EK-ODX20-TM-PRE

DX20 PROGRAMMED DEVICE ADAPTER

CHAPTER 4 — OPERATION/PROGRAMMING

CHAPTER 1
INTRODUCTION

1.1 PURPOSE AND USE

The DX20 Programmed Device Adapter (PDA) is a high performance I/O processor with a high data transfer rate used in the DECSYSTEM-20 to transfer data, in one of nine formats, between the host memory and high speed devices.

1.2 GENERAL DESCRIPTION

Data is transferred between the host and PDA over the system Massbus independent of the DECSYSTEM-20 central processor (CPU) and is controlled by a microcontroller located in the PDA. Figure 1-1 shows the DX20 system configuration. Once the PDA is initialized by an instruction from the CPU, the microcontroller (microprocessor and microstore) takes over the CPU function; operating the DX20 as an I/O processor by executing microcode instructions retained in its writable control store. The microinstructions are loaded from the host computer into the microcontroller store via the DX20 diagnostic link (bus). This bus is also used for transferring information, read from the microcontroller's diagnostic registers, to the host.

1.3 FEATURES

The following features are incorporated in the DX20.

1. Microinstruction parity detection
2. Nine data format modes
3. Executes self-diagnostics when idle.

1.4 CHARACTERISTICS

The DX20 PDA is composed of wire-wrap circuit boards plugged into logic panels which are mounted in either a standard H950-A 19-inch equipment rack or a H9502 corporate cabinet. Input/output cable connections are provided by pairs of jacks for the Massbus cables, channel bus cables, and control unit cables. Cables are attached to the jacks with quick-latch connectors. Power in the DX20 is supplied by: one 861-C power control; one H7420-A (115 Vac) power supply; and four power regulators [three H744 (+5 V), and one H745 (-15 V)].

The DX20 is constructed in a modular manner consisting of four major sections; a microcontroller, computer interface, high speed data path, and a device interface. DX20 circuit boards plug into a system unit-type backplane connecting the boards together via the microbus. The microcontroller consists of two hex boards; one contains the microprocessor, and the other its control store memory. One hex and one quad board plus bus interface boards (where required) compose the computer interface section. The high speed data path, for handling data packing modes, includes a hex board (formatter) and a quad board (data buffer). A single hex board is utilized for the channel (or device) interface section.

Physical and operating characteristics of the DX20 are contained

in Chapter 2.

Data formats supported by DX20 hardware are described in Chapter 5.

1.5 OPTIONS

The DX20 programmed device adapter is packaged in one of two types of cabinets; the standard 19-inch H950 cabinet, or the H9502 corporate cabinet. The following option designations denote the various cabinet configurations.

DX20-AA	H950	115 Vac, 60 Hz
DX20-AB	H950	230 Vac, 50 Hz
DX20-AC	H9502	115 Vac, 60 Hz
DX20-AD	H9502	230 VAC, 50 HZ

1.6 REFERENCES

The following documents contain information that supplements the contents of this manual.

1. RH20 Massbus Controller Unit Description (EK-RH20-UD-001)

2. See Table 3-1 for a list of manuals supplied with the equipment.

Figure 1-1 DX20 System Configuration

CHAPTER 2
SITE PREPARATION AND PLANNING

DESCRIPTION

The DX20 Programmed Device Adapter is used in conjunction with the TX02 Magnetic Tape Controller and one or more TU70/71/72 tape drives. This component part of the TX70/71/72 Magnetic Tape Subsystem will be mounted in the appropriate cabinet dependent on whether it is used in a 1090 or a 2040/2050. Table 2-1 lists the DX20 characteristics and Figure 2-1 shows the service area requirements.

Table 2-1 DX20 Characteristics

Figure 2-1 DX20 Service Area Requirements

CHAPTER 3
INSTALLATION

3.1 GENERAL

The DX20 Programmed Device Adapter can be a subunit of any DECsystem utilizing a Massbus controller, therefore the DX20 installation composes only a part of the entire system installation. This chapter describes the installation procedures related to the positioning, powering up, and initial checkout of the DX20. Installation procedures for other units in the system involved are contained in the appropriate service manuals for those units.

3.2 UNPACKING AND INSPECTION

The DX20 is bolted to a pallet when it is delivered to a DECsystem site. Prior to removal, the shipping container and skid should be checked for external damage or abusive handling. After the shipping container has been removed a thorough visual inspection of the DX20 should be performed to ensure no physical damage was incurred during shipment.

Cabinet doors and panels, connectors, the cooling assembly, pluggable module connections, and cable connections should be thoroughly checked for dents, broken fasteners, looseness, broken connectors, etc. Any damage or excessive looseness should be carefully recorded and reported on suitable forms to the appropriate Field Service Department of Digital Equipment Corporation.

After inspecting the DX20 cabinet, ensure that the additional equipment listed in Table 3-1 has been received and is in good working condition. Any damage should be similarly recorded and reported.

The physical layout of the DX20 is shown in Figure 3-1 and module utilization and busing is shown in Figure 3-2.

3.3 INSTALLATION

If no damage or only minor damage (not serious enough to prevent installation) is found during unpacking and inspection, proceed as follows:

NOTE

The following procedure assumes that the DX20 is being installed in an operating, powered-up DECSYSTEM-20 computer installation.

1. For initial positioning, unbolt the DX20 from the pallet and wheel it (on its casters) into position.

2. Using the leveling legs provided at the bottom of the unit, adjust the cabinet for proper height and alignment with the adjacent cabinet. Cabinet dimensions and clearances required after the cabinet has been positioned and leveled, are shown in Figure 2-1.

3. Connect the power control input power cable to a 120 Vac, 50-60 Hz or 240 Vac 50-60 Hz power source. Energize the DX20 cabinet by placing the power control circuit breaker to ON (Figure 3-3).

4. Set the REMOTE ON/OFF LOCAL ON toggle switch to LOCAL ON.

5. Determine that the cabinet blowers are operating properly and the +5 V, -5 V, and -15 V operating voltages are present on the backplane power connection pins.

6. Install Massbus and IBM channel bus terminator quick-latch connectors if required (see Figure 3-4).

NOTE

Ensure that the control unit and the and/or tape drives have been properly installed and are operating before proceeding with the next step.

7. Connect the Massbus between the DX20 and DECSYSTEM-20 RH20 and connect the channel bus cable between the DX20 and control unit.

Table 3-1 Equipment Supplied

Quantity	Item	Document, Type, or Part No.
Software		
1	DX20 Programmed Device Adapter Diagnostic (RH20) MAINDEC-10-DFDXC	
1	DX20-V100 Magnetic Tape Subsystem Diagnostic (RH20) MAINDEC-10-DFDXD	
1	DX20-V100 Magnetic Tape Subsystem Reliability Diagnostic (RH20) MAINDEC-10-DFDXE	
1	SPAR Diagnostic Magnetic Tape (Note 1)	50055013
Prints		
1	DX20 Print Set	MP00432
1	TX02 Logic Print Set (Volumes I, II, and III) serialized (Note 1)	
1	TU70 Logic Print Set (serialized) (Notes 1, 2, and 3)	
1	TU72 Logic Print Set (serialized) (Notes 1, 2, and 3)	
Manuals		
1	DX20 Programmed Device Adapter Maintenance Manual (Note 5)	EK-0DX20-TM-001
1	TX02 Maintenance Manual (Note 1) includes:	
	a. Theory of Operation Manual	9149
	b. Field Engineering Maintenance Manual	9204
	c. Illustrated Parts Catalog	9152
	d. Installation Manual	9057
1	TX02 P/N Compatibility Listing 3800 IV (Note 1)	9058
1	SPAR User's Guide (Volumes I, II) (Note 1)	9245

1	TU70 Maintenance Manual (Notes 1, 2) includes:	
	a. Field Engineering Maintenance Manual	9043
	b. Illustrated Parts Catalog	9014
1	TU70 P/N Compatibility Listing 34CC (Notes 1, 2)	9153
1	TU72 Maintenance Manual (Notes 1, 2) includes:	
	a. Field Engineering Maintenance Manual	9094
	b. Illustrated Parts Catalog	9063
1	TU72 P/N Compatibility Listing 36XX EVE (Notes 1, 2)	9267

Wirelists

1	DX20 Backplane - Section I WRP SORT	K-WL-DX20-0-WL
1	DX20 Backplane - Section II WRP SORT	K-WL-DX20-1-WL
1	M8601 - WRP SORT	M8601-1-WL
1	M8602 - WRP SORT	M8602-0-WL
1	M8603 - WRP SORT	M8603-0-WL
1	M8604 - WRP SORT	M8604-0-WL
1	M8605 - WRP SORT	M8605-0-WL
1	M8606 - WRP SORT	M8606-0-WL
1	M8607 - WRP SORT	M8607-0-WL

Hardware

1	Channel bus cable	70-10078-XX
1	Massbus cable	BC06-S
1	Massbus terminator	7009938
1	Channel bus terminator	H866-A
1	Ground cable	1210757
1	Remote turn-on cable	7008288

1	TU7X power cable (Note 3)	17-00040-XX
1	TU7X signal cable (Note 3)	17-00041-XX
1	E.P.O plug cable	70-104083-XX
1	Tag terminator	12-12172
1	Bus terminator	12-12171
1	TX02 kick plate assembly kit (Note 1)	
1	TU7X kick plate assembly (Notes 1, 3)	

Accessories

1	Blank magnetic tape (1 per TU7X drive)	18-9543
1	Cleaning kit	TUC01
1	Can tape cleaner	29-15199
1	Package Q-tips	N/A
1	Spare cab filter	

NOTES

1. **Supplied by vendor (STC)**
2. **If TU7X is part of subsystem**
3. **One per unit**
4. **Specify length for part numbers containing XX**
5. **Microfiche only**

Figure 3-1 DX20 Physical Layout

Figure 3-2 Module Utilization and Busing Diagram

Figure 3-3 861C Power Control Panel

Figure 3-4 DX20 Cable Connections

4.1 FUNCTIONAL TESTING

DX20-V100 magtape subsystem fault detection is performed automatically using diagnostic programs. These programs test the following three major operational areas.

1. DX20 Programmed Device Adapter. This test runs independent of the tape subsystem and consists of two parts; tests which are run from the host KL10, and tests which are run from the DX20 microprocessor. The following functional areas are tested by this diagnostic.

   Massbus interface
   data path interface
   channel bus interface
   DX20 microprocessor
   DX20 microprocessor instructions
   DX20 microprocessor control storage and working memory
   DX20 microbus interface

2. DX20-V100 Subsystem. This program tests the channel bus connections, the DX20/tape control unit interface, and the DX20-V100 operational microcode. All microcode operations are tested for proper execution.

3. DECsystem-10(-20)/DX20-V100 Subsystem. This diagnostic program exercises the DX20-V100 Subsystem under maximum dynamic loading conditions to ensure reliable system operation.

Use of the diagnostic programs, including the associated loading procedures, is described in the writeups for each diagnostic program. Identification of these programs is as follows:

1. DX20 Programmed Device Adapter Diagnostic (RH20) (MAINDEC-10-DFDXC).

2. DX20-V100 Magnetic Tape Subsystem Diagnostic (RH20) (MAINDEC-10-DFDXD).

3. DX20-V100 Magnetic Tape Subsystem Reliability Diagnostic (RH20) (MAINDEC-10-DFDXE).

4.2 OPERATING PROCEDURES

4.2.1 Bootstrap Loading

There are two bootstrap loaders which load the microcode into the DX20, namely, the exec (mode) bootstrap and the user (mode) bootstrap. The exec bootstrap loader is called automatically by KLINIT which is the task that brings up the TOPS20 monitor. More details about KLINIT can be found in the DECsystem-20 Operator's Guide.

The exec bootstrap can reside on either the floppy disk or the front-end area of the dual-ported system disk pack. After KLINIT prints out the message on the operator console:
KLI--BOOTSTRAP LOADED AND STARTED
The exec bootstrap starts to load the microcode into the DX20, verifies the microcode just loaded, then starts the DX20. If any error occurs while loading the microcode, the exec bootstrap will print out an error message which tells what caused the error, and tries to load the microcode again if possible. When it succeeds in loading the microcode, the exec bootstrap will print out the following message on the operator console:
DX20 MICROCODE VERSION 0(52) LOADED, VERIFIED AND STARTED
The user bootstrap loader is used while the monitor is running. The purpose is to be able to reload the microcode after the DX20 is powered down for maintenance without reloading the monitor. However, during maintenance of the DX20 no attempt should be made to use the tape drives connected to this DX20. Otherwise, the tape drives may not be available for use without reloading the monitor.

To run the user bootstrap loader, the user must have privileges as an operator, a wheel, or maintenance. After the user bootstrap has started, it prints out the message:
FILE:
and waits for the user to type the filename of the microcode to be loaded into the DX20. The user must specify which DX20 to load if there is more than one DX20 on the system. To select the DX20 to load, the user can give the switch, /D:ab right after the filename, where a, b, are octal digits, and a= the RH20 number and b= the drive number of the DX20. However, if there is only one DX20 on the system, the user can simply type (CR) carriage return after typing in the filename. The user bootstrap will look for the DX20 and type out the RH20 number and the drive number of the DX20 selected. Then, the user bootstrap loads the microcode as specified by the given filename, verifies the microcode loaded, and prints out the following messages if no error occurred:

```
FILE; DXMCA.ADX
DX20 SELECTED:  RH20-2  DRV=1

     MICROCODE LOADED
     MICROCODE VERIFIED
     MICROCODE VERSION 0 (52)
     DX20 STARTED ___
```

The DX20 loaded and started is the first drive connected to the second RH20 on the system.

4.2.2 Microcode Operational Description

The DX20 hardware functionally consists of four parts:

1. The microprocessor has 2K 16-bit words of writable control, 1K 8-bit bytes of data storage, and 9 8-bit bytes of register storage. The control store is not writable by the microprocessor. During normal operation, the microprocessor has primary control over operation of the other three parts.

2. The Massbus interface contains the registers which are accessible from the host for communication with the microprocessor. It contains the necessary control logic to bootstrap and control the microprocessor for diagnostic purposes. It also contains logic for transferring data over the Massbus.

3. The channel bus interface contains the logic to control the IBM bus protocol. Under control of the microprocessor it performs device selection and deselection. It also contains logic to handle high speed data transmission over the channel bus.

4. The data path contains logic to transmit data at high speeds between the Massbus interface and the channel bus interface. The data rates through the data path are far greater than is possible through the microprocessor, although the microprocessor also has the capability of moving data through the DX20. The data path also contains logic to pack and unpack data in the various formats required for magnetic tape handling on the DECsystem10/20.

The only direct control the host system (DECsystem10/20) has over the DX20 is through the Massbus registers contained in the Massbus interface. Some of the information in these registers is available to the microprocessor. This section describes those bits that will normally be used by the host for control of the DX20.

An operation with the Massbus device DX20 is initiated as follows:

Load control information into the appropriate registers of the Massbus device describing information such as (in the case of magnetic tape) record length, data mode, error recovery to be tried, etc.

Load the Massbus controller with channel control information if a data transfer operation is desired. This is a pointer to a channel command list containing pointers to data and a word count for each pointer.

If a data transfer operation is to be performed, a register in the RH20 is loaded with the appropriate command, which the RH20 then loads into register 0 of the device. If an immediate operation is to be performed, register 0 of the device is loaded directly by the program. The controller

looks at its register 0 to determine that the operation is to start, and what kind of data transfer is to occur (read, write, verify, control.)

Note that if the controller is busy with a data transfer operation, it is still possible to initiate a nondata type operation. The registers of the device can be loaded with information, including the command to be performed, providing the device is not already busy.

When the controller determines that it is done with a data transfer, it waits for the device to agree that it is time to stop, and then interrupts the host indicating that the data transfer operation is complete. A nondata operation is performed locally by the device, and the controller is not involved with the operation except to initiate it.

When the device needs some special handling by the host, it raises an attention signal to the Massbus controller. This will usually cause an interrupt in the host. This attention signal can be caused either by the completion of a nondata transfer type operation, or by the device deciding that it knows something the host should be made aware of, like a tape drive just came on-line. When the attention has been recognized by the host, it can write the Massbus registers in the device to clear the condition.

4.2.2.1 Microcontroller Jump Instruction Description - The microcontroller can execute three basic jump instructions. However, these basic jump instructions are flexible and can accomplish a wide variety of tasks as follows:

Jump Immediate

conditional jumps - on any one of seven conditions

jump with an offset (indexing)

jump on condition with indexing

In addition to these, the microprogram can return from a subroutine by popping a PC value previously stored on its stack.

bits 15-13 = 4 octal- jump op code indicating a jump immediate within a page, Bit 7=0 of this JUMP instruction specify location.

bits 15-13 = 6 octal- jump op code indicating an address source of the working memory. Also indexing and/or offsetting can be specified using the AC memory and an ALU function.

bits 15-13 = 7 octal jump op code indicating an address source of the BR register. Also indexing and/or offsetting can be specified using the AC memory and an ALU function.

condition bits 12-10 = specifies the condition under which the jump will take place.

condition=0 = jump/branch always.

condition=1 = jump/branch if any external interrupt flag line is true.

condition=2 = jump/branch if BR register bit 0 is set.

condition=3 = jump/branch if BR register bit 3 is set.

condition=4 = jump/branch if BR register bit 7 is set.

condition=5 = jump/branch if the carry bit from ALU is set.

condition=6 = jump/branch if the zero bit from ALU is set.

condition=7 = PUSHJ; push the PC onto the stack and jump/branch. This is used to go to a subroutine.

Bits 9=8: PAGE bits for jumping between 256 location pages within a 1K section (bank). These bits are JAM loaded into a P,C, bits 9=8 at the end of the JUMP instruction cycle. Bit 7 of the JUMP Op codes 6 & 7 enables section or bank switching.

Indexing and Offset -
By using the JUMP instruction op code 6 & 7, indexing and/or offsetting can be accomplished within the JUMP cycle. The contents of an AC memory location specified by bit 6=4 of these JUMP instructions can be combined with the source by specifying an ALU function in bit 3=0 of these instructions. For example; using JUMP Op Code = 6, the program can index into a Page in the micro-store by adding (ALU function code = 00) the contents of a working memory location to the contents of AC memory location specified by bits 6=4 of the instruction.

Section Switching or Bank Switching -
Jumping between 1K section or banks is accomplished with JUMP instructions op code 6 & 7 with bit 7 in the instruction set to a one, bit 7 acts as a section enable allowing bits 5 & 4 of these instructions to be JAM loaded into bits 11-10 respectively of the PC. These bits remain latched in the PC until another JUMP instruction with bit 7 set is executed. Note that when bit 7 of these instructions is set (section enable) bit 6 must always be zero.

Miscellaneous Jump Information -
A jump push instruction increments the PC before the PC is pushed onto the stack.

A move always increments the PC with carry into the page and bank fields.

A move pop (POPJ) loads the PC from the stack and increments the PC at the end of cycle.

The stack cannot be used for data storage.

4.2.2.2 Microcontroller Move Instruction Description - The microcontroller can execute four basic move instructions. From these four instructions, however, a wide variety of operations/functions can be accomplished, some of these are as follows:

Move to one of five selectable destinations:

BR
I/O register
Working memory
AC memory
BR shifted

Move with one of sixteen ALU functions,

Return from subroutine using the move POP instruction,

Manipulate the MA Register:

Load low order bits - 0 through 7
Load extended bits - 8 and 9
Increment (+1)

Move Instruction Breakdown -

bits

15-13 = 0 octal - Move OP code indicating a move immediate. Bits 7-0 of the instruction is the data source which is moved to a selected destination, Note - no ALU function or AC location can be specified.

15-13 = 2 octal - Move OP code indicating the data source is the working memory location addressed by the MA register. Any ALU, destination or MA control code can be specified.

15-13 = 3 octal - Move OP code indicatnig the data source is the BR. Any ALU, destination or MA control code can be specified.

15-13 = 1 or 5 octal- Move OP code indicating the data source is an I/O register on the microbus addressed by bits 15 and 3-0 of the instruction. Any ALU, destination, or MA control code can be specified.

Destination — Bits 10, 11, 12 are used in the MOVE instruction, destination.

Destination = 0 - NOP - No destination specified.

Destination = 1 - Data destination is the BR.

Destination = 2 or 3- Data destinations an I/O register, Bits 7-4 and 10 specify I/O register address.

Destination = 4 - Data destination is the AC memory location specified by bits 6-4 of the move instruction.

Destination = 6 - Data destination is the BR shifted.

Destination = 7 - Move POP instruction. This instruction is used to return from a subroutine.

Bits

6-4 — Specify an AC memory location if required to be used in the microinstruction.

7-4 — Specify an I/O output register addess.

3-0 — Specify the ALU function to be used in the MOVE instruction.

ALU Function Codes Implemented

= 0	ADD: A + B
= 1	ADD with CARRY: A + B
= 2	1's complement = SUBTRACT with CARRY: 1 = B + C - 1
= 3	Increment A
= 4	A plus C
= 5	A times 2
= 6	A times 2 plus Carry
= 7	DEcrement A
= 10	Select A
= 11	Select B
= 12	A OR NOT B
= 13	A AND B
= 14	A OR B
= 15	A XOR B

= 16	2's complement SUBTRACT: A = B
= 17	1's complement SUBTRACT: A = B - 1

Bits

8-9	Specify the MA register control to be performed in this instruction.
MA Control = 0	NOP = NO MA control function specified.
MA Control = 1	Load MA register extension bits, loads bits 9 and 8 of the MA register.
MA Control = 2	Load the low order bits of the MA register bits 7-0.
MA Control = 3	Increment the MA register by one at the end of this instruction.

4.3 CHANNEL OPERATION

The DX20 can be used either as a channel or a control unit. When used as a channel it provides the interface between the RH20 Massbus controller and peripheral device control units. For control unit operation it interfaces the Massbus controller and a second computer system such as another Massbus controller or IBM 360/370 System. In the latter configuration the DECsystem which is normally the master, becomes the device.

The channel bus interface (CBI) contains: registers, controls, translation circuits, and checking logic, which function as a control unit or provide sequence controls to provide the operating interface between the DX20 and a device control unit. It consists of two 9-bit buses (eight data plus parity) and tag in (interface control) and tag out lines. Data issued to the device is transferred on the 9-bit bus out lines, and conversely data from the device to the DX20 is received on the 9-bit bus in lines. Identification of information on either bus, and control of interface connect/disconnect sequences is effected on the tag in and tag out lines.

CBI functions are produced as directed by the microcontroller via the microbus. Control, CBI conditions, and parameter information transferred to and from the CBI takes place over the microbus. The microcontroller initiates data transfers between the CBI and data path (slave bus). Once the transfer is in progress, the microcontroller is not required for the CBI to complete the high-speed data transfer with the data path.

4.3.1 Channel Bus Interface Registers

Device Register - The CBI device register is a 9-bit register which holds information received from, or to be transmitted to, the channel bus. This information may be a device address, status, commands, parameters, or data.

Buffer Register - Information from either the microbus or from the 16-byte scratch pad buffer is held in this register. This information may be used internally in the diagnostic mode (loop enable) or issued to the channel bus.

Scratch Pad - The scratch pad provides storage for 16 bytes of information received from the microbus. Information stored here may be transferred out or used in the diagnostic mode (loop enable).

Addressing of the scratch pad is controlled by a presettable counter (from the microbus). It is possible to wrap-around within the scratch pad while the CBI is transferring information to the channel bus; since underflow or overflow detection is not provided.

Tag Out Registers - Tag out registers 0 and 1 (addresses 2 and 3) are 8-bit registers which are loaded and read by the microbus.

Outbound tag (control) lines are treated as a channel and inbound tag lines as a control unit; depending on the mode that the CBI is currently running in.

Tag In Registers - Tag in register 0 and 1 are not actually registers, but control gates. Microbus access to the channel bus tag control lines is accomplished via the register select lines (address 4 for 0, and 5 for 1). Tag line operation is possible in channel or control unit applications or in the diagnostic mode.

Channel Registers - The two channel registers (status and mode) are used to hold information which defines and controls the state of the channel bus interface. These registers may be either written into, or read from, via the microbus. In normal operation register 0 basically holds error conditions (parity error information) or internal CBI conditions. Each bit in this register may be individually set or cleared from the microbus. This register may be used during diagnostic mode. Register 1 is used as a control register. Setting and clearing of the specified bits is performed by the microcode. The microcode uses this register in both normal diagnostic mode operations.

Address Select - Address selection in the channel bus interface is determined from five microbus address select lines. These lines are decoded and form 15 register select signals which are used for read operations, or both.

4.3.2 Information Buses

Each bus has nine lines; eight information lines, and one parity line. Data bits are arranged in ascending order from bit 7 (lowest) to bit 0. If a transmitted byte has less than eight information bits, the data will occupy contiguous bit positions from the lowest towards the highest. Unused bus lines will occupy the remainder of the bit positions up to bit 0. The parity position (P) must contain the parity bit (odd parity) in all bytes.

The bus out lines are used to transmit: output data, I/O commands, and device selection (address) codes to the device control units. Timing as well as the type of information on the bus out lines, is signaled to the control unit by activating one of the outbound tag lines.

1. ADDRESS OUT is activated during the initial selection sequence to indicate that a device address is being transmitted on the bus out lines. This alerts all on-line control units for an initial selection sequence.

2. COMMAND OUT is issued by the channel to signal a selected I/O device in response to the following control signals. It responds to ADDRESS IN during initial selection; indicating the receipt of a device address; and that a command code is being transmitted to the control unit via the bus out lines and for the control unit to proceed

with the operations.

COMMAND OUT in response to SERVICE IN or DATA IN during a data transfer sequence causes the control unit to terminate (data transfer stop) and eventually initiate an interrupt sequence.

In response to STATUS IN, COMMAND OUT notifies the control unit toi stack (retain) its current status.

3. SERVICE OUT is a tag line issued by the DX200 to the control unit indicating that the DX20 has accepted status information on the bus in lines, or has provided data to the control unit on the bus out lines.

   SERVICE OUT in response to STATUS IN, acknowledges receipt of the status byte and allows the control unit to clear its status. Status conditions accepted in this manner will not generate another interrupt sequence.

   SERVICE OUT in response to SERVICE IN indicates that the channel has accepted a byte of input data, or has placed a byte of output data on the output bus.

4. DATA OUT is generated by the CBI as a response to DATA IN. This notifies the control unit that the channel accepted a byte of input data; or that a byte of output data is on the bus out lines.

4.3.3 Inbound Control Lines

The inbound control lines are used for timing and to identify data on the input bus. They are activated only by the selected control unit and only one line may be active at a given time.

1. SERVICE IN is a tag line issued from the control unit, to the channel, when it wants to transmit or receive a byte of information.

2. A control unit activates STATUS IN to notify the CBI that a status byte has been placed on the input bus. The status byte has a fixed format consisting of bits describing the current status of the control unit. Depending on whether or not the byte is accepted, the channel replies with SERVICE OUT or COMMAND OUT. The control unit may initiate an additional interrupt sequence only to signal a change in status, since the channel had accepted a status byte from that unit.

   STATUS IN in response to ADDRESS OUT during the initial selection sequence indicates that the selected control unit is unable to supply status, or accept a function byte due to the following:

a. It is executing a previously initiated operation.

b. A shared control unit is holding status for a device other than the one being addressed.

3. ADDRESS IN is activated by a device control unit to indicate that its address has been placed on the CBI input bus lines. This condition is held until the channel signals the control unit to proceed by activating COMMAND OUT.

4. DATA IN indicates that a control unit is ready to receive or transmit a byte of data.

4.3.4 Priority Lines Out

1. SELECT OUT is used to scan control units in sequence, starting at the highest priority, for the purpose of connecting the selected one to the channel. The select line passes through a logic network in each control unit, reamplified, and transmitted to the next unit, thereby forming a distributed priority network. The rules for capturing or passing on the SELECT OUT signal establishes the sequential scan.

   The connection between a control unit and the channel can only be established, when the level of the incoming SELECT OUT signal switches at the control unit (from inactive to active). If the control unit does not require connection at this transition time, it propagates SELECT OUT to the next control unit in the sequence. Once a control unit has propagated the select signal, it may not contact the channel again until the transition reoccurs.

   To capture the interface, the control unit activates operational in (OPL IN) when SELECT OUT is received (leading edge), and inhibits the propagation of SELECT OUT to the next control unit. When a control unit conditions OPL IN, it places its device address (identifying code) on the input channel bus and activates an inbound control line (ADDRESS IN). If ADDRESS OUT is active and the control unit does not recognize the address being transmitted on the bus out lines, the control unit must propagate SELECT OUT. The channel must hold SELECT OUT active until it receives a response on either the inbound control line or SELECT IN.

NOTE

The lowest priority control unit propagates SELECT OUT as SELECT IN to the channel. The terminator assembly provides for the turn-around of SELECT OUT.

2. SUPPRESS OUT is used alone and with other outgoing signals to provide the following special functions:

   a. Suppresses status from control units when required.

   b. Indicates command chaining to control units during a SERVICE OUT response to STATUS IN.

   c. Provides a selective reset function by resetting the control unit transmitting operational in. The channel executes this function by activating and maintaining SUPPRESS OUT, and dropping operational out (OPL OUT).

3. HOLD OUT is issued by the channel to all control units and combined with SELECT OUT, provides the synchronization for control unit selection. It minimizes the propagation of SELECT OUT, during selection, by purging it from the select out path and is active only when OPL OUT is active. Once HOLD OUT drops it must be held off for a duration of 1.5 microseconds. During this period the means for purging, or blocking, SELECT OUT at all control units simultaneously rather than waiting for it to be propagated serially through all units.

4.3.5 Priority Lines In

1. REQUEST IN notifies the CBI that one or more control units requires attention, and will initiate an operation whenever it can capture the interface. REQUEST IN is dropped by the control unit after it gains selection, and has no further service requirements (no later than 250 nanoseconds after operational in is activated).

   REQUEST IN should not be activated until the control unit is ready to transmit information. It is only required for control-unit-initiated selection; once the control unit has control of the interface and is transmitting OPL IN, it need not be activated for each byte of information.

2. SELECT IN is a line that extends the SELECT OUT signal from the lowest priority control unit back to the DX20. It notifies the channel that all control units have propagated SELECT OUT during initial selection or a control requested connection, and did not recognize its device address on the output bus.

4.3.6 Bus Extension

The addition of a second bus to the existing bus allows two bytes in parallel to be transmitted or received. The existing, or current bus is defined as Bus 0 and the extended bus is Bus 1. Both buses are identical in that they contain eight data and one parity line.

When the bus extension feature is available and a two byte wide data transfer is requested by a control unit, the channel activates the MARK 0 OUT and MARK 1 OUT lines.

MARK 0 IN and MARK 1 IN are activated by the control unit to notify the channel, that two parallel bytes are required for the forthcoming function.

Odd parity is provided for the MARK IN and MARK OUT lines.

4.3.7 Special Controls

1. CLOCK OUT is a signal activated by the channel, notifying the control unit that the central processing unit (CPU) is not in a wait or stopped state. When this line is active; on-line control units must be switched to the diagnostic mode (CE mode).

2. METER OUT transmitted by the channel, allows all control units to record time on their usage meters.

3. METERING IN issued to the channel, indicates that the control unit is recording time.

4.3.8 Channel Bus Operation

Channel bus operation is controlled by the microprocessor. For any data transfer operation three distinct signal sequences are necessary to initiate, utilize, and terminate the channel bus; initial selection, data transfer, and ending. In addition to the microprocessor initiated sequences, two other sequences are used to activate the channel bus; channel-initiated, and control unit-initiated service request.

4.3.8.1 Normal Input Operation

Initial Selection Sequence - An operation is initiated by the microprocessor placing an address byte on the BUS OUT lines and the raising of the ADR OUT line. The device control unit decodes the address on the bus which must have the correct parity to be acceptable (see Figures 4-1 and 4-2).

HOLD OUT and SEL OUT are then issued to provide synchronization for control unit selection. The control unit upon receiving SEL OUT, inhibits its propagation and raises OPL IN which causes the microprocessor to respond by dropping ADR OUT. After ADR OUT falls, the control unit places its device code on BUS IN, and ADR IN may rise. For a byte multiplex operation, HOLD OUT and SEL OUT

may drop any time after this point. After the microprocessor checks the address it responds by placing the device command on BUS OUT and raising the CMD OUT line. The control unit processes the command and drops ADR IN, allowing CMD OUT to fall. When CMD OUT drops, the control unit places status information on BUS IN and raises STA IN (status in). If a START I/O instruction had been selected, sufficient information is available at this point to complete instruction execution.

When the microprocessor accepts the status condition, it responds with SRV OUT, which allows the control unit to clear its status (STA IN falls). A CMD OUT response from the microprocessor also causes STA IN to fall via the path STA IN-SRV OUT-STA IN. (NOTE: a response of CMD OUT to STA IN cannot prevent execution of an immediate command operation).

Data Transfer Sequence - A data transfer may be requested by the control unit after the initial selection sequence is completed. To transmit the data to the DX20, the control unit places a data byte on BUS IN and raises SRV IN; the tag line and validity of BUS IN must be maintained until an outbound tag is raised in response.

The control unit requests data from the DX20 by raising SRV IN. The microprocessor responds by placing the data on BUS OUT and signals with SRV OUT. BUS OUT is maintained until SRV IN falls, at which time the microprocessor responds by dropping SRV OUT.

After selection, the control unit remains connected to the DX20 for the duration of the transfer of information. This information can be: a single byte of data; a status report; an initiation of a new command; a string of data bytes; or a complete operation from start to finish of the final status report.

The duration of a connection between the DX20 and device control unit is controlled by both units. HOLD OUT and SEL OUT prevent the control unit from disconnecting, and allow the DX20 to control the duration of the connection. However, the control unit can preserve the logical connection after the DX20 permits it to disconnect (HOLD OUT and SEL OUT drop) by holding OPL IN up. In this manner the control unit can employ burst mode operation.

Depending on the duration of the connection, one of two modes of operation is established: byte multiplex or burst. These modes are established so that the program can schedule concurrent execution of multiple I/O operations. Mode selection is determined by the time duration of OPL IN. When it remains up longer than required for byte multiplexing, selection is in burst mode.

The byte multiplex mode is the normal mode for low speed I/O devices, although all I/O devices are capable of operating in burst mode when required by the channel. Devices incapable of byte multiplex operation force burst mode by holding up OPL IN until channel end status conditions occur. The transfer of one or more data bytes during a single interface sequence where control

unit timing is less than 32 microseconds is considered byte multiplex mode.

Burst mode is the normal mode of operation for high speed I/O devices. These devices force burst mode (OPL IN held up) when attached to a channel capable of byte multiplex operation. Medium speed or buffered I/O devices, which may normally operate in either mode is determined by channel data rate capabilities, are equipped with a manual or programmable switch to select the mode of operation. This switch setting is overridden when burst mode is forced by the channel. An interface disconnect executed by the channel overrides the force burst mode.

An absence of data transfer; such as when reading a long gap on tape, during burst mode must not exceed one-half minute. When an absence of data exceeds this time, equipment malfunction may be indicated.

Ending Procedure and Asynchronous Status - The ending procedure may be initiated by either the device control unit or the DX20. If the procedure is initiated by the control unit, the end of operation is completed with a one signal sequence, assuming that both channel-end and device-end status conditions occur together. When the DX20 initiates the ending procedure, the control unit may require additional time to reach the point where the proper status information is available. In this case a second signal sequence is necessary to complete the ending procedure.

Assuming that selection is already obtained, one of three situations may exist at the initiation of the ending procedure.

1. The DX20 recognizes the end of an operation before the tape unit reaches its ending point. In this situation when the control unit requires service, it raises SRV IN. The microprocessor responds with CMD OUT which indicates stop, causing the control unit to drop SRV IN and proceed to its normal ending point without requesting further service. When the device reaches the point at which it normally issues channel end, the control unit places the ending status on BUS IN and raises STA IN. The microprocessor responds with SRV OUT, unless the status condition is to be stacked; in which case it responds with CMD OUT.

2. The DX20 and device recognize the end of an operation simultaneously. When this occurs ending status is available at the control unit and is placed on BUS IN, and STA IN is raised.

3. The device completes and recognizes the end of an operation before the DX20 reaches it. This situation is completed in the same manner as the previous situation.

   If device end does not occur with channel end, device end

will be presented when available, and an additional status sequence is required.

4.3.8.2 Channel-Initiated Sequences

Immediate Command Operation - Immediate-type commands meet the following conditions (see Figure 4-3):

1. Execution requires no more information than what is contained in the command byte (no data or information bytes are transferred).

2. Channel end time coincides with STA IN, which results in information (channel end), rather than zero, being in the initial status byte.

   If the device is operating during a channel initiated sequence, the control unit issues busy status. When the control unit has status information outstanding from a previous operation or an externally initiated status condition, it issues busy status (except to the all-zero command) in addition to the other status conditions.

Control Unit Busy Sequence - When a device is addressed and the control unit is busy; or status pending for another device; the control unit responds with a status byte indicating the busy conditions. The control unit presents this status in one of two ways: either as in the initial selection sequence, or with a control unit busy sequence. A control unit busy sequence is not used in response to an initial selection sequence, addressed to a device for which chaining (propagation) has been indicated.

The microprocessor begins the control unit busy sequence by placing the device address on BUS OUT and raising ADR OUT. SEL OUT is raised and the control unit decodes the information on the BUS OUT lines. When SEL OUT rises, the control unit blocks the propagation of SEL IN, places the busy status byte on BUS IN, and raises STA IN. OPL is not raised.

After the status byte is accepted, HOLD OUT and SEL OUT are dropped. The control unit responds by dropping STA IN and disconnecting from the interface. ADR OUT is kept up until STA IN drops, thus completing the control unit busy sequence.

4.3.8.3 Control Unit Initiated Sequence (Service Request) - When the control unit requires service, it raises REQ IN to the DX20 (see Figure 4-4). When SEL OUT rises at the control unit, and no selection is being attempted by the DX20 (as indicated by ADR OUT being down), the control unit places the address of the device on BUS IN and signals with ADR IN and OPL IN. Once the microprocessor acknowledges the address, it issues CMD OUT to the control unit, indicating proceed. After ADR IN drops, the microprocessor responds by dropping CMD OUT.

4-1 Initial Selection Sequence

4-4 Control Unit Initiated Sequence

If the service request is for data, a data transfer sequence is performed.

If the service request is for status information, a status cycle sequence is performed.

4.4 INTERFACE PROGRAMMABLE REGISTERS

4.4.1 DX20 Host Massbus Interface

This section describes the registers available in the Massbus interface as seen from the Massbus. Many of the bits and complete registers are identical to those as seen from the microbus. For completeness of this section, the effect of microprocessor access to these bits and/or registers will be described here rather than in secton 4.4.2 Microbus Massbus Interface Programmable Registers. Reference should also be made to section 4.4.2 for more complete descriptions of microprocessor signals referred to in the diagnostic registers. Also, in Massbus-to-Massbus configurations, the slave host system does not have access to the diagnostic registers. The abbreviations listed below are used in the following discussions.

MBRA : Massbus register address

R : read

W : write

RO : read only

WO : write only

R/W : read/write

NU : not used

4.4.1.1 MBRA 00: Control Register (DXCTR) - The control register is used by the host system to pass the function (or command) to be executed to the microprocessor. The GO bit in bit 00 tells the microprocessor to perform the function described by the function code field in bits 05-01. When the function has been completed the microprocessor will clear GO.

Bit(s)	Type	Description
15-12	NU	These bits are reserved by the Massbus controller and are not used by the DX20 subsystem.
11	NU	Not used. Read as logical 0, write is don't care.
10-06	NU	These bits are reserved by the Massbus controller and are not used by the DX20 subsystem.

05-01	R/W	CTR F4 - CTR F0 (F4-F0) = control function 4 - control function 0. The five bit function code field contains the function or command to be executed by the microprocessor. These bits can be cleared by the host writing them or by DX RESET. The DX RESET signal is generated by the host writing the DXRES bit in the maintenance register, by Massbus INIT or during power up/down sequences under the control of CROBAR.
00	R/W	CTR GO (GO) = control go. The GO flip-flop is set by the controller to signal the microprocessor that a new function has been loaded and that it should perform the desired function. The GO flip-flop can be set by the host, but the host cannot clear the GO flip-flop. The GO flip-flop is normally cleared by the microprocessor when it has completed the operation. This is done by writing bit 1 (CLRGO) of the MPSCR1 register (see section 4.4.2.2). The GO bit is reset by DX RESET. Also, if the MP stopped flip-flop (MPSTP) sets during a synchronous data transfer, GO will be cleared on the trailing edge of the EBL pulse. The GO bit is also cleared if occupied (OCC) has not been set by the microprocessor and the microprocessor subsequently stops. In order to set the GO bit, composite error (CERR) must not be asserted.

4.4.1.2 MBRA 01: Status Register (DXSTR) - The status register contains read only information on the overall condition of the subsystem. Further information is available from other registers.

Bit(s)	Type	Description
15	RO	ATA (ATA) = attention active. The ATA flip flop drives the Massbus attention (ATTN) line. It is normally set by the microprocessor to request service from the controller. The reason for the request is either to indicate completion of an operation or to report an error condition. The hardware will also cause ATA to set on the trailing edge of the EBL pulse if the microprocessor has stopped (MPSTP), or if the microprocessor stops and occupied (OCC) has not yet been set, or if GO is not set and either an illegal register (ILR) is accessed by the controller or a control bus parity error occurred (CBPARE). The ATA flip flop can be cleared by the microprocessor, by DX RESET, or by the host writing the attention summary register with the control bus data line asserted corresponding to this interface's device selection number. For example, to clear unit 3s ATA bit, write the

attention summary register with data bit 03 asserted on the control bus.

14	RO	COMP ERR (CERR) = composite error. This signal is a level produced by the inclusive - OR of the eight least significant bits of the Error Register (DXERR). The only way to clear the level is to clear all the error conditions. Composite error is actually the OR of CLASS A ERR and CLASS B ERR. CLASS A ERR is the OR of DB PAR ERR, CB PAR ERR, MP ERR, RMR and ILR. CLASS B ERR is the OR of UB PAR ERR, MP STOPPED and ILF. See section 4.4.1.3 for complete descriptions of these errors.
13	RO	MP LINK PRESENT (LNKPRS) = microprocessor link present. This is a level available to the master host system and when detected by the host indicates full control of the microprocessor. The host that detects LNKPRS can load the microcode into the microstore, start and stop the microprocessor and access the diagnostic registers (DXDR0-DXDR7), etc.
12	RO	MP RUNNING (MPRUN) = microprocessor running. This is a level indicating the state of the UBUS RUNNING line. It is asserted whenever a microprocessor timing cycle is in progress, including single cycle operation.
11:00	NU	Not used. Read as logical 0, write is don't care.

4.4.1.3 MBRA 02: Error Register (DXERR) - The error register can be broken down into two eight bit bytes. The high order 8 bits (15-08) consists of a software defined error code used by the microprocessor to pass error information to the host system. The low order 8 bits (07-00) consists of 6 hardware error flags and 2 software error flags. The low order 8 bits must be clear to negate CERR. The low order eight bits of the error register are cleared in normal operation by the host system writing them to a zero. Additionally, the host can write these bits to a one for purposes of diagnostic checkout.

Bit(s)	Type	Description
15:08	RO	EC7-EC0 (EC7-EC0) = error code 07 - error code 0. The error code register is only writable by the microprocessor and will contain either an error code or ending status information for use with either the operational microcode or diagnostics. The register is cleared by UBUS INITIALIZE. This initialize is generated by either DX RESET or by the microprocessor setting the initialize bit in its internal status register (bit 7 of UBRA 37).
07	R/W	MP ERR (MPERR) = microprocessor error. This flip flop is normally set by the microprocessor to indicate to the host that a valid error code is contained in bits 15-08 of the error code register (EC7-EC0). It can be cleared by the microprocessor writing it to a zero and is cleared by DX RESET.
06	R/W	MP STOPPED (MPSTP) = microprocessor stopped. This flip flop will be set if microprocessor start (MPSTR) is set and UBUS RUNNING transitions to the off state. This will occur whenever the microprocessor hardware detects certain error conditions, e.g., internal parity errors. The flip-flop is cleared by DX RESET.
05	R/W	UB PAR ERR (UBPARE) = microbus parity error. This flip flop will set when even parity is detected by the Massbus interface on microbus writes (DATO) to the registers in the Massbus interface. This bit will in turn cause the microprocessor start (MP START) line to turn off, forcing the microprocessor to stop in order to preserve the current state of the DX20 for analysis by the host. The MP START line to the microprocessor is the AND of the MP START (MPSTR) flip-flop in the maintenance register (MBRA 03) being set and UB PAR ERR (UBPARE) being cleared. The UB PAR ERR flip-flop is cleared by DX RESET.

04	R/W	DB PAR ERR (DBPARE) = data buffer parity error. This flip-flop will set during synchronous data transfers whenever even parity is detected from either the Massbus data bus or the master bus. It is cleared by UBUS INITIALIZE.
03	R/W	CB PAR ERR (CBPARE) = control bus parity error. This flip-flop will set during control bus write sequences if the Massbus interface detects even parity from the Massbus control bus. It is cleared by DX RESET.
02	R/W	RMR (RMR) = register modification refused. This flip flop will set if the host tries to write any Massbus interface register while the GO bit is set except for the following registers: the maintenance register (DXMTR), the attention summary register (DXASR) or the diagnostic registers (DXDR0-DXDR7). This bit will also set if the host tries to write read-only or nonexistent registers and GO is set. This flip-flop is cleared by DX RESET.
01	R/W	ILR (ILR) = illegal register. This flip flop will set if the host tries to read or write a nonexistent register, namely, MBRA 07-MBRA 17. It is cleared by DX RESET.
00	R/W	ILF (ILF) = illegal function: This flip-flop will be set by the microprocessor if the microcode decides it has been given an illegal function to perform. This can be the result of an invalid function code, or other parameters such as an incorrect data mode. The flip-flop can be written to zero by the microprocessor and is cleared by DX RESET.

4.4.1.4 MBRA 03: Maintenance Register (DXMTR) - This register contains a bit to start the microprocessor, single cycle the microprocessor, force the microprocessor to generate even parity for the purpose of verifying parity networks, and a bit to generate a reset (initialize) of the DX20 subsystem. Although the slave-host system also contains this register, setting any of these bits will have no effect except for the reset bit.

Bit(s)	Type	Description
15-05	NU	Not used. Read as logical 0, write is don't care.
04	R/W	MP SNGL CYC (MPSC) = microprocessor single cycle. When the host sets this flip-flop and the microprocessor is then started, the microprocessor will execute the instruction in the IR and then stop. It is cleared by DX RESET.
03	R/W	MP WR EV PA (MPWEVP) = microprocessor write even parity. With this flip-flop set the microprocessor will generate even parity at the output of the ALU before clocking the BALU. This bit is written by the host and cleared by DX RESET.
02	R/W	MP START (MPSTR) = microprocessor start. The host sets this flip-flop to start the microprocessor clock running and allow instruction execution. During single cycle operation this flip-flop must be toggled by the host to execute a series of instructions. This flip-flop is written by the host and cleared by DX RESET. It is always set during normal operation.
01	R/W	DX RESET (DXRES) = DX20 reset. This bit position can be written by the host to generate a reset to the DX20. Almost all flip-flops and registers in the DX20 subsystem will be reset or initialized to the correct state. In the case of the Massbus-to-Massbus configuration, if the slave host system writes this bit, it will only initialize the Massbus interface that is connected to the slave system. However, if the master host system writes this bit, it will initialize everything, including the Massbus interface connected to the slave host system. When this bit is read, it will reflect the state of DX RESET, which is also generated by Massbus INIT or by CROBAR during power up/down sequences. Therefore, unless the output of the gate or multiplexer feeding DX RESET to the Massbus is stuck, this bit will read a logical 0. In the case of the host writing this bit, the width of the DX RESET pulse will be approximately 300 ns. In the case of the Massbus INIT it will be a minimum of 400 ns.
00	NU	Not Used. Read as logical 0, write is don't care.

4.4.1.5 MBRA 04: Attention Summary Register (DXASR) - This register is not really a register at all. Rather, it is a decoder which drives a bit on the control bus data lines corresponding to its device selection number if its attention active (ATA) flip-

flop is set. For example, if unit number 3 has its ATA bit set, when the host reads the attention summary register, bit 03 on the control bus data lines will be asserted. Furthermore, the host does not have to address a particular device or unit. All it has to do is address the attention summary register and each active device on the Massbus with its ATA flip-flop set will drive a data line corresponding to its own device number. The host can clear a particular device's ATA flip-flop by writing a logical 1 into the corresponding bit position (in our example, bit 03) of the attention summary register.

Bits	Type	Description
15-08	NU	Not used. Read as logical 0, write is don't care.
07-00	R/W	(ATA7-ATA0) = attention active 7 - attention active 0. These bits reflect the attention active (ATA) status of the eight possible devices on the Massbus. See description of ATA in section 4.4.1.2.

4.4.1.6 MBRA 05: General Purpose Register 5 (DXGP5) - This register is actually in the eight word RAM residing in the Massbus interface and is identical to MBRA 25. That is, reading or writing MBRA 05 or MBRA 25 are equivalent. Thus, the software must keep track of how this register is being accessed.

Bits	Type	Description
15-00	R/W	The definition of this register is determined by the application software. This location is a RAM location and as such is not cleared by DX RESET.

4.4.1.7 MBRA 06: Drive Type Register (DXDTR) - This register is a read-only register formed by 16 SPST switches. The switches are set up at installation and their setting is dependent on the application. The setting for magtape (DX20-V100) is 050060. A logical 1 is read if a switch is OFF (open) and a logical 0 is read if a switch is ON (closed). The switches are in two DIPs located in E30 and E39.

Bit	**Type**	**Description**
15-00	RO	DT15-DT00 (DT15-DT00) = drive type 15 - drive type 00. The 16 bits of this register form the binary representation of the drive type number.

4.4.1.8 MBRA 07- MBRA 17: not used - These registers are nonexistent and if accessed by the host will cause an ILR. They will be read as logical 0 and write is don't care.

4.4.1.9 MBRA 20 - MBRA 27: General Purpose Register 0 - 7 (DXGP0 - DXGP7) - These registers are actually an 8 word x 16 bit RAM in the Massbus interface. Their definition and use are determined by the application software. This memory is a dual ported memory, one port used by the Massbus, the other port used by the Microbus (microprocessor). The RAM has independent control logic for everything but the write clock. Therefore, the system software must assure that the Massbus controller and the microprocessor do not try to write the same address simultaneously. If this occurs, the location being addressed will receive the OR of the data from the Massbus and microbus. While the microprocessor is running, a write cycle initiated from the Massbus will be synchronized with the microprocessor's timing cycle. If the microprocessor is not running, the Massbus interface will generate the correct timing. Massbus read cycles are independent of microprocessor timing and both the Massbus and the microbus can read the same location simultaneously. Table 4-1 shows the bit mapping between the Massbus and the microbus.

4.4.1.10 MBRA 30: Diagnostic Register 0 (DXDR0) - This register is the first in a group of eight (DXDR0-DXDR7) which provide access to the microcontroller for the master host system (LNKPRS). The slave host system in Massbus-to-Massbus will read these registers as logical zeroes and writes will have no effect (don't care). In order to write the instruction register (IR) or the program counter (PC), microprocessor start (MPSTR) must be cleared. If these registers are written while MPSTR is set, the control bus cycle will be completed but it will have no effect on the IR and PC. Also, no error condition will be asserted: it is up to the programmer to be aware of this. Diagnostic register 0 provides access to the microcontroller instruction register (IR) for loading and examining the microstore, (also called CRAM) and for single cycle instruction execution without using the microstore.

Bit(s)	Type	Description
15-00	R/W	IR 15-IR00 (IR15-IR00) = instruction register 15-00. The microcontroller instruction register is a 16 bit read/write register accessible from the Massbus control bus. Its contents contain the instruction to be performed by the micro-controller. The result of reading or writing the IR depends on the current value of IREN, MSEN, and PCAI (these bits are described in the following section). The register is cleared by DX RESET.

Table 4-1 Massbus-Microbus Bit Mapping

Massbus Address	Massbus Mnemonic	Massbus Bit #	Microbus Address	Microbus Mnemonic	Byte #	Mirobus Bit #
MBRA 20	DXGP0	07-00	UBRA 10	MPGP00	00	7-0
		15-08	UBRA 11	MPGP01	01	7-0
MBRA 21	DXGP1	07-00	UBRA 12	MPGP02	02	7-0
		15-08	UBRA 13	MPGP03	03	7-0
MBRA 22	DXGP2	07-00	UBRA 14	MPGP04	04	7-0
		15-08	UBRA 15	MPGP05	05	7-0
MBRA 24	DXGP4	07-00	UBRA 20	MPGP10	10	7-0
		15-08	UBRA 21	MPGP11	11	7-0
MBRA 25	DXGP5	07-00	UBRA 22	MPGP12	12	7-0
		15-08	UBRA 23	MPGP13	13	7-0
MBRA 26	DXGP6	07-00	UBRA 24	MPGP14	14	7-0
		15-08	UBRA 25	MPGP15	15	7-0
MBRA 27	DXGP7	07-00	UBRA 26	MPGP16	16	7-0
		15-08	UBRA 27	MPGP17	17	7-0

4.4.1.11 MBRA 31: Diagnostic Register 1 (DXDR1) - This register allows access to the microcontroller program counter (PC) in preparation for loading and examining the microstore. This register also contains four bits for controlling the IR, MS (microstore) and PC.

Bit(s)	Type	Description
15	R/W	IR EN (IREN) = instruction register enable. This flip-flop is set by the host system to allow the microcontroller to execute or run a microprogram stored in the microstore. Also, with IR EN set, reading or writing the IR from the Massbus control bus will result in the IR being loaded from the microstore location pointed to by the PC. With IR EN cleared, reading the IR from the control bus will not change the current contents of the IR and writing the IR will load the IR from the control bus data lines. This bit is cleared by DX RESET.
14	R/W	MS EN (MSEN) = microstore enable. This flip flop is cleared during normal operation and inhibits writing the microstore. When the flip-flop is set and IREN is cleared, the microstore location pointed to by the PC will be loaded from the Massbus control bus data lines via the IR multiplexer during a control bus write to the IR. If IREN is set (and MSEN is set) a control bus write to the IR will load ones into the microstore location pointed to by the PC and then load ones into the IR also. The flip flop is cleared by DX RESET.
13	R/W	PC EN (PCEN) = program counter enable. When this flip flop is set it allows loading PC11-PC00 from the Massbus C11-C00 data lines during a control bus write to the PC (DXDR1). It will be set during normal microprogram execution because the Host loads the PC before starting the Microprocessor and does not bother to clear it. However, once the Microprocessor is started a write to the PC will have no effect. The PC EN bit must be cleared before the Host can modify IREN, MSEN or PCAI. It is also possible to set PCEN and load the PC in the same control bus write. Conversely, clearing PCEN will not change the PC. The bit is cleared by DX RESET.
12	R/W	PC AI (PCAI) = program counter auto-increment. This flip flop is set during normal microprogram execution to allow automatic incrementing of the PC. With PCAI and MSEN both set and IREN cleared, repetitive control bus writes of the IR will load successive locations of the microstore via the IR,

without having to load the PC each time. The PC will be incremented following the read or write of the current microstore location. This flip-flop is cleared to allow the microcontroller to repeat execution of an instruction. It is cleared by DX RESET.

11-00	R/W	PC 11-PC 0 (PC11-PC00) = program counter 11-00. This 12-bit register contains the microcontroller program counter used to address the microstore (CRAM). It can only be written from the Massbus when PCEN is set and MP START is cleared. This register is cleared by DX RESET.

Table 4-2 summarizes the results of reading or writing the IR as a function IREN, MSEN, and CTOD. The term CTOD is asserted for a control bus write and not asserted for a control bus read.

Table 4-2

NORMAL FUNCTION	DXDR1 BIT 15 IREN	14 MSEN	CTOD	MICROSTORE WRITTEN	MICROSTORE SOURCE	IR WRITTEN	IR SOURCE
Read IR	0	0	0	N	OV	N	OV
Write IR	0	0	1	N	OV	Y	MB
	0	1	0	N	OV	N	OV
Write MS	0	1	1	Y	MB	Y	MB
Read MS	1	0	0	N	OV	Y	MS
	1	0	1	N	OV	Y	MS
	1	1	0	N	OV	Y	MS
	1	1	1	Y	"1"	Y	"1"

N=No, Y=Yes, MB=Massbus, MS=Microstore, OV=Original Value

4.4.1.12 **MBRA 32: Diagnostic Register 2 (DXDR2)** - The high order 8 bits of this register show the "B" inputs to the ALU, i.e., the outputs of the source multiplexer (SMUX 7-SMUX 0) and the low order 8 bits show the "A" inputs to the ALU, i.e., the output of the ACs (AC 7-AC 0)

Bits	Type	Description
15-08	RO	(ALUB7-ALUB0) = ALU B inputs. The inputs to the "B" side of the ALU are derived from the source multiplexer (SMUX7-SMUX0). The outputs of the source multiplexer are selected by the source ROM. The available outputs are IR 7-IR 0, IMUX 7 - IMUX 0, MEM7-MEM0 and BR7-BR0. Each of these outputs are also available directly, i.e., the IR from DXDR0, the IMUX and BR from DXDR 6 and the MEM from DXDR5. To determine which of these outputs are being read requires decoding of source ROM bits 5 and 6 from DXDR3 (SROM5 and SROM6). The decode of these bits is as follows:

SROM6	SROM5	SOURCE
0	0	IR
0	1	IMUX
1	0	MEM
1	1	BR

07-00	RO	(ALUA7-ALUA0) = ALU A inputs. The inputs to the "A" side of the ALU are from one of the eight accumulator registers. To determine which AC is being read requires decoding IR6-IR4 in a straightforward octal code.

4.4.1.13 **MBRA 33: Diagnostic Register 3 (DXDR3)** - The high order 8 bits of this register show the output of the source ROM (SROM7-SROM0) and the low order 8 bits show the outputs of the destination ROM (DROM7-DROM0). Each of these ROMs contains 32 8-bit bytes. To determine which ROM location is being read requires decoding certain IR bits (see below). The outputs from these ROMs have also been assigned names to indicate their function. See drawing D-FD-M8602-0-7 for the actual ROM bit patterns.

Bit(s)	Type	Description
15-08	RO	(SROM7-SROM0) = Source ROM outputs. The location of the SROM being read is determined from the following IR bits (from MSB to LSB): IR15, IR14, IR13, IR09 and IR08. The 8 outputs have the following names for a logical 1:

```
SROM7 =      -MA EXT BITS EN
SROM6 =       SMUX S1
SROM5 =       SMUX S0
SROM4 =      -MOV INST
SROM3 =       MOV INST
SROM2 =       SEC SEL EN
SROM1 =       ALU PAR CHK EN
SROM0 =       IBUS
```

07-00 (DROM7-DROM0) = destination ROM outputs. The location of the DROM being read is determined from the following IR bits (from MSB to LSB): IR09, IR08, IR12, IR11 and IR10. The 8 outputs are defined as follows for a logical 1:

```
DROM7 =   WRITE MEM
DROM6 =   WRITE AC
DROM5 =   MAR INC EN
DROM4 =  -MAR LD EN
DROM3 =   WRITE OUT
DROM2 =   PUSH/POP
DROM1 =   BR S1
DROM0 =   BR S0
```

4.4.1.14 MBRA 34: Diagnostic Register 4 (DXDR4) - The high order 8 bits of this register show the output of the function ROM (FROM7-FROM0) and the low order 8 bits show the level of the four internal parity bits and the interrupt status of the microbus.

Bit(s)	Type	Description
15-08	RO	(FROM7-FROM0) = function ROM outputs. Which of the 32 locations of the FROM being read is determined from the following IR bits (from MSB to LSB): IR14, IR03, IR02, IR01 and IR00. The 8 outputs have the following names for a logical 1:

```
FROM7 =   CLK C
FROM6 =   EN C
FROM5 =   FORCE C
FROM4 =  -ALU S0
FROM3 =  -ALU S1
FROM2 =  -ALU S2
FROM1 =  -ALU S3
FROM0 =   ALU M
```

07 RO IR PAR (IRPAR) = instruction register parity. This flip-flop indicates the IR parity as read from the microstore or as received from the Massbus depending on when it is read. This flop is cleared by "DX RESET".

06	RO	BALU PAR GEN (BALUPA) = BALU parity generator output. This level feeds the BALU PAR flip-flop and indicates the parity level being generated by the outputs of ALU. This level is complemented by having MPWEVP set in the Massbus interface.
05	RO	AC PAR (ALUAPA) = accumulator parity. This level is the parity bit as stored for one of the eight accumulators. If IR6-IR4 are the same as when DXDR2 was read, this bit will reflect the correct parity for ALUA7-ALUA0 (AC7-AC0). It is called ALUAPA because the output of the parity checker this bit feeds goes to the ALU A parity error flag (ALUAPE).
04	RO	SMUX PAR (ALUBPA) - source mux parity. This level is the parity bit for the source multiplexer. If the IR has not chanaged since reading DXDR2, this bit will reflect the correct parity for ALUB7-ALUB0 (SMUX7-SMUX0) (except for when the IR is selected by the SROM, in which case this bit will be read as a logical 1). It is called ALUBPA because the output of the parity checker this bit feeds goes to the ALU B parity error flag (ALUBPE).
03-00	ROM	INT3-INT0 (INT3-INT0) = interrupt 3-interrupt0. This register contains the status of the microbus interrupt lines (UBUS INT 3 - UBUS INT 0). This register is clocked at microprocessor time state T60 and is cleared by DX RESET.

4.4.1.15 MBRA 35: Diagnostic Register 5 (DXDR5) - The high order 8 bits of this register show the outputs of the BALU register and the low order 8 bits show the output of the working memory location pointed to by the memory address register (see section 4.4.1.17).

Bit(s)	Type	Description
15-08	RO	BALU7-BALU0 (BALU7-BALU0) = buffered ALU 7-0. This register contains the output of the ALU as of the last microprocessor T180 time state. The register is cleared by DX RESET.
07-00	RO	MEM7-MEM0 (MEM7-MEM0) = memory 7 - memory 0. This register reflects the output of one of the 1024 working memory locations pointed to by the memory address register. The data stored in working memory comes from the BALU.

4.4.1.16 MBRA 36: Diagnostic Register 6 (DXDR6) - The high order 8 bits of this register show the outputs of the input multiplexer (IMUX7-IMUX0) and the low order 8 bits show the outputs of the buffer register (BR7-BR0).

Bit(s)	Type	Description
15-08	RO	IMUX7-IMUX0 (IMUX7-IMUX0) = input multiplexer 7-0. The outputs of the input multiplexer are fed to the source multiplexer (SMUX). The outputs available from the input mux are the UBUS data register, the I/O bank register and the status register (the status register inputs are also available as follows: the Z and C bits from DXDR7 (see section 4.4.1.17) and INT3-INT0 from DXDR4). The UBUS data register contains the data strobed from the microbus data lines on the last DATI instruction. The I/O bank register is used for selecting different microbus interfaces for input/output operations. The data seen from the input mux is selected by the following IR bits: IR15 and IR 3 - IR 0. The decoding of these bits and the signals available are shown below. All of these signals are cleared by DX RESET.
7-0	RO	BR 7 - BR 0 (BR7-BR0) = Buffer Register 7-0. The outputs of the buffer register are fed into the source multiplexer (SMUX). The buffer register is loaded from the BALU at microprocessor time state T30. The BR can be used for a "rotate right" operation under the control of DROM bits 0 & 1. The BR is cleared by "DX RESET."

IR BITS	IMUX		
15 3 2 1 0	SEL S2	SEL S1	SOURCE
X X X X X	0	0	not possible
All other decodes	0	1	UBUS data
1 1 1 1 1	1	0	I/O bank (37)
1 1 1 1 0	1	1	Status (36)

	UBUS	I/O BANK	STATUS
IMUX7	DATA 7	INITIALIZE	0
IMUX6	DATA 6	SP RESET	0
IMUX5	DATA 5	OUTPUT SEL 2	Z
IMUX4	DATA 4	OUTPUT SEL 1	C
IMUX3	DATA 3	OUPTUT SEL 0	INT 3
IMUX2	DATA 2	OUTPUT SEL 2	INT 2
IMUX1	DATA 1	INPUT SEL 1	INT 1
IMUX0	DATA 0	INPUT SEL 0	INT 0

4.4.1.17 MBRA 37: Diagnostic Register 7 (DXDR7) - The high order 6 bits of this register contain various microprocessor status bits and the low order 10 bits show the memory address register outputs.

15	RO	C (ALUC) = Carry bit from the ALU. This flip-flop sets to indicate a carry occurred from the most significant bit of the ALU as a result of the last MOVE instruction with an arithmetic ALU function. Cleared by DX RESET.
14	RO	Z (ALUZ) = ALU Z bit. This flip-flop sets when the resultant output of the ALU from a MOVE instruction is equal to all ones. A typical use of the Z bit is in the comparison of the A and B inputs of the ALU. By placing the ALU in the ones complement subtract mode, a magnitude comparison of the A and B inputs is made. Cleared by DX RESET.
13	RO	IR PAR ER (IRPARE) = instruction register parity error. This flip-flop sets when the input to the IR has even parity. Cleared by DX RESET.
12	RO	ST O/U FLW (STKOU) = Stack Over/Under flow. This flip-flop sets when the hardware detects a stack overflow (too many PUSHs) or a stack underflow (too many POPs). Cleared by DX RESET or SP RESET. The SP RESET is a flip flop in the I/O bank register that the microprocessor sets.
11	RO	ALU A PAR ER (ALUAPE) = ALU A parity error. This flip-flop sets when the inputs to the A side of the ALU have even parity. Cleared by DX RESET.
10	RO	ALU B PAR ER (ALUBPE) = ALU B parity error. This flip flop sets when the inputs to the B side of the ALU have even parity. During a microprocessor DATI cycle, checking the ALU B parity will be inhibited unless the I/O interface being accessed asserts UBUS IPAR ENA. Currently, none of the interfaces designed can drive this line. Cleared by DX RESET.
09-00	RO	MA 9 - MA 0 (MA9-MA0) = memory address 9-0. This 10 bit register contains the address for selecting one of 1K (1024 decimal) working memory locations. This register is loaded in two bytes from the BALU. It is cleared by DX RESET.

4.4.2 Microbus Massbus Interface

This section describes the registers available in the Massbus interface as seen from the Microbus (UBUS). Many of the bits and complete registers are identical to those as seen from the Massbus. Where this occurs reference is made to the appropriate section in 4.4.1 Host Massbus Interface Programmable Registers. To address the Massbus interface connected to the master host system the microprocessor must first assert UBUS I/O SEL 1 and to address the Massbus interface connected to the slave host system (Massbus-to-Massbus configuration) the microprocessor must first assert UBUS I/O SEL 3. The term microbus register address is abbreviated UBRA.

4.4.2.1 UBRA 00: Status and Control Register 0 (MPSCR0) - This register provides microprocessor access to the function code, GO bit, Massbus RUN line and a write clock flag.

Bit(s)	Type	Description
7	RO	WCLK (WCLK) = write clock. The WCLK flip flop is set on receiving WCLK from the Massbus controller and occupied (OCC) is set. This flag indicates to the microprocessor that valid data has been strobed into the data buffer when synchronous data transfers are done via the microbus. The WCLK flag is cleared by the microprocessor setting START or by UBUS INITIALIZE. Write is don't care.
6-2	RO	CTR F4-CTR F0 (F4-F0) = control function 4 - control function 0. See section 4.4.1.1.
1	RO	CTR GO (GO) = control go. See section 4.4.1.1.
0	RO	RUN (RUN) = Massbus RUN line. This level reflects the state of the Massbus RUN signal. It is asserted by the Massbus controller to indicate the start-up and continuation of synchronous data transfers. Write is don't care.

4.4.2.2 UBRA 01: Status and Control Register 1 (MPSCR1) - This register provides microprocessor access to various status and control signals in the Massbus interface.

Bit	Type	Description
7	R/W	ATA (ATA) = attention active. See section 4.4.1.2.
6	R/W	MP ERR (MPERR) = microprocessor error. See section 4.4.1.3.
5	R/W	ILF (ILF) = illegal function. See section 4.4.1.3.

4	R/W	OCC (OCC) = occupied. The OCC flip flop is set by the microprocessor to tell the controller that the device is ready to begin a synchronous data transfer. It is cleared by the microprocessor writing it to zero, by UBUS INITIALIZE or on the trailing edge of EBL if the microprocessor has stopped (MPSTP).
3	R/W	DATA TO DEV (DTD) = data to device. The DATA TO DEV flip flop is set by the microprocessor to indicate a device write operation (master-host system to slave-host system) or is cleared to indicate a device read (slave-host system to master-host system). This signal is only used internal to the subsystem to properly initialize the data transfer direction. It is cleared by UBUS INITIALIZE. Note: In Massbus-to-Massbus configurations, the Massbus interface connected to slave-host system will complement the use of this bit. The master and slave DTD flops will be physically independent, but logically the complement of each other.
2	R	COMP ERR (CERR) = composite error. See section 4.4.1.2.
2	W	START DATA XFER (START) = start data transfer. The START DATA XFER flip flop is set by the microprocessor to start synchronous data transfers. If DTD is set, setting START will generate an SCLK to request a word from the controller. If DTD is cleared, setting START will generate MSTR REQ to request a word from the data path via the master bus assuming this is a high speed data transfer. START is cleared once the SCLK synchronizer circuit is "synched" or by UBUS INITIALIZE. For high speed transfers, subsequent SCLKs are automatically generated and START is not used. For non-high speed transfers, the microprocessor must set the START flop each time it is ready to request transfer of data.
1	R	REC EXC (EXC) = receive exception. This level will be asserted whenever the controller drives the exception line on the Massbus or if any device on the Massbus is driving the exception line since the Massbus bidirectional lines form a wired-OR. The DX20 will assert this line when the TRA EXC (transmit exception) flip-flop is set. TRA EXC is set whenever a CLASS A or a CLASS B error occurs and the microprocessor has previously set occupied (OCC). TRA EXC is normally cleared on the trailing edge of EBL or by UBUS INITIALIZE.

1	W	(CLRGO) = clear go. See section 4.4.1.1.
0	R	DONE (DONE) = done. this flip flop is set by the hardware on the trailing edge of EBL if the controller has dropped RUN. It indicatets that the data transfer is completed and that the device should disconnect from the Massbus. If DONE does not set after the EBL, then the device should continue transferring data. DONE is cleared by the microprocessor setting START or by UBUS INITIALIZE.
0	W	EBL (EBL) = end of block. The microprocessor writes this bit with a logical 1 to trigger a 1500 ns (min) one-shot to drive the Massbus EBL line to tell the controller it has transferred as many words as requested by the last read/write command. It will also be triggered if the microprocessor stops (MPSTP) during a synchronous data transfer following the setting of TRA EXC.

4.4.2.3 UBRA 02: Error Code Register (MPECR) - This register forms the 8 most significant bits of the Massbus error register (DXERR).

Bit(s)	Type	Description
7-0	R/W	EC7-EC0 (EC7-EC0) = error code 7 - error code 0. See section 4.4.1.3.

4.4.2.4 UBRA 03: Drive Type Register-Low (MPDTRL) - This register is the 8 least significant bits of the Massbus drive type register (DXDTR).

7-0	RO	DT07-DT00 (DT07-DT00) = drive type 07 - drive type 00. See section 4.4.1.7.

4.4.2.5 UBRA 04: Drive Type Register-High (MPDTRH) - This register is the 8 most significant bits of the Massbus drive type register (DXDTR).

Bit(s)	Type	Description
7-0	Ro	DT15-DT08 (DT15-DT08) = drive type 15-drive type 08. See section 4.4.1.7.

4.4.2.6 UBRA 05-UBRA 07: Data Buffer Register 0-2 (MPDBR0-MPDBR2) - These three registers provide access from the 8 bit microbus to the 18 bit Massbus data bus for purposes of handling non-high speed data transfers and for diagnostic checkout. Additionally, an even parity flip flop is provided, along with the parity error flag and the parity bit itself.

Bit(s)	Type	Description
		UBRA 05: Data Buffer Register 0 (MPDBR0)
7-0	R/W	DB07-DB00 (DB07-DB00) = data buffer 07-data buffer 00. Provides access to bits 07-00 of the 18 bit data buffer. This register is not initialized.
		UBRA 06: Data Buffer Register 1 (MPDBR1)
7-0	R/W	DB15-DB08 (DB15-DB08) = data buffer 08. Provides access to bits 15-08 of the 18 bit data buffer. This register is also not initialized.
		UBRA 07: Data Buffer Register 2 (MPDBR2)
7-5	NU	Not Used. Read as logical 0, write is don't care.
4	R/W	EVEN PAR (DBEVEN) = data buffer even parity. This flip flop is set by the microprocessor in order to force the Massbus interface to complement the Massbus data parity bit (DPA) or the master bus parity bit (DSPAR) when writing the data into the data buffer. If this flip-flop is set and the microprocessor tries to write the parity bit in the data buffer it will also be complemented (provided a logical 1 is maintained in this bit position). This flip-flop is provided for diagnostic verification of parity checking circuits since the controller does not have the capability of generating even parity on the data bus. It is cleared by the microprocessor writing it to a logical 0 or by UBUS INITIALIZE.
3	RO	DB PAR ERR (DBPARE) = data buffer parity error. See section 4.4.1.3.
2	R/W	DB PAR (DBPAR) = data buffer parity. This is the parity bit as stored in the data buffer. This bit is not initialized.
1-0	R/W	DB17-DB16 (DB17-DB16) = data buffer 17 - data buffer 16. Provides access to bits 17-16 of the 18 bit data buffer. These two bits are not initialized.

4.4.2.7 UBRA 20 - UBRA 27: General Purpose Register 00-17 (MPGP00-MPGP17) - These regisers are actually a 16 byte x 8 bit RAM and are identical (except for organization) to Massbus general purpose register 0-7 (DXGP0-DXGP7). See section 4.4.1.9 for the bit mapping between the Massbus and the microbus.

4.4.2.8 UBRA 30 - UBRA 37: Not used - These registers do not exist in the Massbus interface and if accessed by the microprocessor will be read as logical 0s and writes will have no affect (don't care).

4.4.3 Channel Bus Interface

4.4.3.1 Introduction - Control, status, sense, and diagnostic information and data are communicated between the microcontroller and the channel bus interface over the microbus. The interface is organized as 15 8-bit registers. The register names and bit mnemonics are summarized in channel interface diagram UBUS Registers (I/O SEL 3). A more complete description of the bits and their functions appears below. This figure shows several features associated with the interface, namely the diagnostic multiplexer, diagnostic tag loop, tag redefinitions for control unit operation, and the Register 5 Multiplexer, which are also described below.

The following abbreviations are used in the following descriptions:

UBRA = microbus register address

R = read

W = write

RO = read only

WO = write only

NU = not used

TI = tag in line

TIR = tag in register

TO = tag out line

TOR = tag out register

4.4.3.2 UBRA 00: Control and Status Register 0 (CSR0) - Register 0 allows microcontroller access to 8 hardware controlled flip-flops which can be cleared by the microcontroller. Writing UBRA 00 with UBUS DATA 0 asserted clears UBRA 00 Bit 0 and writing UBRA 00 with UBUS DATA 1 asserted clears UBRA 00 Bits 5 - 1; UBUS DATA 7 - 2 are ignored during microcontroller data output (DATO) cyclese to UBRA 00.

Bit	Type	Description
7	R	END XFER (XFER) = end transfer. The END XFER flip-flop sets after the last byte has been sent to the device during high speed transfers (DX HIGH SPEED = 1) in which the DX20 is acting as a channel controller (CHAN MODE = 1). The flip-flop terminates high speed transfers by inhibiting the assertion of DAT OUT or SRV OUT by the CB CTRL

logic in response to DAT IN or SRV IN, disabling the BUS 0 OUT and BUS 1 OUT drivers, and asserting CMD OUT in response to the next DAT IN or SRV IN to indicate channel termination per channel bus protocol requirements. The END XFER flip-flop is direct cleared when DX HIGH SPEED is low.

6 R TIME OUT FLAG (TIMOUT) = time out flag. Writing UBRA 02 sets TIMER OUT, the output of a 30 millisecond retriggerable monostable multivibrator. TIMER EN (UBRA 02 bit 2) is ANDed with - TIMER OUT to form TIME OUT FLAG which is only asserted if TIMER EN is asserted and the monostable multivibrator has timed out (TIMER OUT = 0), UBUS INT 3 is asserted when TIME OUT FLAG is high. TIME OUT FLAG is cleared by asserting UBUS INITIALIZE (which clears the TIMER EN flip-flop).

5 R DP PE FLAG (DPPE) = data path parity error flag. The DP PE FLAG flip-flop is set if a parity error (even parity) is detected in data transferred over the slave bus during device read or write operations. UBUS INT 3 is asserted when DP PE FLAG is high. The flip-flop is cleared by writing UBRA 00 with UBUS DATA 1 asserted or by asserting UBUS INITIALIZE.

4 R UB PE FLAG (UBPE) = microbus parity error flag. If a parity error (even parity) is detected in data stored in the microbus receiver (UB RCVR) register during a microcontroller data output (DATO) cycle, the UB PE FLAG flip-flop is set. UBUS INT 3 is asserted when the UB PE FLAG flip-flop is set. The flip-flop is cleared by writing UBRA 00 with UBUS DATA 1 asserted or by asserting UBUS INITIALIZE.

3 R MK PE FLAG (MKPF) = mark parity error flag. The MK PE FLAG flip-flop is set if a parity error (even parity) is detected on the MK IN lines (MK 1 IN, MK 0 IN, MK P IN) when data in bus in register 0 (UBRA 07) or bus in register 1 (UBRA 13) is read by the microcontroller or loaded into data register 0 (UBRA 06) or data register 1 (UBRA 12), respectively. The MK PE FLAG flip-flop will not set unless the M8608 Channel Extension Board is present. UBUS INT 3 is asserted when the MK PE FLAG flip-flop is set. The flip-flop is cleared by writing UBRA 00 with UBUS DATA 1 asserted or by asserting UBBUS INITIALIZE.

2 R BUS 1 PE FLAG (BUS1PE) = bus 1 parity error flag. The BUS 1 PE FLAG flip-flop is set if a parity error (even parity) is detected in the data in bus

in register 1 (UBRA 13) when it is read by the microcontroller or clocked into data register 1 (UBRA 12). The BUS 1 PE FLAG flip-flop will not set unless the M8608 Channel Extension Board is present. UBUS INT 3 is asserted when the BUS 1 PE FLAG flip-flop is set. The flip-flop is cleared by writing UBRA 00 with UBUS DATA 1 asserted or by asserting UBUS INITIALIZE.

1 R BUS 0 PE FLAG (BUS0PE) = bus 0 parity error flag. The BUS 0 PE FLAG flip-flop is set if a parity error (even parity) is detected in the data in bus in register 0 (UBRA 07) when it is read by the microcontroller or clocked into data register 0 (UBRA 06). UBUS INT 3 is asserted when the BUS 0 PE FLAG flip-flop is set. The flip-flop is cleared by writing UBRA 00 with UBUS DATA 1 asserted or by asserting UBUS INITIALIZE.

0 R SLVE SEL (SLVSEL) = slave selected. The SLVE SEL flip-flop is required for control unit operation to signal the microcontroller that the select propagation logic has blocked SEL IN or SEL OUT for one of two reasons: the channel controller has initiated an initial selection sequence to a control unit address to which the DX20 is set up to respond; or the DX20 has requested service by raising REQ IN which causes the channel controller to assert SEL OUT when it is ready to service the request. The SLVE SEL flip-flop is cleared by the next selection sequence for which neither of the above conditions is true, by writing UBRA 00 with UBUS DATA 0 asserted, or by asserting UBUS INITIALIZE. When SLVE SEL is cleared during control unit operation (CHAN MODE = 0), all BUS 0 IN, all BUS 1 IN, and all TAG IN drivers except SEL IRA and REQ IN are disabled.

4.4.3.3 UBRA 01: Control and Status Register 1 (CSR1) - Register 1 contains 5 read/write control bits (7, 4, 3, 1, 0) and 3 read/write diagnostic bits (6, 5, 2). The register is direct cleared by asserting UBUS INITIALIZE or dropping UBUS RUNNING.

Bit	Type	Description
7	R/W	SP EN (SPEN) = scratch pad enable. When the SP FN flip-flop is set, data in the scratch pad RAM (SP RAM) location addressed by the scratch pad address counter (SP ADR CNTR) will be loaded into the bus out register (BOR) when it is clocked. If SP EN is high, the SP ADR CNTR will auto-decrement when the BOR is clocked or an SP RAM location is written from the microcontroller.

6	R/W	DIAG HSPD (DIHISP = diagnostic high speed. The DIAG HSPD flip-flop output is ORed with DX HIGH SPEED from the M8605 Data Storage Board to facilitate diagnostic testing of the channel bus interface.
5	R/W	EVEN PAR (EVPAR) = even parity. Normally the EVEN PAR flip-flop is cleared, allowing the BUS 0 OUT and BUS 1 OUT parity generators to generate correct (odd) parity. The EVEN PAR flip-flop is set by diagnostic programs to cause the above parity circuits to generate even parity.
4	R/W	EXTENDED BUS (EXTBUS) = extended bus. If the EXTENDED BUS flip-flop is cleared, the channel bus interface operates in the single bus mode, using only BUS 0 IN and BUS 0 OUT to transfer data over the channel bus. If the EXTENDED BUS flip-flop is set to enable the extended bus feature, the interface operates in the dual bus mode, using BUS 0 IN, BUS 1 IN, BUS 0 OUT, and BUS 1 OUT to transfer data. EXTENDED BUS should not be set if the M8608 Channel Extension Board is not present.
3	R/W	360 MODE (MOD360) = 360 mode. If the 360 MODE flip-flop is cleared, data transfers over the channel bus are accomplished with the dual handshake protocol, using SRV IN, SRV OUT, DAT IN, and DAT OUT. If the 360 MODE flip-flop is set, data transfers are accomplished with the single handshake protocol, using only SRV IN and SRV OUT.
2	R/W	LOOP EN (LOOPEN) = loop enable. The LOOP EN flip-flop is used during diagnostic testing and is cleared during normal operation. When the flip-flop is cleared, tag in register 0 (UBRA 04), tag in register 1 (UBRA 05), bus in register 0 (UBRA 07), and bus in register 1 (UBRA 13) are used to read incoming lines from the channel bus as shown on the channel interface diagram UBUS Registers (I/O SEL 3). When the flip-flop is set, signals generated within the channel bus interface are looped back into the above registers to allow diagnostic tests to be isolated from the channel bus and, thus, not require activity on the bus to perform tests involving signals in the above registers or, in the case of control unit operation, not interfere with concurrent activity on the bus. Setting LOOP EN enables the diagnostic tag loop feature which loops bits in tag out register 0 (UBRA 02) into tag in register 0 (UBRA 04) and bits in tag out register 1 (UBRA 03) into tag in register 1 (UBRA 05) as shown in the figure. Additionally, bus out register 0

(UBRA 11) is looped back into bus in register 0 (UBRA 07) and bus out register 1 (UBRA 15) is looped back into bus in register 1 (UBRA 13).

1	R/W	ON LINE (ONLINE) = on line. ON LINE is ANDed with CROBAR, a signal which is high when the DX20 is powered up, to form DX ON LINE which is read back into UBRA 01 bit 1. When DX ON LINE is high, the DX20 is allowed to be active on the channel bus, when DX ON LINE is low, all the channel bus drivers (except SEL TRA) are disabled and the G891 Power Fail and Select Bypass Module bypasses the incoming select signal to SEL TRA, effectively disconnecting the DX20 from the bus. There can be up to 5 milliseconds delay in the response of the G891 to changes in the logic level of DX ON LINE.
0	R/W	CHAN MODE (CHANL) = channel mode. When the CHAN MODE flip-flop is cleared, the channel bus interface is configured for operation as a control unit. When the flip-flop is set, the interface is configured for operation as a channel controller.

4.4.3.4 UBRA 02: Tag Out Register 0 (TORO) - Register 2 contains 8 read/write bits used to control the tag lines. Writing UBRA 02 triggers a retriggerable monostable multivibrator, setting its output (TIMER OUT) for 30 milliseconds after each microcontroller data output (DATO) cycle to UBRA 02. The register is cleared by the assertion of UBUS INITIALIZE. The functions of some of the bits are dependent on whether the DX20 is acting as a channel controller or a control unit. Signal names with dual functionality are named according to their functions as a channel controller in the block diagrams and print sets for the channel bus interface. The redefined bit names are shown in channel interface diagram UBUS Registers (I/O SEL 3). The bits in Register 2 are defined for both channel controller and control unit operation below. More detailed information on the use of the tag lines is given in Section 4.3.1.

CHANNEL CONTROLLER OPERATION (TORO)

Bit	Type	Description
7	R/W	TOR SRV OUT (SRVOUT) = TOR service out.
6	R/W	TOR CLK OUT (CLKOUT) = TOR clock out.
5	R/W	TOR MTR OUT (MTROUT) = TOR meter out.
4	R/W	TOR ADR OUT (ADROUT) = TOR address out.
3	R/W	TOR HLD OUT (HLDOUT) = TOR hold out.
2	R/W	TIMER EN (TMREN) = timer enable. When the TIMER

		EN flip-flop is set, the output of a 30 millisecond retriggerable monostable multivibrator (TIMER OUT) is allowed to assert TIME OUT FLAG when it goes low (timed out). Note that the act of writing UBRA 02 triggers the monostable.
1	R/W	TOR SEL OUT (SELOUT) = TOR select out. TOR SEL OUT asserts SEL TRA.
0	R/W	TOR CMD OUT (CMDOUT) = TOR command out.

CONTROL UNIT OPERATION (TIRO)

Bit	Type	Description
7	R/W	TIR SRV IN (SRVIN) = TIR service in.
6	R/W	TIR DIS IN (DISIN) = TIR disconnect in.
5	R/W	TIR MTR IN (MTRIN) = TIR meter in.
4	R/W	TIR ADR IN (ADRIN) = TIR address in.
3	R/W	Not redefined or used for control unit operation. The logic level of this bit is indeterminate during control unit operation.
2	R/W	TIMER EN (TMREN) = timer enable. Not redefined for control unit operation.
1	R/W	Not redefined or used for control unit operation. TOR SEL OUT should remain cleared during control unit operation. The CU CTRL logic handles the propagation of the select signal without the need for microcontroller action.
0	R/W	TIR STA IN (STAIN) = TIR status in.

4.4.3.5 UBRA 03: Tag Out Register 1 (TOR1) - Register 3 contains 5 read/write bits (7-3) and 3 read only bits (2-0). Bits 7-4 are used to control tag lines or channel bus interface responses to them and bits 3-0 are used for diagnostic testing of the interface. Bit 2 is used to read back the output of the diagnostic multiplexer, and a 16 wide multiplexer addressed by the scratch pad address counter, which allows diagnostic programs to sense the logic levels on internal control signals in the channel bus interface. Bits 7-3 are cleared by the assertion of UBUS INITIALIZE. The functitons of some of the bits are dependent on whether the DX20 is acting as a channel controller or a control unit. By default, signal names with dual funcionality are named according to their functions as a channel controller in the block diagrams and print sets for the channel bus interface. The redefined bit names are shown in channel interface diagram UBUS Registers (I/O SEL 3). The bits in register 3 are defined for

both channel controller and control unit operation below. More detailed information on the use of the tag lines is given in section 4.3.1

CHANNEL CONTROLLER OPERATION (TOR1)

Bit	Type	Description
7	R/W	TOR OPL OUT (OPLOUT) = TOR operational out.
6	R/W	CU RESET EN (CURSEN) = control unit reset enable. When the CU RESET EN flip-flop is cleared, the SYS RST, SEL RST, and HALT IO flip-flops are direct cleared, preventing the assertion of CU RESET. If CU RESET EN is high, CHAN MODE is high, and UBUS INITIALIZE is low, the above flip-flops can be set by the microcontroller or the appropriate combinations of tag lines.
5	R/W	TOR SUP OUT (SUPOUT) = TOR suppress out.
4	R/W	TOR DAT OUT (DATOUT) = TOR data out.
3	R/W	DIAG SLVE ACK (DISACK) = diagnostic slave acknowledge. The DIAG SLVE ACK flip-flop output is ORed with SLVE ACK from the M8605 Data Storage Board to facilitate diagnostic testing of the channel bus interface. In addition to being cleared by the assertion of UBUS INITIALIZE, the flip-flop is cleared approximately 30 nanoseconds after DIAG SLVE ACK or SLVE ACK from the M8605 is set in response to the assertion of SLVE REQ or by a microcontroller data output (DATO) cycle to UBRA 04.
2	RO	DIAG MUX OUT (DIMUX) = diagnostic multiplexer output. DIAG MUX OUT is the output of the diagnostic multiplexer which allows diagnostic programs to sense the logic levels of 16 internal control signals in the channel bus interface via a single bit in the microbus interface. A signal is selected by the contents of the scratch pad address counter as specified below.

ADR	SIGNAL
00	BOR P0 (BORP0) = bus out register 0 parity.
01	BOR P1 (BORP1) = bus out register 1 parity.
02	BUS 0 ODD PAR (BUS0OP) = bus in register 0 odd parity.
03	SLVE END XFER (SEXFER) = slave end transfer.

04 TRA SEL FF (TRASEL) = transmit select flip-flop.

05 ALLOW MK 1 OUT (ALLMK1) = allow mark 1 out.

06 ODD END (ODDEND) = odd end.

07 2ND BYTE (2NDBYT) = second byte.

10 DR READY (DRRDY) = device register ready.

11 SLVE REQ (SLVREQ) = slave request.

12 -DATA REQ DLY (NDRDLY) = not data request delayed.

13 CU INIT (CUINIT) = control unit initialize.

14 EN SRV/DAT OUT (ENSRDA) = enable service Out/Data Out.

15 ST DATA REQ (STDARQ) = start data request.

16 -DIS SLVE REQ (NDISSR) = not disable slave request.

17 DP PE (SLDPPE) = data path (slave) parity error.

1 RO TO DAT OUT (TODOUT) = TO data out. TO DAT OUT is asserted by the CB CTRL logic during high speed transfers to raise the DAT OUT Tag line in response to DAT IN.

0 RO TO SRV OUT (TOSOUT) = TO service out. TO SRV OUT is asserted by the CB CTRL logic during high speed transfers to raise the SRV OUT tag line in response to SRV IN.

CONTROL UNIT OPERATION (TIR1)

Bit	Type	Description
7	R/W	TIR OPL IN (OPLIN) = TIR operational in.
6	R/W	CU RESET EN (CURSEN) = control unit reset enable. Not redefined for control unit operation.
5	R/W	TIR REQ IN (REQIN) = TIR request in.
4	R/W	TIR DAT IN (DATIN) = TIR data in.

3	R/W	DIAG SLVE ACK (DISACK) = diagnostic slave acknowledge. Not redefined for control unit operation.
2	RO	DIAG MUX OUT (DIMUX) = idagnostic multiplexer output. Not redefined for control unit operation.
1	RO	TI DAT IN (TIDIN) = TI data in. TI DAT IN is asserted by the CB CTRL logic during high speed transfers to raise the DAT IN tag line to initiate a data transfer.
0	RO	TI SRV IN (TISIN) = TI service in. TI SRV IN is asserted by the CB CTRL logic during high speed transfers to raise the SRV IN tag line to initiate a data transfer.

4.4.3.6 UBRA 04: Tag In Register 0 (TAGIN0) - Register 4 contains 8 read-only bits through which the tag in lines may be read by the microcontroller during channel controller operation or through which the tag out lines may be read during control unit operation. Writing UBRA 04 clears the SLVE REQ and DIAG SLVE ACK flip-flops (UBUS DATA 7-0 is ignored). By default, signal names with dual functionality are named according to their functions as a channel controller in the block diagrams and print sets for the channel bus interface. The redefined bit names are shown in the channel interface diagram UBUS Registers (I/O SEL 3). The bits in Register 4 are defined for both channel controller and control unit operation below. More detailed information on the use of the tag lines is given in section 4.3.1. To facilitate diagnostic testing of the channel bus interface, a diagnostic tag loop is enabled when LOOP EN (UBRA 01 Bit 2) is set. This allows bits in tag out register 0 to be looped into corresponding bits in tag in register 0 so that logic driven by the tag in lines can be tested without disturbing the tag out lines of the channel bus. The figure referenced above shows the structure of the tag loop for channel controller operation.

CHANNEL CONTROLLER OPERATION (TAGIN0)

Bit	Type	Description
7	RO	TI OPL IN (OPLIN) = TI operational in. When LOOP EN is high, TOR SRV OUT is looped into TI OPL IN.
6	RO	TI MK 0 IN (MK0IN) = Tl mark 1 in. When LOOP EN is high, TOR CLK OUT is looped into TI MK 0 IN.
5	RO	TI MK 1 IN (MK1IN) = TI mark 1 in. When LOOP EN is high, TOR MTR OUT is looped into TI MK 1 IN. If the M8608 Channel Extension Board is not present, TI MK 1 IN is always low.
4	RO	TI ADR IN (ADRIN) - TI address in. When LOOP EN

is high, TOR ADR OUT is looped into TI ADR IN.

3	RO	TO HLD OUT (TOHOUT) = TO hold out. Not used for channel controller operation. When LOOP EN is low, TO HLD OUT is always low for channel controller operation. When LOOP EN is high, TOR HLD OUT is looped into TO HLD OUT.
2	RO	TI MTR IN (MTRIN) = TI meter in. When LOOP EN is high, TI MTR IN is always low.
1	RO	TI SEL IN (SELIN) = TI select in. When LOOP EN is low, the logic level of SEL REC is read via TI SEL IN. When LOOP EN is high, TOR SEL OUT is looped into TI SEL IN.
0	RO	TI STA IN (STAIN) = TI status in. When LOOP EN is high, TOR CMD OUT is looped into TI STA IN.

CONTROL UNIT OPERATION (TGOUT0)

Bit	Type	Description
7	RO	TO OPL OUT (OPLOUT) = TO operational out. When LOOP EN is high, TIR SRV IN is looped into TO OPL OUT.
6	RO	TO MK 0 OUT (MK0OUT) = TO mark 0 out. When LOOP EN is high, TIR DIS IN is looped into TO MK 0 OUT.
5	RO	TO MK 1 OUT (MK1OUT) = TO mark 1 out. When LOOP EN is high, TIR MTR IN is looped into TO MK 1 OUT. If the M8608 Channel Extension Board is not present, TO MK 1 OUT is always low.
4	RO	TO ADR OUT (ADROUT) = TO address out. When LOOP EN is high, TIR ADR IN is looped into TO ADR OUT.
3	RO	TO HLD OUT (TOHOUT) = TO hold out. Not redefined for control unit operation. When LOOP EN is high, TOR HLD OUT (not redefined for control unit operation) is looped into TO HLD OUT.
2	RO	TO MTR OUT (MTROUT) = TO meter out. When LOOP EN is high, TO MTR OUT is always low.
1	RO	TO SEL OUT (SELOUT) = TO select out. When LOOP EN is low, the logic level of SEL REC is read via TO SEL OUT. When LOOP EN is high, TOR SEL OUT (not redefined for control unit operation) is looped into TO SEL OUT.
0	RO	TO CMD OUT (CMDOUT) = TO command out. When LOOP EN is high, TIR STA IN is looped into TO CMD OUT.

4.4.3.7 UBRA 05: Tag In Register 1/Scratch Pad Address (TAGIN1/0)
Register 5 contains 4 read-only bits (7-4) through which the tag in lines may be read by the microcontroller during channel controller operation or through which the tag out lines may be read during control unit operation. By default, signal names with dual functionality are named according to their functions as a channel controller in the block diagrams and print sets for the channel bus interface. The register 5 multiplexer allows dual use of bits 3 - 0 of the register (REG 5 B3 - REG 5 B0) as shown in the channel interface diagram UBUS Registers (I/O SEL 3). Microcontroller may read and write the scratch pad address counter, which is useful only during channel controller operation, via UBRA 05 Bits 3-0 when CHAN MODE = 1, or it may read the CU RESET flip-flops (SYS RST, SEL RST, and HALT IO) and the CU RUN flip-flop, which are used only during control unit operation, via UBRA 05 Bits 3 - 0 when CHAN MODE = 0. The redefined bit names are shown in the previously referenced figure. The bits in register 5 are defined for both channel controller and control

unit operation below. To facilitate diagnostic testing of the channel bus interface, a diagnostic tag loop is enabled when LOOP EN (UBRA 01 Bit 2) is set. This allows bits in tag out register 1 to be looped into corresponding bits in tag in register 1 so that logic driven by the tag in lines can be tested without disturbing the tag out lines of the channel bus. The figure also shows the structure of the tag loop for channel controller operation. The following bit descriptions assume that CHAN MODE = 1 for channel controller operation and CHAN MODE = 0 for control unit operation.

CHANNEL CONTROLLER OPERATION (TAGIN1/SPADR)

Bit	Type	Description
7	RO	TI SRV IN (SRVIN) = TI service in. When LOOP EN is high, TOR OPL OUT is looped into TI SRV IN.
6	RO	TI DIS IN (DISIN) = TI disconnect in. When LOOP EN is high, CU RESET EN is looped into TI DIS IN. If CHAN MODE is high, UBUS INT 3 will be asserted when TI DIS IN is high.
5	RO	TI REQ IN (REQIN) = TI request in. When LOOP EN is high, TOR SUP OUT is looped into TI REQ IN.
4	RO	TI DAT IN (DATIN) = TI data in. When LOOP EN is high, TOR DAT OUT is looped into TI DAT IN.
3-0	R/W	SP ADR 3 - SP ADR 0 (SPA3 - SPA0) = scratch pad address bits 3 - 0. The scratch pad address counter (SP ADR CNTR) can be read and written by the microcontroller via UBRA 05 Bits 3-0. If SP EN (UBRA 01 Bit 7) is set, the counter will be auto-decremented when a scratch pad RAM location is written by the microcontroller via UBRA 10 or when scratch pad RAM data addressed by the counter is clocked into the Bus Out Register (BOR). The SP ADR CNTR counts modulo 16 and, thus, wraps around from 00 to 17 octal as it counts down.

CONTROL UNIT OPERATION (TGOUT1)

Bit	Type	Description
7	Ro	TO SRV OUT (SRVOUT) = TO service out. When LOOP EN is high, TIR OPL IN is looped into TO SRV OUT.
6	RO	TO CLK OUT (CLKOUT) = TO clock out. When LOOP EN is high, CU RESET EN (not redefined for control unit operation) is looped into TO CLK OUT.
5	RO	TO SUP OUT (SUPOUT) = TO suppress out. When LOOP EN is high, TIR REQ IN is looped into TO SUP OUT.

4	RO	TO DAT OUT (DATOUT) = TO data out, when LOOP EN is high, TIR DAT IN is looped into TO DAT OUT.
3	RO	CU RUN (CURUN) = control unit run. When CHAN MODE = 0, the microcontroller is allowed to read CU RUN through UBRA 05 Bit 3 for diagnostic testing of the channel bus interface.
2	R	SYS RST (SYSRST) = system reset. The SYS RST flip-flop is set during control unit operation (CHAN MODE = 0) if CU RESET EN = 1 and the channel bus interface detects TO OPL OUT = 0 and TO SUP OUT = 0 simultaneously. SYS RST may be written by the microcontroller as UBRA 16 Bit 2. CU RESET and UBUS INT 3 are asserted when SYS RST is set.
1	R	SEL RST (SELRST) = selective reset. The SEL RST flip-flop is set during control unit operation (CHAN MODE = 0) if CU RESET EN = 1 and TIR OPL IN = 1 and the channel bus interface detects TO OPL OUT = 0 and TO SUP OUT = 1) simultaneously. SEL RST may be written by the microcontroller as UBRA 16 Bit 1. CU RESET and UBUS INT 3 are asserted when SEL RST is set.
0	R	HALT IO (HALTIO) = halt I/O. The HALT IO flip-flop is set during control unit operation (CHAN MODE = 0) if CU RESET EN = 1 and TIR OPL IN = 1 and the channel bus interface detects TO ADR OUT = 1 and TO HLD OUT = 0 simultaneously. HALT IO may be written by the microcontroller as UBRA 16 Bit 0. CU RESET and UBUS INT 3 are asserted when HALT IO is set.

4.4.3.8 UBRA 06: Data Register 0 (DRLO) - Register 6 provides microcontroller access to the low byte of the DR (DR 07 - 00). The DR is used during high speed transfers (DX HIGH SPEED = 1) to buffer data between the slave device and the data path interface. During device write operations (DATA TO DEV = 1), data on the slave bus is loaded into DR 07 - 00, P0, the output of which is gated onto BUS 0 OUT 0 - 7, P. During device read operations (DATA TO DEV = 0), data on the BUS 0 IN lines is loaded into DR 07-00, PO, the output of which is gated onto slave bus data lines 7-0, P. For diagnostic testing, writing UBRA 06 causes DR 07 = 00, P0 to be loaded with data on the slave bus if DATA to DEV = 1 or with the contents of bus in register 0 if DATA TO DEV = 0.

Bit	Type	Description
7-0	R	DR 07 - DR 00 (DR07-DR00) = data register bits 07 - 00.

4.4.3.9 UBRA 07: Bus In Register 0 (CBILO) - Register 7 is a read only register which provides microcontroller access to the

BUS 0 IN 0 - 7 lines during channel controller operation or the BUS 0 OUT 0 - 7 lines for control unit operation. If LOOP EN is set, the contents of bus out register 0 (BOR 07 - 00, P0) is complemented and looped into CBI 07 - 00, P0 to facilitate diagnostic testing of the channel bus interface. When UBRA 07 is read, the BUS 0 PE FLAG (UBRA 00 Bit 1) is clocked to check for correct (odd) parity of the data in bus in register 0, and the MK PE FLAG (UBRA 00 Bit 3) is clocked to check for correct (odd) parity of the MK IN lines.

Bit	Type	Description
7-0	RO	CBI 07 - CBI 00 (CBI07 - CBI00) = channel bus in bits 07 - 00.

4.4.3.10 UBRA 10: Scratch Pad Data Register 0 (SPDALO) - Register 10 is a write only register through which data is loaded into the low byte of the scratch pad RAM (SP RAM) location addressed by the scratch pad address counter (SP ADR CNTR). If SP EN is set, writing UBRA 10 also auto-decrements the SP ADR CNTR.

Bit	Type	Description
7-0	WO	SP DATA 07 - 00 (SPD07 - SPD00) = scratch pad data bits 07 - 00.

4.4.3.11 UBRA 11: Bus Out Register 0 (BORLO) - Register 11 is a write only register which allows the microcontroller to load data into the low byte of the BOR (BOR 07-00). If SP EN = 0, writing UBRA 11 loads microbus data stored in the microbus receiver (UB RCVR) register directly into the low byte of the BOR. If SP EN = 1, writing UBRA 11 loads the lower byte of a scratch pad RAM (SP RAM) location addressed by the scratch pad address counter (SP ADR CNTR) into bus out regsiter 0 and then auto-decrements the SP ADR CNTR. The BOR is also loaded and the SP ADR CNTR is decremented by the CB CTRL logic during high speed transfers in which SP EN is set to transmit control information stored in the SP RAM by the microcontroller to the device over the BUS 0 OUT lines.

Bit	Type	Description
7-0	WO	BOR 07 - BOR 00 (BOR07 - BOR00) = bus out register Bits 07 - 00.

4.4.3.12 UBRA 12: Data Register 1 (DRHI) - Register 12 provides microcontroller access to the high byte of the DR (DR 15-08). The DR is used during high speed transfers (DX HIGH SPEED = 1) to buffer data between the slave device and the data path interface when the extended bus feature is enabled (EXTENDED BUS = 1). During device write operations (DATA TO DEV = 1), data on the slave bus is loaded into DR 15 - 08, PI, the output of which is gated onto BUS 1 OUT 0 - 7, P. During device read operations (DATA TO DEV = 0), data on the BUS 1 IN lines is loaded into DR 15 - 08, P1, the output of which is gated onto slave bus data lines 7

- 0, P. For diagnostic testing, writing UBRA 12 causes DR 15 - 08, P1 to be loaded with data on the slave bus if DATA to DEV = 1 or with the contents of bus in regsiter 1 if DATA TO DEV = 0. Data register 1 is implemented on the M8608 Channel Extension Board; if the M8608 is not present, UBRA 12 will be read as all zero data.

Bit	Type	Description
7-0	R	DR 15 - DR 08 (DR15 - DR08) = data register bits 15 - 08.

4.4.3.13 UBRA 13: Bus In Register 1 (CBIHI) - Register 13 is a read-only register which provides microcontroller access to the BUS 1 IN 0 - 7 lines during channel controller operation or the BUS 1 OUT 0 - 7 lines for control unit operation. The BUS 1 IN lines are only used when the extended bus feature is enabled (EXTENDED BUS = 1). If LOOP FN is set, the contents of bus out register 1 (BOR 15 - 08, P1) is complemented and looped into CBI 07-00, P1 to facilitate diagnostic testing of the channel bus interface. When UBRA 13 is read, the BUS 1 PE FLAG (UBRA 00 BIT 2) is clocked to check for correct (odd) parity of the data in bus in register 1, and the MK PE FLAG (UBRA 00 Bit 3) is clocked to check for correct (odd) parity of the MK IN lines. The circuitry for bus in register 1 is located on the M8608 Channel Extension Board; if the M8608 is not present, UBRA 13 will be read as all zero data.

Bit	Type	Description
7-0	RO	CBI 15 - CBI 08 (CBI15 - CBI08) = channel bus in bits 15 - 08.

4.4.3.14 UBRA 14: Scratch Pad Data Register 1 (SPDAHI) - Register 14 is a write-only register through which data is loaded into the high byte of the scratch pad RAM (SP RAM) location addressed by the sctrach pad address counter (SP ADR CNTR). The upper bytes of the SP RAM are only used when the extended bus feature is enabled (EXTENDED BUS = 1) and they are located on the M8608 Channel Extension Board.

Bit	Type	Description
7-0	WO	SP DATA 15 - 08 (SPD15 - SPD08) = scratch pad data bits 15-08.

4.4.3.15 UBRA 15: Bus Out Register 1 (BORHI) - Register 15 is a write only register which allows the microcontroller to load data into the high byte of the BOR (BOR 15 - 08). Bus out register 1 is used only when the extended bus feature is enabled (EXTENDED BUS = 1). If SP EN = 0, writing UBRA 15 loads microbus data stored in the microbus receiver (UB RCVB) register directly into the high byte of the BOR. If SP EN = 1, writing UBRA 15 loads the upper byte of the sctrach pad RAM (SP RAM) location addressed by

the scratch pad address counter (SP ADR CNTR) into bus out register 1. The BOR is also loaded and the SP ADR CNTR is decremented by the CB CTRL logic during high speed transfers in which SP EN is set to transmit control information stored in the SP RAM by the microcontroller to the device over the Bus 1 OUT lines. Bus out register 1 is implemented on the M8608 Channel Extension Board.

Bit	Type	Description
7-0	WO	BOR 15 - BOR 08 (BOR15 - BOR 08) = Bus Out Register Bits 15 - 08.

4.4.3.16 UBRA 16: Control Unit Status Register (CUSTAT) - Register 16 is a write only register through which the microcontroller can write the CU RESET flip-flops (SYS RST, SEL RST, and HALT IO). UBUS DATA 2 - 0 is clocked into the CU RESET flip-flops when UBRA 16 is written; UBUS DATA 7 - 3 is ignored. The flip-flops are direct cleared when UBUS INITIALIZE is asserted or CU RESET EN (UBRA 03 Bit 6) is cleared. The states of the flip-flops can be sensed via the register 5 multiplexer which allows SYS RST, SEL RST, and HALT IO to be read back as UBRA 05 Bits 2 - 0, respectively, when CHAN MODE = 0. See Section 4.4.3.7 for more information.

Bit	Type	Description
7-3	NU	Not used.
2	WO	SYS RST (SYSRST) = System Reset.
1	WO	SEL RST (SELRST) = Selective Reset.
0	WO	HALT IO (HALTIO) - Halt I/O.

4.4.3.17 UBRA 17 - UBRA 37: Not Used - Registers 17 - 37 are not implemented in the channel bus interface. If accessed by the microcontroller, they will be read back as zero data; DATO cycles to these registers will have no effect on the interface.

4.4.4 Data Path Interface

4.4.4.1 Introduction - Control, status, and diagnostic

information is communicated between the microcontroller and data path interface over the microbus and is organized as 16 8-bit registers. The register names and bit mnemonics are summarized in the data path diagram UBUS Registers (I/O SEL 3). A more complete description of the bits and their functions appears below.

The following abbreviations are used:

R = read
W = write
RO = read only
NU = not used

4.4.4.2 UBRA 00: Register 0 (REG0) - Register 0 contains 4 flags which are set under hardware control and can only be cleared by the microcontroller, specifically by writing UBRA 00 (UBUS data is ignored).

Bit	Type	Description
7-4	NU	Read as logical 0.
3	R	MC OF FLAG (MCOVF) = massbus counter overflow flag. The 16-bit Massbus counter is incremented every time the data path interface issues a MSTR ACK in response to MSTR REQ. When the counter advances from a maximum count of 177 777 octal to 0, the MC OF FLAG flip-flop is set.
2	R	BC OF FLAG (BCOVF) = byte counter overflow flag. The 16-bit byte counter is incremented every time the data path interface issues a SLVE ACK in response to SLVE REQ. When the counter advances from a maximum count of 177 777 octal to 0, the BC OF FLAG flip-flop is set.
1	R	DP PE FLAG (DPPEFG) = data path parity error flag. If a parity error (even parity) is detected in data stored in the silo buffer (SB) register during device writes or the channel buffer (CB) register during device reads, indicating a data error in the transfer, the DP PE FLAG flip-flop is set.
0	R	UB PE FLAG (UBPEFG) = microbus parity error flag. If a parity error (even parity) is detected in data stored in the microbus receiver (UB RCVR) register during a microcontroller data output (DATO) cycle, the UB PE FLAG flip-flop is set.

4.4.4.3 UBRA 01: Register 1 (REG 1) - Register 1 contains diagnostic bits (7-2) and control bits (1 - 0). Bits 7 - 4 are read only and bits 3 - 0 are read/write.

Bit	Type	Description
7	RO	MSTR REQ (MSTRQ) = master request. MSTR REQ is the logical OR of MSTR REQ from the M8603 Massbus Data Board and the true output of the DIAG MSTR REQ flip-flop.
6	RO	SLVE REQ (SLVRQ) = slave request. SLVE REQ is the logical OR of SLVE REQ from the M8607 Channel Bus Board and the true output of the DIAG SLVE REQ flip-flop.
5	RO	MSTR ACK (MSTACK) = master acknowledge. The MSTR ACK signal goes to the M8603 Massbus Data Board. It is given in response to MSTR REQ when the data path interface has accepted data from or presented data to the Massbus Interface over the master bus and is cleared when MSTR REQ goes low.
4	RO	SLVE ACK (SLVACK) = slave acknowledge. The SLVE ACK signal goes to the M8607 Channel Bus Board. It is given in response to SLVE REQ when the data path interface has accepted data from or presented data to the channel bus interface over the slave bus and is cleared when SLVE REQ goes low.
3	R/W	DIAG MSTR REQ (DMSTRQ) = diagnostic master request. DIAG MSTR REQ is logically ORed with MSTR REQ from the M8603 Massbus Data Board to allow diagnostic programs to exercise the master bus control logic directly.
2	R/W	DIAG SLVE REQ (DSLVRQ) = diagnostic slave request. DIAG SLVE REQ is logically ORed with SLVE REQ from the M8607 Channel Bus Board to allow diagnostic programs to exercise the slave bus control logic directly.
1	R/W	BASE CLK EN (BCLKEN) = base clock enable. When BASE CLK EN is asserted, a 21.84 MHZ clock signal is gated to the formatter control logic.
0	R/W	DX HIGH SPEED (DXHISP) = DX high speed. When DX HIGH SPEED is asserted, the data path interface is enabled to transfer data in response to MSTR REQ and SLVE REQ and in accordance with the program pointed to in the control ROM. DX HIGH SPEED L is bussed to the M8603 Massbus Data Board and the M8607 Channel Bus Board to indicate when the data path interface is enabled.

4.4.4.4 UBRA 02: Register 2 (REG2) - Register 2 contains read only diagnostic bits (7-6) and read/write control bits (5-0).

Bit	Type	Description
7	RO	MSTR REQ HLDOFF (MRHDOF) = master request holdofff. The MSTR REQ HLDOFF flip-flop inhibits the data path interface from processing another MSTR REQ. It is set on the trailing edge of the MSTR REQ and is not cleared until MSTR RDY DLY 2 drops to prevent a successive MSTR REQ during the time that MSTR REQ HLDOFF is asserted from interfering with proper operation of the data path interface.
6	RO	SLVE REQ HLDOFF (SRHDOF) = slave request holdoff. The SLVE REQ HLDOFF flip-flop inhibits the data path interface from processing another SLVE REQ. It is set on the trailing edge of SLVE REQ and is not cleared until SLVE RDY DLY 2 drops to prevent a successive SLVE REQ during the time that SLVE REQ HLDOFF is asserted from interfering with proper operation of the data path interface.
5	R/W	REG 2 B5 (REG2B5) = register 2 bit 5. This read/write bit is not presently used.
4	R/W	REG 2 B4 (REG2B4) = register 2 bit 4. This read/write bit is not presently used.
3	R/W	MEX ON FEX (MEONFE) = master end transfer on formatter end transfer enable. If the MEX ON FEX flip-flop is set, MSTR WOR END XFER will be asserted when FMTR END XFER is asserted.
2	R/W	MEX ON MC (MEMCOV) = master end transfer on massbus counter overflow enable. If the MEX ON MC flip-flop is set, MSTR WOR END XFER will be asserted when the MC OF FLAG flip-flop is set.
1	R/W	SEX ON BC (SEBCOV) = slave end transfer on byte counter overflow enable. If the SEX ON BC flip-flop is set, SLVE WOR END XFER will be asserted when the BC OF FLAG flip-flop is set.
0	R/W	ROM ADR 8 (RMADR8) = ROM address bit 8. ROM ADR 8 is the most significant bit of the data formatter control ROM address register (see Section 4.4.4.10 for more detail).

4.4.4.5 UBRA 03: Register 3 (REG3) - Register 3 contains 3 read only bits (3-1).

Bit	Type	Description
7-1	NU	Read as logical 0.

3	RO	-FMTR END XFER (NFEXR) = not formatter end transfer. FMTR END XFER is asserted after SLVE WOR END XFER has been asserted and the formatter has completed its formatting sequence as described in Section 5.2.4.3.
2	RO	-MSTR WOR END XFER (NMEXFR) = not master end transfer. MSTR WOR END XFER L is communicated between the data path interface and the Massbus interface over a wire-ORed bus which can be pulled low (asserted) by either interface to signal the termination of a Massbus transfer. MSTR WOR END XFER is asserted in the following cases: when the MC OF FLAG flip-flop is set and MEX ON MC (UBRA 02, BIT 2) is set; or when FMTR END XFER is asserted and MEX ON FEX (UBRA 02 bit 3) is set. High speed Massbus transfers are usually terminated in the first case for device write operations and in the second case for device read operations. MSTR WOR END XFER is not asserted by the Massbus interface.
1	RO	-SLVE WOR END XFER (NSEXFR) = not slave end transfer. SLVE WOR END XFER L is communicated between the data path interface and channel bus interface over a wire-ORed bus which can be pulled low (asserted) by either interface to signal termination of a channel bus transfer. SLVE WOR END XFER is asserted in the following cases: by the data path interface when the BC OF FLAG flip-flop is set; or by the channel bus interface when the device signals termination over the channel bus. During normal operation, high speed channel transfers are terminated in the first case for device write operations and in the second for device read operations.
0	NU	Read as logical 0.

4.4.4.6 UBRA 04: Massbus Counter - Low Byte (MCLO) - Register 4 provides microcontroller access to the low byte of the Massbus counter which is incremented every time a MSTR ACK is issued in response to a MSTR REQ. The low byte of the counter may be read and written as UBRA 04.

Bit	Type	Description
7-0	R/W	MC 07 - MC 00 (MC07 - MC00) = Massbus counter bits 07 - 00.

4.4.4.7 UBRA 05: Massbus Counter - High Byte (MCHI) - Register 5 provides microcontroller access to the high byte of the Massbus counter. The high byte of the counter may be read and written as UBRA 05.

Bit	Type	Description
7-0	R/W	MC 15 - MC08 (MC15-MC08) = Massbus counter bits 15 - 08.

4.4.4.8 UBRA 06: Byte Counter - Low Byte (BCLO) - Register 6 provides microcontroller access to the low byte of the byte counter which is incremented every time a SLVE ACK is issued in response to a SLVE REQ. The low byte of the counter may be read and written as UBRA 06.

Bit	Type	Description
7-0	R/W	BC 07 - BC 00 (BC07 - BC00) = byte counter bits 07 - 00.

4.4.4.9 UBRA 07: Byte Counter - High Byte (BCHI) - Register 7 provides microcontroller access to the high byte of the byte counter. The high byte of the counter may be read and written as UBRA 07.

Bit	Type	Description
7-0	R/W	BC 15 - BC 08 (BC15-BC08) = byte counter bits 15 - 08.

4.4.4.10 UBRA 10: Data Formatter Control ROM Address Register **(DFRMAD)** - Register 10 provides microcontroller access to the data formatter control ROM address register which points to the next ROM instruction that will be executed if the data path interface is enabled (DX HIGH SPEED = 1). Bits 7 - 0 of the ROM address may be read and written as UBRA 10; bit 8 is read and written as UBRA 02 Bit 0. DX HIGH SPEED (UBRA 01 bit 0) and CLK PH 0 (UBRA 16 bit 4) must be low to write UBUS data into ROM address bits 3-0. If DX HIGH SPEED = 1 when UBRA 10 is written, the contents of bits 19 - 16 (ROM 19 - ROM 16) of the ROM location addressed by ROM ADR 8 - ROM ADR 0 prior to clocking UBRA 10 will be written into bits 3 - 0 and UBUS data will be written into bits 7 - 4. (See section 4.4.4.11 for more detail). Bits 8 - 4 are manipulated only by the microcontroller; whereas bits 3 - 0 are altered by the formatter as the ROM instructions are executed. (This is a consequence of the fact that the ROM programs are confined within 16 word boundaries.)

Bit	Type	Description
7-0	R/W	ROM ADR 7 - ROM ADR 0 (RMADR7 - RMADR0) = ROM address bits 7 - 0.

4.4.4.11 UBRA 11: ROM Data Register - Low Byte (RMDALO) - Register 11 is a diagnostic register which allows diagnostic programs to read the contents of the data formatter control ROM over the microbus. The contents of the ROM location addressed by ROM ADR 8 - ROM ADR 0 is loaded into the ROM data register when UBRA 11 is written (UBUS data is ignored.) provided CLK PH 0 (UBRA 16 bit 4) is low prior to writing UBRA 11, CLK PH 0 may become permanently set after writing UBRA 11, inhibiting further clocking of the ROM data register, unless certain precautions are taken (see step (1) in recommended procedure below). When UBRA 11 is written, ROM output bits 15 - 08 (ROM 15 - ROM 08) are loaded into UBRA 12.

The recommended procedure for examining the contents of the data formatter control ROM is given below.

1 Set up the data path interface so that CLK PH 1 (UBRA 16 bit 5) = 0, CLK PH 0 (UBRA 16 bit 4) = 0, -RUN (UBRA 16 bit 3) = 1, -CLR RUN (UBRA 16 bit 2) = 0, and BASE CLK EN (UBRA 01 bit 1) = 0.

2 With DX HIGH SPEED (UBRA 01 bit 0) = 0, load ROM ADR 8 - ROM ADR 0 with the address of the ROM location to be examined by writing UBRA 02 bit 0 and UBRA 10 bits 7 - 0.

3 Write UBRA 11 to load ROM 15 - ROM 00 into UBRA 11 and UBRA 12.

4 With DX HIGH SPEED = 1, write UBRA 10 to load ROM 19 - ROM 16 into UBRA 10 bits 3 - 0.

5 Read UBRA 10 - 12 to examine the contents of the ROM location.

6 Go to step (2) if more locations are to be examined.

The ROM data bits have mnemonic names associated with them which are indicative of their function. The names are given below for completeness and will be described in more detail in Section 5.3.4.5.

Bit	Type	Desription
7-4	RO	MASK 3 - MASK 0 (RMDA 07 - RMDA 04) = mask bits 3 - 0 (ROM data 07 - 04).
3-0	RO	CC 3 - CC 0 (RMDA 03 - RMDA 00) = cycle control bits 3 - 0 (ROM data 03 - 00).

4.4.4.12 UBRA 12: ROM Data Register - High Byte (RMDAHI) - Register 12 is a diagnostic register which allows diagnostic programs to read the contents of the data formatter control ROM over the microbus. See Section 4.4.4.11 for more detail. The mnemonic names associated with the ROM data bits are given below and described in more detail in Section 5.3.4.5. Writing UBRA 12 provides the additional diagnostic capability of single stepping the formatter clock phases. With BASE CLK EN (UBRA 01 BIT 1) = 0, BASE CLK is held high and is pulsed low once each time UBRA 12 is written.

Bit	Type	Description
7-4	RO	SHIFT 3 - SHIFT 0 (RMDA 07 - RMDA 04) = shift bits 3 - 0 (ROM data 07 - 04).
3-0	RO	MASK 7 - MASK 4 (RMDA 03 - RMDA 00) = mask bits 7 - 4 (ROM data 03 - 00).

4.4.4.13 UBRA 13: Assembly Register - Low Byte (ARLO) - Register 13 allows diagnostic diagnostic programs to read bits 7 - 0 of the assembly register (AR) over the microbus. The assembly register is an 18-bit register in which words are disassembled into bytes for transmission to the channel bus interface during device write operations or in which bytes are assembled into words for transmission to the Massbus interface during device read

operations. Writing UBRA 13 generates a pulse on HSDP INIT which initializes the data path interface for a new data transfer. The logic level on the DATA TO DEV line must be established for the direction of transfer desired before writing UBRA 13 because flip-flops in the formatter are either set or cleared when HSDP INIT goes high, depending on the level of DATA TO DEV.

Bit	Type	Description
7-0	RO	AR 07 - AR 00 (AR07 - AR00) = assembly register bits 07 - 00.

4.4.4.14 UBRA 14: Assembly Register - High Byte (ARHI) - Register 14 allows diagnostic programs to read bits 15 - 8 of the assembly register (AR) over the microbus. The assembly register is an 18-bit register in which words are disassembled into bytes for transmission to the channel bus interface during device write operations or in which bytes are assembled into words for transmission to the Massbus interface during device read operations. Writing UBRA 14 provides the diagnostic capability of directly setting the RUN flip-flop in the formatter.

Bit	Tyep	Description
7-0	Ro	AR 15 - AR 08 (AR 15 - AR 08) = assembly register bits 15 - 08.

4.4.4.15 UBRA 15: Register 15 (REG15) - Register 15 contains 7 read only diagnostic bits (6-0). Writing UBRA 15 provides the diagnostic capability of directly setting the MSTR REQ HLDOFF flip-flop.

Bit	Type	Description
7	NU	Read as logical 0.
6	RO	EXTEND RUN (EXTRUN) = extend run. The EXTEND RUN flip-flop is used during the execution of the Industry Compatible - Read Forward program to insure that all bytes transferred over the slave bus will be formatted into words and transferred to the Massbus interface over the master bus.
5	RO	LD SB (LDSB) = load silo buffer. The falling edge of LD SB clocks the silo buffer (SB) register and sets the MSTR RDY flip-flop. The silo buffer register is loaded from the master receiver register (MRR) for device write operations or from the assembly register (AR) for device read operations.
4	RO	LD CB (LDCB) = load channel buffer. The falling edge of LD CB clocks the channel buffer (CB) register and sets the SLVE RDY flip-flop. The

		channel buffer register is loaded from the slave receiver register (SRR) for device read operations or from the assembly register (AR) for device write operations.
3	RO	-EN MUX/DEMUX (NENMDM) = not enable data multiplexer/demultiplexer. When the EN MUX/DEMUX flip-flop is set, the data multiplexer in the formatter is enabled during device write operations or the data demultiplexer is enabled during device read operations.
2	RO	-CLR AR (NCLRAR) = not clear assembly register. When CLR AR is asserted, all the bits in the assembly register (AR) are cleared.
1-0	RO	AR 17 - AR 16 (AR17-AR16) = assembly register bits 17-16.

4.4.4.16 UBRA 16: Register 16 (REG16) - Register 16 contains 7 read only diagnostic bits (6 - 0). Writing UBRA 16 provides the diagnostic capability of directly setting the SLVE REQ HLDOFF flip-flop.

Bit	Type	Description
7	NU	Read as logical 0.
6	RO	-RUN DATA (NRNDAT) = not run data. RUN DATA represents the logic level which will be clocked into the RUN flip-flop on the rising edge of BASE CLK, provided CLR RUN = 0.
5	RO	CLK PH 1 (DFCPH1) = clock phase 1, CLK PH 1 is the second phase of the clock derived in the formatter time base from the BASE CLK signal. It is asserted during timing states 2 and 3.
4	RO	CLK RH 0 (DFCPH0) = clock phase 0. CLK PH 0 is the first phase of the clock derived in the formatter time base from the BASE CLK signal. It is asserted during timing states 1 and 2.
3	RO	-RUN (NOTRUN) = not run. When the RUN flip-flop is set, the formatter time base is allowed to advance its clock phases from state 0 (CLK PH 1 = 0, CLK PH 0 = 0) to state 1 (CLK PH 1 = 0, CLK PH 0 = 1) if BASE CLK EN = 1. The time base will cycle through states 2 and 3 back to state 0, in which it will remain unless the RUN flip-flop is still set. One cycle through the 4 clock states executes one ROM instruction.
2	RO	-CLR RUN (NCLRRN) = not clear run. When the CLR

RUN flip-flop is set, the RUN flip-flop is cleared and held cleared.

1	RO	MSTR PE (MSTRPE) = master parity error. MSTR PE is asserted when incorrect (even) parity is detected for data stored in the silo buffer (SB) register.
0	RO	SLVE PE (SLVEPE) = slave parity error. SLVE PE is asserted when incorrect (even) parity is detected for data stored in the channel buffer (CB) register.

4.4.4.17 UBRA 17: Register 17 (REG17) - Register 17 contains 6 read only diagnostic bits (5 - 0). Writing UBRA 17 provides the diagnostic capability of directly clearing the EXTEND RUN flip-flop.

Bit	Type	Description
7-6	NU	Read as logical 0.
5	RO	MSTR RDY DLY 2 (MSRDY2) = master ready delay 2. The MSTR RDY DLY 2 flip-flop is a second stage synchronizer for the MSTR RDY flip-flop. When MSTR RDY DLY 2 is set, the data path interface is reeady to issue a MSTR ACK in response to a pending or forthcoming MSTR REQ and the formatter will stop executing instructions when it encounters an instruction which services a MSTR REQ until MSTR RDY DLY 2 goes low. When HSDP INIT is asserted, MSTR RDY DLY 2 is set if DATA TO DEV = 1 or cleared if DATA TO DEV = 0.
4	RO	MSTR RDY DLY 1 (MSRDY1) = master ready delay 1. The MSTR RDY DLY 1 flip-flop is a first stage synchronizer for the MSTR RDY flip-flop. When HSDP INIT is asserted, MSTR RDY DLY 1 is set if DATA TO DEV = 1 or cleared if DATA TO DEV = 0.
3	RO	MSTR RDY (MSTRDY) = master ready. The MSTR RDY flip-flop is set on the falling edge of LD SB and is cleared when MSTR ACK is issued in response to MSTR REQ. When HSDP INIT is asserted, MSTR RDY is set if DATA TO DEV = 1 or cleared if DATA TO DEV = 0.
2	RO	SLVE RDY DLY 2 (SLRDY2) = slave ready delay 2. The SLVE RDY DLY 2 flip-flop is a second stage synchronizer for the SLVE RDY flip-flop. When SLVE RDY DLY 2 is set, the data path interface is ready to issue a SLVE ACK in response to a pending or forthcoming SLVE REQ and the formatter will stop executing instructions when it encounters an

instruction which services a SLVE REQ until SLVE RDY DLY 2 goes low, when HSDP INIT is asserted, SLVE RDY DLY 2 is set if DATA TO DEV = 0 or cleared if DATA TO DEV = 1.

1	RO	SLVE RDY DLY 1 (SLRDY1) = slave ready delay 1. The SLVE RDY DLY 1 flip-flop is a first stage synchronizer for the SLVE RDY flip-flop. When HSDP INIT is asserted, SLVE RDY DLY 1 is set if DATA TO DEV = 0 or cleared if DATA TO DEV = 1.
0	RO	SLVE RDY (SLVRDY) = slave ready. The SLVE RDY flip-flop is set on the falling edge of LD SB and is cleared when SLVE ACK is issued in response to SLVE REQ. When HSDP INIT is asserted, SLVE RDY is set if DATA TO DEV = 0 or cleared if DATA TO DEV = 1.

4.4.4.18 UBRA 20-UBRA 37: Not Used - Registers 20-37 are not implemented in the data path interface. If accessed by the microcontroller, they will be read back as zero data. DATO cycles to these registers will have no effect on the interface.

5.1 INTRODUCTION

5.1.1 General

The DX20 Programmed Device Adapter (PDA) operates as an independent I/O processor transferring data between high speed peripheral devices, such as magnetic tape units, and data storage areas within (or external) to the KL10 system. Communication and data transfer operations between the PDA and KL10 memory are performed over the Massbus via the RH20 Massbus controller. Up to eight peripheral devices, such as the DX20, can be addressed over the Massbus. Data and control between the DX20 and device control unit(s) is carried out over the system channel bus. The required interface between the DX20 internal buses and the channel bus is implemented through special input/output formatting, transfer logic, and control logic. Data/control interfacing between the channel bus and the selected tape drive is accomplished via the device control unit which contains the logic to select, disconnect, and format data or status.

The microbus within the DX20 provides communication paths between the microcontroller and: Massbus interface, high speed data path, and device interface. Control, channel bus interface (CBI) conditions, and paramater information transferred to or from the CBI takes place over the microbus.

Interconnection between the device interface logic and the high speed data path consists of data and control lines referred to as the slave bus. The high speed data path and Massbus communicate over the master bus. A diagnostic bus (link) connects the microcontroller and Massbus interface. This bus provides a path for loading the microcode data from the KL10 to the microstore, and for diagnostic information contained in the microcontroller registers to be transmitted to the KL10.

5.1.2 Subsystem Description

The DX20 subsystem consists of the: Massbus, RH20 Massbus controller, channel bus, and device controller. The Massbus provides a bidirectional communication path between the DX20 and RH20 controller on which microcode data from the KL10 processor is transmitted to the DX20 microstore. This data is used to perform data transfer operations between the DX20 and a storage device. Data from a device is obtained from the device controller, via the channel bus, processed in the DX20 and placed on the Massbus then transmitted to the RH20. Conversely, data is transferred from the RH20 onto the Massbus to the DX20 data paths, and onto the channel bus to the device controller.

5.1.2.1 Massbus - The Massbus provides the interface between the RH20 controller and the DX20 programmed device adapter. It is composed of two independent buses; a control bus and a data bus, permitting both synchronous and asynchronous communication.

The data bus contains the bidirectional parallel data bits (including parity) and its associated control lines. The asynchronous control bus contains the parallel control and status path (16 data and 1 parity) and its associated bus control lines.

5.1.2.2 RH20 Massbus Controller - The RH20 Massbus controller (MBC) provides a high-speed synchronous/asynchronous data transfer path between the KL10 memory system and the DX20. Up to eight Massbus-compatible devices may be connected to the MBC.

5.1.2.3 Channel Bus - Data and control information is transferred between the DX20 and peripheral device over the channel bus. Data issued to the device is transferred on 9-bit bus out lines, and data being issued to the DX20 is received on 9-bit bus in lines. Identification of information on either bus, and control of interface connect/disconnect sequences is performed on tag lines (in or out depending on the direction).

5.2 INTERFACE DESCRIPTIONS

5.2.1 Massbus and Diagnostic Link

The Massbus can logically be divided into two sections, the asynchronous section, referred to as the control bus, and the synchronous section referred to as the data bus (see Figure 5-1). Physically, these two sections are combined into a single bus comprising 56 signals. The Massbus utilizes differential transmitters and receivers for signaling between the Massbus controller and a Massbus peripheral device such as the DX20. A maximum of eight peripheral devices can be addressed over the Massbus.

The control bus section has a 16-bit wide data path with an odd parity bit and 14 control lines for a total of 31 lines. It is used for transmitting status and control information between the Massbus controller and the device. It is asynchronous in the sense that each data transfer requires a handshake consisting of a request signal from the controller (DEM) and a response signal from the device (TRA). The controller can address up to 32 registers in each device over the control bus.

The data bus section has an 18-bit wide data path with an odd parity bit and 6 control lines for a total of 25 lines. It is used for transmitting high-speed data between the controller and the device. It is synchronous in the sense that no response is required from the controller once a data transfer has been initiated. A device read operation occurs with the device generating a synchronous clock (SCLK) indicating to the controller that valid data is on the bus.

For a device write operation the controller retransmits this signal as a write clock (WCLK) indicating to the device that valid data is on the bus.

Figure 5-1 DX20 Bus Interface

The diagnostic link from the Massbus to the microcontroller is provided to allow the master-host system access to the microcontroller via the control bus. It comprises a 16-bit wide data path with an odd parity bit and 11 control lines for a total of 28 lines (see Figure 5-2). Most of these signals (23) are derived from Massbus control bus lines. This link is used for loading the operational microcode or diagnostics into the microstore to start and stop the microcontroller, initiate single cycle instruction, and to examine several points within the microcontroller or other microbus intefaces. In a Massbus-to-Massbus configuration, the Massbus interface connected to the slave-host system will not have access to the microcontroller via the diagnostic link even though the interface has the same logic.

The DX20 uses three standard M5903 Massbus transceiver modules for interfacing to the Massbus. The modules are connected to the Massbus via 6 flat 40-conductor cables (BC06R) internal to the cabinet. These cables connect to the Massbus input/output transition connectors. The other side of these connectors are connected to the Massbus controller and/or other devices via a heavy duty round cable (BC06S) containing 60 twisted pair. If the DX20 is on the end of the Massbus, the Massbus output connector must be terminated with the standard terminator assembly #7009938.

The M5903 transceiver modules utilize line drivers with TTL-inputs and tri-state differential-outputs. The differential output characteristics are V_{OL} = 0.4 volt (max) at 40 mA and V_{OH} = 2.0 volts (min) at -40 mA. The line receivers are differential -input, TTL-output, devices with a differential input sensitivity of 15 millivolts. They have open-collector outputs with a maximum sink capability of 16 mA at V_{OL} = 0.4 (max) and a maximum high state leakage current of 250 microamps at V_{OH} = 7 volts (max). Each receiver output has a 3.3K-ohm resistor pullup to +5 V on the M5903s except for the 16 control bus data lines and the control bus and data bus parity lines. The pullups for these lines are provided by the device. In the DX20, these pullups are 2K-ohms for the control bus lines, located on the M8604 and 3.3K-ohms for the data bus parity line located on the M8603.

The open-collector output characteristics of the receivers is utilized by the DX20 to form a wired-OR bus internal to the DX20. The diagnostic link is part of this internal bus, which has both tri-state and open-collector devices driving the bus. Logic levels on the internal bus will be a high for a logic one when receiving data from the control bus and a low for a logic one when transmitting to the control bus.

5.2.1.1 Control Bus Signal Definition

DS2-DS0: Device or unit select lines (3) for selecting one of eight possible devices in the Massbus.

RS4-RS0: Register select lines (5) for selecting one of

Figure 5-2 DX20 Block Diagram

32 possible registers in each device.

DEM: Demand line from the controller to the device, indicating that the controller wants to transfer control bus data to or from the device.

TRA: Transfer line from the device to the controller acknowledging DEM.

C15-C00,CPA: The 16-bit bidirectional control bus data path with odd parity between the controller and the device.

CTOD: Controller to device line when asserted indicates the transfer of control bus data is from the controller to the device (control bus write); and when not asserted indicates the transfer of control bus data is from the device to the controller (control bus read).

ATTN: The attention line from the device to the controller when asserted indicates a Massbus device is requesting service from the controller. The controller determines which device(s) is driving the attention by reading the attention summary register.

INIT: The initialize line from the controller to the device when asserted performs a system reset/ initialize function.

FAIL: The fail line from the controller to the device, when asserted indicates the controller has a power failure.

5.2.1.2 Data Bus Signal Definition

RUN: The run line from the controller to the device; when first asserted, indicates the controller is ready to begin a data bus transfer. Thereafter, the device samples the RUN line at the trailing edge of each EBL pulse, and if it is still asserted, the operation normally continues.

OCC: The occupied line from the device to the controller indicating that the device is ready to begin data bus transfer. The assertion of occupied is not dependent on receiving RUN.

SCLK: The sync clock line from the device to the controller. This signal tells the controller that data is available on the data bus for a device read or requests data from the controller for a device write.

WCLK: The write clock line from the controller to the device returns the SCLK from the device, telling the device data is available on the data bus for a device write.

D17-D00,DPA: The 18-bit bidirectional data bus path with odd parity between the controller and the device.

EBL: The end of block line from the device to the controller, when asserted, indicates that the device has completed transferring as many data words as it was requested to transfer by the last read/write command. This line is driven by a one-shot with a 1500 ns minimum pulse width.

EXC: The bidirectional exception line between the controller and the device. This line is asserted by the controller, or the device to indicate that an error condition was detected during a data bus transfer. It should be noted that the RH20 does not drive the exception line.

5.2.1.3 Diagnostic Link Signal Definition

RS2-RS0: The three low-order register select lines from the Massbus control bus. These lines are used in conjunction with DIAG DEM for accessing the diagnostic registers DXDR0 (MBRA 30) - DXDR7 (MBRA 37).

DIAG DEM: This signal is generated in the Massbus interface when a unit has been selected; DEM has been asserted by the controller; and the register address is from 30 to 37.

DIAG DEM DLY: This is a 100 ns pulse (nom.) generated by the Massbus interface when the controller is writing a register in the DX20 and the microprocessor has not been started. Thus, the controller will not be able to write a diagnostic register once the microprocessor is started.

C15-C00,CPA: The 16-bit bidirectional data path from the internal control bus to the microcontroller.

CTOD: The controller to device line. See control bus signal definitions.

MP LINK PRESENT: A level from the microcontroller to the master-host system, when asserted, indicates that the diagnostic link is present.

MP START: This is a signal from the Massbus interface to

the microcontroller to allow the microprocessor clock to run for single cycle or continuous instruction execution. This signal is negated if the Massbus interface detects a microbus parity error.

MP WR EV PA: The microprocessor write even parity line from the Massbus interface to the microprocessor. This flip-flop is set by the host to force even parity generation within the microprocessor.

MP SNGL CYC: The microprocessor single cycle line from the Massbus interface to the microprocessor. This flip-flop is set by the host to allow only one microprocessor timing cycle to occur.

DX RESET: This signal is generated in the Massbus interface, and is used to reset/initialize the DX20. It is generated from either a Massbus INIT; by the Massbus controller writing the DXRES bit in the maintenance register (DXMTR = MBRA 03); or by CROBAR during power up/down sequences.

5.2.1.4 Control Bus Read Sequence - The control bus read cycle begins with the controller asserting the appropriate DS and RS lines and negating the CTOD line. After waiting for the deskew and set up time, and waiting, if necessary, for the TRA line to be negated, the controller asserts DEM. Following a cable delay, the selected device receives the DEM assertion. The controller holds the DS and RS lines constant until the assertion of TRA is received from the device. Not more than 700 ns after receiving DEM the device gates the contents of the selected register onto the C lines, generates CPA, and asserts TRA. After a cable delay the controller receives TRA at which time it can change the DS, RS, and CTOD lines. The controller waits for deskew time, then strobes or gates in the C lines and CPA, and checks for correct parity. It then negates DEM. Following a cable delay the device receives the DEM negation and disables the C lines and negates TRA. After a cable delay the controller receives the negation of TRA, which completes the read cycle.

The control bus read cycle is somewhat modified if the register being read is the attention summary register (MBRA 04). The differences are as follows. If the register address is 04, the device ignores the DS lines. Therefore, any device with its ATA bit set will gate this bit out to the appropriate C line, device 0 onto C00, etc. After asserting DEM, the controller waits 1450 ns, before strobing the C lines to be sure that all devices have had a chance to respond. The controller then strobes the C lines but does not check parity; since it is impossible for any one device to generate a valid parity bit (each device will have generated its own parity, and since it is only driving one bit if ATA is set, the CPA line will always be negated). The controller

then must wait for all devices to negate TRA before beginning the next cycle.

5.2.1.5 Control Bus Write Sequence - The control bus write cycle begins with the controller asserting the appropriate DS and RS lines; asserting the CTOD line; and gating a word onto the C lines. The controller keeps all these lines constant until receiving the assertion of TRA. Following the deskew and set up time, and waiting, if necessary, for the TRA line to be negated from the previous cycle, the controller asserts DEM.

After a cable delay, the selected device receives the DEM assertion and more than 700 ns later the device will have strobed or gated in the C lines; checked for correct parity; and asserted TRA. The controller receives TRA (after a cable delay) at which time it can change DS, RS, CTOD and C lines and negate DEM. Following the cable delay, the device receives the DEM negation and negates TRA, after which the controller receives the negation of TRA, completing the write cycle.

The control bus write cycle is also modified if the register being written is the attention summary register (MBRA 04). The essential difference is that the DS lines are ignored by the device. Each device will gate the appropriate C line to clear its ATA bit, device 0 from C00, etc. After asserting DEM the controller keeps the C lines constant for 1450 ns and then negates DEM. The controller must wait for all devices to negate TRA before beginning the next cycle.

If an illegal (nonexistent) register is selected for a write; or if the GO bit is set, the DX20 will check for correct C bus parity and set the appropriate error bit (ILR or RMR), but otherwise will ignore the C lines. The controller can write the attention summary and maintenance registers while GO is set without getting an RMR.

5.2.1.6 Diagnostic Link Read/Write Sequences - The read/write cycles over the diagnostic link are essentially the same as the control bus timing.

5.2.1.7 Data Bus Read Sequence - The data bus read cycle is initiated when the controller loads a read command into the control register of the device via the control bus. If the device determines the command is valid, it enables the D lines and asserts OCC. Not more than 100 microseconds after loading the command the controller asserts RUN which is issued to the device. When the device has the first word it gates it onto the D lines and asserts SCLK. After the controller receives the SCLK assertion and following a time that is approximately 60 percent of the nominal SCLK period, the device negates SCLK. The controller then receives the SCLK negation at which time it strobes the D lines and checks for correct parity. If there is more data to be read; after a time that is not less than the remaining time of the nominal SCLK period, the device gates the next word onto the D

lines and asserts SCLK. This sequence continues until the last word of the transfer. The device then asserts EBL, which is issued to the controller. At this time the controller must decide whether or not to have the device continue the read operation, without disconnecting from the data bus (the controller may have already negated RUN). If the controller decides it doesn't want to continue, it negates the RUN line not later than 500 ns after receiving EBL. After a cable delay the device receives the RUN negation. Not less than 1500 ns after asserting EBL, the device negates EBL and samples the RUN line. If RUN has been negated, the device disables the D lines and negates OCC. After the controller receives the EBL negation another data transfer can be started.

5.2.1.8 Data Bus Write Sequence - The data bus write cycle is initiated when the controller loads a write command into the control register. If the device determines that the command is valid, it enables the data bus receivers and asserts OCC. Not more than 100 microseconds after loading the command, the controller has gated the first word onto the D lines and asserted RUN. After a cable delay, the device receives the RUN assertion. When the device is ready to accept the first word it asserts SCLK. The controller receives the SCLK assertion and then asserts WCLK which is issued to the device. The device receives the WCLK assertion and strobes the data from the D lines and checks parity. After a time that is approximately 60 percent of the nominal SCLK period the device negates SCLK. The controller receives the SCLK negation, then negates WCLK and gates the next word onto the D lines. After a delay the device receives the WCLK negation. If more words are to be written; then after a time that is not less than the remaining time of the nominal SCLK period; the device again asserts SCLK, and the above sequence is repeated. After the negation of SCLK for the last word to be transferred, the device asserts EBL. From this point, the termination of the transfer is the same as for a data bus read operation.

5.2.1.9 Error Handling - Error handling in the DX20 is normally performed by the microprocessor, under control of the microcode. There are several conditions where the hardware provides error handling. One of these is the setting of ATA which occurs either on the trailing edge of EBL and the microprocessor has stopped, or if OCC (occupied) is not set and the microprocessor stops. A second condition occurs with the setting of ATA on the following: if GO is not set and a control bus parity error is detected by the Massbus interface, or, the controller accesses an illegal register.

In the case of EXC (exception), all error conditions that cause composite error to be asserted will cause EXC to be set if OCC is set. Under the control of the microprocessor, an EBL can be generated to terminate the transfer. If the microprocessor stops while OCC is set, this will set EXC, which in turn will generate an EBL. On the trailing edge of EBL; OCC and EXC will be cleared (and GO, if it has not been cleared previously) and ATA will be

set, assuring a proper termination sequence.

5.2.2. Massbus Length and Throughput Considerations

The maximum length of the Massbus is determined by several factors, including; control bus timing requirements; system throughput requirements; and any limitation imposed by the particular Massbus controller the DX20 is connected to.

The control bus timing limits the effective length (the sum of cable lengths used plus loading effects of multiple devices) of the Massbus to 160 feet, assuming a total propagation delay of 375 ns. The total propagation delay is based on a delay of 2 ns/ft of cable plus 30 ns max. transmitter delay and 25 ns max. receiver delay.

Calculating the length of the Massbus on the basis of system throughput requirements is quite difficult. The following calculations assume a device write operation, and a data path response of 80 ns (this latter time may be arbitrarily long depending on the transfer rate of the slave device). A device write presents the worst case, since an SCLK, issued by the DX20, propagates along the bus to the controller, and then back along the bus as WCLK to the DX20. Assuming the longest possible Massbus, this could take 750 ns (2 x 375). Before another SCLK can be generated, the DX20 issues a MSTR REQ to the data path and must receive a MSTR ACK. This sequence, including other internal delays is as follows:

WCLK to MSTR REG	60 ns
MSTR REQ to MSTR ACK	80 ns
MSTR ACK to SYNC ENA*	60 ns
SYNC ENA to SCLK	116 ns (2 x 58)
TOTAL	316 ns

* SYNC ENA is not really a signal, but an intermediate point between receiving MSTR ACK and generating SCLK. This is done to separate fixed delays from time base figures.

Thus, the total time between SCLK's would be 750 + 316 = 1066 ns. This implies that the maximum rate of transfer over the Massbus would be 0.938 MHz (1066 ns per 18-bit word). However, due to the discrete SCLK values available, the closest to this rate is 0.823 MHz (1215 ns). Even though the sequence would be completed in 1066 ns, another 149 ns would have to elapse before another SCLK could be generated. Therefore, for SCLK periods greater than 1066 ns, reducing the length of the Massbus will not improve the transfer rate. This example is used only for determining the point at which cable length becomes a factor in throughput calculations.

For SCLK periods less than 1066 ns a reduction in the length of the Massbus is necessary to maintain the maximum rate of transfer over the Massbus. The specified maximum rate of transfer is 2 MHz or 500 ns per 18-bit transfer (see restrictions below). Again, because of discrete time rates, the nearest to this value is 1.919 MHz or 521 ns. To find the maximum cable length, the internal delays discussed above, plus transmitter/receiver delays, must be subtracted from this figure:

SCLK PERIOD	521 ns
INTERNAL DELAYS	-316 ns
TRANS/RECR DELAYS	-110 ns
REMAINING FOR CABLE DELAY	95 ns

Remembering that this is a two-way cable delay and at 2 ns/ft, the maximum cable length would be 95/4=24 feet. The above calculation leads to the following formula:

$$\text{Max. Cable Length} = \frac{\text{SCLK PERIOD} - 426}{4}, \quad \text{SCLK PERIOD} \leq 1066 \text{ ns}$$

The restriction is required since SCLK periods greater than 1066 ns would indicate a cable length greater than 160 feet, which is the limit set by the control bus. The graph in Figure 5-3 shows Massbus cable length versus SCLK period. The variation in SCLK period and cable length is shown as being continuous, where in reality, both have discrete increments.

The maximum rate of transfer over the Massbus is restricted further, due to current design limitations in the RH20. It is expected that this limitation will be eliminated in the near future. The current maximum transfer rates are as follows in Table 5-1.

Table 5-1 Massbus Transfer Rates

System Clock (MHz)	RH20 Port Priority	Min. SCLK Period(ns)	Nearest SCLK Period(ns)	Transfer Rate(MHz)	% Below Max.Rate
25	Low	1695	1736	0.576	2.4
25	High	1130	1215	0.823	7.5
29	Low	1483	1563	0.640	5.4
29	High	989	1042	0.960	5.4
Massbus spec		500	521	1.919	4.2

Figure 5-3 Massbus Length vs. SCLK Periods

It is apparent that until the RH20 restriction is overcome, the Massbus cable length is not a limiting factor, except in the case of a 29MHz CPU, and a high priority port.

Table 5-2 lists the available SCLK periods. These periods assume that there is enough time after the negation of SCLK to complete the data path handshake before the full SCLK period has elapsed. If this is not the case, the period of the SCLK will be extended, but not the time that SCLK is asserted. The period of the SCLK is adjusted by setting four switches on the M8603 Massbus data board. The times listed assume a 17.28 MHz crystal oscillator.

Table 5-2 DX 20 SCLK Periods

Switch Setting (OCTAL)	SCLK Period (ns)	Transfer Rate (MHz)	Duty Cycle (%)
0	2951	0.339	64.7
1	2778	0.360	64.6
2	2604	0.384	64.4
3	2431	0.411	64.3
4	2257	0.443	64.1
5	2083	0.480	63.9
6	1910	0.524	63.6
7	1736	0.576	63.4
10	1563	0.640	63.0
11	1389	0.720	62.5
12	1215	0.823	61.9
13	1042	0.960	61.1
14	868	1.152	60.0
15	694	1.441	58.4
16	521	1.919	55.5
17	00	0	100.0

5.2.3 Channel Bus Interface

5.2.3.1 Introduction - The channel bus interface provides a control and data link between the slave device and the DX20. Logic in the interface handles communication between the microcontroller and the slave device, and high speed data transfers between the data path interface and the device. The channel bus interface provides the hardware necessary to interface to the IBM-style channel bus and implement the protocols required of a channel controller or a control unit on the bus. The signal name conventions used in this document and in the print sets for the boards in the channel bus interface assume that the DX20 is acting as a channel controller with one or more devices connected to it via the channel bus and appropriate control units. When the DX20 is acting as a control unit, some of the signals are redefined and their names changed; these reassignments will be noted. Physically, the logic for the channel bus interface is contained on the M8607 channel bus board, the optional M8608 channel extension board, and the G891 power fail and select bypass module, which plug into slots AF17, AF29, and C16, respectively, of the section II backplane. The M8608 is optional and can be added to provide the extended bus feature.

5.2.3.2 Simplified Block Diagram - A simplified block diagram for the channel bus interface is shown in Figure 5-4. The channel bus interface communicates with the rest of the system over three bidirectional buses: the microbus (UBUS), the slave bus (SLVE), and the channel bus (IBCB). Control, status, sense and diagnostic information and data can be transferred over the microbus to the M8602 Microprocessor Board under control of the microcontroller. the microbus interface is organized as fifteen 8-bit registers; eleven are implemented on the M8607 Channel Bus Board (UBRA 00-11, 16 octal) and the remaining four are implemented on the M8608 Channel Extension Board (UBRA 12-15 octal). Control, status, and sense information and data are communicated between the device and the channel bus interface over the channel bus. Channel data is transferred between the channel bus interface and the data path interface over the slave bus as 6, 7, or 8 bit bytes, depending on the data format. The microbus, slave bus, and channel bus are described in detail in Section 5.2.

The channel bus consists of 61 unidirectional signal lines (20 are optional with the extended bus feature) which are bussed in parallel to all control units on the bus, with the exception of SEL IN and SEL OUT which are run serially through the control units. In general, the term 'out' in connection with the channel bus denotes outward signal propagation from the DX20 when it is acting as a channel controller and 'in' denotes inward signal propagation to the DX20. Signals are also classified as either 'bus' lines which carry data, commands, addresses, status, and sense information or 'tag' lines which identify and control information flow on the BUS lines. Normally, the TAG IN and TAG OUT lines are used with the bus 0 in and bus 0 out lines to communicate in single bus mode with the control units on the bus.

Figure 5-4 Channel Bus Interface - Simplified Block Diagram

To increase throughput, however, the width of the bus can be doubled by adding the M8608 channel extension board which implements bus 1 in and bus 1 out to permit the dual bus mode of operation. Most of the data path logic is duplicated on the M8608.

The bus out lines may be driven by either the output of the device register (DR) in the slave bus transceiver or the bus out register (BOR). the DR buffers data transferred to and from the data path interface over the slave bus and is involved in high speed data transfers over the bus lines. The BOR is used to transmit control information over the channel bus and can be loaded by the microcontroller via the microbus directly or indirectly through the 16-word scratch pad. The bus in lines can be read by the microcontroller over the microbus, or the data can be transferred to the data path interface over the slave bus for high speed transfers. The direction of data transfer is established by the logic level on the DATA TO DEV H line (+3 V = device write, GND = device read) from the Massbus data board. High speed transfers are indicated by the assertion of DX HIGH SPEED L (GND = asserted) by the data path interface. The channel bus control (CB CTRL) logic provides limited response by the channel bus interface to events for which real-time constraints prevent direct microcontroller control and handles the request-acknowledge handshake sequences required to transfer data between the data path interface and the slave device. Additional control unit control (CU CTRL) logic provides features required for operation as a control unit.

The G891 power fail and select bypass module performs two functions: it provides a CROBAR H signal which is grounded when power fails; and electromechanically bypasses the SEL IN or SEL OUT signal, which is serially propagated by selection logic in the control units, in the event power fails or the DX20 goes off line.

5.2.3.3 Data Transfer Rate

Introduction - An important characteristic of the DX20 is its high speed transfer rate. The maximum transfer rate that can be supported over the channel bus can only be expressed in general terms due to the many interdependent factors involved in data transfers between the host and slave device. One significant aspect of the protocol used to transfer data over the channel bus is the DC interlocked request-acknowledge sequence of the associated tag lines. The device requests bus transfers as specified by the requirements of the storage medium and, in the case of constant motion devices such as magnetic tape and disk units, failure of the channel controller to respond to a request in the proper time causes an overrun condition. Consequently, the maximum inherent transfer rate capability of any subsystem including the DX20 must exceed the highest transfer rate required by the device. Factors which influence the maximum possible transfer rate are discussed in the following sections which are organized hierarchically to reveal the degree to which system

components affect the throughput capability of the system. The discussion assumes a system configuration in which the DX20 is acting as a channel controller with one or more TX01 or TX02-type control units attached to the channel bus.

Inherent Channel Transfer Rate of the Channel Bus Interface - If there were no time delay in the request-acknowledge response for data transfers over the slave bus or channel bus, that is, SLVE ACK is issued immediately in response to SLVE REQ and control unit responses on the DAT IN and SRV IN lines to activity on the DAT OUT and SRV OUT lines are instantaneous, the channel bus interface will transfer data at its inherent speed, limited only by its own internal delays. The maximum time to cycle through the transfer of one byte in single bus mode is 400 nanoseconds, which corresponds to a transfer rate greater than 2.5 megahertz (2.5 megabytes per second). In dual bus mode, two bytes must be obtained serially from the data path interface for parallel transmission over BUS 0 and BUS 1 to the device during device write operations and vice-versa for read operations. The additional slave bus cycle adds another 185 nanoseconds to the total cycle, raising it to a maximum of 400 + 185 = 585 nanoseconds, which corresponds to a transfer rate in the dual bus mode greater than 1.7 megahertz (3.4 megabytes per second).

Inherent Channel Transfer Rate of the DX20 - The inherent transfer rate of the DX20 treated as a complete unit is a function of the Massbus interface, the data path interface, and the channel bus interface. The calculation will be made assuming immediate request-acknowledge responses over the channel bus, and the Massbus interface transfer rate is not a limiting factor. If the Massbus interface is allowed to transfer data at rates up to 1.2 megahertz by appropriate set up of the interface (See Section 5.2.4.4), the transfer rate will not be Massbus limited for any of the data formats implemented in hardware (See Section 5.2.4.3). In single bus mode, a SLVE REQ will be honored within 80 nanoseconds by a SLVE ACK, thus increasing the maximum cycle time of the channel bus interface to 400 + 80 = 480 nanoseconds, which corresponds to a channel transfer rate greater than 2.0 megahertz (2.0 megabytes per second). In the dual bus mode, however, the channel bus interface must make two requests to the data path interface during each transfer cycle. The second request may occur in as little as 185 + 30 = 215 nanoseconds after the first, causing greater latency in servicing the second request because of the time the data path interface takes to format the second byte. Assuming all worst-case conditions (including worst data format with requests made to the data path interface at the worst possible time), the first request will be honored within 80 nanoseconds and the second within 165 nanoseconds, increasing the maximum cycle time of the channel bus interface in dual bus mode to 585 + 80 + 165 = 830 nanoseconds, which corresponds to a channel transfer rate greater than 1.2 megahertz (2.4 megabytes per second).

Channel Transfer Rate of the DX20 with a Control Unit - The maximum channel bus transfer rate that can be supported by the system is dependent on the Massbus transfer rate, timing in the DX20 and control unit, and the length of the channel bus cable. The results of an analysis of the performance of the tape subsystem with regard to the Massbus transfer rate, the channel bus transfer rate, and cable length are shown in Figure 5-5 for TX01 and TX02-type control units. For the case in which the channel bus transfer rate is Massbus limited, the transfer rate is averaged over four bytes, so the analysis is applicable for control units such as the TX01 and TX02 which buffer at least four bytes of data.

TX01 Control Unit - The timing diagram for a TX01-type control unit shows the DC interlocked handshake of the SRV IN (request) and SRV OUT (acknowledge) tag lines used to transfer data over the bus lines. The diagram depicts the limiting case in which the capacity of the channel bus has been reached. A transfer cycle can be divided into four time intervals: T1A which represents the time the channel controller takes to assert SRV OUT after the rising edge of SRV IN; T1B which represents the time the channel controller takes to drop SRV OUT after the falling edge of SRV IN; T2A which represents the time the control unit takes to drop SRV IN after the rising edge of SRV OUT; and T2B which represents the time the control unit takes to assert SRV IN to begin the next transfer cycle after SRV OUT falls. T1 = T1A + T1B represents the total delay introduced by the channel controller during a transfer cycle and T2 = T2A + T2B represents the total delay introduced by the control unit. Additionally, TC represents the round trip time required for a signal to travel the length of the channel bus cable and back. The timing diagram shows signals as viewed at the channel controller, so TC is grouped with T2A and T2B.

For the case in which the channel bus transfer rate is not Massbus limited (SCLK rate not limited to less than 1.2 megahertz) and a TX01 is transferring data at the maximum rate that the DX20 will support, T1A will be less than 480 - 50 = 430 nanoseconds for the single bus mode. Graphs in Figure 5-5 show the maximum guaranteed data rate as a function of cable length for different values of TD, where TD = T2, for single bus mode and dual bus mode. The permissible data rates are bounded by the 200 foot maximum allowed cable length and the line corresponding toi the maximum value of T2 for the control unit.

The case in which the channel bus transfer rate is limited by the maximum allowable Massbus transfer rate, interdependent factors which are a function of the data format, are involved in the transfer rate calculation. It is possible, however, to bound the maximum channel bus transfer rate with a guaranteed value which is a simple function of the channel bus cable length, the maximum allowed Massbus transfer rate, and the inherent delay of the control unit. If the maximum Massbus transfer rate is not limited to less than 1.2 megahertz, the channel bus transfer rate will not be Massbus limited and the preceding analysis applies. If

Figure 5-5 Single and Dual Handshake Timing

limitations of the Massbus controller, however, necessitate reducing the transfer rate to less than 1.2 megahertz (830 nanoseconds per transfer) by appropriate setting of switches in the Massbus interface (See Section 5.2.4.4), the maximum channel bus transfer rate that can be supported for a given control unit and channel bus cable length may drop. TM, the minimum period allowed by the Massbus interface for a Massbus transfer cycle, has been included in the expression which bounds TD in Figure 5-5. The additional term RX(TM - 830 NS) in the expression for TD assumes that the additional time over the 830 nanosecond limiting value required to complete a Massbus transfer is not hidden and, therefore, must be added onto the following channel bus transfer cycle, but since every Massbus word contains at least two bytes, the added time (TM = 830 NS) can be averaged over a minimum of two bytes. Consequently, when operating in single bus mode with one byte transferred per bus cycle, R = 0.5, and when operating in dual bus mode with two bytes transferred per bus cycle, R = 1.0. The expression for TD, then includes terms representing delays caused by the control unit and Massbus transfer rate limitations.

TX02 Control Unit - The timing diagram for TX02-type control units shows the dual handshake protocol used over the channel bus. The SRV IN (request) and SRV OUT (acknowledge) tag lines form one pair of interlocked signals and the DAT IN (request) and DAT OUT (acknowledge) tag lines form a second pair. The control unit alternates SRV IN and DAT IN, each tag line initiating a bus transfer. The diagram depicts the limiting case in which the capacity of the channel bus has been reached. A transfer cycle can be divided into two time intervals: T1 which represents the time the channel controller takes to assert DAT OUT after SRV IN falls or to assert SRV OUT after DAT IN falls, and T2 which represents the time the control unit takes to drop DAT IN after DAT OUT is asserted or to drop SRV IN after SRV OUT is asserted. Additionally, TC represents the round trip time required for a signal to travel the length of the channel bus cable and back. The diagram shows signals as viewed at the channel controller, so TC is grouped with T2. A fourth pertinent interval, T3, is the time that elapses between the fall of SRV IN and the rise of DAT IN or the fall of DAT IN and the rise of SRV IN. A transfer cycle begins with the fall of SRV IN or DAT IN, but its execution is suspended 50 nanoseconds after beginning, awaiting the assertion of DAT IN or SRV IN, respectively, if it has not already occurred.

When control units such as the TX02 operate at the maximum rate that the DX20 will support, T3 is less than 50 nanoseconds. For the case in which the channel bus transfer rate is not Massbus limited (SCLK rate not limited to less than 1.2 megahertz) and T3 is less than 50 nanoseconds, T1 will be less than 480 nanoseconds for the single bus mode and less than 480 + 350 = 830 nanoseconds for the dual bus mode. Measurements on a tape subsystem employing a TX02 which is capable of data rates up to 1.25 megabytes per second show T2 to be approximately 100 nanoseconds. TC is 3 nanoseconds per linear foot of cable. Graphs in Figure 5-5 show the maximum guaranteed data rate as a function of cable length for

different values of TD, where TD = T2, for single bus mode and dual bus mode. The permissible data rates are bounded by the 200 foot maximum allowed cable length and the line corresponding to the maximum value of T2 for the control unit.

The case in which the channel bus transfer rate is limited by the maximum allowable Massbus transfer rate requires the addition of another term to the corresponding expression for TD. The added term is the same as that for TX01-type control units.

5.2.4 Data Path Interface

5.2.4.1 Introduction - The data path interface provides an auxiliary data link between the Massbus interface and the channel bus interface for transferring data at higher rates than is possible with the microcontroller. It also performs the bit/byte manipulation required to pack data into or unpack data from the host memory as it is received or transmitted over the channel bus. Physically, the logic for the data path interface is contained on the M8605 data storage board and the M8606 data formatter board which plug into slots AD10 and AF12, respectively.

5.2.4.2 Simplified Block Diagram - A simplified block diagram for the data path interface is shown in Figure 5-6. The data path interface communicates with the rest of the system over three bidirectional buses: the microbus (UBUS), the master bus (MSTR), and the slave bus (SLVE). Status, control, and diagnostic information is transferred over the microbus to the M8602 microprocessor board under control of the microcontroller. The microbus interface is organized as sixteen 8-bit registers; eight are implemented on the data storage board (UBRA 00 - 07 octal) and the remaining 8 are implemented on the data formatter board (UBRA 10 - 17 octal). Host data is transferred between the Massbus interface and the data path interface over the master bus as 16-bit PDP-11 or 18-bit PDP-10/20 Massbus words. Channel data is transferred between the channel bus interface and the data path interface over the slave bus as six, seven, or eight bit bytes, depending on the data format.

The formatter links the master and slave bus transceivers on the data formatter board and performs the bit/byte manipulation required to pack and unpack the data as it passes it between the transceiver registers. Data formatting and the arbitration of master and slave requests is handled by microprograms which reside in a 512 20-bit word ROM. There are a total of 27 separate, closed loop programs which implement the 9 data formats available in hardware for device write, read forward, and read reverse operations. The direction and type of transfer is established by the level of the DATA TO DEV H line (+3 V = device write, GND = device read) from the Massbus data board and the particular program which the formatter is directed to execute, both of which are under control of the microcontroller over the microbus. When the data path interface is enabled to transfer data, the DX HIGH SPEED L line is asserted (GND = asserted). The MSTR, SLVE, and

Figure 5-6 Data Path Interface - Simplified Block Diagram

UBUS signals are defined in Chapter 4.

Two data counters on the data storage board, a Massbus counter (MC) which counts 16 or 18-bit words transferred over the master bus and a byte counter (BC) which counts 6, 7, or 8-bit bytes transferred over the slave bus, can be used to indicate the total number of words and bytes involved in a transfer or to terminate the transfer when a predetermined number of words and/or bytes have been processed.

5.2.4.3 Data Formats - The following 9 formats are supported by the hardware:

1. Industry compatible (9-track)
2. Core dump (9-track)
3. High density (9-track)
4. 6-bit ASCII (7-track)
5. 7-bit ASCII - Mod 1 (9-track with bit 35)
6. 7-bit ASCII - Mod 2 (9-track without bit 35)
7. 11 - Mod 1 (9-track with normal byte sequence)
8. 11 - Mod 2 (9-track with reverse byte sequence)
9. COBOL EBCDIC (9-track)

The relationship between the 18-bit host words and the 6, 7, or 8-bit channel words for the various formats is shown in data format charts (see drawing package). The diagram shows how bytes transferred over the slave bus data lines are mapped into the words transferred over the master bus data lines and vice versa; no additional manipulation within words or bytes occurs in the Massbus interface or channel bus interface. Device write forward, read forward, and read reverse operations are possible for all of the data formats with the following additional capabilities:

1. For write operations, transfers can be terminated after any number of bytes have been transferred, irrespective of word boundaries.

2. For read operations, records of any number of bytes can be read; zeros are mapped into remaining byte positions within words before they are sent to the host. The transfer ends when the formatter has sent the last word (with the last byte) in the sequence when reading forward or the first word (with the first byte) in the sequence when reading reverse.

3. For read reverse operations, byte packing can start with any byte in the sequence to permit proper formatting of records of known length with byte counts which do not correspond to an integral number of 18-bit words for the 11 modes, 36-bit words for the industry compatible, core dump, 6-bit ASCII, 7-bit ASCII, and COBOL EBCDIC modes, or 72-bit words for the high density mode.

5.2.4.4 Data Transfer Rate - The data transfer rate is ultimately

determined by the device attached to the channel bus which drives the data path by requesting data as it needs it. The Massbus interface establishes the transfer rate over the Massbus by the rate at which it issues SCLKs to the host. Circuitry in the interface limits the maximum SCLK rate to prevent overrunning the host, and thus sets a limit on the maximum data rate that can be achieved over the channel bus. The limit on the maximum SCLK rate is adjustable via DIP switches on the M8603 Massbus data board from 0.34 to 1.92 megahertz in 14 increments, to allow optimization of the maximum transfer rate of the DX20 for the I/O capabilities of the particular host machine and the requirements of the peripheral device.

The Massbus and channel bus protocols and the handshaking over internal buses in the high speed data path (including the master bus, the slave bus, and buses within the data path interface) prevents the possibility of data overruns within the DX20; eliminates the need for overrun detection in the high speed data path; and renders ineffective the use of SILO buffer storage on the M8605 data storage board to minimize overruns. Overruns occur in and are detected by either the host or the device.

The data path interface is capable of transferring data over the slave bus at rates up to 2.4 megabytes per second; provided the Massbus interface is set up to allow a maximum SCLK rate of 1.2 megahertz, the closest higher switch setting being 1.44 megahertz. The maximum data rate that can be attained over the channel bus is dependent on timing within the channel bus interface and device controller and the length of the cable connecting the two.

5.3 DX20 CONTROLLER DESCRIPTION

5.3.1 Microcontroller Description

The DX20 microcontroller (Figure 5-7) is a high-speed microprocessor (MP) which executes a sequence of microinstructions retained within a writable control store. Microinstructions are loaded through the MP's diagnostic link; a bus containing a host-MP control interface and a 16-bit (plus parity) bidirectional data bus. This bus is used for both loading microinstructions and reading the MP's diagnostic registers.

The MP interfaces with the RH20 Massbus controller and a device via the Massbus interface and device interface modules. Bidirectional data and parity lines; addressing; handshake; interrupts; and other control between the MP and interfaces is provided over the microbus.

Sequencing (addressing) of the stored microinstructions is controlled by a program counter (PC). A PC multiplexer for user loading and stack nesting is implemented for use in high-performance microstore applications. The PC can be loaded from either the stack, the Massbus for user loading; or from the arithmetic logic with (ALU) for conditional jumps. PC data is saved every cycle and placed in the stack buffer (STB) in anticipation of a stack push operation. If a stack push is decoded, the contents of STB will be written into the stack.

The stack pointer (SP) is incremented before pushing, and decremented before popping operations. The SP consists of a 4-bit synchronous reversible up/down counter. On initialization the counter is loaded at zero, and then incremented to one prior to pushing the stack. From that point it is incremented by one; prior to each successive push operation. Pop operations are similar with the counter being decremented by one for each operation. Attempting to perform either operation in the wrong direction when the counter is at minimum (0) or maximum (15); results in the setting of the over/underflow flip-flop. For example, a pop (MOV instruction with destination bits = 7) when SP is equal to zero causes an underflow. A push (JUMP instruction with condition bits = 7) when SP equals 15 causes an overflow.

Microinstructions are loaded into an instruction register (IR) for subsequent decoding to define the various MP operations to be performed. They are then decoded by a source read-only memory (SROM), destination ROM (DROM), and an ALU function ROM (FROM). The ALU results are clocked into the ALU buffer (BALU) and transmitted to the destination specified by the DROM. Destinations can be: output data to the microbus (data out); the working memory; the buffer register (BR); or the general purpose accumulator registers (AR). Input data can be received from: the microbus (data in); the working memory; the BR shift-right register; or the operand (immediate data) of the current microinstruction.

5.3.1.1 **Microbus** - The microbus is the internal control bus used

Figure 5-7 Microcontroller - Simplified Block Diagram

by the microcontroller to communicate with the other DX20 interfaces. It contains address, data, and control lines which allow the microbus interface to be accessed and controlled by the microcontroller. Microbus timing is shown in Figure 5-8. The following paragraphs contain a description of the microbus signal lines.

Data Lines - Microbus data lines transmit data between the microbus interfaces. They are bidirectional and perform the following functions.

1. During a read operation; data is transmitted from the external microbus interface to the microcontroller.

2. During a write operation; data is transmitted to the external microbus interface from the microcontroller.

Data Parity - The microbus parity line contains odd parity for the data being transferred on the microbus lines.

Address Lines - I/O selection information for addressing a microbus interface is transmitted on the microbus address lines. This address consists of two fields; as follows:

1. Eight I/O bank select lines; each line generally specifying a microbus interface.

2. Five register select lines; generally specifying a register within a microbus interface.

I/O Strobe (STR) - A control signal issued to the microbus interfaces indicating the start of an I/O operation. Only the addressed microbus interface is involved in this operation, and returns a reply to the microcontroller.

Data Transfer Direction - A control signal issued by the microcontroller to the microbus interfaces to indicate the data transfer direction for an I/O operation. The two signals used for this purpose are:

1. DATO: data out from the microcontroller
2. DATI: data in to the microcontroller

Interrupt - This signal is issued to the microcontroller by a microinterface indicating it requires attention or servicing. The microcontroller scans the interrupt lines periodically and determines the priority of servicing.

Initialize - This signal is used to reset/initialize all interfaces on the microbus. It is generated by one of the following conditions:

1. INIT: issued by massbus
2. DX RESET: issued by the host

Figure 5-8 Microbus Timing (2 sheets)

3. Power OK: received from power supply on power up
4. INIT: issued by microcontroller

MP Running - The microcontroller signal that indicates its clock has started and is running. This signal notifies interfaces connected to the microbus when the clock has stopped as a result of an abnormal condition.

5.3.1.2 Program Storage - The MP control store memory contains 2K of 16-bit words (Figure 5-9). Each 1000 words is defined as a section consisting of four pages (page = 256 words). Sequencing (addressing) of the 16-bit microinstructions is controlled by the program counter. The microinstruction read from the addressed location is loaded into the instruction register to be decoded in order to define the MP operation required.

5.3.1.3 Instruction Cycles - There are four types of microcontroller instruction cycles: data in, data out, data in/out, and non I/O cycle. The first three types require the MP to utilize the microbus for execution which causes the MP to wait for a response from the module accessed by the instruction.

5.3.1.4 Instruction Set Summary - The 16-bit microinstruction word, decoded by the instruction decode logic, determines the events that are to occur during the remainder of the instruction cycle. Bits 15, 14, and 13 (OP CODE) determine the source of the current operation; bits 12, 11, and 10 determine the condition for a JUMP instruction or the destination of data for a MOVE instruction. Bits 9 through 0 are used as shown in the following description.

Move immediate - MOV(I) - This instruction moves the operand to the destination specified and at the end of execution, alters the address register of the working memory, as specified by the MA CONT field.

Move input - MOVE(INP) - This instruction moves information from the specified register (bits 3 through 0, and bit 15) of a microbus interface, selected by the input bank register, to the specified destination. When the destination is to be a microbus interface, bits 7 through 4, and bit 10 specify the register of the selected microbus interface. Alteration of the working memory address register is controlled by the MA CONT field.

Move Memory - MOVE(MEM) - The results contained in the ALU are transferred to the specified destination. ALU data is formed from input from the working memory and the accumulator specified by AC ADDR. The ALUF field (bits 3 through 0) specifies the ALU function to be used to determine the result. At the conclusion of execution, the working memory address is altered as specified by MA CONT.

Move BR - MOVE(BR) - Operation of this move instruction is identical to the MOVE(MEM), with the exception that the BR

Figure 5-9 Microstore Addressing

register control is the source. An end around shift can be accomplished from BR bit 0 through the source multiplexer, ALU, and BALU back to the serial input of BR.

Jump Immediate - JUMP(I) - This instruction performs a jump immediate operation within a 1K microstore bank. The condition field, of the instruction, specifies the type of jump operation. When jumping, the operand and page field are loaded into the PC denoting the address of the next microinstruction.

Jump memory - JUMP(MEM) - This instruction utilizes the working memory and an accumulator to form the eight least significant bits of the branch address. If the condition field requirement is met, the eight bits plus the page bits (8 and 9) are loaded into the PC. If the section select enable (bit 7) is a one, bits 4 and 5 of the microinstruction are loaded as the section address (bits 10 and 11) of PC. For proper operation, when section select is enabled, the ALUF field should contain select B as its function code.

Jump BR - JUMP(BR) - This instruction is similar to a JUMP(MEM), except that the contents of BR are used for the source data.

5.3.1.5 **Internal Registers** - Basic block diagrams of the following microcontroller registers are shown in Figure 5-10.

Working Memory - The working memory contains 1K x 9-bit (8 data plus parity) read access programmable memory (RAM). Each bit consists of a 1K x 1 bit RAM, addressable by a 10-bit (9-0) memory address (MA) counter. Bits 9 and 8 are the most significant bits of the address and form the MA extension. These two bits are derived from the contents of the two least significant bits (1 and 0) of the BALU. They are generated during an interrupt operation when the extension mode operation is specified, as indicated by the contents of the MA CONT field of the instruction word. The remaining eight MA bits (7 through 0) are derived from bits 7 through 0 of the BALU. Incrementing of the MA occurs at the end of the instruction cycle.

Accumulator - The microprocessor provides eight 8-bit general purpose accumulator registers. These registers have read/write capability when addressed via bits 6-4 (AC ADDR field) of the instruction word. Odd parity is maintained in the register and checked on each access operation. Register outputs are applied to the A input of the ALU.

BR Register - The BR register is an 8-bit shift-right register which shifts "end off", providing an end around shift from the BR through the source mux, ALU, BALU back to the BR. During the shift, the most significant bit of BR is loaded from the least significant bit of BALU. JUMP instructions which require branch operations include a CONDITION field containing either BR0, 4, or 7.

Figure 5-10 Microcontroller Registers (4 sheets)

Status Register - The 6-bit status register supplies the microcode with information concerning internal or external (interrupt) events.

An ALU carry (C bit) indicates a carry occurred from the most significant bit position of the ALU as a result of the last move instruction with an arithmetic ALU function.

The Z bit set, indicates that the result in the ALU from a MOVE instruction is equal to all 1's. Typically, the Z bit is utilized when comparing the A and B inputs of the ALU. A magnitude comparison is made by placing the ALU in the 1's complement subtract mode.

Interrupt bits (INT 3-0) are set from four independent lines of the microbus. Sampling of these lines occurs every T60 time of each cycle. Activity on the lines is sensed by the microcode sampling (branching) the interrupt sense line which is generated by any one of the interrupts. The microcode interrogates the status register to determine which interrupt occurred and resolves the action which is to follow.

I/O Select Logic - The I/O select logic consists of the; I/O bank register, I/O field select register and the I/O select register.

Outputs from BALU and IR are issued to the bank register to determine input/output select control.

The input and output select levels are decoded in the I/O register select register and form the microbus I/O select control signals and microbus address signals.

Bit 6 of the bank register provides control to reset the stack pointer. This bit must then be reset prior to using the stack pointer function.

Clearing (initialization) of the microbus interface is accomplished either from the host through the Massbus or by the setting of bit 7 in the bank register. Reset is wrapped around and clears bit 7 when it is generated by bit 7 being written.

Diagnostic Link - The diagnostic link provides the host interface with auxiliary access to the microcontroller. It allows access to the internal microprocessor registers and microstore loading. Diagnostic controls within the microcontroller permit host diagnostics to interrogate, scan, and diagnose problems.

Parity Checking - Parity error detection in the data path is provided by parity logic at the A and B inputs to the ALU. AC outputs determine the ALU-A input check, and SMUX outputs determine the ALU-B check. Parity checking on the ALU-B source input is disabled when a source without parity is decoded.

Parity is generated for the data output of the ALU and is issued

to the BALU (ALU buffer) coincidental with the ALU data.

Parity checking of the 16-bit microinstruction word (from the IR MUX) occurs when a machine cycle is executed, or when a diagnostic write of the RAM is performed. When an IR parity error is sensed, the current cycle is inhibited from being executed.

System Error Detection - Various types of error conditions can be detected by the internal hardware of the microcontroller. When an error is detected, the microcontroller clocks are stopped.

Stack over/under flow errors occur when a push operation is attempted on a full stack, or when a pop operation is attempted from an empty stack.

The detection of even parity in the IR register or at the A or B input of the ALU causes the microcontroller to stop.

5.3.2 Massbus Interface Description

5.3.2.1 Introduction - The Massbus interface is composed of two boards which physically and logically divide it into the two sections that correspond to the functions of the Massbus. The Massbus control board (M8604) provides interfacing to the control bus section of the Massbus, the internal microbus of the DX20, the diagnostic link, and communicates with the Massbus data board. The Massbus data board (M8603) provides interfacing to the data bus section of the Massbus, the master bus (MSTR) connecting the Massbus and data path interfaces, and communicates with the Massbus control board. In addition, the Massbus interface contains the data buffer input multiplexer, data buffer output multiplexer, and data buffer control logic. The M8604 is a hex board located in slot AF08 and the M8603 is a quad board located in slot AD06. Additionally there are three M5903 Massbus Transceiver Boards, which are double height boards, located in slots EF07, EF10, and EF11. These boards do not contain control logic, but convert Massbus differential signals to standard TTL levels, and vice-versa. Details of the various buses are contained in Section 5.2.

5.3.2.2 Simplified Block Diagram - A simplified block diagram of Massbus interface is shown in Figure 5-11. The figure is divided into two sections showing the control board and data board with connections to the remainder of the system.

The control bus portion provides device and register selection, generates timing, and provides access to registers over a 16-bit bidirectional data path. As viewed from the host, the interface is organized as 23 16-bit registers (MBRA 00-06 and 20-37 octal). These registers are described in Section 4.4.

The microbus portion of the Massbus control board provides device and register selection, generates timing, and supplies access to registers over an 8-bit bidirectional data path. This interface is organized as 24 8-bit registers (as viewed from the microcontroller) denoted as UBRA 00-27 octal. Many of these registers are common to the host. A complete description of these registers is contained in Section 4.4.2.

The diagnostic link contains a control path and a 16-bit bidirectional data path to the microcontroller. This link is used for: loading and verifying microcode into the microstore, starting and stopping the microcontroller, and providing diagnostic information to the host for maintenance purposes and system debugging.

Prior to a checkout, there are three DIP switches located on the M8604 board that must be set up. A DIP switch located in E48 is used for device or unit number selection. Only switches 1, 2, and 3 are used in this board, switch 3 being the least significant (refer to drawing MC01). DIP packages located in E30 and E31 are used for setting up the drive type number (refer to drawing MC02).

Figure 5-11 Massbus Interface - Simplified Block Diagram

The Massbus data board contains the logic which recognizes and generates signals for transferring data between the host and the data path interface. It is organized such that data can be transferred between the data bus and the microbus. This provides an alternate data path from the host to the device for system maintenance. The data bus control section includes a circuit for generating clock signals (SCLK) to the host. In high speed data transfer mode, the rate at which clocks are generated is determined by the settings of four switches. For lower speed data transfers, without the data path, the microcontroller triggers the clock for each transfer over the Massbus data bus.

5.3.3 Channel Bus Interface

5.3.3.1 Introduction - The channel bus interface logic is contained on three boards in the DX20, the M8607 Channel Bus (CB) Board, the optional M8608 channel extension (CE) board, and the G891 power fail and select bypass module. A brief description of the logic on these boards is given in this section with the aid of the block diagrams in Figure 5-4. The channel bus interface circuitry has been partitioned into 35 blocks to permit identification of the circuit functions implemented in the interface. Drawing numbers for the circuit schematics are included next to the blocks to locate the logic circuits which correspond to each of the blocks.

5.3.3.2 Microbus Interface and Control

Introduction - The microbus interface and control logic provides communication between the microcontroller and the channel bus interface by allowing the microcontroller to generate clock signals and write registers in the channel bus interface via DATO instructions or sense the logic levels of internal signals and read registers in the interface via DATI instructions. The following blocks comprise the microbus interface and control logic.

Microbus Control (UB CTRL) - The microbus control logic on the M8608 Channel Bus Board buffers and decodes microbus signals driven from the microcontroller to generate internal channel bus interface signals. UB INIT is the buffered equivalent of UBUS INITIALIZE, UB RUN is the buffered equivalent of UBUS RUNNING, UB DATI is the buffered equivalent of UBUS DATI, and UB RS 3 - 0 are the buffered equivalents of UBUS ADRS 3 - 0. The internal UB STB signal is asserted (I/O SEL 3 asserted) and one of the implemented lower 15 decimal registers is addressed (UBUS ADRS 4 low). DATO STB is asserted when UB STB is asserted during DATO cycles (DATI low). DATI STB is asserted when UB STB is asserted during DATI cycles (DATI high) which address registers 0 - 7. The microbus interrupt line assigned to the channel bus interface (UBUS INT 3) is asserted if one or more of the following conditions are true: TIME OUT FLAG = 1, DP PE FLAG = 1, UB PE FLAG = 1, MK PE FLAG = 1, BUS 1 PE FLAG = 1, BUS 0 PE FLAG = 1, CU RESET = 1, or TI DIS IN =

1 and CHAN MODE = 1.

Microbus Receiver (UB RCVR) - The microbus receiver on the M8608 Channel Bus Board buffers and latches microbus data on the rising edge of UB STB for use during microcontroller DATO cycles. UBUS DATA 7 - 0, P is latched as UB DATA 7 - 0, P with both true and complement data available. All of the bits in the register are direct set when UB INIT is asserted.

Microbus Parity Checker (UB PAR CHK) - The microbus parity checker on the M8608 Channel Bus Board checks the parity of the microbus data latched in the microbus receiver register. UB PE H is asserted if incorrect (even) parity is detected on the data.

Microbus Address Demultiplexers (UB ADR DEMUX) - There are two microbus address demultiplexers on the M8607 Channel Bus Board. One decodes the microbus address lines to generate a pulse on one of 16 decimal lines during microcontroller DATO cycles to the channel bus interface to load registers, clock or clear flip-flops, and assert signals in the interface as defined in Section 4.4.3. Fifteen of the lines are identified as LD REG 1 - LD REG 17 and correspond to microbus addresses 01 - 17, respectively; if a microcontroller DATO cycle to the channel bus interface involves one of these registers, the appropriate LD REG 1 signal will be asserted while UBUS STROBE is high. The remaining line is identified as CLR REG 0 and corresponds to microbus address 00. A 50 nanosecond pulse will be generated on CLR REG 0 on the trailing edge of UBUS STROBE if UBRA 00 is written or on the trailing edge of UB INIT. Another demultiplexer decodes the microbus address lines to generate a pulse on one of three lines during microcontroller DATI cycles to the channel bus interface to read registers on the M8608 Channel Extension Board and clock the BUS 0 PE FLAG, BUS 1 PE FLAG, and MK PE FLAG flip-flops as described in Section 4.3.3. The lines are identified as RD REG 7, RD REG 12, and RD REG 13 and correspond to microbus addresses 07, 12, and 13 octal, respectively.

Register 1 (REG 1) - The eight read/write bits of register 1 (UBRA 01 Bits 7 - 0) are implemented as D-type flip-flops with both true and complement data available on the M8607 Channel Bus Board. The trailing edge of LD REG 1 clocks the output of the microbus receiver register into register 1. The register is direct cleared by asserting UB INIT or dropping UB RUN.

Tag Out Registers (TAG OUT REG) - There are two tag out registers consisting of D-type flip-flops with both true and complement data available on the M8607 Channel Bus Board. The trailing edge of LD REG 2 clocks output bits 7 - 4 of the microbus receiver register into the corresponding bits of Tag Out Register 0 (UBRA 02 Bits 7 - 4), and the trailing edge of LD REG 3 clocks the output of the microbus receiver register into Tag Out Register 1 (UBRA 03 Bits 7 - 0). The registers are direct cleared by asserting UB INIT.

Microbus Data Multiplexer (UB DATA MUX) - Microbus registers UBRA 00 - 07 in the channel bus interface are multiplexed onto the microbus data lines (UBUS DATA 7 - 0) via the microbus data multiplexer on the M8607 Channel Bus Board. The appropriate multiplexer inputs are selected by UB RS 2 - 0 and DATI STB L from the microbus control logic enables the tristate drivers which gate the data onto the microbus data lines.

Channel Extension Data Multiplexer (CE DATA MUX) - Microbus registers UBRA 12 and UBRA 13 in the channel bus interface are multiplexed onto the microbus data lines (UBUSDATA 7 - 0) via the channel extension data multiplexer on the M8608 Channel Extension Board. When RD REG 12 is asserted, the bits in data register 1 (DR 15 - 08) are gated onto the microbus data lines via tristate drivers, and when RD REG 13 is asserted, the bits in bus in register 1 (CBI 15 - 08) are gated onto the microbus data lines via tristate drivers.

Diagnostic Multiplexer (DIAG MUX) - The diagnostic multiplexer multiplexes one of sixteen signals into UBRA 03 Bit 2 of the microbus data multiplexer. The output of the scratch pad address counter is used to select the signal as described in Section 4.4.3.7.

Register 5 Multiplexer (REG 5 MUX) - The register 5 multiplexer allows either the output of the scratch pad address counter (SP ADR 3 - 0) when CHAN MODE = 1 or the outputs of the CU RUN, SYS RST, SEL RST, and HALT 10 flip-flops when CHAN MODE = 0 to be read by the microcontroller as UBRA 05 Bits 3 - 0 through the UB DATA MUX. See channel interface diagram UBUS Registers (I/O SEL 3) for the bit assignments.

5.3.3.3 Bus 0 Data Path

Introduction - The Bus 0 data path logic interfaces the BUS 0 IN and BUS 0 OUT lines of the channel bus with the data path interface and the microcontroller. BUS 0 IN is used to transfer data to the channel bus interface; data on the bus can be read by the microcontroller as microbus register 7 (Bus In register 0) or transferred to data register 0 (DR 7-0) for transmission to the data path interface over the slave bus during high speed transfers (DX HIGH SPEED = 1). BUS 0 OUT is used to transfer data to the device; it can be driven from either bus out register 0 (BOR 7-0) when DX HIGH SPEED = 0 or data register 0 (DR 7-0) which contains data received from the data path interface during device write operations when DX HIGH SPEED - 1. BOR 7-0 can be loaded directly from the microcontroller when SP EN = 0 or indirectly through the scratch pad 0 RAM (SP 7-0 RAM) when SP EN = 1. The blocks comprising the Bus 0 data path logic are all implemented on the M8607 Channel Bus Board and are shown in the upper half of Figure 5-4.

BUS 0 OUT Drivers (BUS 0 OUT DRVR) - The BUS) OUT DRVR logic is composed of nine drivers which interface DR 7-0 and BOR 7-0 to the

channel bus. For channel controller operation (CHAN MODE = 1 and CU RESET EN = 0), the open-emitter pull-up outputs drive IBCB BUS 0 OUT 0 = 7, P when enabled (DX ON LINE = 1 and END XFER = 0), and for control unit operation (CHAN MODE = 0), the outputs drive IBCB BUS 0 IN 0 - 7, P when enabled (DX ON LINE = 1, CU RESET = 0, and SLVE SEL = 1). If DX HIGH SPEED = 1 and DATA TO DEV = 1, DR 07 - 00, P is selected to drive IBCB BUS 0 OUT 0 - 7, P, and if DX HIGH SPEEC = 0, BOR 07 - 00, P is selected (note the reversal of bit numbering).

BUS 0 IN Receivers (BUS 0 IN RCVR) - The BUS 0 IN RCVR logic is composed of nine receivers which interface to the channel bus and provide loop back capability. The outputs of the registers are denoted CBI 07 - 00, P0 and are bused to the BUS 0 IN PAR CHK, DR 7-0, UB DATA MUX, and CU ADR COMP circuits. When LOOP EN = 0, IBCB BUS 0 IN 0 - 7, P are gated to CBI 07 - 00, P0, respectively, for channel controller operation or IBCB BUS 0 OUT 0 - 7, P are gated to CBI 07 - 00, P0, respectively, for control unit operation (note the reversal of bit numbering); when LOOP EN = 1, BOR 07 - 00, P0 are complemented and looped back into CBI 07 - 00, P0 by the BUS 0 IN RCVR logic.

BUS 0 IN Parity Checker (BUS 0 IN PAR CHK) - The BUS 0 IN PAR CHK circuit asserts BUS 0 ODD PAR while odd (correct) parity is detected in the data on the CBI 07 - 00, P0 lines.

Data Register 0 (DR 7-0) - DR 7-0 is composed of nine strobe-type latches which are transparent when LD REG 6 is asserted during a microcontroller DATO cycle to UBRA 06 or when CLK DR BYTE 0 is asserted by the CB CTRL logic during high speed transfers and latched with stable data otherwise. The outputs of the register are denoted DR 07 - 00, P0 and drive inputs to the SLVE BUS MUX/DRVR and BUS 0 OUT DRVR. A multiplexer at the latch inputs selects data on the slave bus when DATA TO DEV = 1 or data from Bus In Register 0 when DATA TO DEV = 0. DR 7-0 is used to buffer data as it is transferred between the data path interface and a device on the channel bus during high speed transfers. For device write operations, data received from the data path interface over the slave bus is latched in DR 7-0 and gated onto BUS 0 OUT by BUS 0 OUT DRVR. For device read operations, data received from the device over BUS 0 IN is latched in DR 7-0 and gated onto the slave bus by the SLVE BUS MUX/DRVR logic. If the extended bus feature is enabled (EXTENDED BUS = 1), the bytes of slave bus data alternate between DR 7-0 and DR 15-8.

Bus Out Register 0 Parity Generator (BOR 7-0 PAR GEN) - The BOR 7-0 PAR GEN circuit generates a parity bit for the data latched in BOR 7-0. The parity of the data including the parity bit is odd (correct) when EVEN PAR = 0 and is even (bad) when EVEN PAR = 1. The ability to generate bad parity allows the BUS 0 TN PAR CHK and BUS 0 PE FLAG logic to be thoroughly tested; when LOOP EN is asserted, BOR 07 - 00, P0 is complemented and looped back into CBI 07 - 00, P0 and thus presented to the BUS 0 IN parity detection circuits.

Scratch Pad 0 RAM (SP 7-0 RAM) - A 16-byte RAM is implemented in the bus 0 data path logic to allow the microcontroller to pass a series of parameters to a device over BUS 0 OUT more quickly than is possible with the microcontroller handling the transfer directly. The SP 7-0 RAM is addressed by the output of the SP ADR CNTR (-SP ADR 3 - 0). Data is loaded into the RAM location addressed by the SP ADR CNTR from the UB RCVR register by writing UBRA 10. The RAM outputs (SP DATA 07 - 00) drive the inputs of BOR 7-0 which are selected when SP EN = 1. The microcontroller initially loads the parameters into sequential locations of the RAM via UBRA 10, the first parameter being loaded into the highest address. The SP ADR CNTR is then set to point to the first parameter and SP EN is set. Logic in the channel bus interface will arbitrate the SRV IN/DAT IN requests and transfer the data in the RAM to the device using BOR 7-0 and BUS 0 OUT and decrementing the SP ADR CNTR after each byte is transferred. Devices which use this feature must request a predetermined number of bytes because there is no mechanism in the channel bus interface by which a transfer can be terminated or an address underflow condition can be detected.

Scratch Pad Address Counter (SP ADR CNTR) - The SP ADR CNTR is a loadable down-counter whose outputs (-SP ADR 3 - 0) address the scratch pad RAMs (SP 7-0 RAM and SP 15-8 RAM) and the diagnostic multiplexer (DIAG MUX). The counter can be loaded by the microcontroller as UBRA 05 Bits 3 - 0 and can be read back as the same bits through the register 5 multiplexer (REG 5 MUX) when CHAN MODE = 1. The counter is enabled to count when SP EN = 1 and will decrement when the SP 7-0 RAM is written by the microcontroller (UBRA 10) or immediately after BOR 7-0 is clocked. Note that the counter is loaded with complemented microbus data and its outputs form a complemented address that is actually incremented each time the counter is clocked, which is equivalent to decrementing the uncomplemented address.

Slave Bus Multiplexer/Driver (SLAVE BUS MUX/DRVR) - The SLVE BUS MUX/DRVR is used during device read operations to gate data from DR 7-0 or DR 15-8 onto the slave bus data lines. A multiplexer selects DR 7-0 when 2ND BYTE = 0 and selects DR 15-8 when 2ND BYTE = 1. In single bus mode, all transfers are accomplished with DR 7-0 and 2ND BYTE remains low. In dual bus mode, however, both DR 7-0 and DR 15-8 are involved in the parallel two-byte transfers of data over BUS 0 IN and BUS 1 IN and 2ND BYTE continuously toggles so that the data transferred to the data path interface over the slave bus during device read operations will consist of alternate bytes of BUS 0 IN and BUS 1 IN data. The tristate outputs of the SLVE BUS MUX/DRVR are enabled to drive the slave bus data lines during device read operations (DATA TO DEV = 0).

Slave Parity Checker (SLVE PAR CHK) - The SLVE PAR CHK circuit asserts DP PE (data path parity error) when bad (even) parity is detected on the slave bus data lines.

5.3.3.4 Bus 1 Data Path

Introduction - The bus 1 data path logic interfaces the BUS 1 IN and BUS 1 OUT lines of the channel bus with the data path interface and the microcontroller. BUS 1 IN is used to transfer data to the channel bus interface in conjunction with BUS 0 IN when the extended bus feature is enabled (EXTENDED BUS = 1): data on the bus can be read by the microcontroller as microbus register 13 or transferred to data register 1 (DR 15-8) for transmission to the data path interface over the slave bus during high speed transfers (DX HIGH SPEED = 1). BUS 1 OUT is used to transfer data to the device in conjunction with BUS 0 OUT when the extended bus feature is enabled; it can be driven from either bus out register 1 (BOR 15-8) when DX HIGH SPEED = 0 or Data Register 1 (DR 15-8) which contains data received from the data path interface during device write operations when DX HIGH SPEED = 1. BOR 15-8 can be loaded directly from the microcontroller when SP EN = 0 or indirectly through the scratch pad 1 RAM (SP 15-8 RAM) when SP EN = 1. The blocks comprising the bus 1 data path logic are all implemented on the optional M8608 channel extension board and are shown in Figure 5-4.

BUS 1 OUT drivers (BUS 1 OUT DRVR) - The BUS 1 OUT DRVR logic is composed of nine drivers which interface DR 15-8 and BOR 15-8 to the channel bus. For channel controller operation (CHAN MODE = 1 and CU RESET EN = 0), the open-emitter pull-up outputs drive IBCB BUS 1 OUT 0 - 7, P when enabled (DX ON LINE = 1, END XFER = 0, and EXTENDED BUS = 1), and for control unit operation (CHAN MODE = 0), the outputs drive IBCB BUS 1 IN 0 - 7, P when enabled (DX ON LINE = 1, CU RESET = 0, SLVE SEL = 1, and EXTENDED BUS = 1). If DX HIGH SPEED = 1 and DATA TO DEV = 1, DR 15 - 08, P is selected to be driven onto the channel bus, and if DX HIGH SPEED = 0, BOR 15 - 08, P is selected (note the reversal of bit numbering).

BUS 1 IN Receivers (BUS 1 IN RCVR) - The BUS 1 IN RCVR logic is composed of nine receivers which interface to the channel bus and provide loop back capability. The outputs of the registers are denoted CBI 15 - 08, P1 and are bused to the BUS 1 IN PAR CHK, DR 15-8, and UB DATA MUX circuits. When LOOP EN = 0, IBCB BUS 1 IN 0 - 7, P are gated to CBI 15 - 08, P1, respectively, for channel controller operation, or IBCB BUS 1 OUT 0 - 7, P are gated to CBI 15 - -8, P1, respectively, for control unit operation (note the reversal of bit numbering); when LOOP EN = 1, BOR 15 - 08, P1 are complemented and looped back into CBI 15 - 08, P1 by the BUS 1 IN RCVR logic. The BUS 1 IN RCVR logic participates in channel bus operations in which the extended bus feature is used.

BUS 1 IN Parity Checker (BUS 1 IN PAR CHK) - The BUS 1 IN PAR CHK circuit asserts BUS 1 ODD PAR while odd (correct) parity is detected in the data on the CBI 15 - 08, P1 lines.

Data Register 1 (DR 15-8) - DR 15-8 is composed of nine strobe-type latches which are transparent when LD REG 12 is asserted during a microcontroller DATO cycle to UBRA 12 or when CLK DR BYTE

1 is asserted by the CB CTRL logic during high speed transfers and latched with stable data otherwise. The outputs of the register are denoted DR 15 - 08, P1 and drive inputs to the SLVE BUS MUX/DRVR and BUS 1 OUT DRVR. A multiplexer at the latch inputs selects data on the slave bus when DATA TO DEV = 1 or data from bus in register 1 when DATA TO DEV = 0. DR 15-8 is used to buffer data as it is transferred between the data path interface and a device on the channel bus during high speed transfers. For device write operations in which EXTENDED BUS = 1, alternate bytes received from the data path interface over the slave bus are latched in DR 15-8 and gated onto BUS 1 OUT by the BUS 1 OUT DRVR. For device read operations in which EXTENDED BUS = 1, data received from the device over BUS 1 IN is latched in DR 15-8 and gated onto the slave bus by the SLVE BUS MUX/DRVR logic; the bytes of slave bus data alternate between DR 7-0 and DR 15-8. DR 15-8 is not involved in data transfers when EXTENDED BUS = 0.

Bus Out Register 1 Parity Generator (BOR 15-8 PAR GEN) - The BOR 15-8 PAR GEN circuit generates a parity bit for the data latched in BOR 15-8. The parity of the data including the parity bit is odd (correct) when EVEN PAR = 0 and is even (bad) when EVEN PAR = 1. The ability to generate bad parity allows the BUS 1 IN PAR CHK and BUS 1 PE FLAG logic to be thoroughly tested; when LOOP EN is asserted, BOR 15 - 08, P1 is complemented and looped back into CBI 15 - 08, P1 and thus presented to the BUS 1 IN parity detection circuits.

Scratch Pad 1 RAM (SP 15-8 RAM) - The SP 15-8 RAM implemented in the bus 1 data path logic complements its counterpart (SP 7-0 RAM) in the bus 0 data path logic. It is used in conjunction with the SP 7-0 RAM to transfer a series of parameters to a device on the channel bus over BUS 0 OUT and BUS 1 OUT simultaneously when EXTENDED BUS = 1. The SP 15-8 RAM is addressed by the output of the SP ADR CNTR (-SP ADR 3 - 0). Data is loaded into the RAM location addressed by the SP ADR CNTR from the UB RCVR register by writing UBRA 14. The RAM outputs (SP DATA 15 - 08) drive the inputs of BOR 15-8 which are selected when SP EN = 1. The microcontroller initially loads the parameters into sequential locations of the SP 15-8 RAM via UBRA 14 and the SP 7-0 RAM via UBRA 10; the first parameter being loaded into the highest address of the SP 7-0 RAM; the second parameter into the highest address of the SP 15-8 RAM; the third parameter into the second highest address of the SP 7-0 RAM, etc. The SP ADR CNTR is then set to point to the first parameter and SP EN is set. Logic in the channel bus interface will arbitrate the SRV IN/DAT IN requests and transfer the data in the RAM to the device using BOR 7-0, BOR 15-8, BUS 0 OUT, and BUS 1 OUT and decrementing the SP ADR CNTR after each pair of bytes is transferred. Devices which use this feature must request a predetermined number of bytes because there is no mechanism in the channel bus interface by which a transfer can be terminated or an address underflow condition can be detected.

5.3.3.5 Channel Bus Control

Introduction - The channel bus control logic performs the following functions: controls the tag out lines (TAG OUT DRVRS) and senses the tag in lines (TAG IN RCVRS); decodes the control bits in control and status register 1 (UBRA 01) and controls signals from other interfaces and responds appropriately (BUFFERS and CB CTRL); provides status bits to the microcontroller via UBRA 00 and interrupts the microcontroller to indicate important control states or error conditions which require immediate action (INT FLAGS and CB CTRL); arbitrates high speed transfers without the aid of the microcontroller for both single and dual bus operation by generating signals which manipulate the bus 0 data path and bus 1 data path logic and handle the slave bus and channel bus request-acknowledge sequences (CB CTRL); and in general, handles events for which real-time constraints preclude direct control by the microcontroller (CB CTRL). The blocks which comprise the channel bus control logic are shown in Figure 5-4. Most of the logic is implemented on the M8607 Channel Bus Board; the remaining logic is required for dual bus operation only and is included on the optional M8608 Channel Extension Board.

Buffers (BUFFERS) - The Buffers block is a collection of simple subcircuits on the M8607 Channel Bus Board which buffer and, in some cases; provide additional conditioning of signals which are used throughout the channel bus interface. Each of the sub-circuits is described below.

DATA TO DEV Buffer - DATA TO DEV from the Massbus interface is buffered by high output current drivers which generate true and complemented equivalents of DATA TO DEV which drive logic in the channel bus interface.

LOOP EN Buffer - The LOOP EN flip-flop output is buffered by high output current drivers which generate true and complemented equivalents of LOOP EN which drive logic in the channel bus interface.

SLVE END XFER Buffer - SLVE WOR END XFER is buffered by drivers which generate the equivalent signal, SLVE END XFER, and its complement, -SLVE END XFER, which drive logic in the channel bus interface.

DX HIGH SPEED Buffer - DX HIGH SPEED from the data path interface and DIAG HSPD from the REG 1 logic of the channel bus interface are logically ORed and buffered by high output current drivers to form the DX HIGH SPEED signal and its complement which are used in the channel bus interface.

DX ON LINE Buffer - The output of the ON LINE flip-flop and CROBAR from the G891 power fail and select bypass module are logically ANDed and buffered to form DX ON LINE and its complement. When DX ON LINE is low, the DX20 is effectively disconnected from the channel bus.

EN CB DRVRS Gate - Allow TAG/BUS OUT (the logical OR of CHAN MODE and SLVE SEL), -CU RESET (the logical OR of SYS RST, SEL RST, and HALT IO), and DX ON LINE are logically ANDed to form EN CB DRVRS (enable channel bus drivers) which prevents all the BUS 0 OUT, all the BUS 1 OUT, and most of the tag out drivers from asserting channel bus lines when it is low. During channel controller operation in which CHAN MODE = 1 and CU RESET EN = 0, EN CB DRVRS is asserted whenever DX ON LINE is high. During control unit operation in which CHAN MODE = 0, EN CB DRVRS is asserted only when DX ON LINE = 1 (the DX20 is on line), SLVE SEL = 1 (the DX20 has been selected by the channel controller), and CU RESET = 0 (no special sequences have been issued by the channel controller to disconnect the DX20 from the channel bus).

ACLO Inverter - ACLO from the H7420 power supply is inverted to generate the ACLO signal required by the G891 power fail and select bypass module. When ACLO is asserted, CROBAR is grounded, indicating a power failure condition, and the channel bus select signal is bypassed.

Tag Out Drivers (TAG OUT DRVRS) - IBM-compatible channel bus drivers with open-emitter pull-up outputs are used to drive the tag out lines when operating as a channel controller or the tag in lines when operating as a control unit. Descriptions of the conditions which assert the tag lines are given below. IBCB is used as a prefix to denote signals which drive the channel bus directly. Unless stated otherwise, EN CB DRVRS must be asserted to enable a driver and channel controller operation is assumed. The redefinitions of the tag lines for control unit operation are given in the channel interface diagram UBUS Registers (I/O SEL 3). All of the tag drivers are implemented in the TAG OUT DRVRS logic on the M8607 Channel Bus Board, except in the CE CTRL logic on the M8608 Channel Extension Board.

IBCB SRV OUT (Service Out) - IBCB SRV OUT is asserted in two cases: when TOR SRV OUT is high; or during transfers arbitrated by the CB CTRL logic.

IBCB CLK OUT (Clock Out) - IBCB CLK OUT is asserted when TOR CLK OUT is high.

IBCB MTR OUT (Meter Out) - IBCB MTR OUT is asserted when TOR MTR OUT is high.

IBCB ADR OUT (Address Out) - IBCB ADR OUT is asserted when TOR ADR OUT is high.

IBCB HLD OUT TRA (Hold Out Transmit) - IBCB HLD OUT TRA is asserted when TOR HLD OUT is high. It is not used for control unit operation.

CB04 SEL TRA (Select Transmit) - CB04 SEL TRA is asserted in two cases: when TOR SEL OUT is high; or during control unit operation

(CHAN MODE = 0) when LOOP EN = 0 (to insure that the driver is disabled during diagnostic testing with the loop back feature enabled), DCLO = 0 (DC supply voltages are within regulation), TRA SEL FF = 1 (a flip-flop in the CU SEL logic which is set if the select signal is to be propagated to the next control unit), regardless of the level of EN CB DRVRS. Case 1 is used exclusively when operating as a channel controller; the microcontroller must handle the channel bus sequences involving the select signal. Case 2 is used exclusively when operating as a control unit; TOR SEL OUT should not be asserted. The logic in the channel bus interface controls the propagation of the select signal because of timing constraints which preclude the use of the microcontroller. CB04 SEL TRA goes to the G891 Power Fail and select bypass module where it is connected to the channel bus via priority selection switches which connect CB04 SEL TRA to IBCB SEL 1 TRA for the low priority setting or to IBCB SEL 0 TRA for the high priority setting. See Figure .

IBCB CMD OUT (Command Out) - IBCB CMD OUT is asserted when: TOR CMD OUT is high; or at the end of a high speed transfer (DX HIGH SPEED = 1) when END XFER is raised by the CB CTRL logic as described in Section 4.4.3.2.

IBCB OPL OUT (Operational Out) - IBCB OPL OUT is asserted when TOR OPL OUT is high.

IBCB SUP OUT (Suppress Out) - During channel controller operation (CHAN MODE = 1 and CU RESET EN = 0), IBCB SUP OUT is asserted when TOR SUP OUT = 1 is high. During control unit operation (CHAN MODE = 0), TBCB SUP OUT is redefined as REQ IN and is asserted when TIR REQ IN = 1, TO SUP OUT = 0, CU RESET = 0, and DX ON LINE = 1, regardless of the level on EN CB DRVRS.

IBCB DAT OUT (Data Out) - IBCB DAT OUT is asserted when: TOR DAT OUT is high; or during transfers arbitrated by the CB CTRL logic.

IBCB MK 0 OUT (Mark 0 Out) - IBCB MK 0 OUT is always asserted when TOR OPL OUT is high to indicate that the bus 0 lines will be used to transfer data. The IBCB MK 1 OUT and IBCB MK P OUT lines are driven by circuitry in the CE CTRL logic on the M8608 Channel Extension Board when ALLOW MK 1 OUT is asserted by the CB CTRL logic to indicate that the bus 1 lines will be involved in a transfer.

Tag In Receivers (TAG IN RCVRS) - IBM-compatible channel bus receivers are used to sense the logic levels on the tag in lines for channel controller operation or the tag out lines for control unit operation. The receiver inputs are wired as multiplexers which allow channel bus lines to be received during normal operation (LOOP EN = 0) or tag out register bits to be looped back into tag in register bits during diagnostic testing (LOOP EN = 1) as specified in the channel interface diagram UBUS Registers (I/O SEL 3). Descriptions of the conditions under which the outputs of the tag in receivers, designated TI XXX IN for channel controller

operation or TO XXX OUT for control unit operation, are asserted as given below. IBCB denotes a channel bus signal line. Unless stated otherwise, channel controller operation is assumed; the redefinitions of the channel bus lines for control unit operation are given in the figure . All the tag in receivers are implemented in the TAG IN RCVRS logic on the M8607 Channel Bus Board, except the IBCB MK 1 IN and IBCB MK P IN receivers which are implemented in the CE CTRL logic on the M8608 Channel Extension Board.

TI OPL IN (Operational In) - TI OPL IN is asserted if IBCB OPL IN is raised when LOOP EN = 0 or if TOR SRV OUT is high when LOOP EN = 1.

TI MK 0 IN (Mark 0 In) - TI MK 0 IN is asserted if IBCB MK 0 IN is raised when LOOP EN = 0 or if TOR CLK OUT is high when LOOP EN = 1.

TI MK 1 IN (Mark 1 In) - TI MK 1 IN is asserted if IBCB MK 1 IN is raised when LOOP EN = 0 or if TOR MTR OUT is high when LOOP EN = 1 only if the M8608 Channel Extension Board is present; otherwise TI MK 1 IN is always low. The receivers for the MK 1 IN and MK P IN signals are implemented in the CE CTRL logic on the M8608 Channel Extension Board.

TI ADR IN (Address In) - TI ADR IN is asserted if IBCB ADR IN is raised when LOOP EN = 0 or if TOR ADR OUT is high when LOOP EN = 1.

TO HLD OUT (Hold Out) - TO HLD OUT is asserted if IBCB HLD OUT REC is raised when LOOP EN = 0 or if TOR HLD OUT is high when LOOP EN = 1. IBCB HLD OUT REC is functional only during control unit operation; it is left open during channel controller operation and is prone to noise pick-up.

TI MTR IN (Meter In) - TI MTR IN is asserted if IBCB MTR IN is raised when LOOP EN = 0 or is always low when LOOP EN = 1.

TI SEL IN (Select In) - TI SEL IN is asserted if CB04 SEL REC is raised when LOOP EN = 0 or if TOR SEL OUT is high when LOOP EN = 1. The CB04 SEL REC signal comes from the G891 Power Fail and Select Bypass Module. If the DX20 is off line (DX ON LINE = 0 or CROBAR = 0), CB04 SEL REC is returned to ground through a 95 ohm terminator and cannot be driven high. If the DX20 is on line (DX ON LINE = 1 and CROBAR = 1), priority selection switches on the G891 connect CB04 SEL REC to IBCB SEL 1 REC if set up for low priority or to IBCB SEL 0 REC if set up for high priority.

TI S TI STA IN (Status In) - TI STA IN is asserted if IBCB STA IN is raised when LOOP EN = 0 or if TOR CMD OUT is high when LOOP EN = 1.

TI SRV IN (Service In) - TI SRV IN is asserted if IBCB SRV IN is raised when LOOP EN = 0 or if TOR OPL OUT is high when LOOP EN =

1.

TI DIS IN (Disconnect In) - TI DIS IN is asserted if IBCB DIS IN is raised when LOOP EN = 0 or if CU RESET EN is high when LOOP EN = 1.

TI REQ IN (Request In) - TI REQ IN is asserted if IBCB REQ IN is raised when LOOP EN = 0 or if TOR SUP OUT is high when LOOP EN = 1.

TI DAT IN (Data In) - TI DAT IN is asserted if IBCB DAT IN is raised when LOOP EN = 0 or if TOR DAT OUT is high when LOOP EN = 1.

Interrupt Flags (INT FLAGS) - The INT FLAGS logic on the M8607 Channel Bus Board consists of six flags which assert UBUS INT 3 when set. DP PE FLAG, UB PE FLAG, MK PE FLAG, BUS 1 PE FLAG, and BUS 0 PE FLAG signals are the outputs of D-type flip-flops which are set by hardware when parity error conditions in the channel bus interface are detected and remain set until direct cleared by either asserting UB INIT or executing a microcontroller DATO cycle to UBRA 00 of the channel bus interface with UBUS DATA 1 asserted. The parity detection circuits for the MK 0 - 1, P IN lines and the BUS 1 IN 0 - 7, P lines are located on the M8608 Channel Extension Board; consequently, if the M8608 is not present, the MK PE FLAG is the logical AND of TIMER EN (UBRA 02 Bit 2) and -TIMER OUT (the output of a 30 millisecond retriggerable monostable multivibrator which is triggered every time UBRA 02 is written). See Section for descriptions of the conditions under which these signals are asserted.

Channel Bus Control (CB CTRL) - Logic in the channel bus control (CB CTRL) block controls requests and responses and provides timing and synchronization between the channel bus and the slave bus. The logic can be divided into the following sub-blocks: (1) channel request, (2) slave interface, (3) channel response, (4) control unit initialization, and (5) extended bus control. Each of these sub-blocks is described below with the aid of Figures 5-12 through 5-17.

Channel Request - The channel request logic senses and remembers a request, the activation of either TI SRV IN (service in) or TI DAT IN (data in), until a response is sent (Figures 5-12 and 5-13). Once a request has been made, the CLK SLVE REQ (clock slave request) signal is generated and sent to the slave interface sub-block.

Slave Interface - The slave interface logic generates requests to the data path interface over the slave bus by using the CLK SLVE REQ signal to set the SLVE REQ (slave request) flip-flop, which remains set until a response, the assertion of SLVE ACK (slave acknowledge), is received from the data path interface. If the extended bus feature is disabled (EXTENDED BUS = 0), SLVE ACK resets SLVE REQ and generates the DR READY DATA signal which is

sent to the channel response logic along with the CLK DR READY signal. If the extended bus feature is enabled (EXTENDED BUS = 1, Figure 5-14), SLVE ACK clears SLVE REQ and sets 2nd BYTE. With 2ND BYTE set, SLVE REQ is set again, approximately 130 nanoseconds after SLVE ACK is received, requesting a transaction for the second byte of data over the slave bus. The response to this transaction (SLVE ACK = 1 and 2ND BYTE = 1) causes DR READY DATA to be asserted.

The slave interface looks for the completion of a data transfer (Figure 5-15) by sampling the SLVE WOR END XFER (slave end transfer) line at the end of each channel bus transaction (request and response cycle). The setting of the END XFER flip-flop conditions the CB CTRL logic for a termination response to the next request from the channel bus.

Channel Response - This logic receives the CLK DR READY and DR READY DATA signals from the slave interface logic. These two signals cause the DR READY flip-flop to set when the request from the channel request logic is satisfied. By using the remembered request in the channel request logic and the DR READY flip-flop output, the respective response is generated, TO SRV OUT for TI SRV IN or TO DAT OUT for TI DAT IN.

Control Unit Initialization - The control unit initialization logic initiates channel bus interface activity (Figures 5-16 and 5-17). When the DX20 is operating as a control unit (CHAN MODE = 0). When DX HIGH SPEED is asserted and CHAN MODE = 0, the CU INIT (control unit initialize) flip-flop is set. The control unit initialization logic generates a request to the channel controller over the channel bus by asserting TI DAT IN. After the first request is serviced, the CU RUN (Control Unit Run) flip-flop is set and CU INIT is cleared.

Extended Bus Control - The extended bus control logic provides odd byte termination (ODD END = 1) when the 2ND BYTE flip-flop is set and SLVE WOR END XFER is asserted when a SLVE ACK is issued to the channel bus interface. This logic also enables the slave interface to request two slave bus transfers per channel bus transfer by allowing the 2ND BYTE flip-flop to toggle.

Channel Extension Control (CE CTRL) - NOTE: The channel extension control logic is implemented on the optional M8608 Channel Extension Board which has not yet been implemented. The signals which this block receives and generates have been defined and are shown in Figure 5-4.

5.3.3.6 Control Unit Control

Introduction - Some additional logic on the M8607 Channel Bus Board is used solely for control unit operation. It is described briefly below. Channel bus signal names will be referenced using the mnemonics for control unit operation.

Figure 5-12 High Speed Data Transfer, Channel Mode, Data to Device

Figure 5-13 High Speed Data Transfer, Channel Mode, Data from Device

Figure 5-14 High Speed Data Transfer, Channel Mode, Data to Device, Extended Bus Enabled

Figure 5-15 High Speed Data Transfer Termination, Channel Mode, Data to Device

Figure 5-16 Control Unit Initialization, Data to Device (Channel)

Figure 5-17 Control Unit Initialization, Data from Device (Channel)

Control Unit Reset (CU RESET) - SYS RST, SEL RST, and HALT I/O are the outputs of D-type flip-flops which are individually direct set when the control unit detects the corresponding command being issued by the channel controller and which are direct cleared when UB INIT is high, CU RESET EN is low, or CHAN MODE is high, regardless of the logic levels at the set inputs. The flip-flops may also be written by the microcontroller as UBRA 16 Bits 2 - 0, and they can be read back as UBRA 05 Bits 2 - 0 when CHAN MODE = 0. SYS RST, SEL RST, and HALT I/O are ORed to form CU RESET which asserts UBUS INT 3 when high. The conditions under which the flip-flops are set are described in Section 4.4.3.7.

Control Unit Address Comparator (CU ADR COMP) - Each control unit on the channel bus is assigned one or more contiguous addresses. In the DX20 this is accomplished by the appropriate setting of two 8-pole DIP switches in the CU ADR COMP logic on the M8607 Channel Bus Board. The poles of each switch are numbered 1 through 8, where 1 represents the most significant bit and 8 represents the least significant bit of a binary number set up in the switch; an OPEN or OFF switch represents a one and a CLOSED or ON switch represents a zero. The switch at E90 determines the range of addresses to which the DX20 will respond, where the binary number loaded into the switch is one less than the number of contiguous addresses assigned to the control unit. The number of contiguous addresses (n) is restricted to being a power of 2. The 256-location address spaces are partitioned into 256/n blocks of n locations. The switch at location E108 is loaded with the binary value of an address (any) within the block to which the DX20 must respond. The CU ADR COMP circuit looks for a match between the address placed on the BUS 0 OUT lines by the channel controller (such as during an initial selection sequence when ADR OUT is raised) and one of the addresses to which the DX20 is set up to respond. ADR MATCH is asserted when such a match occurs.

Control Unit Selection (CU SEL) - The CU SEL logic controls the states of the REC SEL FF (receive select flip-flop), TRA SEL FF (transmit select flip-flop), and SLVE SEL (slave selected) flip-flops to provide the proper response to the detection of the serially propagated select signal. SEL REC FF is set if both the serially propagated TO SEL OUT and the parallel propagated TO HLD OUT signals are both high simultaneously, indicating that a valid select signal has been received. SEL REC FF is cleared when TO HOLD OUT drops, indicating that the select signal is no longer valid. When REC SEL FF is low, TRA SEL FF is direct cleared, inhibiting the propagation of the select signal. Either SLVE SEL will set if the DX20 is selected or TRA SEL FF will set if it is not selected on the rising edge of REC SEL FF. In other words, when the select signal is received, either the control unit must block the propagation of the select signal by not setting TRA SEL FF and signal the microcontroller that it has been selected by setting the SLVE SEL flip-flop, or the control unit must propagate the select signal on to the next control unit by setting TRA SEL FF and clear SLVE SEL, if it was set, since the DX20 was not selected. There are two events for which a control unit will be

selected: an initial selection sequence for which the CU ADR COMP logic detected an address match with good parity on the BUS 0 OUT lines (TO ADR OUT = 1, ADR MATCH = 1, and BUS 0 ODD PAR = 1); or the channel controller servicing an interrupt generated by the control unit (TI REQ IN = 1 and TI ADR IN = 0). SLVE SEL will normally clear on the next rising edge of REC SEL FF for which neither of the above conditions is true, or it may be cleared by the microcontroller (writing UBRA 00 with UBUS DATA 0 = 1). The CU SEL logic generates a signal denoted ALLOW TAG/BUS OUT, the logical OR of CHAN MODE and SLVE SEL, which must be high to assert EN CB DRVRS.

5.3.4 Data Path Interface

5.3.4.1 Introduction - The data path interface logic is contained on two boards in the DX20, the M8605 Data Storage (DS) Board and the M8606 Data Formatter (DF) Board. A brief description of the logic on these boards is given in this section with the aid of the block diagrams in Figure 5-6. The data path interface circuitry has been partitioned into 29 blocks to permit easy identification of the circuit functions implemented in the interface. Drawing numbers for the circuit schematics are given next to the blocks to locate the logic circuits which correspond to each of the blocks.

5.3.4.2 Microbus Interface and Control

Introduction - The microbus interface and control logic provides communication between the microcontroller and the data path interface by allowing the microcontroller to generate clock signals and write registers in the data path interface via DATO instructions or sense the logic levels of internal signals and read registers in the interface via DATI instructions. The following blocks comprise the microbus interface and control logic.

Microbus Control (UB CTRL) - The microbus control logic on the M8605 Data Storage Board buffers and decodes Microbus signals driven from the microcontroller to generate internal data path interface signals. UB INIT is the buffered equivalent of UBUS INITIALIZE and UB RS 3 - 0 are the buffered equivalents of UBUS ADRS 3 - 0. The internal UB STB signal is asserted by UBUS STROBE only when the data path interface is selected (I/O SEL 2 asserted) and one of the implemented lower 16 decimal registers is addressed (UBUS ADRS 4 low). DATO STB is asserted when UB STB is asserted during DATO cycles (DATI low). DATI STB 0 is asserted when UB STB is asserted during DATI cycles (DATI high) which address registers 0 - 7, and DATI STB 1 is asserted when UB STB is asserted during DATI cycles which address registers 10 - 17 octal. The microbus interrupt line assigned toi the data path interface (UBUS INT 2) is asserted if either the UB PE FLAG or the DP PE FLAG is set.

Microbus Receiver (UB RCVR) - The microbus receiver on the M8605 Data Storage Board buffers and latches microbus data on the rising edge of UB STB for use during microcontroller DATO cycles. UBUS DATA 7 - 0, P is latched as UB DATA 7 - 0, P with both true and complement data available. All of the bits in the register are direct set when UB INIT is asserted.

Microbus Parity Checker (UB PAR CHK) - The microbus parity checker on the M8605 Data Storage Board checks for the correct (odd) parity of the microbus data latched in the microbus receiver register on the falling edge of DATO STB, which occurs at the end of microcontroller DATO cycles. If incorrect parity is detected, the UB PE FLAG flip-flop is set and remains set until it is cleared by either asserting UB INIT or writing register 0.

Microbus Address Demultiplexer (UB ADR DEMUX) - The microbus address demultiplexer on the M8605 Data Storage Board decodes the microbus address lines to generate a pulse on one of 16 decimal lines during microcontroller DATO cycles to the data path interface. The 16 lines correspond to microbus addresses 00 - 17 octal and are defined below.

Address	Description
00	CLR REG 0 = clear register 0. When CLR REG 0 is asserted, the flag flip-flops which comprise UBRA 00 Bits 3 - 0 are direct cleared. CLR REG 0 is also asserted by UB INIT.
01	LD REG 1 = load register 1. The trailing edge of LD REG 1 clocks the output of microbus receiver register bits 3 - 0 into UBRA 01 Read/Write bits 3 - 0.
02	LD REG 2 = load register 2. The trailing edge of LD REG 2 clocks the output of microbus receiver register bits 5 - 0 into UBRA 02 Read/Write Bits 5 - 0.
03	LD REG 3 = load register 3. UBRA 03 currently has only read only bits, so LD REG 3 is not used.
04	LD MC LB = load Massbus counter low byte. LD MC LB strobes the output of the microbus receiver register into the least significant byte of the Massbus counter (MC 07 - 00).
05	LD MC HB = load Massbus counter high byte. LD MC HB strobes the output of the microbus receiver register into the most significant byte of the Massbus counter (MC 15 - 08).
06	LD BC LB = load byte counter low byte. LD BC LB strobes the output of the microbus receiver register into the least significant byte of the byte counter (BC 07 - 00).
07	LD BC HB = load byte counter high byte. LD BC HB strobes the output of the microbus receiver register into the most significant byte of the byte counter (BC 15 - 08).
10	LO ROM ADR = load ROM address register bits 7 - 0. The trailing edge of LD ROM ADR clocks the data formatter control ROM address register (UBRA 10). Bits in the register are loaded from either the microbus receiver register or the ROM outputs as described in Section 4.4.4.10.
11	LD ROM DATA = load ROM data register. LD ROM DATA allows diagnostic programs to read the contents of the data formatter control ROM and test the ROM data register by

providing the means whereby the microcontroller can directly clock ROM data into the ROM data register. The basic procedure for examining the contents of a ROM location is given in Section 4.4.4.11.

12 SINGLE STEP = single step. When BASE CLK EN (UBRA 01 bit 1) is low, BASE CLK is normally high and is pulsed low every time a pulse is generated on SINGLE STEP.

13 HSDP INIT = high speed data path initialize. To set up the data path interface for the pending operation, a pulse must be generated on HSDP INIT by executing a microcontroller DATO to address 13 octal of the data path interface after the level of DATA TO DEV has been established and the data formatter control ROM address register (ROM ADR 8 - 0) has been loaded and prior to setting DX HIGH SPEED (UBRA 01 bit 0) to set up the data path interface for the pending operation. HSDP INIT is also generated by asserting UB INIT. The effects of asserting HSDP INIT are summarized below.

1. The master request holdoff (MSTR REQ HLDOFF) and slave request holdoff (SLVE REQ HLDOFF) flip-flops are direct cleared.

2. The clear run (CLR RUN) flip-flop is direct set, which direct clears the Run (RUN) flip-flop; and extend run (EXTEND RUN) is cleared.

3. The master receiver register (MRR 17 - 00, P), slave receiver register (SRR 7 - 0, P), and assembly register (AR 17 - 00) are cleared.

4. The master ready flip-flop and its synchronizers (MSTR RDY, MSTR RDY DLY 1, and MSTR RDY DLY 2) are direct set for pending device write operations (DATA TO DEV = 1) and direct cleared for pending read operations (DATA TO DEV = 0). The slave ready flip-flop and its synchronizers are direct set for pending device read operations and direct cleared for pending write operations.

5. The silo buffer register (SB 17 - 00) has zeroes clocked into it on the falling edge of HSDP INIT for pending device write operations (DATA TO DEV = 1) and has ones clocked into it for pending read operations (DATA TO DEV = 0). The channel buffer register (CB 7 - 0) has zeroes clocked into it on the falling edge of HSDP INIT for pending device read operations and has ones clocked into it for pending write operations.

14 SET RUN = set run. The run (RUN) flip-flop is direct set.

15 SET M R HLDOFF = set master request holdoff. The master request holdoff (MSTR REQ HLDOFF) flip-flop is direct set.

16 SET S R HLDOFF = set slave request holdoff. The slave request holdoff (SLVE REQ HLDOFF) flip-flop is direct set.

17 CLR EXTEND RUN = clear extend run. Extend run (EXTEND RUN) is cleared.

Load ROM Address (LD ROM ADR) - Further conditioning of the LD ROM ADR signal from the microbus address demultiplexer is provided by the load ROM address logic on the M8605 Data Storage Board to create a 50 nanosecond pulse on the trailing edge of LD ROM ADR called LD ROM ADR 7-4, which is used by the control ROM circuitry to clock the data formatter control ROM address register. The use of this signal is discussed in more detail in Section 5.3.4.5.

Register 1 (REG 1) - The read/write bits of register 1 (UBRA 01 Bits 3 - 0) are implemented as D-type flip-flops with both true and complement data available on the M8605 Data Storage Board. The trailing edge of LD REG 1 clocks output bits 3 - 0 of the microbus receiver register into the corresponding bits of register 1. The read/write bits in register 1 are direct cleared by asserting UB INIT.

Register 2 (REG 2) - The read/write bits of register 2 (UBRA 02 Bits 5 - 0) are implemented as D-type flip-flops on the M8605 Data Storage Board. The trailing edge of LD REG 2 clocks output bits 5 - 0 of the microbus receiver register into the corresponding bits of register 2. The combinational logic which drives the MSTR WOR END XFER and SLVE WOR END XFER wire-ORed lines is also included in the register 2 logic. The conditions under which these lines are asserted are described in Section 4.4.4.4.

Microbus Data Multiplexer (UB DATA MUX) - Microbus registers UBRA 00 - 07 in the data path interface are multiplexed onto the microbus data lines (UBUS DATA 7 - 0) via the microbus data multiplexer on the M8605 Data Storage Board. The appropriate multiplexer inputs are selected by UB RS 2 - 0 and DATI STB 0 L from the microbus control logic enables the tristate drivers which gate the data onto the microbus data lines.

Diagnostic Multiplexer (DIAG MUX) - Microbus registers UBRA 10 - 17 in the data path interface are multiplexed onto the microbus data lines (UBUS DATA 7 - 0) via the diagnostic multiplexer on the M8606 Data Formatter Board. The appropriate multiplexer inputs are selected by UB RS 2 - 0 and DATI STB 1 L from the microbus control logic enables the tristate drivers which gate the data onto the microbus data lines.

5.3.4.3 Master and Slave Bus Control

Introduction - The master bus control (MSTR BUS CTRL) and slave bus control (SLVE BUS CTRL) logic handles the request-acknowledge handshake sequence required to transfer data over the respective bus. The logic must arbitrate requests made over the busses with internal handshake signals between the master/slave bus control logic and the formatter control logic which controls data flow through the data path interface. The master bus and slave bus protocols are identical and have the same timing characteristics (See Section 5.2), lending symmetry to the design of the data path interface.

Master Bus Control (MSTR BUS CTRL) - The protocol established for communication over the master bus is described in Section 3.5. A master bus transfer is initiated by the assertion of either MSTR REQ by the Massbus interface or DIAG MSTR REQ (UBRA 01 Bit 3) by the microcontroller. The data path interface issues MSTR ACK when it has honored the request to transfer data over the master bus by latching the data on the master bus data lines into the master receiver register (MRR) during device write operations or gating data in the silo buffer register (SB) onto the master bus data lines during device read operations. The data path interface is ready to process a master bus request when MSTR RDY DLY 2 sets. The MSTR REQ HLDOFF flip-flop is direct cleared when MSTR RDY DLY 2 goes high, allowing a pending or forthcoming request to be serviced.

CLR MSTR RDY and SET MSTR ACK are logically equivalent signals which are true when MSTR REQ = 1, MSTR RDY DLY 2 = 1, and MSTR REQ HLDOFF = 0. The massbus counter (MC) is incremented and the massbus counter overflow flag (MC OF FLAG) flip-flop is clocked by the leading edge of SET MSTR ACK and the MSTR ACK flip-flop is set typically 35 nanoseconds after the leading edge of SET MSTR ACK. Simultaneously, the leading edge of CLR MSTR READY clocks master bus data into the MRR (important for device write operations) and a high on CLR MSTR RDY direct clears MSTR RDY, signalling the formatter control circuitry that a master bus transfer is in progress.

The handshake concludes with the Massbus interface clearing MSTR REQ approximately 75 nanoseconds after receiving MSTR ACK. When MSTR REQ is cleared, the MSTR REQ HLDOFF flip-flop is clocked high and the MSTR ACK flip-flop is direct cleared. MSTR REQ HLDOFF remains asserted, preventing a subsequent MSTR REQ from being serviced until MSTR RDY DLY 2 sets, signalling the master bus control that the formatter control is ready to accept, or present, new data over the master bus data lines.

Slave Bus Control (SLVE BUS CTRL) - The protocol established for communication over the slave bus is described in Section 3.6. A slave bus transfer is initiated by the assertion of either SLVE REQ by the Massbus interface or DIAG SLVE REQ (UBRA 01 Bit 2) by the microcontroller. The data path interface issues SLVE ACK when it has honored the request to transfer data over the slave bus by

latching the data on the slave bus data lines into the slave receiver register (SRR) during device read operations or gating data in the channel buffer register (CB) onto the slave bus data lines during device write operations. The data path interface is ready to process a slave bus request when SLVE RDY DLY 2 sets. The SLVE REQ HLDOFF flip-flop is direct cleared when SLVE RDY DLY 2 goes high, allowing a pending or forthcoming request to be serviced.

CLR SLVE RDY and SET SLVE ACK are logically equivalent signals which are true when SLVE REQ = 1, SLVE RDY DLY 2 = 1, and SLVE REQ HLDOFF = 0. The byte counter (BC) is incremented and the byte counter overflow flag (BC OF FLAG) flip-flop is clocked by the leading edge of SET SLVE ACK and the SLVE ACK flip-flop is set typically 35 nanoseconds after the leading edge of SET SLVE ACK. Simultaneously, the leading edge of CLR SLVE READY clocks slave bus data into the SRR (important for device read operations) and a high on CLR SLVE RDY direct clears SLVE RDY, signalling the formatter control circuitry that a slave bus transfer is in progress.

The handshake concludes with the channel bus interface clearing SLVE REQ approximately 40 nanoseconds after receiving SLVE ACK. When SLVE REQ is cleared, the SLVE REQ HLDOFF flip-flop is clocked high and the SLVE ACK flip-flop is direct cleared. SLVE REQ HLDOFF remains asserted, preventing a subsequent SLVE REQ from being serviced until SLVE RDY DLY 2 sets, signalling the slave bus control that the formatter control is ready to accept, or present, new data over the slave bus data lines.

5.3.4.4 Formatter Data Path

Introduction

Overview - The formatter data path consists of 11 logic blocks located on the M8606 Data Formatter Board which provide data paths between the master bus and slave bus data lines that are capable of transferring data in either direction and formatting the data as it is passed through the data path interface. The logic comprising the formatter control controls the transfer of data between registers and provides the signals which pack and unpack the data according to formatting programs executed by the formatter control circuitry. The upper half of Figure 5-7 shows the logic blocks in the formatter data path. The block diagram shows the inherent symmetry about the assembly register (AR); with the exceptions of the data multiplexer (DATA MUX) and the data demultiplexer (DATA DEMUX),the only major difference between corresponding blocks in the formatter data path is the width of the data path, eighteen bits on the master bus side and eight bits on the slave bus side. The formatter data path is configured for the direction of the data transfer by the level on DATA TO DEV which is buffered by the direction buffer (DIR BUF) on the M8605 Data Storage Board to form DIR TO MSTR and its complement. DIR TO MSTR is low for device write operations and high for device read

operations. The data flow involved in device write and device read operations is described briefly below, followed by separate descriptions of each block in the succeeding sections.

Device Write Operations - Referring to the M8606 Data Formatter Board block diagram (Figure 5-7), data flow is from left to right for device write operations. Data requested from the host by the Massbus interface is placed on the Massbus data lines. The data is latched in the Massbus interface and gated onto the master bus data lines. The data path interface latches the data in the master receiver register (MRR) and transfers it to the silo buffer (SB). The parity of the data in the SB is checked by the master parity (MSTR PAR) logic. The data multiplexer (DATA MUX) formats the data by setting bits in the assembly register (AR) according to a mapping function specified by the formatter control logic. When a byte has been assembled in the AR, it is transferred to the channel buffer (CB) and gated onto the slave bus data lines by the slave bus driver (SLVE BUS DRVR). The slave parity (SLVE PAR) logic generates odd parity on the data. The slave bus data is latched in the data register (DR) in the channel bus interface. The data is transmitted to the device over the channel bus data lines.

Device Read Operations - Referring to Figure 5-7 data flow is from right to left for device read operations. Data from the device is transmitted to the channel bus interface over the channel bus data lines. The data is latched in the data register (DR) in the channel bus interface and gated onto the slave bus data lines. The formatter data path latches the data in the slave receiver register (SRR) and transfers it to the channel buffer (CB). The parity of the data in the CB is checked by the slave parity (SLVE PAR) logic. The data demultiplexer (DATA DEMUX) formats the data by setting bits in the assembly register (AR) according to a mapping function provided by the formatter control logic. When a word has been assembled in the AR, it is transferred to the silo buffer (SB) and gated onto the master bus data lines by the master bus driver (MSTR BUS DRVR). The master parity (MSTR PAR) logic is latched in the Massbus interface and transmitted to the host over the Massbus data lines.

Master Receiver Register (MRR) - The master receiver register (MRR) is a register composed of nineteen D-type flip-flops which latch the master bus data and parity lines (MSTR WOR DS17 - MSTR WOR DS00, MSTR WOR DSPAR) as MRR 17 - MRR 00, MRR P. The register is used during device write operations to form a two-rank (double) buffer, the first rank being the MRR and the second rank being the SB. This allows a word to be fetched from the host via the Massbus interface while a previously fetched word in the SB is being processed by the formatting logic, thus improving throughput by pipelining operations. The MRR is clocked by the leading edge of CLR MSTR RDY, which occurs when master bus data is stable during device write operations. The register is direct cleared by HSDP INIT.

Silo Buffer (SB) - The silo buffer (SB) is a two-ported register composed of nineteen D-type flip-flops which latch the output of the MRR (MRR 17 - MRR 00, MRR P) for device write operations (DIR TO MSTR = 0) or the complemented output of the AR (-AR 17 - -AR 00) for device read operations (DIR TO MSTR = 1). The output of the SB is SB 17 - SB 00, SB P. Thus, during device write operations the SB contains true MRR data with the odd parity bit, and during device read operations the SB contains complemented AR data with SB P always low. The trailing edge of the LD SB pulse clocks the register. The SB participates in the double buffering of data for transfers in either direction. For device write operations, the SB is the second rank and the MRR is the first rank of a two-rank buffer which provides the capability described in this section. For device read operations, the SB is the second rank and the AR is the first rank of a two rank buffer; after an 18-bit word is formed in the AR, it is transferred to the SB and gated onto the master bus data lines by the MSTR BUS DRVR logic, improving throughput by allowing master bus transfers to take place while words are formed in the AR.

Master Parity (MSTR PAR) - The master parity (MSTR PAR) logic is used for checking the parity of data in the SB during device write operations and for generating correct parity for data stored in the SB during device read operations. MSTR PE (master parity error) is asserted when bad (even) parity is detected on the data in the SB; it is sensed by the formatter control logic only during device write operations. The level on MSTR PAR is significant only during device read operations when the MSTR BUS DRVR logic is enabled.

Master Bus Driver (MSTR BUS DRVR) - The master bus driver (MSTR BUS DRVR) drives the master bus data and parity lines with inverting, open-collector gates which are enabled during device read operations (DIR TO MSTR = 1). The output of the SB, which is complemented data from the AR, is inverted again by the MSTR BUS DRVR, presenting true data on the master bus. The master bus parity line is driven by the complemented MSTR PAR signal. Since SB 17 - SB 00 combined with MSTR PAR forms a 19-bit word with even parity, the inversion introduced by the MSTR BUS DRVR produces data with odd parity on the master bus.

Slave Receiver Register (SRR) - The slave receiver register (SRR) is a register composed of nine D-type flip-flops which latch the slave bus data and parity lines (SLVE WOR DF07 - SLVE WOR DF00, SLVE WOR DFPAR) as SRR 7 - SRR 0, SRR P. The register is used during device read operations to form a two-rank (double) buffer, the first rank being the SRR and the second rank being the CB. This allows a word to be accepted from the device via the channel bus interface while a previously received word in the CB is being processed by the formatting logic. The SRR is clocked by the leading edge of CLR SLVE RDY, which occurs when slave bus data is stable during device read operations. The register is direct cleared by the assertion of either SLVE END XFER (equivalent to SLVE WOR END XFER) or HSDP INIT.

Channel Buffer (CB) - The channel buffer (CB) is a two-ported register composed of nine D-type flip-flops which latch the output of the SRR (SRR 7 - SRR 0, SRR P) for device read operations (DIR TO MSTR = 1) or the complemented output of the AR (-AR 17 - -AR 00) for device write operations (DIR TO MSTR = 0). The output of the CB is CB 7 - CB 0, CB P. Thus, during device read operations the CB contains true SRR data with the odd parity bit, and during device write operations the CB contains complemented AR data with CB P always low. The trailing edge of the LD CB pulse clocks the register. Additionally, the CB P flip-flop is direct set if SLVE END XFER is asserted during a device read operation. Since the SRR is cleared by SLVE END XFER, zero data will be loaded into CB 7 - CB 0 if SLVE END XFER is asserted when a LD CB pulse is generated, but CB P will remain high so that correct (odd) parity will be presented to the SLVE PAR logic to prevent a parity error from being detected under these conditions. The CB participates in the double buffering of data for transfers in either direction. For device read operations, the CB is the second rank and the SRR is the first rank of a two-rank buffer which provides the capability described in this section. For device write operations, the CB is the second rank and the AR is the first rank of a two-rank buffer; after a byte is formed in the AR, it is transferred to the CB and gated onto the slave bus data lines by the SLVE BUS DRVR logic, improving throughput by allowing slave bus transfers to take place while bytes are formed in the AR.

Slave Parity (SLVE PAR) - The slave parity (SLVE PAR) logic is used for checking the parity of data in the CB during device read operations and for generating correct parity for data stored in the CB during device write operations. SLVE PE (slave parity error) is asserted when bad (even) parity is detected on the data in the CB; it is sensed by the formatter control logic only during device read operations. The level on SLVE PAR is significant only during device write operations when the SLVE BUS DRVR logic is enabled.

Slave Bus Driver (SLVE BUS DRVR) - The slave bus driver (SLVE BUS DRVR) drives the slave bus data and parity lines with inverting, open-collector gates which are enabled during device write operations (DIR TO MSTR = 0). The output of the CB, which is complemented data from the AR, is inverted again by the SLVE BUS DRVR, presenting true data on the slave bus. The slave bus parity line is driven by the complemented SLVE PAR signal. Since CB 7 - CB 0 combined with SLVE PAR forms a 9-bit word with even parity, the inversion introduced by the SLVE BUS DRVR produces data with odd parity on the slave bus.

Assembly Register (AR) - The assembly register (AR) is a special register consisting of 18 set-reset type flip-flops which are set individually by output signals from the DATA MUX and DATA DEMUX logic and parallel cleared by the assertion of CLR AR from the formatter control logic. The output of the AR is named AR 17 - AR 00 and both the true and complemented data are available. For

device write operations, only the DATA MUX outputs (MUX 7 - MUX 0) are enabled to set bits in the AR. Since the widest byte that needs to be formatted is 8 bits, only bits AR 08 - AR 01 are involved in device write operations. -AR 08 - -AR 01 are loaded into the CB when a LD CB pulse is generated during device write operations. For device read operations, only the DATA DEMUX outputs (DEMUX 7 Bi - DEMUX 0 Bi where i=00:02,04:06,08:15 decimal) are enabled to set bits in the AR. The entire AR is used for formatting the 18-bit words during device read operations. -AR 17 - -AR 00 are loaded into the SB when a LD SB pulse is generated during device read operations.

Data Multiplexer (DATA MUX) - The data multiplexer (DATA MUX) logic maps are master bus data stored in the SB into AR bits AR 08 - AR 01, which correspond to slave bus data lines SLVE WOR DF07 - SLVE WOR DF00, according to the mapping function supplied by the formatter control logic. The DATA MUX outputs (MUX 7 - MUX 0) are only enabled for device write operations (DIR TO MSTR = 0). The bit mapping required to implement the data formats is shown in formatter bit maps (see drawing package). The value of the data on the SHIFT 3 - SHIFT 0 lines determines which 8 consecutive bits (including end wrap-around) of the master bus data will be mapped into slave bus data by the DATA MUX logic as specified in the bit maps. An 8-bit mask (MASK 7 - MASK 0) determines which bits will be set in the AR. If and only if MASK i is set and the master bus data bit mapped into SLVE WOR DS0i is asserted, will the corresponding bit in the AR (AR 0i where i=i+1) be set when EN MUX/DEMUX is asserted, where i=0:7.

Data Demultiplexer (DATA DEMUX) - The data demultiplexer (DATA DEMUX) logic maps the slave bus data stored in the CB into AR bits AR 17 - AR 00, which correspond to master bus data lines MSTR WOR DS17 - MSTR WOR DS00, according to the mapping function supplied by the formatter control logic. The DATA DEMUX outputs (DEMUX 7 Bi - DEMUX 0 Bi where i-00:02,04:06,08:15 decimal) are only enabled for device read operations (DIR TO MSTR = 1). The bit mapping required to implement the data formats is shown in the formatter bit maps. The value of the data on the SHIFT 3 - SHIFT 0 lines determines how the slave bus data will be justified as master bus data by the DATA DEMUX logic as specified in the bit map. An 8-bit mask (MASK 7 - MASK 0) determines which slave bus data bits stored in CB 7 - CB 0 will be enabled to set bits in the AR. If and only if MASK i is set and the corresponding slave bus data bit (SLVE WOR DF0i or CB i) is asserted, will the AR bit into which the slave bus data bit was mapped be set when FN MUX/DEMUX is asserted, where i=0:7.

5.3.4.5 Formatter Control

Introduction - The formatter control logic has the task of directing the operation of all the elements in the formatter data path to transfer data through the data path interface. The data must be properly formatted and internal operations must be coordinated with external events involved with the transfer to insure proper handling of the master bus and slave bus protocols. At the heart of the formatter control logic is a 512 20-bit word control ROM (CTRL ROM) which contains 27 independent, closed-loop programs which provide device write, read forward, and read reverse capabilities for the formatting modes implemented in hardware (see Section 5.2.4.3).

Each program contains 3 to 15 instructions that are sequenced through cyclically by the formatter time base (FMTR TB). The formatter time base is driven by a crystal controlled base clock (BASE CLK) and generates clock phases (CLK PH 0 and CLK PH 1) used by the control ROM and formatter control (FMTR CTRL) to time events that occur during a ROM instruction cycle. The formatter time base can stop during the execution of ROM instructions to wait for a required external response to occur before continuing.

Direction Buffer (DIR BUF) - The direction buffer (DIR BUF) on the M8605 Data Storage Board is a simple high output drive buffer which produces both true and complemented outputs for the DIR TO MSTR signal. DATA TO DEV is buffered as -DIR TO MSTR.

Control ROM (CTRL ROM)
Hardware - There are three basic elements in the control ROM (CTRL ROM) logic on the M8606 Data Formatter Board: the 512 20-bit word ROM, the data formatter control ROM address register, and the ROM data register. The ROM consists of five 512 4-bit word programmable ROMs which are addressed by ROM ADR 8 - ROM ADR 0 and whose outputs are identified as ROM 19 - ROM 00. The data formatter control ROM address register consists of nine D-type flip-flops identified as ROM ADR 8 - ROM ADR 0. ROM ADR 8 (actually part of register 2 on the M8605 Data Storage Board) is loaded as UBRA 02 Bit 0 by a pulse on LD REG 2; ROM ADR 7 - ROM ADR 4 are loaded as UBRA 10 bits 7 - 4 by a pulse on LD ROM ADR 7-4 from the LD ROM ADR logic; and a multiplexer selects either UB DATA 3 - UB DATA 0 (DX HIGH SPEED = 0) or ROM 19 - ROM 16 (DX HIGH SPEED = 1) to be loaded on the rising edge of LD ROM ADR 7-4 from the LD ROM ADR logic or CLK PH 0 from the FMTR TR logic. See Section 4.4.4.10 for additional information. Normally, the entry point of the program to be executed is initially loaded into the data formatter control ROM address register by the microcode when DX HIGH SPEED is low. When DX HIGH SPEED is asserted, the lower four bits of the next ROM address to be executed are obtained from the ROM program. ROM output bits 15 - 00 are latched in the ROM data register by the rising edge of CLK PH 0. See Sections 4.4.4.11 and 4.4.4.12 for additional information.

ROM Instruction Organization - Each ROM word is an instruction

which contains information that controls the operation of the formatter data path and the internal operation of the formatter control. The instruction is divided into four fields as defined below.

Bit	Description
19-16	NXT ROM ADR 3 - 0 = next ROM address bits 3 - 0. Each ROM instruction specifies the lower four bits of the next instruction to be executed. This limitation reduces the maximum size of individual programs to 16 instructions and confines them to 16 word boundaries.
15-12	SHIFT 3 - 0 = shift bits 3 - 0. The SHIFT data serves two purposes in the data path interface. First, it is part of the mapping function which controls the operation of the DATA MUX and DATA DEMUX logic in the formatter data path as specified by the bit maps in the drawing package (see Section 5.3.4.4). Second, a shift code of 7 decimal is not normally used as part of a mapping function because it performs no useful mapping, but it is used during device read operations to set EXTEND RUN, which forces the formatter control logic to process all data in the formatter data path registers and pass it on to the Massbus interface. This capability allows instructions to be executed in a non normal sequence to improve throughput for formats such as the Read Industry Compatible Forward mode (see Section 5.3.4.5).
11-04	MASK 7 - 0 = mask bits 7 - 0. The MASK data is part of the mapping function which controls the operation of the DATA MUX and DATA DEMUX logic in the formatter data path as specified by the bit maps.
03	CC 3 = cycle control bit 3. If CC 3 is set, the assembly register is cleared (CLR AR asserted) prior to formatting new data into it per the mapping function specified by the MASK and SHIFT data in the instruction.
02	CC 2 = cycle control bit 2. If CC 2 is set, the instruction tags the formatter cycle during which it is executed as a slave cycle (SLVE CYC). During device write operations, the data path interface transfers a word to the channel bus interface over the slave bus before executing the instruction; and during device read operations, the data path interface obtains a word from the channel bus interface over the slave bus before executing the instruction.
01	CC 1 = cycle control bit 1. If CC 1 is set, the instruction tags the formatter cycle during which it is executed as a master cycle (MSTR CYC). During device write operations, the data path interface obtains a word from the Massbus interface over the master bus before

executing the instruction; and during device read operations, the data path interface transfers a word to the Massbus interface before executing the instruction.

00 CC 0 = cycle control bit 0. CC 0 is intended to be used only during device read forward and read reverse operations to finish executing formatting programs when the number of bytes read from the device does not correspond to the number required to complete a full pass through the program. For device write programs, CC 0 should always be low. For device read programs, CC 0 should be asserted for all but the first instruction in a program loop. When SLVE WOR END XFER is asserted by the channel bus interface during a read operation, the data path interface is enabled to continue formatting bytes of zero data without the normal slave bus handshake until the beginning of the program loop is encountered, as identified by CC 0 being low.

ROM Programs - The control ROM consists of thirty-two 16-word pages. Programs cannot extend across page boundaries because the ROM instructions can only alter the least significant four bits of the data formatter control ROM address register. Several programs, however, may be contained on a single page. There are no restrictions on the starting addresses of programs and a single program may have multiple entry points as in the case of read reverse programs.

There are twenty-seven programs in the ROM for implementing the data format modes described in Section 5.2.4.3 plus some additional diagnostic programs. A listing of all the programs in the control ROM is given in the ROM program charts (see drawing package). ROM addresses and data are specified in hexadecimal notation because the 4 and 8-bit fields of the ROM instructions and the 16-word pages lend themselves to this base. Generally, programs are coded in line to make them more readable. The columns in the ROM program listing are defined below.

1. Starting byte. The bytes transferred to the device are identified by their positions in the 18, 36, or 72-bit words from which they are unpacked during device write operations and into which they are packed during device read operations. As shown in the data format chart the bytes are numbered by the sequence in which they are written. Valid entry points to the ROM programs have starting bytes associated with them which specify the first byte that a write program will unpack from the host data or the first byte that a read program expects to receive from the device when program execution is initiated at the entry point. Note that write and read forward programs always begin execution at byte 1, but read reverse programs have multiple entry points to start formatting at any byte position to allow proper packing of bytes into words when reading records of known length

with byte counts which do not correspond to an integral number of 18, 36, or 72-bit words, depending on the data format.

2. Entry Point. All of the valid entry points for the ROM formatting programs are marked with an arrow (-->). The diagnostic programs are not so marked.

3. ROM address. The ROM address of each instruction is specified in hexadecimal.

4. ROM data. The ROM data consists of the four fields described above. They are specified in hexadecimal and dashes (-) indicate don't care fields that are programmed as zeroes. The NXT ADR field usually specifies the next sequential ROM address or, in the case of the last instruction in the sequence, points to the address of the first instruction in the program loop. The SHIFT and MASK fields determine the mapping function to be performed. It is important to note that the mapping function is carried out during every ROM instruction cycle, although it can be rendered ineffective by specifying a MASK of all zeroes.

5. CC field expanded. The cycle control field is expanded to allow easy identification of the type of instruction. Ones (1s) mark those bits which are asserted in the ROM. To maintain maximum throughput, all instruction cycles are either slave cycles (SLVE CYC) or master cycles (MSTR CYC). The clear assembly register (CLR AR) function is performed before the mapping function and is usually specified in instructions which begin formatting new bytes in write programs or new words in read programs. The continue (CONT) bit is low for write programs and asserted for all but the first instructions in the loops of read forward and read reverse programs.

6. Cycle type/byte or word. All instruction (with the exception of some of those in the diagnostic programs) execute either slave (S) cycles or master (M) cycles during which bytes or words are transferred over the respective bus. The actual byte or word involved in the transfer is given, where the number corresponds to byte or word position in the format as shown in the ROM program charts.

7. Comment field. Each program name is preceded by an asterisk (*). The basic function of each of the instructions is specified in the comments that follow it. Write programs get words from the host during master cycles and send bytes to the device during slave cycles. One or two instruction cycles are needed to form a complete byte as identified by the -P (partial) and -C (complete) suffixes appended to the format byte comment.

Read programs obtain bytes from the device during slave cycles and send words to the host during master cycles. Several instruction cycles may be needed to form a complete word as identified by a -P and -C suffixes appended to the format word comments.

For the purpose of the following discussion, the diagnostic programs will be ignored. Without exception, write program instructions are executed in sequential order of increasing addresses with the last instruction looping back to the first instruction in the sequence. In general, read forward programs are sequential in nature also, with the exception of the program for the industry compatible mode which loops from the last instruction to the second instruction in the sequence (see formatter control). The read reverse programs have multiple entry points, some of which are to instructions outside the program loop (at the end of the program). An instruction at a nonsequential entry point, however, always returns control to the loop with a jump to an appropriate point in the loop. These nonsequential instructions are only executed once upon starting a read reverse operation and invariably send an initial 18-bit word of zero data to the host. They are used in the PDP-10/20 modes to send an even numbered 18-bit word of zero data before an odd numbered word in the program loop to properly justify the first odd numbered 18-bit word, and all succeeding odd words, in the right half of the 36-bit host memory words.

Base Clock (BASE CLK) - The formatter control logic is driven by a 21.84 megahertz oscillator source in the base clock (BASE CLK) logic on the M8605 Data Storage Board. The BASE CLK logic is controlled by two signals generated by the microbus interface and control logic. BASE CLK EN (UBRA 01 Bit 1) permits the 21.84 megahertz signal to be gated through to BASE CLK when it is asserted and holds BASE CLK high when it is low. If SINGLE STEP is pulsed by writing UBRA 12 when BASE CLK EN is low, BASE CLK will be pulsed low. These two control signals are useful during diagnostic testing of the data path interface.

Formatter Time Base (FMTR TB) - The formatter time base (FMTR TB) on the M8606 Data Formatter Board generates the clock signals which are decoded by the formatter control logic to time events that occur during instruction cycles. The FMTR TB logic can be divided into two basic parts: the clock phase generator, and the start/stop logic. Reference will be made to the formatter time base waveforms in Figure 5-18 in the description that follows.

Clock Phase Generator - A two-stage twisted-tail ring counter (Mobius counter) generates two clock phases (CLK PH 0 and CLK PH 1) from the BASE CLK signal with both the true and complemented outputs available. As shown in Figure 5-18, the phase changes occur on the leading edge of BASE CLK with the rising edge of CLK PH 0 leading the rising edge of CLK PH 1. If RUN (from the start/stop logic) is held high, the clock phases will run uninterrupted with a change occurring in one of the phases every BASE CLK

Figure 5-18 Formatter Control

period. If RUN goes low, the clock phases run uninterrupted until they both go low, at which time clock phases changes stop until run is asserted again. The four possible time states are labelled T0 - T3 as shown in Figure 5-18 and define an instruction cycle.

Start/Stop Logic - The start/stop logic controls the initial starting of the FMTR TB and the interruption of clock phase changes in state T0 during normal ROM instruction execution. Two flip-flops are involved in this activity. The RUN flip-flop synchronizes external events with the ROM program by stopping the clock phase generator at the beginning of an instruction cycle (T0) until an external condition required for that instruction is satisfied. Data at the input to the RUN flip-flop is clocked into it on the rising edge of BASE CLK. The logic equation shown for RUN in Figure 5-18 defines the conditions required to execute an instruction. ROM 02 - ROM 00 is the data that is clocked into CC 2 - 0 at T1, when instruction execution resumes. The significance of each item in the equation is given below.

1. The first term is asserted during master cycles when MSTR RDY DLY 2 goes low, an event which signals the FMTR TB that master bus data required for the instruction has been clocked into the MRR during a device write operation or that master bus data for the previous master cycle has been transferred to the Massbus interface during a read operation.

2. The second term is asserted during slave cycles when SLVE RDY DLY 2 goes low, an event which signals the FMTR TB that slave bus data required for the instruction has been clocked into the SRR during a device read operation or that slave bus data for the previous slave cycle has been transferred to the channel bus interface during a write operation.

3. The third term is asserted during slave cycles when SLVE END XFER (equivalent to SLVE WOR END XFER) is asserted by the channel bus interface and the continue cycle control bit is asserted. This condition occurs during device read operations (ROM 00 is not asserted during write operations.) When the device signals the channel bus interface that it has finished reading a record, but the data path interface is not at the beginning of its formatting program. During such slave cycles which do not involve actual slave bus transfers, bytes of zero data are processed by the formatter data path logic to complete an 18, 36, or 72-bit word, depending on the data format.

4. The fourth term is asserted during slave cycles when SLVE END XFER is asserted by the channel bus interface and EXTEND RUN is set. This condition also occurs during device read operations when the device signals the channel bus interface that it has finished reading a

record, but the data path interface has not transferred all the data in its internal registers to the Massbus interface because of a non-normal instruction sequence in a read program that set EXTEND RUN. The EXTEND RUN logic is discussed in this section.

RUN can be direct set for diagnostic testing by asserting SET RUN (writing UBRA 14) and is direct cleared by setting the CLR RUN flip-flop. The CLR RUN flip-flop synchronizes DX HIGH SPEED with the FMTR TB; CLR RUN assumes the level of -DX HIGH SPEED on the rising edge of BASE CLK, allowing the RUN flip-flop to assume the level of the data at its input on the next rising edge of BASE CLK after CLR RUN goes low. CLR RUN is direct set by HSDP INIT (writing UBRA 13) to insure proper initialization of the data path interface and direct cleared by SET RUN (writing UBRA 14) to allow SET RUN to direct set RUN.

Formatter Control (FMTR CTRL) - The formatter control (FMTR CTRL) block consists of control logic on the M8606 Data Formatter Board which can be partitioned into the following sub-blocks: data path register control, master synchronizer, slave synchronizer, extend run, data path parity error flag, and end transfer.

Data Path Register Control - The LD SB, LD CB, CLR AR, and EN MUX/DEMUX signals are generated by the FMTR CTRL logic. The effects of these signals have already been described briefly in Section 4.4.4.15 and in more detail in Section 5.3.4.4. The instruction cycle timing diagram in Figure 5-18 shows the time states during which they clock, clear, and/or enable the various elements of the formatter data path. LD SB and LD CB are asserted during state T0 for master cycle (ROM 01 = 1) and slave cycle (ROM 02 = 1) instructions, respectively, if CLR RUN is not asserted. All register and flip-flop clocking occurs on the trailing edge of the LD SB and LD CB pulses. CLR AR is asserted during timing state T2 if ROM cycle control bit 3 is asserted (CC 3 = 1). The EN MUX/DEMUX signal is asserted every instruction cycle during the last half of timing state T2 and all of T3.

Master Synchronizer - The MSTR RDY flip-flop and its synchronizers (MSTR RDY DLY 1 and MSTR RDY DLY 2) comprise the master synchronizer. The manner in which these flip-flops are controlled has been described in Section 4.4.4.17, the role they play in controlling master bus transfers in Section 5.3.4.3, and the means by which the master synchronizer coordinates master bus transfers with the execution of the ROM program through the start/stop logic in the FMTR TB. The two stages of synchronization serve two purposes. First, the synchronizers reduce the probability of error within the data path interface in resynchronizing the clearing of MSTR RDY by CLR MSTR RDY, an asynchronous event, to less than one error in 10 years. Second, the setting of MSTR RDY DLY 2, which permits MSTR ACK to be asserted, is delayed approximately 90 nanoseconds after the trailing edge of LD SB. This insures that master bus data will be stable before MSTR ACK is issued during device read operations and allows enough time for

data to stabilize in the MRR before being clocked into the SB register during device write operations.

Slave Synchronizer - The SLVE RDY flip-flop and its synchronizers (SLVE RDY DLY 1 and SLVE RDY DLY 2) comprise the slave synchronizer. The manner in which these flip-flops are controlled has been described in Section 4.3.4.17 and the role they play in controlling slave bus transfers is explained in Section 5.3.4.3 and the means by which the slave synchronizer coordinates slave bus transfers with the execution of the ROM program through the start/stop logic in the FMTR TB. The two stages of synchronization serve two purposes. First, the synchronizers reduce the probability of error within the data path interface in resynchronizing the clearing of SLVE RDY by CLR SLVE RDY, an asynchronous event, to less than one error in 10 years. Second, the setting of SLVE RDY DLY 2, which permits SLVE ACK to be asserted, is delayed approximately 90 nanoseconds after the trailing edge of LD CB. This insures that slave bus data will be stable before SLVE ACK is issued during device write operations and allows enough time for data to stabilize in the SRR before being clocked into the CB register during device read operations.

Extend Run - The extend run logic has been necessitated by the use of non normal instruction sequences, namely in the industry compatible read-forward program. It is desirable to always sandwich at least two slave cycles between successive master cycles so that master bus transfers can be overlapped with slave bus transfers with little or no reduction in throughput due to the transfer rate limitations of the Massbus interface. In every formatting program but one, the normal sequence of words and bytes results in master cycles being separated by at least two slave cycles. For the industry compatible read-forward mode, however, it is only necessary to get byte 4 from the device between sending words 1 and 2 to the host. The problem has been resolved by getting byte 1 for the next word immediately after byte 4, but not formatting it until the master cycle instruction which sends word 2 to the host. Then, the program loops back to the instruction which gets byte 2 instead of the entry point instruction. The extend run logic is needed to handle the case in which the number of bytes involved in the transfer is one greater than a multiple of 4. Were it not for the additional logic, the program would stop after sending the second word containing the second to last byte (byte 4) and never finish formatting the additional word with the last byte (byte 1) in it or send it on to the host.

EXTEND RUN is set at the beginning of timing state T2 if SLVE WOR END XFER is not asserted and the SHIFT code of the instruction being executed during that cycle is 7. It is cleared as a matter of course during timing state T1 of an instruction in which the continue cycle control bit (CC 0) is not set. Asserting HSDP INIT (writing UBRA 13) or CLR EXTEND RUN (writing UBRA 17) will also clear EXTEND RUN.

Referring again to the industry compatible read-forward program,

note that the SHIFT code for the instruction that gets byte 1 (address 00D hexadecimal) is 7, which will cause EXTEND RUN to set when the instruction is executed. Since byte 1 is the last byte, the program will stop after sending word 2, waiting for another byte (byte 2) before proceeding. When the device signals the end of the transfer, the channel bus interface will assert SLVE WOR END XFER. If EXTEND RUN were not set, the program would still not execute another instruction because the continue cycle control bit is low, but EXTEND RUN overrides CC 0 as described in this section causing the instruction at address 009 hexadecimal to be executed, which clears EXTEND RUN. Bytes of zero data are formatted into two additional 18-bit words which are sent to the host. Note that when the instruction at address 00D hexadecimal is executed this time, SLVE WOR END XFER is still asserted, preventing EXTEND RUN from being set a second time. Consequently, the program stops after sending word 2.

Data Path Parity Error Flag - The DP PE FLAG flip-flop is clocked at the beginning of timing state T2 during every instruction cycle. During slave cycle instructions for device read operations (DIR TO MSTR = 1), the DP PE FLAG will set if SLVE PE from the SLVE PAR logic in the formatter data path is asserted at the beginning of state T2, indicating that bad (even) parity has been detected in the slave bus data stored in the CB register. During master cycle instructions for device write operations (DIR TO MSTR = 0), the DP PE FLAG will set if MSTR PE from the MSTR PAR logic in the formatter data path is asserted at the beginning of state T2, indicating that bad (even) parity has been detected in the master bus data stored in the SB register. Once set, the DP PE FLAG can only be cleared by asserting CLR REG 0 (See Section 5.3.4.2).

End Transfer - The end transfer logic generates SLVE END XFER and FMTR END XFER. SLVE END XFER is the buffered equivalent of SLVE WOR END XFER. FMTR END XFER is asserted when SLVE WOR END XFER is asserted, RUN is low, and the continue cycle control bit (ROM 00) is low, indicating that the data path interface has completed formatting all the data received from the device during device read operations.

CHAPTER 6
PREVENTIVE MAINTENANCE

(To Be Supplied)

(To Be Supplied)