA must transmit information and arbitrates successfully for the SBI, the microsequencer supplies the signals necessary to generate the tag code to be asserted with the information, as shown in Figure 2-61.

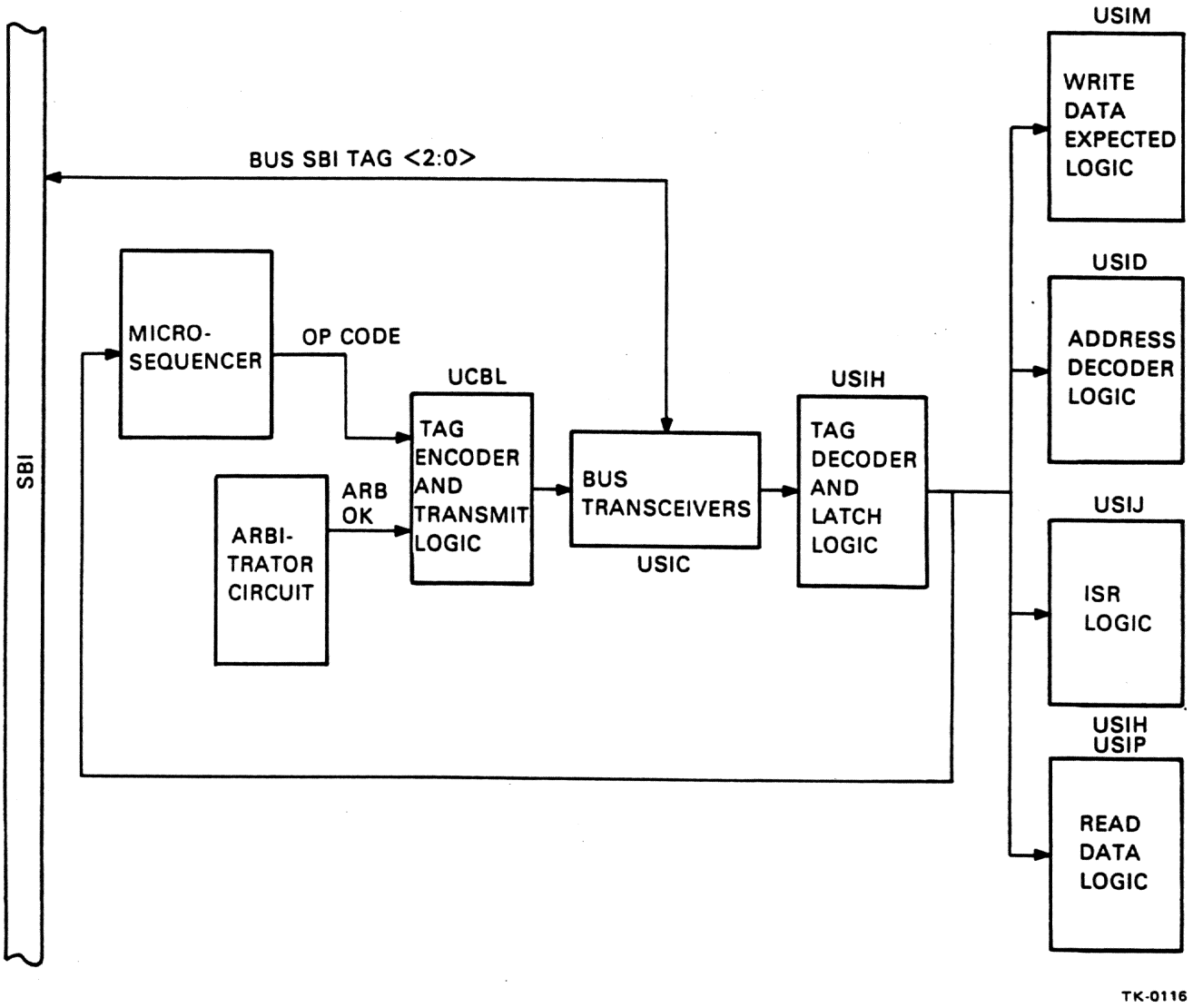


Figure 2-61 Tag Logic, Block Diagram

2.10.7 Parity Field

The UBA parity logic checks the parity of all information lines on the SBI with every SBI cycle. When the UBA becomes the transmitter, the parity generator circuit generates parity bits on the signals transmitted. Parity on the SBI is even.

If the UBA transmits a field with an odd number of bits asserted, the odd output of the parity generator will be true, causing the SBI transceiver to assert one parity bit and making the total even. If the UBA asserts an even number of bits on the SBI, the odd parity output will be false, and the parity bit on the SBI will remain unasserted.

When the UBA latches information at the bus transceivers (data that it has transmitted as well as data transmitted by other commanders), Parity OK will be true only if the number of SBI bits asserted is even.

2.10.8 Fault Field

The UBA checks for parity, write sequence, unexpected write data, interlock sequence, and multiple transmitters on every cycle on the SBI. When it detects an error, it latches the signal indicating the error, and transmits FAULT on the SBI. At this point the appropriate bit of the configuration register is set, as shown in Figure 2-62. Once FAULT has been asserted on the SBI, the data latched in the configuration register remains unchanged until the software can read the register.

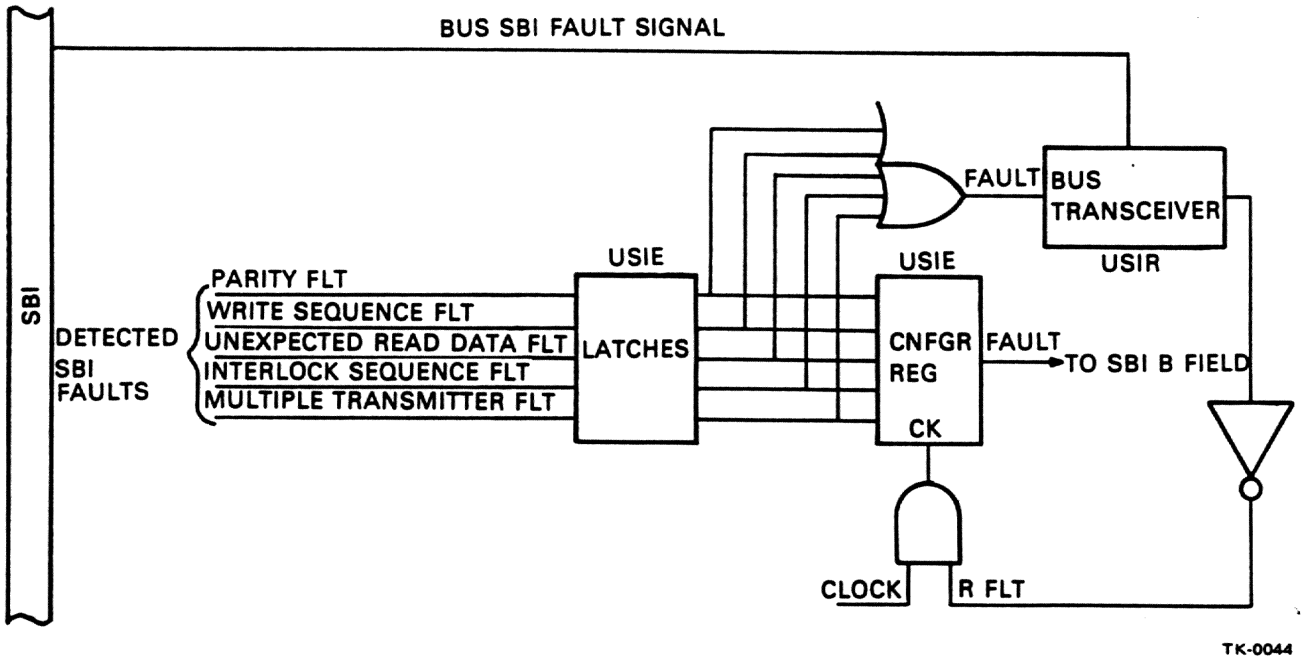


Figure 2-62 Fault Logic, Block Diagram

2.11 UNIBUS INTERFACE

The UAI module functions as an interface between the Unibus and the UBA. It includes the Unibus transceivers (with the exception of the DATO transceivers) as well as the arbitration, control, interrupt fielding, and power fail logic for the Unibus.

2.11.1 NPR Arbitration

When the UBA requires control of the Unibus, it must assert NPR, like the I/O devices. However, the NPR signal may be generated at any of three sources, and the UBA must respond differently in each case.

First, the UBA may issue NPR in order to pass a bus grant [BG(7:4)] to a Unibus device in response to a previous bus request [BR(7:4)]. Second, the UBA may enable NPR in order to carry out a data transfer initiated by the software. Third, an I/O device may assert NPR on the Unibus in order to initiate a DMA transfer.

NPR in Response to a Bus Request – When an I/O device asserts one of the four BR signals on the Unibus, the UBA will pass the request directly to the SBI if enabled by software (IFS) as explained in Paragraph 2.9.8.

Following the ISR exchange, the UBA interrupt service routine reads the BRRVR. If the BRRVR is empty and the terminator (UBT) has not already asserted NPG, the UBA will assert BUS NPR OUT L on the Unibus. This enables the UBA to gain control of the Unibus, so that no other device can start a transfer before the UBA can issue BG and pass control to the interrupting device (refer to Paragraph 3.5.4.2 for further details).

NPR in Response to a Data Transfer Command Issued by the Software – When the software issues a read or write command to a Unibus I/O device, the UBA will decode the command from the SBI, and assert UIP H (Unibus In Progress). If conditions are appropriate on the Unibus and the UBA, the UBA will assert BUS NPR OUT L.

NPR issued by an I/O Device – Any Unibus device capable of initiating high speed data transfers to and from memory without assistance from the processor may issue NPR once the software has set up the parameters of the transfer.

Handling the NPR – The terminator (UBT) receives the NPR signal and, if the bus is not busy, it issues NPG.

If the software is responding to a bus request, the UBA will accept the NPG and assert the appropriate BG signal on the Unibus as explained. If the software has initiated a data transfer to the Unibus, the UBA will accept the NPG, becoming the next bus master. If an I/O device has issued the NPR as the first step of a DMA transfer, the UBA will pass the NPG through to the I/O device, allowing it to become the next Unibus master, as shown in Figure 2-63.

2.11.2 Unibus Control by the UBA

The UBA must become Unibus master to handle software initiated data transfers to Unibus I/O space. After gaining control of the Unibus, the UBA follows the Unibus protocol like any other Unibus device.

The UBA uses an eight-state counter located on the UAI module to step through the various stages of the Unibus protocol. The counter should be in the zero state when the UBA is idle. Then, when the software initiates a transfer, the UBA asserts NPR to gain control of the bus. When the terminator asserts NPG, the UBA takes the grant and becomes the next bus master. The eight-state counter calls for the attention of the microsequencer only in the case of a read access to the Unibus when UBA must gain control of the SBI and then assert the read data on the SBI.

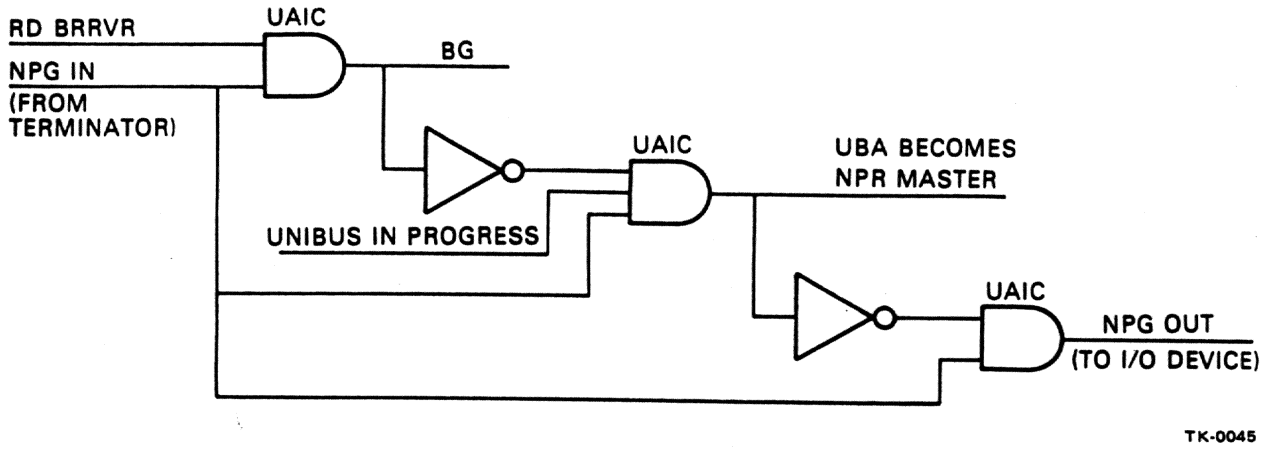


Figure 2-63 NPR and NPG Logic, Functional Block Diagram

2.12 UBA FUNCTIONAL CONTROL (UCB MODULE)

The microsequencer (microprocessor) and its associated logic occupy one of the four modules on the UBA. The circuit consists of a 44 bit by 512(10) word ROM, attention and address generation logic, and UBA control circuits fed by the ROM word. The UBA is able to perform some functions, such as software-initiated write data transfers, without calling the microsequencer. However, action by the microsequencer is necessary for UBA initialization, software access to some registers internal to the UBA, DMA transfer, software-initiated read data transfer to a Unibus device, data transfer timeout control, and interrupts from the UBA and from a Unibus device. Figure 2-64 shows the microsequencer in block diagram form.

When the microsequencer is in the idle state, it loops at location 003(8), waiting for signals from the attention select logic. An event requiring attention will cause the microsequencer to jump to the starting address of the appropriate microroutine. The selected routine will control the flow of data and address information through the UBA; monitor conditions on the SBI, the Unibus, and the UBA; and return the microsequencer to the idle state when finished.

2.13 MAJOR CONTROL FUNCTIONS ON THE UBA

The UBA responds to commands and bus requests on the SBI and the Unibus by activating various logic circuits that perform the required function. For example, when the UBA receives a command address on the SBI, it decodes the address, function, and mask field. If it detects a fault, the UBA enables the fault logic. If the decoder circuits interpret the command/address as a write to the CNFGR or the DCR, the USI register strobe logic will be enabled. Notice that most, but not all, of the control functions involve activation of the microsequencer. Figure 2-65 is a simple diagram showing the relation of the major UBA control functions and logic circuits on the UBA.

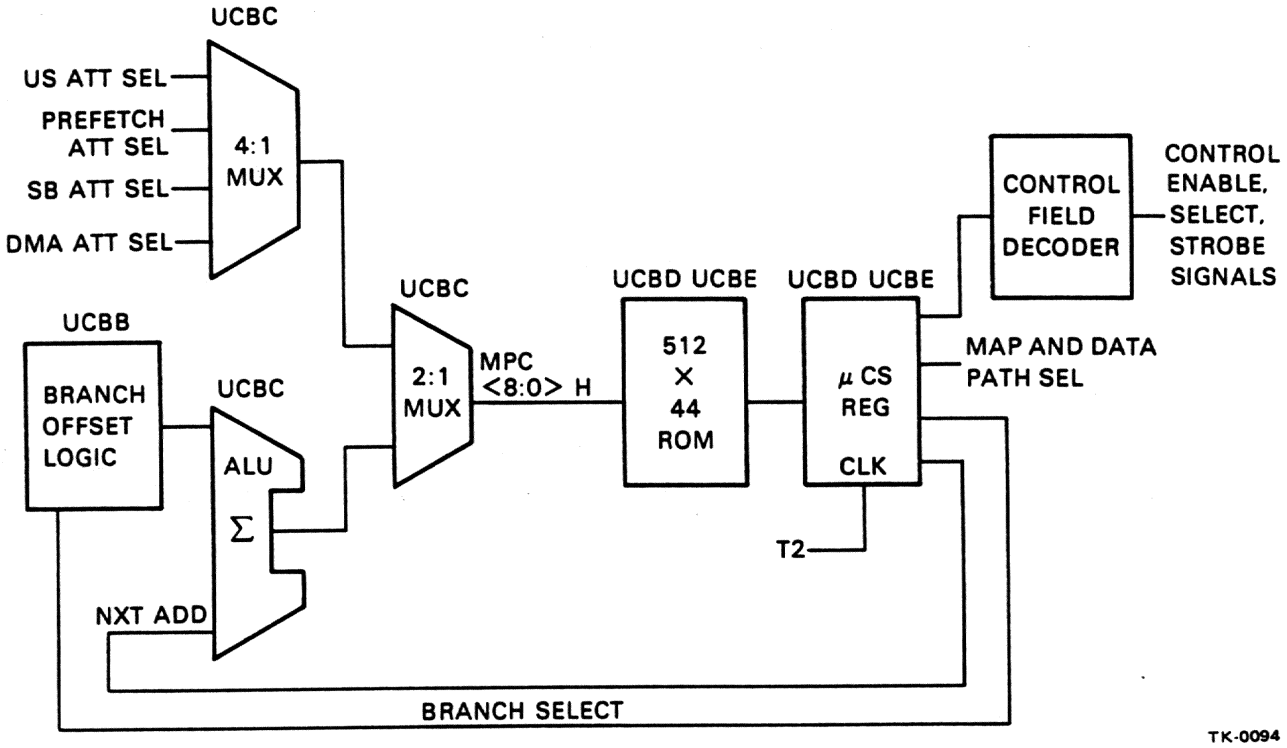


Figure 2-64 Microsequencer, General Block Diagram

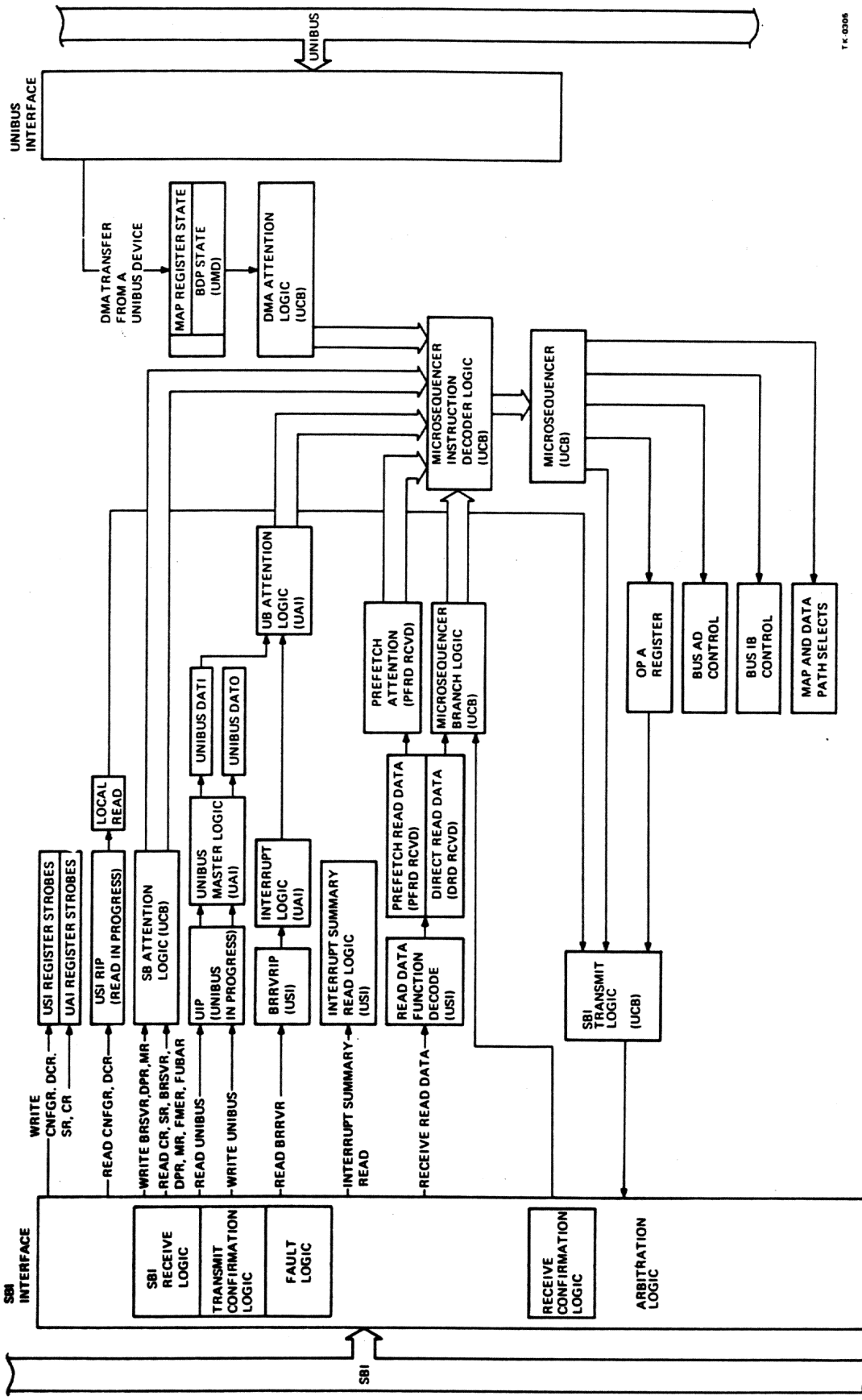


Figure 2-65 Simplified Flow of Major Control Functions within the UBA

CHAPTER 3
DETAILED LOGIC DESCRIPTION

3.1 MICROSEQUENCER LOGIC

A thorough knowledge of the workings of the UBA microsequencer is essential to an understanding of UBA operation. The microsequencer consists of a 512(10) × 44(10) bit ROM, ROM address generation logic, a 44 bit microsequencer register that is clocked every 200 ns, and signal decoder and distribution logic. The UCB module contains the microsequencer logic. Figure 3-1 shows the circuit in block diagram form.

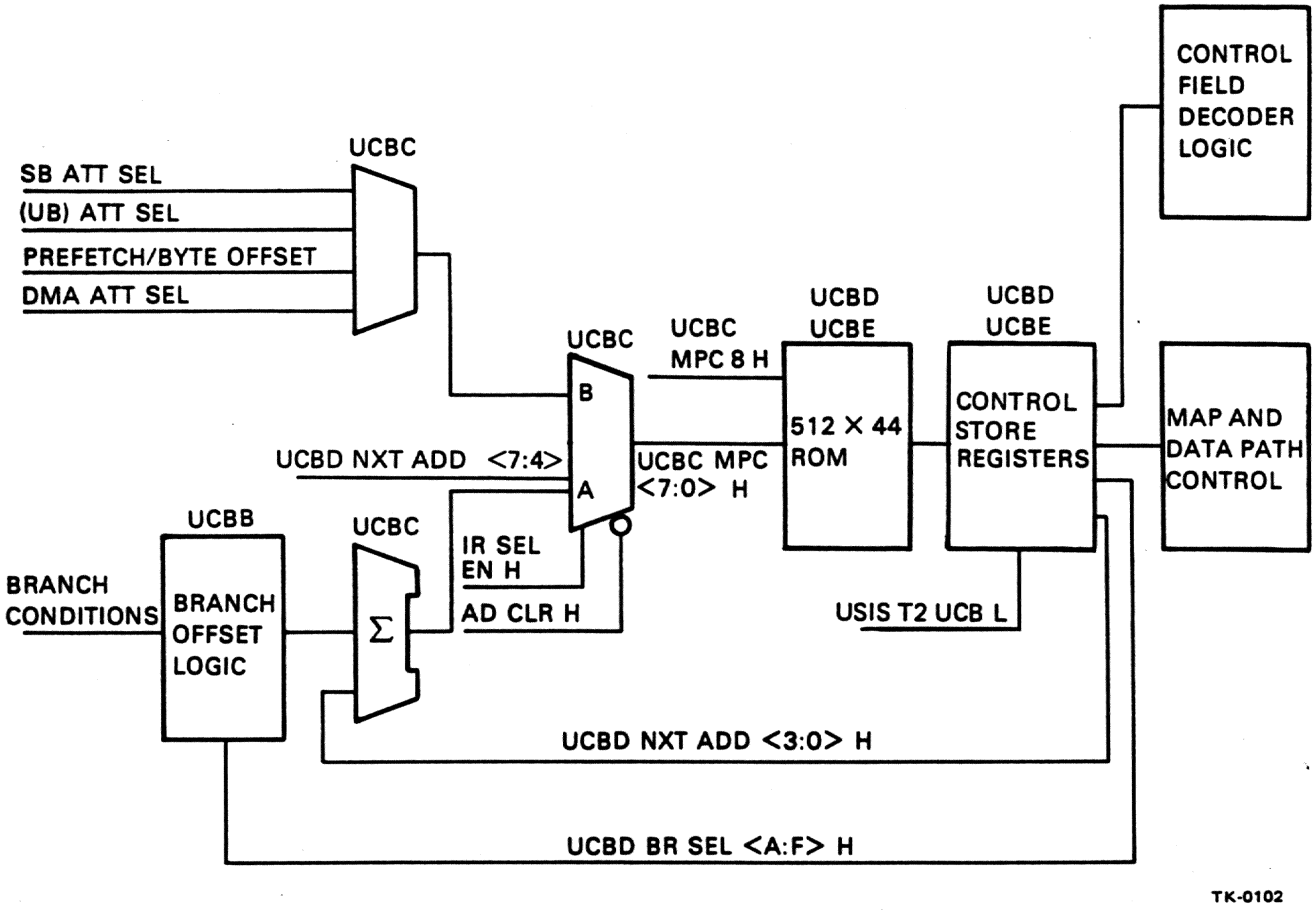


Figure 3-1 Microsequencer Block Diagram

The 44-bit ROM word is broken up into seven fields, as shown in Figure 3-2.

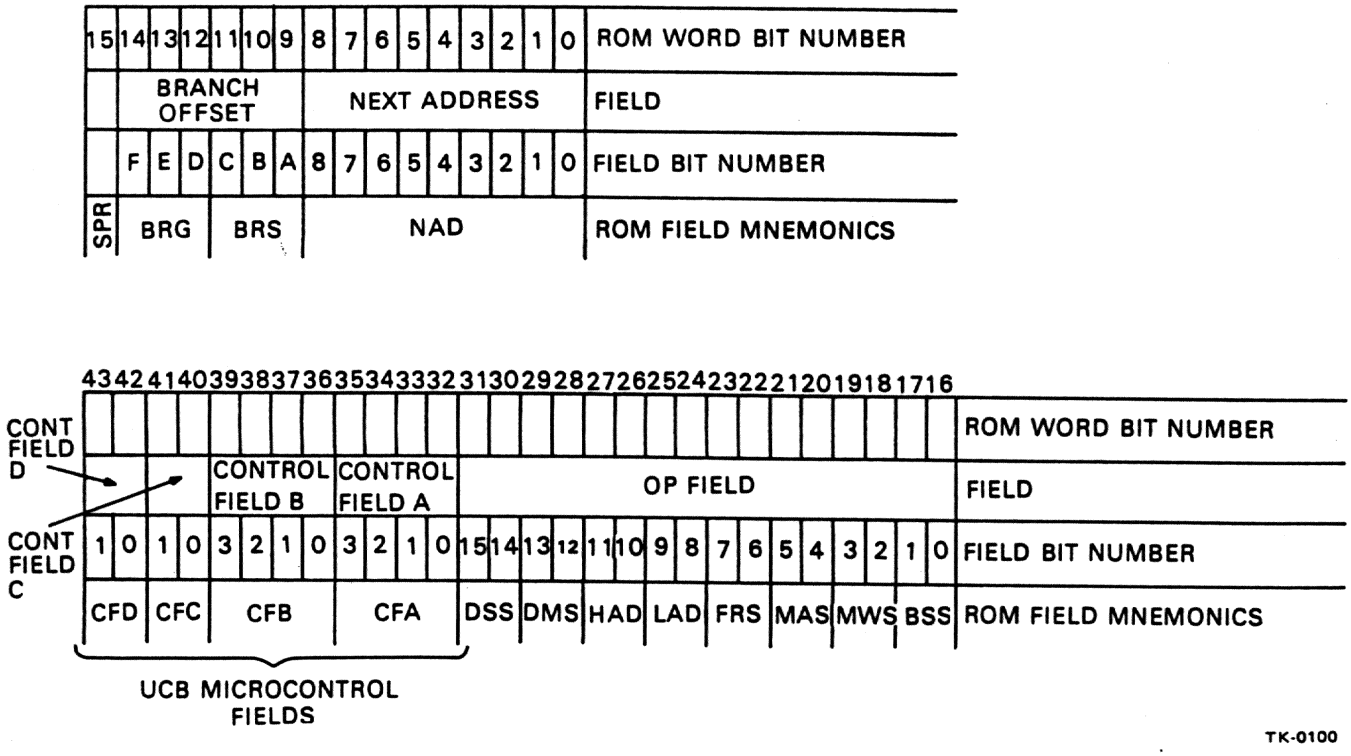


Figure 3-2 Microsequencer ROM Fields

3.1.1 Next Address Field

The first nine bits of the ROM word make up the next address field (NAD). The microsequencer will always take its next ROM word from the location specified in the next address field, from the next address field plus a branch offset, or from the attention select circuit. The next address field is given in the ROM listing as a column of three octal digits. The nine NAD bits are brought out of the ROM, summed with the branch offset (if any) and fed into a 2:1 multiplexer to form UCBC MPC (8:0) H, the ROM address lines, as shown in Figure 3-1. Note that although the contents of the location addressed will always be enabled on the ROM output lines, they do not form the control word until clocked into the microcontrol store register at T2, as shown in Figure 3-3.

3.1.2 Branch Offset Field

The branch offset field is six bits wide. It is made up of two subsets, BRG (Branch Group) and BRS (Branch Select), as indicated in the microsequencer guide. The branch offset field implements four types of branch functions. The branch functions enable the microsequencer to test conditions on SBI, the Unibus, or on the UBA, pertinent to the routine in progress, and to take appropriate action. For instance, the microsequencer will test the BRRVR Full bit to determine whether or not to send the contents of the BRRVR directly to the VAX-11/780 CPU during the interrupt dialogue.

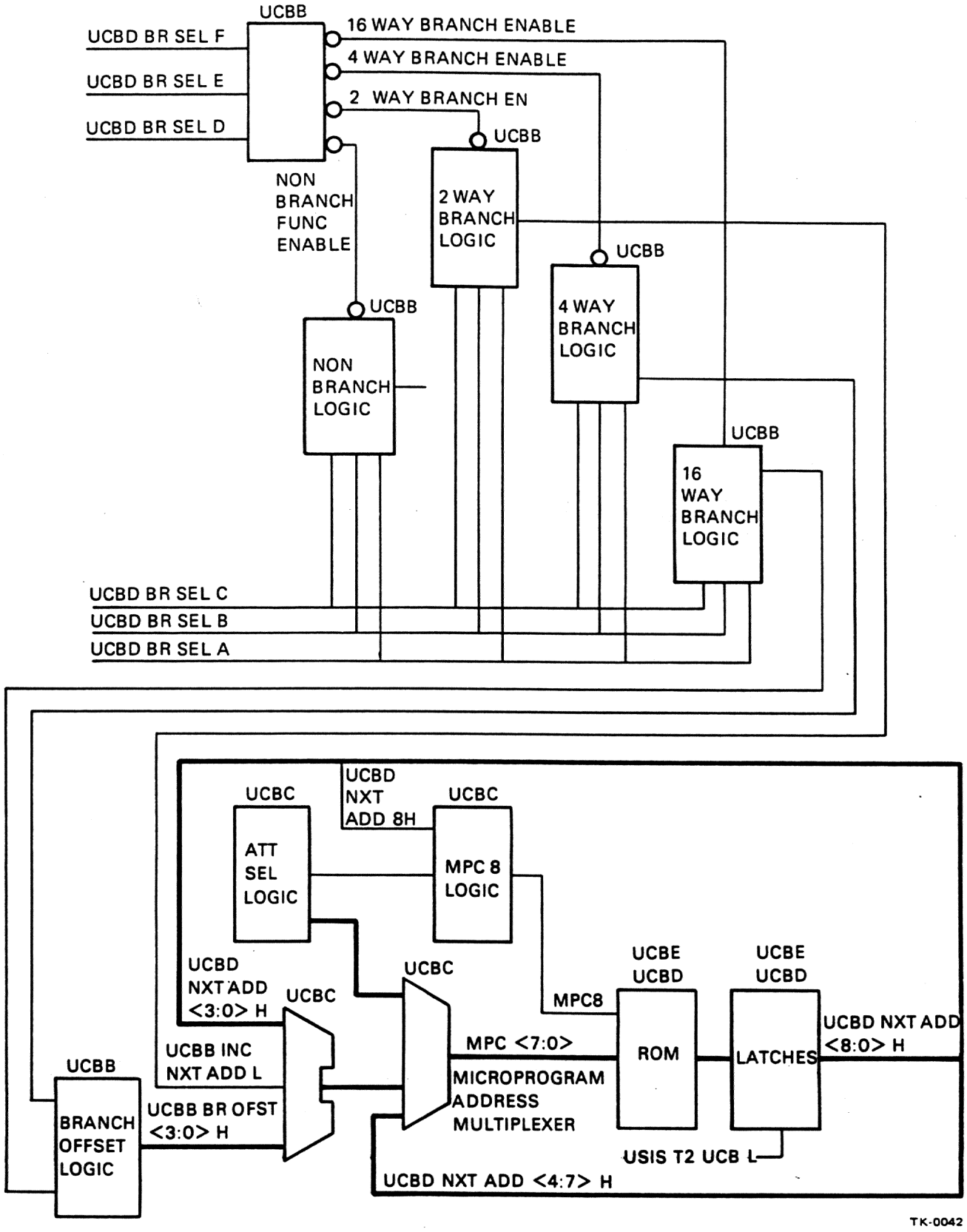


Figure 3-3 Next Address and Branch Logic, Block Diagram

Two Way Branch
If the number in the BR field is in the range of 0–27(8), the microsequencer will test the condition of the signal specified on the chart and in Table 3-1.

If the condition of the signal tested is true, the contents of the next address field are increased by one. Otherwise, the next address field remains unaltered.

Table 3-1 Two Way Branch Functions

BR		Two Way Branch Conditions	Test
G	S		
0	0	NEVER	F
0	1	SBI REQ FF	T
0	2	DP>0	T
0	3	EXTENDED FUNCTION	T
0	4	DTE (DATA TRANSFER)	T
0	5	SPARE	
0	6	BDP ACTIVE	T
0	7	ACK	T
1	0	UA01 (Unibus Address bit 1)	T
1	1	MSYN RLSD (Master Sync released)	T
1	2	DPPE (STRD) (Data path parity error)	T
1	3	SBI I/O PAGE	T
1	4	STRD BYTE OFFSET	T
1	5	RD FAILSAFE	T
1	6	PRG ACCESS OK	T
1	7	PRG RQD	T
2	0	COUNTER BIT 4	T
2	1	COUNTER FULL	T
2	2	SPARE	
2	3	TXPB (Transmit parity bit)	T
2	4	UB ATT SEL 2	T
2	5	SB ATT SEL 2	T
2	6	BRRVR FULL	T
2	7	INTR RLSD (Interrupt Released)	T

Four Way Branch
If the number in the BR field is between 40(8) and 52(8), the microsequencer will test the conditions of the two signals specified on the chart and in Table 3-2.

The branch offset logic forms a four-bit field based on the condition of the two signals selected by the BR field. The arithmetic logic unit shown in Figure 3-3 then adds the branch offset bits to the NAD field to generate the next address.

Table 3-2 Four Way Branch Functions

BR		Four Way Branch Conditions		
G	S	2^1	2^0	Notes
4	0	DP ADD 3	DP ADD 2	
4	1	SBI BIT 31	SBI BIT 30	1
4	2	UB ATT SEL 1	UB ATT SEL 0	
4	3	CNF 1	CNF 0	2
4	4	SPARE	SPARE	
4	5	UA 01 H	UA 00 H	
4	6	PFRD RCVD	RD FAILSAFE	
4	7	DRD RCVD	RD FAILSAFE	
5	0	SPARE	SPARE	
5	1	MIC STATE 1	MIC STATE 0	OP A REG
5	2	SB ATT SEL 1	SB ATT SEL 0	

Notes:

1 SBI B Field Bits — Function (when the software has written to a data path register)

31	30	
0	0	No-op
0	1	Clear BTE
1	0	Purge
1	1	Clear BTE - Purge

2. CNF Bits — SBI Response

1	0	
0	0	No Response
0	1	Acknowledge
1	0	Busy
1	1	Error

Non-Branch Functions

The six non-branch functions are listed in Table 3-3. The microsequencer uses them to develop strobe signals that enable or manipulate other fields in the ROM word.

Table 3-3 Non-Branch Functions

BR		
G	S	Function
6	0	IR SELECT
6	1	CHANGE MASK
6	2	CHANGE STATE
6	3	STROBE OPA REGISTER
6	4	STROBE OPB REGISTER
6	5	CHANGE MASK AND CLEAR DMA ATT
6	6	SPARE
6	7	SPARE

A BR field of 63, for instance, enables the OPA Register strobe signal.

Sixteen Way Branch
The microsequencer implements the two 16 way branch functions only in connection with DMA transfers. The four signals selected (Table 3-4) are used to produce a four bit branch offset to be added to the NAD field.

Table 3-4 16 Way Branch Functions

BR		16 Way Branch Conditions				
G	S	2^3	2^2	2^1	2^0	Function
7	0	0	Byte Offset	Unibus Address Bit UA02	Unibus Address Bit UA01	Prefetch Read Data Received
7	1	Byte Offset	Unibus Address Bit UA02	Unibus Address Bit UA01	Unibus Address Bit UA00	Prefetch Read Data Received Byte Offset

3.1.3 OP Field
The OP field of the ROM word is 16 bits wide and serves five general purposes. Figure 3-4 shows the distribution of these signals in block diagram form.

3.1.3.1 Data Path and Map Register Control – First, the OP field is divided into eight two-bit subsets. These signals form the select signals for some of the multiplexers, the RAMs, and the shifter, in the data path and map register circuits, as shown in Figure 3-5. An outline of the functions of these control signals follows.

Buffered Data Path Source Select (BSS)
These signals select one of four sources for the buffered data path address. The four sources are the data path designator bits [bits (24:21)] of the selected map register, register address bits (3:0) (from the SBI), ground (0), and BDP NO IN (3:0) from bits (4:1) of the initialization and timeout counter.

Map Word Select (MWS) (one of this pair is a spare)
This signal selects the high or low portion of the 32 bit word on the internal bus as the input to the map registers. It also forms map address bit 00 (Figure 3-5) placing the selected information in the appropriate 16-bit portion of the map register RAM.

Map Address Select (MAS) (one of this pair is a spare)
This signal selects Unibus address bits UA (17:09), converted to BUS AD (15:07), or register address bits (8:0) as the map address.

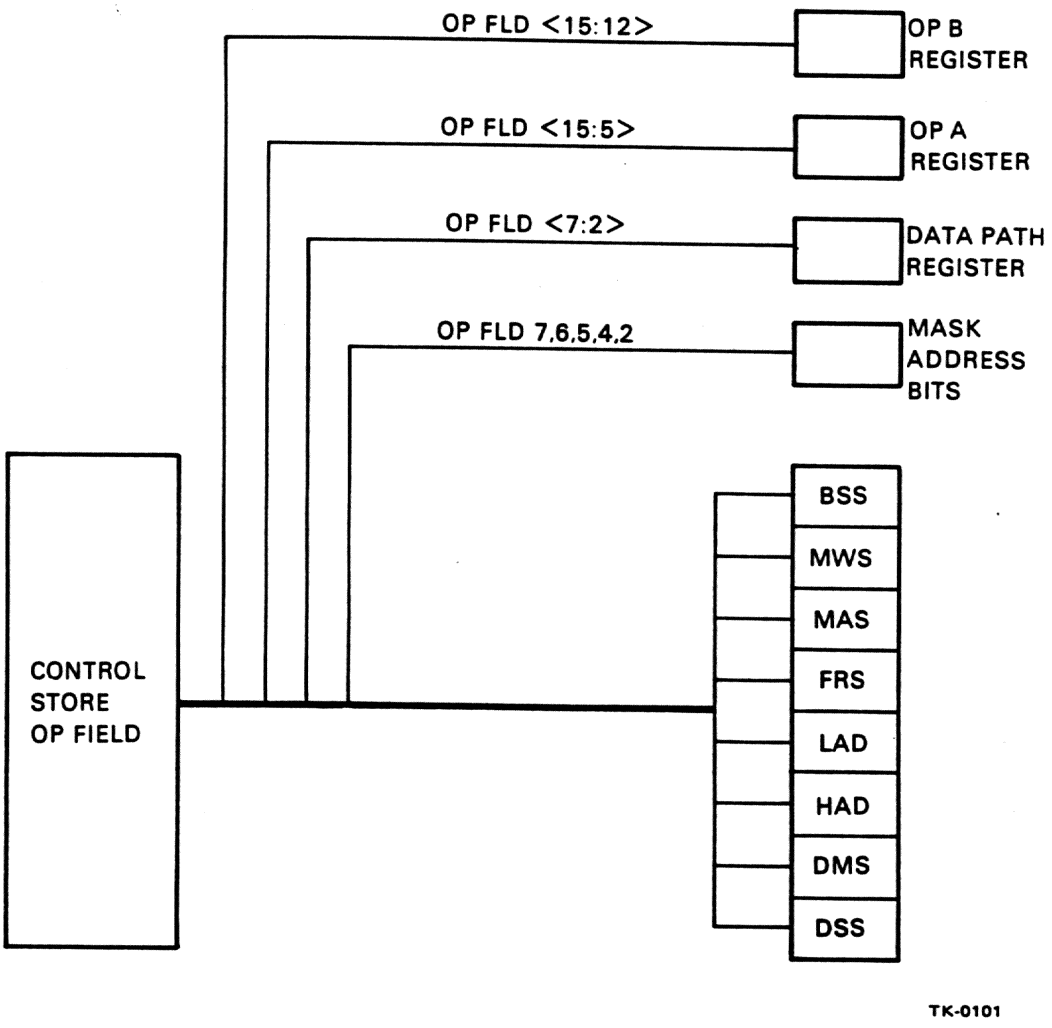


Figure 3-4 Control Store OP Field Signal Distribution

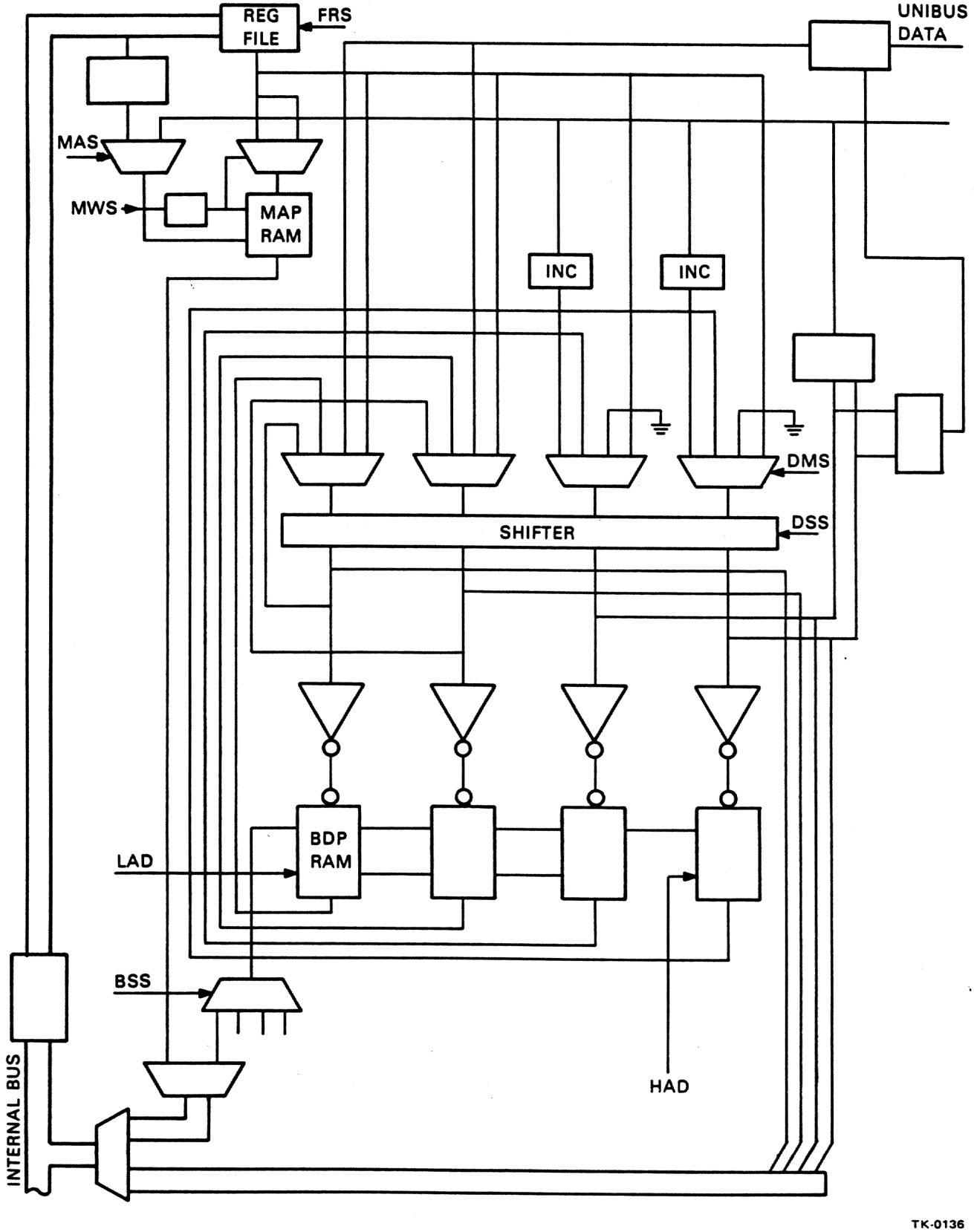


Figure 3-5 OP Field Signal Distribution to Map and Data Path Logic

File Read Select (FRS)
These two bits select one of four 32-bit words in the data file register for output to the internal bus, as shown in Table 3-5.

Table 3-5 FRS Functions

FRS	Data File Output	UBA Function
0	WRITE DATA	MAP WRITE
1	READ DATA 1	DMA READ
2	READ DATA 2	DMA READ
3	SPARE	

Low Data Path Address (LAD)
Each buffered data path is organized as four longwords. The two LAD bits are multiplexed with UCBE WD2 SEL H and ground to generate UCBF LDP ADD 1, 0 to select the lower three bytes of one of the four words in any of the 16 data path RAM addresses (15 BDPs and one address for the BRSVRs and the BRRVRs).

High Data Path Address (HAD)
Likewise, these two bits are multiplexed with UCBE WD2 SEL H and ground to generate UCBF HDP ADD 1, 0 to select the upper byte of one of the four longwords in any of the 16 data path addresses. The upper byte is separately addressable to enable the UBA to perform the byte offset function on DMA transfers. Figure 3-6 shows the function of the LAD and HAD signals with respect to the organization of the data path RAM.

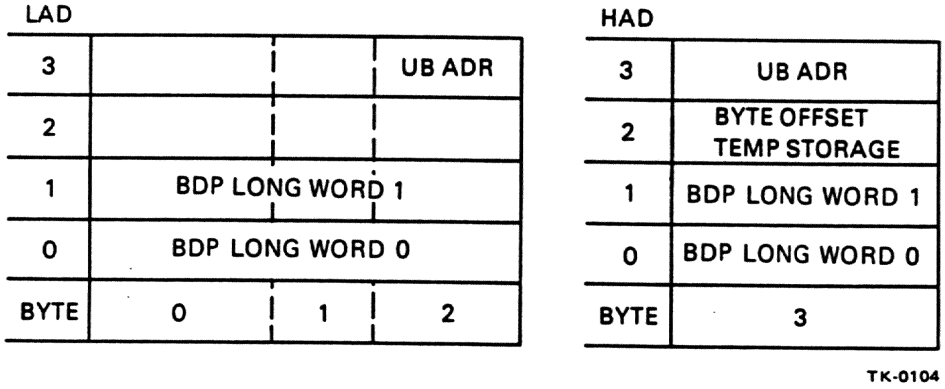


Figure 3-6 HAD and LAD Functions, BDP 1-15

Data Path Multiplexer Select (DMS)
The two DMS bits select one of four sets of input signals to the data path multiplexer (Figures 3-5 and 3-7).

	DMS <1:0>	DATA PATH MUX OUTPUT SFTR BYTE 0	SFTR BYTE 1	SFTR BYTE 2	SFTR BYTE 3
4 BYTES OF SBI DATA	0	FILE (0)	FILE (1)	FILE (2)	FILE (3)
2 BYTES OF UNIBUS DATA	1	RECD (0)	RECD (1)	MAP OUT 0	MAP OUT 0
4 BYTES OF DATA PATH RAM	2	DP RAM (0)	DP RAM (1)	DP RAM (2)	DP RAM (3)
2 BYTES WRAP AROUND 2 BYTES INC UB ADR	3	TMPSTR (0)	TMPSTR (1)	ADR INC (0)	ADR INC (1)

TK-0108

Figure 3-7 DMS Functions

Data Path Shifter Select (DSS)
These two bits control the data path shifter. With a DSS field of 2, for example, the data will be shifted 16 bits down (from MSB to LSB) with the lower two bytes wrapped around to the top, as shown in Figure 3-8.

DSS <1:0>	SFTR (0) OUT	SFTR (1) OUT	SFTR (2) OUT	SFTR (3) OUT } SHIFTER BYTE OUTPUT
0	SFTR (0) IN	SFTR (1) IN	SFTR (2) IN	SFTR (3) IN
1	SFTR (1) IN	SFTR (2) IN	SFTR (3) IN	SFTR (0) IN
2	SFTR (2) IN	SFTR (3) IN	SFTR (0) IN	SFTR (1) IN
3	SFTR (3) IN	SFTR (0) IN	SFTR (1) IN	SFTR (2) IN

TK-0177

Figure 3-8 DSS Functions

3.1.3.2 Mask PROM Address Bits – Bits 7, 6, 5, 4, and 2 of the OP field form the address bits for the mask PROM. These signals are always enabled, causing the mask PROM to display the contents of the location selected at all times. However, the data path register and the latches that store the PROM outputs will not change unless enabled by other fields in the microsequencer word. Refer to Paragraph 3.4.6 and Figure 3-32 for further details.

3.1.3.3 Buffer Status Bit Control – Six of the OP field bits are used to form the upper three bits of the data path register, as shown in Figure 3-9.

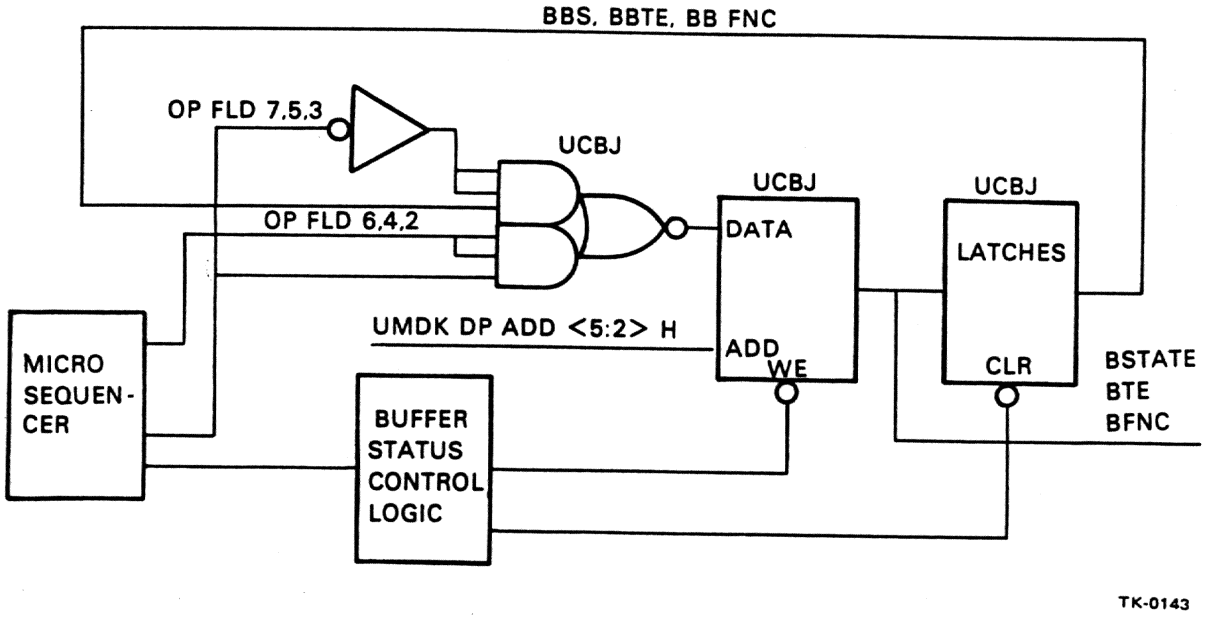


Figure 3-9 Buffer Status Bit Logic

With OP field bit 7, 5 or 3 false, the corresponding latched output bit of the data path register is wrapped around to feed the register input line. In this case the states of these bits remain unchanged. With OP field bit 7, 5, or 3 true, OP field bit 6, 4, or 2 will be enabled as the data path register input. In this case, the states of the bits are changed to the values of OP field bits 6, 4, or 2. These upper three bits of the data path register feed the select and branch logic of the microsequencer.

3.1.3.4 OP A Register – When the BR field equals 63, the UCBH OP A STRB H signal strobes twelve of the OP field bits into the OP A register [bits (15:5) and 2]. (Table 3-6). The lowest OP A register bits form the OP code bits (4:0). These bits constitute strobe and select signals that control the function and mask logic and other multiplexers and registers used on SBI transmit operations. Table 3-7 lists the OP code bit patterns that correspond to various UBA functions.

The upper two OP A register bits form inputs to the four way branch logic. The names of the other bits are self-explanatory.

3.1.3.5 OP B Register – The fifth major use of the OP field concerns the OP B register. When the BR field of the ROM word equals 64, the UCBA OP B STRB H signal strobes the upper four bits of the OP field into the OP B register, as shown in Figure 3-10. The OP B register signals control the flow of information on the UBA bus, BUS AD (15:00). Notice that the OP A and OP B registers will not change with every SBI cycle, when the microsequencer word changes. These registers will change only when strobed by microsequencer signals or cleared by AD CLR L during the power fail, initialization, or UNJAM sequences.

Table 3-6 OP A Register Functions

OP Field	
Bit No.	**Function**
0	
1	
2	OP CD 0
3	
4	
5	OP CD 1
6	OP CD 3
7	OP CD 2
8	OP CD 4
9	ENABLE CNTR INC
10	IB EN UAI
11	IB EN UMD
12	DP MUX L DIS
13	DP MUX H DIS
14	MIC STATE 0
15	MIC STATE 1

Table 3-7 OP Code Functions

OP CD Bit 43210	Adaptor to SBI Function and Mask
00100	Read Data, Mask = 0
00110	Read Data, Mask = F(PB)
01100	Read Data, Mask = F(DP PAR)
01110	Read Data, Mask = F(0)
00011	BDP Read: MASK EXT = 0, MASK LW = 1
10011	BDP Write, Mask = STRD
X0101	DDP Function, F (A1, A0, C1, C0)
01001	Prefetch Extended Read
11001	Purge
10100	BRRVR RD Mask = 0
11100	BRRVR RD Mask = DP

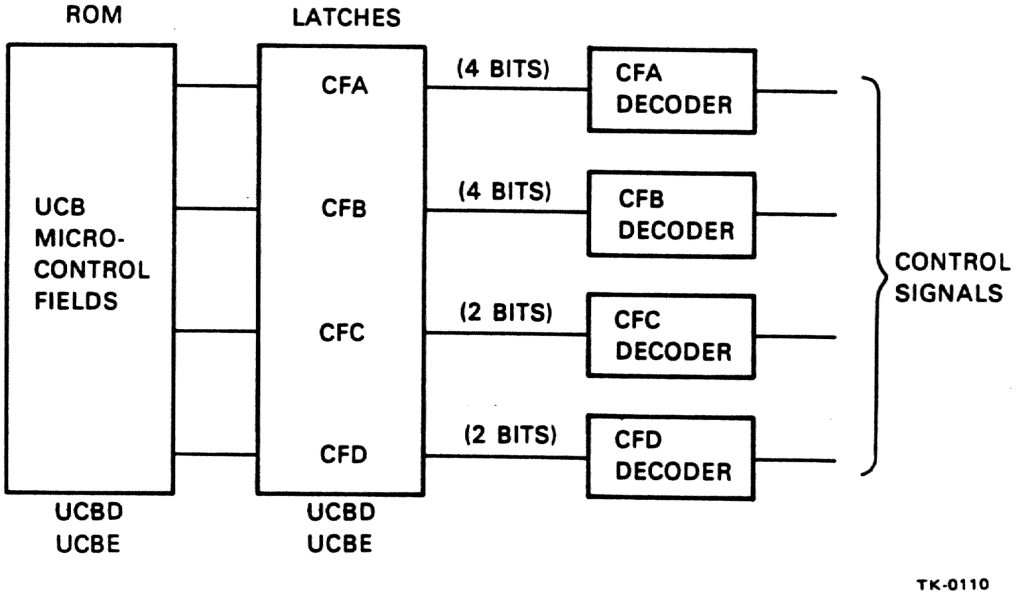
OP FIELD	OP B REGISTER	FUNCTION
12	CNTR SEL	SELECT COUNTER OR DPR → BUS AD LINES, DISABLE MAP
13	BAS TSE	ENABLE STORED ADDRESS FROM SHIFTER HIGH WORD OUTPUT TO BUS AD LINES (USED ON PURSE)
14	AD TSE	ENABLE COUNTER/DPR MUX OUTPUT TO BUS AD LINES
15	ADD HLD TSE	ENABLE UNIBUS ADDRESS HOLD LATCHES TO BUS AD LINES

TK-0109

Figure 3-10 OP B Register Functions

3.1.4 UCB Microcontrol Fields

The four UCB microcontrol signals form the upper 12 bits of the ROM word. Decoder circuits on the UCB module receive the ROM word bits and enable appropriate control, clear, and strobe signals, as shown in Table 3-8 and Figure 3-11.



TK-0110

Figure 3-11 UCB Microcontrol Fields, Block Diagram

Table 3-8 UCB Microcontrol Field Functions

Octal	CFD	CFC	CFB	CFA	Octal
0	NO OP	NO OP	NO OP	NO OP	0
1	DP3 WE	DP2 WE	DP1 WE	DP0 WE	1
2	EN SSYN	EN PB CLK	HI DOUT STRB	LO DOUT STRB	2
3	CLR CNTR	SBI REQ EN	CNG STATE, SND SSYN	BUF ADD STRB	3
			CNG MASK AND STATE	TMP STR STRB	4
			CLR MR HI GATE	CLR RD RCVD	5
			EN DPPE CATCH	CLR SB ATT	6
			AD INC EN	CLR DMA ATT	7
				CLR UB ATT	10
			EN DPPE STAT	EN MR HI LAT	11
			EN CXTER	CLR BRRVR FULL	12
				INTLK SET EN	13
				EN MAP STAT	14
			EN CATO	END DMA CLR H	15
			WRITE MAP	SPARE	16
				MIC OK	17

3.2 MICROSEQUENCER IDLE STATE AND IR SELECT LOGIC

When the UBA is not active, and when the microsequencer is not currently executing a microroutine, the microsequencr remains in the idle state, looping at address three. All fields in the ROM word contained in location three are zero, with the exceptions listed below. Notice that ROM contents and addresses are listed in octal notation in the ROM listing.

Field	ROM Output
NAD	003(8)
BR	60(8)
MWS	2(8)
CFA	17(8)

The NAD field of 003(8) indicates that the address of the next microsequencer word to be accessed is 003(8), producing a one instruction loop. The BR field of 60 enables the UCBH IR SEL EN signal shown in Figure 3-12. The A side of the 2:1 multiplexer remains enabled, gating the NAD field, unincremented, through to the ROM address lines as long as the input signals to the attention encoder are all unasserted. Then, when an event on the SBI, the Unibus, or the UBA occurs, calling for attention by the microsequencer, the GS output of the attention encoder will be enabled, causing the 2:1 multiplexer to select the B inputs, which come from the attention select logic. The address produced will be the starting address of a microroutine. Table 3-9 lists the starting addresses and corresponding functions of the UBA microroutines.

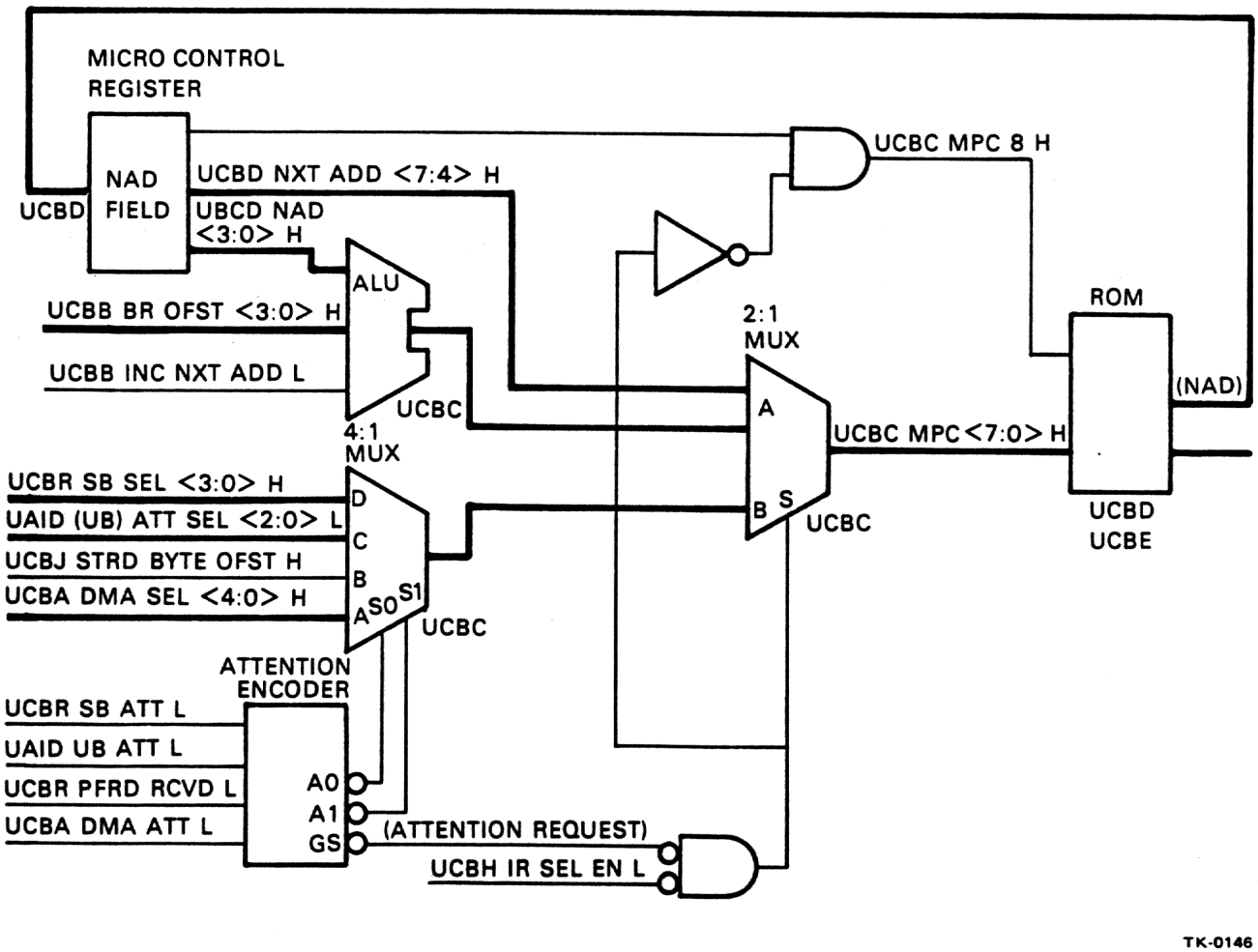


Figure 3-12 IR Select Logic

Table 3-9 Microsequencer Starting Addresses

Starting Address (Octal)	IR Select	Flowchart Symbol	Function		
0	X	CLR UBA	INITIALIZE UBA		
1	X		RSVD		
2	X		RSVD		
3	X	STRT	BR (IR) ATT		
4	A	DMA D0	SBI READ (DDP)	DDP	DMA
5	A	DMA D1	SBI WRITE (DDP)	DDP	DMA
6	A	DMA D2	STORE RD, SBI READ (DDP)	DDP	DMA
7	A	DMA D3	SBI WRITE = D1	DDP	DMA
10	A	DMA B0	SBI READ (BDP)	BDP READ	DMA
11	A	DMA B1	STORE RD SBI READ (BDP)	BDP EMPTY	DMA
12	A	DMA B2	SBI READ (BDP) = B0		DMA
13	A	DMA B3	WAIT RD RTN		DMA
14	A	DMA A0	MEM DATO	BDP WRITE	DMA
15	A	DMA A1	MEM DATOB		DMA
16	A	DMA A2	I/O DATO		DMA
17	A	DMA A3	I/O DATOB		DMA
20	A	DMA C0	DATI B0, 1 MEM	BDP READ	DMA
21	A	DMA C1	DATI B2, 3 MEM		DMA
22	A	DMA C2	DATI B4, 5 MEM	BDP	DMA
23	A	DMA C3	DATI B6, 7 MEM	CONTAINS	DMA
24	A	DMA C4	DATI B1, 2 MEM (BYTE OFST)	DATA	DMA
25	A	DMA C5	DATI B3, 4 MEM		DMA
26	A	DMA C6	DATI B5, 6 MEM		DMA
27	A	DMA C7	DATI B7 - B0 MEM		DMA
30	A	DMA C8	SBI BDP READ, DATI B0, 1		DMA
31	A	DMA C9	DATI B2, 3 I/O		DMA
32	A	DMA C10	SBI BDP READ, DATI B0, 1 = C8		DMA
33	A	DMA C11	DATB2, 3 I/O = C9		DMA
34	A	DMA C12	SBI BDP READ DATI B0, 1 = C8		DMA
35	A	DMA C13	DATI B2, 3 I/O = C9		DMA
36	A	DMA C14	SBI BDP READ DATI B0, 1 = C8		DMA
37	A	DMA C15	DATI B2, 3 I/O = C9		DMA
40	C0		RESERVED		
41	C0		RESERVED		
42	C0		RESERVED		
43	C0		RESERVED		
44	C0		RESERVED		
45	C0		RESERVED		
46	C0		RESERVED		
47	C0		RESERVED		
50	C1	UBC 0	MASTER DATI TIMEOUT	UBC (UAI)	
51	C1	UBC 1	BRRVR SND RAM	UBC (UAI)	
52	C1	UBC 2	MASTER DATI RTN W1	UBC (UAI)	

Table 3-9 Microsequencer Starting Address (Cont)

Starting Address (Octal)	IR Select	Flowchart Symbol	Function	
53	C1	UBC 3	MASTER DATI RTN W0	UBC (UAI)
54	C1		RESERVED	
55	C1	UBC5	INTR FLD OK - RTN VECTOR	UBC (UAI)
56			RESERVED	
57	C1	UBC 7	NO-OP	UBC (UAI)
60	D	SB 0	NO-OP	SB (USI)
61	D	SB 1	UAI REG READ	SB (USI)
62	D	SB 2	DPR READ	SB (USI)
63	D	SB 3	BRSVR READ	SB (USI)
64	D	SB 4	MAP REG READ	SB (USI)
65	D	SB 5	DPR WRITE	SB (USI)
66	D	SB 6	BRSVR WRITE	SB (USI)
67	D	SB 7	MAP REG WRITE	SB (USI)
70	B	PF 0	PFRD RCVD	SB (USI)
71	B	PF 1	PFRD RCVD BYTE OFST	SB (USI)
400	X	CLR UBA	INITIALIZE UBA	

Table 3-10 shows the concept involved in selecting the microroutine starting addresses.

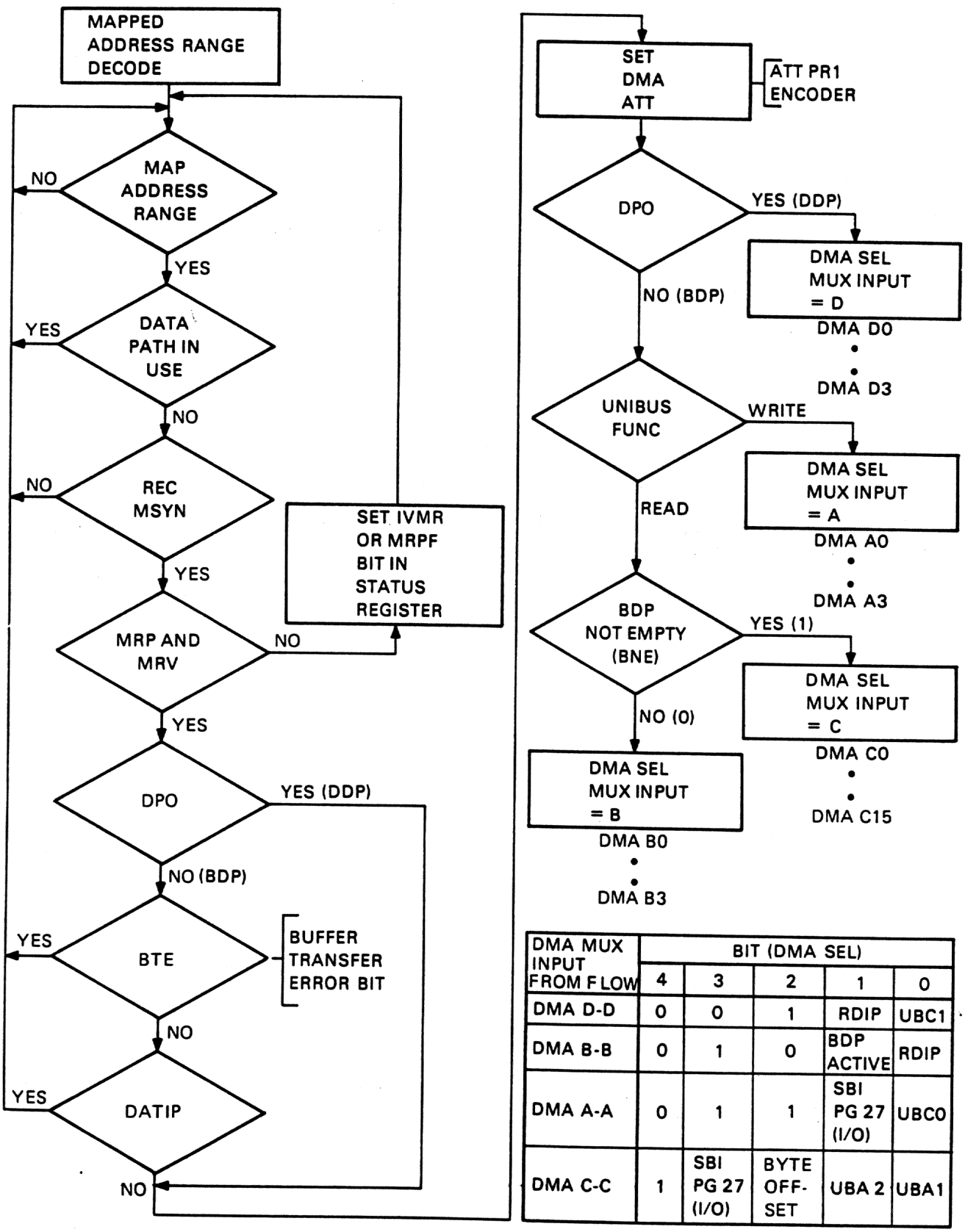
Table 3-10 Microroutine Starting Address Generation

MUX Inputs	Address Bits	Function	Priority
	8 7 6 5 4 3 2 1 0		
A	0 0 0 0 X X X X X	DMA ATT	3
C	0 0 0 1 0 1 X X X	UB ATT	1
D	0 0 0 1 1 0 X X X	SB ATT	0
B	0 0 0 1 1 1 0 0 B0	Prefetch RD RCVD	2

For example, the starting addresses for the Unibus attention microroutines run from 50(8) through 57(8).

Figure 3-13 further defines the starting address selection process for the DMA attention routines.

Notice that the D input of the DMA select multiplexer is enabled if the transfer is being mapped through the direct data path, regardless of whether a read or a write is being performed.



DMA MUX INPUT FROM FLOW	BIT (DMA SEL)				
	4	3	2	1	0
DMA D-D	0	0	1	RDIP	UBC1
DMA B-B	0	1	0	BDP ACTIVE	RDIP
DMA A-A	0	1	1	SBI PG 27 (I/O)	UBC0
DMA C-C	1	SBI PG 27 (I/O)	BYTE OFF-SET	UBA 2	UBA1

TK-0111

Figure 3-13 DMA Attention Flowchart

3.3 REPRESENTATIVE MICROSEQUENCER ROUTINES

The UBA microflow charts provide a step-by-step analysis of each microroutine. The reader should reference the flowcharts and the control store ROM listing in connection with the following explanations.

3.3.1 DMA Transfer Routines

When a Unibus device gains control of the Unibus and asserts control, address, and Master Sync signals (and data on a DMA write transfer) on the Unibus, the DMA sequence on the UBA begins. If the microsequencer is in the idle state, the selected map register has been validated by software, map parity is OK, the address and function asserted on the Unibus are valid (note that a DATIP mapped through a buffered data path is not valid), and UAIP REC MSYN H is latched, then UCBA DMA ATT H will be latched and asserted when USIS TO UCBL goes high, as shown in Figure 3-14.

Signals from a 4:1 multiplexer circuit feed the DMA ATT SEL latches and are latched on the positive edge of UCBA DMA ATT H, as shown in Figure 3-15. A decoder circuit is used to select one of the four multiplexer inputs, depending on the states of UCBJ DPO L (DDP), UCBK HOLD CI L (write), and UCBJ B STATE H (BNE).

3.3.1.1 Microroutine for DMA Read Through the DDP – If the Unibus I/O device has initiated a read transfer to be mapped through the direct data path, the DMA ATT SEL logic asserts the microsequencer address for the first location of the DMA D0 routine on the MPC lines (microflow sheet MF17).

The microsequencer assembles the high and low map words for command/address, requests the SBI, asserts command/address when enabled, and branches on the confirmation received. The UBA will repeat the command/address sequence on receipt of a BSY or NR confirmation. It will set a status register bit (CXTER) and return to the idle state on an ERR confirmation. If the UBA receives an ACK confirmation, the microsequencer will loop waiting for the return of read data or the read data timeout. When the UBA latches the read data in location RD1 of the data file, the microsequencer sets the Interlock flip-flop (UCBK) if the Unibus function is DATIP, branches on the state of Unibus address bit UA01 to assert the high or low Unibus word on the Unibus data lines, and returns to the idle state.

3.3.1.2 Microroutine for DMA Write Through the DDP – When the DMA attention logic selects the routine for the DDP to SBI write transfer (DMA D1, MF18), the microsequencer sets up the C/A word and requests the SBI. When the UBA gains control of the SBI, it transmits command/address, asserts the write data in the high or low portion of the B field on the following SBI cycle, and clears the Interlock flip-flop, if appropriate. If a timeout occurs or an ERR confirmation is received, the transfer is aborted and the microsequencer will wait for the commanding Unibus device to release Master Sync and then return to the idle state. The UBA will repeat transmission of C/A and write data upon receipt of a BSY or no response confirmation. If the UBA receives ACK for both the C/A word and the write data, it asserts Slave Sync on the Unibus and then waits for the release of MSYN before the microsequencer returns to idle. If no ACK is received for the write data, the entire write data sequence will be repeated.

3.3.1.3 Microroutines for DMA Read Through a BDP

3.3.1.3.1 Buffer Empty – The DMA attention logic will select the DMA B0 routine (MF9) when a DMA read transfer is to be mapped through an empty buffered data path. The microsequencer assembles and asserts the C/A word on the SBI and branches on the confirmation received, as in the read transfer mapped through the DDP. If the UBA receives ACK, it then branches to one of two subroutines (I/O WAIT or MEM BRANCH) depending on whether the memory or the I/O page has been addressed. The microsequencer then loops until it receives the data or a timeout occurs. Read data from SBI I/O space will be a longword, latched in RD1 of the data file. Read data from SBI memory will also be a longword, latched in RD1 of the data file, if bit A2 of the Unibus address asserted is true. Otherwise (SBI I/O PG and UA02 false), the read data will be a quadword from SBI memory, latched in RD1 and RD2 of the data file, in response to an extended read transfer.

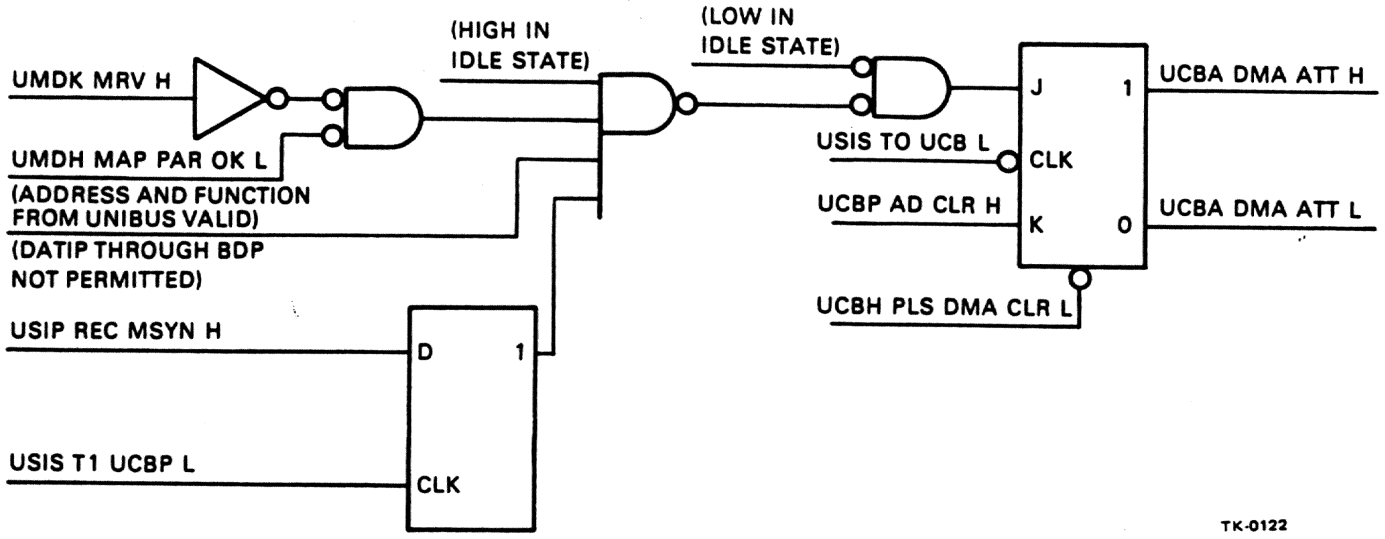


Figure 3-14 DMA Attention Logic

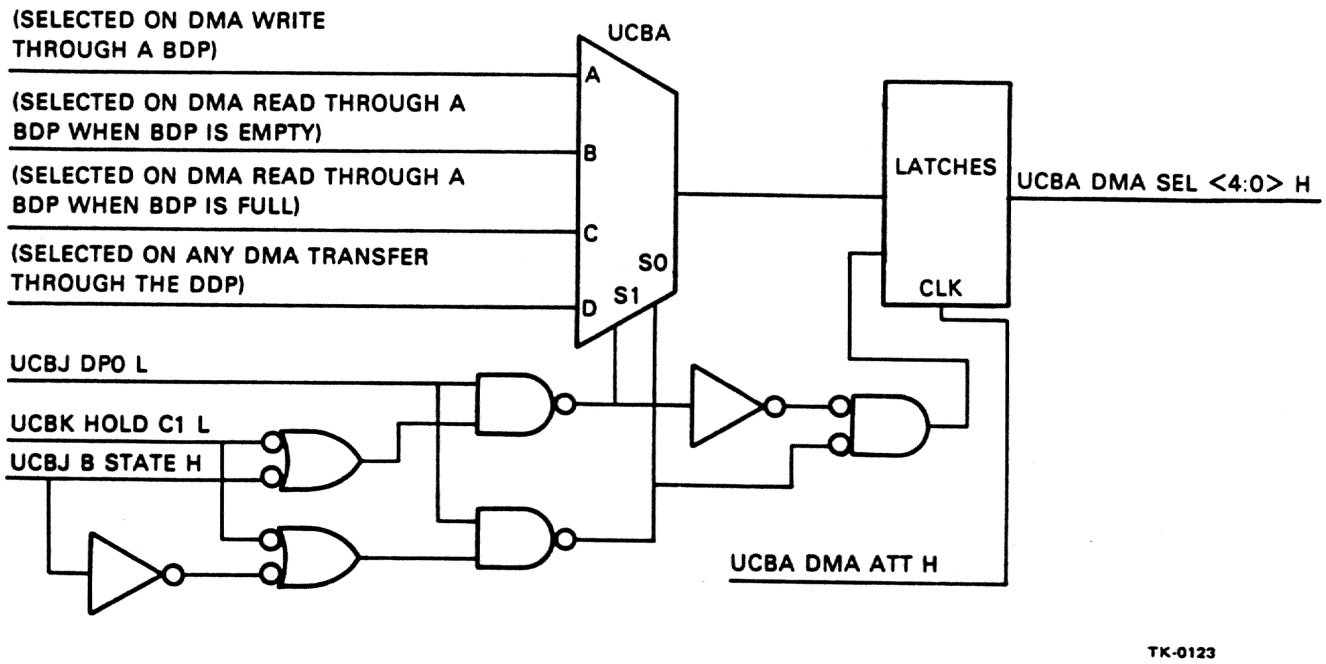


Figure 3-15 DMA Attention Select Logic

If the read data is returned from SBI memory, the microsequencer enables the 16 way branch logic, branching on the conditions of Unibus address bits UA02 and UA01 and the byte offset bit (see sheet MF10, MEM BRANCH).

The word addressed is then moved to the Unibus data transceivers and the remaining bytes (if any) are stored in the appropriate location in the buffered data path. For example, if the transfer is not in the byte offset mode and the address bits UA02 and UA01 are both false, the first word is addressed (bytes 0 and 1) and bytes 0–7 are stored in the BDP, as shown in Figure 3-16.

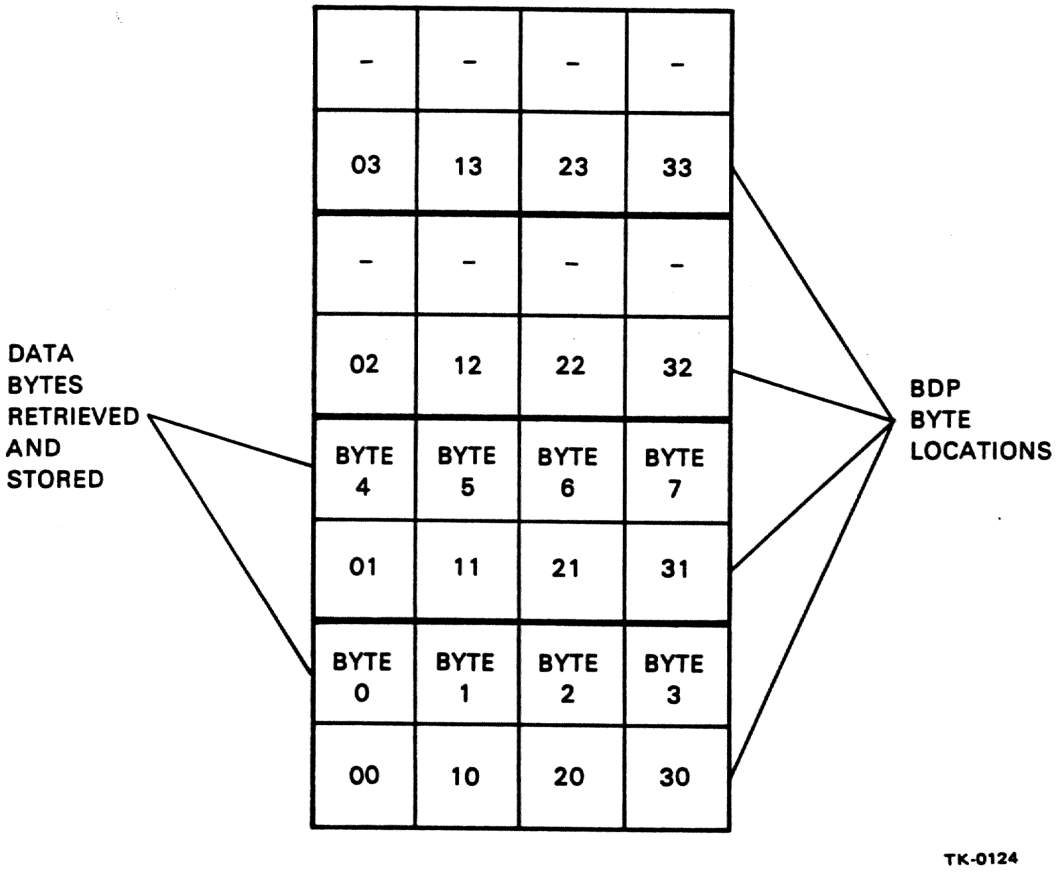


Figure 3-16 Use of BDP on DMA Read with B0, A2, A1 = 0

If UA01, only, is true (B0, A2, A1 = 1), the second word is addressed. Bytes 0 and 1 are ignored, bytes 2 and 3 are moved to the Unibus data transceivers via DOUT, and bytes 4–7, RD2, are stored in BDP locations 01, 11, 21, 31. If UA02, only, is true (B0, A2, A1 = 2), only four data bytes will have been retrieved. The first word, bytes 0 and 1 of RD1, is latched as DOUT, and the entire longword is stored in the first four locations of the BDP.

If UA02 and UA01 are both true (B0, A2, A1 = 3), the last word of the quadword is addressed. Bytes 2 and 3 of RD1 are latched as DOUT for transmission on the Unibus, and the microsequencer jumps to the prefetch routine.

In the byte offset mode, the data retrieved is shifted down one byte location before being sent to the Unibus. For example, if UA02 and UA01 are both false (B0, A2, A1 = 4), bytes 1 and 2 are latched as DOUT, the first longword (RD1) is moved to locations 00, 10, 20, 30 of the BDP, and the second longword is moved to locations 01, 11, 21, and 32, as shown in Figure 3-17.

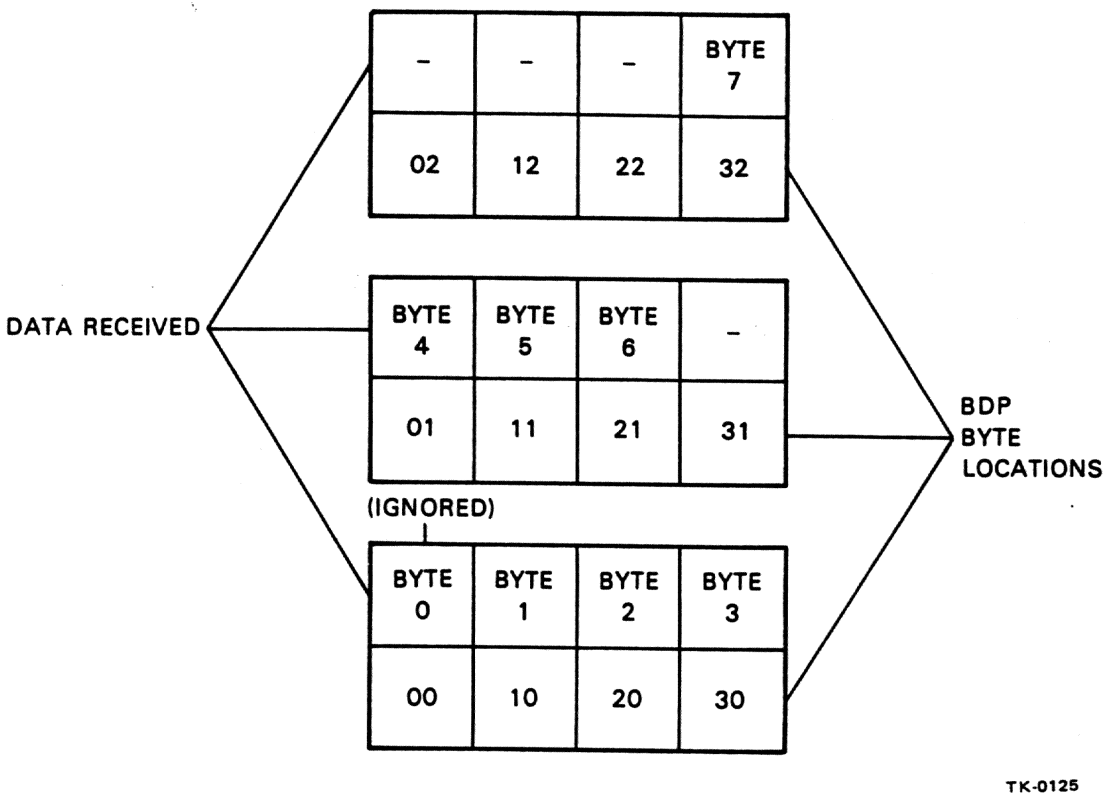


Figure 3-17 Use of BDP on DMA Read in Byte Offset Mode; B0, A2, A1 = 4

Notice that the data is shifted from the data file to the BDP with byte 7 displaced from location 31 to location 32. The function of this displacement is related to subsequent prefetch operations, as explained in Paragraph 3.3.1.3.3.

If address bit UA02 is true in the byte offset mode (B0, A2, A1 = 6), RD1 bytes 1 and 2 are latched as DOUT, RD1 is moved to BDP locations 01, 11, 21, 32, and the microsequencer jumps to the prefetch routine.

However, if address bits UA02 and UA01 are both true in the byte offset mode (B0, A2, A1 = 7), byte 3 of RD1 is latched as DOUT, and the 2ND READ subroutine (MF11) is invoked to retrieve the second data byte from SBI memory. The UBA, therefore, increments the Unibus address and performs an extended read transfer. Eight succeeding data bytes will be returned, the first of which is gated to the upper byte of the DOUT latches for completion of the DATI transfer.

3.3.1.3.2 Buffer Not Empty – When a Unibus device initiates a DMA read transfer that is mapped through a buffered data path containing data, no transfer on the SBI is required. The DMA attention select logic decodes the Unibus address and control bits and causes the microsequencer to perform one of the DMA C(0:15) routines listed on sheet MF1 of the prints.

For example, if data bytes 0 and 1 (word 0) were read on the previous DATI to SBI memory mapped through the BDP specified, the following six bytes from the quadword accessed will remain stored in the BDP. If the Unibus device is performing a block transfer, and the next two bytes (word 1) are addressed on a DATI, the microsequencer will jump from idle to the starting address for the DMC C1 routine (MF15). This routine moves word 1 (bytes 2, 3) to the DOUT latches for transfer on the Unibus, tests for parity errors, and returns to idle.

3.3.1.3.3 Prefetch Microroutine – The UBA performs a prefetch operation following the DATI transfer that accesses the last word of an SBI memory quadword. In this way the UBA anticipates and prepares for subsequent DATI transfers to adjacent consecutively increasing addresses. The last complete word remaining in the BDP will be bytes 6 and 7 normally and bytes 5 and 6 in the byte offset mode. Note that in the byte offset mode, byte 7 will be stored in BDP location 32. After the transfer of the last word to the Unibus, the prefetch operation begins. The UBA then increments the stored Unibus address, requests the SBI, and performs an extended read transfer on the SBI. The microsequencer returns to idle once the command transfer has been completed. When the read data returns, the microsequencer goes to the PF0 routine if the Unibus transfer using this BDP is not in the byte offset mode. If the Unibus transfer is in the byte offset mode, the microsequencer goes to PF1. If the byte offset bit is not set (microroutine PF0, MF16), the first four bytes retrieved are stored in RD1 of the data file, and then loaded in BDP locations 00, 01, 02, 03. The second four bytes are moved from RD2 to BDP locations 01, 11, 21, 31, as shown in Figure 3-18.

However, if the transfer is made in the byte offset mode (microroutine PF1, MF16), byte 7 from the preceding quadword, at this point, will still be stored in location 32 of the BDP. The microsequencer moves this data from location 32 to the temporary storage latches. It then gates RD2 to locations 01, 11, 21, and 32, as shown in Figure 3-19, steps 1 and 2.

Next the microsequencer moves the old data (byte 7 from the preceding quadword) from temporary storage to location 31, and then it moves RD1 from the data file to BDP locations 00, 10, 20, 30, as shown in Figure 3-20, steps 3 and 4. The data used to make up the Unibus word for the next DATI transfer within the block will be supplied from BDP locations 31 and 00 (see the flowchart for DMA C7, sheet MF15).

The data retrieved in the prefetch operation is not transferred to the Unibus, however, unless and until the Unibus device performing the block transfer performs another DATI to the next consecutive address.

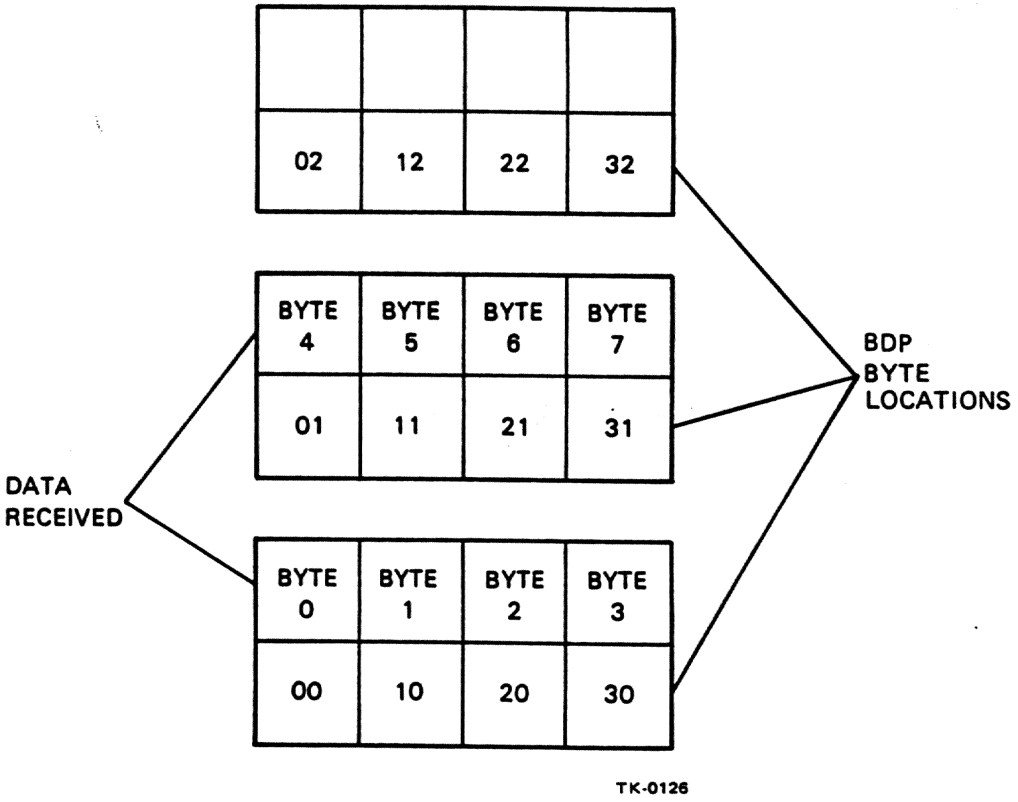


Figure 3-18 Use of BDP on a Prefetch Operation Without Byte Offset

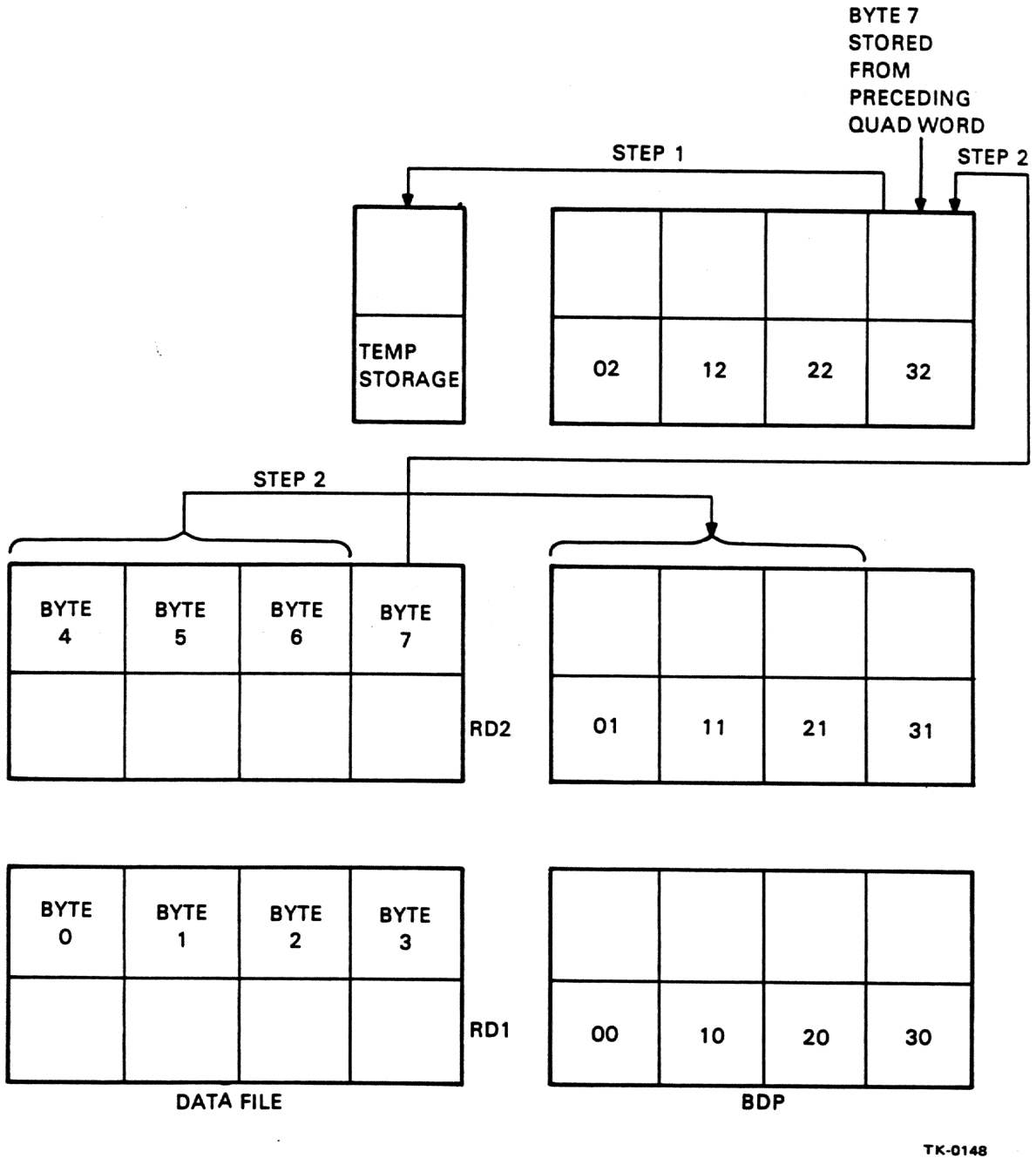


Figure 3-19 Use of BDP on a Prefetch Operation
With Byte Offset, Steps 1 and 2

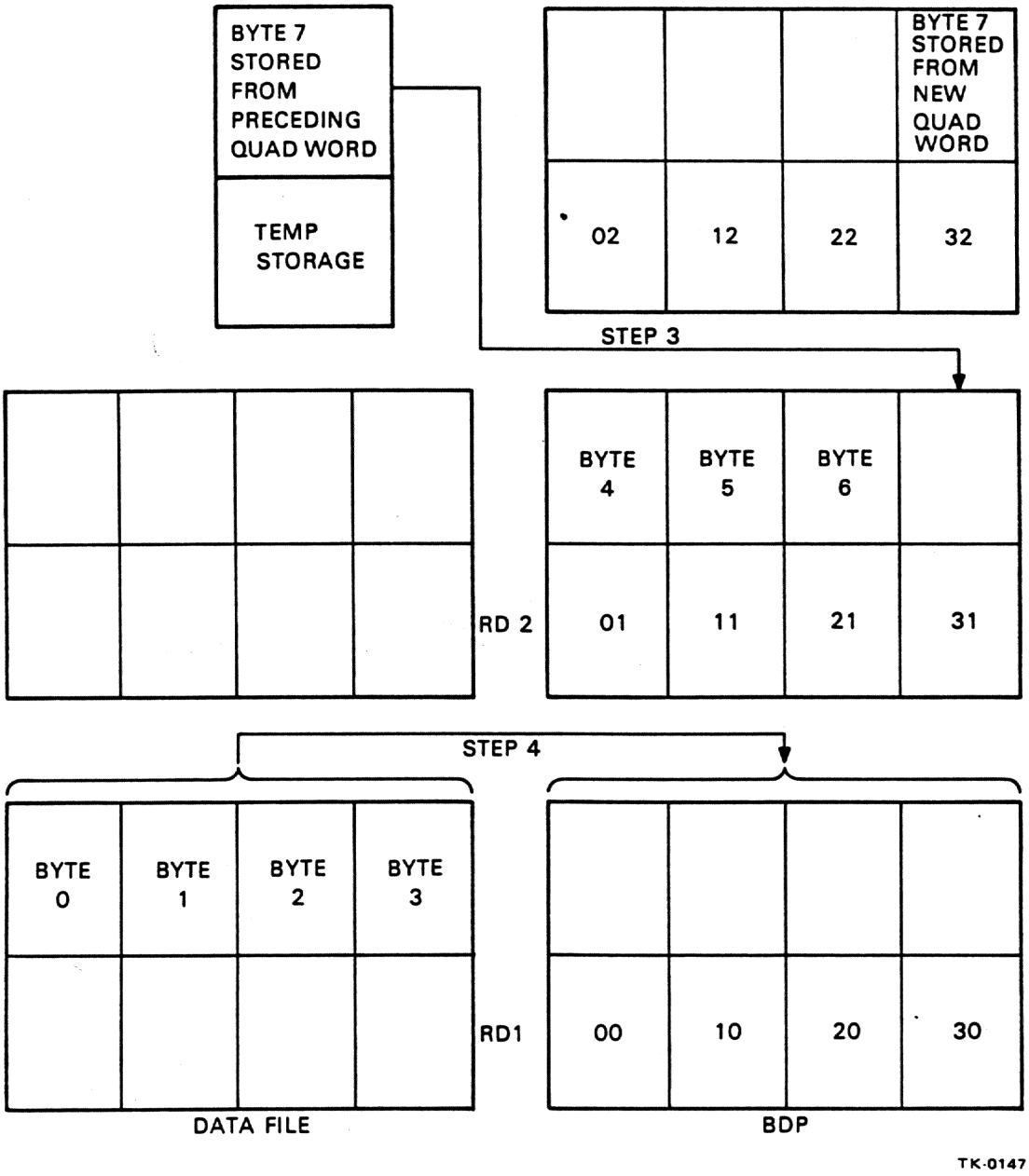


Figure 3-20 Use of BDP on a Prefetch Operation With Byte Offset, Steps 3 and 4

3.3.1.4 Microroutine for DMA Write Through a BDP – The DMA attention select logic selects one of four routines DMA A(0:3), when a Unibus device performs a DATO or DATOB transfer on the Unibus that is mapped through a BDP. For example, if the transfer is a DATO to SBI memory, the microsequencer will jump from the idle state to the starting address of the DMA A0 routine (MF3). The routine loads the Unibus address bits (17:02) into byte locations 23 and 33 for use on subsequent purge operations. Note that these two byte locations also form the lower half of the corresponding data path register. The microsequencer then branches depending on the states of the byte offset bit and address bits UA02 and UA01. With B0, A2, A1 = 0, the Unibus data is moved into locations 00 and 10, mask bits 0 and 1 are set, and the microsequencer returns to the idle state. It is important to realize that no transfer on the SBI will be initiated unless B0, A2, A1 = 3 (for a DATO operation) or 7 (for a DATOB operation). The data in the first six byte locations will not be transferred to SBI memory until the last one or two byte locations of the quadword are addressed on a DATO transfer, or until the software initiates a purge operation.

If the active Unibus device continues its block transfer, the UBA will initiate an extended write masked transfer on the SBI when the buffer is full (e.g., B0, A2, A1 = 3). The microsequencer forms the C/A word from the map register addressed and the function encoder logic, requests the SBI, and then asserts C/A, WD1, and WD2 sequentially on the SBI. Note that the signals that address the longword locations within the BDP RAM (UCBF LDP ADD 0, 1 H and UCBF HDP ADD 0, 1 H) shift when WD2 is selected for transmission, but that they are not manipulated by the LAD and HAD signals from the microsequencer, as might be expected. Instead, the microsequencer enables UCBF OP CD 0 H in the OPA register at the beginning of the BDP write routine, selecting the B inputs to the LAD/HAD multiplexer shown in Figure 3-21.

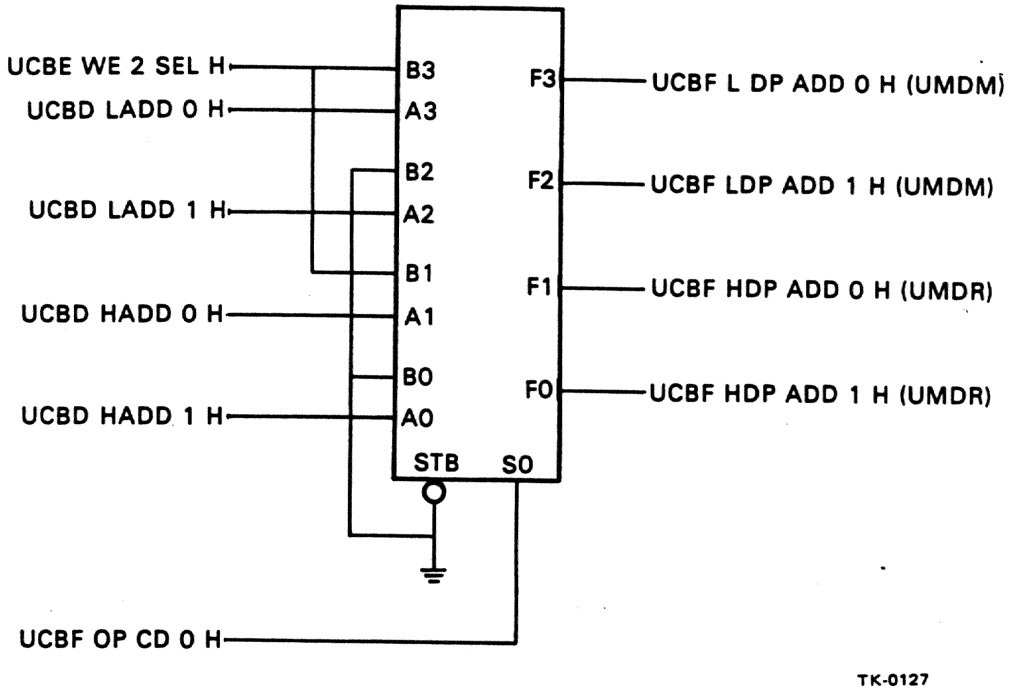


Figure 3-21 LAD/HAD Multiplexer Logic

When UCBE WD2 SEL H is false, the low longword in the BDP is accessed. Then, when WD1 is enabled on the SBI, UCBL DTEA H is also true, enabling UCBL XTND WR L. This, in turn, enables UCBI WD2 SEL H, selecting WD2, the second longword location in the BDP.

The microsequencer checks the confirmation code corresponding to each transmission on the SBI. A BSY or no response confirmation will cause the UBA to repeat the extended write transfer sequence. The microsequencer will abort the transfer and return to the idle state on receipt of an ERR confirmation or detection of a data path parity error. An ACK confirmation for each of the three transmissions on the SBI allows the microsequencer to complete the operation successfully and return to the idle state.

If the Unibus I/O device initiates a DATO transfer to SBI memory address space with address bits UA02 and UA01 true, and the map register addressed is set up for the byte offset mode (B0, A2, A1 = 7, sheet MF3), the microsequencer will branch within the DMA A0 routine to set mask bit 7. It will then move the low byte of the Unibus data word to location 31 of the BDP (as byte 6), move the high byte to the TMPSTR latches [UMDM TMPSTR (07:00) H], increment and store the 16 high order Unibus address bits in BDP locations 23 and 33, and load the OP A register for the BDP write transfer. The UBA performs this transfer in the same way that it performs the write transfer without byte offset, except that after ACK confirmation for the last WD is received and the BDP is empty, the microsequencer gates the contents of the TMPSTR latches (7:0) to byte location 00 of the BDP RAM for storage until the next write or purge transfer on the SBI, as shown in Figure 3-22.

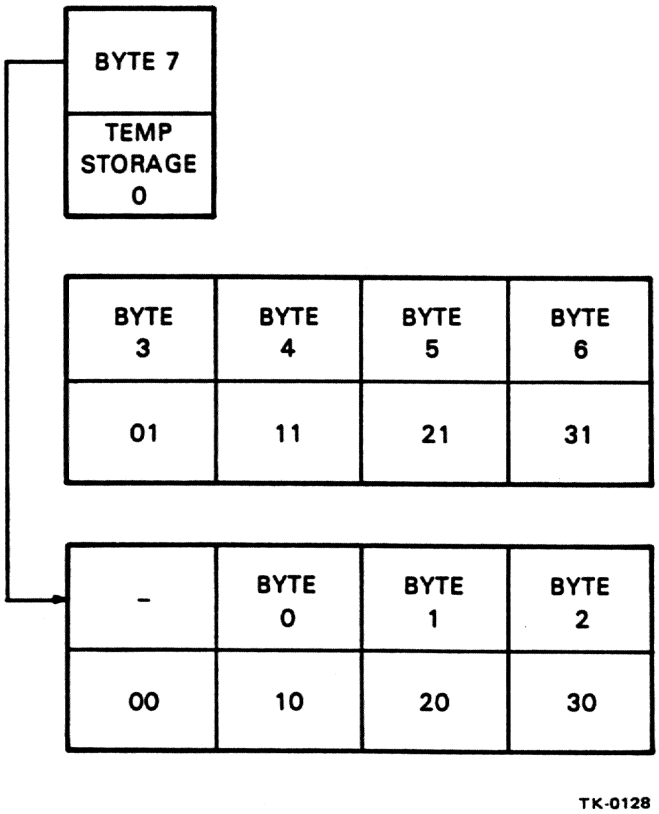


Figure 3-22 Use of BDP on DMA Write Transfer with Byte Offset; B0, A2, A1 = 7

Figure 3-23 shows the relative positions of the data bytes being transferred in a block between a Unibus device and SBI memory. The top of the figure shows the relative positions without byte offset. The bottom shows the positions with byte offset.

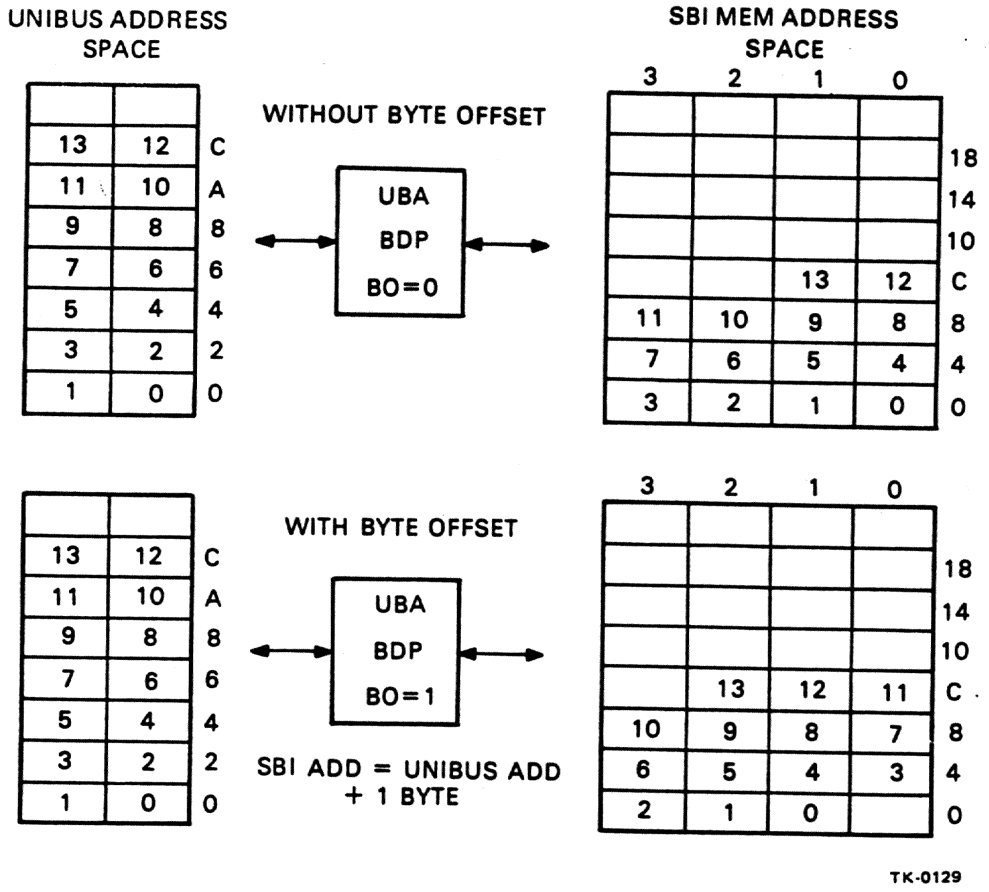


Figure 3-23 Relative Positions of Data in Unibus and SBI Address Space

3.3.2 Purge Microroutine

The software writes a 1 to the BNE bit of the data path register corresponding to the BDP just used at the end of every block transfer operation. When the address and function decoder logic decodes the C/A word, it enables USIL BRS + DPR WD XPCT L. USIF STRD A9 will be false and USIF STRD A4 will be true, enabling USIN MIC IR QA and QC H, which cause the UCBR SB ATT L flip-flop to set at T2 and enable UCBR SB SEL 2 and 0 H. The attention select logic then asserts 065 on the MPC lines, causing the microsequencer to jump from the idle state to the starting address of SB 5 (MF 24), the DPR write routine.

The microsequencer then tests the condition of data bit 31 (BNE, also called B state) received from the B field. If the bit is set, the microsequencer then tests the UCBJ PRG RQD H signal. This signal will be true if and only if DPR bit 29 (B FUNC) and UCBK MSK H are both true (on a write operation when the BDP contains data).

If PRG RQD is false (read operation) and the BDP in question is active (UCBJ BDP ACTIVE H is true meaning that read data from the SBI is expected from a prefetch), the microsequencer will loop, waiting for the read data, clear the read data when it comes in, clear DPR bits 31 and 29 (B STATE, B FUNC), clear the mask bits, and return to idle. The BDP is now in its initialized state and ready for a new transfer.

If PRG RQD and BDP ACTIVE are both false, the mask bits and DPR bits 31 and 29 are cleared, and the microsequencer returns to the idle state.

However, if PRG RQD is true (indicating that the BDP contains write data for transfer to the SBI) then the microsequencer initiates a purge write transfer operation. The Unibus address stored in locations 23 and 33 of the BDP (it will have been incremented before storage) is moved to the buffered stored Unibus address latches (UMDS) and then routed to the map register address lines. The microsequencer then forms the C/A word from the selected map register. If no map register or data path parity errors are detected, the UBA requests the SBI, and performs a write masked or an extended write masked transfer (as required) on the SBI as explained in Paragraph 3.3.1.4. Upon receipt of ACK for the last write data word, the mask and state bits are cleared. If no data path parity errors are detected at this stage, the microsequencer returns to the idle state.

3.3.3 Initialization Microroutine

When the UBA is powered up (PS DC LO true), a power failure in the SBI clock circuit is pending (DEAD true), or the software writes a 1 to the AD INIT bit of the control register, the UBA will assert UCBP AD CLR H forcing the UCBC MPC (7:0) H lines low, and select 000(8) or 400(8) as the address of the next control store ROM word. These two addresses constitute alternative starting locations for the initialization routine. However, the microsequencer will not start the routine until AD CLR is released and the next ROM location can be addressed.

At the beginning of the routine (AD CLR, sheet MF2) the initialization and timeout counter (sheet UCBN) is cleared. The OP B and OP A registers are loaded to disable the data inputs to the map registers, enable the output of the counter onto the map register address lines, and select the lowest four bits of the counter as the BDP RAM address inputs. The microsequencer then steps through the first of two loops, clearing the high and low portions of the first 16 map registers, the four longword locations of the first BDP RAM location (the four BRRRVRs and the four BRSVRs). When the low portion of the map register addressed has been cleared, the counter is incremented. The microsequencer then branches to the beginning of the first loop, to clear the next Map Register, BDP RAM location, and the associated DPR. When the output of the counter equals 16, the microsequencer branches out of the first loop and into the second loop, which clears the remaining 480(10) map registers. When UCBN CNTR FULL H is true, the microsequencer branches out of the loop to the idle state.

3.4 USE OF SBI FIELDS

The USI module of the UBA contains the SBI interface circuits. The logic that feeds the interface, however, is distributed over all of the UBA modules.

3.4.1 UBA Timing

The UBA receives six clock signals from the SBI, ensuring that operations on the adaptor are fully synchronized with other subsystems using the SBI. Figure 3-24 is a block diagram showing how the four time states (T0, T1, T2, and T3) are derived from the SBI signals.

Figure 3-25 is a timing diagram showing the relation of the SBI clock signals to the signals derived from them. Note the reference to the timing on the bus transceivers at the bottom of the figure.

Since the derived clock signals are distributed throughout the UBA, the signals are buffered and the signal names change to reflect their functions.

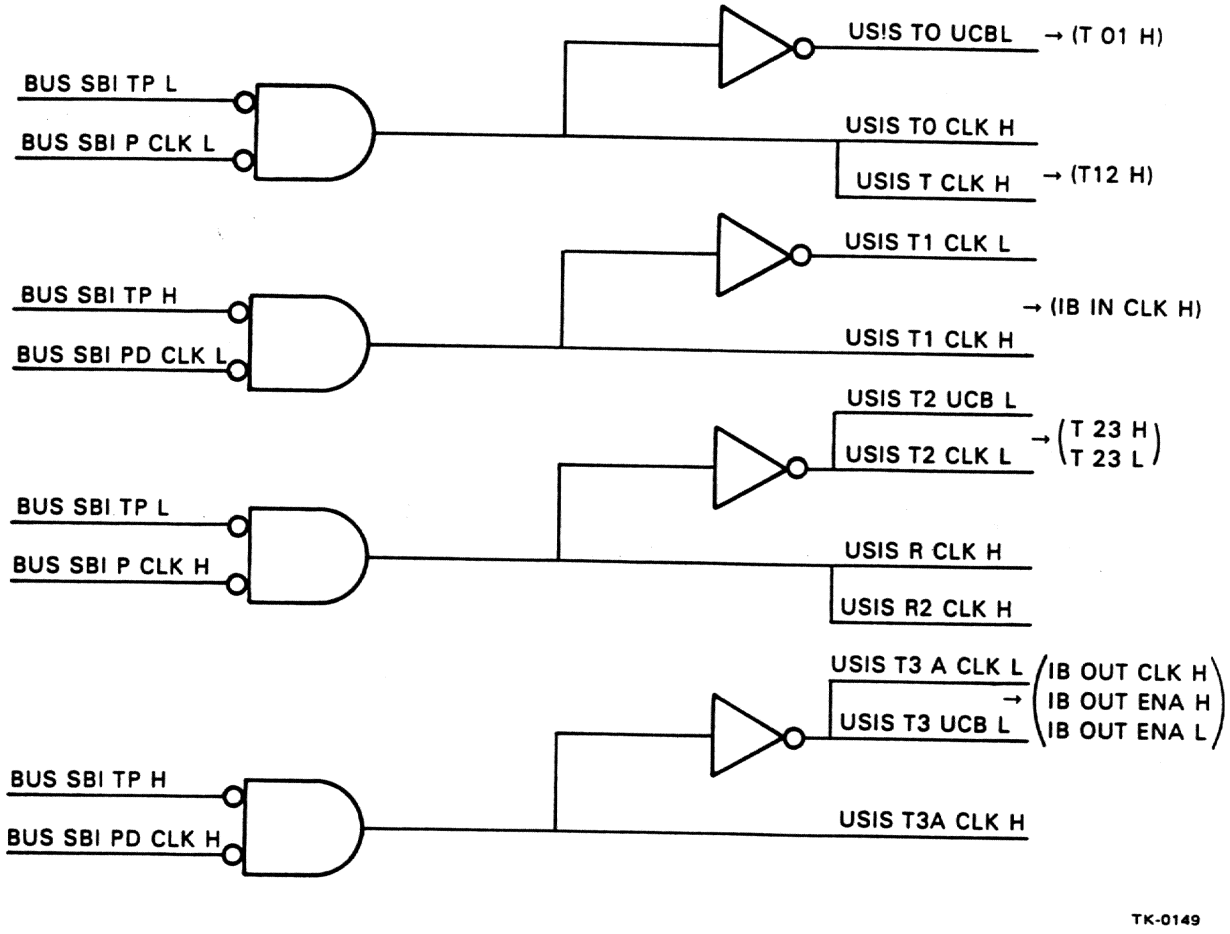


Figure 3-24 UBA Timing Signals

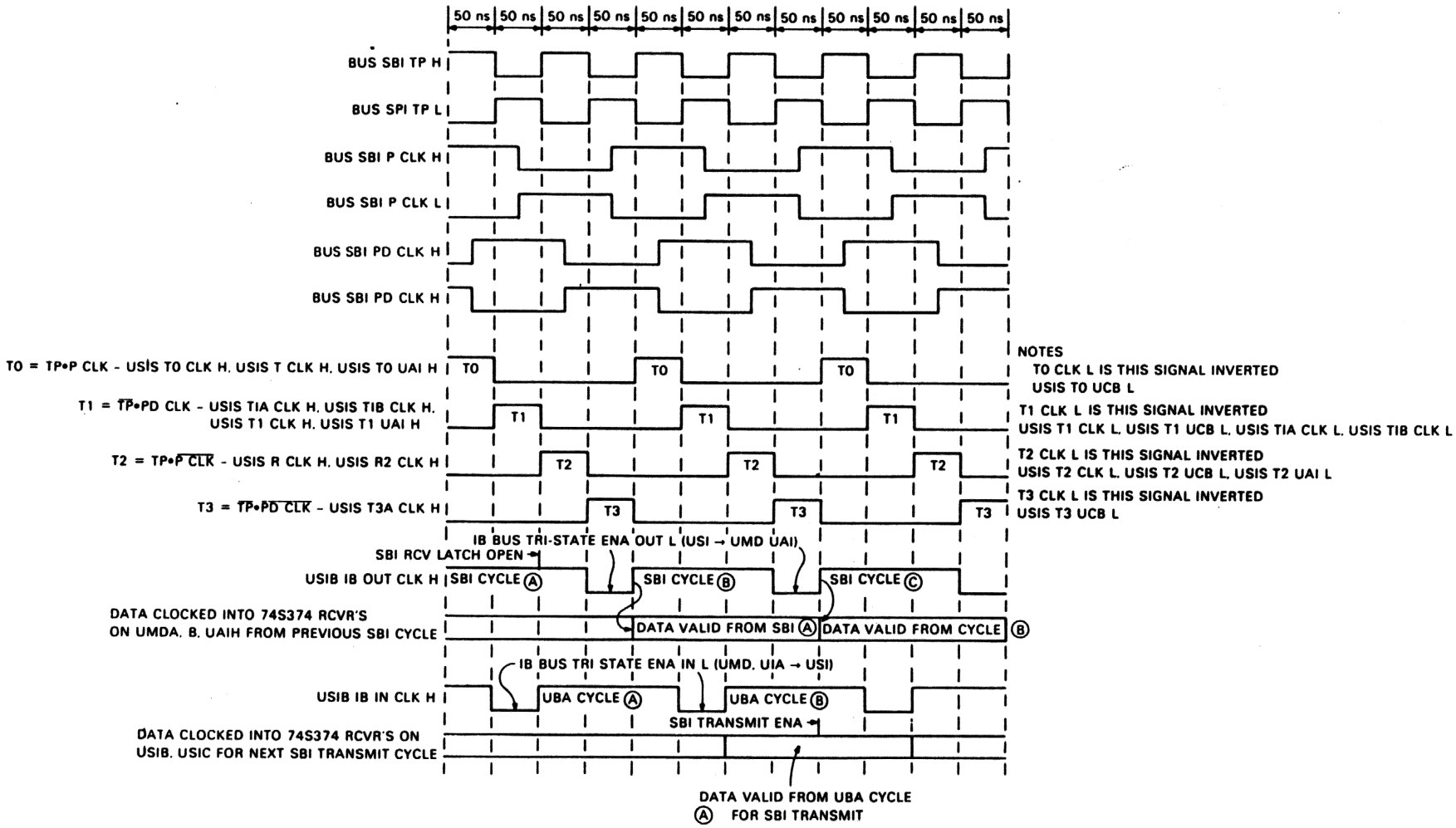


Figure 3-25 UBA Timing Diagram

3.4.2 Data Transfer Timing on the SBI

The UBA latches the information in each of the SBI fields with every SBI cycle. It then takes further action in accordance with the current state of the UBA and the nature of the information latched, as shown in the flowchart in Figure 3-26. If the parity of the fields latched is correct, the tag decoder logic will enable one of four circuits: C/A decoder logic, interrupt summary read/response logic, read data logic, or write data logic.

The following is an outline of the action of the UBA during three consecutive SBI cycles when the receive SBI logic is in the idle state checking parity on the SBI.

First SBI Cycle – the SBI transceiver latches on the UBA open with the assertion of USIS R CLK H and USIS R2 CLK H at time T2 in the SBI cycle. The output of the latches remains indeterminate for the duration of T2. At T3, R CLK H and R2 CLK H are low and the data is latched.

The UBA parity checker logic checks for even parity on the SBI fields on every SBI cycle during T3, whether or not the data is intended for the UBA.

Second SBI Cycle – USIA P OK L will be valid after T0 of the following cycle if the SBI parity is, in fact, OK. If P OK L is false, it will be latched high at T1, causing the assertion of USIE T FLT H. Note that the fault signal will not be asserted on the SBI until the following cycle. The bus transceiver latches open during T2, again, and the old data latched on the previous cycle is lost if the UBA has not been identified as the receiver.

Third SBI Cycle – The UBA transmits FAULT on the SBI at T0 if a parity error has set the Parity Fault flip-flop (USIE). The data on the SBI is then sampled and latched at T3, as in the previous cycles. The appropriate confirmation (ACK, BSY, NR, or ERR) is asserted on the SBI during this cycle for the information received during the first cycle. The confirmation and action taken by the UBA can be determined from the flowchart shown in Figure 3-26.

3.4.3 B Field Logic

The bus transceivers for the SBI B field open and close with the assertion and negation of T2. The latched data is then buffered and asserted on the internal bus [BUS IB (31:00) H] when T3 is true. The IB bus is the main data path between modules of the UBA. Immediately following this, while T3 is still true, the data is latched on the UMD module for transmission to the map and data path logic and on the UAI module for use as write data or an address to be transmitted to the Unibus, as shown in Figure 3-27.

Figure 3-28 shows the relation of the B field logic to the UBA as a whole.

Six other data sources feed the internal bus: the map registers, the data path logic, the status register, the control register, the failed map entry register, and the failed unibus address register. The data from these circuits will be asserted on the internal bus with T1 if enabled by the OP A register. When the UBA arbitrates for control of the SBI B field to transmit data from one of these six sources, the microsequencer will enable the appropriate signal in the OP A register to route the data from the desired source to the IB latches shown on sheet USIB of the prints. These latches, together with the interrupt summary response logic, the configuration register, and diagnostic control register, form the inputs to the T B bus [BUS T B (31:00) H] on the USI module. Data from one of these three inputs will always be enabled on the T B bus with the assertion of T1. Data from the IB latches will be asserted on the T B bus by default, if USIJ ISR DTE L and USIJ LOCAL READ L are both false.

When the UBA has successfully arbitrated for control of the SBI B field, the USIJ DATA DTE L signal will be true, and the data on the T B bus will be asserted on the SBI with the enabling of USIS T CLK H at T0.

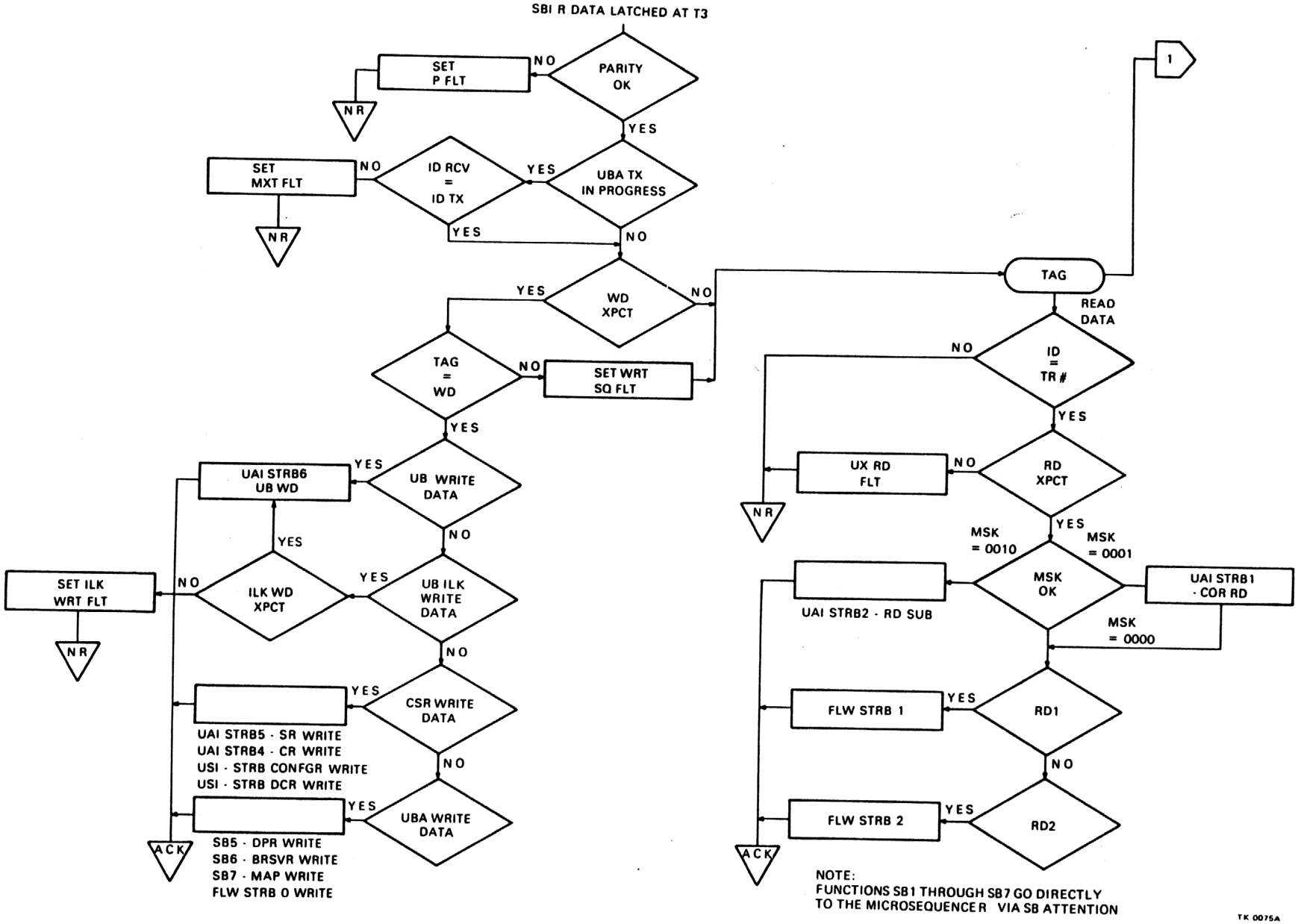


Figure 3-26 Receipt of Information from the SBI (Sheet 1 of 2)

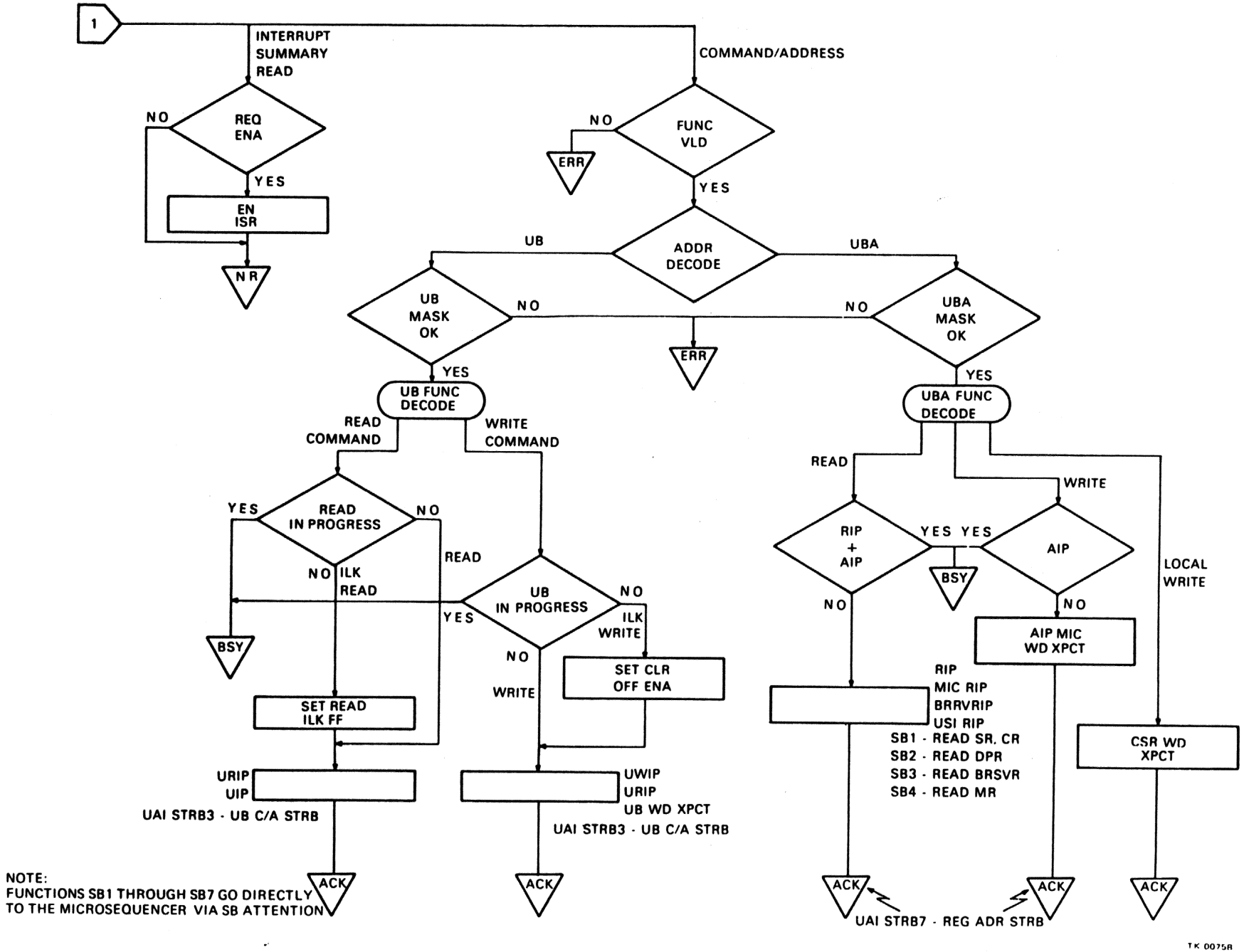


Figure 3-26 Receipt of Information from the SBI (Sheet 2 of 2)

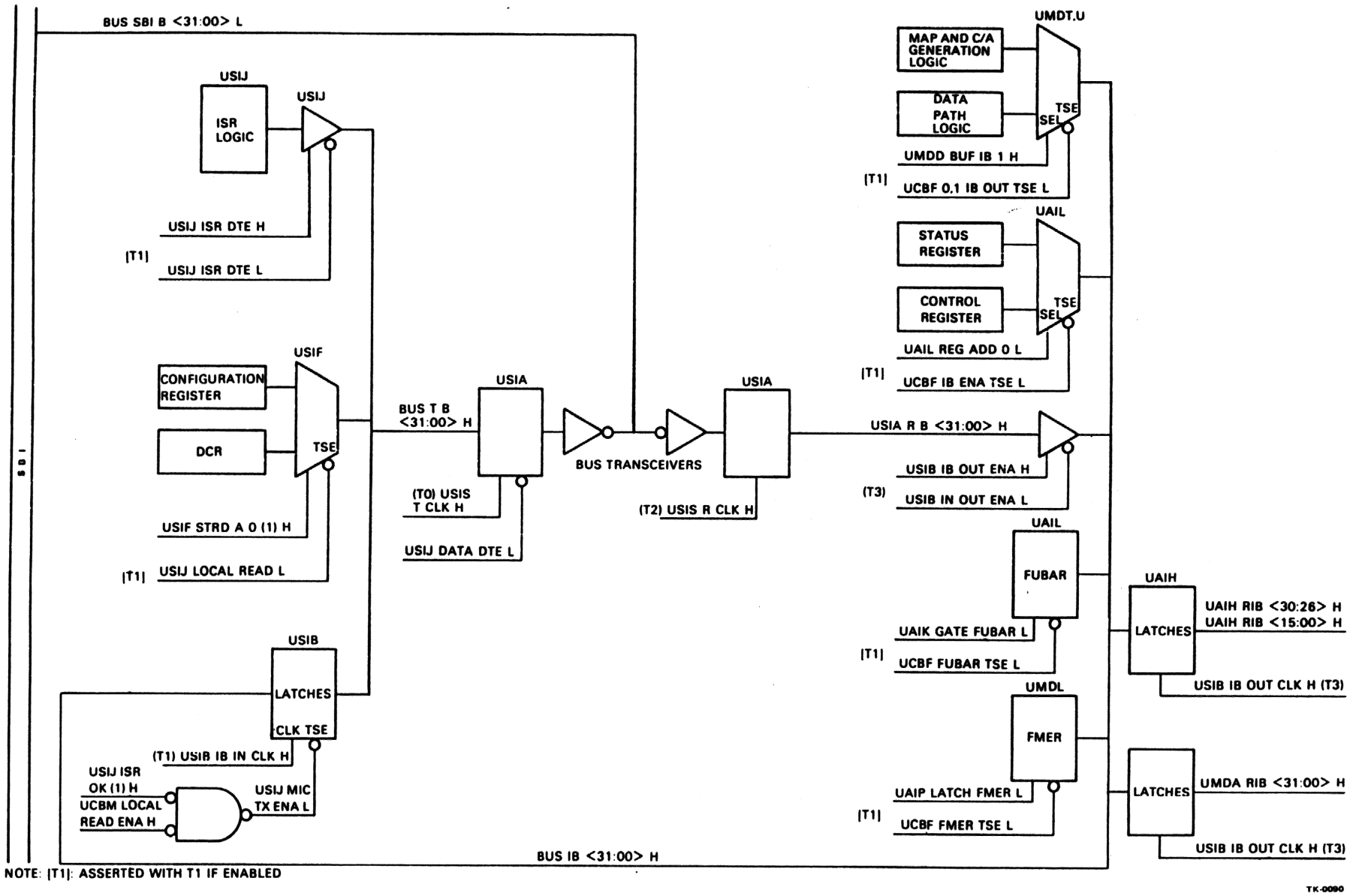


Figure 3-27 B Field Logic, Block Diagram

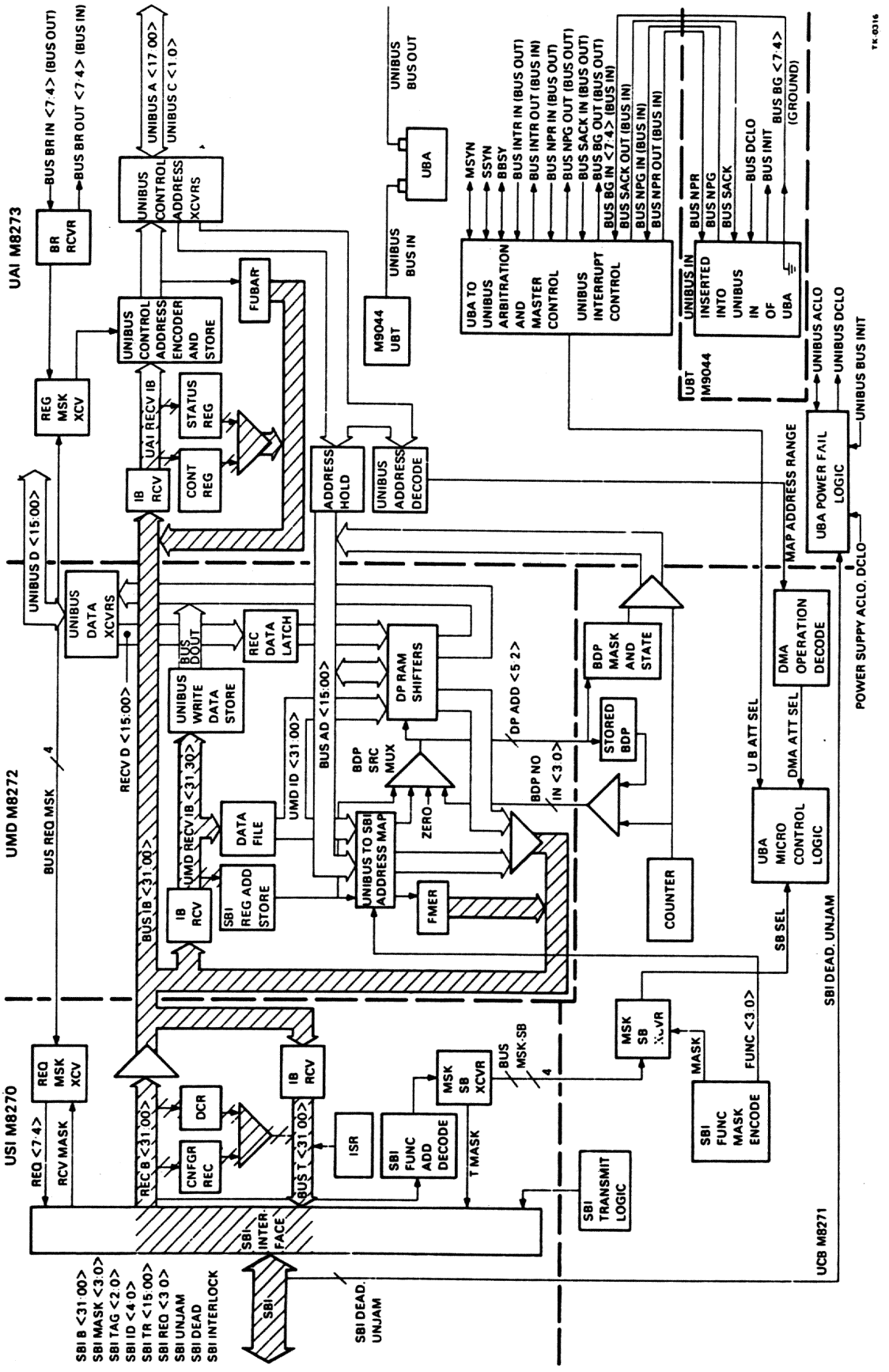


Figure 3-28 B Field Logic/UBA, Block Diagram

3.4.4 Arbitration and Data Transmit Functions

When the UBA must transmit data on the SBI, it sets up the data for transmission in one of the circuits feeding the internal bus or the T B bus, as previously explained above.

The data to be sent may be C/A, read data requested by the CPU, write data, or the UBA interrupt summary response. If the data is the interrupt summary response, it will be transmitted two cycles following the receipt of the interrupt summary read command. The UBA does not arbitrate for the SBI, because the VAX-11/780 CPU holds the bus for two cycles with TR00, while waiting for the interrupt summary response. If the data to be sent is a C/A word or read data, microsequencer attention is called. The microroutine selected then sets the SBI request flip-flop. UCBL REQ FF (1) L, in turn, causes the arbitrator circuit to send the TR signal assigned to the UBA, BUS TR L, with T0. At T3 of the same cycle the arbitrator circuit samples the 16 TR lines and compares the highest level asserted with the assigned TR code hardwired to USIC TR SEL (A-D) L on the backplane (see sheet USIC of the prints).

If no higher priority TR signal has been asserted on the SBI, the arbitrator circuit will enable USIC ARB OK L. This signal, in turn, enables USIJ DATA DTE L, as shown in Figure 3-29. Note that if the UBA issues a write masked or an extended write masked command, it will assert UCBM SEND HOLD H, causing the arbitrator circuit to assert BUS SBI TR O L at the same time, in order to hold the bus for one or two succeeding cycles.

3.4.5 ID Field Logic

The four signals USIC TR SEL (A-D) L produce the ID number of the UBA in 2's complement form, as well as the TR number used in the arbitration circuit. The output of the adder shown in Figure 3-30 is multiplexed with the five signals USIH STD ID (4:0) H.

When the UBA asserts C/A or write data on the SBI, the ID signals for the UBA are selected and asserted on the SBI. When the Unibus asserts read data or an interrupt summary response on the SBI, the stored ID signals, which identify the read commander nexus, are selected and asserted with the data on the SBI. Note that tag bit 0 is used to differentiate between the two ID functions.

3.4.6 Mask Logic

The mask logic is used in conjunction with the function logic on all transfers involving the UBA. Whenever the UBA transmits data on the SBI (read data, C/A, write data, or ISR), it asserts a four bit mask. Figure 3-31 shows the mask and interrupt request logic in block diagram form (see also sheet UCBK). Note that the four SBI request signals and the four mask signals use the same internal bus [BUS REQ MSK (3:0) H]. The signals share the bus for the sake of efficiency. The mask and request functions do not overlap. The nine sources of the mask bits that the UBA asserts on the SBI are developed as follows.

3.4.6.1 DMA Read Masked Transfer Mask (BDP) - On a DMA read masked transfer to the SBI mapped through a buffered data path, the microsequencer will set OP code bits 1 and 0 high, as shown in Table 3-7.

The function bits, UCBK FUNC (03:01) H, will be false (Table 3-11). The B inputs to the 4:1 mask multiplexer will be enabled (Figure 3-31 and sheet UCBK of the print set). A mask field of all ones will thus be placed on the BUS MIC IR (3:0) H lines to be transmitted with the C/A word. The read data will be four bytes.

3.4.6.2 DMA Extended Read Transfer Mask (BDP) - When the UBA performs an extended read transfer on the SBI in response to a DMA read command from the Unibus, the microsequencer will assert OP code bits 1 and 0; and function bit UCBK FUNC 01 H will be false, as on the read masked transfer, causing the 4:1 mask multiplexer to select the B inputs. But the signal which feeds the B inputs, UCBK FUNC 03 L, will be true, and the mask field formed will be all zeros. This mask will be ignored, and eight bytes will be returned as read data.

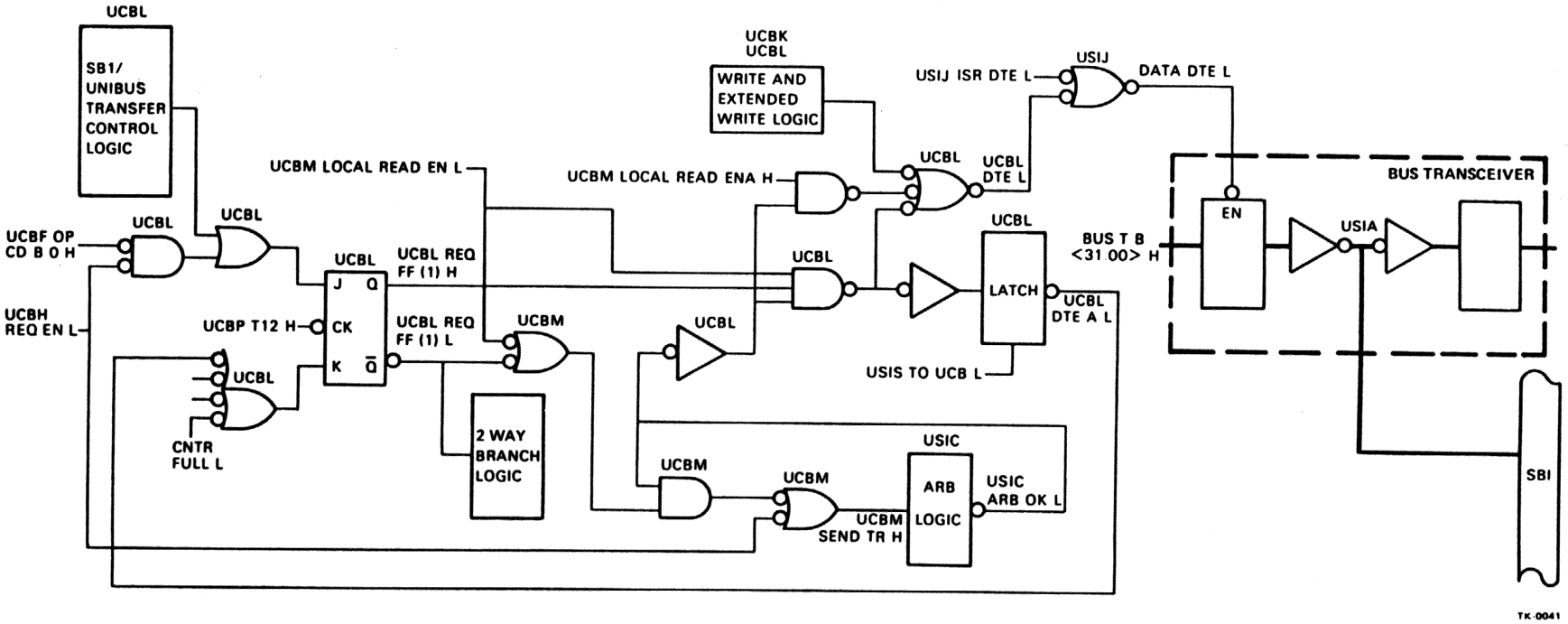


Figure 3-29 SBI Request Logic

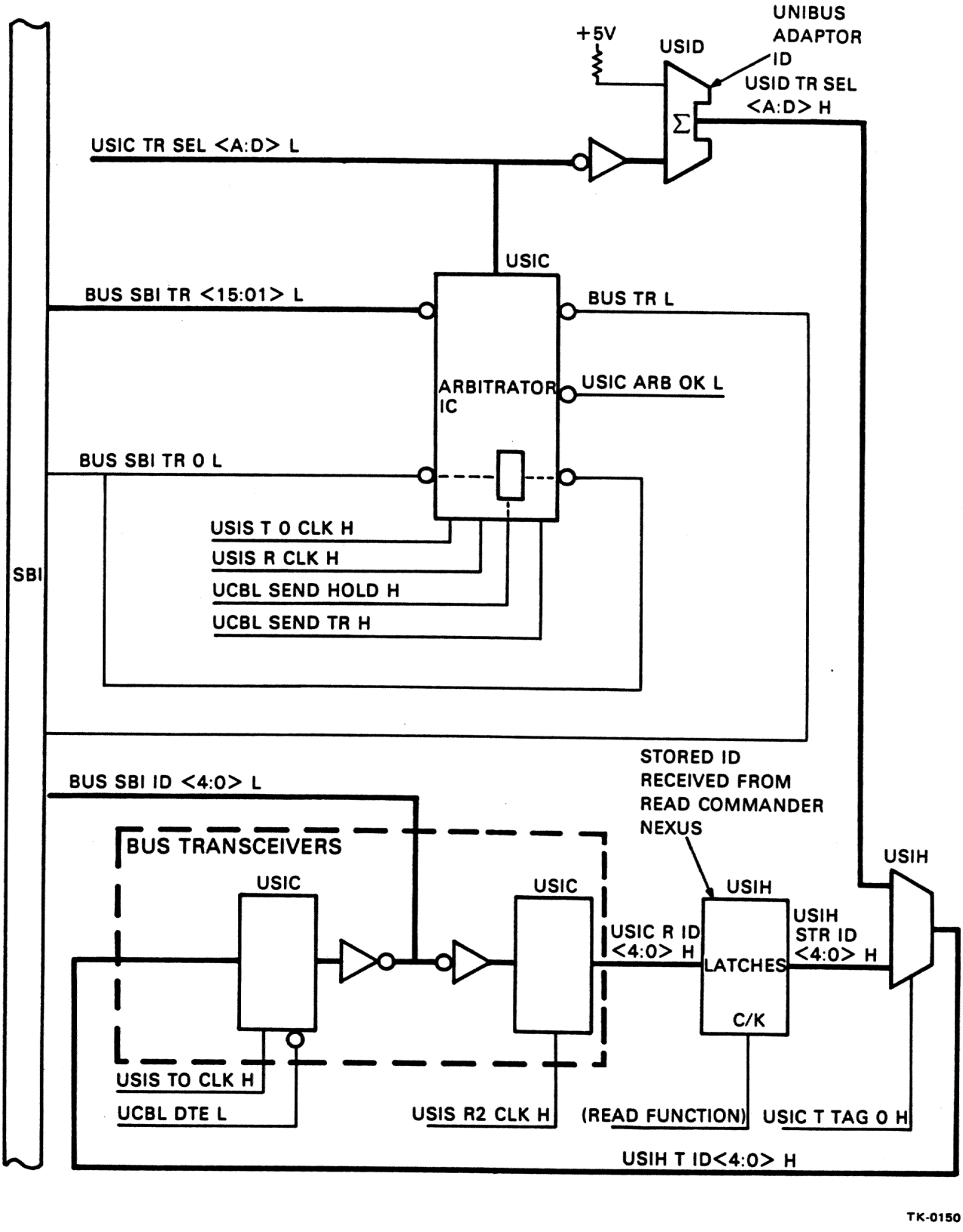


Figure 3-30 Arbitration and ID Logic, Block Diagram

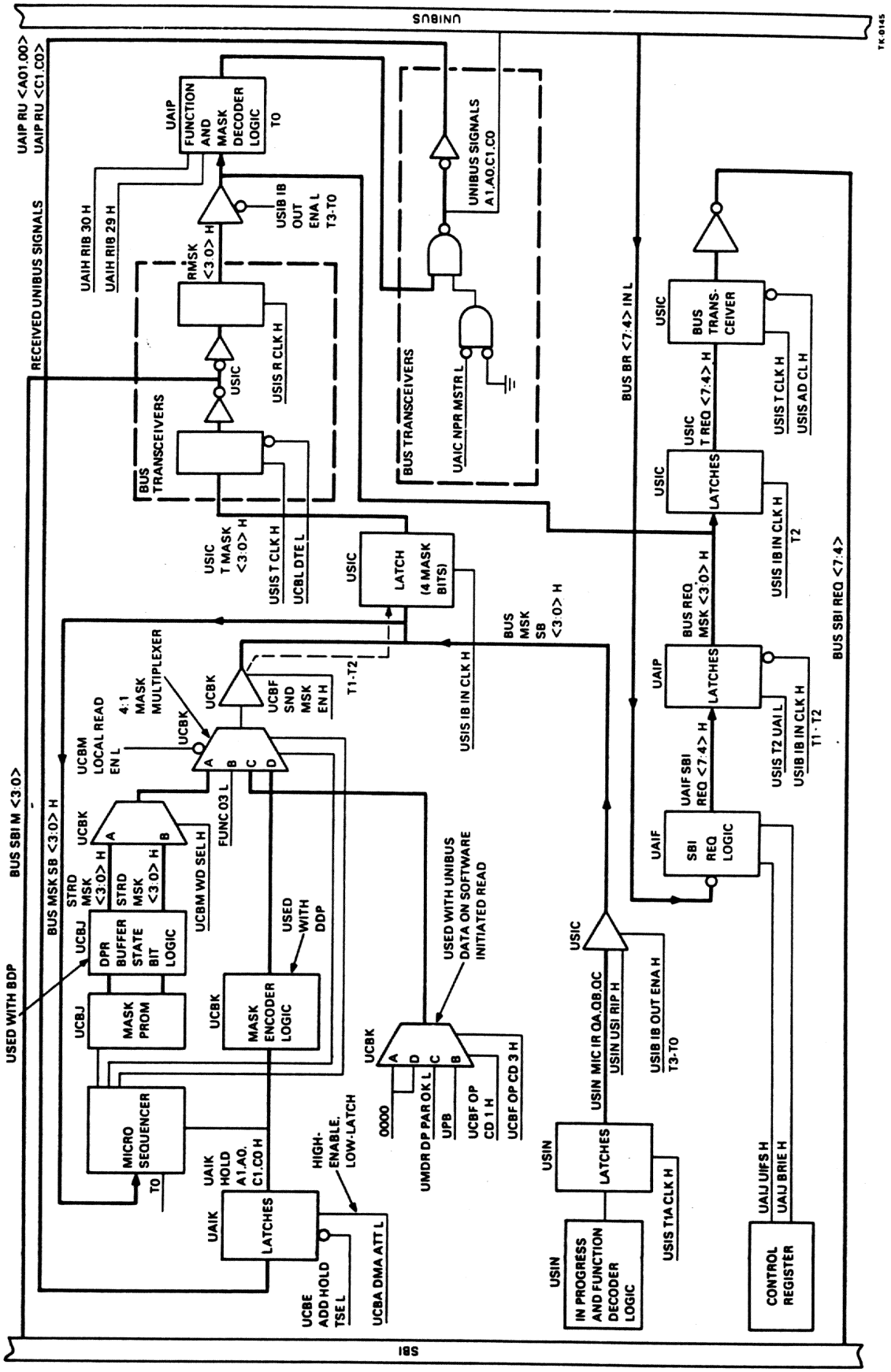


Figure 3-31 Mask and Request Logic, Block Diagram

3.4.6.3 DMA Write Transfer Mask (BDP) - When a Unibus device initiates a DMA write transfer that is mapped through a buffered data path, the UBA stores the bytes to be written in the BDP. Each time the Unibus device performs another write transfer, the microsequencer addresses the appropriate location in the mask PROM with OP field bits 7, 6, 5, 4, and 2. The mask PROM outputs correspond to the data bytes asserted on the Unibus [as defined by Unibus address bits (2:0)]. They are inclusive ORed with buffer state bits from the associated data path register, which have been stored on previous DATO sequences to the same buffered data path, as shown in Figure 3-32.

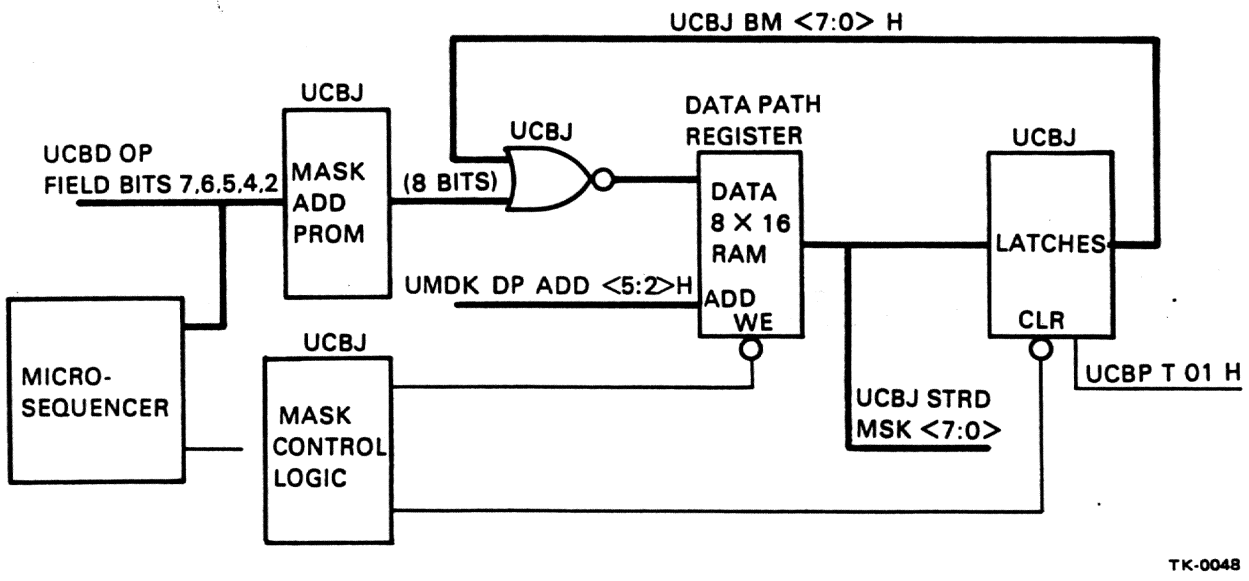


Figure 3-32 Mask PROM and Mask Bit Storage

In this way as many as eight buffer state bits for each buffered data path can be stored. Each buffer state bit corresponds to one of the eight data byte locations in the BDP. When the Unibus device performing the write transfers fills up the highest byte locations in the BDP, the UBA initiates a write masked or an extended write masked transfer on the SBI. The microsequencer asserts OP code bits 4, 1, and 0, and the function encoder enables bit UCBK FUNC 01 H (note that function bits 0 and 3 will also be true on an extended write masked sequence). This combination causes the 4:1 mask multiplexer to select the A inputs, which are fed from the four outputs of a 2:1 multiplexer, as shown in Figure 3-31. Before the UBA sends C/A, the signal UCBM WD SEL H is false causing the 2:1 multiplexer to select the A inputs, UCBJ STRD MSK (3:0) H. These stored mask bits (from the buffer state bits) then form the mask field to be transmitted with the C/A word. This mask field defines the first longword of the data (WD1) to be sent on the cycle immediately following the command/address. When the UBA performs a write sequence, it asserts UCBL SEND HOLD H which, in turn, enables BUS SBI TR 00 L on the SBI. SEND HOLD H also enables UCBM WD SEL H. This signal, in turn, selects the B inputs to the (2:1) stored mask multiplier, and the upper four mask bits form the mask, describing the second longword (WD2). This mask, however, is sent with the first longword, in accordance with the SBI protocol.

3.4.6.4 DMA Read or Write Transfer Mask (DDP) – When a Unibus device initiates a DMA read or write sequence which is mapped through the direct data path, the microsequencer asserts OP code bits 2 and 0, as shown in Table 3-10. This OP code combination causes the 4:1 mask multiplexer to select the D inputs, which are derived from Unibus address bits UA(01:00) and control bits C(1:0) (sheet UCBK). The mask field generated is sent with the C/A word, whether the function to be performed is a read or a write transfer. Note that only two mask bits can be asserted when the direct data path is used, and that only one mask bit will be asserted if the Unibus device is making a DATOB transfer.

3.4.6.5 Read Data Mask: CNFGR and DCR – When the software initiates a read transfer to either the configuration register or the diagnostic control register, the function decoder asserts UCBM LOCAL READ ENA H. This signal disables the 4:1 mask multiplexer, causing the UBA to send a mask field of all zeros with the read data.

3.4.6.6 Read Data Mask: BRRVR – When the UBA interrupt service routine initiates a read transfer to a BRRVR, the microsequencer enables OP code bits 4 and 2, selecting the C inputs to the 4:1 mask multiplexer. OP code bits 1 and 3 will be false, selecting the grounded inputs to another 4:1 multiplexer which feeds the C inputs of the multiplexer feeding the BUS MSK SB lines. The mask field will thus be all zeros.

3.4.6.7 Read Data Mask: Unibus Data – When the software addresses a read command to a Unibus address, the UBA retrieves the data on the Unibus and then the microsequencer enables OP code bits 2 and 1. The C inputs to the 4:1 mask multiplexer are thus selected, and a mask field of 0000 or 0010 is sent, depending on the state of the Unibus parity bit, BUS UPB L.

3.4.6.8 Prefetch Operation Mask – When a buffered data path is emptied as the UBA responds to a DATI transfer to SBI memory with Unibus address bits UA02 and UA01 high, the microsequencer initiates a prefetch routine following completion of the transfer. The microsequencer first enables OP code bits 3 and 1. The function code for the extended read transfer, 1000, in conjunction with the OP code causes the 4:1 mask multiplexer to select the B inputs. UCBK FUNC 03 L, feeding the B inputs, will be true, yielding a mask field of all zeros to be transmitted with C/A. The mask field will be ignored and eight data bytes will be retrieved as read data.

3.4.6.9 Purge Mask – When the software initiates a purge routine on the UBA, to clear a buffered data path, the software will clear the buffer state bits if the BDP contains read data. However, if the BDP contains write data, the UBA will execute a write transfer to the SBI. (If one or more of the upper four buffer state bits is high, the UBA will perform an extended write masked transfer.) The microsequencer enables OP code bits 4, 3, and 0, and these signals, in combination with UCBK FUNC 01 H true, select the A inputs to the 4:1 mask multiplexer. The buffer state bits [UCBJ STRD MSK (7:0) H] thus form the mask field for WD1 and WD2 in the case of an extended write masked transfer.

3.4.6.10 Mask Field Transmission Timing – The signal UCBF SND MSK EN H goes high from T1 to T2, on every SBI cycle. The output of the 4:1 mask multiplexer is thus enabled on BUS MIC IR (3:0) H. USIB IB IN CLK H is also true at T2 and the mask signals to be transmitted are latched as USIC T MASK (3:0) H. These signals are then transmitted on the SBI on a subsequent cycle, with the assertion of USIS T CLK H and UCBL DTE L.

3.4.6.11 VAX-11 CPU Initiated Transfer: Mask Translation – In all of the above cases, the mask bits generated are latched and then asserted on the SBI by the bus transceivers with C/A or read or write data as appropriate.

In contrast, the source of the UBA generated Unibus signals UA00, UA01, C0, and C1 is always the same. The SBI bus transceivers on the UBA latch the four incoming mask bits with C/A and data asserted by an SBI nexus on a CPU initiated transfer. The UBA then decodes the mask bits, forming Unibus signals UA00, UA01, C0, and C1, and asserts them on the Unibus together with the address bits UA (17:02), and, on a write transfer, write data, as shown in Figure 3-31.

3.4.7 Function Field Logic (UCB Module)

3.4.7.1 Function Encoder Logic – When the UBA transmits a C/A word on the SBI, the function encoder logic produces the four function bits. The upper three function bits can be associated readily with specific transfer types, as follows:

- Bit 3 – true for an extended transfer.
- Bit 2 – true for an interlock transfer.
- Bit 1 – true for a write transfer.

Figure 3-33 shows the function encoder logic.

The microsequencer will always be actively engaged in the transfer of data mapped through the direct data path or one of the buffered data paths when the UBA forms and transmits the C/A word. The 2:1 multiplexer shown in Figure 3-33 forms the upper three function bits.

The microsequencer signal UCBD FLR SEL 0 H selects the A inputs when the transfer is mapped through a BDP. The B inputs are used in conjunction with the DDP. An explanation of the development of each function bit follows.

3.4.7.1.1 Function Bit 3 (Extended Transfer) – On a DMA write transfer mapped through a BDP, function bit 3 is formed by the ANDing of UCBK MSK HI H (true when one or more of the upper four stored mask bits is true) with UCBD LADD 0 H, from the microsequencer.

On a DMA read transfer mapped through a BDP, function bit 3 will be enabled if UMDK SBI PG 27 H (from the map register selected), BUS AD 00 H (from Unibus Address bit A2), and UCBD LADD 0 H (from the microsequencer) are all false, and UCBD MAS 1 H (from the microsequencer) is true.

When the UBA performs a prefetch operation, function bit 3 is derived entirely from microsequencer signals. Function bit 3 will be true if UCBD FLR SEL 1 H is true and UCBD LADD 0 H is false.

When a DMA transfer is mapped through the direct data path, the grounded B input to the 2:1 multiplexer is selected, ensuring that function bit 3 is always false and that the UBA will not initiate an extended transfer.

3.4.7.1.2 Function Bit 2 (Interlock Transfer) – When a DMA transfer is mapped through a buffered data path, the grounded A input to the 2:1 multiplexer ensures that function bit 2 remains false.

On a DMA read transfer through the DDP, function bit 2 will be set if the transmitting Unibus device has issued a DATIP command, making UAIK HOLD C0 H true and UAIK HOLD C1 H false, thereby putting a high level on the B input of the 2:1 multiplexer. UCBK FUNC 02 H is then latched true, and it is ANDed with UCBH INTLK EN H, true, and UCBK FUNC 01 H, false, to set the Interlock flip-flop when UCBP T23 L is enabled with T2. The Interlock flip-flop will remain set until the interlock process has been completed.

The active Unibus device will follow the completion of the DATIP interlock read masked function with a DATO or DATOB transfer to the same location. When the UAIK HOLD C1 H signal is enabled, on the write transfer mapped through the direct data path, it is ANDed with the Q output of the Interlock flip-flop (high following the DATIP). The B input of the 2:1 multiplexer is selected, and true, yielding a true function bit 2.

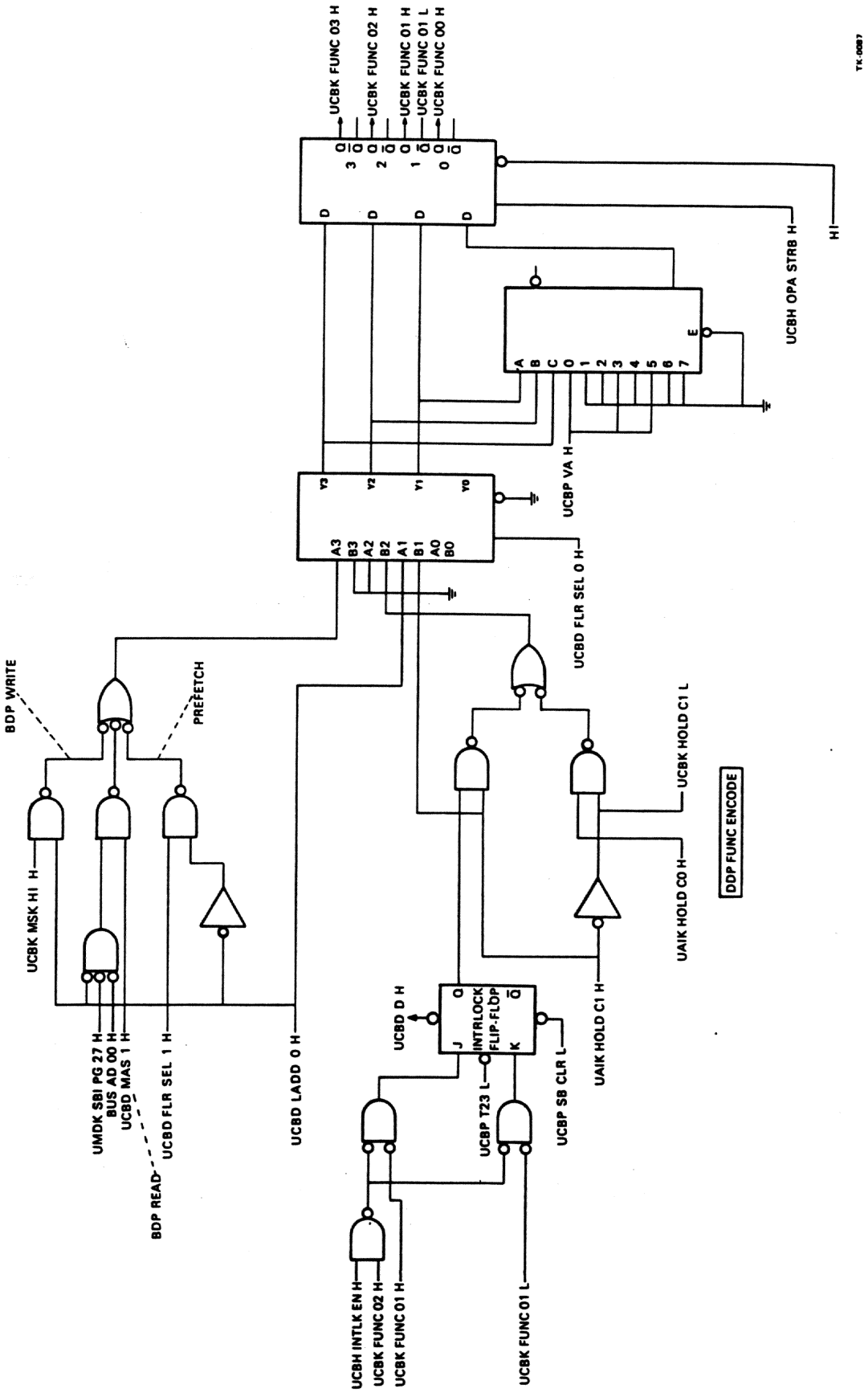


Figure 3-33 Function Encoder Logic

3.4.7.1.3 Function Bit 1 (Read/Write Transfer) – When the UBA maps a read transfer through a BDP, the state of UCBD LADD 0 H, fed through the A input of the 2:1 multiplexer, controls the state of function bit 1.

If a Unibus device performs a DMA transfer to be mapped through the direct data path, the state of UAIK HOLD C1 H determines the state of function bit 1. If the transfer is a DATO/B and the Interlock flip-flop is set, UCBK FUNC 01 L (true) will be ANDed with UCBH INTLK EN H (true) and UCBK FUNC 02 H (true) to reset the Interlock flip-flop.

3.4.7.1.4 Function Bit 0 – Function bit 0 is derived from the upper three function bits. The three outputs of the 2:1 multiplexer form the select inputs to the data selector circuit shown in Figure 3-33. The output of the data selector then forms function bit 0.

3.4.7.2 Function Decoder Logic – The function decoder logic on the USI module becomes active whenever the UBA receives a C/A word addressing Unibus or UBA address space. The address decoder logic determines whether a Unibus device or an internal register is addressed, as shown in Figure 3-26.

An 8:1 data selector, shown on sheet USIK of the prints, determines whether or not the function is valid. Only the four function codes listed in Table 3-11 are valid when the UBA or Unibus address space is addressed.

Table 3-11 Functions Valid When UBA Space Is Addressed

Function	Code			
	F3	F2	F1	F0
Read Masked	0	0	0	1
Write Masked	0	0	1	0
Interlock Read Masked	0	1	0	0
Interlock Write Masked	0	1	1	1

Notice that extended transfer functions are not valid.

3.4.7.2.1 Unibus Function Decoder Logic – USIK FUNC VLD H, USIA P OK H, and USIK UB OK H must all be true to qualify any Unibus functions. Received function bits USIC F1 and F2 H are combined with in progress and interlock status signals as shown in Figure 3-34 and on sheet USIK of the prints.

An interlock read masked transfer to the Unibus address space will enable USIK RIFF (J) H, which in turn sets the Read Interlock flip-flop and enables the read interlock timeout counter shown on sheet USIK of the prints. When an interlock write masked transfer follows, it will enable USIK IW H and clear the Read Interlock flip-flop (and thus the timeout counter). This sequence becomes a DATIP-DATO sequence on the Unibus. Either a write masked transfer or an interlock write masked transfer will enable USIK SET UB WD XPCT H, setting the Unibus Write Data Expected flip-flop shown on sheet USIM. When the UBA receives the write data on the next SBI cycle, USIN WD TAG L will enable USIM WD STRB L. This signal then sets the Unibus Write In Progress flip-flop [USIN UWIP (1) H]. Both the read masked transfer and the interlock read masked transfer will enable USIK URIP (J) H which, in turn, sets the USIN URIP (I) H flip-flop when USIS T1A CLK L is true. Either USIN UWIP (I) H or USIN URIP H will enable USIN UIP H, which sets the NPR logic in motion so that the UBA can become the next Unibus master and thereby carry out the function required by the SBI commander nexus. Receipt of any of the four valid function codes, together with a valid Unibus address, will enable USIK UB C/A STRB L, which latches the lower 16 SBI address bits in the C/A word as UAIH TUA (17:02) H (see sheet UAIH of the prints). USIK UB C/A STRB L also enables confirmation bit CNF 0, yielding an ACK confirmation as long as an error has not been detected, enabling confirmation bit CNF 1 (see sheet USIR of the prints).

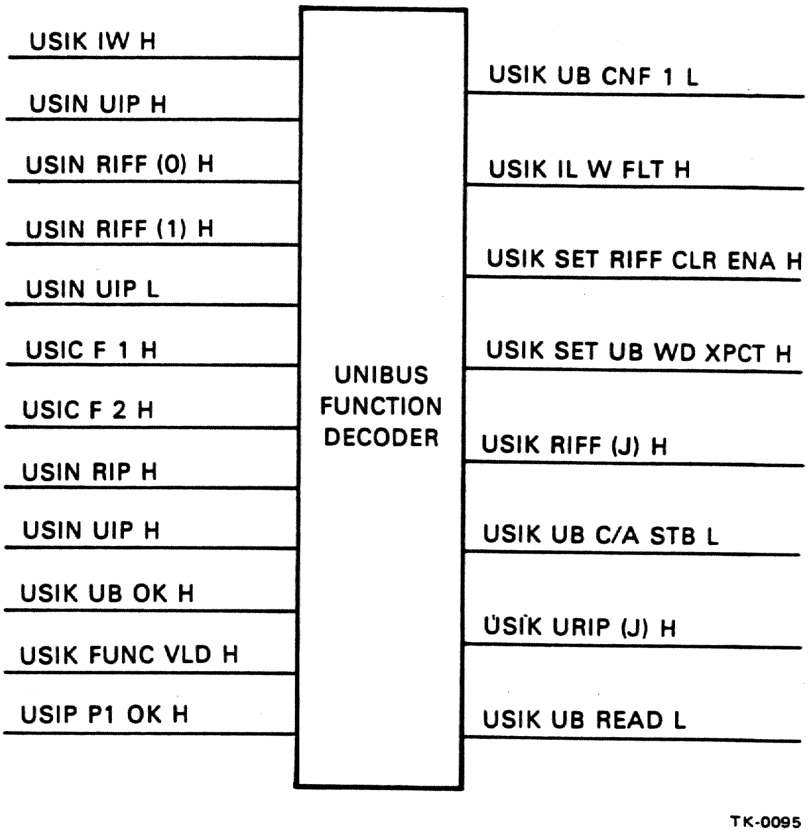


Figure 3-34 Unibus Function Decoder, Block Diagram

3.4.7.2.2 UBA Function Decoder Logic – An address and function decoder circuit enables read and write functions to all of the UBA internal registers, with the following exceptions. Write functions to the BRRVRs, FUBAR, and FMER are prohibited, enabling USIL I WRITE BSY L, to produce an ERR confirmation code. When one of the lower four internal registers (CNFGR, CR, SR, or DCR) is addressed on a write function, USIM CSR WD XPCT L is enabled with T1. When one of the other writable registers is addressed on a write function (BRSVRs, DPRs, or MAPs), the signal USIM MIC WD XPCT H is enabled.

The write data expected logic is necessary because the UBA receives the C/A word and the write data from the SBI sequentially. When the write data comes in, following C/A, the USIH WD TAG L signal is ANDed with the appropriate write data expected signal (see sheet USIM of the prints) to gate the data to the register addressed and enable the appropriate in-progress signal.

3.4.7.3 Simultaneous Activities on the UBA – Since only one Unibus device (including the UBA) can be Unibus master at one time, all activities on the Unibus are sequential. While the Unibus and/or the UBA is busy executing a function, however, the UBA may, in some cases, be free to perform other functions initiated by the software simultaneously. The in-progress logic locks out certain function combinations and allows others. Table 3-12 lists the various software-initiated transfers and the corresponding in-progress signals.

Table 3-12 In Progress Signals

Category	Function	Signal Generated
A	Unibus Read	USIN URIP (1) H
B	DCR or CNFGR Read	USIN USI RIP (1) H
C	CR or SR Read	USIN MIC RIP (1) L
C	FMER Read	
C	FUBAR Read	
C	BRSVR Read	
C	MR Read	
C	DPR Read	
D	BRRVR Read	USIN BRRVRIP (1) H
E	Unibus Write or Interlock Write	USIN UWIP (1) H
F	Unibus Interlock Read	USIN RIFF (1) H
H	BRSVR Write	USIN MIC WIP (1) L
H	MR Write	
H	DPR Write	

The signals generated form three other in-progress signals as shown in Table 3-13.

Table 3-13 Derived In Progress Signals

Category from Table 3-12	Function	Signal
A + E + F	Unibus in Progress	USIN UIP L
A + B + C + D + F	Read in Progress	USIN RIP L
B + C + D + H	Adaptor in Progress	USIN AIP L
F	Interlock	USIN RIFF L

Figure 3-35 provides a key to UBA response to received SBI functions. The in-progress logic may be in any of twelve states, as shown across the top of the figure, depending on the combination of UIP, RIP, AIP, and RIFF. The column on the left lists the possible SBI C/A functions that may be received by the UBA. The matrix formed shows the types of confirmation that will be asserted on the SBI in response to SBI functions for the various UBA in-progress logic states. Note that the receipt of certain SBI functions subsequently alters the state of the in-progress logic. For example, when the in-progress logic is in state 8, a write map register function will be confirmed by an ACK, set AIP, and change the in-progress state to 9. The equations shown in the right column show how BSY, ACK, and FAULT are generated. States are cleared by the completion of the function within the UBA.

	IN-PROGRESS SIGNALS	STATE (IDLE) 1	2	3	4	5	6	7	8	9	10	11	12	EQUATIONS
	UIP	0	1	1	0	1	0	1	1	1	0	0	0	
	RIP	0	0	1	0	0	1	1	1	1	0	1	0	
	AIP	0	0	0	1	1	1	1	0	1	0	1	1	
	RIFF	0	0	0	0	0	0	0	1	1	1	1	1	
SBI C/A FUNCTIONS	UNIBUS WRITE	ACK SET UIP →STATE 2	BSY	BSY	ACK SET UIP →STATE 5	BSY	ACK SET UIP →STATE 7	BSY	BSY	BSY	BSY	BSY	BSY	ACK = $\overline{UIP} \cdot \overline{RIFF}$ BSY = UIP + RIFF
	UNIBUS READ	ACK SET UIP SET RIP →STATE 3	BSY	BSY	ACK SET UIP SET RIP →STATE 7	BSY	BSY	BSY	BSY	BSY	BSY	BSY	BSY	ACK = $\overline{UIP} \cdot \overline{RIFF} \cdot \overline{RIP}$ BSY = UIP + RIFF + RIP
	DCR, CR, SR, CNFGR WRITE	ACK	ACK	ACK	ACK	ACK	ACK	ACK	ACK	ACK	ACK	ACK	ACK	ACK = ALWAYS BSY = 0
	MR, BRSVR, DPR WRITE	ACK SET AIP →STATE 4	ACK SET AIP →STATE 6	ACK SET RIP →STATE 7	BSY	BSY	BSY	BSY	ACK SET AIP →STATE 9	BSY	ACK SET AIP →STATE 12	BSY	BSY	ACK = $\overline{AIP}$ BSY = AIP
	DCR, CNFGR READ SR, CR READ BRRVR READ BRSVR, FMER, FUBAR, MR, DPR READ	ACK SET AIP SET RIP →STATE 6	ACK SET AIP SET RIP →STATE 7	BSY	BSY	BSY	BSY	BSY	BSY	BSY	ACK SET AIP SET RIP →STATE 11	BSY	BSY	ACK = $\overline{AIP} \cdot \overline{RIP}$ BSY = AIP + RIP
	UNIBUS INTERLOCK READ	ACK SET RIFF SET UIP SET RIP →STATE 8	BSY	BSY	ACK SET UIP SET RIP SET RIFF →STATE 9	BSY	BSY	BSY	BSY	BSY	BSY	BSY	BSY	ACK = $\overline{UIP} \cdot \overline{RIFF} \cdot \overline{RIP}$ BSY = UIP + RIFF + RIP
	UNIBUS INTERLOCK WRITE	FAULT	FAULT	FAULT	FAULT	FAULT	FAULT	FAULT	BSY	BSY	ACK SET UIP CLR RIFF →STATE 2	ACK SET UIP CLR RIFF →STATE 7	ACK SET UIP CLR RIFF →STATE 5	ACK = RIFF • $\overline{UIP}$ BSY = RIFF • UIP FAULT = $\overline{RIFF}$

TK-0307

Figure 3-35 SBI Functions/In Progress State/Confirmation

3.4.8 Confirmation Field Logic

The UBA will send a confirmation code on the SBI in response to every transmission addressed to it. A valid C/A word, write data word, or read data word addressed to UBA controlled address space will cause the UBA to send ACK, unless the UBA is busy (BSY confirmation) performing a function that is not compatible with the new transfer. In response to an invalid address, mask, or function code, the UBA will send an ERR confirmation. Figure 3-36 shows the signals that develop the confirmation code. Note that USIR CNF 0, 1 (1) H will be latched at time T1 and that the signals latched will be transmitted with T0 of the following SBI cycle. The confirmation for a particular SBI cycle can be determined from the flowcharts shown in Figures 3-26 and 3-35.

Confirmation signals latched from the SBI following data transmission by the UBA perform two functions. USIR R CNF 1 H and USIR R CNF 0 H are decoded to test for the acknowledge code. First, UCBM ACK H is fed into the microsequencer branch logic, as shown in Figure 3-37. Second, the signal that produces ACK is also ANDed with UCBE DTED H, which is enabled on the third SBI cycle following a UBA transmission on the SBI. If the transmission was a C/A for the read transfer, UCBM EN RDRQD L, together with signals from the microsequencer, will enable the read data expected logic. The confirmation signals are also fed directly into the microsequencer branch logic. The microsequencer tests the condition of the CNF signals when making the decision as to whether or not to retransmit C/A information on the SBI.

3.4.9 Tag Field Logic

The UBA transmits a tag code whenever it transmits information on the B field of the SBI. UCBL TAG SEL 1 and 0 H are transmitted as BUS SBI TAG 2 and 1 H, respectively, as shown in Figure 3-38. The tag encoder logic enables UCBL TAG SEL 1 H only when the UBA transmits write data. Note that the microsequencer will always be executing a routine when the UBA is transmitting a command, and it will assert UCBF OP CD BO H for write data and command/address. TAG SEL 1 H can be enabled only on the first or second SBI cycle following transmission of C/A [UCBL DTEA OR DTEB (1) H true], whereas the TAG SEL 0 H can be enabled only if the SBI is being requested and read data is not to be sent. Note also that if read data is to be transmitted, UCBF OP CD B OH will be false. TAG SEL 1 and 0 H will, therefore, be false; their inclusive OR, USIC T TAG 0 H, will be false; and a tag code of zeros will be sent on the SBI.

When the UBA latches data and the accompanying tag signals from the SBI, a 3:8 decoder enables one of four logic circuits, depending on whether the information in the B field is an interrupt summary read command, write data, C/A, or read data, as shown in Figure 3-39.

3.4.10 Parity Field Logic

The parity logic on the UBA checks the parity of all information received on the SBI. It generates parity on all information that the UBA transmits. P0 is checked and generated for the tag, ID, and mask fields. P1 is a function of the B field, as shown in Figure 2-5. Figure 3-40 shows the parity logic for P1 in block diagram form. Notice that the circuit uses the odd side of the parity generator and the even side of the parity checker.

If the UBA transmits a field with an odd number of bits asserted, the odd output of the parity generator will be true, causing the bus transceiver to assert one parity bit and making the total even. If the UBA asserts an even number of bits on the SBI, the odd parity output will be false, and the parity bit on the SBI will remain unasserted.

The Parity OK signal (USIA P OK H, L) enables the following circuits when true:

Interrupt Summary Response Logic
Function Decoder Logic
Write Data Expected Logic
Read Data Expected Logic

When Parity OK is false, it sets bit 31 in the configuration register and enables FAULT on the SBI.

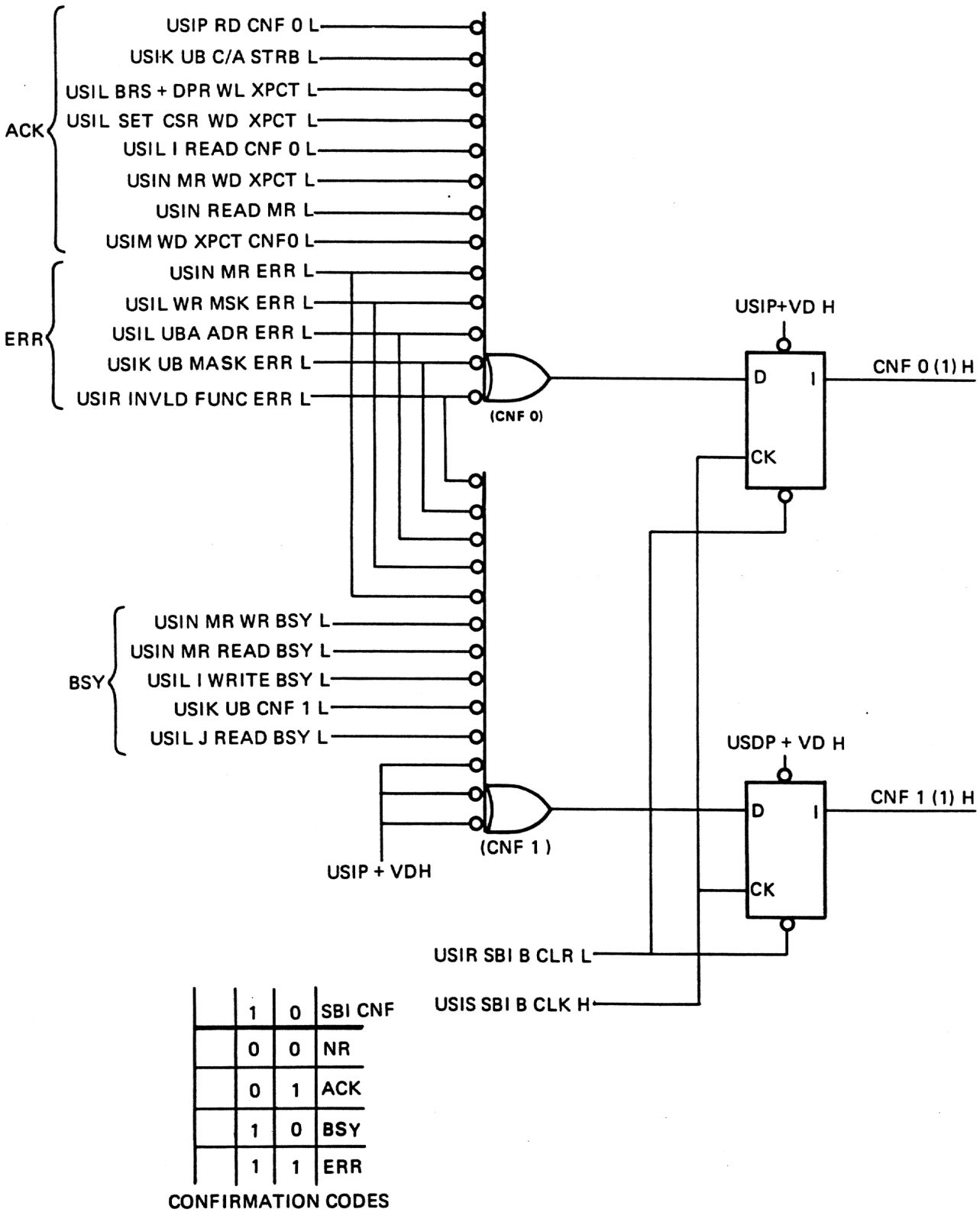


Figure 3-36 Confirmation Encoder Logic

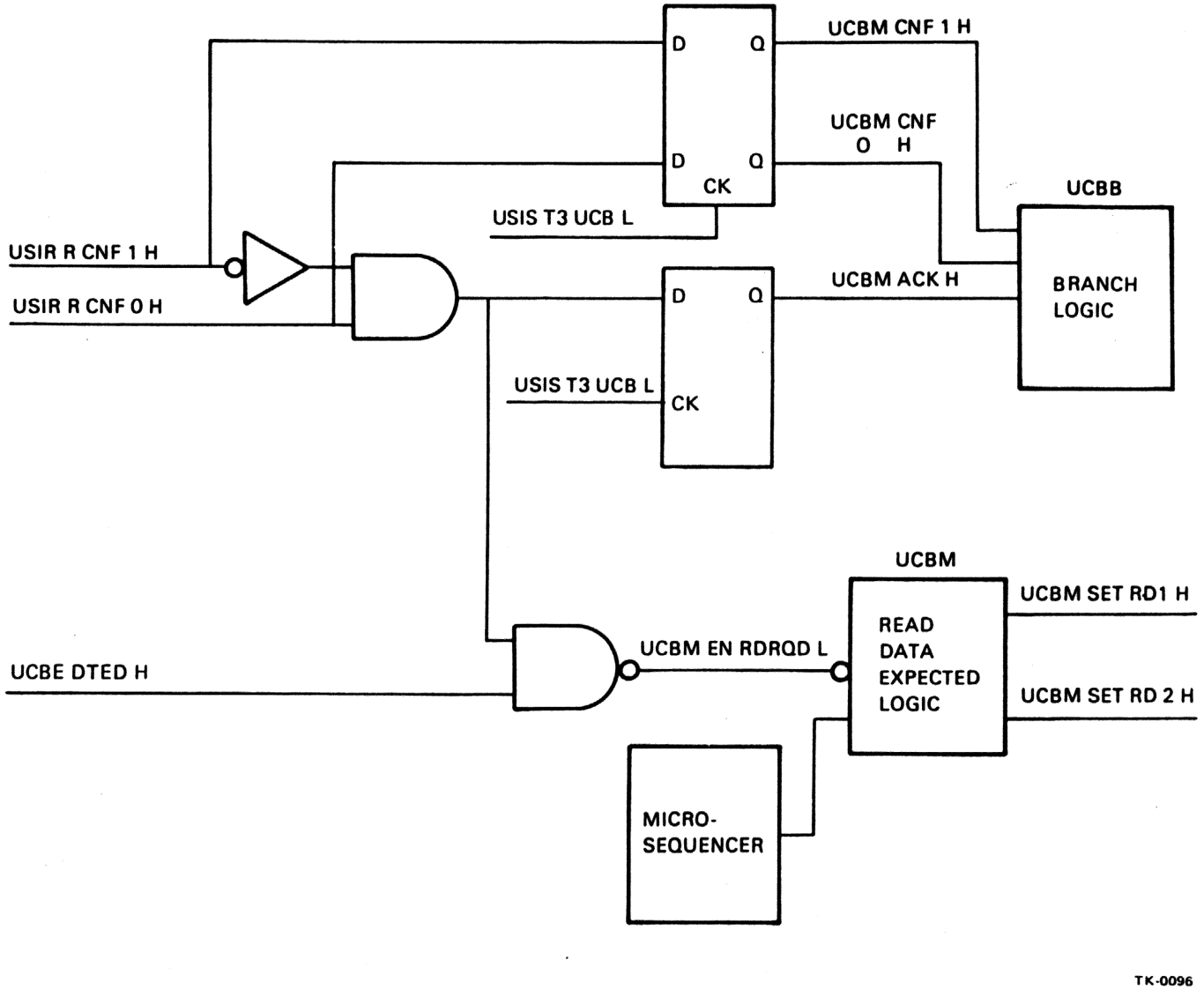


Figure 3-37 Receipt of ACK Confirmation

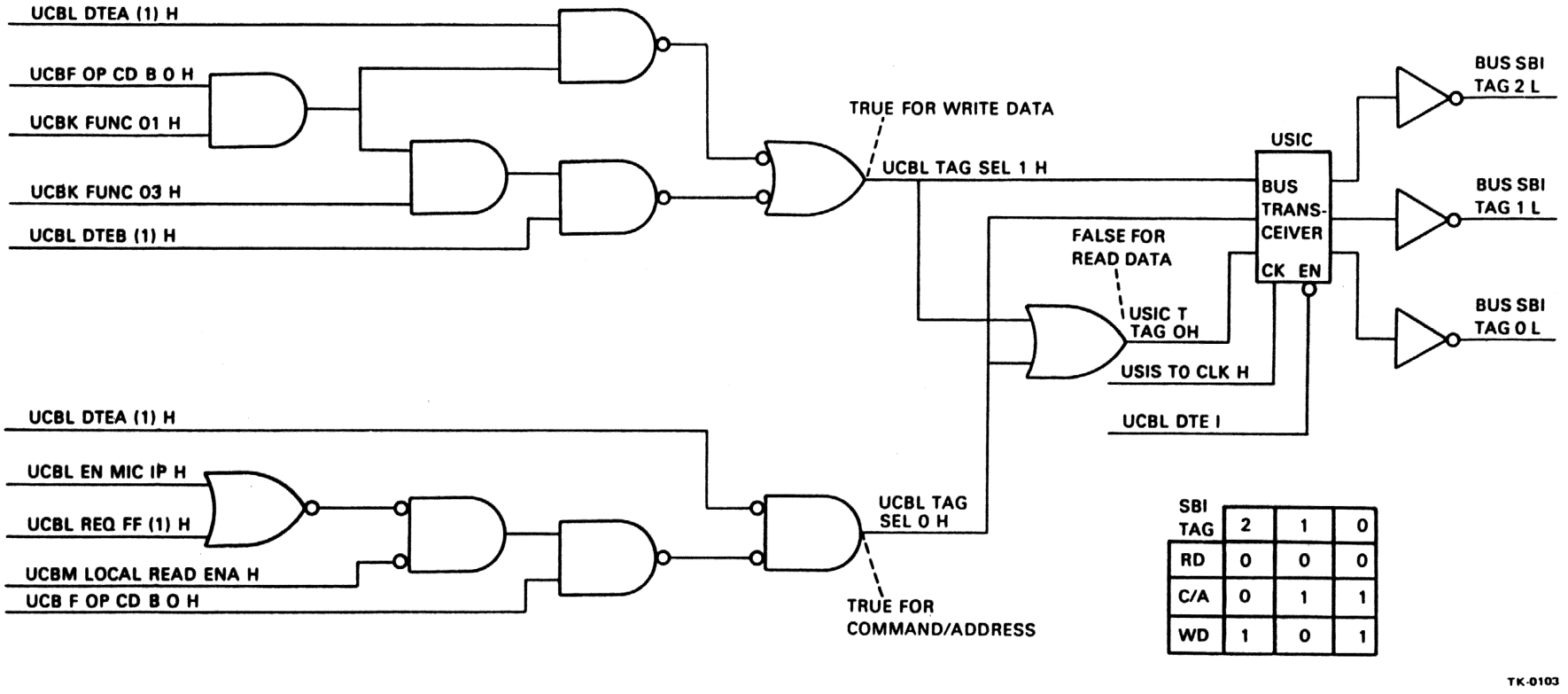


Figure 3-38 Tag Encoder Logic

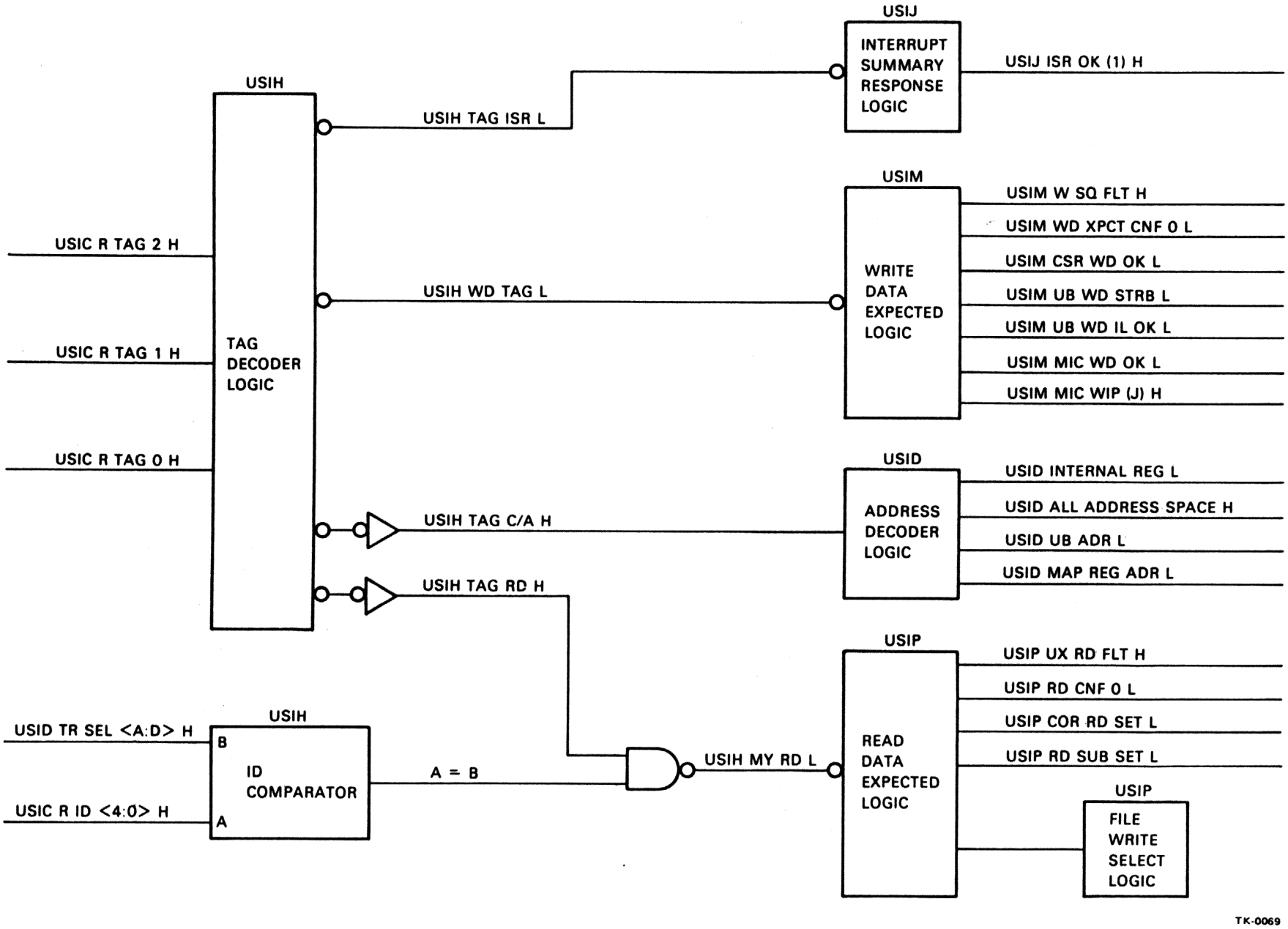


Figure 3-39 Tag Decoder Logic

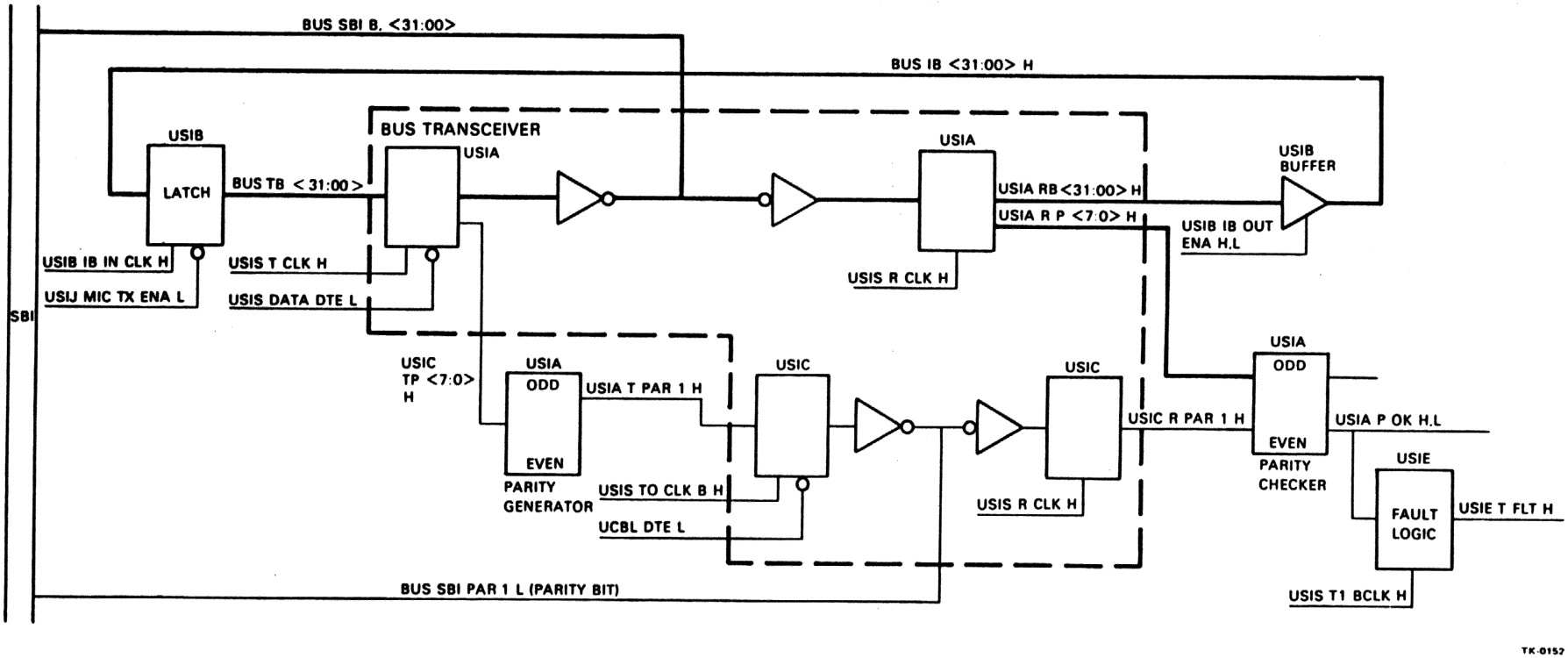


Figure 3-40 Parity Logic (P1)

3.4.11 Fault Field Logic

The SBI fault field contains one bit only, as shown in Figure 3-41. When the UBA detects a fault condition on the SBI, it generates USIE T FLT H. This signal causes the UBA to transmit FAULT on the SBI at T0 of the following cycle. At the same time, USIS T CLK H enables the clock input to the flip-flops for bits (31:27) of the configuration register. When USIR R FLT H is subsequently enabled at T3 of the same cycle, the clock input to CONFG (31:27) is disabled, and the fault bits are latched. Paragraph 2.9.1 gives a summmary and interpretation of the five conditions that this circuit will detect.

Note that the VAX-11/780 CPU will keep FAULT L asserted on the SBI after the UBA releases the signal. The CPU thus ensures that the fault bits remain latched in the configuration register until the software can read them.

After the UBA latches the fault signal from the SBI, it checks to determine whether or not it was transmitting when the fault was detected. The signal UCBL DTE L is true when the UBA transmits data on the SBI, and this is latched as USIR DTE A (1) H with the occurrence of T1 of the same SBI cycle. This signal feeds a chain of two other flip-flops, as shown in Figure 3-42. The second flip-flop is set on the first SBI cycle following the transmission. At the same time, if the UBA has detected an SBI fault, it latches the signal indicating the fault type as shown in Figure 3-41. The output of the second flip-flop, inverted and ANDed with the false signal USIC R B FLT H, qualifies the D input to the third flip-flop. At T0 of the next SBI cycle (second cycle following the faulty transmission) the UBA asserts BUS SBI FAULT L on the SBI. Then at T1 of the same cycle, the third flip-flop is set. When the UBA receives the FAULT signal (which it is still transmitting on the SBI), USIC RB FLT H is enabled, qualifying USIR MY XMT FLT H and setting the latch which forms bit 26 of the configuration register. Notice that when FAULT is released later, USIC RB FLT L will be false and the latch will be reset.

3.5 UNIBUS INTERFACE LOGIC

The Unibus interface logic is distributed over both the UMD and the UAI modules.

3.5.1 Unibus In Side and Unibus Out Side

All Unibus signal lines are connected to the UBA. However, two sets of Unibus cables must be connected to the UBA: one for peripheral devices (slot 5), and the other for the Unibus terminator board (slot 6). The peripheral side is called the Unibus Out side. The Unibus In side runs to the terminator.

Most Unibus signals are common to both sides. However, the Unibus In side signals associated with bus arbitration are electrically separate from the corresponding signals on the Unibus Out side as shown in Table 3-14.

Notice that the signal names are unique for each line (with the exception of BUS NPG OUT H). Notice also that a given signal may have two names. BUS NPR OUT L and BUS NPR L, for example, are two names for the same signal. The name changes with the point of reference.

3.5.2 Unibus Address and Control Logic

On every DMA transfer the I/O device initiating the transfer asserts an address and a control code on the Unibus lines BUS UA (17:00) L and BUS UC (1,0) L, as shown in Figure 3-43. The signal UCBA DMA ATT L will be false if the microsequencer is in the idle state, enabling the address information into the address hold latches. The idle state of the OP B register enables UCBE ADD HLD TSE L, and the address signals are routed through the latches to the BUS AD (15:00) H lines, where the upper nine bits are selected by a 2:1 multiplexer (sheet UMDD of the prints) to form the address of a map register. Then, if the map register has been validated, the microsequencer is in the idle state, and the map address range logic and the Unibus control signals qualify UCBA DMA ATT L, the Unibus address will be latched, and the starting address of a ROM routine will be enabled on the MPC lines.

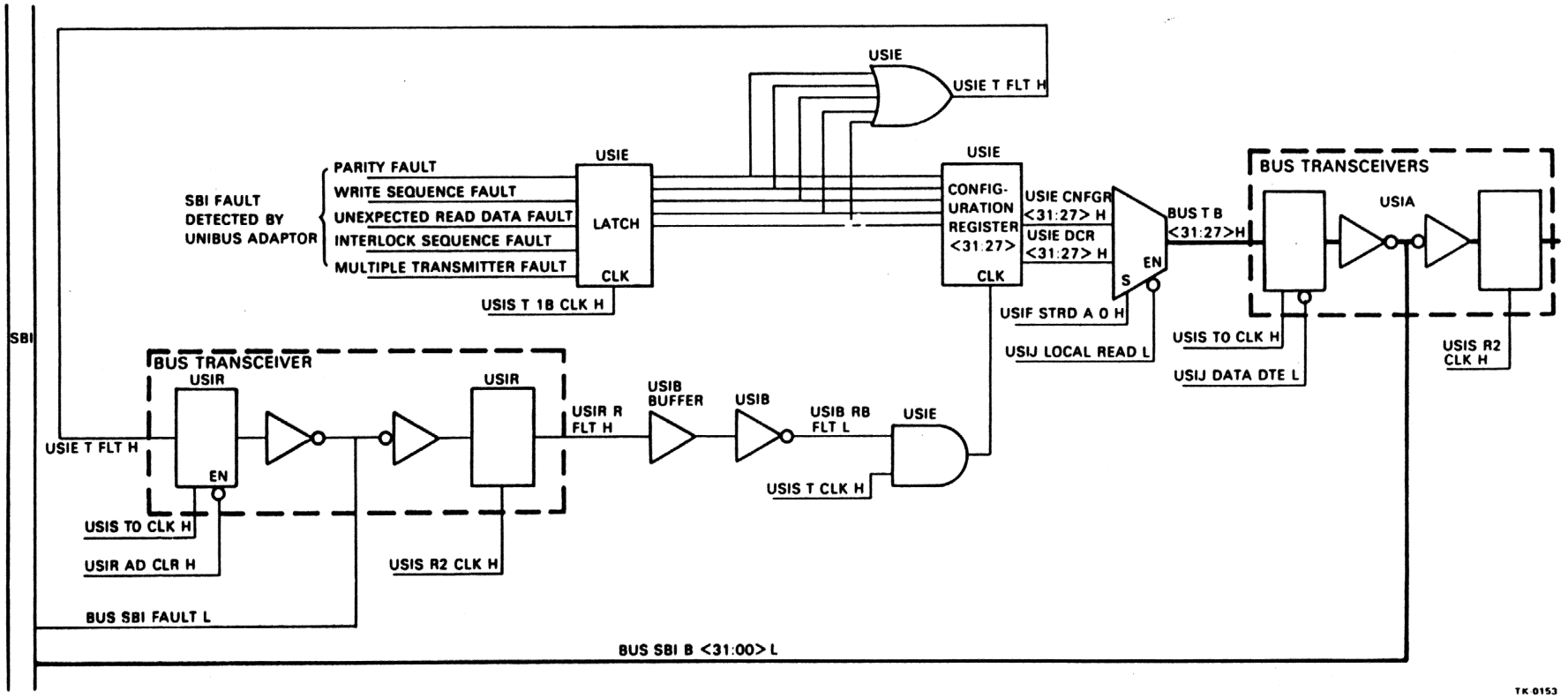


Figure 3-41 Fault Logic

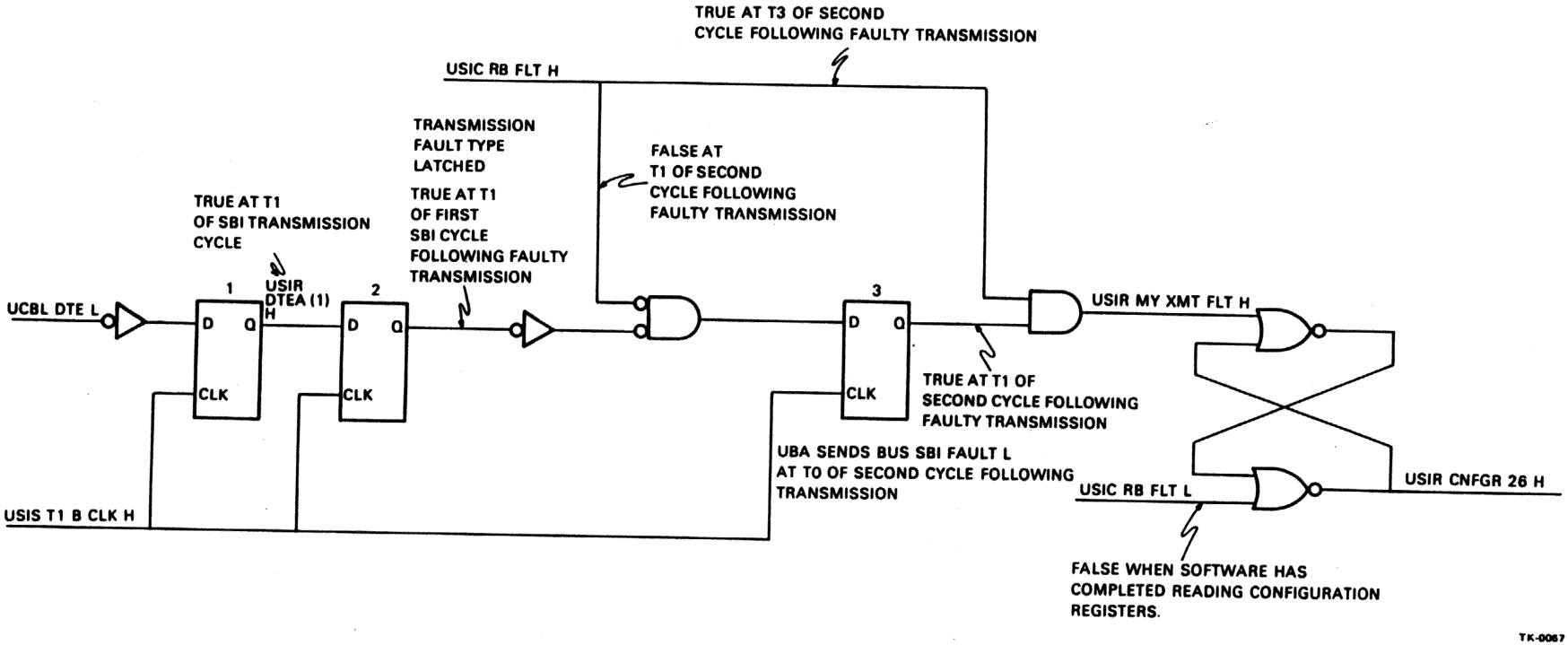


Figure 3-42 MY Fault Logic

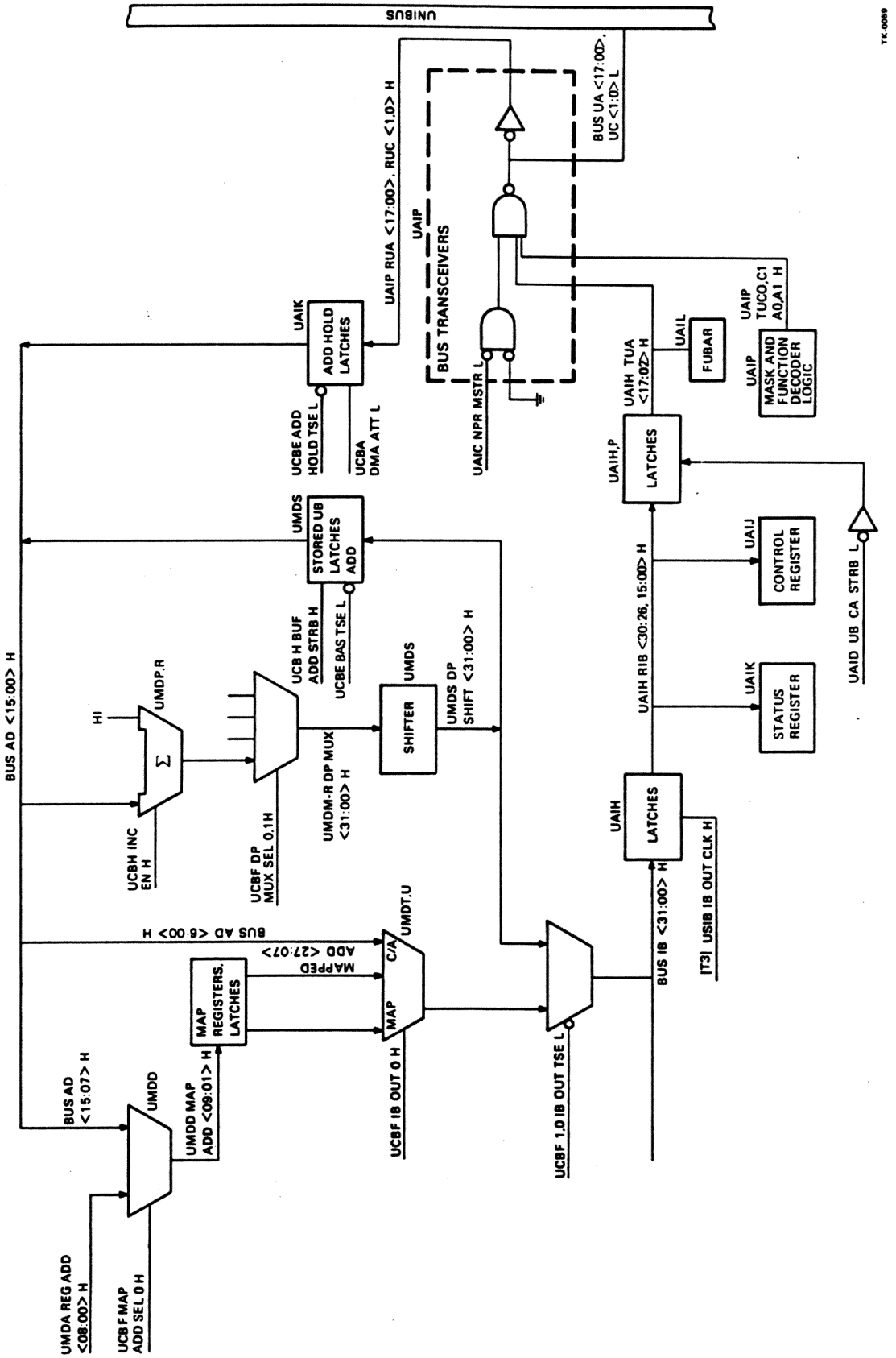


Figure 3-43 Unibus Address and Control Lines and Associated Logic

Table 3-14 Unibus In Side and Unibus Out Side Signal Names

Terminator		UB In Side		UB Out Side	
Terminator Input	**Terminator Output**	**Output to Terminator**	**Input From Terminator**	**Output to I/O Device**	**Input From I/O Device**
BUS NPR L		BUS NPR OUT L	←	←	BUS NPR IN L
	BUS NPG OUT H	→	BUS NPG IN H	→ BUS NPG OUT H	
BUS SACK L	←	BUS SACK OUT L	←	←	BUS SACK IN L
		BUS INTR OUT L	←	←	BUS INTR IN L
		BUS BR (7:4) OUT L	←	←	BUS BR (7:4) IN L
		BUS BG (7:4) IN H	→	→ BUS BG (7:4) OUT H	

If the DMA transfer is a DATO or DATOB and the transfer is mapped through a BDP, the upper 16 Unibus address bits are stored in the buffered data path RAM, locations 23 and 33, in anticipation of a purge operation.

When the software initiates a purge operation following a DMA write transfer, the microsequencer routes the stored Unibus address through the data path multiplexer and the shifter, to the BUS AD (15:00) lines. The upper nine bits are then selected as the map address inputs, so that the transfer can be mapped to the correct SBI address. Figure 3-44 shows the Unibus address and control logic with relation to the UBA as a whole.

The Unibus address will be incremented by the adder shown in Figure 3-43, in three cases, as explained in the following text.

- DATO/B transfer with B0, A2, A1 = 7 – The adder will increment the Unibus address asserted on BUS AD (15:00) by the address hold latches and then store it in the BDP RAM, in anticipation of a purge operation in which the single byte that overflows into the next quadword will be transferred.

- Prefetch – When the UBA performs a prefetch operation following a DMA read sequence that has emptied the BDP, the microsequencer will cause the adder to increment the Unibus address in the address hold latches. The incremented address is then gated through the data path multiplexer and shifter and back on the BUS AD (15:00) H lines to select the map register and form the SBI address.

- DATI with B0, A2, A1 = 7 - When a Unibus device performs a DMA read transfer with byte offset to the last word of a quadword (B0, A2, A1 = 7), and this transfer is mapped through an empty buffered data path, then the UBA initiates a read masked transfer on the SBI to obtain the first byte. When the data is received, it is moved to the low byte of the DOUT latches. The adder then increments the Unibus address stored in the address hold latches so that the UBA can read the next quadword from SBI memory to obtain the high byte of the data requested (second read routine, MF11).

Whenever the Unibus address is incremented, the new address will always point to a new longword location in memory and may point to a new map register as well, if it has crossed a page (512 byte) boundary.

When the software initiates a transfer to Unibus address space, the lower 16 bits of the C/A word are latched from the BUS IB lines and enabled to the transmit side of the Unibus address transceivers as BUS UA (17:02) when the in-progress logic asserts UAID UB C/A STR B L. The lower two Unibus address bits and control bits C1 and C0 are formed in the mask and function decoder logic and gated directly to the Unibus transceivers, as shown in Figure 3-43 and on sheet UAIP of the print set. Then, when the UBA becomes the bus master, UAIC NPR MSTR L enables the address and control signals onto the Unibus.

3.5.3 Unibus Data Logic

The UBA receives data from the Unibus on a DMA write transfer and on a software-initiated read transfer. In either case, UAIA REC DATA GATE H will be true, as shown in Figure 3-45, enabling the received data, and gating it to the data multiplexer.

Data retrieved from the SBI memory on a DMA read transfer is gated through the data path shifter. The microsequencer enables the high or low Unibus word (both bytes) from the shifter, depending on the state of the received Unibus address bit UA01. The signal UAIA EN UB WD H will be false (Figure 3-46), enabling UMDV DOUT BUFF TSE L, and gating the data to the bus transceivers, via the BUS UB DOUT lines (Figure 3-47).

When the software initiates a write transfer to the Unibus address space, the data is gated from the SBI, via the BUS IB lines to the latches shown in Figure 3-47 and on sheets UMDA and UMDB of the prints. The 32 bit longword [UMDA, UMDB RIB (31:00)] then forms two inputs to a 2:1 multiplexer, where the high or low Unibus word is selected, depending on the state of UAIU TUA 01 H. If the in-progress logic enables the transfer, UAID UB WD STRB L opens the latches shown on sheet UMDV of the prints. UAIA EN UB WD H will be true, enabling UMDV UB WD TSE L, and the data from the IB lines will be gated to the Unibus transceivers via the BUS UB DOUT lines, as shown in Figure 3-47. Figure 3-48 shows the Unibus data logic in the context of the UBA block diagram.

The Unibus data line control logic will assert UAIA UB DATA EN L on either type of transfer, as shown in Figure 3-46, enabling the data to be asserted on the Unibus data lines.

3.5.4 Unibus Arbitration Logic

3.5.4.1 Bus Request Logic - When a Unibus device asserts its assigned BR level on the Unibus, the UBA can respond in any of the three ways. If the UFIS and BRIE bits (5 and 6) of the control register are set, the request will be passed on directly to the VAX-11/780 CPU, as shown in Figure 3-49. If the UFIS bit is true and the BRIE bit is false, the request will be ignored. If the UFIS bit is false, the request will be passed to the Unibus Terminator, where it is, in effect, ignored.

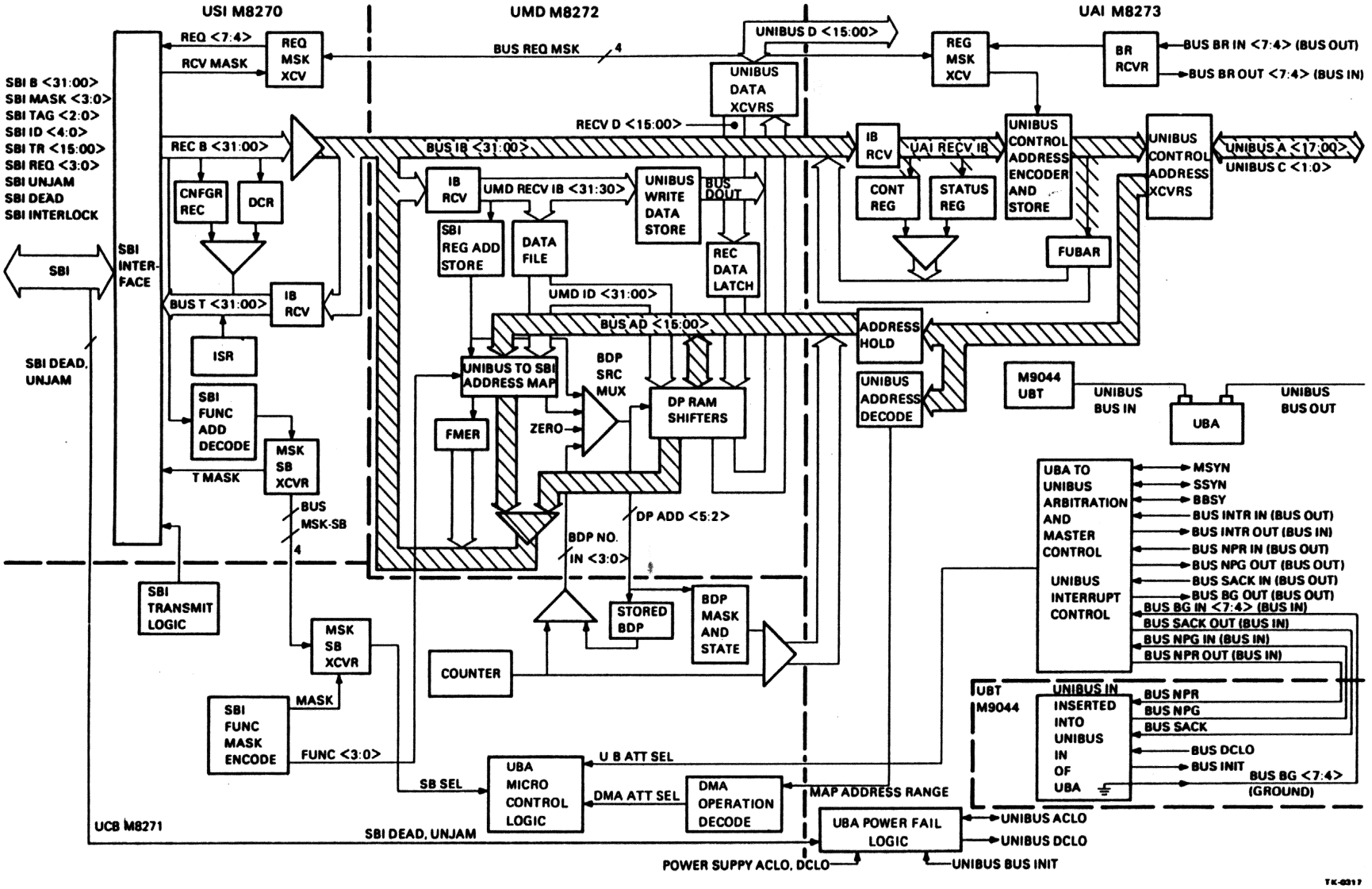


Figure 3-44 Unibus Address and Control Logic/ UBA Block Diagram

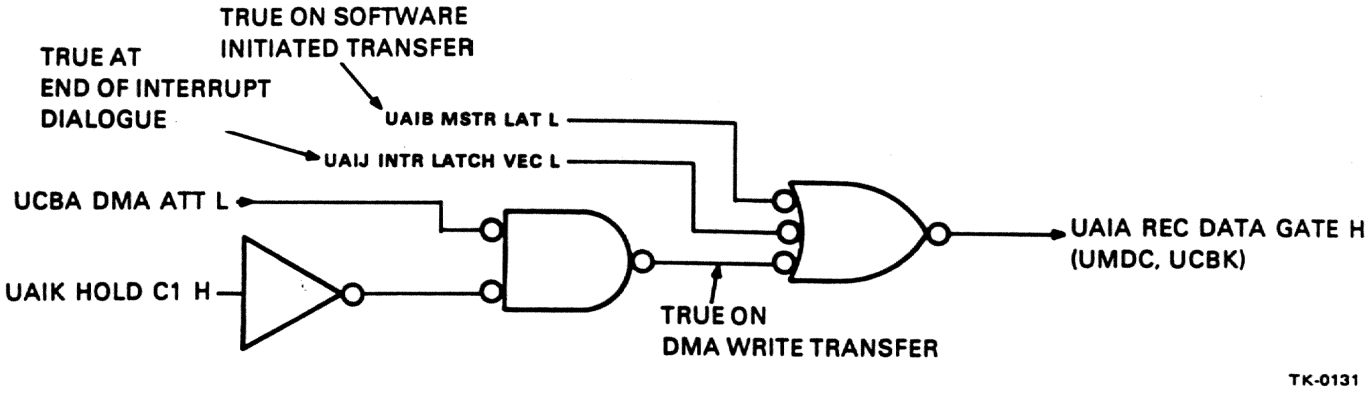


Figure 3-45 Qualifying the REC DATA GATE Signal

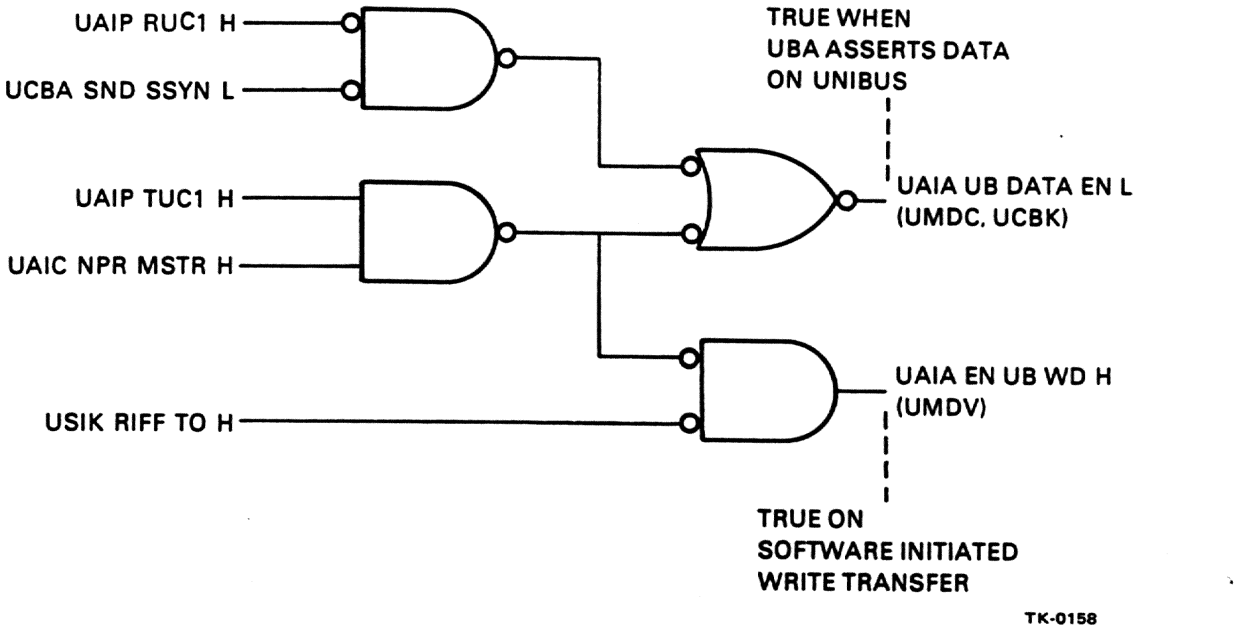


Figure 3-46 Unibus Data Line Control Signals

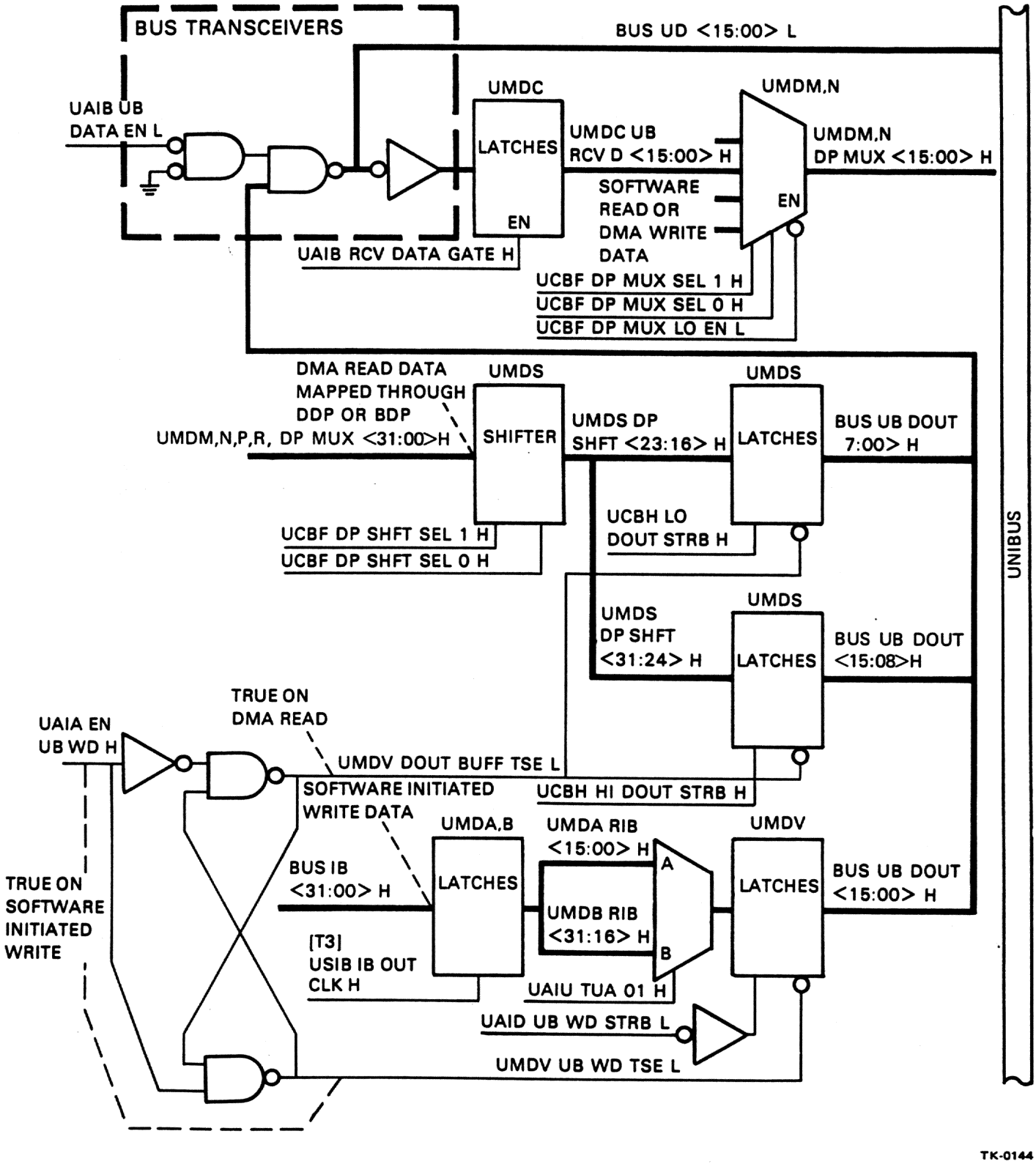


Figure 3-47 Unibus Data Lines and Associated Logic

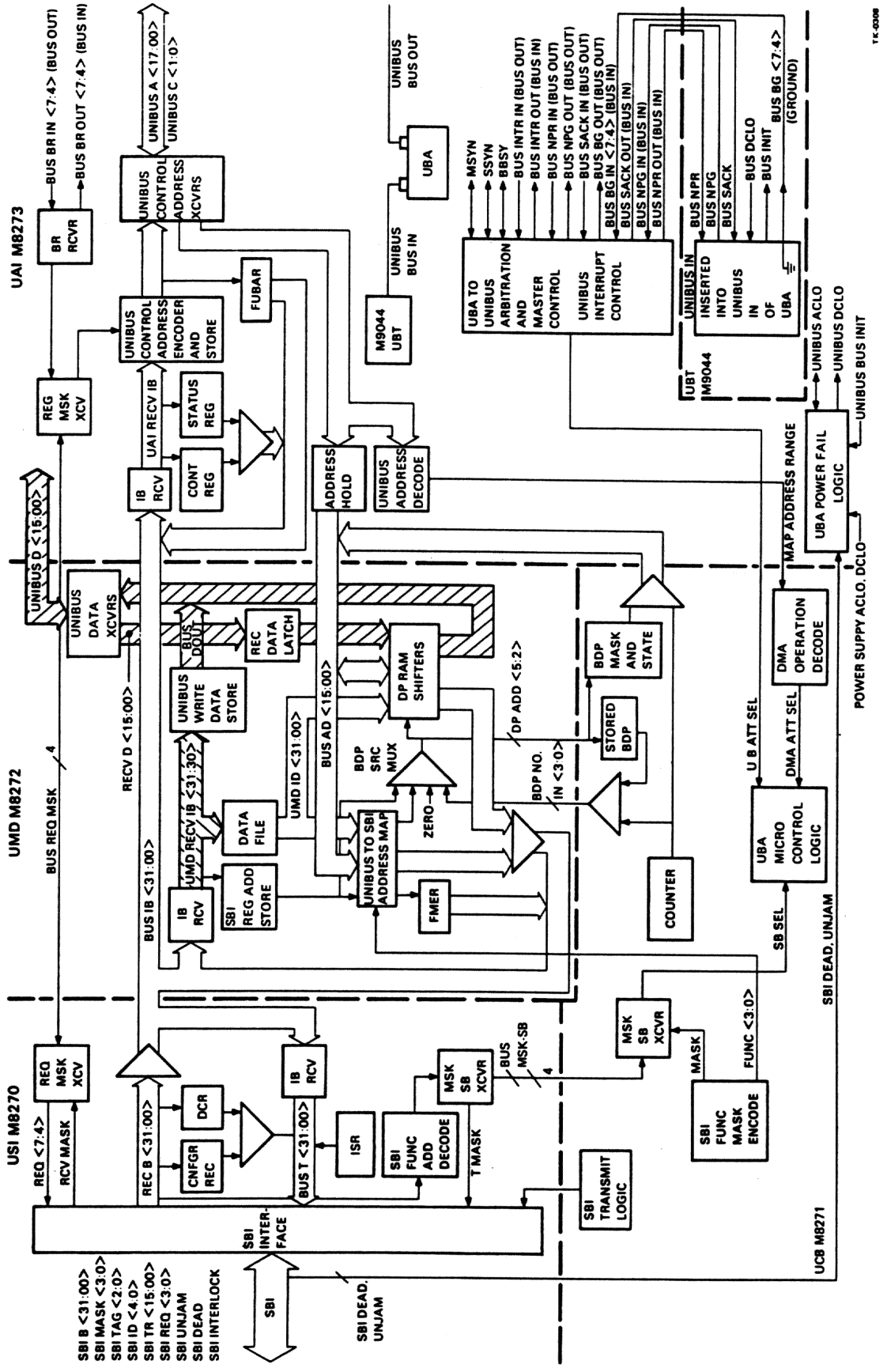


Figure 3-48 Unibus Data Logic/UBA Block Diagram

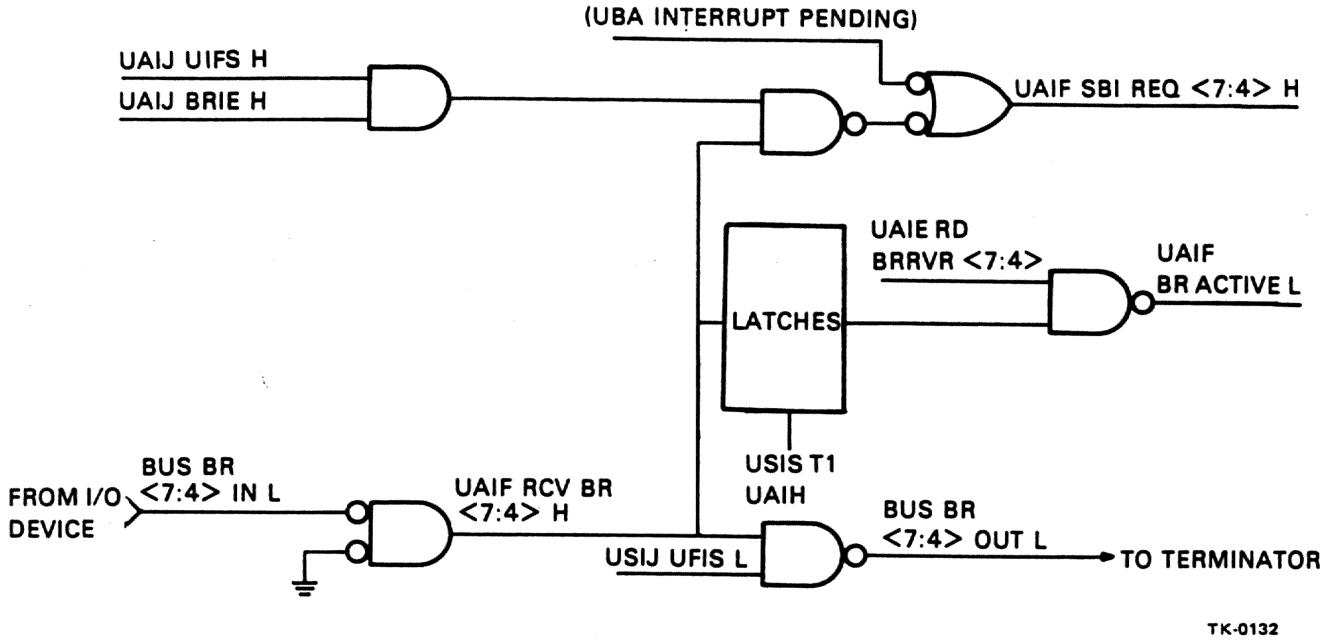


Figure 3-49 BR Logic

3.5.4.2 NPR and NPG Logic – The UBA will send NPR to the terminator in response to any of three events, as shown in Figure 3-50. An NPR from an I/O device will be passed on directly. When the software initiates a Unibus transfer, the in-progress logic will enable UAIB UIP H enabling the UBA to assert NPR so that the UBA can become the next Unibus master. When the software performs a read BRRVR routine following a bus request from an I/O device and an interrupt summary read/response exchange, the UBA asserts NPR to hold the bus until the UBA can issue the bus grant.

Once the NPR signal has been asserted on the Unibus, no second device can issue an NPR. The UBA receives the signal from the I/O device as BUS NPR IN L, as shown in Figure 3-50. The reader should note that the signal names change when passing from point to point in this circuit. The UBA passes the signal to the terminator as BUS NPR OUT L. The terminator receives the same signal as BUS NPR L as shown in Figure 3-51. If BUS INIT L and BUS SACK L are both false, the terminator will grant the request, enabling BUS NPG OUT H. The UBA then receives this signal, renamed as BUS NPG IN H. The UBA will react to the grant in any of three ways as explained in the following text.

NPG in Response to a Bus Request [BR (7:4)] – When the UBA receives BUS NPG IN H from the terminator and the software is responding to a bus request (as outlined in Paragraph 3.5.4.1), the UBA asserts UAIC NPG RCVD H. This enables one of four BG signals [BUS BG OUT (7:4) H] and stops the NPG grant from going further. Refer to Paragraph 3.5.5 for further details.

NPG in Response to a Data Transfer Command – When the CPU initiates a data transfer to a Unibus device, the USIN UIP H signal (Unibus in-progress) will be asserted on the UBA. As shown in Figure 3-51, if UAIC NPR MSTR H and UAIC T SACK H are true, the UBA will become the next bus master, asserting SACK and BBSY on the Unibus. In this way the UBA takes the grant and prevents it from passing on to the Unibus.

NPG in Response to an NPR from an I/O Device – If the NPR originates on one of the Unibus I/O devices, and the UBA does not take the NPG, the UBA will pass the grant from the terminator straight through to the Unibus as BUS NPG OUT H.

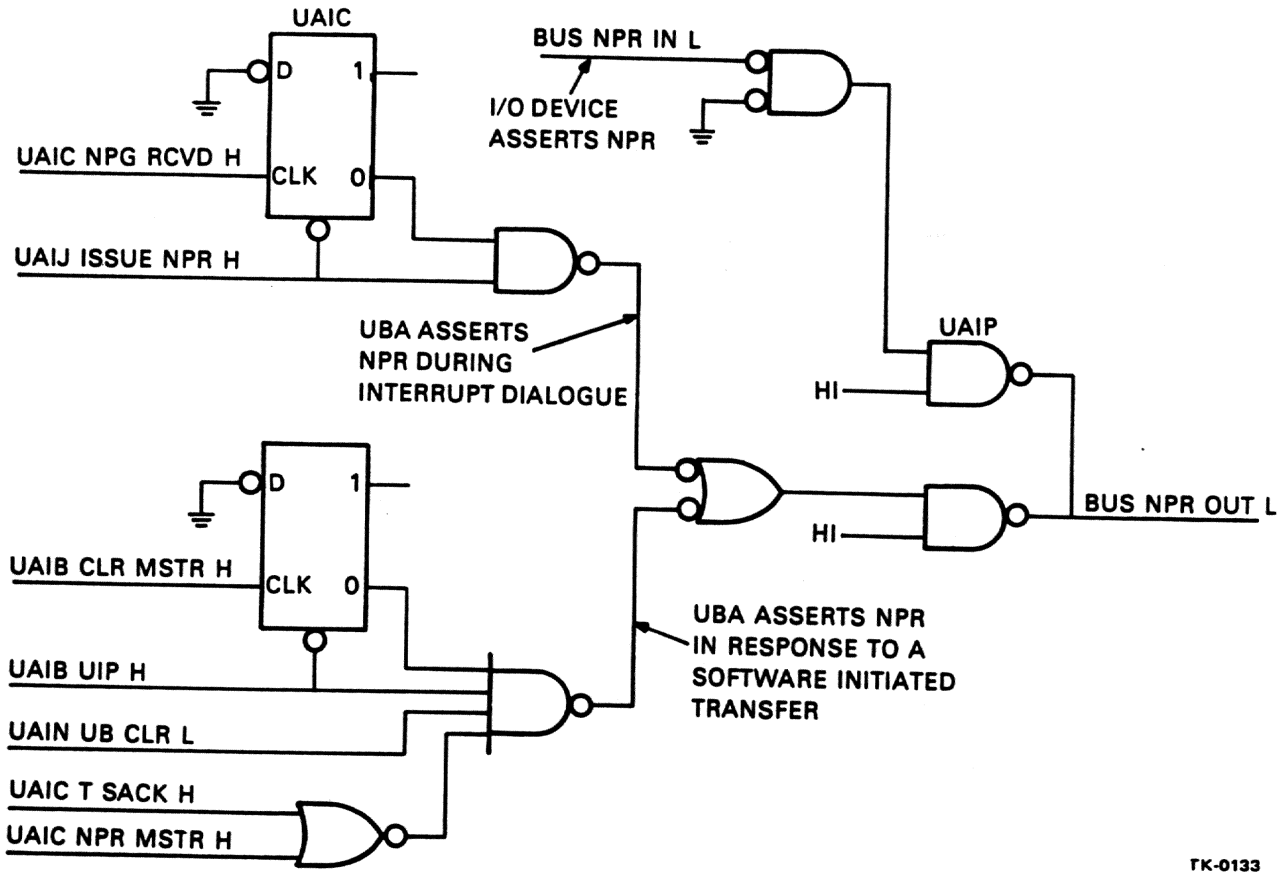


Figure 3-50 NPR Logic

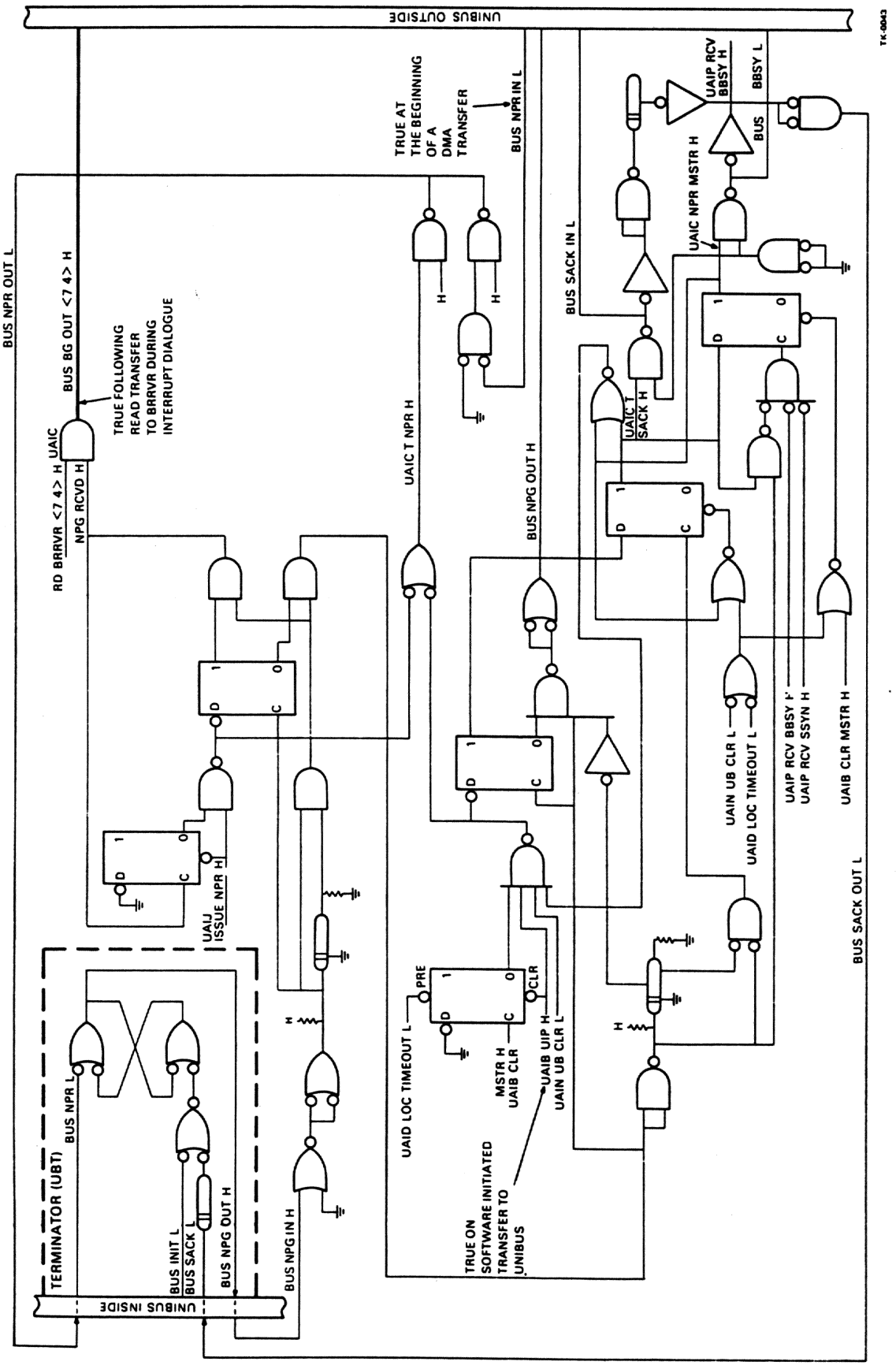


Figure 3-51 NPR and NPG Logic

3.5.5 Unibus I/O Device Initiated Interrupts

The UBA will pass Unibus I/O device generated interrupts on to the VAX-11/780 CPU if control register bits 5 (BRIE) and 6 (UFIS) are set. Figure 3-52 shows the circuits involved in block diagram form. If these control register bits are not set, the UBA will not pass interrupts to the CPU.

An eight-step explanation of the interrupt dialogue follows. Notice that the numbers circled on the diagram correspond to the paragraph numbers below.

1. A Unibus I/O device asserts a BR line [BR (7:4) L].

2. If enabled by the control register, the UBA passes the request on to the CPU via one of the four SBI lines [SBI REQ (7:4) L]. Some time later, the software responds with an interrupt summary read command corresponding to the request level asserted.

3. When the UBA receives the interrupt summary read command and the bit set in the SBI B field matches the SBI request level transmitted, the interrupt summary response logic becomes active. A 4:16 decoder circuit translates the 4-bit TR code, which identifies the UBA, to a bit pair within a 32-bit field. The UBA then transmits this 32 bit word, the request sublevel, as the interrupt summary response, in the B field of the SBI. The CPU then transfers control to the UBA interrupt service routine.

   The service routine executes a read command to retrieve the interrupt vector from the BRRVR corresponding to the level of the interrupt.

   If the corresponding BRRVR FULL bit of the status register is set, the UBA will assert the contents of the BRRVR addressed as read data on the SBI. However, the BRRVR FULL bit should not be set at this time.

4. If the BRRVR FULL bit is not set, the UBA will issue NPR OUT, sending a request to the terminator board for arbitration.

5. The terminator board should respond with NPG.

6. The NPG from the terminator enables the UBA to assert the appropriate level bus grant signal on the Unibus, providing that the BR line corresponding to the selected BRRVR is still asserted. Note that the UBA will continue to inhibit the assertion of NPG on the Unibus while UAIJ ISSUE NPR H is true, thus ensuring that no other Unibus device interferes with the interrupt dialogue. The requesting device then accepts the grant and asserts SACK. The UBA receives SACK and transmits it to the terminator, which in turn negates NPG. SACK inhibits arbitration of other NPRs by the terminator until the BR device gains control of the Unibus. The negation of NPG at the terminator in turn disables the BG.

7. When the requesting device becomes Unibus master, it negates SACK and asserts INTR L and the interrupt vector on the Unibus. The negation of SACK then disables UAIJ ISSUE NPR H on the UBA. The UBA receives INTR L and the vector and it loads the vector into the appropriate BRRVR. The corresponding BRRVR FULL bit is set at this time.

8. The UBA then asserts the contents of the BRRVR on the SBI B lines as read data. The BRRVR FULL bit is cleared if and when ACK for the read data is received. If the UBA does not receive ACK for the read data, it retains the vector in the BRRVR. A subsequent read to the BRRVR will cause the UBA to send the vector as read data again. ACK will clear the BRRVR FULL bit. At this point, with the interrupt dialogue finished, the UBA interrupt service routine invokes the Unibus I/O device service routine indicated by the interrupt vector received. Figure 3-53 shows the Unibus interrupt sequence in flowchart form.

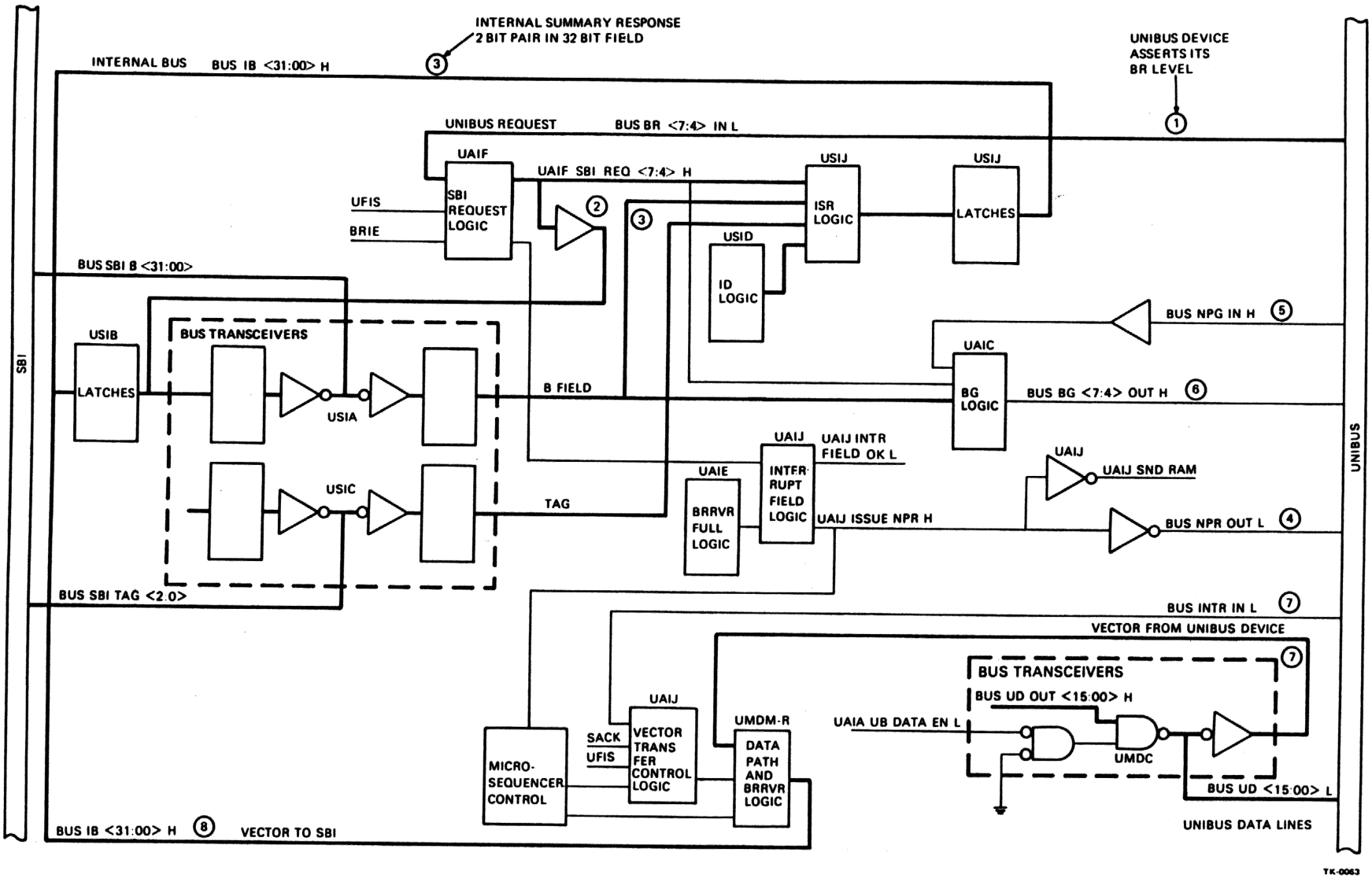


Figure 3-52 Unibus Device Interrupt Logic
on the UBA, Block Diagram

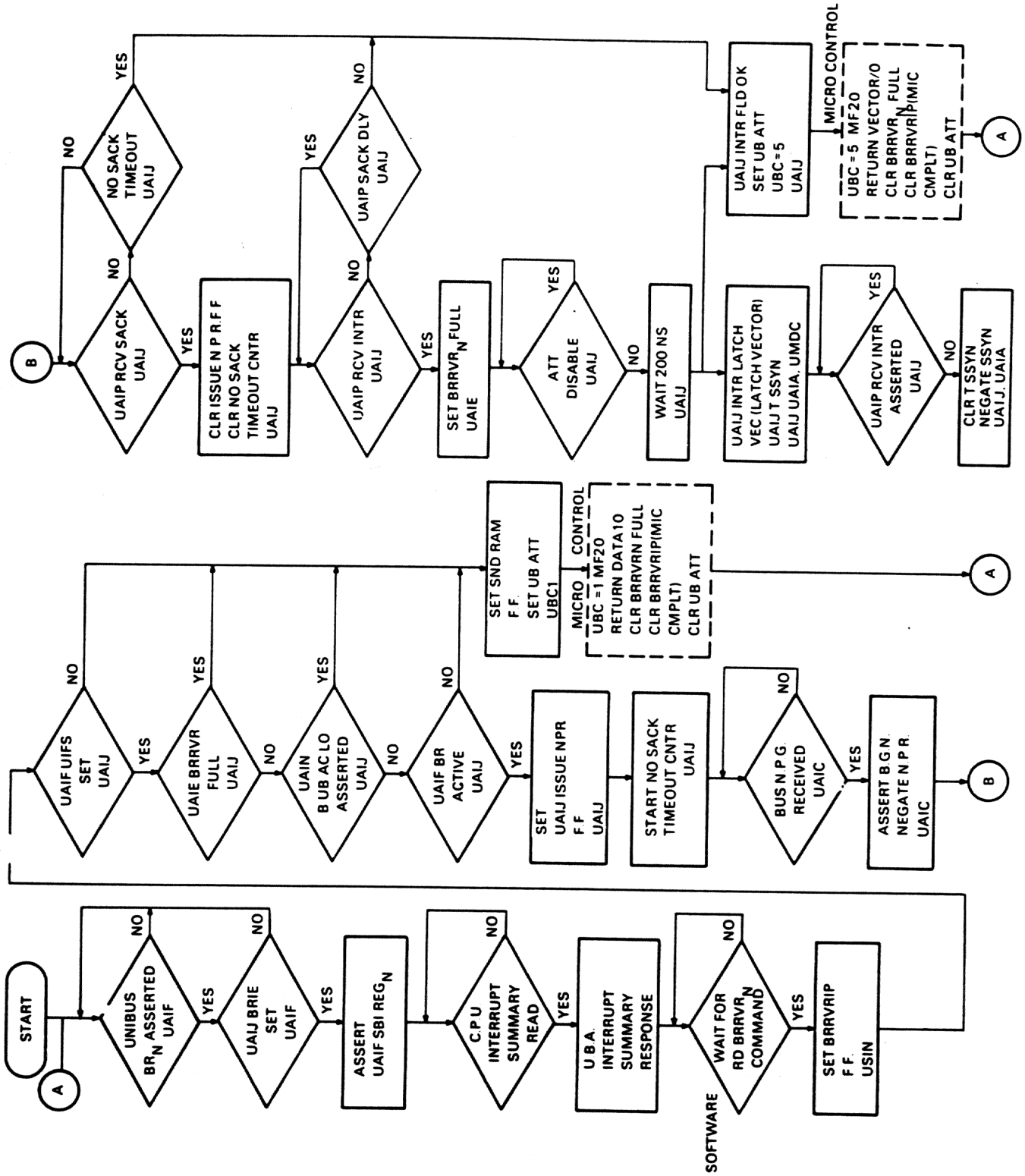


Figure 3-53 Unibus Interrupt Sequence Flowchart

3.5.6 Unibus Master Control and Timeout Logic

The UBA controls the Unibus during software initiated transfers through an eight state counter circuit shown in Figure 3-54.

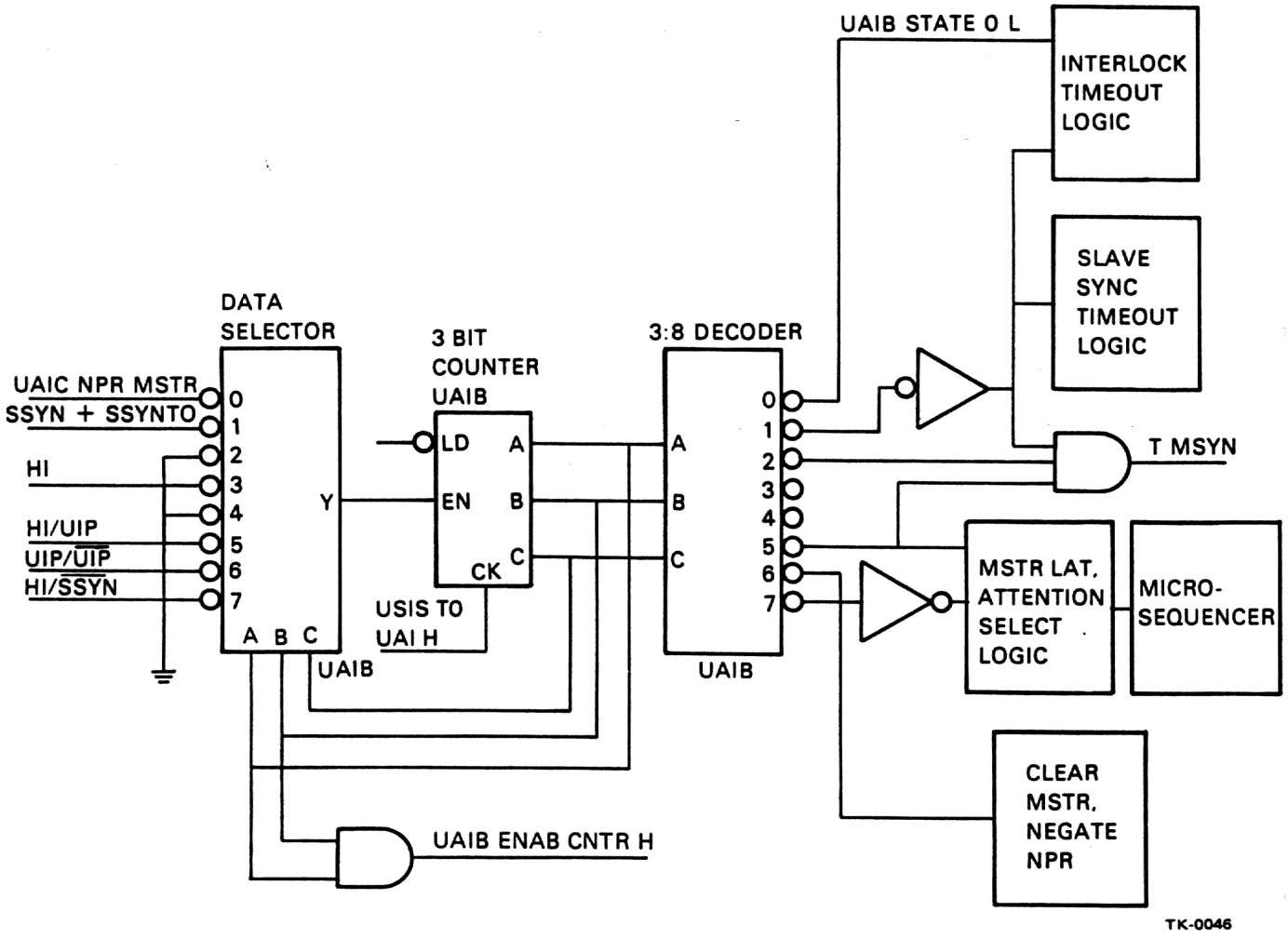


Figure 3-54 Unibus Control State Counter, Block Diagram

When the circuit is not active, the state counter should be in the zero state. USIN UIP L will be false, enabling UAIB ZERO CNTR L, as shown in Figure 3-55. This signal keeps the Unibus select and Slave Sync timeout counter (sheet UAID) loaded with zeros. When the software initiates a Unibus transfer, the in-progress logic enables USIN UIP L, starting the timeout counter. If the UBA does not become the NPR master within 256 SBI cycles (51.2 μs), the Set Unibus Select Time Out signal (UAID SET UBSTO H) will set bit 1 of the status register, causing the UBA to lock the FUBAR and initiate an interrupt to the CPU. The Unibus select timeout also enables UAID LOC TIMEOUT L at T2, disabling the signal UAIC T NPR H, so that the bus request is withdrawn.

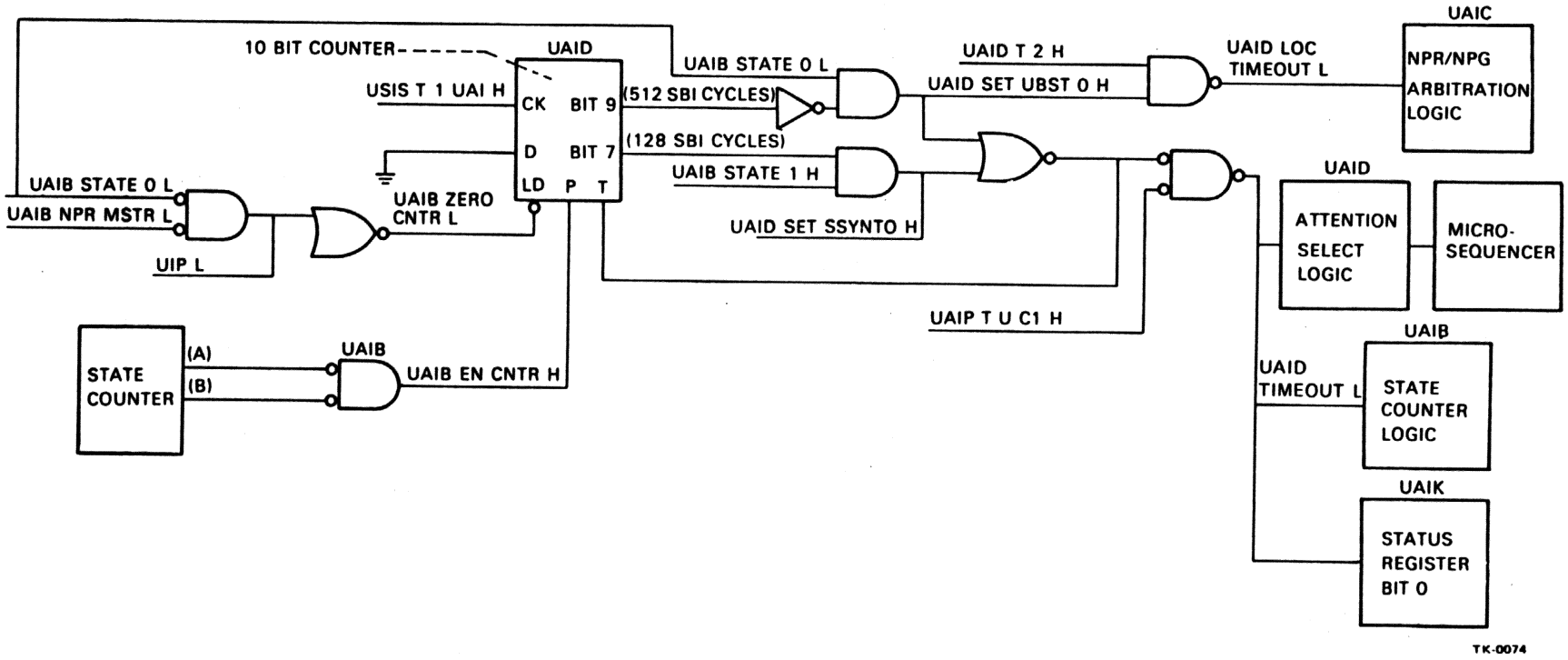


Figure 3-55 Unibus Select and Slave Sync Timeout Counter Control Logic

If the UBA passes through state 0 successfully, it will assert Master Sync at state 1. Then, if the I/O device addressed does not respond with Slave Sync before 64 SBI cycles (12.8 μs) have elapsed, the Set Slave Sync Time Out signal (UAID SET UBSSYNTO H) will be asserted locking the FUBAR. SET SSYNTO will advance the state counter and set bit 0 in the status register.

Either timeout will cause the microsequencer to jump from the idle state to the starting address of the MASTER DATI Time Out routine (UB C0, MF20), if the function attempted was a read transfer. Figure 3-55 shows the Unibus slave sync and timeout counter logic. Table 3-15 shows the signal combinations which produce the timeout indication.

Table 3-15 Timeout Conditions

UIP	NPR MSTR	STATE 1	STATE 0	SET UBSTO	SET SSYNTO
F	F	F	F	F	F
F	F	F	T	F	F
F	F	T	F	F	F
F	T	F	F	F	F
F	T	F	T	F	F
F	T	T	F	F	F
T	F	F	F	F	F
T	F	F	T	T	F
T	F	T	F	F	T
T	T	F	F	F	F
T	T	F	T	F	F
T	T	T	F	F	T

Note: These conditions must remain static for 64 SBI cycles in order to qualify UAID SET SSYNTO H and for 256 SBI cycles in order to qualify UAID SET UBSTO H.

The 3-bit eight state counter shown in Figure 3-54 enables one of the eight outputs of the 3:8 decoder, enabling or disabling one of the functions of the Unibus control logic. The state of the counter also enables one of the eight inputs of the 8:1 data selector. USIS T0 UAI H clocks the counter every 200 ns. The circuit will remain in its current state until the selected input to the data selector becomes low, enabling the state counter to advance to the next state. The load input will cause the state counter to skip from state 1 to state 5 on a read and from state 3 to state 0 on a write.

Figure 3-56 is a flowchart showing the sequence of events controlling and controlled by the state counter logic on a transfer to Unibus address space.

Notice that the read portion of the interlock read masked-interlock write masked (DATIP-DATO/B) sequence differs from the regular read masked (DATI) sequence. The state counter will not advance from state 5 to state 6 in the interlock sequence until USIN UIP L is false, indicating that the tag field has been shifted from the code designating read data (000). Notice also that the state counter will wait in state 6 until USIN UIP L is asserted again, indicating that the SBI function is in progress.

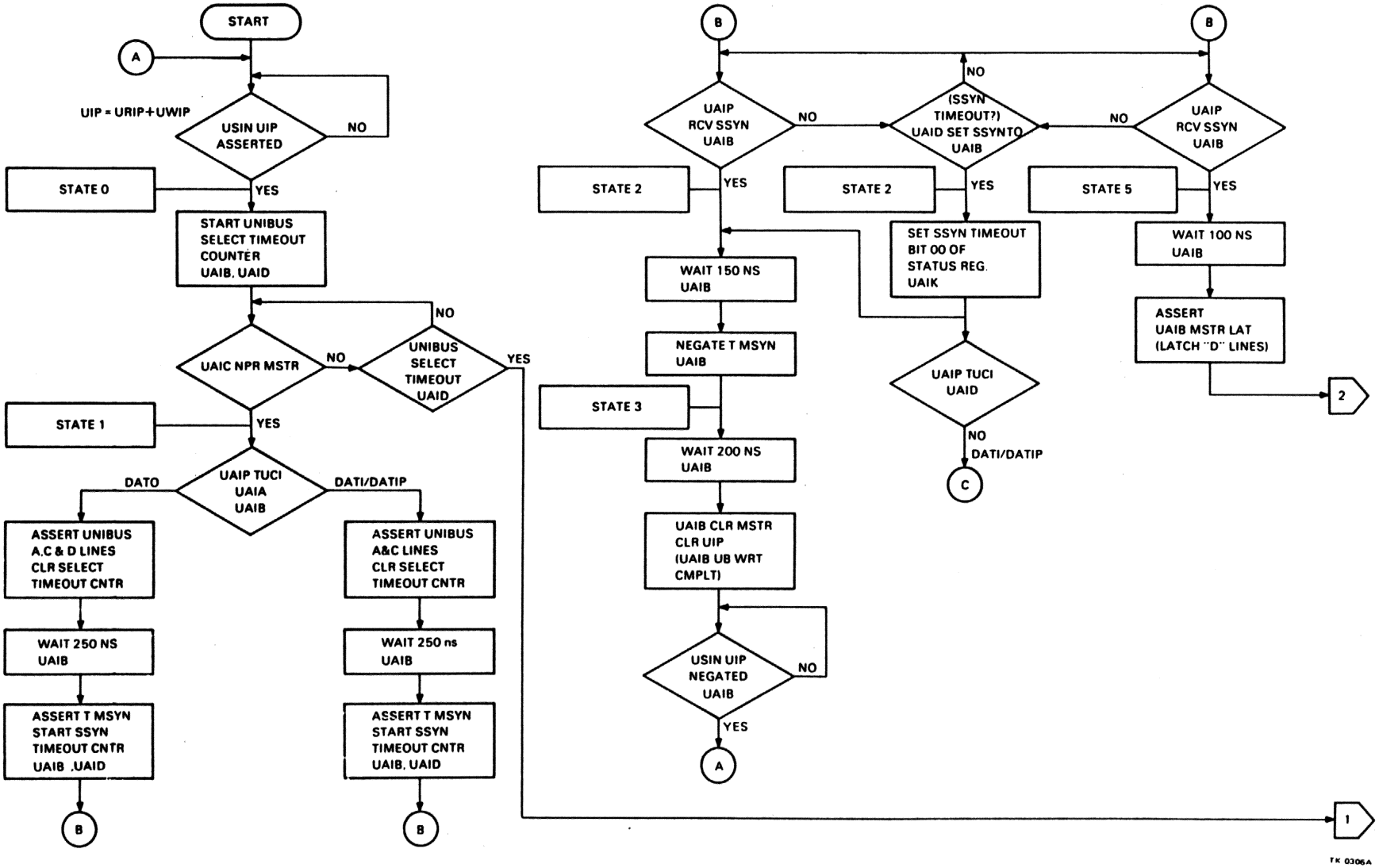


Figure 3-56 State Counter Flowchart for a Software Initiated Transfer to the Unibus (Sheet 1 of 2)

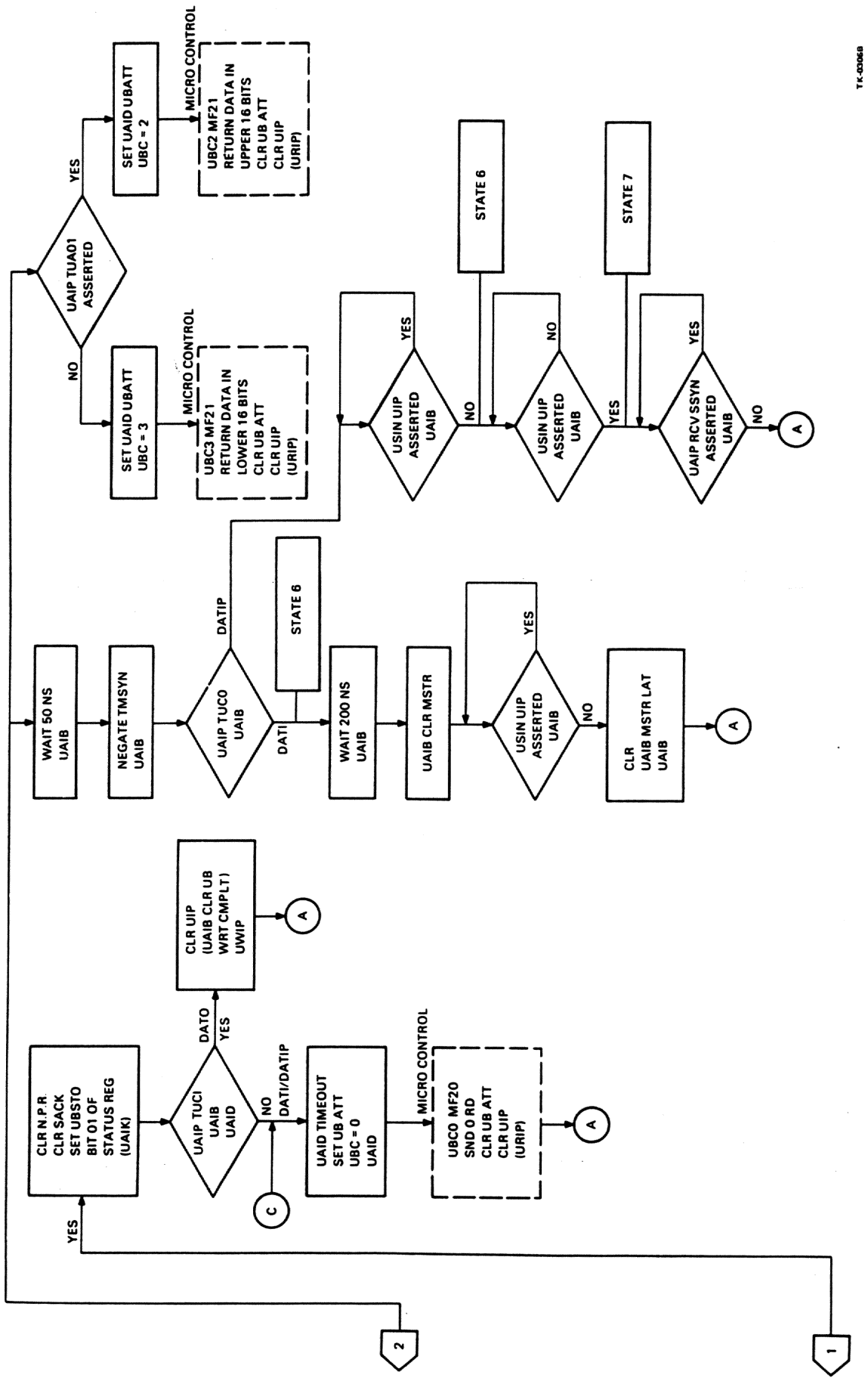


Figure 3-56 State Counter Flowchart for a Software Initiated Transfer to the Unibus (Sheet 2 of 2)

3.5.7 Unibus Attention Select Logic

If an event requiring microsequencer attention occurs on the Unibus or in the Unibus control logic of the UBA, and the microsequencer is in the idle state, the attention select logic will cause the microsequencer to jump to the starting address of the appropriate microroutine. Figure 3-57 shows the attention select logic in block diagram form.

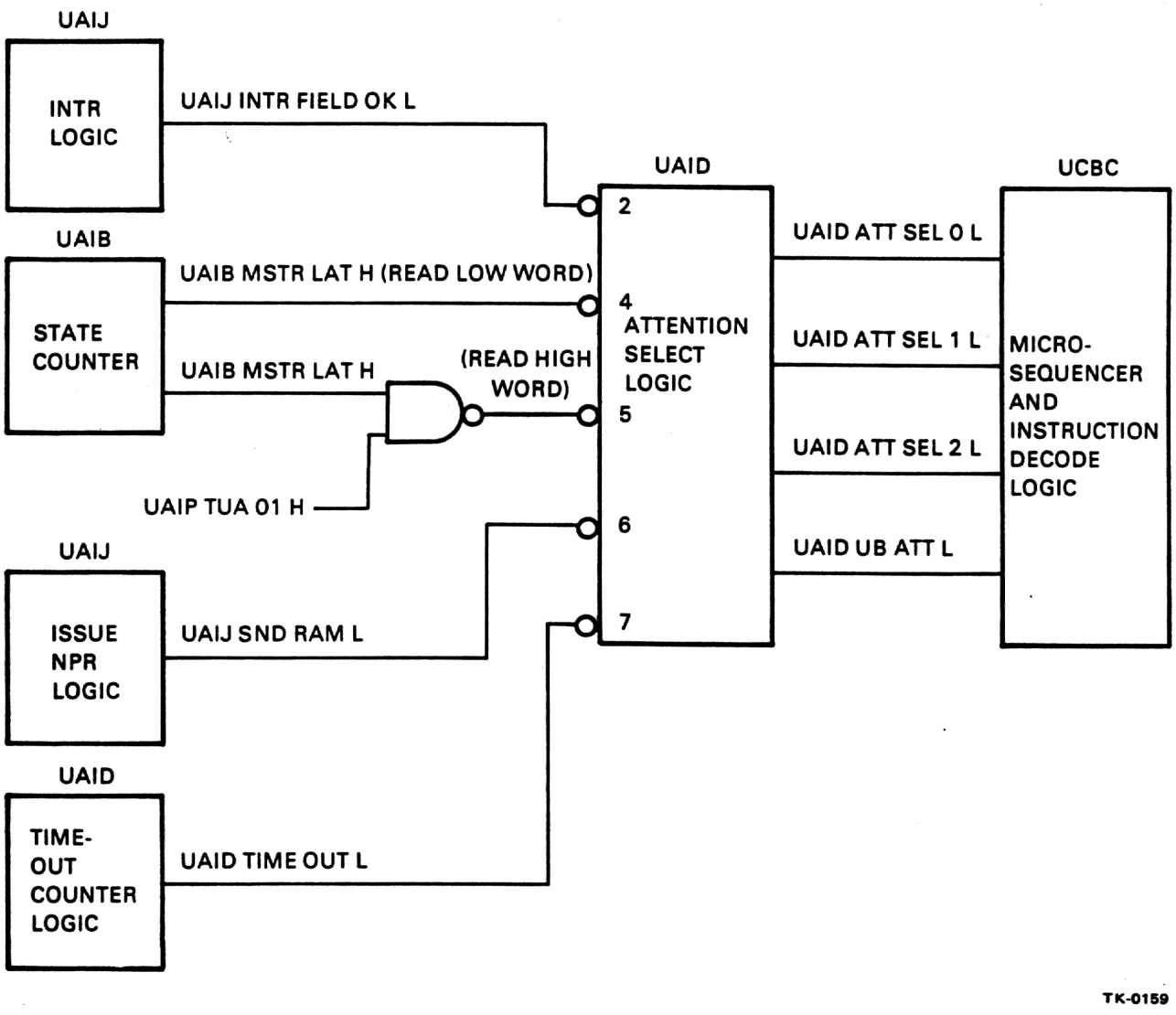


Figure 3-57 Unibus Attention Select Logic, Block Diagram

When the five input signals to this circuit are false, the four outputs will also be false. Then, when one of the input signals is enabled, UAID UB ATT L and one or more of the three ATT SEL signals will also be enabled. A priority encoder on the input side of the circuit ensures that the logic will respond only to the condition that demands the highest priority.

The UAIJ INTR FIELD OK L signal (lowest priority) calls for microsequencer attention after receiving the vector from the interrupting I/O device during an interrupt dialogue.

UAIB MSTR LAT H will be enabled following the receipt of SSYN on a read data transfer initiated by software when the low word is addressed (Unibus address bit A01 is false).

UAIB MSTR LAT H and UAIP TUA 01 H will be true when the software reads the high word, with the receipt of SSYN.

The UAIJ SND RAM L signal demands microsequencer attention if NPR is not issued and the UBA does not enable a bus grant during an interrupt dialogue. SND RAM causes the microsequencer to send the current contents of the BRRVR addressed as the interrupt summary response, instead of obtaining a vector from the interrupting device on the Unibus.

UAID TIMEOUT L (highest priority) calls for attention from the microsequencer whenever the slave sync timeout or the Unibus select timeout occurs.

3.5.8 Map Address Range Logic

All addressing on DMA transfers is mapped from a Unibus address to an SBI address through the map registers. If the address asserted by the device initiating the transfer is valid and the UBA is not busy, the Map Address Range (UAIA MAR H) signal will enable the DMA attention logic, as shown in Figure 3-58.

If the upper five Unibus address bits are high, an I/O device register is being addressed and MAR H will be disabled.

If a Unibus memory is placed on the Unibus, the UBA must not attempt to map corresponding addresses to SBI address space. The software must set bits (30:26) of the control register, the map register disable bits, to indicate the size of the Unibus memory, where each increment is equal to 4K words of memory.

When Unibus memory is used, it must be placed at the bottom of the Unibus address space beginning at address 0. If the upper five bits of address received from the device initiating the DMA transfer represent a number that is less than the number placed in bit (30:26) of the control register, the Unibus memory is being addressed, and MAR H and the DMA ATT logic will be disabled. The invalid map register bit and the map register parity fail bit cannot be set if the Unibus address received falls within the range of the disabled map registers. Finally, if the UBA is already busy performing an interrupt or a data transfer on the Unibus, UAIA ATT DIS L will disable MAR H.

3.5.9 Power Fail and Initialization Logic

The power fail and initialization logic ensures that the UBA powers up and down in an orderly fashion and that the registers and other storage elements are brought to a known condition when the system is initialized.

3.5.9.1 System Power Up – When power is applied to the UBA, the power supply status signals DC LO and AC LO will remain asserted for at least 5 ms and 10 ms, respectively. These signals enable the clear signals that initialize registers and other storage elements on the UBA, and trigger the initialization sequences on the Unibus and on the UBA microsequencer. Figure 3-59 is a block diagram showing the clear signals and their destinations. Figure 3-60 shows the timing involved. Paragraph 3.3.3 describes the UBA initialization microroutine.

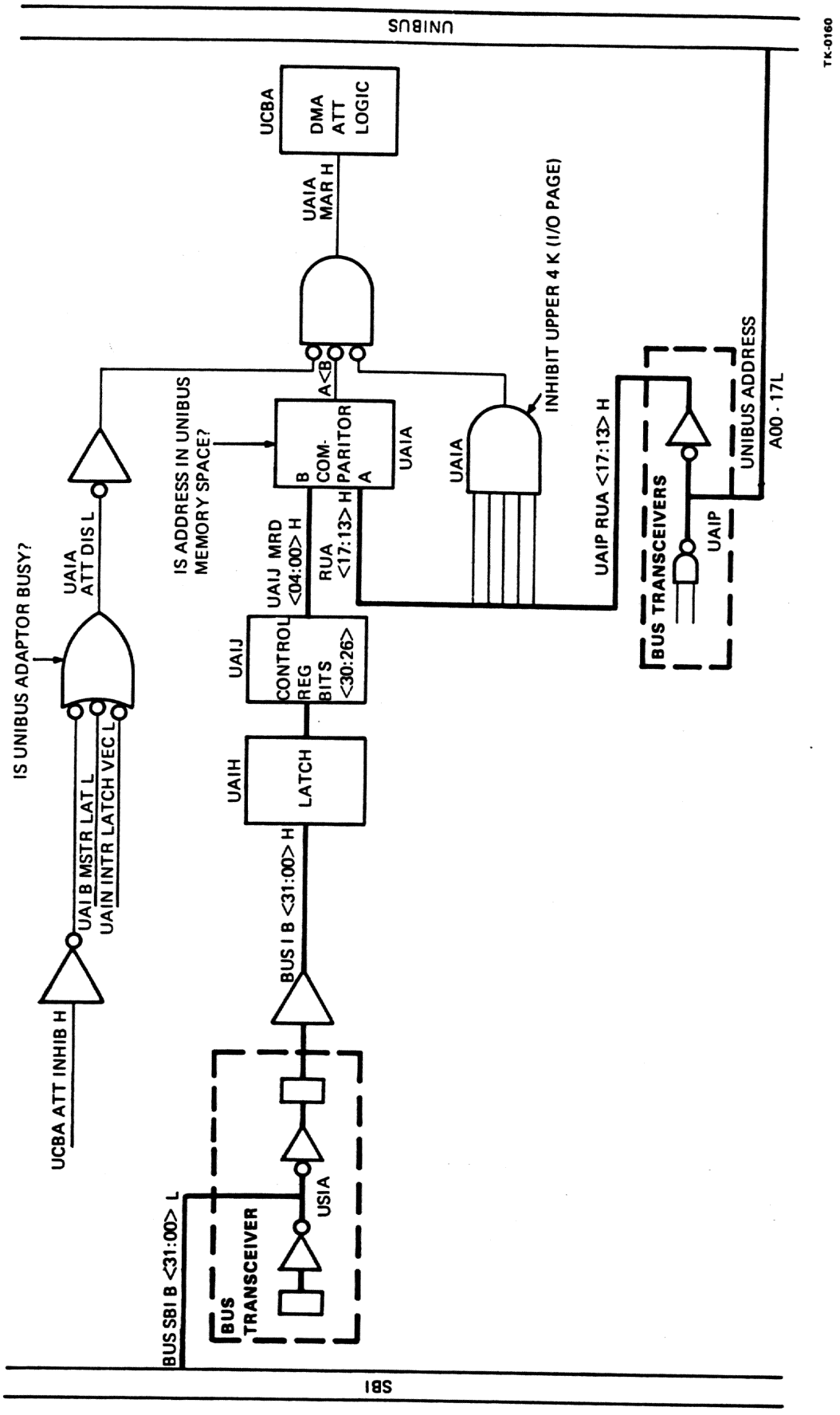


Figure 3-58 Map Address Range Logic, Block Diagram

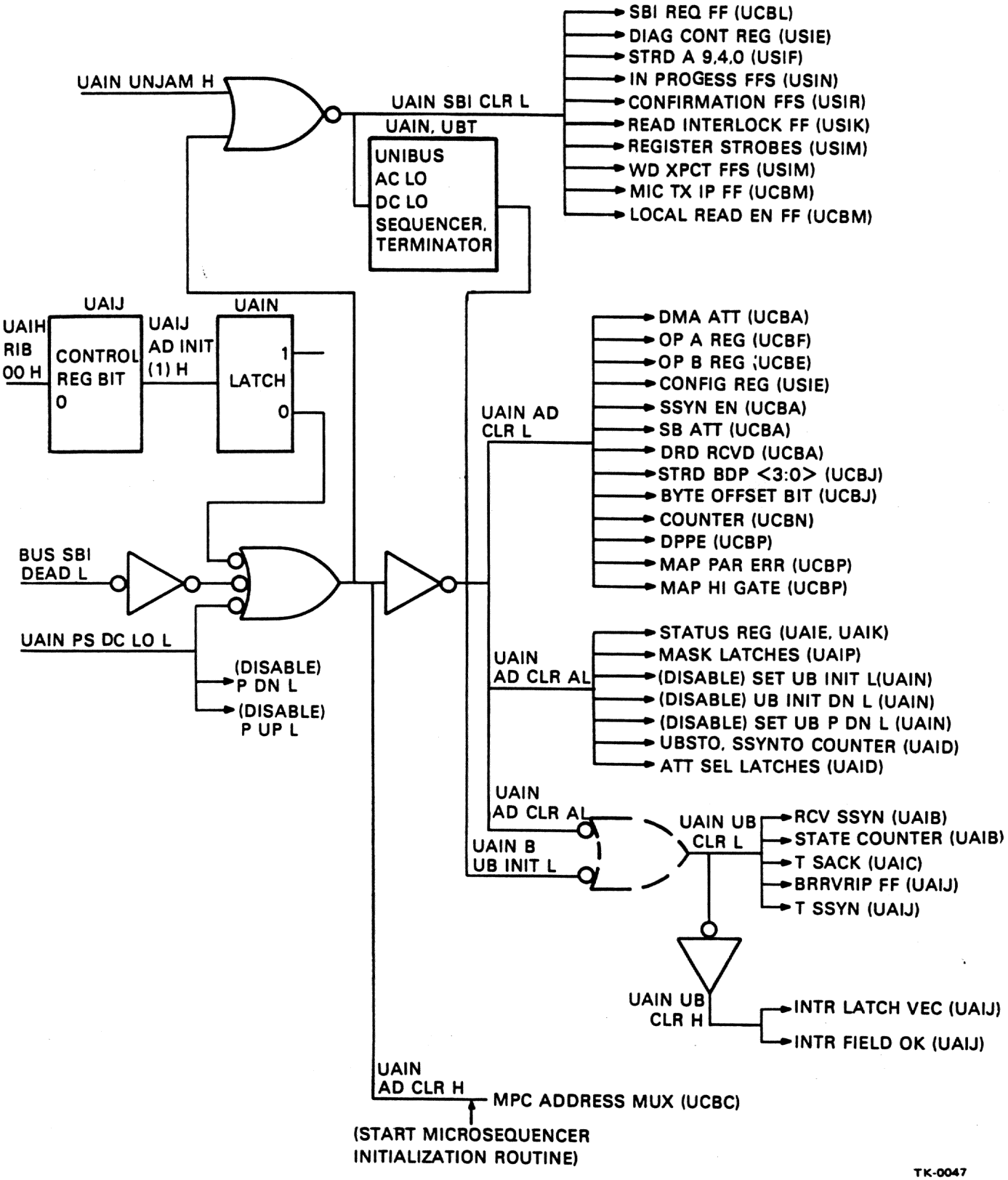


Figure 3-59 Power Fail and Initialization Logic Block Diagram

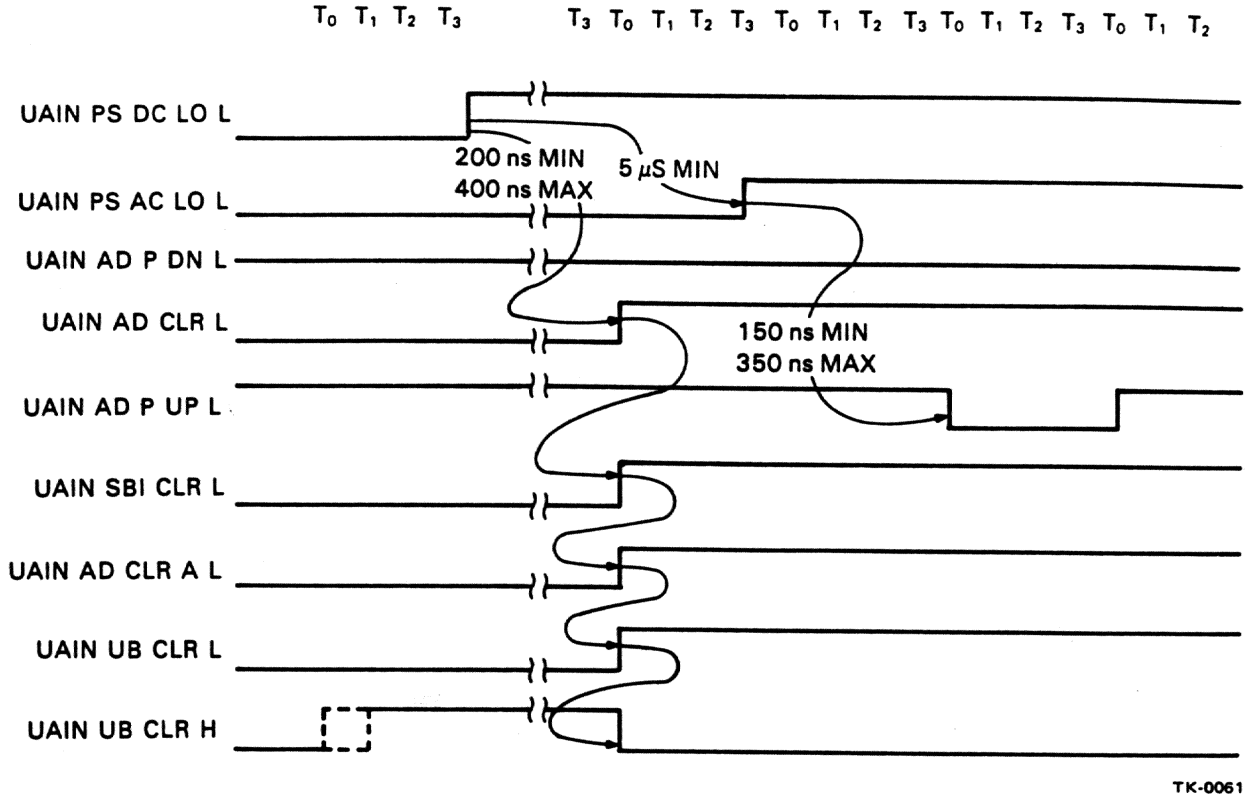


Figure 3-60 System Power Up Sequence, Timing Diagram

The UBA initialization sequence requires only 500 μs, while the Unibus initialization process requires 25 ms. Note, however, that the UBA initialization sequence cannot be completed until the Unibus, as well as the UBA, has power. At the completion of the sequence, the power up and initialization done bits of the configuration register will be set, causing the UBA to interrupt the VAX-11/780 CPU, via the request line assigned to the UBA.

3.5.9.2 System Power Down – When the VAX-11/780 system is powered down, or the UBA begins to lose power for some other reason, Supply AC LO H will be asserted by the power supply. This is the first indication of a power failure on the UBA. The UBA status sequencer, shown on sheet UAIN, then asserts UAIN AD P DN L, setting the Adaptor Power Down bit in the configuration register, and initiating an interrupt if enabled by bit 2 of the control register (CNFIE) (Figure 3-61).

At the same time, UAIN SET UB P DN L will set the Unibus power down bit of the configuration register.

A minimum of 5 ms later, if the power is still failing, the UBA power supply asserts SUPPLY DC LO H, enabling UAIN AD CLR A L and the other clear signals shown in Figures 3-59, and 3-62. The UBA registers and control logic are thus reset to a known condition before power is lost, and the microsequencer jumps to location 003(8), the first step of its initialization microroutine.

Whether or not the power supply asserts Supply DC LO H, the Unibus AC LO DC LO sequencer will assert BUS UB DC LO L 5 ms following Supply AC LO H. The terminator responds with BUS INIT L, triggering the Unibus initialization sequence. This resets the Unibus control logic, inhibits further activity on the Unibus, and sets the appropriate bits in the configuration register (Figure 3-63).

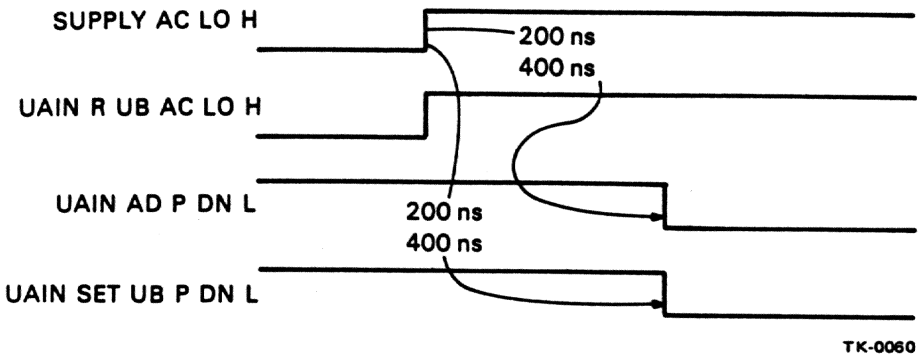


Figure 3-61 Beginning the System Power Down Sequence, Timing Diagram

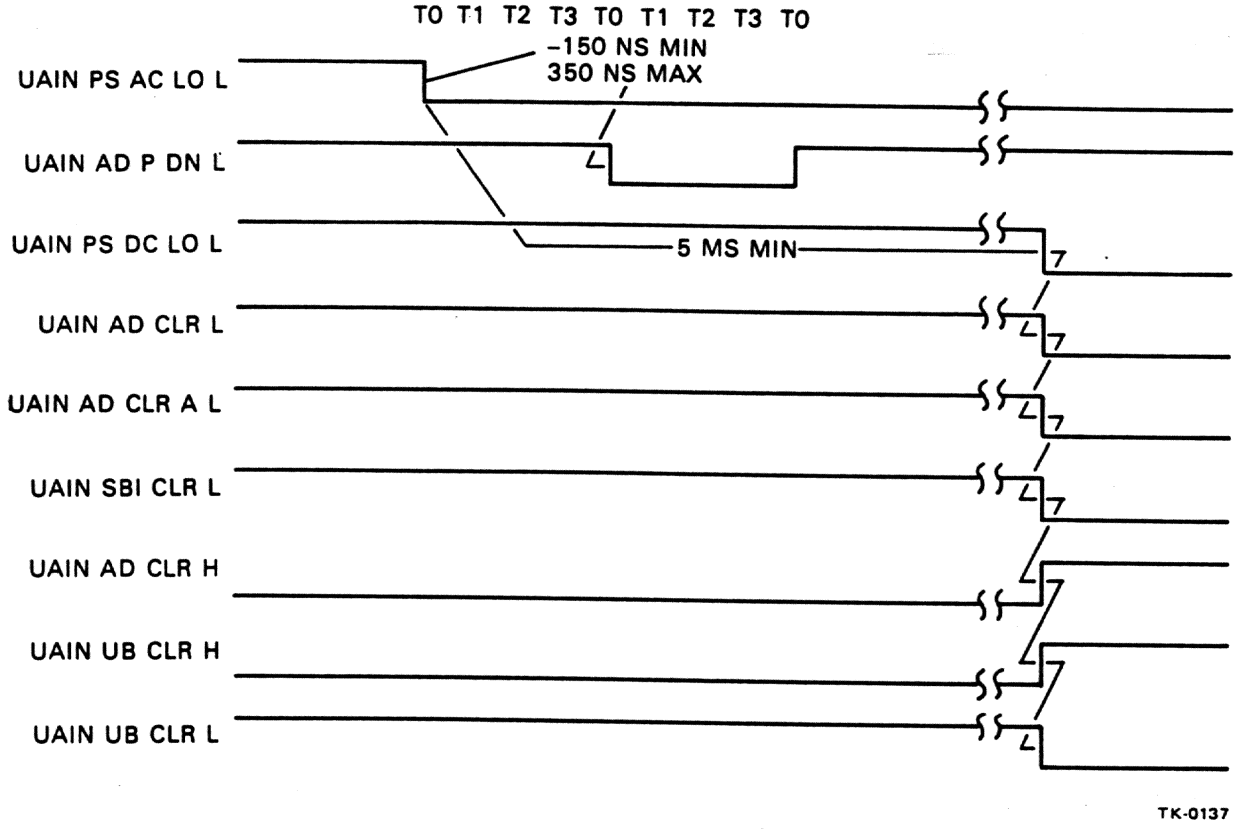


Figure 3-62 System Power Down Sequence, Timing Diagram

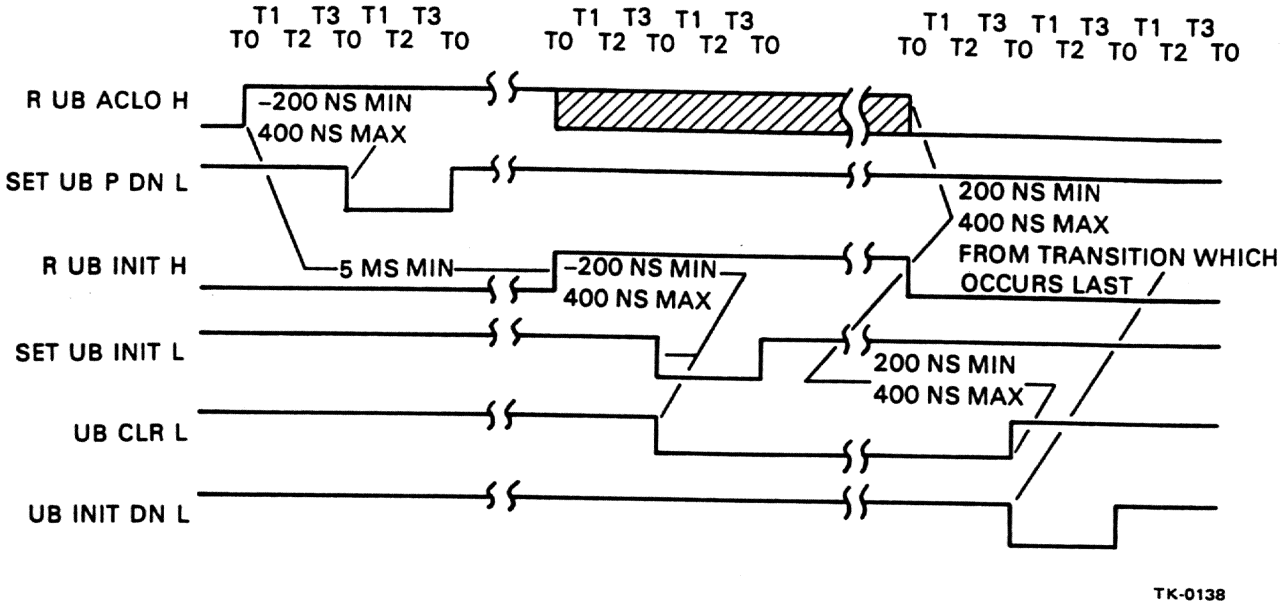


Figure 3-63 Unibus Status Sequencer, Timing Diagram

3.5.9.3 Unibus Power Fail – If a Unibus device asserts AC LO, the Unibus status sequencer on the UBA in turn sets the Unibus power down bit of the configuration register. Then, after a delay of 5 ms, the Unibus AC LO DC LO sequencer asserts BUS UB DC LO L on the Unibus. The terminator responds with BUS INIT L, resetting the Unibus control logic and setting the Unibus Init bit of the configuration register (Figure 3-63). The Unibus will remain in a powered down state until Unibus power has been restored, at which time the UBA will initiate a Unibus power up sequence and set the Unibus Init Done bit in the configuration register. The software can force the UBA to perform the Unibus power fail-initialization sequence by setting and then clearing bit 1 in the control register (UPF). UAIJ UPF L causes the UBA to assert AC LO on the Unibus. The UBA then responds as it would to a real Unibus power failure. Note that the initialization process will not be completed until UPF is reset, releasing AC LO.

UNJAM Function – The CPU (console) asserts UNJAM on the SBI to restore the system to a known (initialized) condition. UNJAM resets the configuration interrupt enable bit of the control register (CNFIE), asserts UAIN SBI CLR L (Figure 3-59), and triggers the Unibus initialization sequence previously described.

3.6 ACCESSING UNIBUS ADAPTOR REGISTERS

All of the major registers internal to the UBA are accessible to the software. The contents of each register can be read, but the failed Unibus address register (FUBAR), the failed map entry register (FMER), and the BR Receive Vector Registers (BRRVRs) cannot be written by the software, as explained in Paragraph 2.9.

3.6.1 Access to a Map Register

The software writes information into a map register with an SBI write function, using two bus cycles on the SBI. On the first cycle, the commander nexus asserts C/A, providing the address of the map register and the appropriate tag and function and mask codes on the SBI. When the UBA decodes the C/A, it recognizes the address as that of a map register. At this point, the attention of the UBA microsequencer is needed.

The 496(10) 32-bit map registers are implemented in a RAM in a 992(10) × 16 bit configuration. Nine address bits are used to select one of the map registers. When the software writes data to a map register these nine bits are channeled from the SBI through the register address latches [UMDA REG ADD (08:00)] and through a 2:1 multiplexer shown in Figure 3-64. Notice that bits (08:00) from the B field of the SBI form address bits (09:01) of the map registers. Map register address bit 0, generated by the microsequencer, is necessary to select the high or low 16-bit portion of the map word. Figure 3-65 shows the map register logic in relation to the UBA as a whole.

On the following SBI cycle the UBA latches the information to be written to the selected map register, storing it in the first location of the 4 × 32 bit data file shown on sheet UMDA of the print set. The microsequencer then gates the information to the map register, 16 bits at a time.

When the software initiates a read transfer to a map register, the UBA decodes the address and function portions of the C/A word, as on a write transfer to a map register. A microsequencer routine is then initiated. The high 16 bits of the map register addressed are latched with the enabling of UCBP MAP HI GATE H, as shown in Figure 3-64. The microroutine then enables the low 16 map register bits through two multiplexers to the IB bus, where they are asserted. At the same time, the microroutine gates the latched high 16 bits through two other multiplexers to the IB bus.

Once the data is available on the IB bus, the UBA arbitrates for control of the SBI to send the data.

3.6.2 Access to the Configuration and Diagnostic Control Registers (CNFGRs and DCRs)

The configuration register address constitutes the base of the UBA address space, as shown in Table 2-9. The addresses of the other internal registers are relative to that of the configuration register. The address offsets can also be derived from Table 2-9.

Notice that the data input signals to these two registers are fed directly from the B (31:00) SBI transceiver latches. Also, the outputs of these registers are multiplexed to feed the BUS TB (31:00) lines, which supply the data for transmission to the B field transceivers. These two registers, therefore, are accessible to the software with the least possible intervening logic.

On a write to CNFGR or DCR, the SBI address and function decoder logic asserts USIM CNFGR STRB EN L or USIM DCR STRB EN L as appropriate, latching the data in the register addressed from USIA RB (31:00) H.

On a read to CNFGR or DCR, the SBI address and function decoder logic asserts USIL USI RIP(J) H.

Two cycles later, UCBM LOCAL READ EN L enables UCBM SEND TR H, causing the UBA to arbitrate for the SBI. UCBM LOCAL READ EN H enables USIJ LOCAL READ L, enabling the data from the tristate latches and multiplexers for CNFGR or DCR onto the BUS TB lines, as shown in Figure 3-66. When USIC ARB OK L is subsequently asserted, it enables UCBL DTE L, gating the read data onto the SBI at the next occurrence of T0. Figure 3-67 shows the CNFGR and DCR logic in relation to the UBA as a whole.

3.6.3 Access to the Control and Status Registers

The control and status registers form a pair as shown in Figure 3-68. Their address offsets are 2 and 3, respectively. Figure 3-69 shows the relation of these registers to the UBA as a whole.

SBI address bits 1 and 0 discriminate between them. The data inputs to these registers are latched from the IB bus as UAIH RIB (30:26, 15:00) H. The outputs of the two registers are multiplexed, with UMDA REG ADD 0 H selecting one or the other for output to the IB lines. Note that the software may read or write bits in the control register, but that the status register bits are read and write 1 to clear, since the data from the SBI is fed to the K inputs of the register flip-flops. Microsequencer attention is required for a read transfer for either register, but not necessary for a write transfer.

3.6.4 Failed Map Entry Register (FMER)

In the course of a DMA transfer or purge operation, an error may occur that sets one of the following bits of the status register: IVMR, MRPF, DPPE, CXTMO, CXTER, RDS, and RDTO. In any such case, the signal UAIK LATCH FMER L is enabled, disabling the clock input to the FMER flip-flops, and locking the data present on the D inputs, as shown in Figure 3-70. Note that data inputs to the FMER are also the address inputs to the map register RAMs, so that the data latched is, in fact, the address of the map register selected for the current DMA transfer. When the software subsequently reads the FMER, a microroutine is invoked to complete the transfer. When the software then resets the appropriate error bits in the status register, the UAIK LATCH FMER L signal goes high, enabling the clock input to the FMER once more. Figure 3-71 shows the FMER on the UBA block diagram.

3.6.5 Failed Unibus Address Register (FUBAR)

The FUBAR contains the upper 16 bits of the Unibus address translated from an SBI address during a software initiated transfer to Unibus address space. In a transfer of this type the UBA will attempt to become the Unibus master. If the Unibus Select Time Out or the Unibus Slave Sync Time Out bits of the status register set during the transfer, the signal UAIK GATE FUBAR L will be enabled, disabling the clock inputs to the FUBAR flip-flops, as shown in Figure 3-72. Since the data inputs to the FUBAR are the same as the inputs to the address transceivers for Unibus lines UA (17:02), the software can determine the failing Unibus address on a subsequent read transfer to the FUBAR. Figure 3-73 shows the FUBAR on the UBA block diagram.

3.6.6 Data Path Registers (DPRs)

The 16 DPRs store information relating to the current state and function of the corresponding buffered data paths. Although they are accessed as 32-bit registers, the storage devices that make up the registers are not configured in a typical manner. Figure 3-74 shows the DPRs in block diagram form.

All of the DPR bits can be read, but only bits 31 and 30 can be written by the software. An explanation of the process involved in reading a DPR follows.

When the VAX-11/780 CPU asserts C/A in order to read one of the data path registers, the address decoder logic alerts the microsequencer (SB ATT L) and the microsequencer, in turn, jumps to the starting address of the SB2 microroutine (MF 22). The microsequencer causes the state and mask bits (the upper two bytes of the DPR) to be selected by a 2:1 multiplexer and asserted on BUS AD (15:00) as shown in Figure 3-74. These bits are then selected by the 4:1 data path multiplexer and gated to the shifter, where they are rotated two byte positions. The state and mask bits are then loaded into the lower two byte positions of longword address three of the selected buffered data path RAM. Bits (15:00) of this data path register, the upper 16 Unibus address bits, will have been automatically loaded into the upper two byte positions of the selected buffered data path RAM at the time of the last DMA transfer through that data path. The entire contents of the DPR addressed are at this point stored in the upper word of the corresponding buffered data path RAM, with the upper and lower portions of the register reversed. The microsequencer, therefore, must rotate the contents of the register two byte positions in the shifter in order to produce the proper bit configuration, as shown in Figure 3-75.

The output of the shifter is then selected by a 2:1 multiplexer and asserted on the internal bus [IB (31:00)] for gating to the SBI as read data.

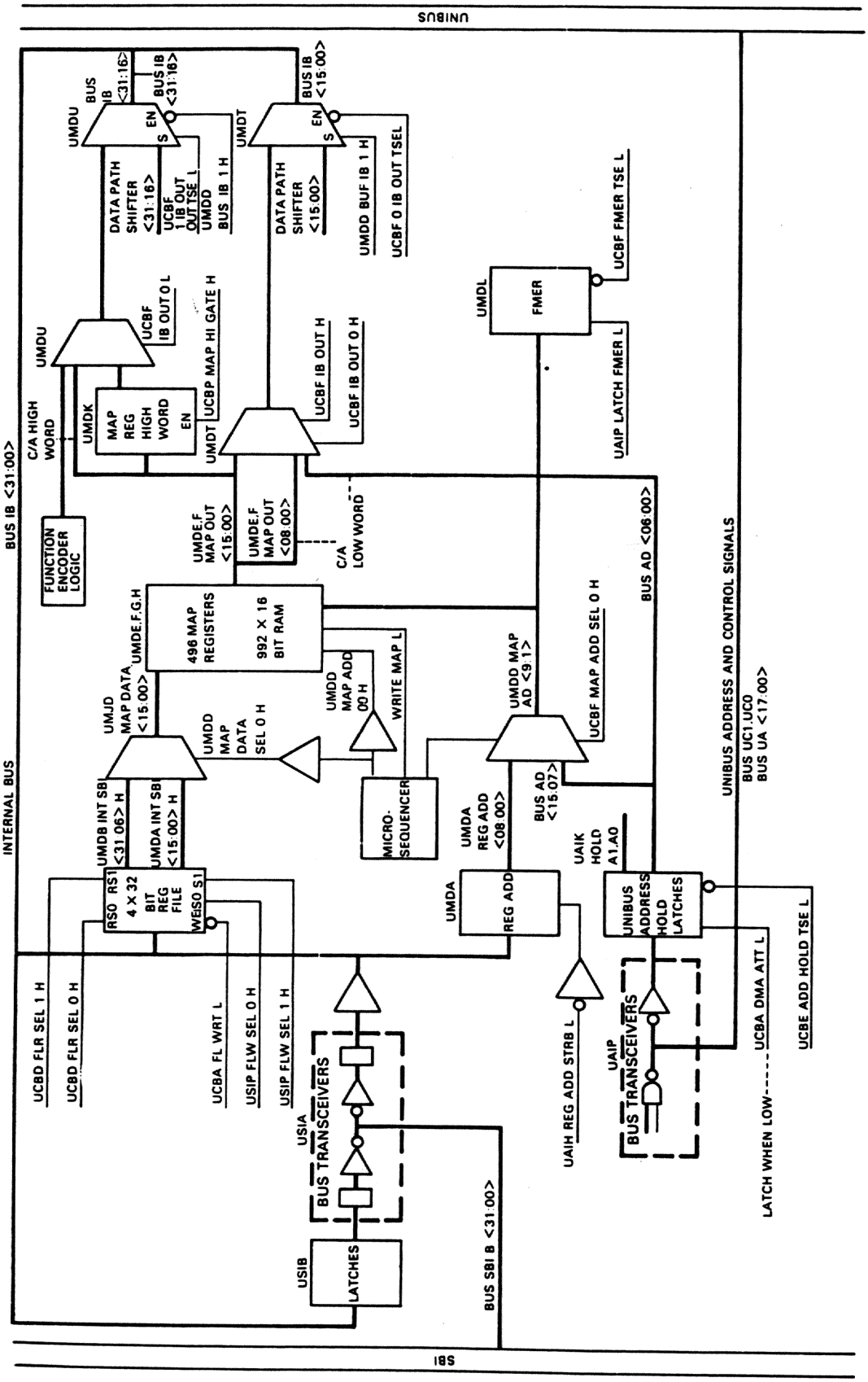


Figure 3-64 Map Registers, Block Diagram

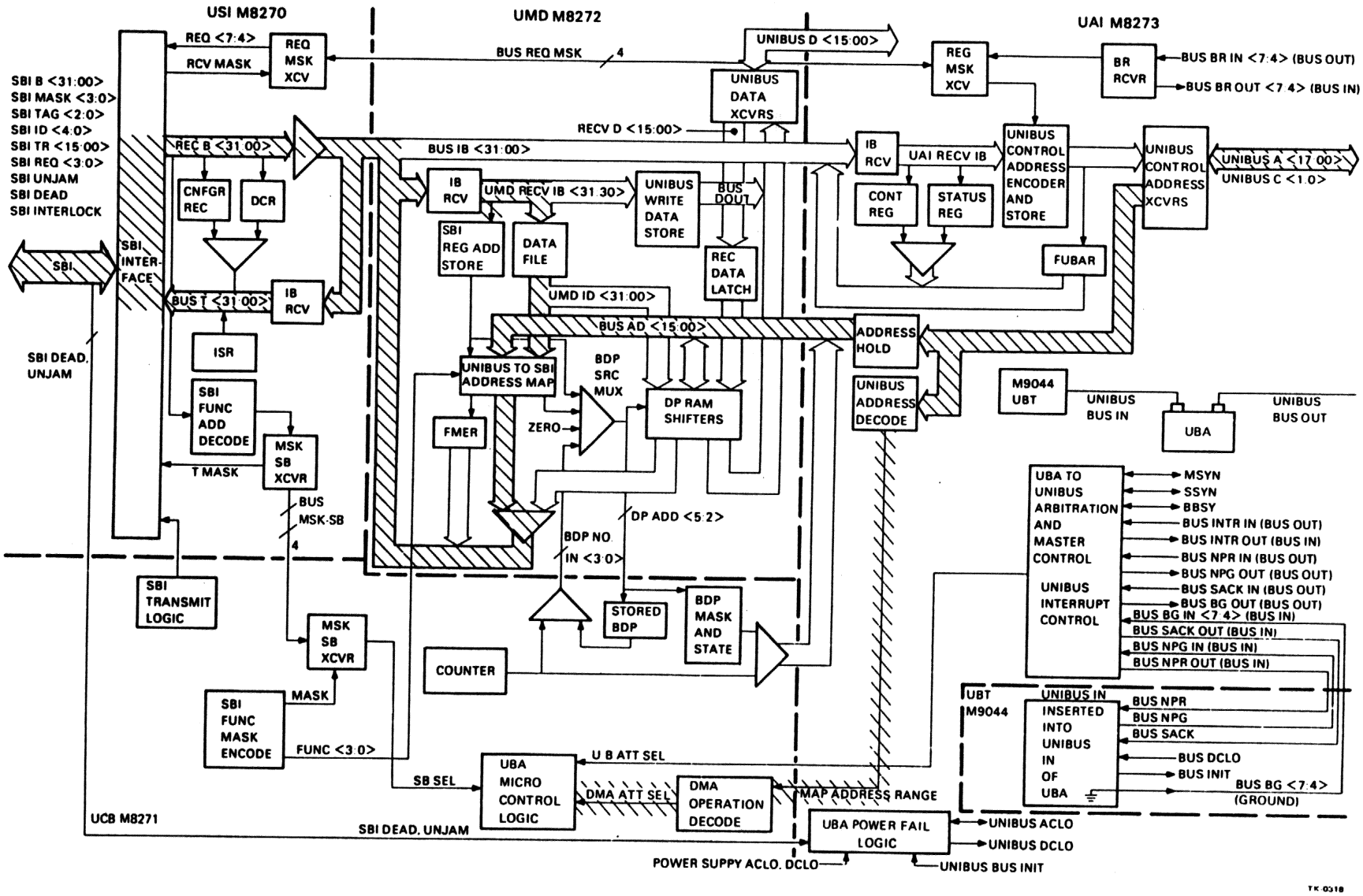


Figure 3-65 Map Register Logic/UBA Block Diagram

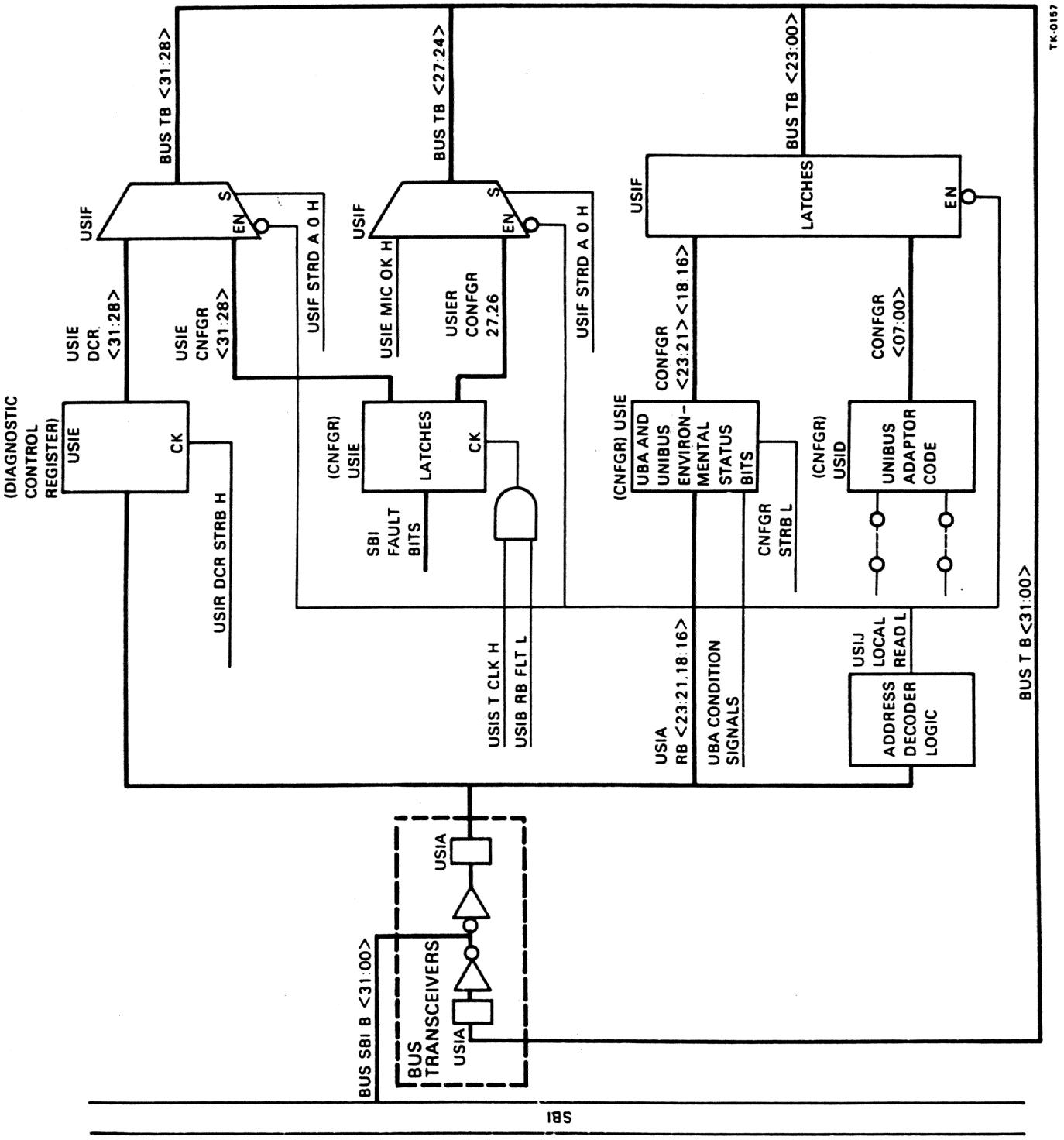


Figure 3-66 Configuration and Diagnostic Control Registers Block Diagram

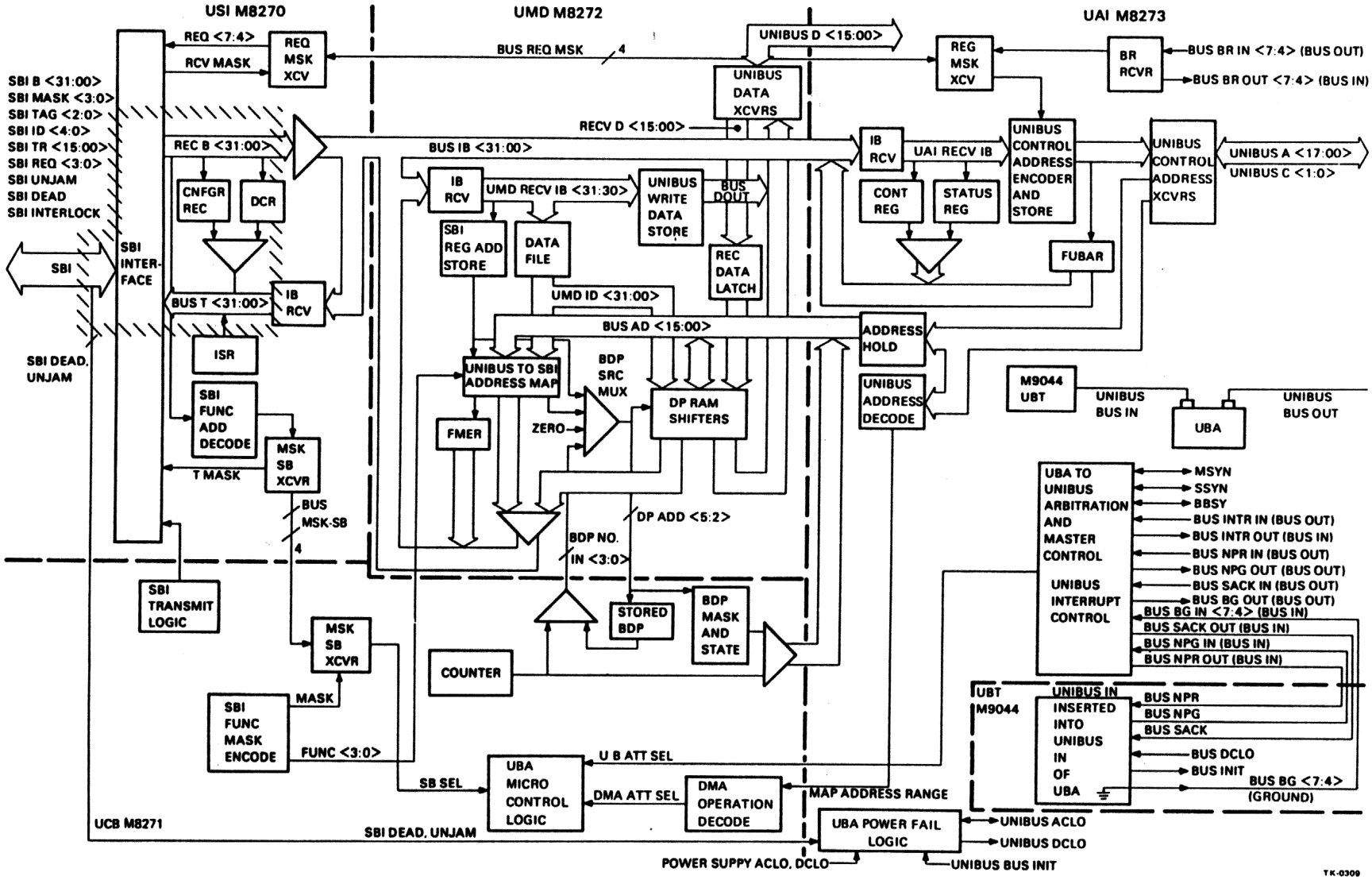


Figure 3-67 CNFGR and DCR Logic/UBA Block Diagram

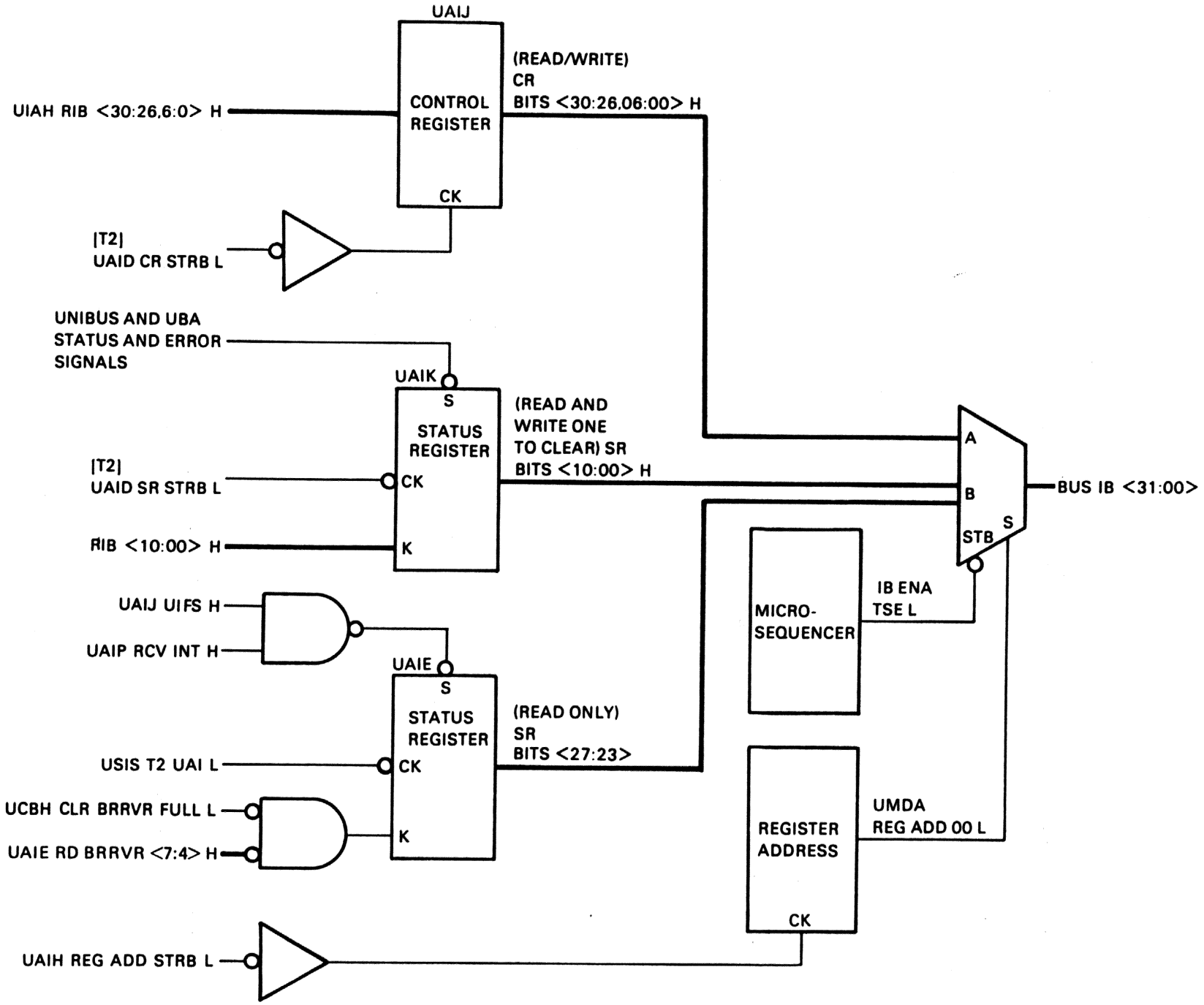


Figure 3-68 Control and Status Registers, Block Diagram

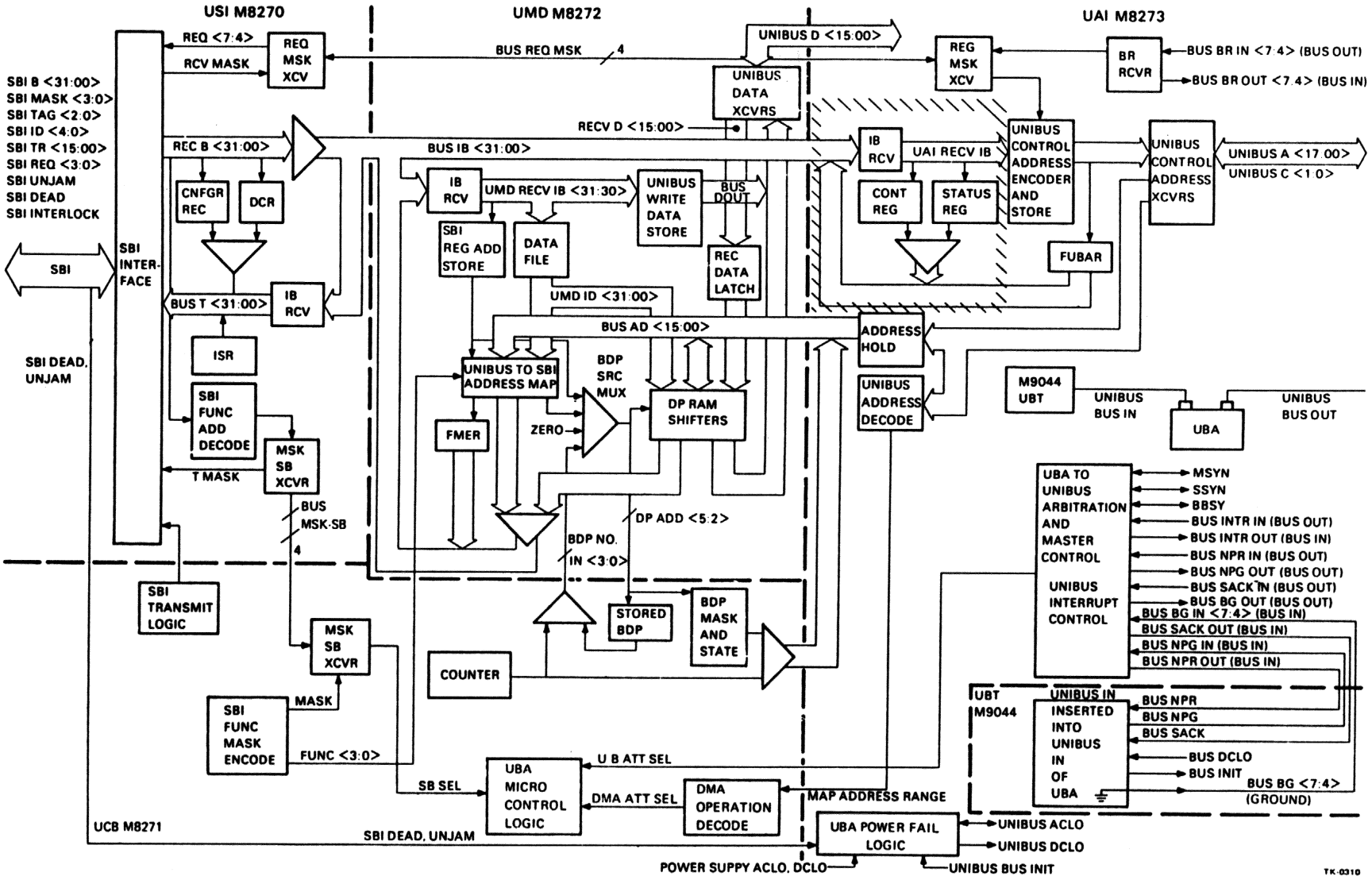


Figure 3-69 CR and SR/UBA Block Diagram

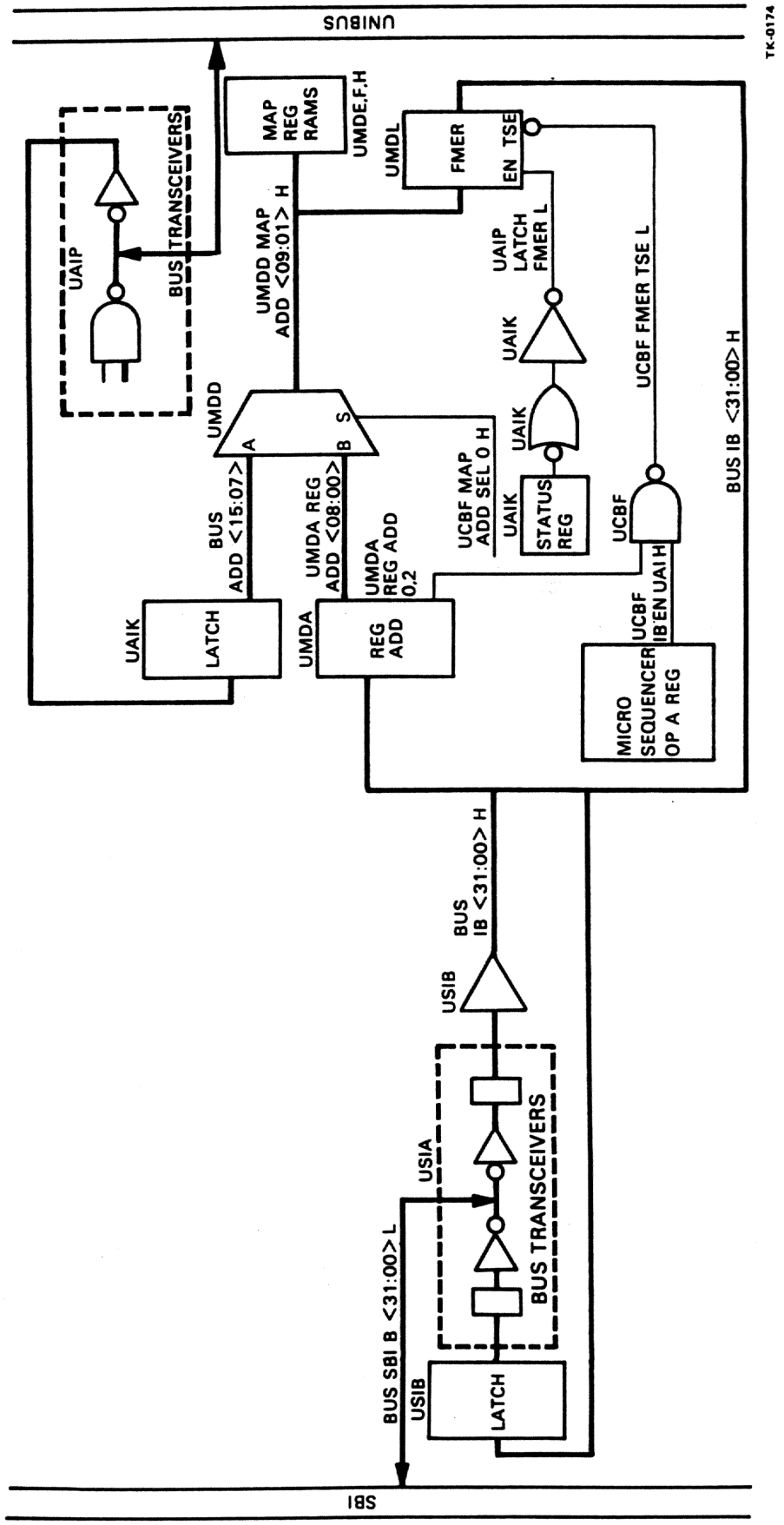


Figure 3-70 FMER and Associated Logic, Block Diagram

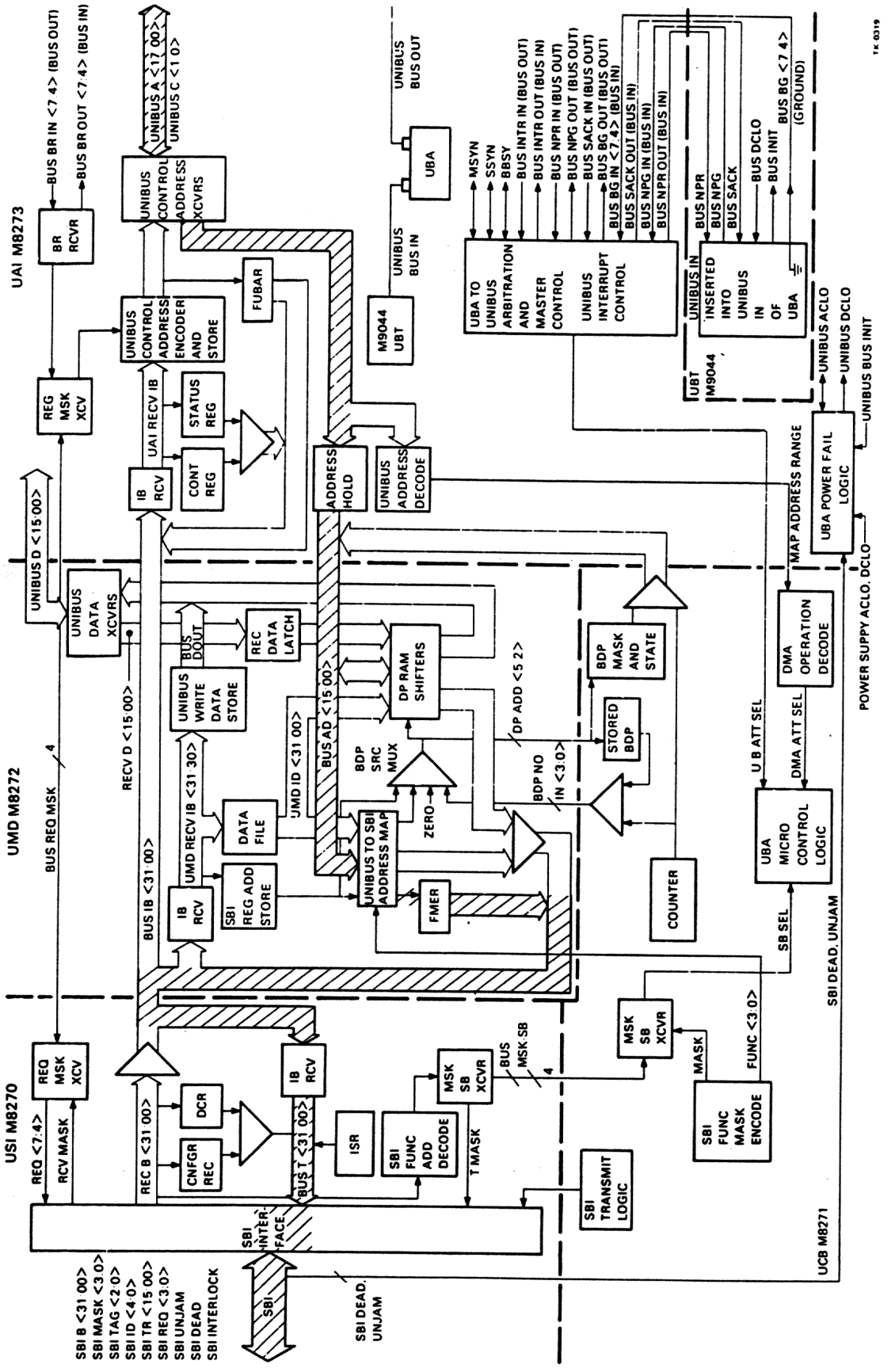


Figure 3-71 FMER/UBA Block Diagram

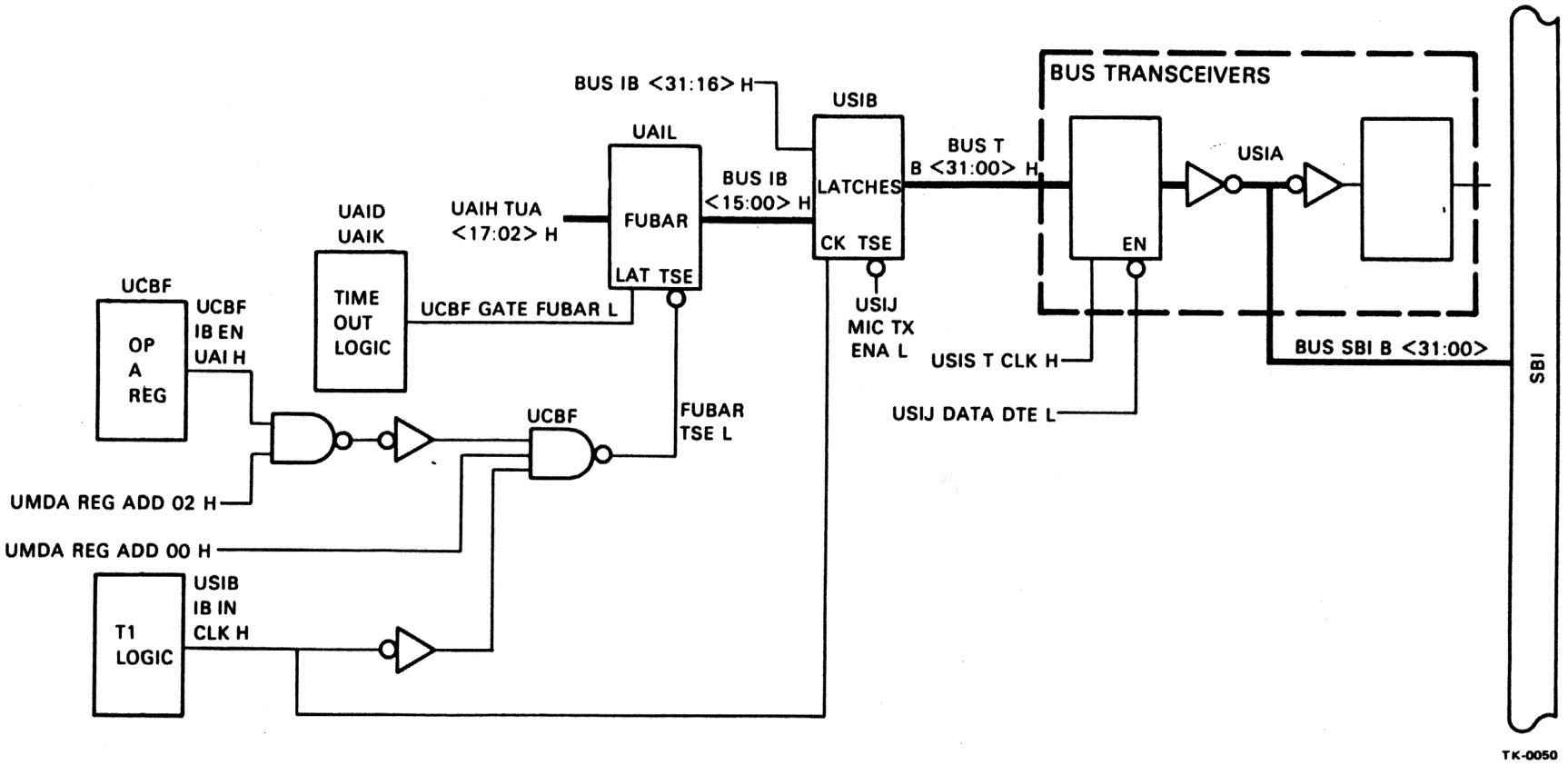


Figure 3-72 Failed Unibus Address Register and Associated Logic, Block Diagram

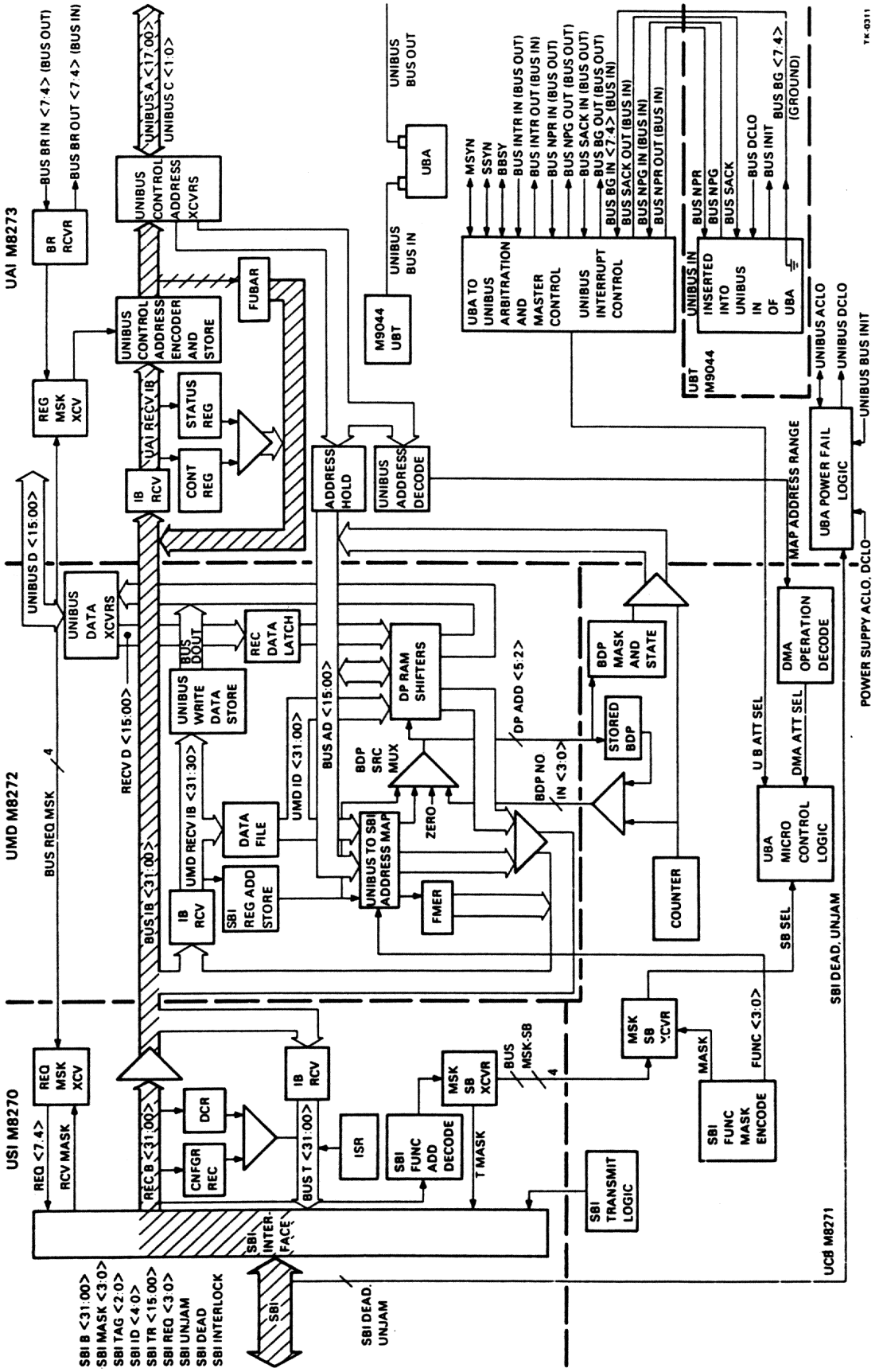


Figure 3-73 FUBAR/UBA Block Diagram

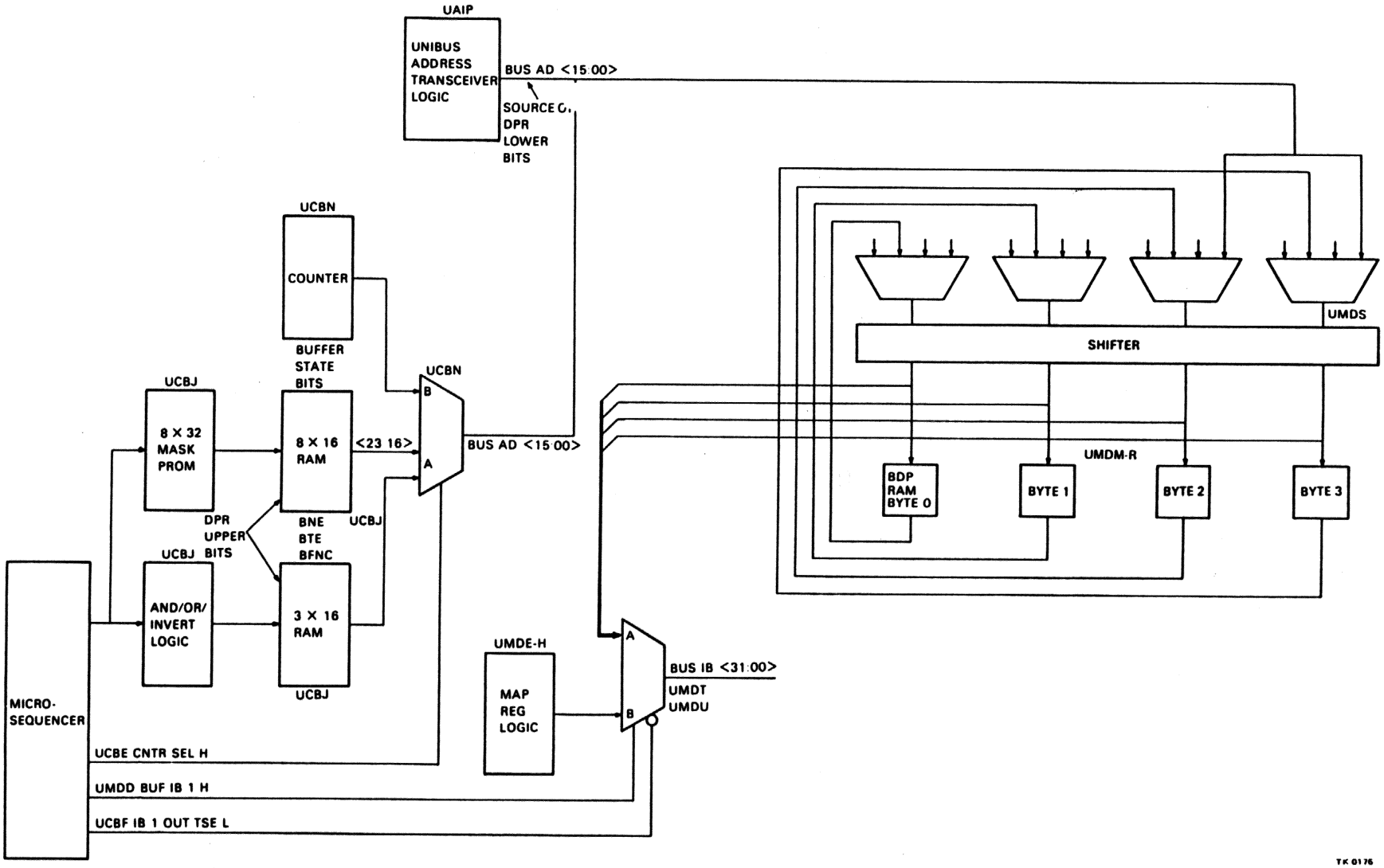


Figure 3-74 Data Path Registers, Block Diagram

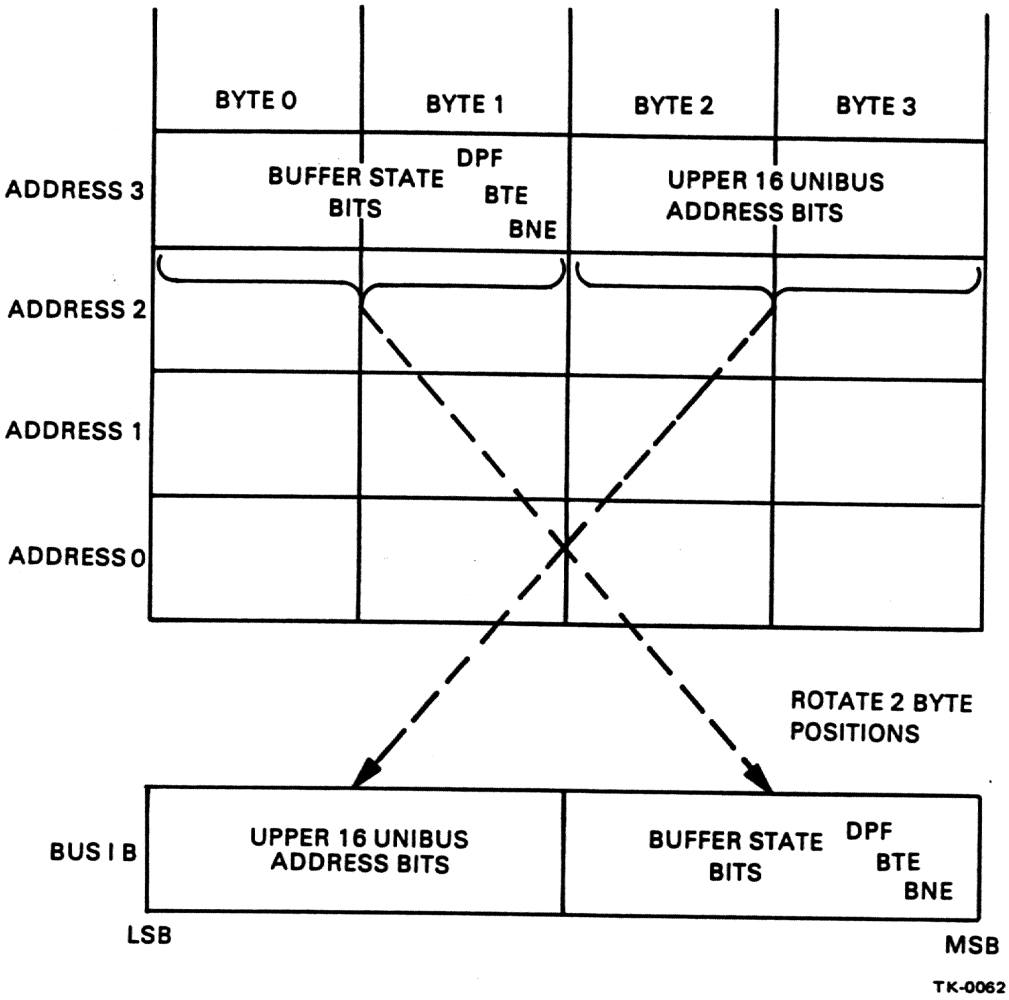


Figure 3-75 Byte Manipulation on a Read Data Path Register Transfer

3.6.7 UBA Generated Interrupts

The UBA will initiate an interrupt to the VAX-11/780 CPU when an event or condition on the UBA sets one of certain bits in the configuration register or the status register. Nine of the status register bits and six of the configuration register bits will cause the UBA to interrupt the CPU if the appropriate interrupt enable bits (02:00) on the control register are set. Figure 3-76 shows how the SBI request signal is generated.

A step-by-step summary of the interrupt dialogue follows. Notice that the paragraph numbers in the summary correspond to numbers on the block diagram shown in Figure 3-77.

1. The UBA asserts one of the four SBI request signals [SBI REQ (7:4)], beginning the interrupt process. Some time later, the VAX-11/780 CPU responds with an interrupt summary read command to identify the requesting device.

2. The UBA decodes the command, encodes its TR number as the UBA request sublevel, and asserts this as an interrupt summary response on the B field of the SBI. Having thus obtained the request level and the request sublevel, the software branches to the UBA service routine.

3. The UBA service routine reads the BRRVR corresponding to the level of the interrupt. Since bit 31 of the BRRVR will be set in this case, the software reads a negative value and then branches to a routine that reads the configuration register and the status register to determine the service required.

An explanation of the process by which the hardware sets BRRVR bit 31 follows. When the in-progress logic enables USIN BRRVRIP L in response to a read transfer command to the BRRVR, the UAIE RDBRRVR (7:4) H signal, corresponding to the BRRVR addressed, will be enabled. If the configuration or status register error bit causing the interrupt is still set and the appropriate enabling bit in the control register is also set, the jumper selected UBA interrupt level signal is ANDed with the corresponding UAIE RDBRRVR signal, enabling UAIF ADP REQ ACTIVE H, as shown in Figure 3-78.

Since the BR logic will not be active (unless a Unibus interrupt has also been started) UAIF BR ACTIVE L will be false, enabling the BRRVRIP L signal to set the UAIJ SND RAM L flip-flop (no vector from the Unibus is required). SND RAM L, in turn, calls the microsequencer attention, causing the contents of the BRRVR addressed to be sent on the SBI as read data.

Note that the SBI request line will remain asserted until all of the pertinent status and configuration register bits have been cleared by the software.

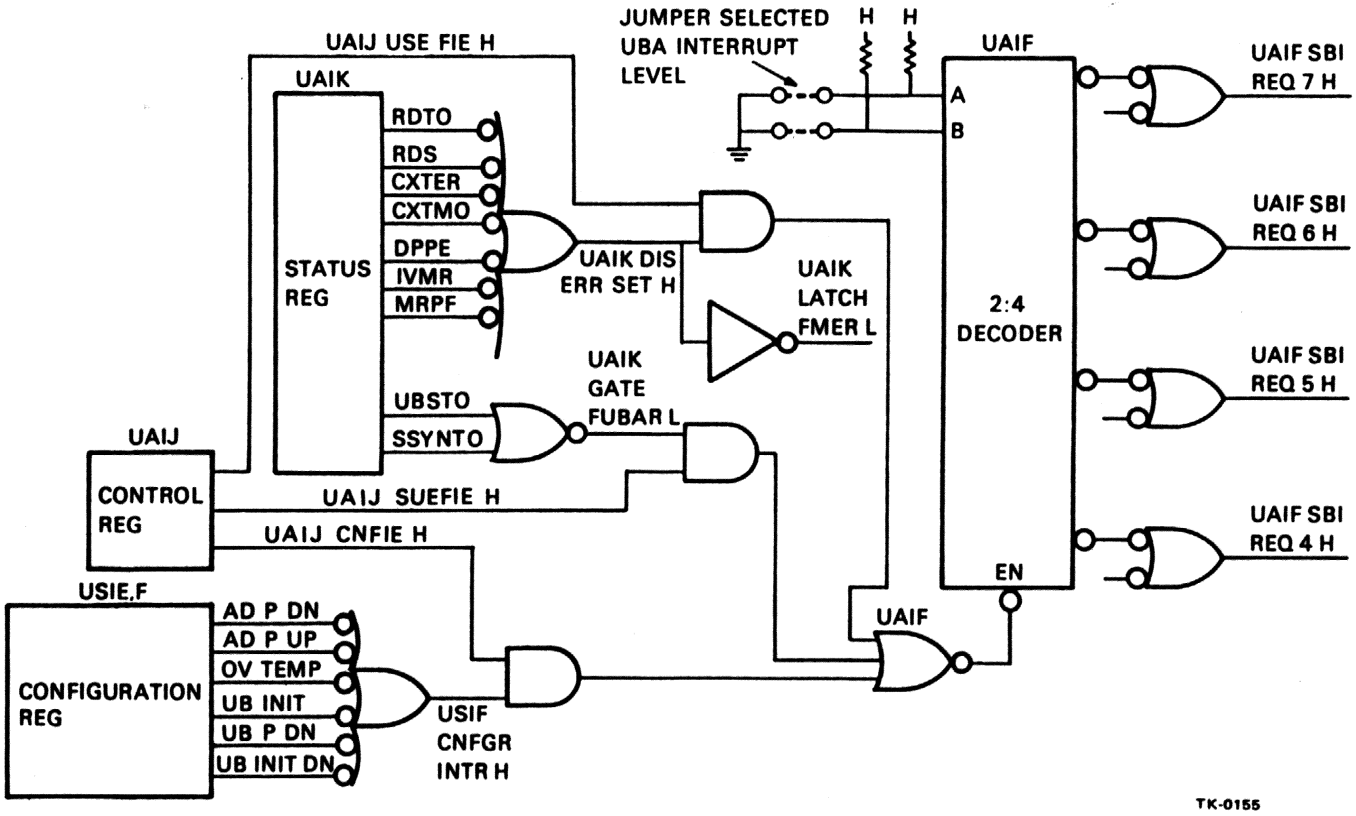


Figure 3-76 SBI Request (7:4) Generation on a UBA Interrupt, Block Diagram

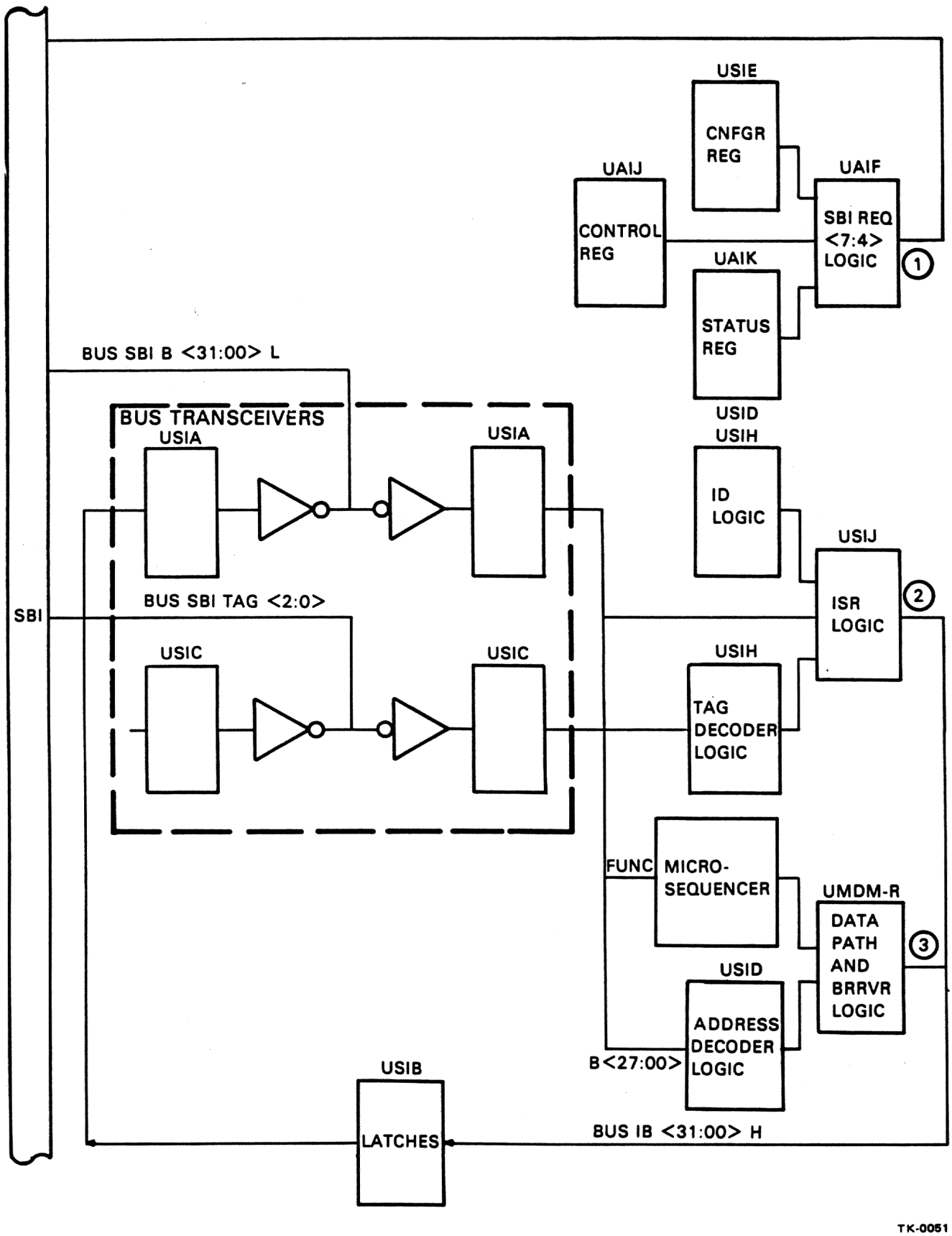


Figure 3-77 UBA Generated Interrupt Logic, Block Diagram

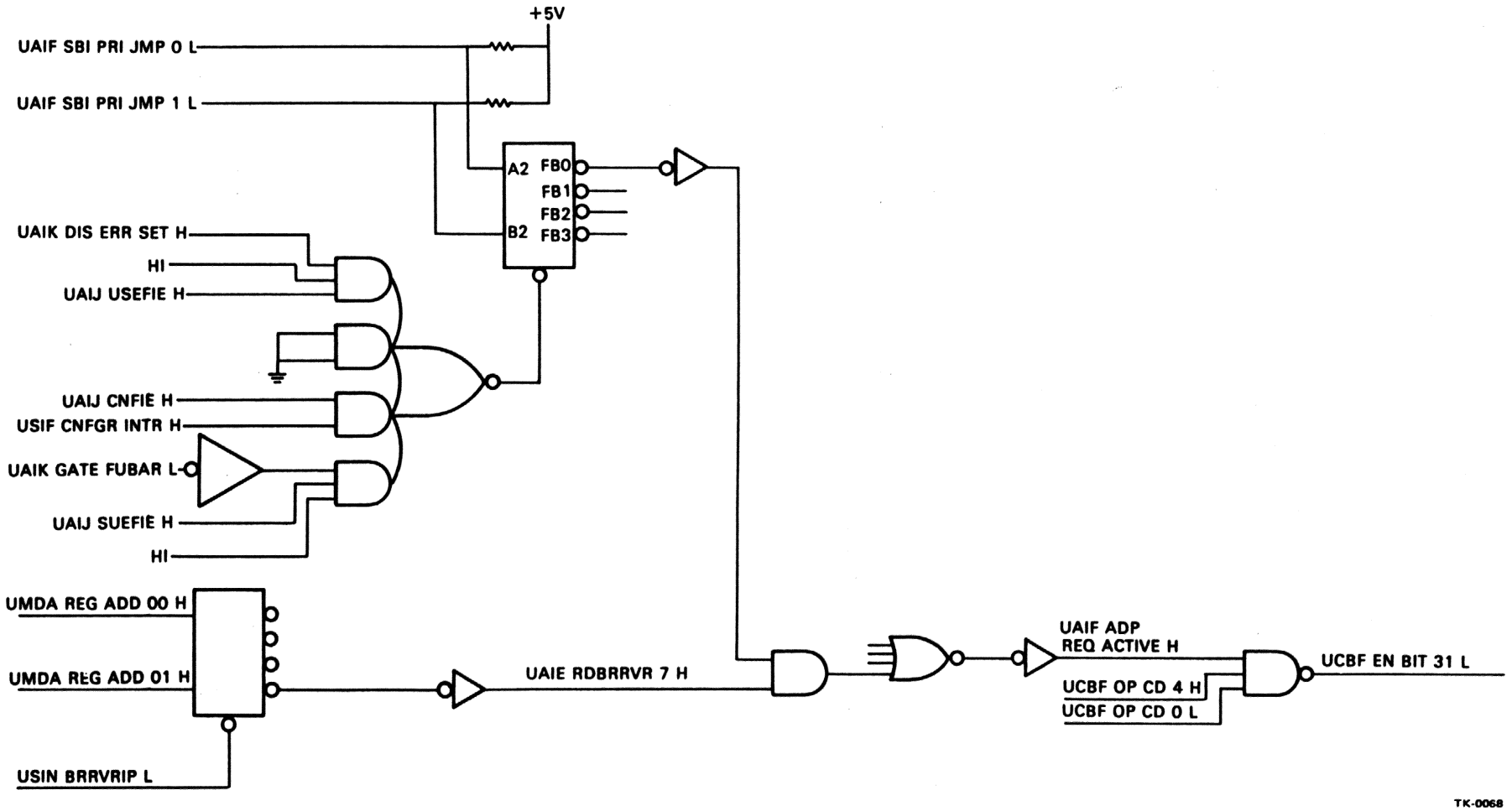


Figure 3-78 Development of BRRVR Bit 31

APPENDIX A
LONGWORD ALIGNED 32-BIT RANDOM ACCESS MODE

The longword aligned 32-bit random access mode of the UBA provides the following capabilities for nonprocessor request (NPR) transfers (direct memory access) from Unibus devices.

- Transfer of random 32-bit quantities to or from SBI memory.
- SBI read, write, and read-modify-write transfer on 32 bits at a time.
- Elimination of the purge operation.
- Elimination of the prefetch operation.

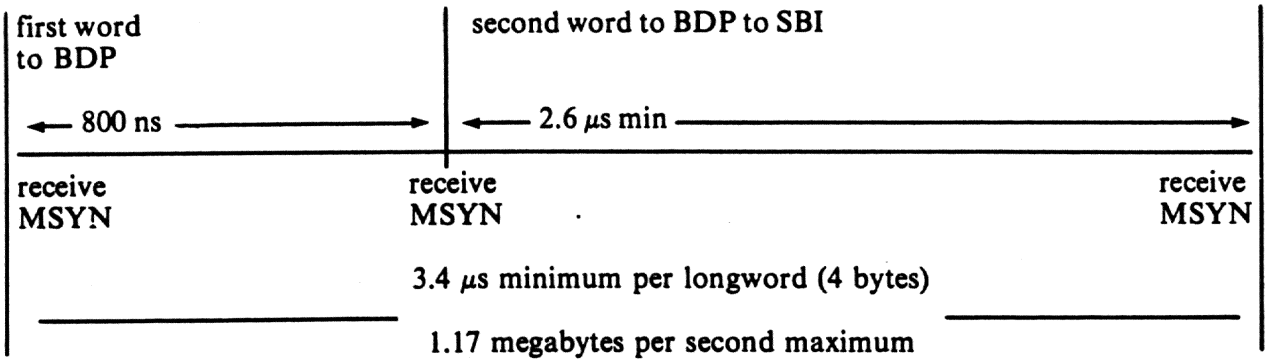
Five restrictions are placed on the use of this mode.

- Transfers must use a buffered data path.
- Transfers cannot be a Unibus DATIP operation.
- Data must be longword aligned.
- Transfers must operate on data in the proper order (see following text).
- The byte offset mode must not be used.

The software must set the longword access enable (LWAE) bit (26) of the map register corresponding to the Unibus transfer in order to select the longword aligned 32-bit random access mode.

In this mode, the Unibus device must first operate on the low-order word of the longword and then on the high-order word. An operation is considered to be a read from memory (DATI), a write to memory (DATO), or a read-write (DATI-DATO). The Unibus DATIP function is not valid for transfers using buffered data paths, and any device performing a DATIP through a buffered data path will receive an SSYN timeout (NXM).

This mode enables random (non-sequential) access to longword aligned 32-bit quantities, because with bit 26 of the corresponding map register set, the buffered data path does not perform the prefetch operation. Furthermore, the mode eliminates the need for the purge operation at the completion of the transfer, provided that the Unibus device operates on both words of the longword and operates on them in order (i.e., low word first, then high word). The maximum throughput of data in this mode is approximately 1.17 megabytes per second.



The operation of the UBA for the longword aligned 32-bit random access mode is determined by the function (DATI, DATO, and DATOB) and address (A1, A0) received from the Unibus and by the state of the buffer not empty (BNE) bit of the data path register corresponding to the BDP being used for this operation.

- BNE set = buffer not empty.
- BNE clear = buffer empty.

The following statements summarize the operation of the UBA for the longword aligned 32-bit random access mode of operation.

DATI Functions

1. SBI reads will occur when a DATI operation is received and the BDP is empty (BNE = 0).

2. The BNE bit will be set in response to a successful SBI read generated by a DATI operation to the low-order word (A1 = 0). Longword data from memory is stored in the buffered data path.

3. The BNE bit will be cleared by a DATI operation to the high-order word (A1 = 1).

4. If the BNE bit is set, data from the buffered data path will be returned to the Unibus device.

DATO/DATOB Functions

1. The BNE bit will be set by a DATO or DATOB operation. The data from the Unibus device will be stored in the BDP and the byte mask bits will be set within the data path register to indicate the bytes or words that have been written by the Unibus device.

2. SBI writes will occur when a DATO operation occurs to the high-order word or a DATOB operation occurs to the high-order byte. The bytes or words that were written (i.e., those for which the byte mask bits are set) are written into main memory.

3. The BNE bit will be cleared after an SBI write operation.

Table A-1 shows the UBA operations performed for each Unibus access as a function of the BNE bit, the received Unibus function, and the address, with the LWAE bit of the corresponding map register set.

Table A-1 UBA Functions in the Longword Aligned 32-Bit Random Access Mode

Present BNE State	Function	A1,A0	UBA Operations	Next BNE State
0	DATI	0 X	SBI read, return low word, store data	1
0	DATI	1 X	SBI read, return high word	0
1	DATI	0 X	Return low word	1
1	DATI	1 X	Return high word	0
0	DATO	0 X	Store low word,	1
0	DATO	1 X	Store high word, SBI write	0
1	DATO	0 X	Store low word,	1
1	DATO	1 X	Store high word, SBI write	0
0	DATOB	0 0	Store byte 0,	1
0	DATOB	0 1	Store byte 1,	1
0	DATOB	1 0	Store byte 2,	1
0	DATOB	1 1	Store byte 3, SBI write	0
1	DATOB	0 0	Store byte 0,	1
1	DATOB	0 1	Store byte 1,	1
1	DATOB	1 0	Store byte 2,	1
1	DATOB	1 1	Store byte 3, SBI write	0
X	DATIP	X X	UBA does not respond (NXM to Unibus device)	No change

Note: X = Don't care.

Bit 26 of the map register has been changed from a reserved bit to the longword access enable bit (LWAE). When this bit is set and a buffered data path is selected, the longword aligned 32-bit random access mode is enabled. LWAE is a read-write bit and is cleared on UBA initialization.

Programming the UBA for the longword aligned 32-bit random access mode requires loading the map registers to be used with the following data.

Bit 31	MRV	Map register valid, must be set.
Bit 25	LWAE	Longword access enable must be set. This bit is ignored during direct data path transfer.
Bit 25	BO	Byte offset, must be zero.
Bits 24–21	DPDB	Data path designator bits must indicate a buffered data path. (1–15). The LWAE bit is ignored when DPDB = 0 (direct data path).
Bits 20–00	PFN	Page frame number, SBI page address.

The following Unibus sequences are allowed for this mode of operation.

		A1,A0	
1.	DATI	0 0	SBI read – Low word returned to Unibus device.
	DATI	1 0	High word from BDP is returned to Unibus device.
2.	DATO	0 0	Low word is written to BDP.
	DATO	1 0	SBI write – High word is written to BDP; then low word and high word are transferred to memory.
3.	DATOB	0 0	
	DATOB	0 1	
	DATOB	1 0	
	DATOB	1 1	SBI write
4.	DATI	0 0	SBI read – Low word returned to Unibus device, and both words are stored in BDP.
	DATO	0 0	Low word of BDP is written by Unibus device.
	DATI	1 0	High word from BDP is returned to Unibus device.
	DATO	1 0	SBI write – High word of BDP is written by Unibus device and modified long word is returned to the memory.