Data General

Customer Documentation

Designing Interfaces for the ECLIPSE® I/O Bus

Designing Interfaces for the ECLIPSE® I/O Bus

014-001856-00

Ordering No. 014-001856

Printed in the United States of America
Rev. 00, January 1990

Notice

DGC reserves the right to make changes in specifications and other information contained in this document without prior notice, and the reader should in all cases consult DGC to determine whether any such changes have been made.

THE TERMS AND CONDITIONS GOVERNING THE SALE OF DGC HARDWARE PRODUCTS AND THE LICENSING OF DGC SOFTWARE CONSIST SOLELY OF THOSE SET FORTH IN THE WRITTEN CONTRACTS BETWEEN DGC AND ITS CUSTOMERS. NO REPRESENTATION OR OTHER AFFIRMATION OF FACT CONTAINED IN THIS DOCUMENT INCLUDING BUT NOT LIMITED TO STATEMENTS REGARDING CAPACITY, RESPONSE-TIME PERFORMANCE, SUITABILITY FOR USE OR PERFORMANCE OF PRODUCTS DESCRIBED HEREIN SHALL BE DEEMED TO BE A WARRANTY BY DGC FOR ANY PURPOSE, OR GIVE RISE TO ANY LIABILITY OF DGC WHATSOEVER.

IN NO EVENT SHALL DGC BE LIABLE FOR ANY INCIDENTAL, INDIRECT, SPECIAL OR CONSEQUENTIAL DAMAGES WHATSOEVER (INCLUDING BUT NOT LIMITED TO LOST PROFITS) ARISING OUT OF OR RELATED TO THIS DOCUMENT OR THE INFORMATION CONTAINED IN IT, EVEN IF DGC HAS BEEN ADVISED, KNEW OR SHOULD HAVE KNOWN OF THE POSSIBILITY OF SUCH DAMAGES.

Designing Interfaces for the ECLIPSE® I/O Bus
014–001856–00

Revision History:

Original Release – January 1990

Preface

The ECLIPSE I/O bus is a characteristic feature of most models in the ECLIPSE® MV/Family of computers.

The input/output subsystem of a computer featuring an ECLIPSE I/O bus consists of an input/output controller (IOC) or equivalent circuitry that controls the input/output bus and the burst multiplexor channel (BMC) controller that controls the BMC bus. In this manual, for the sake of clarity, we refer to the IOC as the input/output bus controller and the BMC controller as the BMC bus controller.

This manual describes how to design input/output (I/O) bus interface circuitry for custom device controllers that connect to the Data General ECLIPSE I/O bus. While little knowledge of the input/output architecture of the system is needed to use a peripheral device sold by Data General Corporation (DGC), the design of an input/output device controller requires a much greater understanding of this architecture. This manual describes the programmed input/output (PIO), data channel (DCH), and burst multiplexor channel (BMC) facilities of the system and describes the structure, functions, signals, and timing characteristics of their buses. The manual also describes the general-purpose interface boards that Data General makes available to simplify the job of designing and building an interface. The reader should have a working knowledge of digital circuits and some experience with digital computers.

Organization of This Manual

The manual is divided into chapters as follows:

Chapter 1 introduces the standard ECLIPSE I/O bus subsystem and describes how it integrates with ECLIPSE MV/Family computers.

Chapter 2 describes the various I/O bus functions and the bus signals that are used for each.

Chapter 3 presents detailed timing characteristics and signal levels for the various I/O bus functions, provides circuit specifications for line drivers and receivers that are to be connected to the I/O bus, and lists suggestions for minimizing electrical noise.

Chapter 4 describes the various BMC bus functions and the bus signals that are used for each.

Chapter 5 presents functional timing characteristics and signal levels for the various BMC bus functions. This chapter also describes how to connect to and terminate the BMC bus.

Chapter 6 describes how to connect device controllers in an ECLIPSE MV/Family computer chassis to adapters and devices outside the chassis, how to install internal and external I/O cables, and how to terminate the buses.

Chapter 7 provides information on interfacing boards available from Data General Corporation that will aid the user in constructing custom device controller boards.

Chapter 8 describes the Model 4040A General Purpose I/O Interface Board, a 15-inch by 15-inch printed circuit board that provides basic I/O bus interface circuitry and an area where a user can construct custom device controller circuitry.

Reader, Please Note:

The following sections describe conventions in this manual.

Backpanel and Backplane

In this manual, *backpanel* and *backplane* are synonymous, both referring to the interconnecting printed-circuit board that passes bus and power signals to other boards. The *backpanel* (*backplane*) contains connectors into which the boards of the system are plugged.

Logic Functions

Data General logic prints are drawn in close accordance with Mil-Std-806C. With this convention, logical functions are drawn as physically implemented; that is, where discrete gates are used to implement a function, those gates are shown. On the other hand, where a more complex integrated circuit is used (for instance, a multiplexor) it is shown as a rectangular box instead of showing the gates comprising the function.

Signal Levels

Throughout this manual, we frequently make a distinction between electrical levels and logical values. To minimize confusion, electrical levels are always indicated by an H or L, and logical values by a 1 or 0. As an electrical level, an H indicates that the signal is high (greater then +2.0 volts) and an L indicates that it is low (less than +0.7 volts). An asserted, or true, signal is indicated by a logical 1, and a false signal by a 0.

Signal Names

The voltage level at which a signal asserts true is a matter of definition. To distinguish between signals that are asserted high (0 = L, 1 = H) and those that are asserted low (0 = H, 1 = L), we use a naming convention that defines the relationship between the logical value and electrical level of a signal. If a horizontal bar appears over the name, as $\overline{\textbf{WRITE}}$, then that signal is asserted when it is at a low electrical level; conversely, a signal without the bar, **WRITE**, is asserted when high. To ease the notation in text and in tables, signals that assert low are shown with a ^ preceeding the signal name; for example ^WRITE asserts low.

The following summary shows the signal notation used within the manual:

Where Used	**Asserts High**	**Asserts Low**
Text	SIGNALNAME	^SIGNALNAME
Tables	**SIGNALNAME**	**^SIGNALNAME**
Figures	SIGNALNAME	$\overline{\text{SIGNALNAME}}$

To be expressed, logical functions may often require more than one binary signal. For instance, three lines are required to express an octal digit. Generally, these closely related signals are individually named by adding an identifying number to a common label. For instance, suppose that BUS0 through BUS5 are all required to completely specify a function. All or part of such a group of signals is identified by angle brackets around the range of items included; for example, BUS<0–5>. In this case, the range carries the information that there are six BUS lines, BUS0 through BUS5, inclusive.

Related Manuals

This manual is one of a series of technical manuals available for your ECLIPSE MV/Family computer. These manuals provide detailed technical information on configuring, starting, operating, and programming the system hardware. In general, the manuals are oriented to those with computer hardware background. For designing your software interface (at the assembly language level), you will need the *ECLIPSE MV/Family (32–Bit) Systems Principles of Operation* manual (014–0001371) as well as the *Principles of Operation Supplement* for your specific system. For further information on manuals pertaining to your system, contact you Data General sales representative.

Contacting Data General

Data General wants to assist you in any way it can to help you use its products. Please feel free to contact the company as outlined below.

Manuals

If you require additional manuals, please use the enclosed TIPS order form (United States only) or contact your local Data General sales representative.

If you have comments on this manual, please use the prepaid Comment Form that appears at the back. We want to know what you like and dislike about this manual.

Telephone Assistance

If you are unable to solve a problem using any manual you received with your system, and you are within the United States or Canada, contact the Data General Service Center by calling 1–800–DG–HELPS for toll-free telephone support. The center will put you in touch with a member of Data General's telephone assistance staff who can answer your questions.

Free telephone assistance is available with your warranty and with most Data General service options. Lines are open from 8:30 a.m. to 8:30 p.m., Eastern Time, Monday through Friday.

For telephone assistance outside the United States or Canada, ask your Data General sales representative for the appropriate telephone number.

End of Preface

Contents

Chapter 1 – Introduction

Chapter 2 – The I/O Bus

Chapter 3 – I/O Bus Timing and Electrical Specifications

Chapter 4 – The BMC Bus

Chapter 5 – BMC Bus Specifications

Chapter 6 – Device Connections and Connectors

Chapter 7 – Device Controller Boards

Chapter 8 – Model 4040A General–Purpose I/O Interface Board

Tables

Table

Figures

Figure

Chapter 1
Introduction

This chapter introduces the standard ECLIPSE I/O bus subsystem and describes how it integrates with ECLIPSE® MV/Family computers.

As shown in Figure 1-1, the ECLIPSE I/O bus subsystem connects peripheral devices, via their controllers, with the host computer. The ECLIPSPE I/O bus subsystem consists of two buses: a data channel and input/output bus (referred to as the I/O bus), and a burst multiplexor channel (BMC) bus. Each of these buses consists of several conductors used for transmitting control and information signals between the computer and the device controllers. The I/O bus is shared by all the device controllers as well as by the computer, while the BMC bus is shared only by the device controllers that incorporate the BMC facility and the computer. Since these buses are shared by both the computer and the peripherals, they are, by necessity, half-duplex buses; that is, only one operation can occur at any one time on a particular bus. However, both buses may have operations occurring simultaneously, independent of each other.

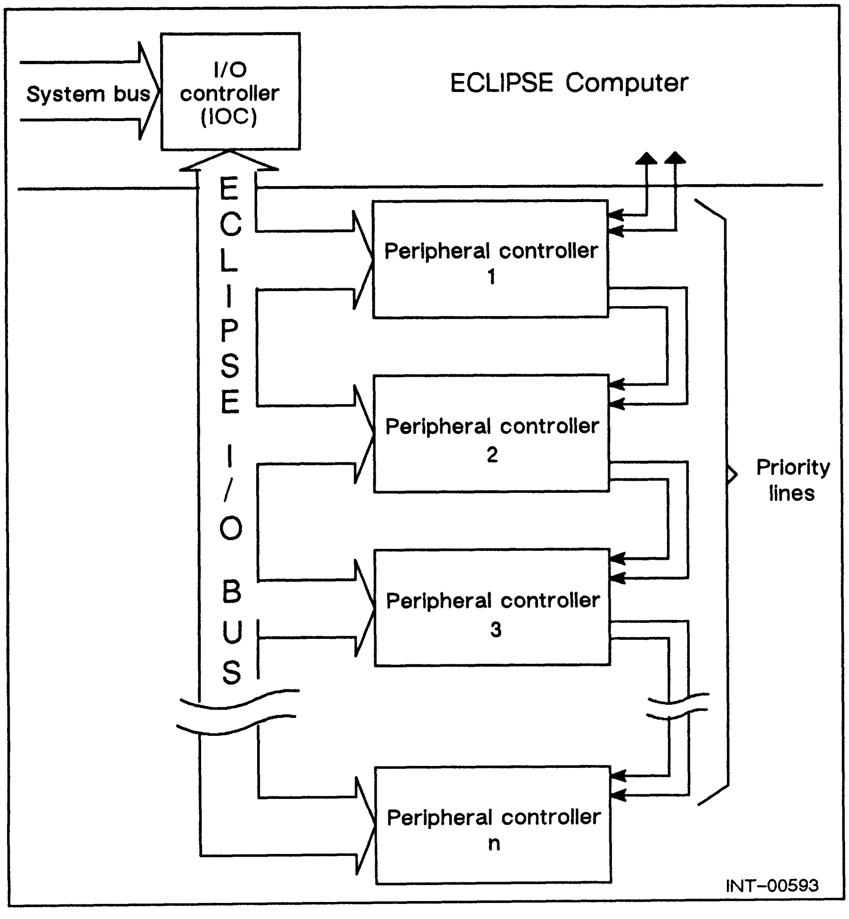


Figure 1-1 I/O Bus Interconnection

The BMC bus is the preferred I/O link for mass storage devices. Since the channel has to be activated only once and a memory address has to be supplied only once for each burst of words transferred, block transfers over the BMC can occur at much higher rates than those under I/O.

The direction of all information transfers on the buses is relative to the computer. *Output* always refers to information moving from the computer to a device controller; *input* always refers to information moving from a device controller to the computer.

Integration with System

A basic ECLIPSE computer system using the ECLIPSE I/O bus consists of a computer, its peripheral devices, and an I/O bus that connects the peripheral devices with the computer. Figure 1–2 shows these parts of the system as well as other components mentioned in the following paragraphs and it shows their interconnections.

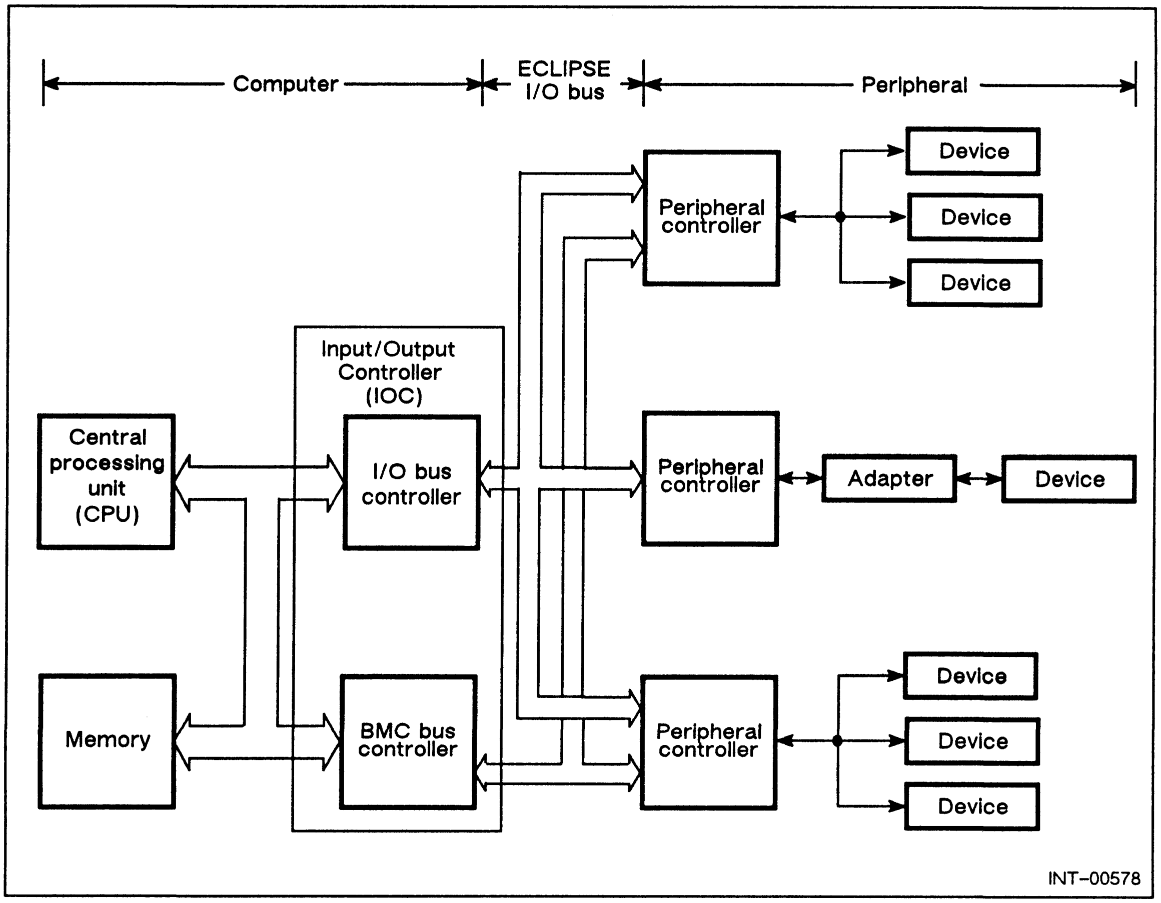


Figure 1–2 An ECLIPSE Computer System

Computer

A basic ECLIPSE computer consists of a central processing unit (CPU), a memory, and an IOC. The IOC incorporates the two types of I/O bus controllers: an I/O bus controller and a BMC bus controller. The computer may include other elements, such as a hardware floating-point unit (FPU), but Figure 1–2 shows the simplest implementation.

Input/Output

Input/output is the process of moving information in a computer system between the computer and peripheral devices such as line printers, terminals, and mass storage devices (tape and disk units). Some devices, such as line printers, transfer information out of the system. Some devices, such as terminals, transfer information both into and out of the system. Others, such as mass storage devices, not only transfer information into or out of the system, but also store information within the system. Devices, therefore, serve two main purposes: they provide the computer system with a means of communicating with its surroundings, and they supplement memory with secondary information storage capacity.

Input/Output Controller (IOC)

The IOC generates the ECLIPSE I/O bus and acts as an intermediary between I/O devices on the bus and the rest of the system. It communicates with the CPU and memory via the internal system buses and with peripheral devices via the ECLIPSE I/O bus. To the operating system, the IOC appears as an I/O channel, supporting data channel and burst multiplexor channel (BMC) devices with internal and external bus protocol transformations. It maintains maps that change data channel and BMC addresses to physical memory addresses. Although the IOC allows intelligent device controllers to load their own memory maps, it also maintains protection bits that prevent devices from gaining access to protected memory areas. Several IOCs can be present in some ECLIPSE MV/Family systems, each generating an ECLIPSE I/O bus. Other implementations of ECLIPSE MV/Familiy systems may have only one IOC and thus only one ECLIPSE I/O bus. When more than one ECLIPSE I/O bus is present, they are identical.

Peripherals

A *peripheral* generally consists of several units: a device or devices; a device controller; and, sometimes, an adapter.

Devices

A *device* is a unit that reads, writes, stores, or processes information. Devices include disk drives, magnetic tape transports, and display and hardcopy terminals. For example, a terminal's keyboard *reads* information; a plotter *writes* information; a disk drive and magnetic tape transport *stores* information; and an analog–to–digital (A/D) converter *processes* information.

Device Controller

A *device controller* is the interface between the computer and the device, interpreting commands from the computer to the device and passing information between them. Thus, a moving–head disk controller can translate the track address received from the computer into positional commands for the disk drive's access mechanism. Once the access mechanism positions the read/write heads, the disk controller translates the data words it receives from the computer into the sequence of serial bits required by the disk drive write circuitry. Likewise the disk controller translates the sequence of serial bits it receives from the disk drive read circuitry into the data words required by the computer.

Adapter

An *adapter* is an additional unit required by some peripherals to complete the communications link between the device and the device controller. An example of an adapter is a modulator–demodulator (modem) connected between a communications controller and a remote terminal device.

End of Chapter

Chapter 2
The I/O Bus

The data channel and input/output (I/O) bus is a single bus connecting an I/O controller (IOC) to the device controllers for peripheral devices (see Chapter 1). The I/O bus handles both the programmed input/output and data channel operations of the computer system.

Data and addresses are transferred on the bus along 16 parallel, bidirectional data lines. Data and address transfers are synchronous; no *handshaking* occurs between the interface and the I/O bus controller. Control signals are carried along dedicated, unidirectional control lines that specify a unique function to be performed. In addition to specifying a unique function, each control signal generated by the I/O bus controller provides all timing necessary to perform that function. The data channel and program interrupt facilities each uses its own single request and single priority lines. The two request lines run in parallel to all peripheral interfaces so that an interface requiring either data channel or program interrupt service need only assert the appropriate line and wait for the response. The serial priority lines are independent and are chained from interface to interface so that priority for service is granted to the interface closest on the chain to the I/O bus controller. The I/O bus is transistor–transistor logic (TTL) compatible.

The I/O bus signals are carried by etching contained on the chassis' printed circuit backpanel (backplane). When there are less than three input/output controllers in a system, internal cabling carries the I/O bus signals between the groups of I/O printed circuit board slots designated for each I/O port. In addition, the ECLIPSE I/O bus(es) can be connected to an expansion chassis when system configurations require expansion. A bus repeater printed circuit board is required when connecting the I/O bus(es) to an expansion chassis. The I/O bus signals are terminated by resistor networks that are solder–mounted on the chassis' backpanel.

I/O Bus Signals

The I/O bus comprises 48 signals that can be divided functionally into 5 groups. Figure 2–1 shows the I/O bus signals in their respective groups and Table 2–1 provides a brief description of each signal's function. (Refer to "Reader, Please Note" in the Preface for signal level and signal name descriptions.) Specific timing information for the I/O bus signals is shown in Chapter 3. Note that, with the exception of the two priority lines, all I/O bus signal lines run in parallel to the I/O bus controller and all device controllers connected to the bus.

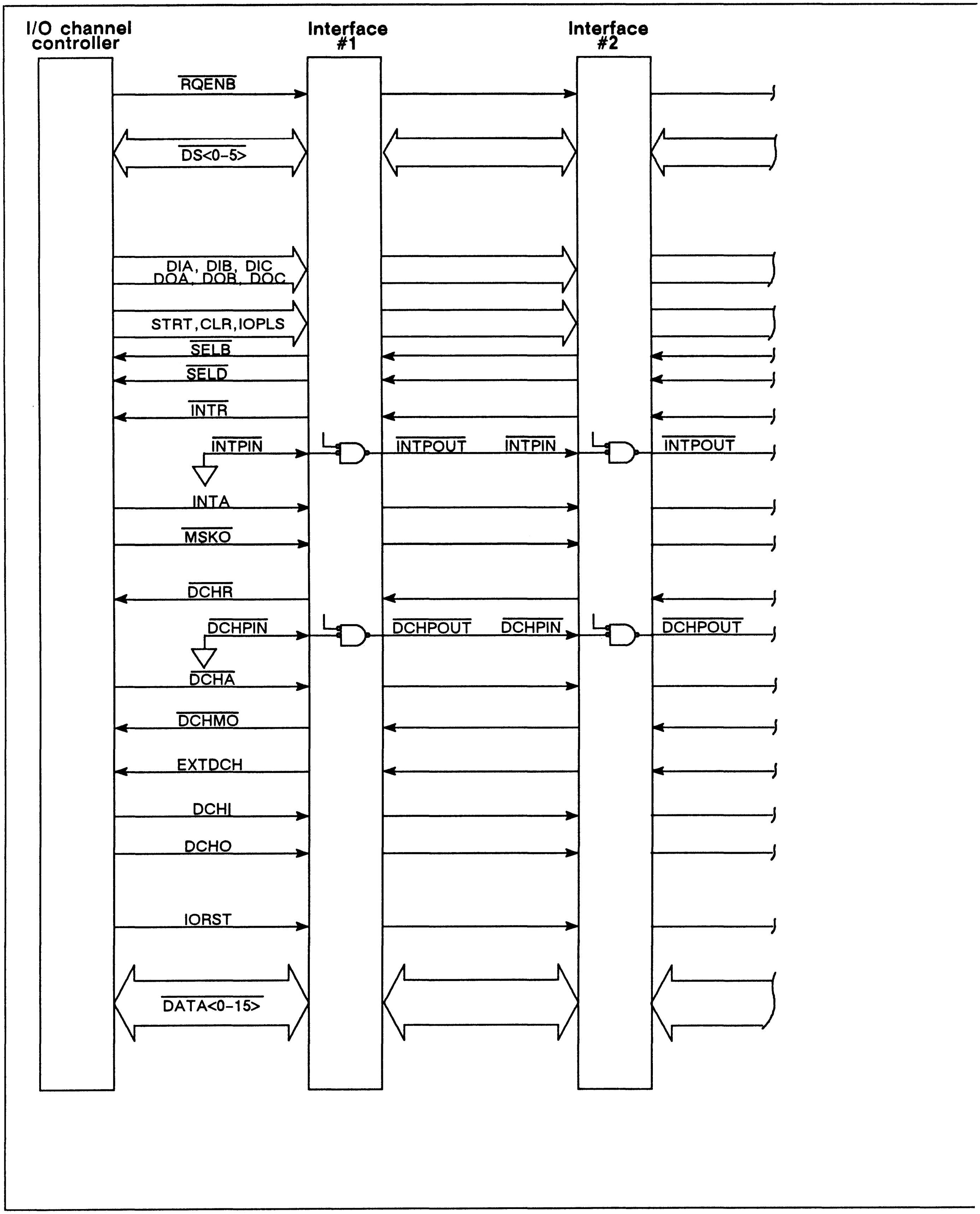
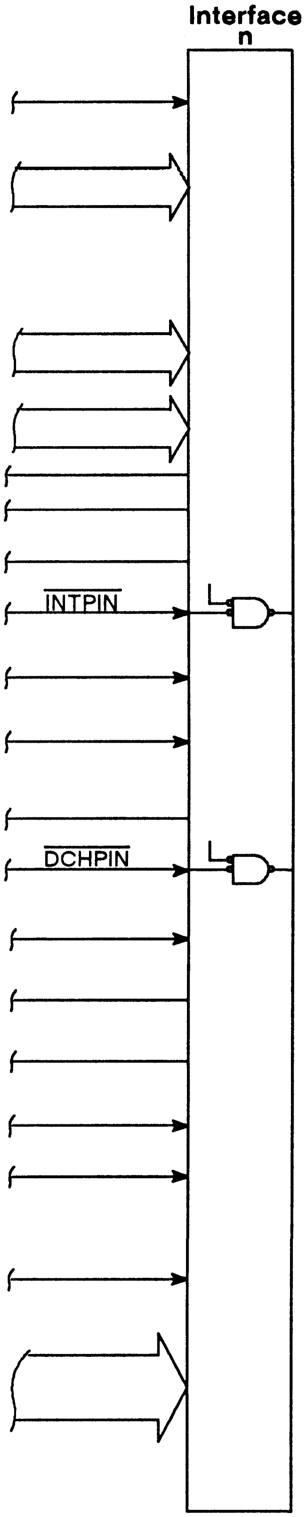
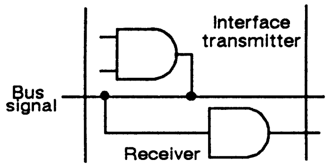


Figure 2–1 The I/O bus



Group	Signals	Description
ENABLE	RQENB	Enables interrupt request and data channel requests
PROGRAMMED I/O	DS<0–5>	DS<0–5> to device controller – Programmed I/O device select DS<0> to I/O channel controller – Data channel transfer mode select (see also DCHMO) DS<4–5> to I/O channel controller – Data channel memory mapping select (see also EXTDCH)
	DIA, DIB, DIC, DOA, DOB, DOC	Data input or data output command strobe
	STRT, CLR, IOPLS	Flag control or flag test command
	SELB	Select Busy flag
	SELD	Select Done flag
PROGRAM INTERRUPT	INTR	Interrupt request
	INTPIN/INTPOUT	Serial interrupt priority chain
	INTA	Interrupt acknowledge
	MSKO	Interrupt mask out strobe
DATA CHANNEL	DCHR	Data channel request
	DCHPIN/DCHPOUT	Serial data channel priority chain
	DCHA	Data channel acknowledge
	DCHMO	Data channel transfer mode select
	EXTDCH	Data channel memory mapping select (see also DS<4–5>
	DCHI	Data channel input strobe
	DCHO	Data channel output strobe
SYSTEM CONTROL	IORST	I/O reset command
DATA & ADDRESS	DATA<0–15>	Transfers: Programmed I/O data Interrupt acknowledge device code Data channel address Data channel data

NOTE: All interface to bus connections are parallel connections (see diagram) unless shown otherwise.



INT–00594

Table 2-1 I/O Bus Signals

Signal	Description
Data	
^DATA<0-15>	*Data*. All data and addresses, for both programmed I/O and data channel, are transferred between the I/O bus controller and device controllers connected to the I/O bus via these 16 bidirectional lines. The Interrupt Disable mask and Interrupt Acknowledge information are also carried on these lines.
Programmed I/O	
^DS<0-5>	*Device Select*. These lines, when asserted by the I/O bus controller, carry the low-order 6 bits of an I/O instruction; that is, the device code. Only the device controller whose device code corresponds to that carried on these lines should respond to control signals generated on the I/O bus. ^DS<0> is bidirectional and can be asserted by device controllers to select data channel transfer mode. (See the ^DCHMO entry later in this table.) ^DS<4-5> are bidirectional and can be used by device controllers for data channel map selection during data transfers (see section "Data Channel Map Slot Selection"). ^DS<4-5> can also be used when loading data channel map slots from the device controller. (See the ^EXTDCH entry later in this table.)
DATIA	*Data In A*. This signal line is asserted by the I/O bus controller during the execution of a *Data In A* (DIA) instruction. Should cause the device controller selected by ^DS<0-5> to place the contents of its A input buffer on ^DATA<0-15>.
DATIB	*Data In B*. This signal line is asserted by the I/O bus controller during the execution of a *Data In B* (DIB) instruction. Should cause the device controller selected by ^DS<0-5> to place the contents of its B input buffer on ^DATA<0-15>.
DATIC	*Data In C*. This signal line is asserted by the I/O bus controller during the execution of a *Data In C* (DIC) instruction. Should cause the device controller selected by ^DS<0-5> to place the contents of its C input buffer on ^DATA<0-15>.
DATOA	*Data Out A*. This signal line is asserted by the I/O bus controller during the execution of a *Data Out A* (DOA) instruction, after the I/O bus controller has placed the contents of the specified CPU accumulator on ^DATA<0-15>. Should cause the device controller selected by ^DS<0-5> to load its A output buffer with the data on ^DATA<0-15>.
DATOB	*Data Out B*. This signal line is asserted by the I/O bus controller during the execution of a *Data Out B* (DOB) instruction, after the I/O bus controller has placed the contents of the specified CPU accumulator on ^DATA<0-15>. Should cause the device controller selected by ^DS<0-5> to load its B output buffer with the data on ^DATA<0-15>.
DATOC	*Data Out C*. This signal line is asserted by the I/O bus controller during the execution of a *Data Out C* (DOC) instruction, after the I/O bus controller has placed the contents of the specified CPU accumulator on ^DATA<0-15>. Should cause the device controller selected by ^DS<0-5> to load its C output buffer with the data on ^DATA<0-15>.
STRT	*Start*. This signal line is asserted by the I/O bus controller during the execution of any I/O instruction (except an *I/O Skip* instruction) in which bits 8 and 9 = 01 (that is, any I/O instruction in which the *Start (S)* flag control function is specified). Asserted after the data transfer has occurred during DIA, DIB, DIC, DOA, DOB, DOC instructions. Usually used to initiate peripheral operation by setting the Busy flag to 1 and the Done flag to 0.
CLR	*Clear*. This signal line is asserted by the I/O bus controller during the execution of any I/O instruction (except an *I/O Skip* instruction) in which bits 8 and 9 = 10 (that is, any I/O instruction in which the *Clear (C)* flag control function is specified). Asserted after the data transfer has occurred during DIA, DIB, DIC, DOA, DOB, DOC instructions. Usually used to terminate peripheral operation by setting the Busy and Done flags to 0.

(continued)

Table 2-1 I/O Bus Signals

Signal	Description
IOPLS	*I/O Pulse*. This signal line is asserted by the I/O bus controller during the execution of any I/O instruction (except an *I/O Skip* instruction) in which bits 8 and 9 = 11 (that is, any I/O instruction in which the *Pulse (P)* flag control function is specified). Asserted after the data transfer has occurred during **DIA**, **DIB**, **DIC**, **DOA**, **DOB**, **DOC** instructions. Usually used to initiate special peripheral operations.
^SELB	*Selected Busy*. This signal line is conditioned by the device controller, selected by the device select lines, to specify the state of the controller's Busy flag. Asserted low if the controller's Busy flag is set to 1.
^SELD	*Selected Done*. This signal line conditioned by the device controller, selected by the device select lines, to specify the state of the controller's Busy flag. Asserted low if the controller's Done flag is set to 1.
Program Interrupt	
^INTR	*Interrupt Request*. This signal line is asserted low by a device controller to request program interrupt service.
^MSKO	*Mask Out*. This signal line is asserted low by the I/O bus controller during the execution of the *Mask Out* (**MSKO**) instruction, after the contents of the designated CPU accumulator have been placed on **^DATA<0-15>**. Used to load the contents of **^DATA<0-15>** into the interrupt disable flip-flops of all device controllers using the program interrupt system.
^INTP	*Interrupt Priority*. This signal line is always asserted true to the first device controller, on the I/O bus, using the program interrupt facility, and transmitted in series through each successive device controller. A device controller should not issue an asserted **^INTPOUT** unless it is receiving an asserted **^INTPIN** and is not requesting interrupt service.
INTA	*Interrupt Acknowledge*. This signal line is asserted by the I/O bus controller during the execution of the *Interrupt Acknowledge* (**INTA**) instruction. If a device controller receives **INTA** while it is also receiving an asserted **^INTPIN** and while it is requesting interrupt service, it should place its device code on **^DATA<10-15>**. *Note that multiple* **INTA**s can be received by a device before its interrupt is processed. Note that the I/O bus controller for I/O bus 0 will not assert **INTA** onto its I/O bus during the execution of the **INTA** instruction if one of the devices resident on the MCU/IOC/DRP board (primary asynchronous interface, real-time clock, programmable interval timer, and memory control unit and/or input/output channel error) requires interrupt service.
Data Channel	
^DCHR	*Data Channel Request*. This signal line is asserted low by a device controller when it requires data channel service.
^DCHP	*Data Channel Priority*. This signal line is always asserted true to the first data channel device controller, on the I/O bus, and transmitted in series through each device controller. A device controller should not issue an asserted **^DCHPOUT** unless it is receiving an asserted **^DCHPIN** and it is not requesting data channel service.
^DCHA	*Data Channel Acknowledge*. This signal line is asserted low by the I/O bus controller at the beginning of each data channel cycle. Should cause the device controller that is receiving an asserted **^DCHPIN** signal and whose DCH REQ flip-flop is set. 1. The memory address (physical or logical) to be used for this transfer on signal lines **^DATA<2-15>** of the I/O bus. 2. The additional DCH map slot address bits to be used for this transfer on signal lines **^DS<4-5>**, **^EXTDCH**, and **^DATA0** of the I/O bus.

(continued)

Table 2-1 I/O Bus Signals

Signal	Description
^DCHA (continued)	3. And the type of transfer on signal lines ^DCHM0, and ^DS0 of the I/O bus. Note that not all data channel device controllers use the additional map slot address bits and that some controllers do not use all of the additional map slot address bits (see "Data Channel Map Slot Selection" later in this chapter). Note also that not all data channel device controllers use ^DS0 to specify the type of transfer. (See "Data Channel Transfer Modes" later in this chapter.)
^EXTDCH	*Extended Data Channel.* This signal line is conditioned by the device controller whose DCH SEL flip-flop is set for memory mapping address selection. (See the information on signals ^DCHA and ^DS<4-5>.) Note that not all data channel device controllers use ^EXTDCH as a map slot address bit. (See "Data Channel Map Slot Selection" later in this chapter.)
^DCHM0	*Data Channel Mode.* This signal line is conditioned by the device controller whose DCH SEL flip-flop is set to inform the I/O bus controller of the data channel transfer type to be performed. Used in conjunction with ^DS0 by some device controllers. (See the information on signal ^DS0.) (Also see description of data channel modes.)
DCHI	*Data Channel Input.* This signal line is asserted by the I/O bus controller for data channel input (^DS0 = 0, ^DCHMO = 1) and for the write portion of a read-modify-write transfer (^DS0 and ^DCHMO both = 1). Should cause the device controller whose DCH SEL flip-flop is set to place the contents of its input register on ^DATA<0-15>.
DCHO	*Data Channel Output.* This signal line is asserted by the I/O bus controller for data channel output (^DS0 and ^DCHMO = 0) and for the read portion of a read-modify-write transfer (^DS0 and ^DCHMO both = 1), after the data word has been placed on ^DATA<0-15>. Should cause the priority-selected device controller to load the data from ^DATA<0-15>.
System Control	
IORST	*I/O Reset.* This signal line is asserted by the I/O bus controller during the *I/O Reset* (IORST) instruction or when a Reset function occurs. IORST is also issued prior to central processing unit operation at powerup. This signal should be used to initialize the machine state of all device controllers in the system.
^RQENB	*Request Enable.* This signal line is asserted low by the I/O bus controller to synchronize program interrupt and data channel requests from all device controllers. In any device controller, ^INTR and ^DCHR should be clocked only on the transistion from high to low level of ^RQENB. I/O bus controllers do not assert ^RQENB while an *Interrupt Acknowledge* INTA instruction is being executed (see "Interrupt Request" later in this chapter).

(concluded)

Programmed I/O Protocol

Programmed input/output transfers are controlled by a unique signal on the I/O bus. When selected, device controllers must respond to programmed input/output transfer signals as described in the following sections.

Device Selection

Every programmed input/output instruction includes a 6-bit device code that uniquely references the device controller involved in the transfer. During input/output instruction execution, the device select lines (^**DSO** through ^**DS5**) carry the contents of the low- order 6 bits of the instruction, which is the device code. The device controller must not assert the ^**DATA<0-15>** lines of the bus or initiate any other function as a result solely of the device select lines. Rather, the selected device controller should respond only to the assertion of control signals on the I/O bus when it is selected.

A device controller can decode the device select lines in several ways. The device select lines corresponding to the bit positions containing a 1 in the device controller's device code could be inverted and an And function performed on the resultant 6 lines, as shown in Figure 2-2 for device code 13_8.

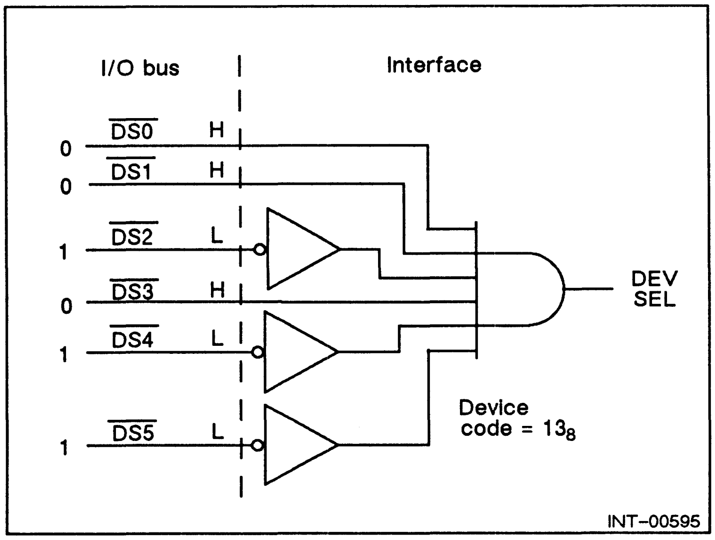


Figure 2-2 Single Device Code Selection Network

Similarly, the lines corresponding to 0s could be inverted and the six lines applied to a NOR gate. There are other possibilities; but the two important details are that the lines are asserted low and that ^**DSO** is the high-order bit of the device code. Note that it is possible to have the device controller respond to more than one device code by decoding fewer than six lines. The remaining lines might be clocked into a register to select a particular device controller mode as shown in Figure 2-3.

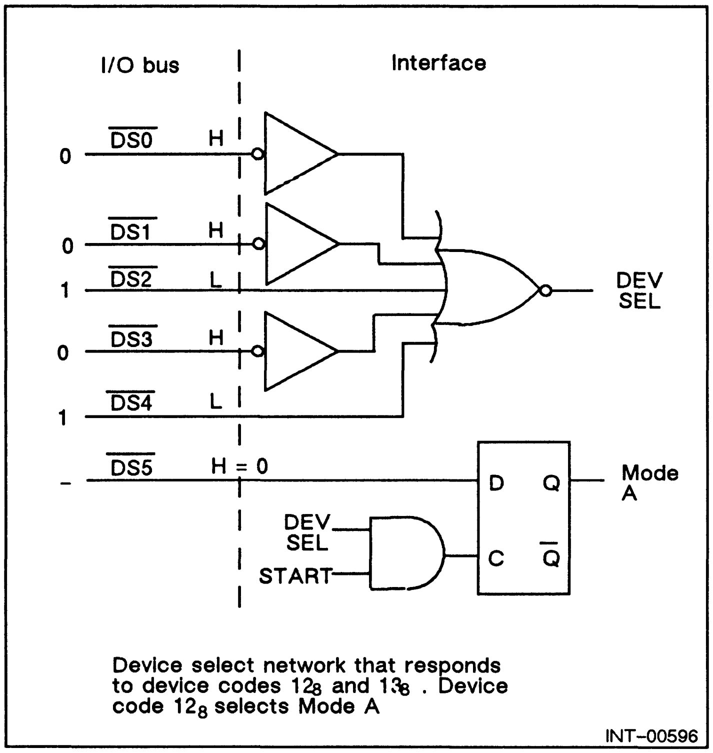


Figure 2-3 Multiple Device Code Selection Network

Assigning Device Codes

There are a number of factors to be considered when assigning a device code to a device controller. A 6-bit device code allows 64 possible codes, 1 to 77_8. Device code 77_8 is assigned to the central processing unit to implement such special functions as Mask Out and Interrupt Acknowledge; hence it cannot be assigned to a device controller. In addition, several device codes are assigned to resident devices and a number of programming considerations restrict the use of device codes as assigned by Data General and used in all Data General software. (Standard ECLIPSE device codes can be found in the *ECLIPSE MV/Family (32-Bit) Systems Principles of Operation* manual as well as the *Principles of Operation Supplement* for your specific system). When you assign a device code to a custom device controller, it is important to consider these reserved device codes and any other devices that are currently in the system or may be installed in the future.

NOTE: Reserved or assigned device codes can be used as long as those codes are not already in use for devices attached to your system.

Data Transfer Signals

Six data transfer signals of the I/O bus move a word or part of a word under direct program control from a device controller to central processing unit accumulators and vice versa.

Data Input

The three programmed data input instructions — *Data in A, B, and C* (**DIA**, **DIB**, and **DIC**) — allow data to be transferred from up to three distinct sources per device code in the device controller and loaded into any one of the four CPU accummulators. Additionally, if specified in the instruction, one of the following flag control signals will be issued by the I/O bus controller to control the status flags or other device controller functions: *Start*, *Clear*, orI/O *IO Pulse*.

The execution of a data input instruction consists of two parts. The first is the data transfer, followed by the (optional) flag control pulse. Three signals are used for the data input transfer, one for each of the three possible sources in the device controller. Through the use of three separate signals, the need for a decoding network in the device controller is avoided.

During the first half of data input instruction execution, one of the three transfer control signals — **DIA**, **DIB**, or **DIC** — is asserted on the I/O bus by the I/O bus controller, as shown in Figure 2-4. This signal should be used by the addressed device controller to cause data from the proper sources to be asserted on the **^DATA<0-15>** lines of the I/O bus. After a time delay, the I/O bus controller will transfer the information on the **^DATA<0-15>** lines into the CPU accumulator specified by the instruction and remove the transfer control signal.

NOTE: Because all device controllers are wired to the I/O bus in parallel, it is extremely important that only the device controller referenced by the device select code assert any of the **^DATA<0-15>** lines — and then only in response to the transfer control signal.

During the second half of the instruction execution, the appropriate flag control pulse — *Start*, *Clear*, or *IO Pulse* — is asserted. These pulses are specified in the flag control (*f*) field of the instruction. The response of the device controller to each of these signals will be discussed later.

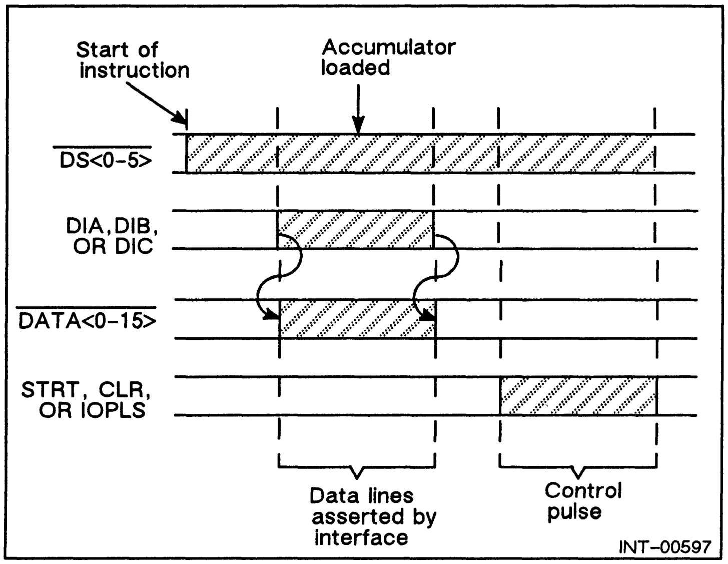


Figure 2-4 Data In Sequence

Data Output

The three programmed data output instructions — *Data Out A, B, and C* (**DOA**, **DOB**, and **DOC**) — transfer data from any one of the four CPU accumulators through the I/O bus controller to one of up to three destinations per device code in the device controller. Additionally, if specified in the instruction, a flag control signal — *Start*, *Clear*, or *IO Pulse* — will be issued by the I/O bus controller to control the status flags or other device controller functions.

The execution of a data output instruction consists of two parts: the data transfer, followed by the (optional) control pulse. Three signals are used for the data output transfer, one for each of the three possible data destinations in the device controller. Through the use of three separate signals, the need for a decoding network in the device controller is avoided. Note that the three destinations used for data output needn't have any relation to the three data sources used for data input.

As shown in Figure 2–5, the I/O bus controller asserts the ^**DATA<0–15>** lines with the contents of the CPU accumulator specified in the instruction. After allowing time for the data to propagate down the I/O bus and for the lines to settle, the I/O bus controller asserts one of the three transfer control signals — **DOA**, **DOB**, or **DOC**. This signal should be used by the selected device controller to gate the contents of the ^**DATA<0–15>** lines to the proper destination and to load a register. After removing the transfer control signal, the I/O bus controller will remove the data from the ^**DATA<0–15>** lines.

NOTE: Because all device controllers are wired to the I/O bus in parallel, it is extremely important that no device, including that referenced by the device select code, be allowed to assert any of the ^**DATA<0–15>** lines during the data output instruction.

During the second half of the instruction execution, the appropriate flag control pulse — *Start*, *Clear*, or *IO Pulse* — is asserted. These pulses are specified in the flag control (*f*) field of the instruction. The response of the device controller to each of these signals will be discussed later.

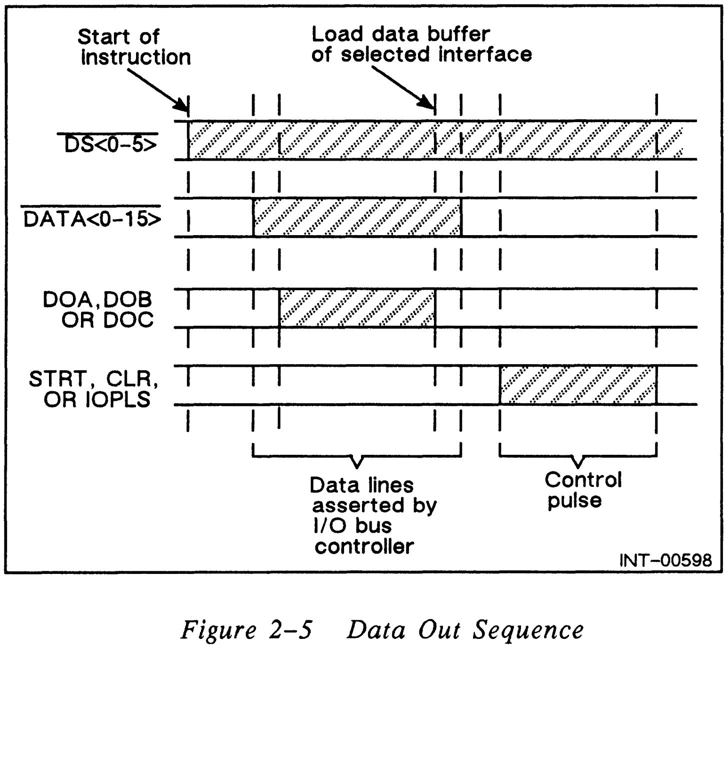


Figure 2–5 Data Out Sequence

I/O Skip

The operation of most peripherals is not synchronous to the operation of the CPU. Because of its faster processing rate, the CPU will generally have to wait for the completion of a peripheral's operation. It is usually important for the CPU not to issue any new instructions to the peripheral until the peripheral has completed its previous operation. This asynchronous relationship between the CPU and peripheral requires that the CPU be able to test the status of the peripheral.

The *I/O Skip* (**IOSKP**) instruction allows the program to test the state of two I/O bus lines: **^SELB** and **^SELD**. Whenever a device controller recognizes its device code on the device select lines, as discussed above, it should assert the **^SELB** and/or **^SELD** lines depending on the internal state of the device controller. (Ordinarily, **^SELB** will be asserted if the Busy flag is set, and **^SELD** will be asserted if the Done flag is set.) During the execution of the *I/O Skip* instruction, the I/O bus controller passes the content of these two lines to the CPU for testing. The CPU checks the state of the appropriate line and skips the next sequential instruction if the line matches the condition specified in the instruction.

The test condition is specified in the test *t* field of the instruction as shown in Table 2-2.

Table 2-2 Test Conditions

Test Field	Mnemonic	Next Instruction Skipped If
00	BN	^SELB = L
01	BZ	^SELB = H
10	DN	^SELD = L
11	DZ	^SELD = H

Start, Clear, I/O Pulse

During the second half of the instruction execution, the appropriate flag control pulse — *Start*, *Clear*, or *IO Pulse* — is asserted. These pulses are specified in the flag control (*f*) field of the instruction. The response of the device controller to each of these signals will be discussed later. Though a convention is followed in using these signals in Data General device controllers, as explained below, the designer should realize that the signals may be used for virtually any purpose.

Busy/Done Network

In Data General device controllers, two flags — Busy and Done — carry elementary status information needed by the program. Any time a device controller detects its device code on the device select lines, it asserts the **^SELB** line if its Busy flag is set and/or **^SELD** if its Done flag is set. During the execution of an *I/O Skip* (**IOSKP**) instruction, the I/O bus controller passes the content of these two lines to the CPU for testing. The CPU checks the state of the line as specified by the test control (*t*) field of the instruction and skips the next sequential instruction if the line matches the condition specified in the instruction.

Although the significance of the flags may vary somewhat depending on the particular device controller in question, they do carry a similar meaning in many cases. The Busy flag generally indicates that the device is currently processing data or waiting for some response from an external system. When this flag is set, any interference from the program, such as an attempt to transfer information to this device, may produce unpredictable results. The Done flag indicates that the device has completed an operation and is awaiting a response by the program. In many devices, it is important that this response come within a maximum time period to prevent a degradation of system performance. See Figure 2–6 for Busy/Done sequencing.

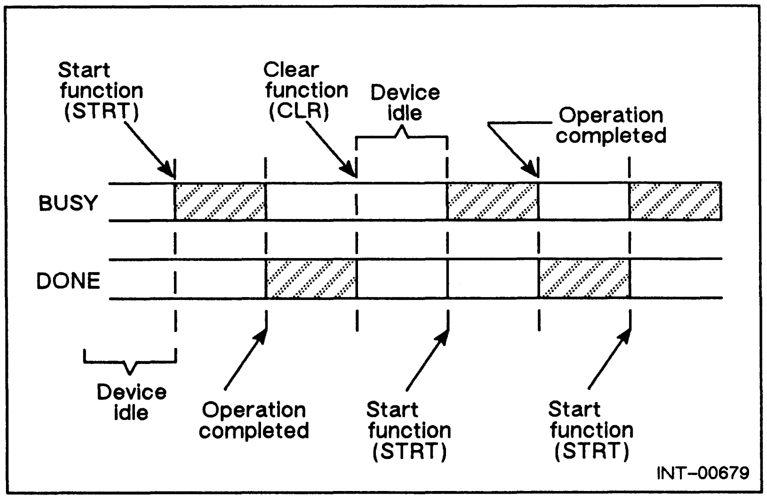


Figure 2–6 Busy/Done Sequencing

In addition to conveying status information to the program, these flags can be used as switches by the program to control the device controller. Generally, the *Start* pulse causes the Busy flag to be set and the Done flag to be cleared, initiating device controller operation. At the completion of the operation, a signal originating in the device controller sets Done and clears Busy. If at any time the *Clear* pulse is issued by the I/O bus controller, both flags should be cleared.

Figure 2–7 shows one implementation of the Busy/Done network. The **IORST** signal clears both the Busy and Done flags directly. Signals generated by the flag control function part of an I/O instruction affect the flags only if the device has recognized its device code on the device select lines. When the device completes its operation, it generates a completion signal, DEV COMPLETE, that clears the Busy flag and sets the Done flag. The signal need not act on both flags directly; it can just as well clear the Busy flag, whose state change sets the Done flag. Note that in the configuration shown here, the *D* input to the Done flag is the output of the Busy flag. Therefore, the completion signal will not set the Done flag if the program has previously cleared the Busy flag.

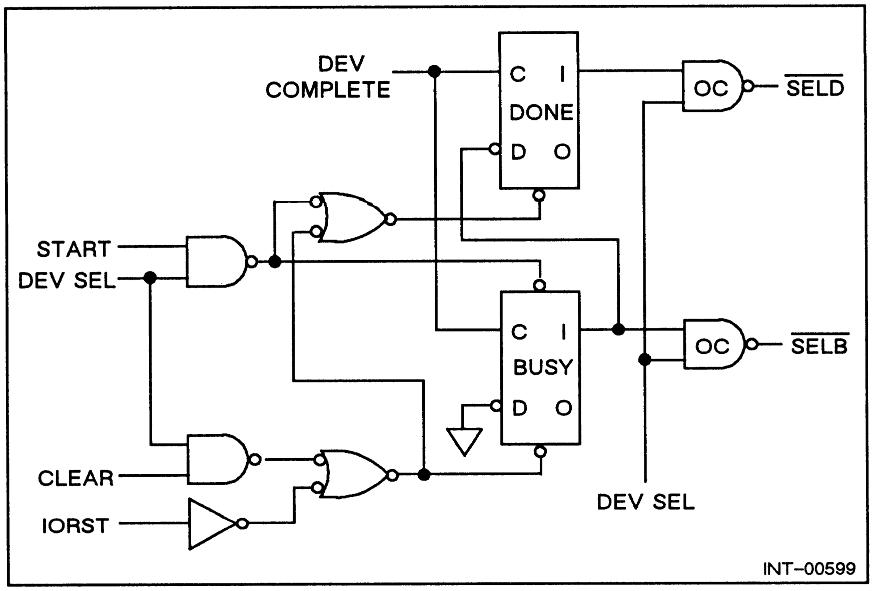


Figure 2–7 Typical Busy/Done Network

Program Interrupt System

The program interrupt system provides a device controller with a convenient means of notifying the central processing unit that the controller requires service by the program.

Interrupt Request

A device controller issues a program interrupt request by asserting the ^**INTR** line of the I/O bus. The CPU checks this line at the end of every instruction and at specific points during the execution of lengthy instructions. If the interrupt line is asserted (and the CPU's Interrupt On *ION* is 1), it executes the program interrupt function. The interrupt in no way affects the device controller itself; any action to actually service the device must be the result of the software interrupt handler.

The signal, ^**RQENB**, generated by the I/O bus controller is used to clock the interrupt requests. The usual convention is to use the transistion of ^**RQENB** from a high to low level to clock the interrupt request (INTR) flip–flop when the peripheral has set its Done flag and the device controller does not have its interrupts disabled. The INTR flip–flop in turn drives the ^**INTR** line. It is very important that INTR be asserted and removed synchronous with the transistion from high to low level of ^**RQENB**.

I/O bus controllers do not assert ^**RQENB** while an *Interrupt Acknowledge* (**INTA**) instruction is being executed. This ensures that no higher priority device sets its interrupt and applies its device code to the I/O bus during the **INTA** signal. It could be possible that the time might not be long enough to ensure that the device code is properly received by the I/O bus controller.

Interrupt Disable

In addition to the three data output transfer control signals, there is a signal, ^**MSKO**, that functions in much the same way. This signal allows a 16–level priority system to be established for the program interrupt facility. The signal, ^**MSKO** is issued during the data transfer portion of a *Data Out B* (**DOB**) instruction when a device code of 77_8 is specified (**CPU**).

Conventionally, each device controller using the interrupt system is assigned a hardware priority level corresponding to 1 of the 16 bits of a data word. When the ^**MSKO** signal is received by the device controller (regardless of the device code), an Interrupt Disable flag (INT DISABLE) should be loaded from the ^**DATA<0–15>** signal corresponding to the priority assignment of that device controller, as shown in Figure 2–8. Whenever this INT DISABLE flag is set, the device controller should be inhibited from issuing an interrupt request. Additionally, if the device controller is issuing an interrupt request when the INT DISABLE flag is set, the request should be removed on the next transistion from high to low level of ^**RQENB**.

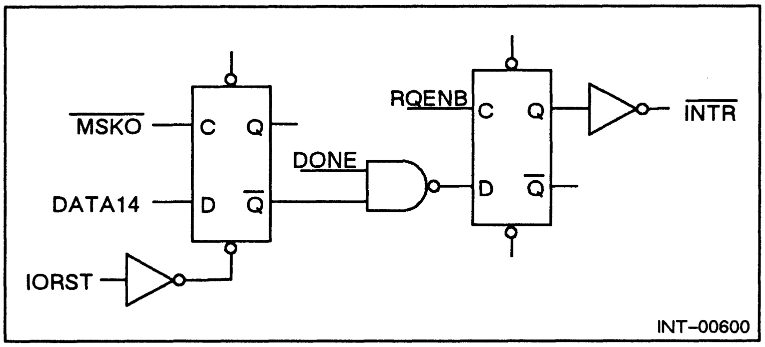


Figure 2–8 Typical Priority Mask Bit Circuit

Interrupt Priority

There is an elementary hardware priority assignment on the I/O bus that arbitrates between two or. more device controllers requesting interrupt service at the same time. In the event of simultaneous interrupt requests, this priority system grants service to the device controller that is requesting service and is closest to the I/O bus controller on the I/O bus.

The interrupt priority signal should be passed through a priority network in every device controller that uses the program interrupt facility. This signal, which must be passed undisturbed by other device controllers (and any printed circuit boards installed in device controller slots of the chassis), is called ^**INTPIN** as it enters each device controller and ^**INTPOUT** as it leaves. For each I/O bus implemented, ^**INTPIN** starts on the chassis' backpanel at the first I/O printed circuit board slot of the I/O bus(es) as a low (asserted) signal, but it is pulled high to succeeding device controllers by any device controller that requests interrupt service. Any device controller that receives a high ^**INTPIN** signal should pass a high ^**INTPOUT** to the device controllers further along the bus.

The circuit in Figure 2–9 shows the suggested implementation of this priority network. In many cases, more than one device controller using the program interrupt facility will be built on a single board. Here, each device controller built on the board will require its own priority network. As many elements as needed, each similar to that shown below, will be chained together, with the ^**INTPOUT** signal of one feeding ^**INTPIN** of the next. Note that the terminating resistors shown are used on the signals that enter and leave the board.

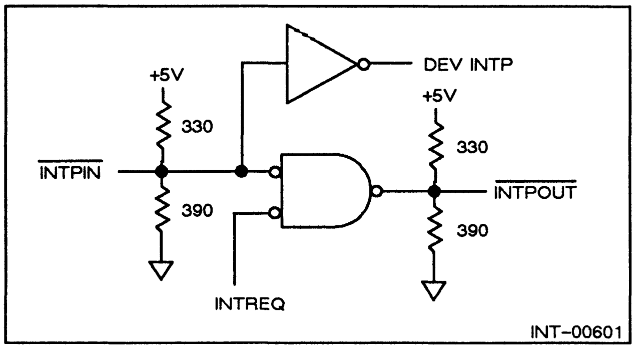


Figure 2-9 Typical Interrupt Priority Chain

Timing on this interrupt priority chain can be critical and becomes especially so for large systems with many device controllers. Often it is possible to build several device controllers on a single board. In these cases, the propagation time can be significantly reduced by replacing the priority chain on such a board with two separate chains. One chain, consisting of a single network element, determines the priority of the entire board and quickly passes the ^**INTP** signal on to the next board. A separate chain determines the priority of the various device controllers on the board. See Figure 2-10.

Figure 2-11 shows a typical interrupt control circuit as implemented in each device controller.

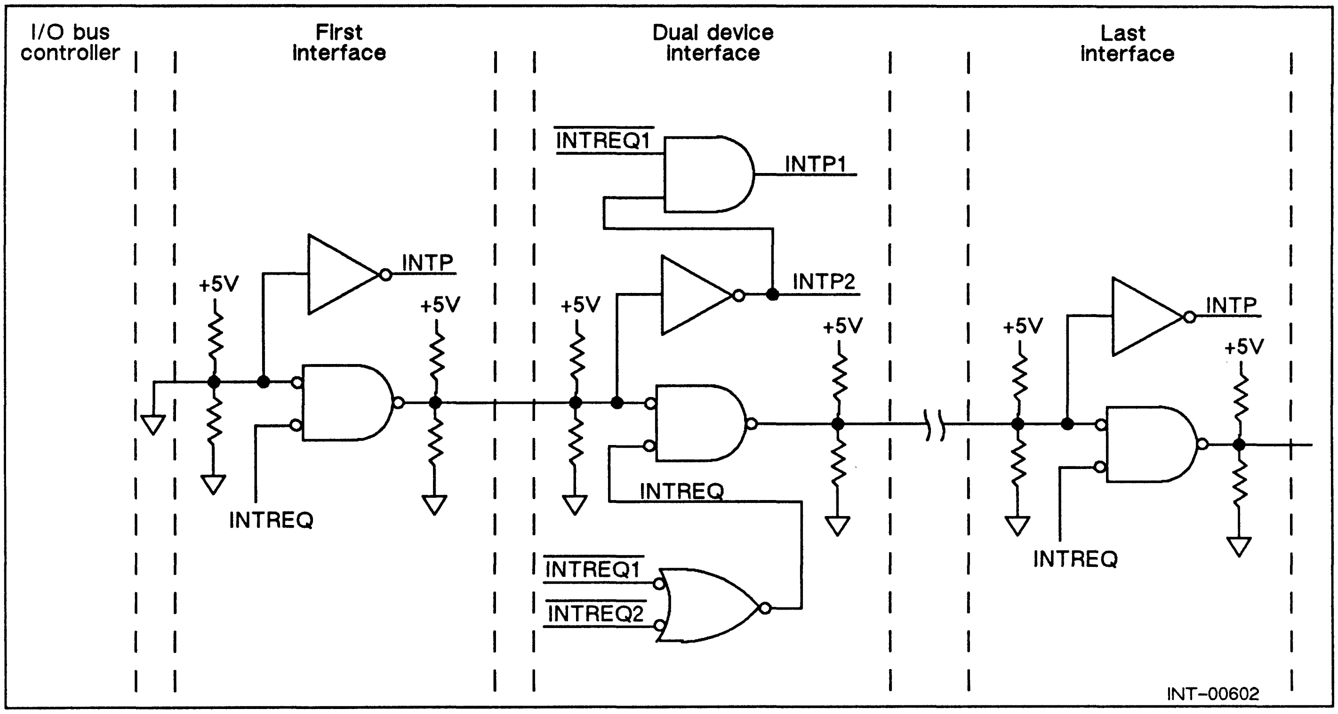


Figure 2-10 Interrupt Priority Chain

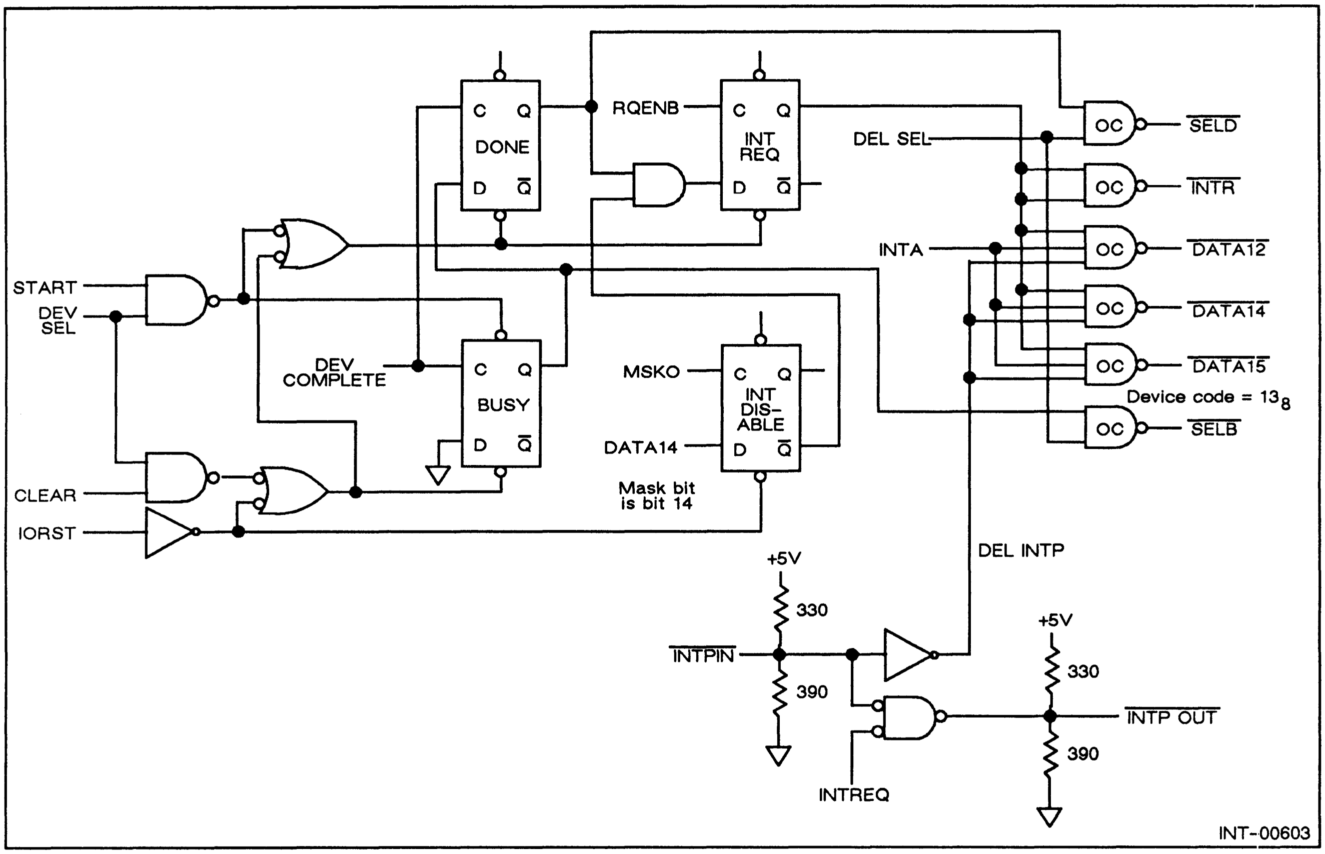


Figure 2–11 Typical Interrupt Control Circuit

Interrupt Acknowledge

Once the central processing unit has received an interrupt request and transferred control to the interrupt service routine, the software must service the device controller that caused the interrupt. Before the program can even attempt to service the device controller, it must determine which device controller did, in fact, cause the interrupt. The simplest way that this can be achieved is by the use of the *Interrupt Acknowledge* (**INTA**) instruction. This instruction is equivalent to a *Data Input B* (**DIB**) instruction with a device code of 77_8 (**CPU**). This instruction executes as a data input transfer instruction, but the I/O bus controller asserts the **INTA** signal on the I/O bus during the data transfer portion of the instruction.

Note that the I/O bus controller for I/O bus 0 will not assert **INTA** onto its I/O bus during the execution of the *Interrupt Acknowledge* (**INTA**) instruction if one of the devices resident on the MCU/IOC/DRP (MID) board requires interrupt service. Devices that are resident on the MID board are the primary asynchronous interface, the real–time clock, the programmable interval timer, the memory control unit, and input/output channel error.

When the **INTA** signal is received by a device controller (regardless of device code), the condition of the **^INTPIN** line to that device controller should be checked. If this line is asserted and the device controller is currently issuing an interrupt request, it should assert its device code on **^DATA<10–15>** of the I/O bus for the duration of the **INTA** signal. At the end of the **INTA** period, the I/O bus controller transfers the

I/O channel code and device code to the CPU. The CPU then loads the I/O channel code and device code into the specified CPU accumulator.

Data Channel

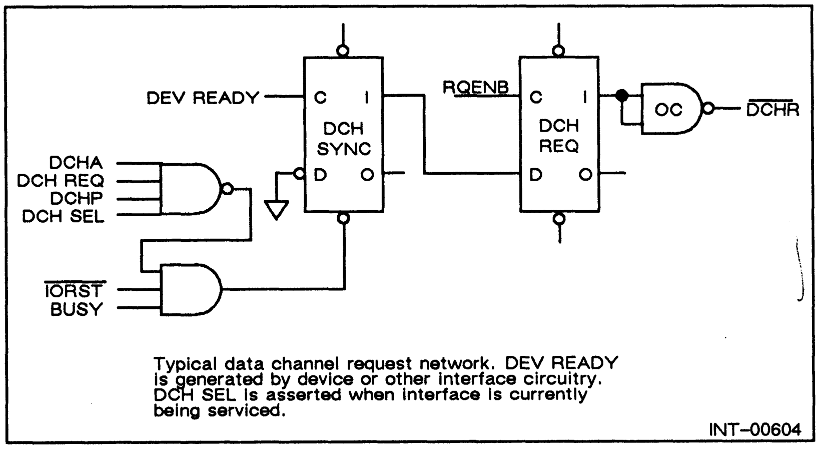
Unlike programmed input/output transfers, which are each controlled by one unique signal on the I/O bus, data channel transfers are somewhat more complex. The interaction between the I/O bus controller and the device controller is much more involved, because of the number of functions performed for each transfer. Unlike the design of a programmed I/O device controller, where certain liberties can be taken in the designed response to a bus signal, the design of a data channel device controller is relatively restricted, and we recommend that such a design correspond to the following description.

Data Channel Request

A device controller issues a data channel request by asserting the ^**DCHR** line of the I/O bus. Unlike interrupts, the I/O bus controller services a data channel request completely without software intervention.

The signal, ^**RQENB**, generated by the I/O bus controller is used to clock the data channel requests in the same manner as the interrupt requests. The usual convention is to use the transistion from high to low level of ^**RQENB** to clock the state of a flip–flop, controlled by the device controller (DCH SYNC in Figure 2–12), into a data channel request (DCH REQ) flip–flop, which in turn drives the ^**DCHR** line.

NOTE: It is very important that ^**DCHR** be clocked only on the transistion from high to low level of ^**RQENB**.



Typical data channel request network. DEV READY is generated by device or other interface circuitry. DCH SEL is asserted when interface is currently being serviced.

Figure 2–12 Typical Data Channel Request Circuit

When the I/O bus controller sees ^**DCHR** asserted, it transfers data to or from the highest priority controller requesting service. Just before the end of every data channel transfer, the I/O bus controller pulses the ^**RQENB** signal again. And if the ^**DCHR** signal is still asserted at the end of the current transfer, the I/O bus controller performs data channel transfer to the highest priority controller that is still requesting data channel service. The I/O bus controller continues to perform data channel transfers in this way until no controller on the I/O bus is requesting data channel service. Figure 2–13 shows the timing sequence required for data channel transfers.

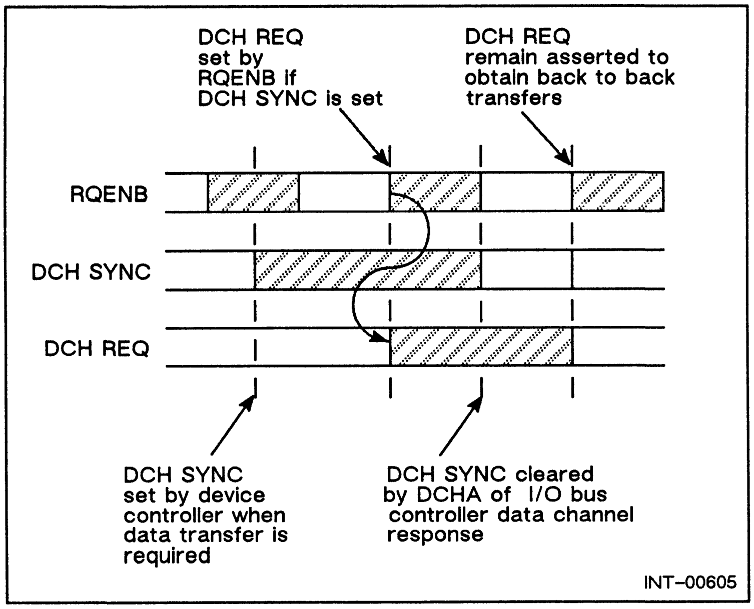

DCH REQ set by RQENB if DCH SYNC is set
DCH REQ remain asserted to obtain back to back transfers
RQENB
DCH SYNC
DCH REQ
DCH SYNC set by device controller when data transfer is required
DCH SYNC cleared by DCHA of I/O bus controller data channel response
INT-00605

Figure 2–13 Typical Data Channel Sequence

Data Channel Priority

There is an elementary hardware priority assignment system on the I/O bus that arbitrates between two or more device controllers requesting data channel service at the same time. In the event of simultaneous data channel requests, this priority system grants service to the device controller that is requesting service and is closest to the I/O bus controller on the I/O bus.

The data channel priority signal should be passed through a priority network in every device controller that uses the data channel. This signal, which must be passed undisturbed by other device controllers (and any printed circuit boards installed in device controller slots of the chassis), is called ^**DCHPIN** as it enters each device controller and ^**DCHPOUT** as it leaves. For each I/O bus implemented, ^**DCHPIN** starts on the chassis backpanel, immediately adjacent to the respective I/O bus controller board(s), as a low (asserted) signal, but it is pulled high to succeeding device controllers by any device controller that requests data channel service. Any device controller that receives a high ^**DCHPIN** signal should pass a high ^**DCHPOUT** to the device controllers further along the bus.

The only device controller that should respond to the I/O bus controller's data channel signals is that which is requesting data channel service and is receiving a low level ^**DCHPIN**; that is, the device controller closest to the I/O bus controller, on the I/O bus, that is requesting data channel service.

The circuit shown in Figure 2–14 shows the suggested implementation of this priority network. In some cases, more than one device controller using the data channel facility will be built on a single board. Here, each device controller will require its own priority network. As many elements as needed, each similar to that shown below, will be chained together, with the ^**DCHPOUT** signal of one feeding ^**DCHPIN** of the next one. Note that the terminating resistors shown are used on the signals that enter or leave the board.

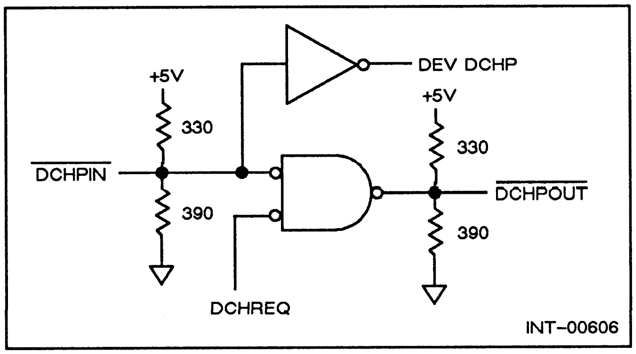


Figure 2-14 Typical Data Channel Priority Circuit

Timing on this data channel priority chain can be critical and becomes especially so for large systems with many device controllers. Often it is possible to build several device controllers on a single board or in a single external chassis. In these cases the propagation time can be significantly reduced by replacing the priority chain on such a board with two separate chains. One chain, consisting of a single network element, determines the priority of the entire board and quickly passes the ^**DCHP** signal on to the next board. A separate chain determines the priority of the various device controllers on the board. (See Figure 2-15).

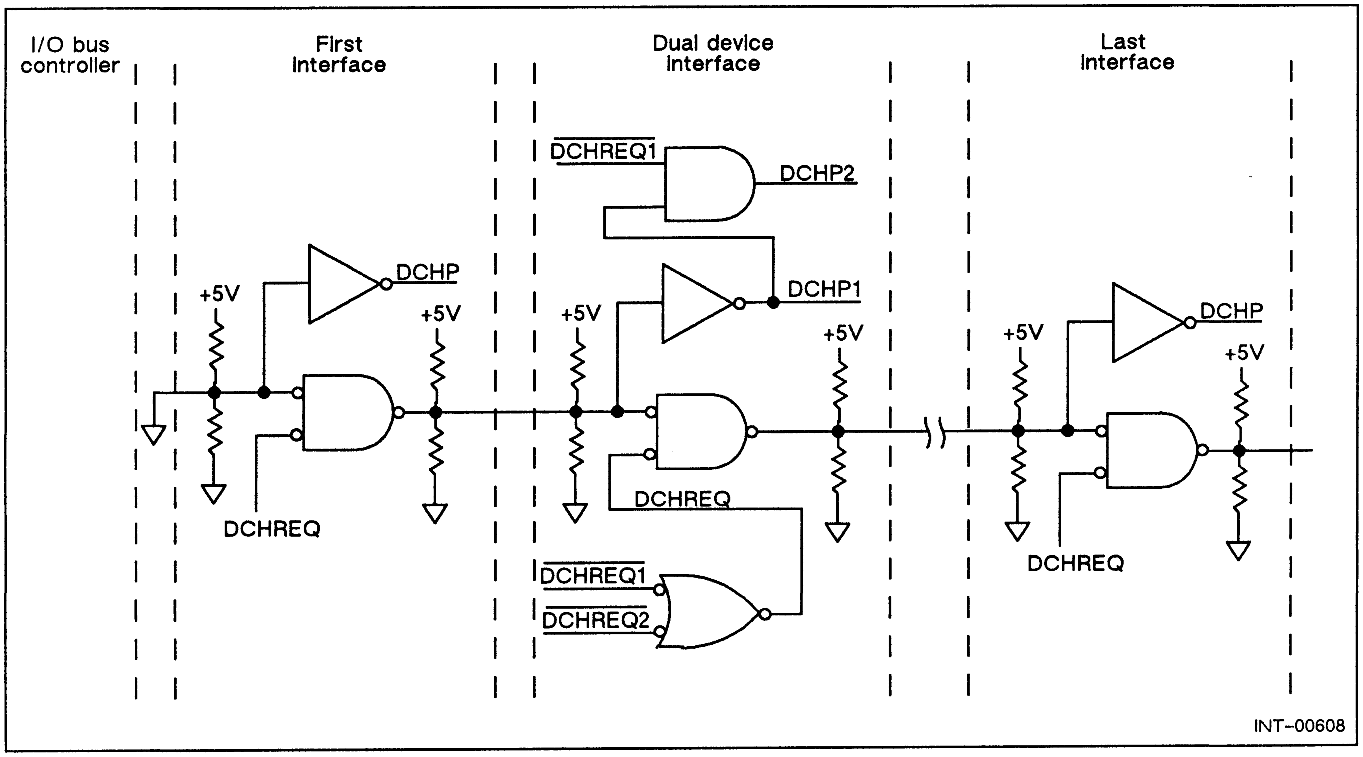


Figure 2-15 Data Channel Priority Chain

Data Channel Acknowledge

As the first step in any data channel transfer the I/O bus controller will issue the data channel acknowledge signal, ^**DCHA**, to all device controllers on the I/O bus. The I/O bus controller expects two types of information from the device controller in response to this signal: the memory address (physical or logical address) and the mode of this transfer. Beyond this, however, the ^**DCHA** signal alerts the device controller to the beginning of a transfer, allowing it to perform some additional functions.

All of the data channel signals are issued on the I/O bus without an accompanying code on the device select lines. The priority network is used to determine which device controller is to respond to these signals. Before one data channel cycle is complete, the data channel request from the device controller being serviced must be cleared to prevent an immediate second transfer. Therefore, a storage unit must be provided to maintain a device controller's selected state even after a change in **DCH REQ** or ^**DCHPIN**.

Figure 2-16 shows the use of a flip-flop, called DCH SEL, which serves this purpose. On the leading edge of ^**DCHA**, this flip-flop will be set if **DCH REQ** is set and the device controller is receiving an asserted ^**DCHPIN**. Otherwise, it will be cleared on this signal. Thus, DCH SEL of the proper device controller will remain set from the beginning of one data channel cycle until the beginning of the next. A device controller should not respond to data channel control signals unless its DCH SEL flag is set.

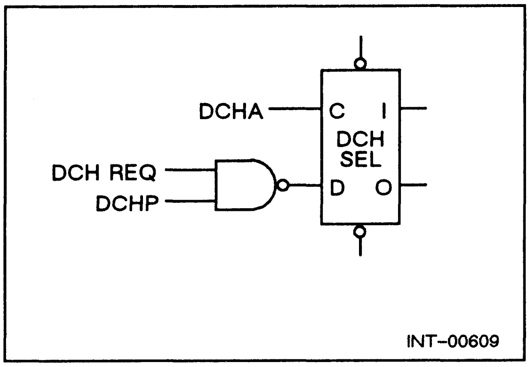


Figure 2-16 Typical Data Channel Select Circuit

While it is receiving ^**DCHA**, the device controller whose DCH SEL flag is set should return the memory address, map slot address (if mapping is to be performed), and transfer mode to the I/O bus controller via the I/O bus. Finally, unless back-to-back transfers are desired, the ^**DCHA** signal should be used to clear the DCH SYNC flag so that **DCH REQ** will be cleared on the next transistion from high to low level of ^**RQENB**. Figure 2-17 shows typical data channel sequencing. Figure 2-18 shows a complete typical data channel control circuit.

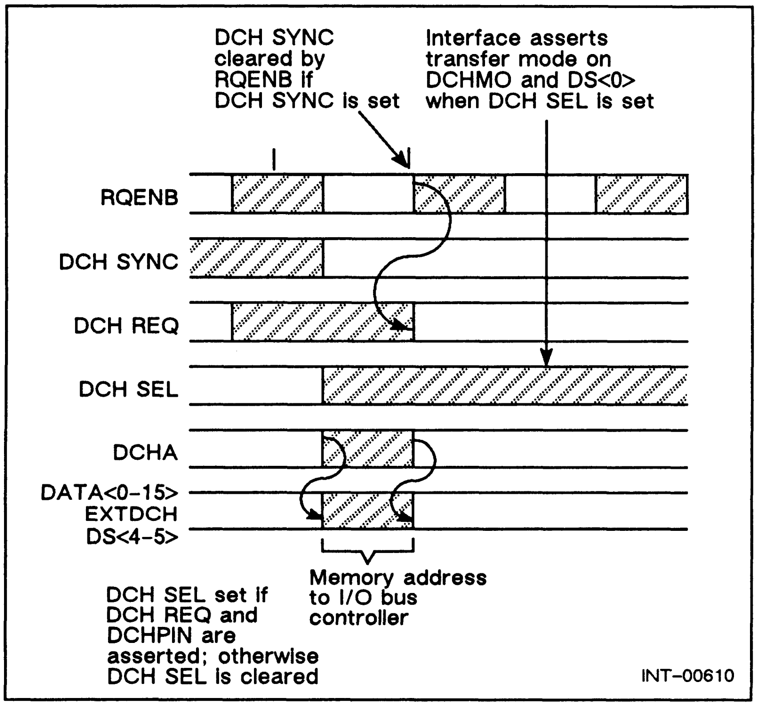

DCH SYNC cleared by RQENB if DCH SYNC is set
Interface asserts transfer mode on DCHMO and DS<0> when DCH SEL is set
RQENB
DCH SYNC
DCH REQ
DCH SEL
DCHA
DATA<0-15>
EXTDCH
DS<4-5>
Memory address to I/O bus controller
DCH SEL set if DCH REQ and DCHPIN are asserted; otherwise DCH SEL is cleared
INT-00610

Figure 2-17 Data Channel Sequencing

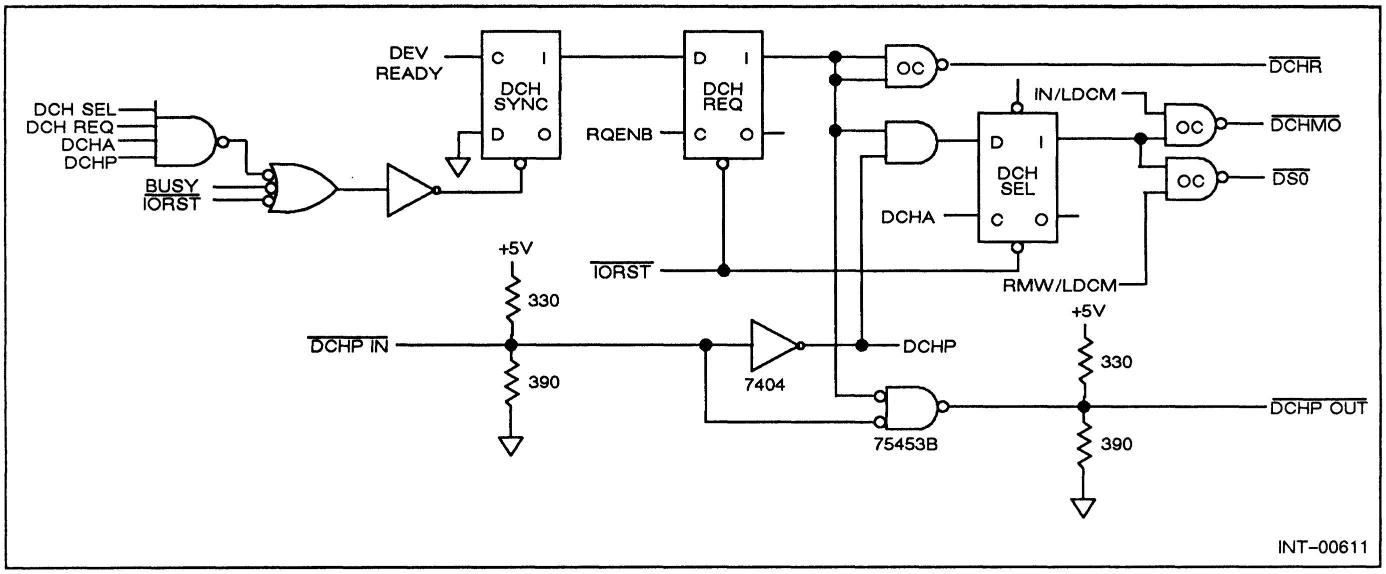


Figure 2-18 Typical Data Channel Control Circuit (Complete)

Data Channel Map Slot Selection

Dependent upon their design, data channel I/O device controllers can assert a 15–, 16–, 12–, or 19–bit memory address as shown in Figure 2–19. When data channel mapping is enabled, this address is handled as a logical memory address that is translated (mapped) to a physical memory address. Data channel mapping is enabled when the CPU program sets the data channel map enable (DME) bit of the I/O Channel Definition register to 1. When this bit is set to 0, all data channel address are unmapped and only the least significant 15–bits of the address provided by the device controller are used to physically address memory. Note that when an ECLIPSE system has multiple input/output channels each channel has its own I/O. Channel Definition register. (For additional information on the I/O Channel Definition register, refer to the *ECLIPSE MV/Family (32–Bit) Systems Principles of Operation* manual as well as the *Principles of Operation Supplement* for your specific system.)

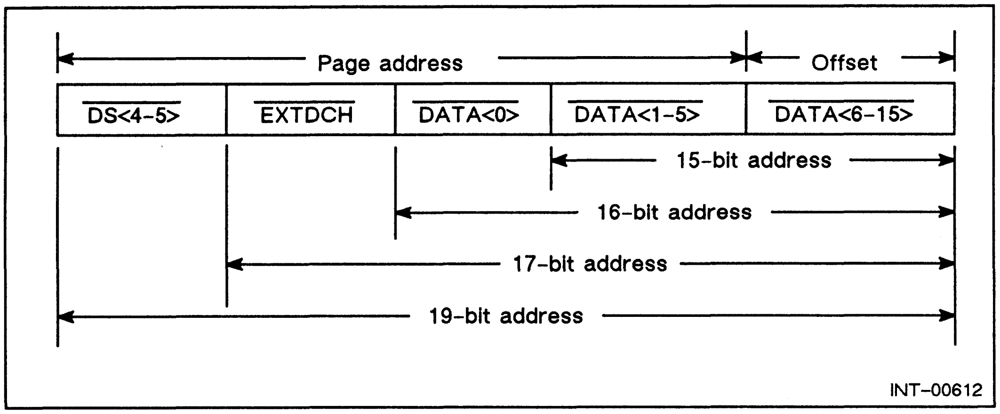

Page address
Offset
$\overline{\text{DS<4-5>}}$
$\overline{\text{EXTDCH}}$
$\overline{\text{DATA<0>}}$
$\overline{\text{DATA<1-5>}}$
$\overline{\text{DATA<6-15>}}$
15-bit address
16-bit address
17-bit address
19-bit address
INT-00612

Figure 2–19 Data Channel I/O Controller Memory Addressing Schemes

When a data channel I/O device controller asserts a 19–bit memory address, it uses the 16–bit I/O data bus. In addition, it uses device select lines **^DS4**, **^DS5**, and the extended data channel line **^EXTDCH** as shown in Figure 2–20. When data channel mapping is enabled, these 19–bits are a logical address that is translated into a physical memory address. In translating the logical address to a physical address, the high–order 9 bits of the logical address form a logical page address that the enabled data channel map translates into a physical page starting memory address of up to 16 bits depending on memory capacity. This page starting memory address then has the 10 low–order bits from the logical address appended to provide the page offset addressing (low–order bits) within the physical page of memory. This combination of physical page starting address and page offset address form the physical address used to access memory.

Data channel device controllers that assert 19–bit memory addresses can select any one of 512 data channel map slots (I/O Channel Definition register numbers 4000_8 through 5777_8).

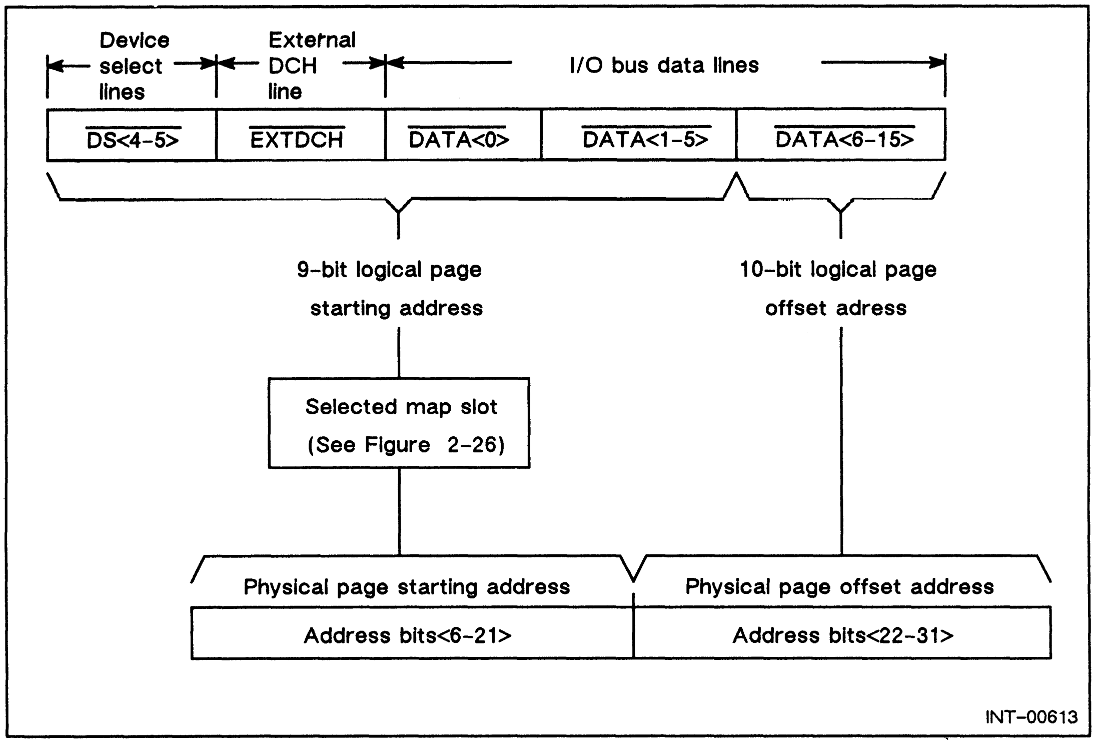

Device select lines
External DCH line
I/O bus data lines
DS<4-5>
EXTDCH
DATA<0>
DATA<1-5>
DATA<6-15>
9-bit logical page starting address
10-bit logical page offset adress
Selected map slot (See Figure 2-26)
Physical page starting address
Physical page offset address
Address bits<6-21>
Address bits<22-31>
INT-00613

Figure 2-20 19-Bit Logical-to-Physical Address Translation

When data channel mapping is enabled, memory addresses from data channel I/O device controllers that assert 15-, 16-, or 12-bit addresses are translated in the same manner except that the high-order 5, 6, or 7 bits of the logical address form a logical page address as shown in Figures 2-21 through 2-23.

Data channel device controllers that assert 15-bit memory addresses can select any one of 32 data channel map slots (I/O Channel Definition register numbers 4000_8 through 4077_8).

Data channel device controllers that assert 16-bit memory addresses can select any one of 64 data channel map slots (I/O Channel Definition register numbers 4000_8 through 4177_8).

Data channel device controllers that assert 12-bit memory addresses can select any one of 128 data channel map slots (I/O Channel Definition register numbers 4000_8 through 4377_8).

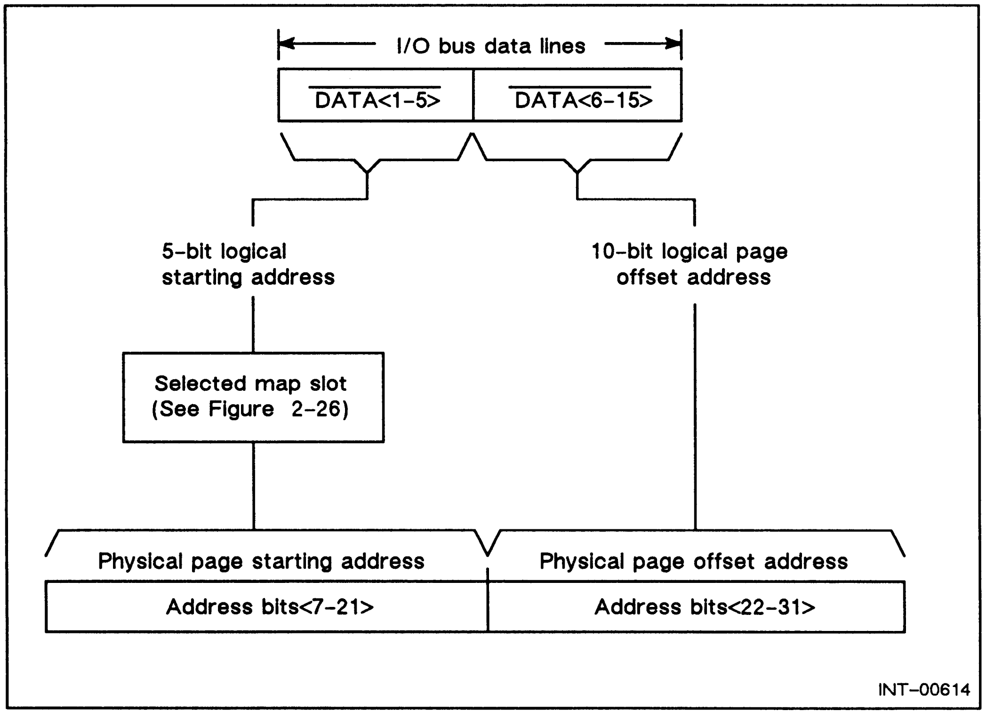

I/O bus data lines
DATA<1-5>
DATA<6-15>
5-bit logical starting address
10-bit logical page offset address
Selected map slot (See Figure 2-26)
Physical page starting address
Physical page offset address
Address bits<7-21>
Address bits<22-31>
INT-00614

Figure 2-21 15-Bit Logical-to-Physical Address Translation

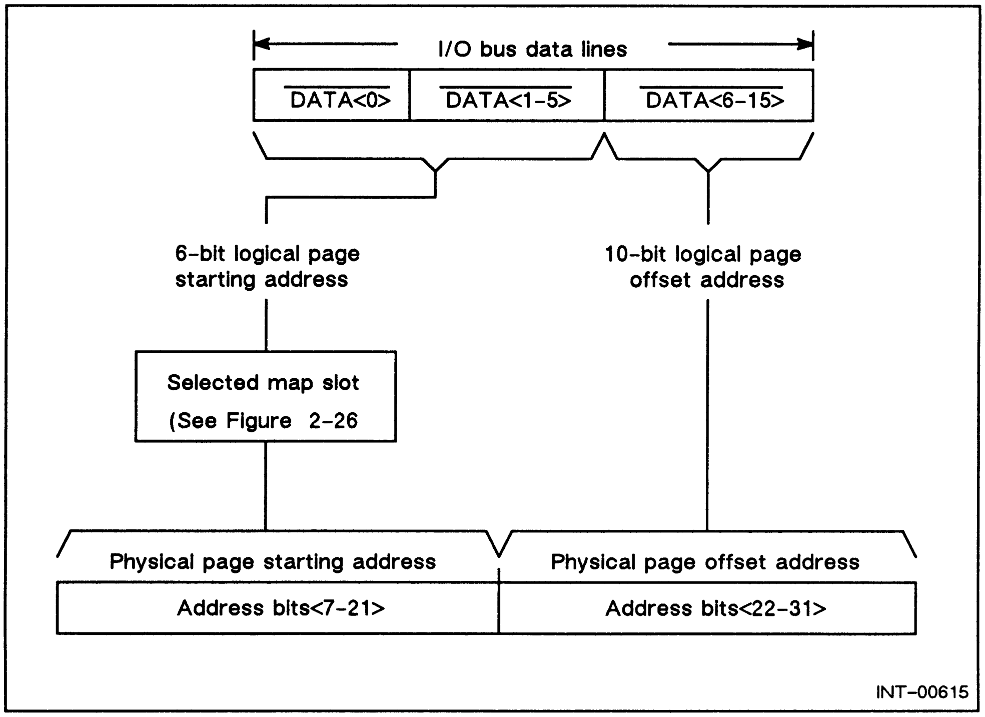

I/O bus data lines
DATA<0>
DATA<1-5>
DATA<6-15>
6-bit logical page starting address
10-bit logical page offset address
Selected map slot (See Figure 2-26
Physical page starting address
Physical page offset address
Address bits<7-21>
Address bits<22-31>
INT-00615

Figure 2-22 16-Bit Logical-to-Physical Address Translation

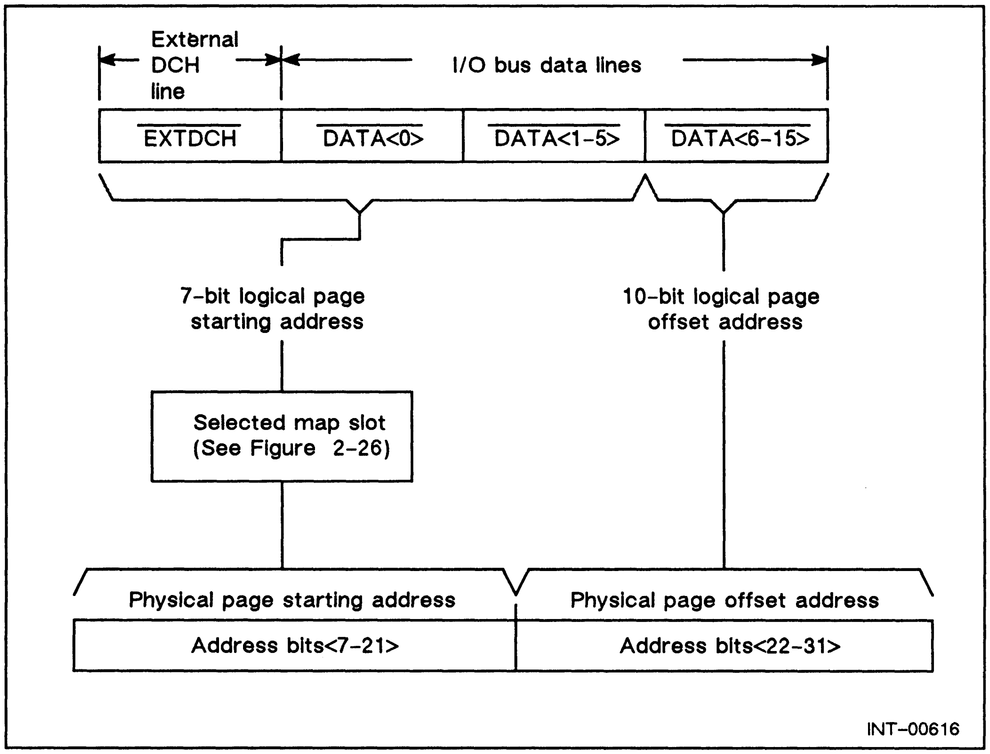

External DCH line
I/O bus data lines
EXTDCH
DATA<0>
DATA<1-5>
DATA<6-15>
7-bit logical page starting address
10-bit logical page offset address
Selected map slot (See Figure 2-26)
Physical page starting address
Physical page offset address
Address bits<7-21>
Address bits<22-31>
INT-00616

Figure 2-23 12-Bit Logical-to-Physical Address Translation

Data Channel Transfer Modes

As part of the device controller's response to ^**DCHA**, it should return the mode of the transfer to the I/O bus controller. All data channel device controllers must designate the mode of transfer using the ^**DCHMO** signal line. In addition, DCH device controllers can designate additional modes of transfer using the ^**DSO** signal line. These signal lines should be asserted whenever DCH SEL (or its equivalent) is set in the device controller. Transfer modes are listed in Table 2-3, followed by a description of the transfer for each mode.

Table 2-3 Data Channel Transfer Modes

^DSO	^DCHMO	Function
0 = H	0 = H	Output transfer
0 = H	1 = L	Input transfer
1 = L	0 = H	Reserved
1 = L	1 = L	Load data channel map (from device controller)

Input

Following the data channel address cycle (^**DCHA**), the I/O bus controller will assert **DCHI** as shown in Figure 2–24. The device controller whose priority conditions are satisfied (that is, whose DCH SEL flag is set) should respond to this by conditioning the I/O bus ^**DATA<0–15>** lines with the data word to be transferred. At the end of the **DCHI** period, the I/O bus controller will load the contents of the ^**DATA<0–15>** lines and write it into the memory location specified during the previous data channel address cycle (^**DCHA**).

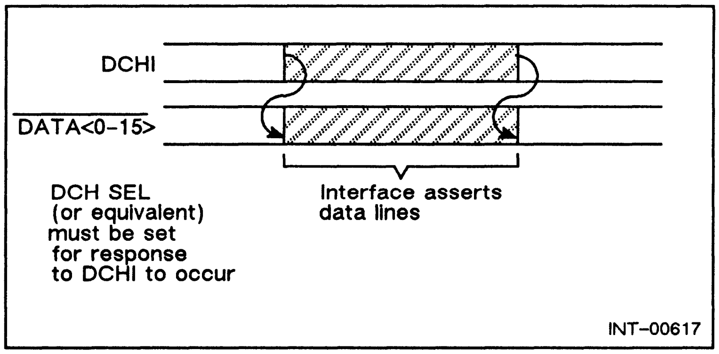


Figure 2–24 Data Channel Input Sequence

Output

Following the data channel address cycle (^**DCHA**), the I/O bus controller will condition the I/O bus ^**DATA<0–15>** lines with the data word read from the memory location specified during the data channel address cycle (^**DCHA**) as shown in Figure 2–25. After allowing time for the lines to settle, **DCHO** will be asserted by the I/O bus controller signaling the device controller to load the data from the ^**DATA<0–15>** lines. The device controller should load the data from the ^**DATA<0–15>** lines at the trailing edge (transistion from low to high level of (**DCHO**). Shortly thereafter the I/O bus controller will remove the data from the ^**DATA<0–15>** lines.

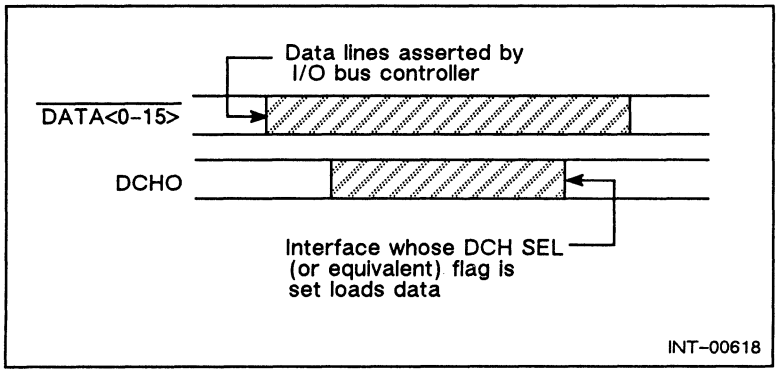


Figure 2–25 Data Channel Output Sequence

Data Channel Map Table

The data channel map table can define up to 512 map slots. Each map slot provides the physical page starting memory address for a 2-Kbyte block of memory. These map slots are located within the data channel map table portion of the channel input/output (CIO) registers (numbered 4000 through 5777_8). Each map slot consists of an even-numbered register and its corresponding odd-numbered register as shown in Figure 2-26.

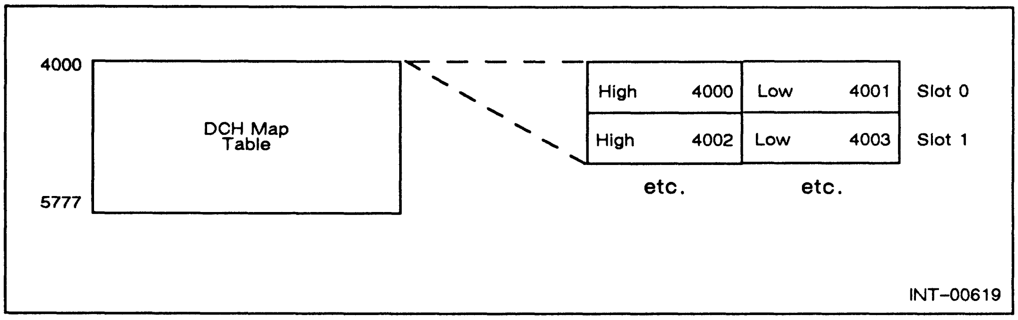


Figure 2-26 Data Channel Memory Map Register Locations

Loading Data Channel Map from I/O Device

In addition to the normal configuring of the data channel map by the processor (using the WLMP, CIO, and CIOI instructions), the map can be upstream loaded from a device controller. In this method, the controller performs two data channel map write operations that write two 16-bit words into a map slot: one to the high-order register, and one to the low-order register. These registers may be written to in any order or, if an existing map is being modified, only one register need be loaded. Data bit 15 of the address word defines which register (high-order or low-order) receives the data word.

Note that the specified data channel map must be enabled for upstream loading to be successful. Data channel mapping is enabled by setting the data channel map enable (DME) bit of the I/O Channel Definition register to 1. When this bit is set to 0, all data channel addresses are unmapped. Note also that when an ECLIPSE system has multiple input/output controllers, each controller has its own I/O Channel Definition register. (For additional information on the I/O Channel Definition register, refer to the ECLIPSE *MV/Family (32-Bit) Systems Principles of Operation* manual as well as the *Principles of Operation Supplement* for your specific system.)

In order for a data channel device controller to specify a load map operation, the controller must assert **^DS0** and **^DCHMO** lines along with the address word when it responds to the data channel acknowledge (**^DCHA**).

Depending upon their design, data channel device controllers that implement the loading of the data channel map function can address any one of 32, 64, 128, or 512 map slots. They do this by asserting a 5-, 6-, 2-, or 9-bit map slot number. Address word formats for data channel map upstream loading are shown in Figure 2-27. Note that **^DATA15** of the address word defines which register (high-order or low-order) receives the data word.

Data formats to be written to the odd- and even-registers are shown in Figures 2-28 and 2-29.

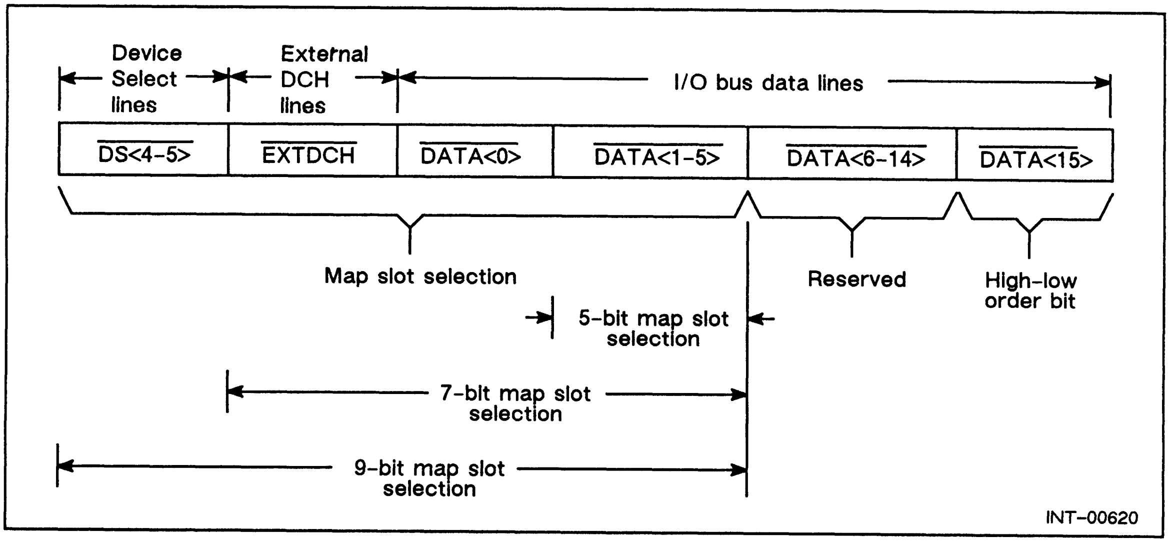
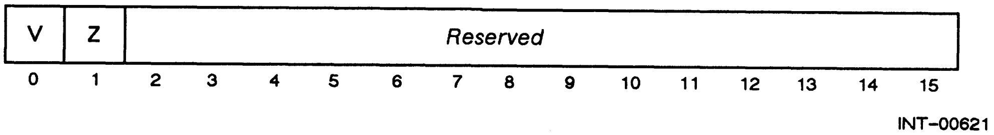


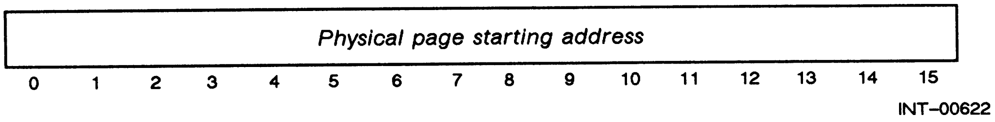
Figure 2-27 Address Word Format for Data Channel Map Upstream Loading



Bits	Field Name	Contents or Function
0	V	Valid page bit: 0 = page access enabled 1 = page access denied*
1	Z	Data transfer bit: 0 = data is transferred 1 = zero bits are transferred
2-15	Reserved	Hardware reserved; returns zero bits when read, should be written with zeros

* If access is attempted, a MAP validity fault condition is generated.

Figure 2-28 Input/Output Map Even-Numbered Register Format (High-Order Word)



Bits	Field Name	Contents or Function
0-15	Physical page starting address	Starting physical memory address of logical page referenced for the particular map slot

Figure 2-29 Input/Output Map Odd-Numbered Register Format (Low-Order Word)

End of Chapter

Chapter 3
I/O Bus Timing and Electrical Specifications

This chapter presents detailed timing characteristics and signal levels for the various I/O bus functions that were described in Chapter 2. The chapter also provides circuit specifications for line drivers and receivers that are to be connected to the I/O bus. The chapter ends by listing suggestions for minimizing electrical noise and provides a suggested input circuit to be used by bus receivers.

Timing

The following timing diagrams, Figures 3–1 through 3–14, show typical timing characteristics of the various ECLIPSE I/O bus functions. These figures show two types of timing information. The times shown for the functions originated in the I/O bus controller (for example, the **DCHI** pulse) are the times for which the device controller I/O bus interface must be designed. The times between I/O bus controller bus functions and the device controller I/O bus interface responses on the bus to those functions are shown as the maximum allowable times. A device controller I/O bus interface designed to operate with these function times and maximum response times will in most cases operate with any NOVA®, ECLIPSE®, or ECLIPSE MV/Family computer. All times are given in nanoseconds and are measured at the backpanel pins of the device controller slot.

NOTE: Signals that have no maximum duration or delay may have an indefinite duration or delay.

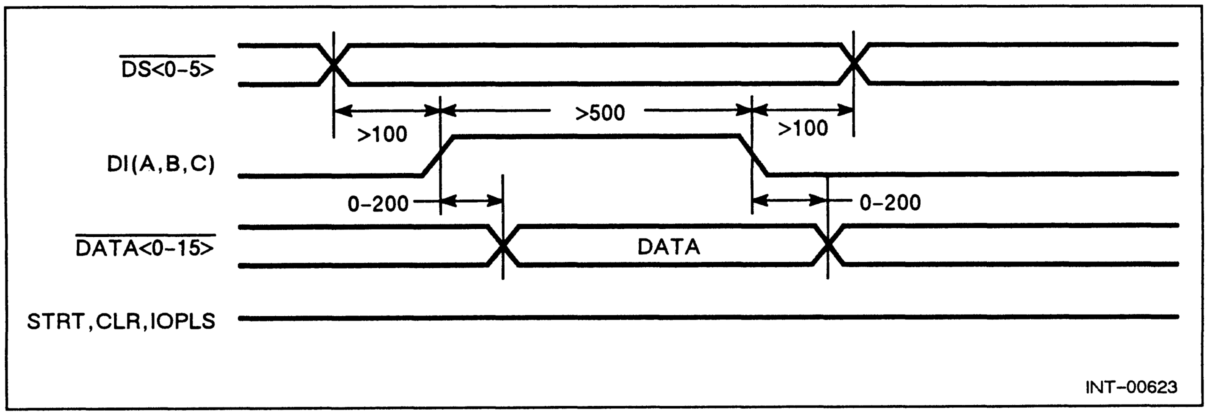


Figure 3–1 Programmed I/O Input Without Start, Clear, or I/O Pulse

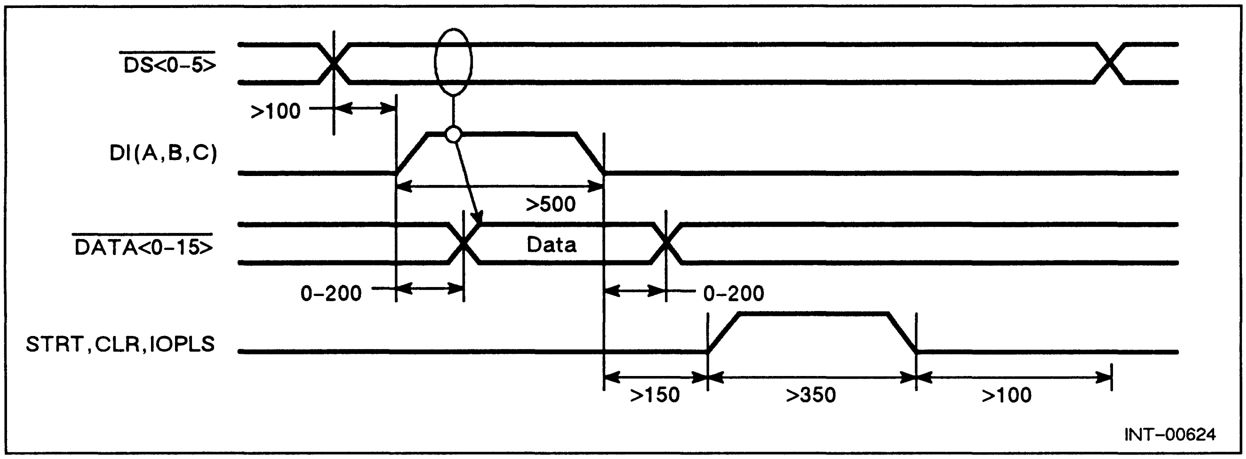


Figure 3-2 Programmed I/O Input with Start, Clear, or I/O Pulse

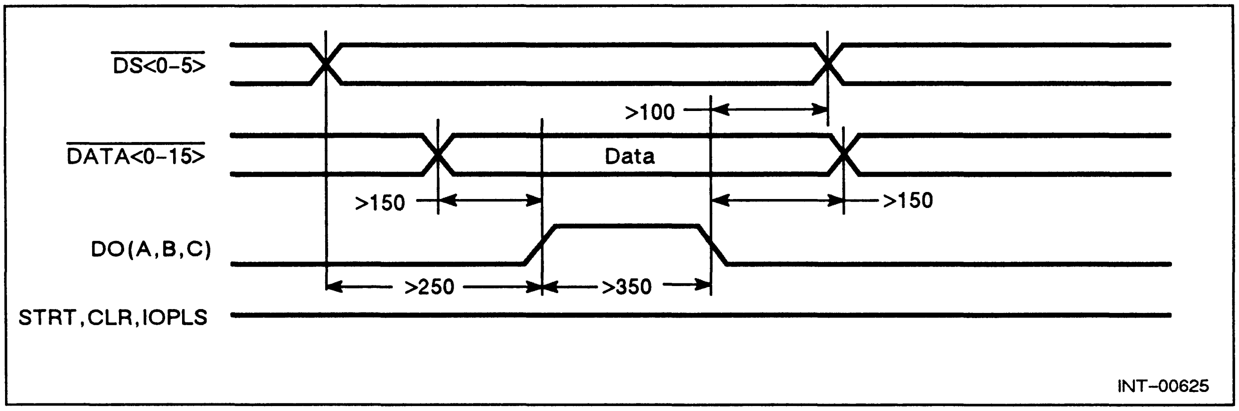


Figure 3-3 Programmed I/O Output Without Start, Clear, or I/O Pulse

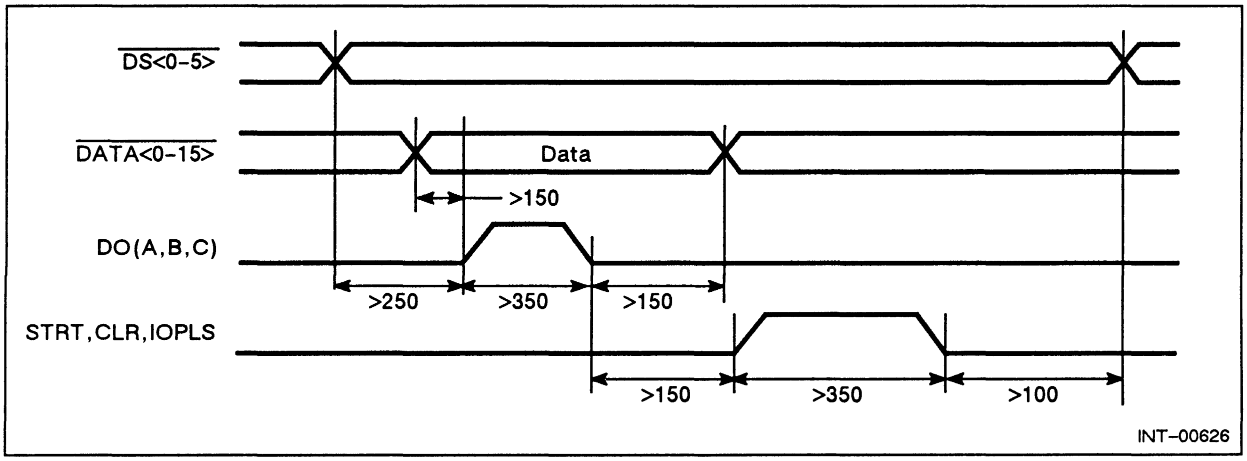


Figure 3-4 Programmed I/O Output with Start, Clear, or I/O Pulse

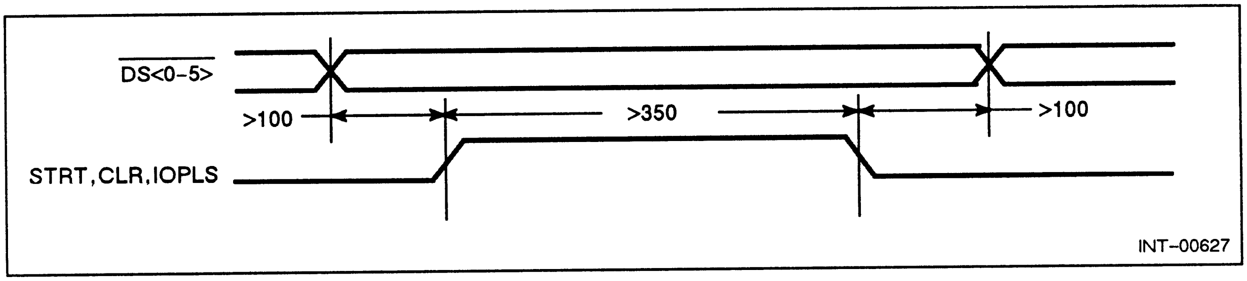


Figure 3-5 Programmed I/O No Data Transfer With Start, Clear, or I/O Pulse

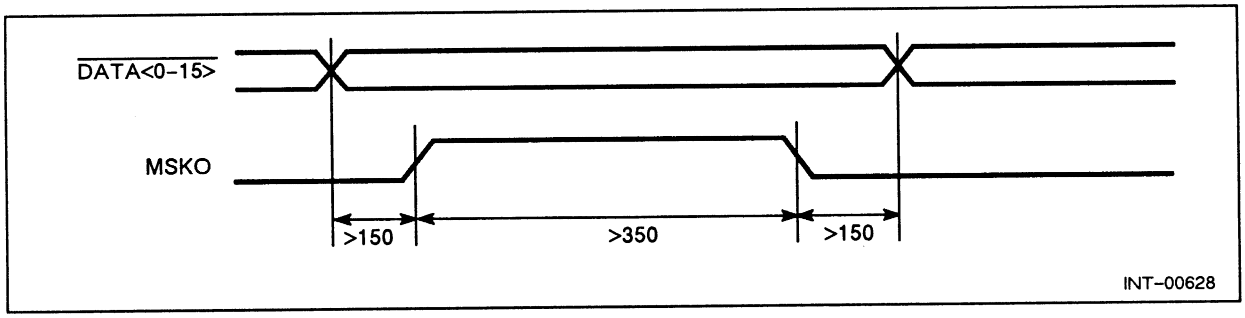


Figure 3-6 Mask Out

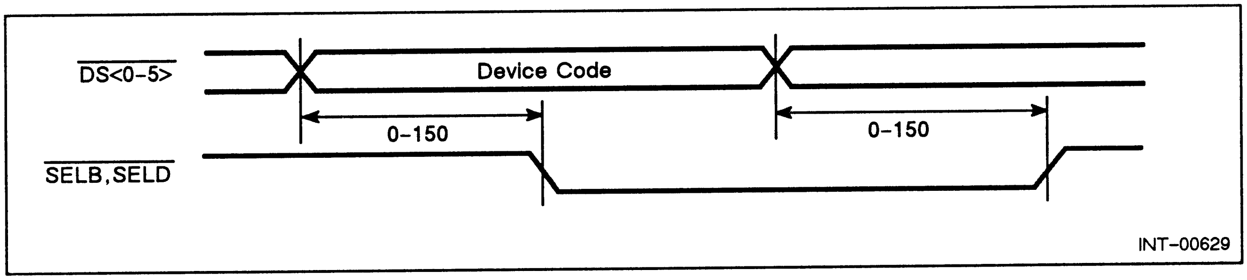


Figure 3-7 Skip Test

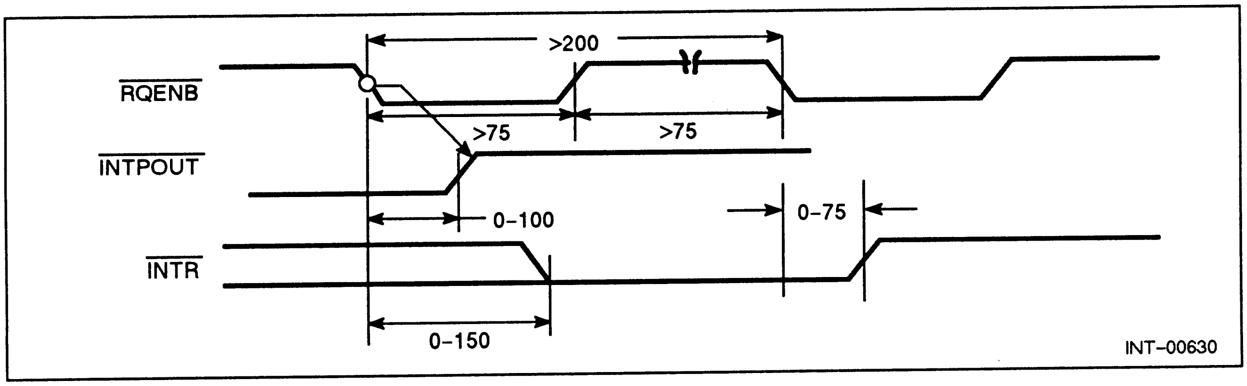


Figure 3-8 Program Interrupt

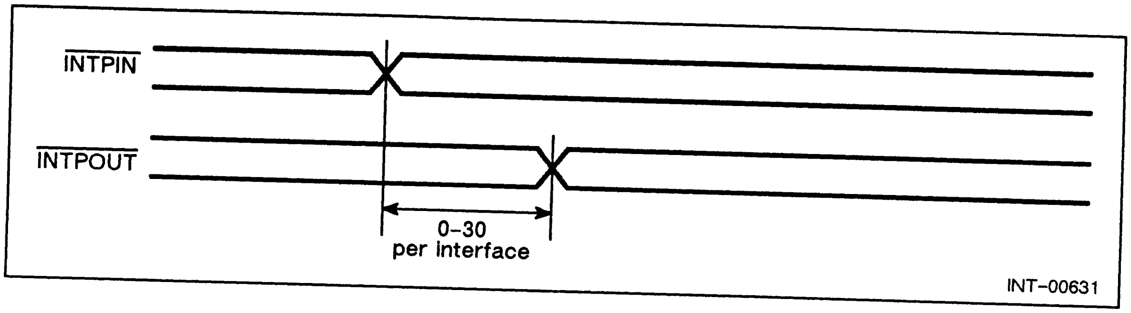


Figure 3-9 Interrupt Priority Chain

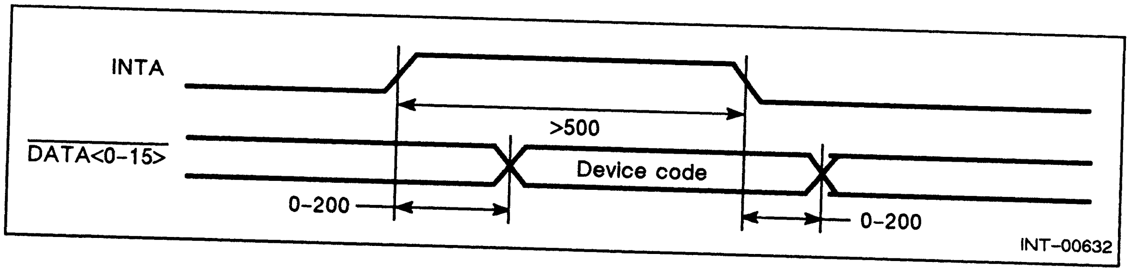


Figure 3-10 Interrupt Acknowledge

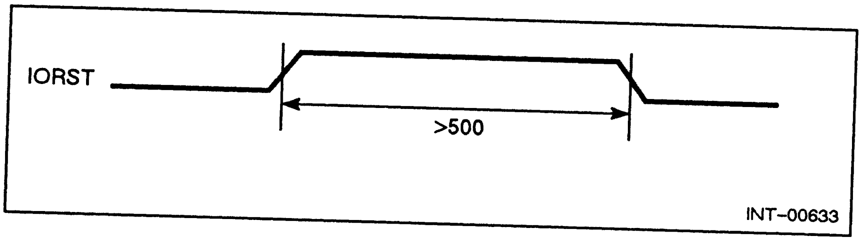
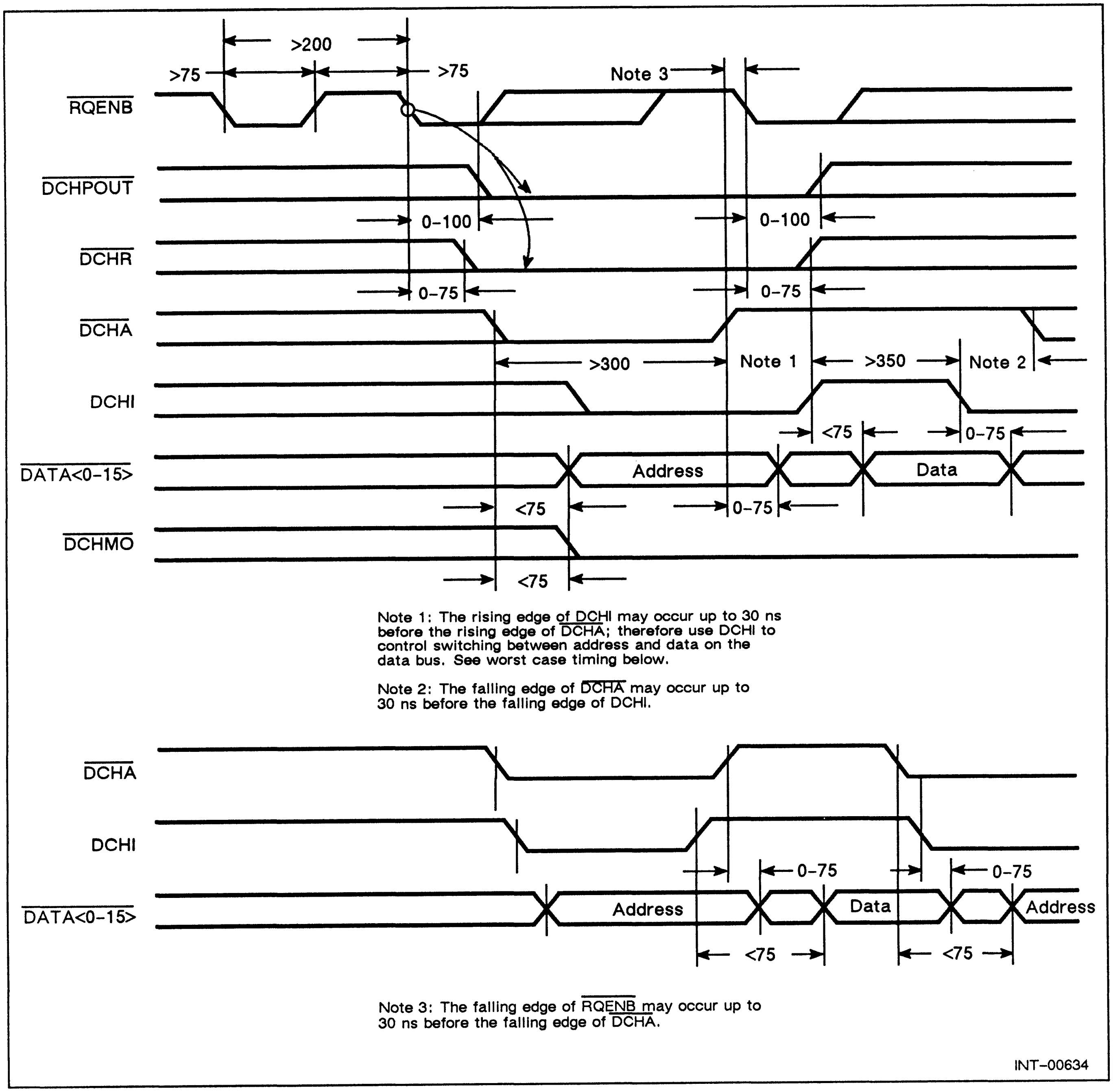


Figure 3-11 I/O Reset

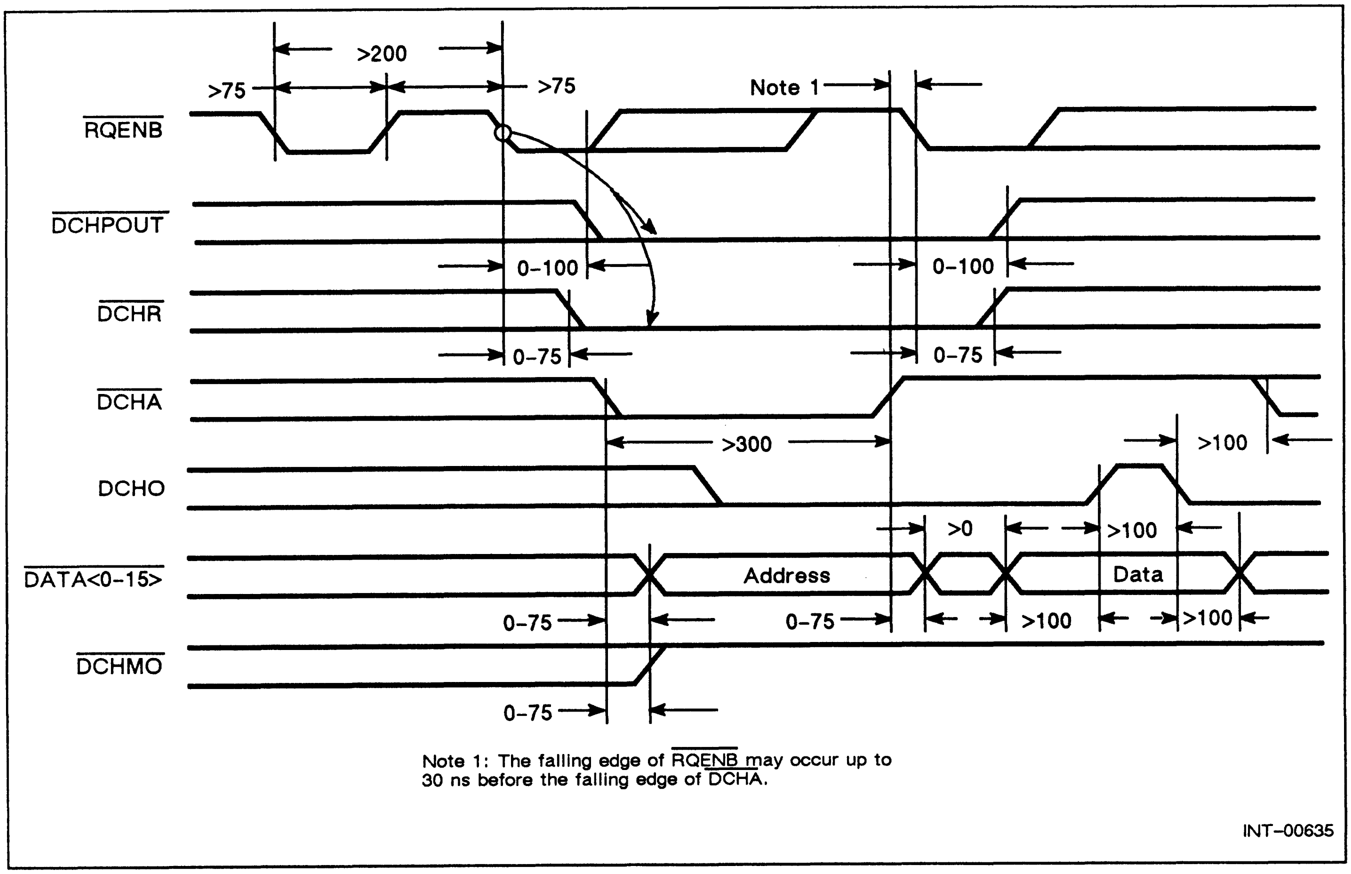


Note 1: The rising edge of DCHI may occur up to 30 ns before the rising edge of DCHA; therefore use DCHI to control switching between address and data on the data bus. See worst case timing below.

Note 2: The falling edge of DCHA may occur up to 30 ns before the falling edge of DCHI.

Note 3: The falling edge of RQENB may occur up to 30 ns before the falling edge of DCHA.

Figure 3–12 Data Channel Input



Note 1: The falling edge of RQENB may occur up to 30 ns before the falling edge of DCHA.

Figure 3-13 Data Channel Output

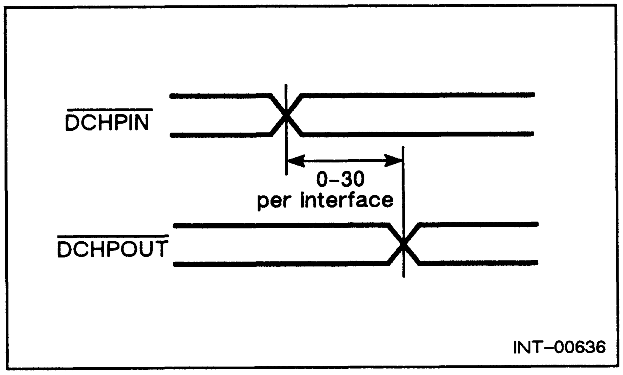


Figure 3-14 Data Channel Priority Chain

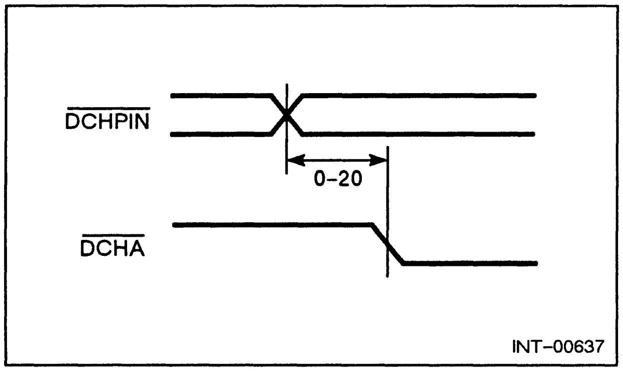


Figure 3–15 Data Channel Priority to Acknowledge Delay

Signal Levels

All of the signals on the ECLIPSE I/O bus are digital signals and thus have two electrical states. As shown in Figure 3–16, any voltage between 0 and +0.7 volts is considered low while any voltage greater than +2.0 volts is considered high. The normal voltages for low and high levels are 0.5 and +2.7 to +3 volts, respectively. The relation between the electrical level of a signal and its logical value depends on the particular signal in question. Signals that are asserted when low are identified by a bar over the signal name (for example, ^**CLR**) as explained in the Preface of this manual.

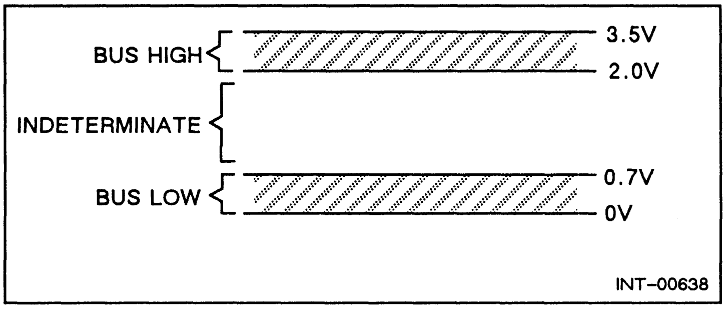


Figure 3–16 Bus Electrical Levels

Drivers

All signals that may be generated by the device controller interface as well as several other signals normally float high and are driven by pulling them towards the 0 volt level. This is done by causing enough current to flow through the line's terminating resistor from the +5 volt source so that the full 5 volts are dropped across the resistor. This is usually accomplished by using the collector of an NPN transistor, as shown in Figure 3–17. Generally, a discrete transistor is not used for this purpose; rather the output of an open-collector transistor–transistor–logic (TTL) gate would be used. In any case, the device must be able to sink at least 64 milliamperes with a saturated collector–emitter voltage not greater than 0.5 volts.

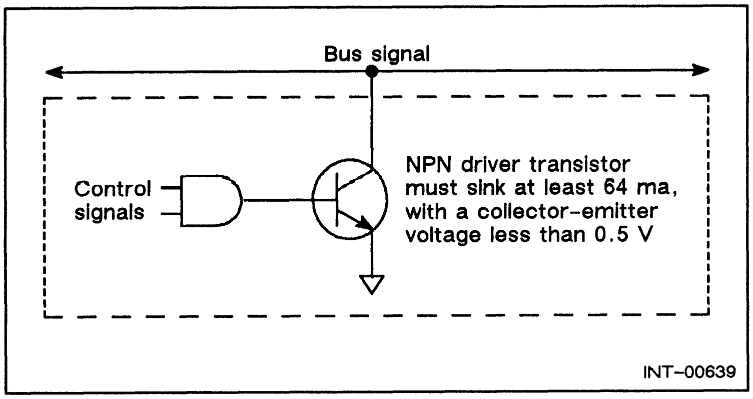

Bus signal
Control signals
NPN driver transistor must sink at least 64 ma, with a collector-emitter voltage less than 0.5 V
INT-00639

Figure 3-17 Typical Line Driver

Receivers

Receivers for the I/O bus signals can be TTL gates. The net load that can be attached to the bus must be limited to 1.6 mA per signal per board.

Noise

In any high-speed circuit design, there should be a concern for noise generation and protection from it. There are many types of noise that should concern the designer. For example, crosstalk between adjacent high-speed signals, ringing of improperly or unterminated signals, effects of ac (capacitive) loading, and transmission line effects are all noise sources. Several suggestions listed below will decrease noise generation:

- Use noise resistant receivers for all high-speed signals, especially those that travel a long distance.
- Keep all I/O bus signals very short. That is, locate the receivers and drivers for I/O bus signals physically close to the printed circuit boards (PCBs) edge connectors where they come off the backpanel. Lay out all bus signal etch to minimize capacitance and crosstalk.
- Synchronize all data, address, or status information so it does not change once it is enabled onto the I/O bus.
- Use only one receiver for each signal. That is, if a signal is used at more than one point on a single board or device controller interface assembly, that board should nevertheless cause only one load on the bus for that signal.
- Treat these signals as transmission lines; their physical as well as electrical characteristics are important.
- Consider the effect on signal delays and pulse widths caused by bus receivers, transmitters, repeaters, and filters.

The logical, electrical, and mechanical design of bus device controller interfaces and interconnects must minimize the generation of, and vulnerability to, electrical noise. Nevertheless, the noise immunity of 7400-series logic circuits may not be adequate, in all cases, to receive I/O bus signals. Bus receivers for the signals listed below may require the filter circuit shown in Figure 3–18.

DATIA
DATIB
DATIC
DATOA
DATOB
DATOC
^DCHA
DCHI
DCHO
STRT
CLR
IOPLS
INTA
^MSKO
IORST

All other signals should use Schmitt trigger receivers.

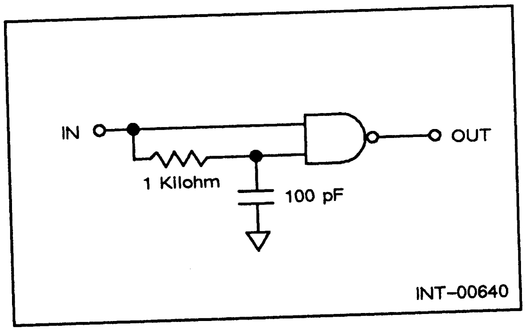


Figure 3–18 Bus Receiver

End of Chapter

Chapter 4
The BMC Bus

The burst multiplexor channel (BMC) bus structure contains a single bus connecting the ECLIPSE BMC facility of the Input/Output channel to the peripheral device controllers.

Summary of BMC Bus Signals

The BMC bus comprises 64 signal lines that can be grouped into 6 categories. Figure 4–1 shows the BMC bus signals in their respective groups and Table 4–1 provides a brief description of the functions for each signal. All signals on the bus are active low. (Refer to "Logic Conventions" in the preface of this manual for signal level and signal name descriptions.) Timing information for the BMC bus signals is shown in Chapter 10. All BMC bus signal lines run in parallel to the BMC bus controller and all device controllers connected to the bus.

Data and an associated parity bit are transferred on the bus along 17 parallel, bidirectional data lines. Addresses and an associated parity bit are transferred along 22 parallel, unidirectional address lines. Control signals are carried along dedicated, unidirectional control lines that specify a unique function. Each device controller that uses the BMC facility has its own single request line. The eight request lines run in parallel to all device controllers in order that each controller can independently determine if it is the highest priority controller that is making a request. All activities on the BMC bus are synchronized by a clock signal supplied by the controller. The BMC bus is TTL compatible.

The BMC bus signals are carried by two flat–ribbon cables that connect the BMC bus controller to device controllers that use the BMC facility. These cables originate from two 40–pin connectors on the front edge of the board containing the BMC bus controller and daisy chain through two 40–pin connectors on the front edge of each device controller connected to a specific BMC bus controller. The BMC bus chain must be terminated at the device controller that is furthest from the BMC bus controller on the chain.

Interactions between the BMC bus controller and the BMC device controller are somewhat complex because of the number of functions performed for each transfer sequence. Design of a burst multiplexor channel interface is relatively restricted, and Data General recommends that such a design correspond to the description presented in this chapter.

The BMC may connect to as many as 8 device controllers, typically (the number may be less on some systems because of slot, power, or BMC bus current limitations). One or more BMC facilities may be present, depending on the specific system. Figure 4–2 shows a typical interconnection of peripheral devices and a common BMC controller.

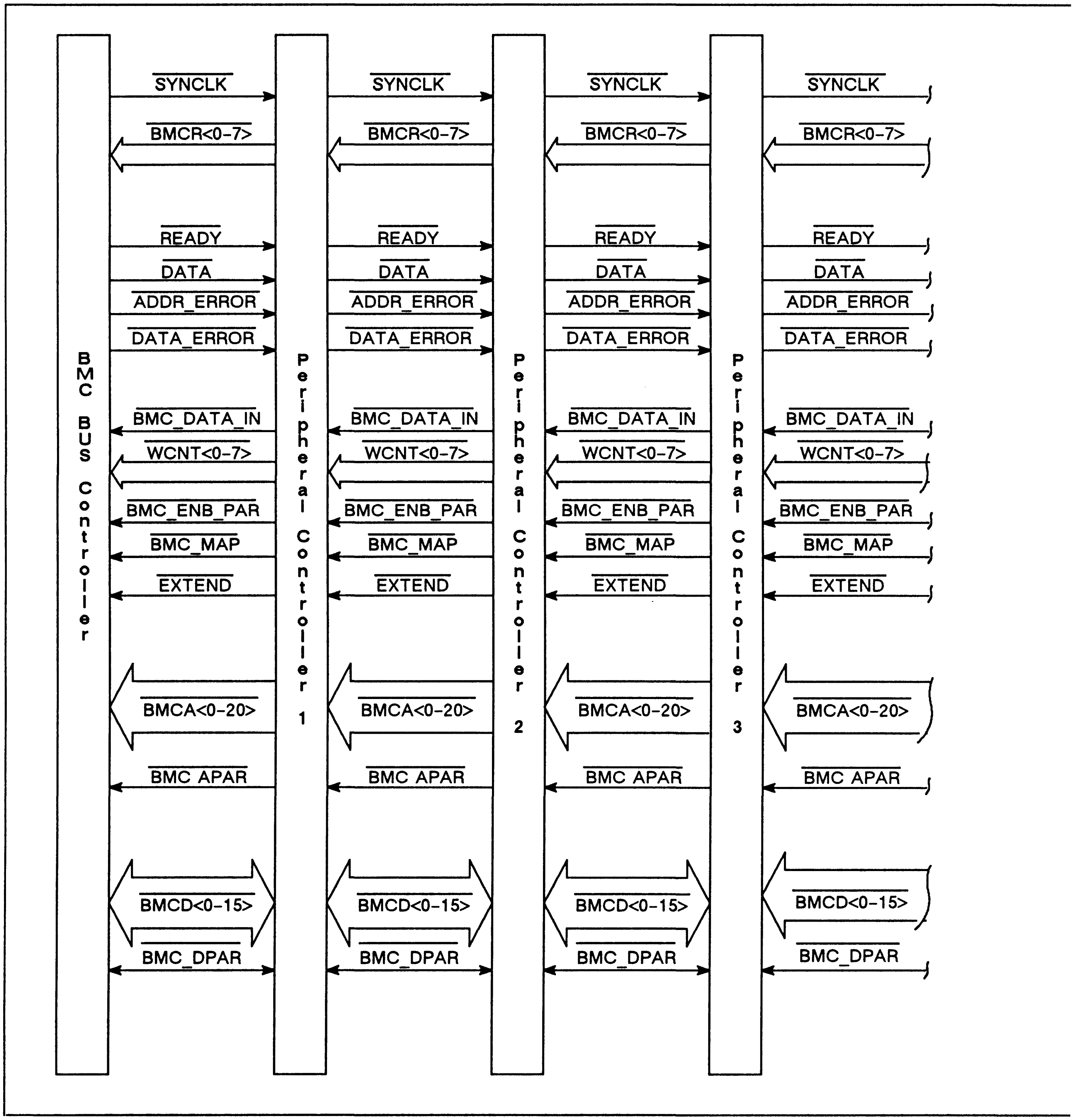
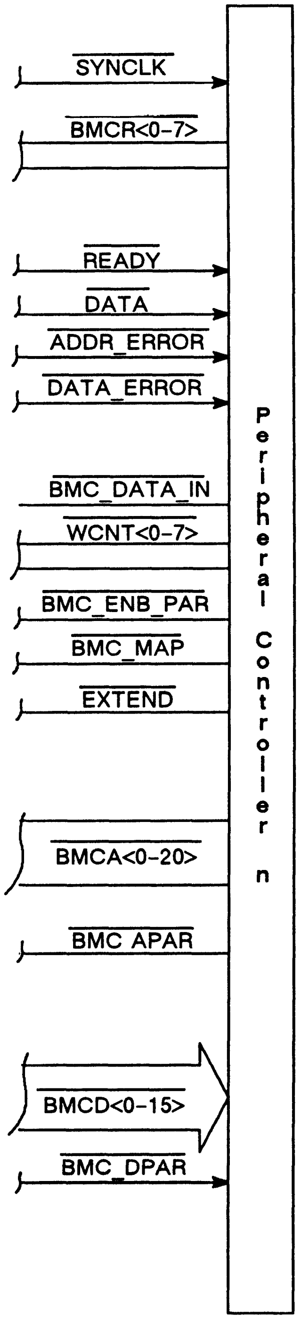


Figure 4-1 The BMC Bus



Group	Signal	Description
Sync	SYNCLK	Synchronizing clock
Transfer request	BMCR<0-7>	Transfer sequence request
Transfer controls	READY	Ready to accept transfer sequence request
	DATA	Ready to transfer data
	ADDR_ERROR	Address parity error or validity check error
	DATA_ERROR	Data parity error on input data transfer
Channel controls	BMC_DATA_IN	Direction of data transfer (in or out)
	WCNT<0-7>	Number of data words to transfer
	BMC_ENB_PAR	Enable address and data parity checking
	BMC_MAP	Enable address mapping
	EXTEND	Type of data transfer (map or normal)
Address	BMCA<0-20>	Address word
	BMC APAR	Address word parity bit
Data	BMCD<0-15>	Data word
	BMC_DPAR	Data word parity bit

INT-00642

Table 4-1 BMC Bus Signals

Signal	Description
Synchronization Signals	
^SYNC_CLK	*Synchronization Clock.* This signal line synchronizes all activities on the BMC bus. This signal is generated by the BMC bus controller. Each period of ^SYNC_CLK is called a BMC bus cycle. BMC bus cycles can be as short as 100 nanoseconds and can be extended if the BMC bus controller is delayed waiting for a memory cycle. All information flowing between the BMC bus controller and device controllers must be strobed by the active edge (transistion from high to low level) of ^SYNC_CLK.
Address Signal Lines	
^BMC_ADDR<00-20>	*Burst Multiplexor Channel Address.* These signal lines transfer a starting memory address (21-bit physical or 20-bit logical) between the active BMC device controller and the BMC bus controller.
^BMC_APAR	*Burst Multiplexor Channel Address Parity.* This signal line transfers the address parity bit between device controllers attached to the BMC bus and the BMC bus controller when BMC parity is enabled (see also the ^BMC_ENB_PAR entry).
Data Signal Lines	
^BMC_DATA<00-15>	*Burst Multiplexor Channel Data.* These signal lines transfer all data between the BMC bus controller and device controllers attached to the BMC bus.
^BMC_DPAR	*Burst Multiplexor Channel Data Parity.* This signal line transfers the data parity bit between the BMC bus controller and device controllers attached to the BMC bus when BMC parity is enabled (see also the ^BMC_ENB_PAR entry).
Transfer Request Signal Lines	
^BMCR<0-7>	*Burst Multiplexor Channel Request.* These signal lines request the BMC bus controller to initiate a transfer sequence. Each BMC device controller is assigned a unique request line. (^BMCR7 is highest priority; ^BMCR0 is the lowest).
Transfer Control Signal Lines	
	Using the transfer control signals, the BMC bus controller directs the sequence of events that occurs during a data transaction and notifies the active device controller if an addressing (parity or protected memory area) error or data parity error occurs.
^READY	*Ready.* This signal line specifies that the BMC bus controller has accepted a BMC device controller transfer request. The highest priority device controller requesting service must place an address and channel control word on the BMC bus for transfer to the BMC bus controller during the next bus cycle.
^DATA	*Data.* This signal line specifies that the BMC bus controller is in the data transfer phase of a transfer sequence.

(continued)

Table 4-1 BMC Bus Signals

Signal	Description
Transfer Control Signal Lines (Continued)	
^ADDR_ERROR	*Address Error.* This signal line notifies the device controller that a parity error was detected on the address transferred during an address and control word bus cycle when parity checking is enabled (see also the ^BMC_ENB_PAR entry). Also notifies the device controller that access to a protected memory area (validity protected) was attempted during a data cycle that specified that mapping is enabled (see also the ^BMC_ENB_PAR entry).
^DATA_ERROR	*Data Error.* This signal line notifies the device controller that a parity error was detected on a data-in word transfer cycle when parity checking is enabled (see also the ^BMC_ENB_PAR entry).
Channel Control Signal Lines	
	Using the channel control signals, the device controller provides, to the BMC bus controller, information that details the transfer to be performed. This control information indicates the parity mode, memory addressing mode, transfer mode, and number of data words minus 1 that are to be transferred for the subsequent data transfer.
^BMC_ENB_PAR	*Burst Multiplexor Channel Enable Parity.* This signal line specifies whether or not parity checking is to be performed on the starting memory address and on each data-in word transferred during the subsequent data transfer. When asserted low, it specifies parity is to be checked.
^BMC_MAP	*Burst Multiplexor Channel Mapping.* This signal line specifies whether the starting memory address contained on the ^BMC_ADD<00-20> signal lines is a logical or physical memory address. When asserted low, it specifies that the address is logical and that addresses are to be translated to physical memory addresses (mapped) when memory is accessed during the subsequent data transfer.
^EXT	*Extend.* This signal line specifies whether data transferred during a data-in transfer is to be written into the BMC Map (BMC upstream Map load) or into memory. This signal line has meaning only when channel control signal ^BMC_DATA_IN is asserted at the same time. When ^EXT and ^BMC_DATA_IN are both asserted low, data words from the device controller are written into the BMC Map during the subsequent data transfer. The BMC upstream Map load feature is not implemented on all BMC device controllers.
^BMC_DATA_IN	*Burst Multiplexor Channel Data In.* This signal line specifies the direction of the subsequent data transfer. When asserted low, it specifies that data words are to be transferred from the device controller into memory or the BMC Map (see also the ^EXT entry); when asserted high, it specifies that data words are to be transferred from memory to the device controller.
^WCNT<0-7>	*Word Count.* This signal line specifies the number of data words to be transferred during the subsequent data transfer. The word count specifies one less than the actual number of words to be transferred.

(concluded)

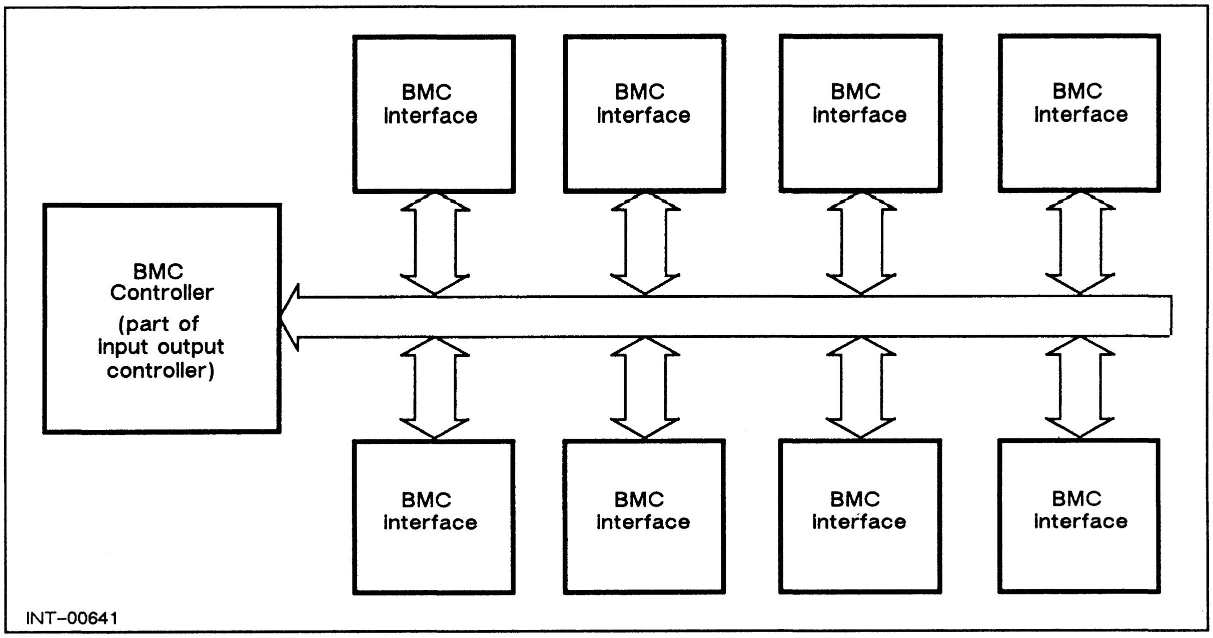


Figure 4-2 BMC Bus Interconnection

Bus Synchronization

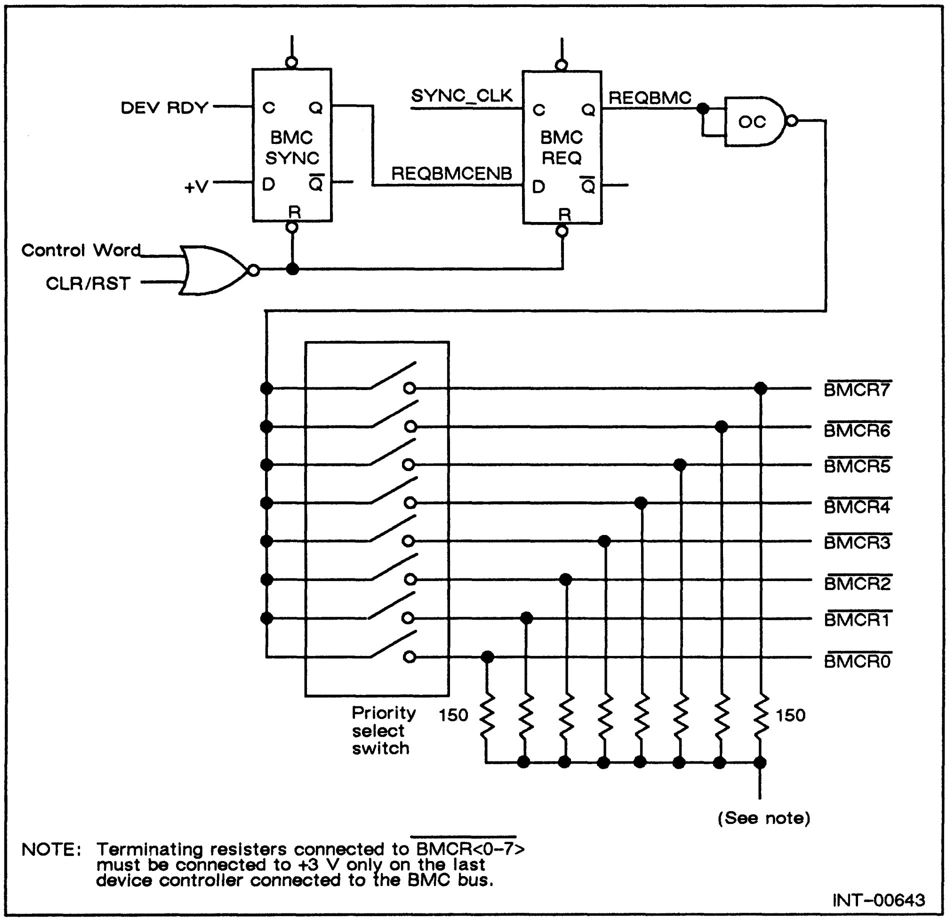
All activity of the BMC facility is synchronized with a clock signal (^SYNC_CLK). A period of this synchronizing clock (from active edge to active edge) is referred to as a BMC bus cycle. (The *active edge* of the clock signal is the transition from a high voltage level to a low voltage level.) The minimum period of a BMC bus cycle is 100 ns. However, BMC bus cycles vary in length depending on the amount of time the BMC facility is delayed while waiting for memory operations to complete. Device controllers that utilize the BMC facility must make transfer requests and place information on or take information from the BMC bus only on the active edge of the synchronizing clock.

Transfer Sequence

Each transfer sequence between the BMC device controller and BMC bus controller consists of a transfer request by the device controller, an acceptance of that request by the bus controller, an address and channel control word transfer by the device controller, and a burst (group) of 16-bit data words (1-256 words) transferred into or from the device controller. The BMC bus controller services a BMC request completely without software intervention.

Burst Multiplexor Channel Request

When a device controller requires BMC bus controller sevice to transfer data into or from memory or to the BMC map table, its BMC interface logic issues a transfer sequence request to the BMC bus controller by asserting its respective request signal line, ^**BMCR<0-7>** on the BMC bus. (See Figure 4-3). Each BMC device controller is assigned a unique request line and more than one request line can be active at a time. (^**BMCR7** is highest priority; ^**BMCR0** is the lowest.) The BMC bus controller monitors all the request lines, but it does not distinguish priority among them. Conflicting access requests must be resolved by the device controllers (see the "Burst Multiplexor Channel Priority" section).



NOTE: Terminating resisters connected to $\overline{\text{BMCR<0-7>}}$ must be connected to +3 V only on the last device controller connected to the BMC bus.

INT-00643

Figure 4-3 Typical BMC Request Circuit

The BMC bus controller assumes that

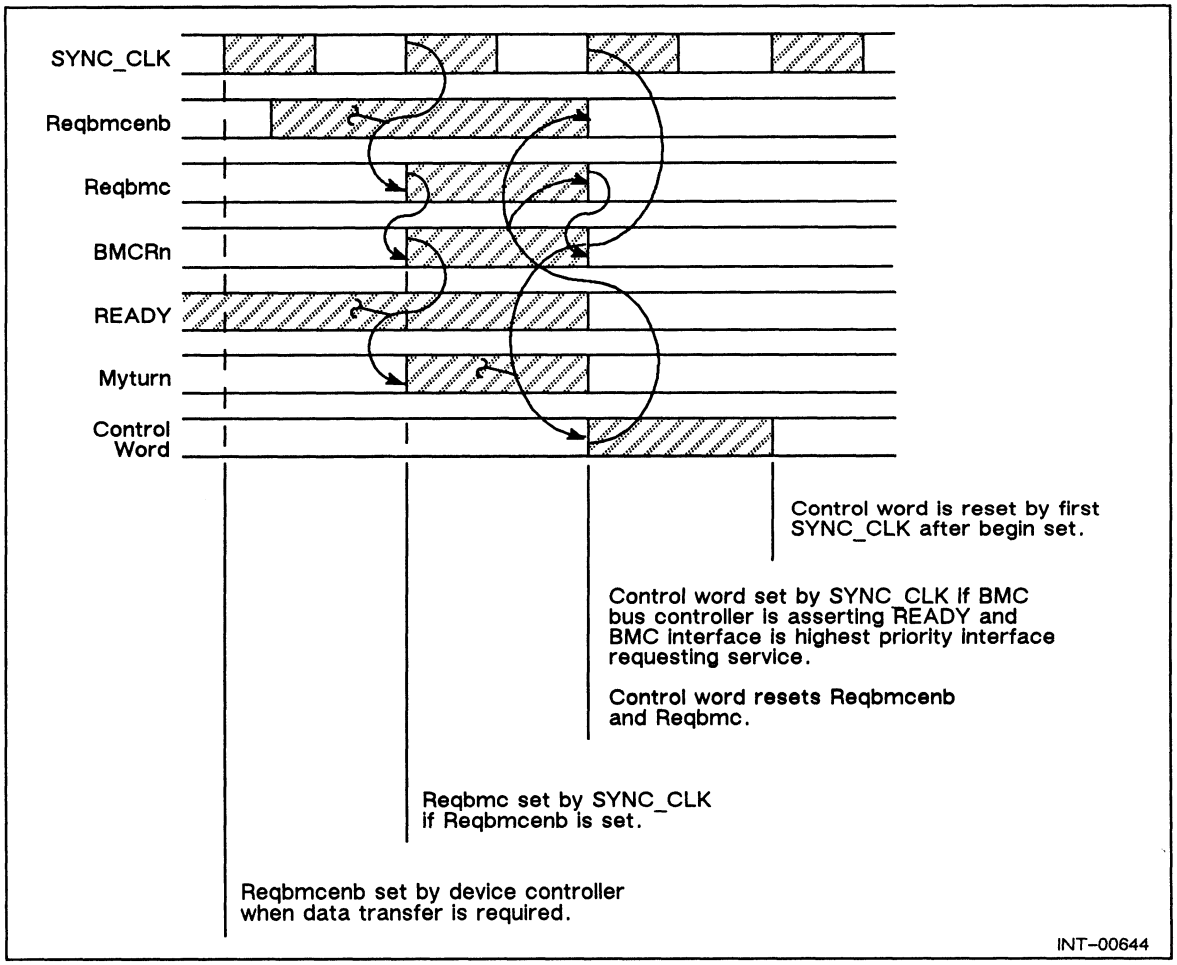
- Each BMC device controller knows which other BMC device controllers are making requests,
- Each BMC device controller determines independently whether it is the one making the highest priority request,
- Only one BMC device controller determines that it is making the highest priority request,
- The BMC device controller making the highest priority request presents an address and channel control word to the BMC bus controller and transfers the data burst when requested by the BMC bus controller.

The signal ^**SYNC_CLK**, generated by the BMC bus controller, is used to clock BMC requests. BMC request signal lines must be clocked only on the leading edge of ^**SYNC_CLK**. The usual convention is to use ^**SYNC_CLK** to clock the state of a flip–flop that is controlled by the device controller (BMC SYNC in Figure 4–3) into a BMC request (BMC REQ) flip–flop, which in turn drives the respective ^**BMCR<0–7>** signal line.

NOTE: A device controller should not request back–to–back BMC transfer requests in order that lower priority BMC interfaces requiring service can periodically obtain use of the BMC facility. (Refer to the section "BMC Break Circuitry" later in this chapter for detail.)

When the BMC bus controller sees any ^**BMCR<0–7>** signal line asserted, it performs a BMC transfer sequence from the highest priority device controller requesting service. During the transfer sequence, the BMC bus controller asserts ^**SYNC_CLK** to synchronize transfers. These ^**SYNC_CLK** signals allow other BMC device controllers to assert their respective request line; and if any BMC request line is asserted at the end of the current transfer sequence, the BMC bus controller performs another BMC transfer sequence to the highest priority controller that is requesting service. The BMC bus controller continues to perform BMC transfer sequences in this way until no BMC device controller is requesting service.

Figure 4–4 and Figure 4–5 show the timing sequence for BMC transfer requests and acknowledgments.



Control word is reset by first SYNC_CLK after begin set.

Control word set by SYNC_CLK if BMC bus controller is asserting READY and BMC interface is highest priority interface requesting service.

Control word resets Reqbmcenb and Reqbmc.

Reqbmc set by SYNC_CLK if Reqbmcenb is set.

Reqbmcenb set by device controller when data transfer is required.

Figure 4–4 Typical BMC Request Sequence (BMC Bus Controller Ready)

Burst Multiplexor Channel Priority

The only BMC device controller that should respond to the BMC bus controller's transfer signals is that which is the highest priority BMC device controller requesting BMC service.

Because more than one BMC bus request line can be active at a time, a hardware priority system serves to arbitrate between two or more device controllers requesting BMC service at the same time. In the event of simultaneous BMC requests, this priority system must grant service to the device controller that is requesting service and is assigned the highest priority on the BMC bus.

Each BMC device controller must determine independently whether it is the highest priority controller making the bus request. Therefore, each device controller must know which other device controllers are making bus requests. Besides connecting to the BMC bus controller, all BMC request signal lines run in parallel to all BMC device controllers so that each controller can determine which other controllers are requesting service during any BMC bus cycle.

Figure 4–5 shows the timing sequence for BMC transfer requests when the BMC bus controller is not ready.

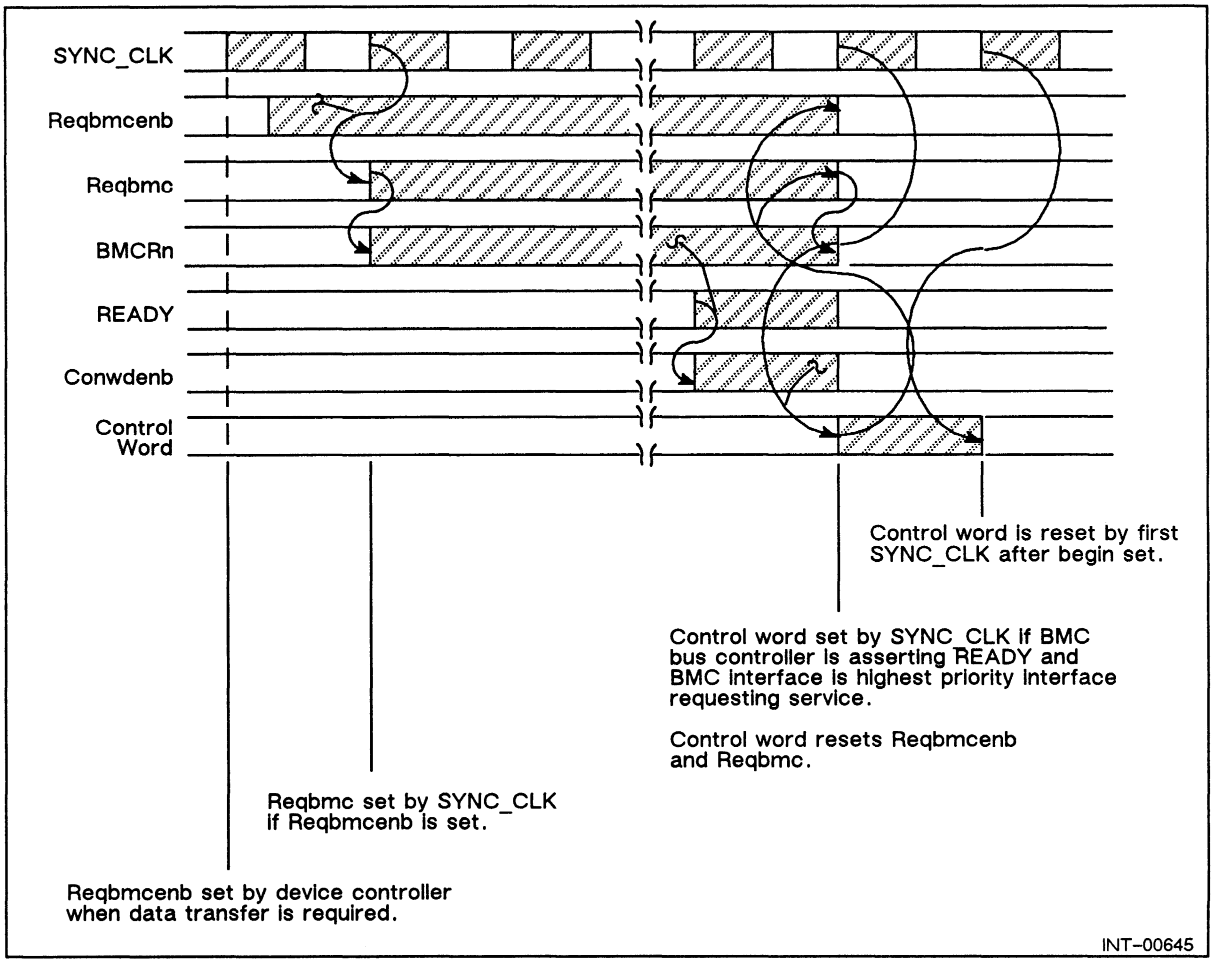
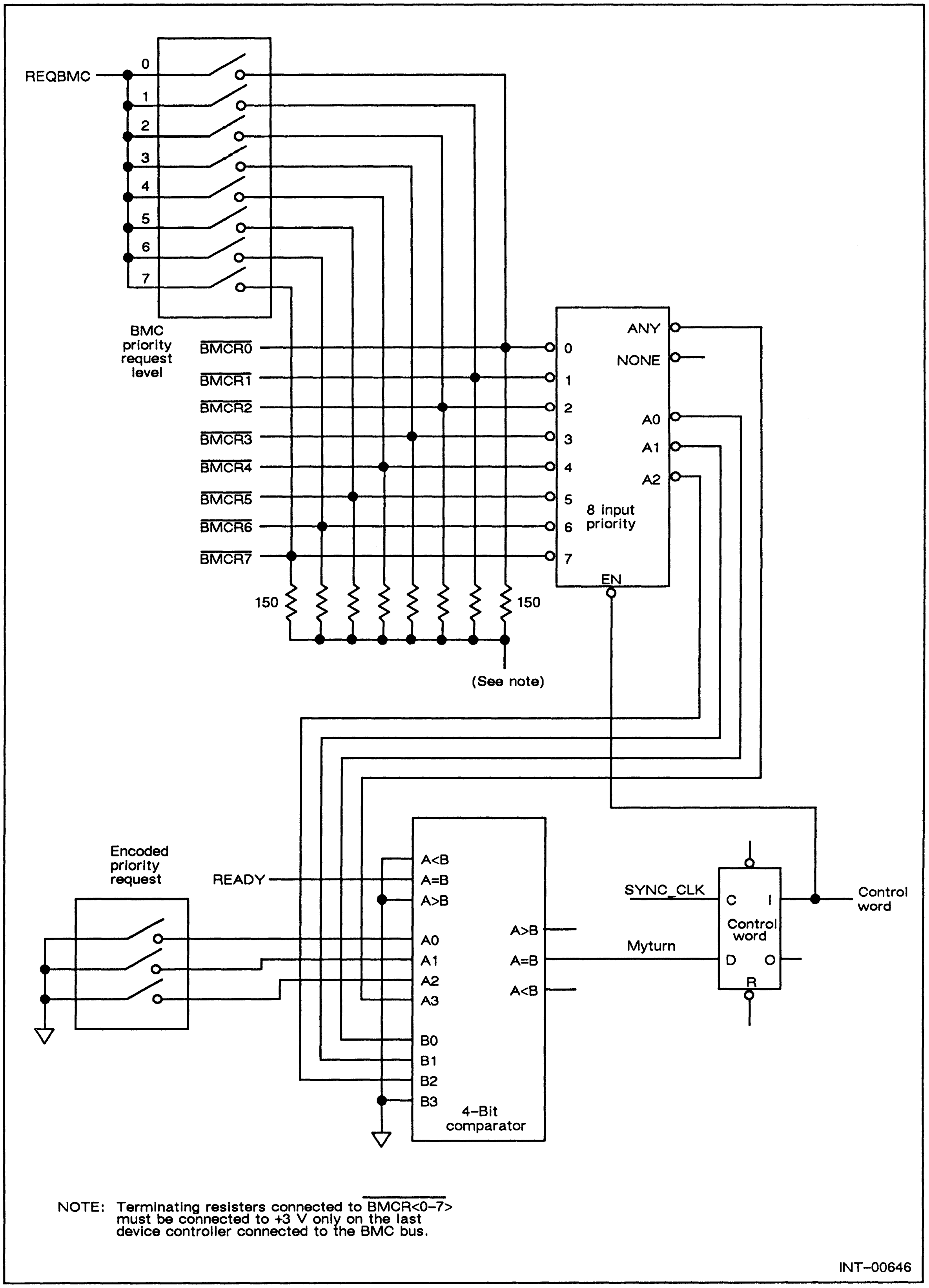

SYNC_CLK
Reqbmcenb
Reqbmc
BMCRn
READY
Conwdenb
Control Word
Control word is reset by first SYNC_CLK after begin set.
Control word set by SYNC_CLK if BMC bus controller is asserting READY and BMC interface is highest priority interface requesting service.
Control word resets Reqbmcenb and Reqbmc.
Reqbmc set by SYNC_CLK if Reqbmcenb is set.
Reqbmcenb set by device controller when data transfer is required.
INT–00645

Figure 4–5 Typical BMC Request Sequence (BMC Bus Controller Not Ready)

Each BMC device controller must monitor all request lines to determine when it is the highest priority controller making a request. The BMC interfaces treat each request line as having fixed priorities (^**BMCR7** is highest priority; ^**BMCR0** is the lowest). The circuit shown in Figure 4–6 shows a suggested implementation for monitoring request priorities.



NOTE: Terminating resisters connected to $\overline{\text{BMCR<0-7>}}$ must be connected to +3 V only on the last device controller connected to the BMC bus.

Figure 4-6 Typical BMC Priority Circuit

BMC Request Acknowledgement

When the BMC bus controller is ready to service a device controller's transfer sequence request, it issues a BMC request acknowledgement signal to all device controllers on the bus by activating its ^**READY** signal line. Transfer requests that occur before the BMC bus controller is ready are delayed until the ^**READY** signal is asserted.

When the BMC bus controller signals acceptance of the transfer request, the highest priority device controller requesting bus controller service must place an address and channel control word on the bus for transfer to the BMC bus controller. This address and control word must be placed on the bus upon receipt of the first active synchronous clock edge following the assertion of ^**READY**. The BMC bus controller negates the ^**READY** signal at this time and begins processing the data transfer.

Note that the ^**READY** signal line will be active whenever the BMC bus controller is not busy performing a transfer sequence. Note also that the BMC bus controller may assert ^**READY** near the completion of a transfer sequence to set up the next transfer sequence.

Figure 4–7 shows typical BMC sequencing from initiating the request through the memory address and channel control word bus cycle.

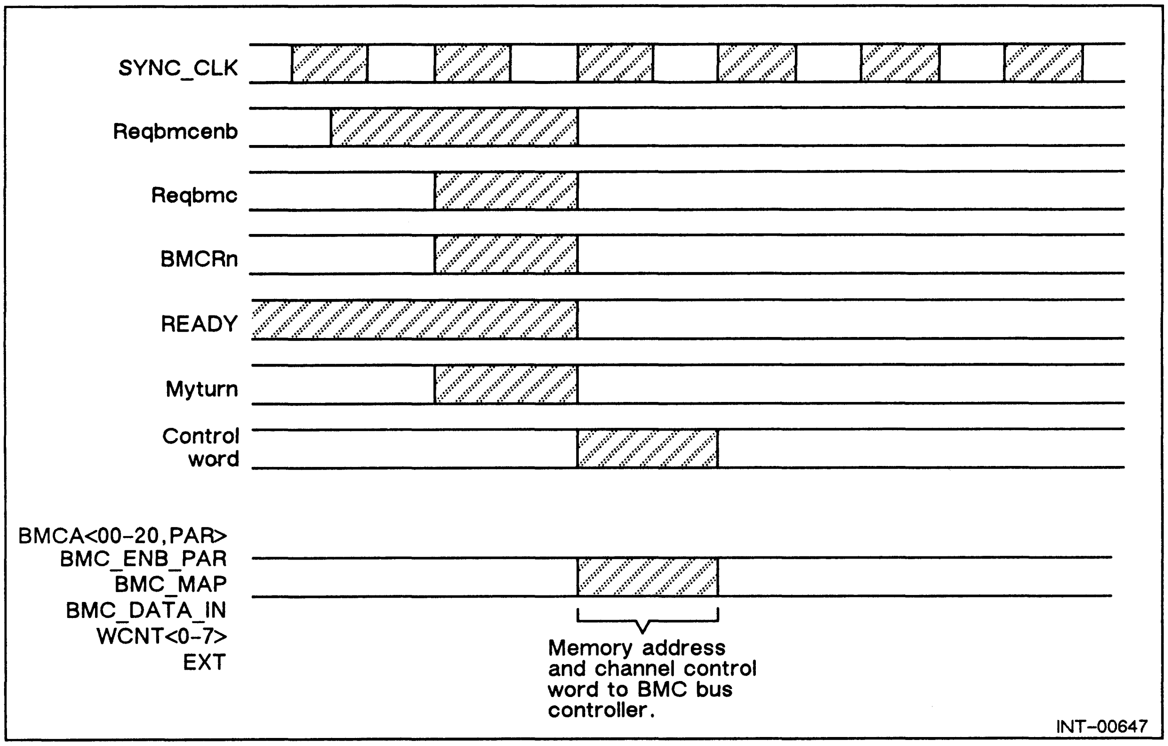


Figure 4–7 Typical BMC Memory Address and Channel Control Word Sequence

Before the transfer sequence is complete, the BMC bus request from the device controller being serviced must be cleared to prevent an immediate second transfer. Therefore, circuitry must be provided to maintain a controller's selected state once the transfer sequence begins.

Figure 4–8 shows the use of a flip–flop, called *Data Enable* (or equivalent), which serves this purpose. On the leading edge of ^**SYNC_CLK**, this flip–flop sets if the device controller is the highest priority controller requesting service and is receiving ^**READY** from the bus controller (signal **Myturn**). This flip–flop is cleared when the data transfer is completed (signal CHANNEL DONE). A device controller must not respond to the BMC bus data transfer signal unless its BMC Selected flag is set.

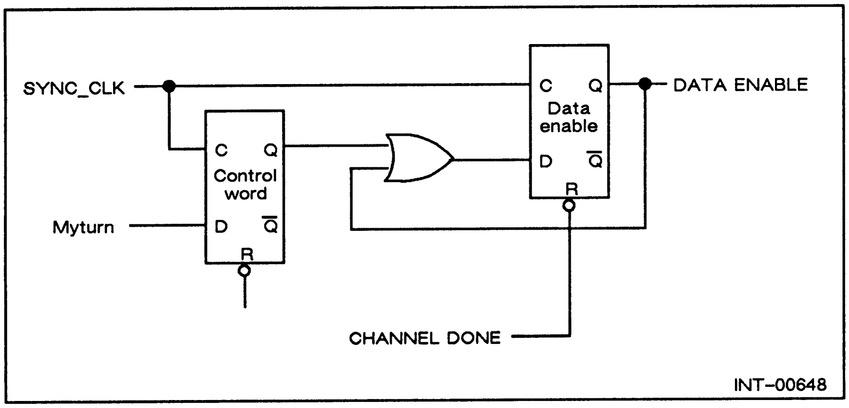
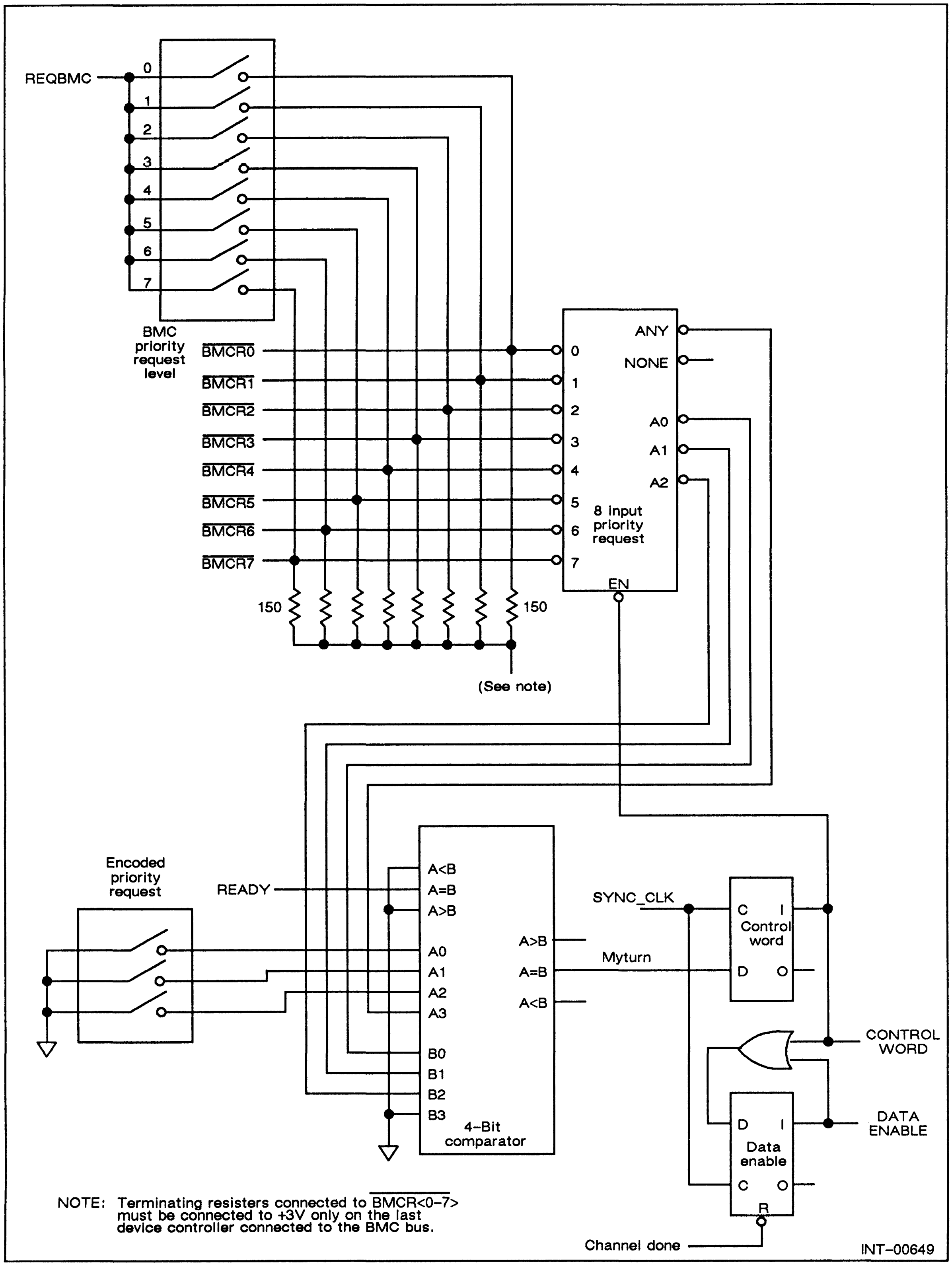


Figure 4–8 Typical BMC Select Circuit

Figure 4–9 shows a complete typical BMC control circuit.



NOTE: Terminating resisters connected to $\overline{\text{BMCR<0-7>}}$ must be connected to +3V only on the last device controller connected to the BMC bus.

Figure 4-9 Typical BMC Control Circuit (Complete)

Address and Channel Control Word

The address and channel control word sent to the BMC bus controller by the highest priority BMC device controller when it receives **^READY** consists of

- Starting memory address (either logical or physical) for the burst of data (^**BMCA<0-20>** and ^**BMCA<PAR>**),
- Parity Enable flag (^**BMC_ENB_PAR**),
- Map Enable flag (^**BMC_MAP**),
- Number of data words (burst length) to be transferred in the burst (^**WCNT<0-7>**),
- Two flags denoting the transfer mode; one the direction of data transfer (input or output — ^**BMC_DATA_IN**), and the other the Map Table Load Operation flag (^**EXT**).

Address

The 21 unidirectional address and the address-parity signal lines are conditioned by the active device controller during the address and control-word bus cycle. This address specifies either a 21-bit physical or 20-bit logical starting memory address for the subsequent data transfer (see the section "Mapped/Unmapped Transfers" below). Note that when a logical starting memory address is specified, ^**BMC_ADDR00** is not used as an address bit.

When parity checking is specified (^**BMC_ENB_PAR**), the address parity signal line is conditioned to maintain odd parity on the combined address signal lines (^**BMC_ADDR<00-20**) and the address parity line (^**BMC_APAR**) — that is, the sum total of ones in the address word, including the parity bit, is odd. All address bits are significant in maintaining parity.

Channel Controls

The highest priority device controller requesting service transmits channel control information along with the starting memory address. This channel control information indicates the memory addressing mode, parity mode, transfer mode, and number of data words minus one that are to be transferred for the subsequent data transfer. The device controller provides the channel control information by conditioning the following signal lines during the address and channel control word bus cycle.

Memory Addressing Mode

As part of the channel control information, the device controller must specify whether the starting memory address supplied on the address signal lines is to be handled as a logical or physical memory address. The device controller accomplishes this by asserting the Map Enable (^**BMC_MAP**) signal line to specify that the address is logical and is to be translated before addressing memory.

Parity Mode

As part of the channel control information, the device controller must specify whether or not the BMC bus controller is to check parity on the starting memory address and all data words transferred to the bus controller. The device controller accomplishes this by asserting the Parity Enable (^**BMC_ENB_PAR**) signal line to specify that parity is to be checked on the address and data words.

Transfer Mode

As part of the channel control information, the device controller must specify the mode of the transfer. BMC device controllers designate the mode of transfer using their ^**BMC_DATA_IN** and ^**EXT** signal lines. Transfer modes are listed in Table 4–2. A description of each transfer mode is presented later in this chapter.

Table 4–2 BMC Transfer Modes

^EXT	^BMC_DATA_IN	Transfer Mode
0 = H	0 = H	Output data transfer
0 = H	1 = L	Input data transfer
1 = L	1 = L	Load BMC map (from device controller)

Word Count

As part of the channel control information, the device controller must specify the number of data words to be transferred during the subsequent data transfer. The device controller accomplishes this by conditioning the ^**WCNT<0–7>** signal lines.

Although the word count allows data transfers of up to 256 words per transaction, each device controller should limit the count to a small number (8 or 16 words) to minimize latency for lower priority device controllers when multiple device controllers are connected to the BMC bus.

Address Parity Check

When the Parity Enable flag is present in the channel control word received from the device controller, the BMC bus controller checks the starting memory address for odd parity. If a parity error is detected, the bus controller records the address parity error in its Channel Definition register, notifies the device controller of the error by asserting its ^**ADDR_ERROR** signal line in the bus cycle that immediately follows the address and channel control word cycle. When an address parity error occurs the data transaction is canceled. The device controller must not transfer any data words to or expect any data words from the bus controller for the current transfer sequence. The device controller is responsible for the corrective action. (For additional information on the I/O Channel Definition register, refer to the *ECLIPSE MV/Family (32–Bit) Systems Principles of Operation* manual as well as the *Principles of Operation Supplement* for your specific system.*)*

NOTE: An address parity error will be generated if no BMC address signal lines including parity are driven during the address and channel control word bus cycle.

If the BMC bus controller detects an address parity error, it allows another transfer sequence to begin by asserting the ^**READY** signal line.

Unmapped/Mapped Transfers

Device controllers that transfer data via the burst multiplexor channel can specify unmapped or mapped data transfers for individual data blocks via the Map Enable flag in the control word. Unmapped transfers are performed when physical memory addresses are used to access memory. Mapped transfers are performed when physical memory address space differs from logical memory address space. During mapped transfers, the BMC bus controller translates logical memory addresses it receives from the device controller, to physical memory addresses.

When a device controller specifies an unmapped transfer (the Map Enable flag is not active) for a data burst, it sends a 21-bit physical starting memory address to the BMC bus controller. This physical address is used to address memory during the data burst transfer. The BMC bus controller increments this address during the data burst transfer causing consecutive memory locations to be accessed for each successive word transferred.

When a mapped data transfer is specified (the Map Enable flag is active), the device controller sends a 20-bit logical starting memory address to the BMC bus controller. The BMC bus controller translates the logical memory address to a physical memory address before accessing memory during the data burst transfer. The BMC bus controller increments the logical memory address during the data burst transfer, causing consecutive logical memory locations to be accessed for each successive word transferred.

During each memory access of a mapped data burst transfer, the BMC bus controller translates logical memory addresses into physical memory addresses by converting the ten high-order bits of the logical memory address, called the *logical page number*, into a number of bits, called the *physical page number*. This conversion is accomplished by using the ten high-order bits to select a BMC map slot that contains the starting memory address of the physical page. The number of physical page bits varies depending on the amount of memory present in the system. These physical page number bits then combine with the ten low-order bits of the logical memory address, called the *page offset address*. This combination of physical page starting address and page offset address form the physical memory address used to access memory. Figure 4-10 illustrates the bus controller address translation procedure.

NOTE: Because it is the logical memory address that is incremented during the data burst transfer, successive physical memory locations may not be accessed. If the incrementing of the logical address results in a carry from the page offset address into the logical page number, the logical page number is incremented and the physical page number is taken from the next sequential logical page number map slot. Therefore, in mapped data burst transfers, successive data words may not be accessed at consecutive memory locations if the transfer overlaps two logical pages.

As part of the address translation procedure, the BMC bus controller also checks for valid address translations (protected memory areas) during each memory access (see the "Protected Memory Check" section later in this chapter.)

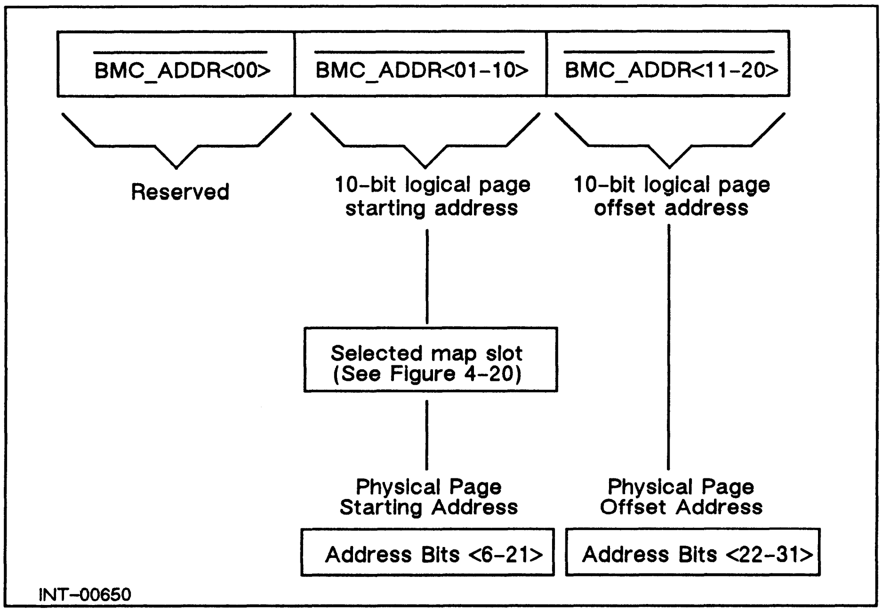

BMC_ADDR<00>
BMC_ADDR<01-10>
BMC_ADDR<11-20>
Reserved
10-bit logical page starting address
10-bit logical page offset address
Selected map slot (See Figure 4-20)
Physical Page Starting Address
Physical Page Offset Address
Address Bits <6-21>
Address Bits <22-31>
INT-00650

Figure 4-10 Logical-to-Physical Address Translation

Data Transfer

During bus cycles that follow the address and channel control word transfer, the number of data words specified by the device controller in the channel control word (^**WCNT<0-7>**) are transferred between the device controller and memory. The direction of the data transfer is determined by the Direction flag (^**BMC_DATA_IN**) contained in the channel control word. Note that when a data-in transfer is specified, the device controller can specify that the data be placed into the BMC map rather than into memory. (Device controllers cannot read data from the BMC Map.)

The BMC bus controller notifies the device controller when the data burst is to be transferred by activating its ^**DATA** signal line. ^**DATA** may be asserted in any clock cycle following the transfer of the address and channel control word. Once asserted, ^**DATA** will be asserted during each ^**SYNC_CLK** high to low level transistion until the entire requested number of data words have been transferred. If either a data parity or address validity error is detected, the signal ^**DATA** may or may not be removed.

During the data transfer, the BMC bus controller

- Performs a data parity check (when parity is enabled) on each data-in word transferred.
- Performs a data parity generation on each data-out word transferred whether parity is enabled or not.
- Decrements the burst word count.
- Translates logical memory addresses to physical memory addresses when mapping is enabled.
- Checks for protected memory area accesses when mapping is enabled.
- Accesses memory as required to store or retrieve data words.

- Increments the memory address as required each time memory is accessed.
- Notifies the active BMC device controller when a data parity error or an access attempt to a protected memory area is detected.

One word (16 data bits plus parity) of the data burst is transferred via the BMC bus data signal lines (^**BMCD<0–15>** and ^**BMCD<PAR>** on the high to low level transistion of each synchronous clock (^**SYNC_CLK**, until the entire burst is transferred. The 16 bidirectional data signal lines are conditioned by the BMC bus controller (data–out) or the active device controller (data–in) for each data word transfer.

The bidirectional data parity signal line is conditioned by the active device controller during each data–in word transfer to maintain odd parity on the combined data signal lines (^**BMC_DATA<00–15>**) and the data parity signal line (^**BMC_DPAR**) — that is, the sum total of ones in the data word, including the parity bit, is odd. This signal line is always conditioned by the BMC bus controller during each data–out word transfer bus cycle to maintain odd parity whether parity is enabled or not.

While waiting for memory operations to be completed, the BMC bus controller may delay the clock signal for short periods during the data transfer.

The BMC bus controller can reassert the ^**READY** signal at the beginning of the last word of a data–in transfer or at the beginning of the next to last word of a data–out transfer (the last word of a data–out transfer can overlap a subsequent control word transfer).

A device controller should not request back–to–back BMC transfer requests in order that lower priority BMC interfaces requiring service can periodically obtain use of the BMC facility. (Refer to the section "BMC Break Circuitry" later in this chapter for detail.)

Input

During data transfers into memory, the active device controller must place the first data word on the BMC bus on the bus cycle immediately following the address and control word bus cycle. This may be before the BMC bus controller asserts ^**DATA**. Succeeding data words must be placed on the bus at each high to low level transistion of ^**SYNC_CLK** until the requested count of data words has been transferred. The BMC bus controller accepts a data word on each high to low level transistion of ^**SYNC_CLK** in the bus cycles when ^**DATA** is asserted. Figure 4–11 shows the timing sequence for a four–word input data transfer.

During BMC input transfers, the device controller must remain selected for one additional bus cycle following the last data word transfer to monitor for a data parity error or protected memory access error (refer to the "Data Parity Check" and "Protected Memory Check" sections later in this chapter for details.)

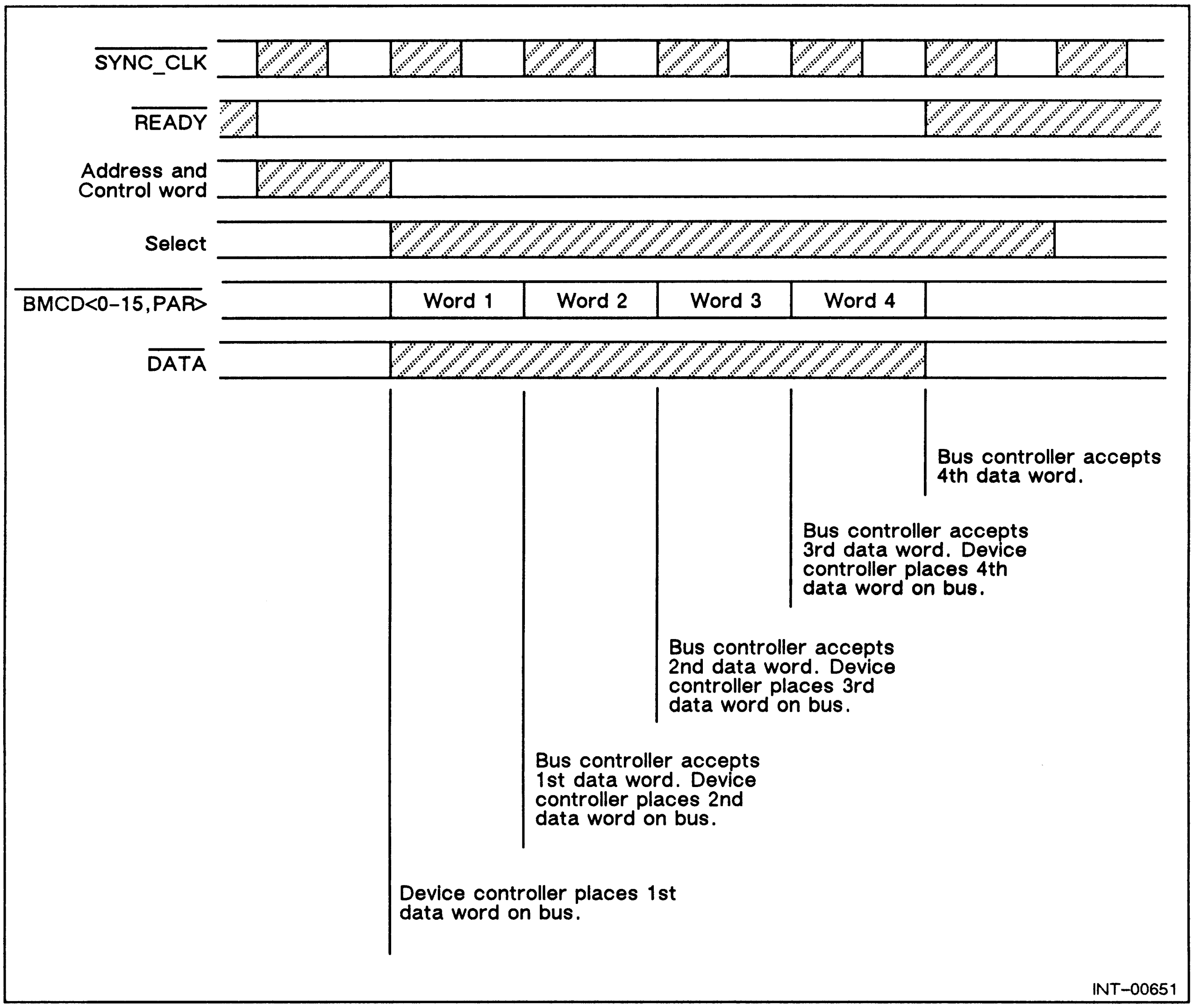

SYNC_CLK
READY
Address and Control word
Select
BMCD<0-15,PAR>
Word 1
Word 2
Word 3
Word 4
DATA
Bus controller accepts 4th data word.
Bus controller accepts 3rd data word. Device controller places 4th data word on bus.
Bus controller accepts 2nd data word. Device controller places 3rd data word on bus.
Bus controller accepts 1st data word. Device controller places 2nd data word on bus.
Device controller places 1st data word on bus.
INT–00651

Figure 4–11 BMC Data Input Sequence

Output

During data transfers out of memory, the BMC bus controller retrieves data words from memory, places the first data word on the BMC bus, and asserts ^DATA. Succeeding data words are placed on the bus at each high to low level transistion of the following ^SYNC_CLK signals when ^DATA is asserted. These actions continue until the requested count of data words has been transferred. The device controller must store a data word from the data lines on each high to low level transistion of ^SYNC_CLK following the BMC bus controller assertion of ^DATA. Figure 4-12 shows the timing sequence for a four-word output data transfer.

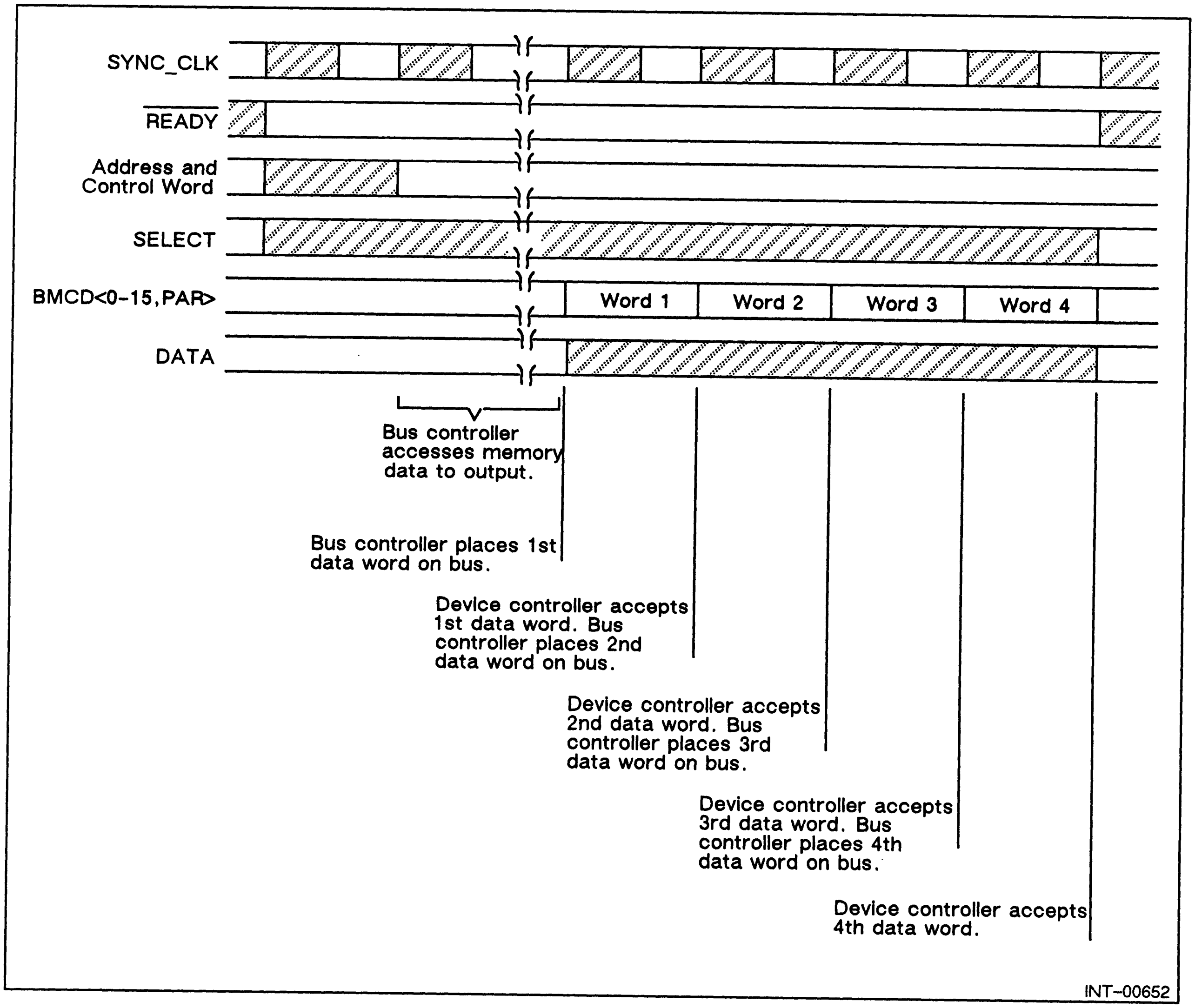

SYNC_CLK
READY
Address and Control Word
SELECT
BMCD<0-15,PAR>
DATA
Word 1
Word 2
Word 3
Word 4
Bus controller accesses memory data to output.
Bus controller places 1st data word on bus.
Device controller accepts 1st data word. Bus controller places 2nd data word on bus.
Device controller accepts 2nd data word. Bus controller places 3rd data word on bus.
Device controller accepts 3rd data word. Bus controller places 4th data word on bus.
Device controller accepts 4th data word.
INT-00652

Figure 4-12 BMC Data Output Sequence

Data Parity Check

If Parity Enable (^**BMC_ENB_PAR**) was specified in the channel control word received from the device controller, the BMC bus controller checks each data–in word (contents of ^**BMCD<0–15>** and ^**BMCD_PAR**) for odd parity. If a parity error is detected, the bus controller records the error in its Channel Definition register, notifies the device controller by asserting its ^**DATA_ERROR** signal line as shown in Figure 4–13, and continues the current transfer sequence writing the erroneous word(s) as if no error had occurred. The device controller must continue to finish its data burst transfer, error or not. When a data parity error is flagged, the device controller is responsible for remembering that the data error has occurred and for taking corrective action at the end of the transfer sequence.

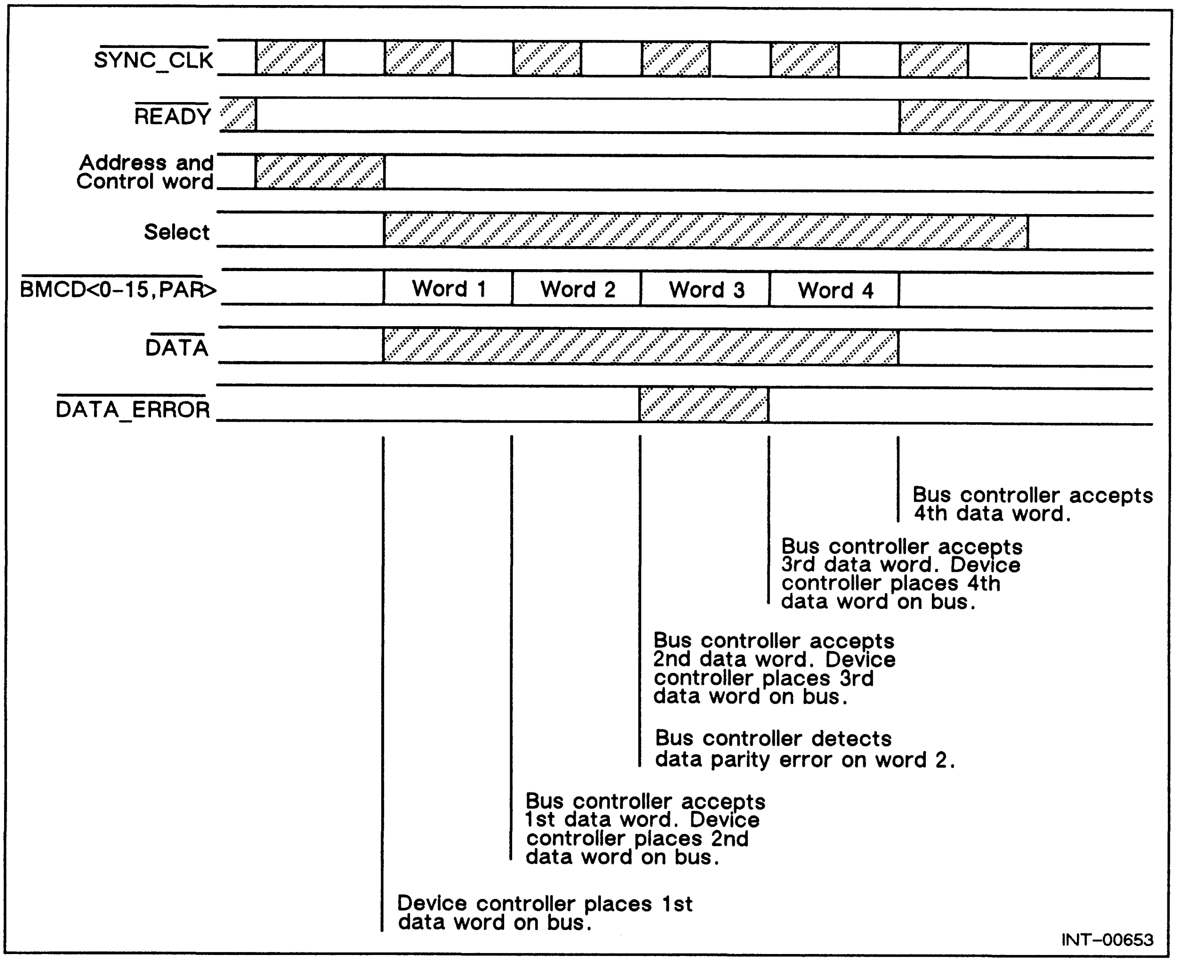

SYNC_CLK
READY
Address and Control word
Select
BMCD<0-15,PAR>
Word 1
Word 2
Word 3
Word 4
DATA
DATA_ERROR
Bus controller accepts 4th data word.
Bus controller accepts 3rd data word. Device controller places 4th data word on bus.
Bus controller accepts 2nd data word. Device controller places 3rd data word on bus.
Bus controller detects data parity error on word 2.
Bus controller accepts 1st data word. Device controller places 2nd data word on bus.
Device controller places 1st data word on bus.
INT-00653

Figure 4–13 Data Parity Error Detected on Input Transfer (Not on Last Word)

The BMC bus controller asserts ^**DATA_ERROR** during the BMC bus cycle that follows the cycle in which the erroneous data was received. If a data parity error occurs on the last word transferred in the data burst, the device controller will be notified of the parity error in the bus cycle that follows the data burst transfer as shown in Figure 4–14. Therefore, a device controller must remain selected for one additional bus cycle after transferring data to the BMC bus controller. In this case, the error may be reported during the address and channel control word transfer from another device controller using the BMC facility but will not affect the controller transferring the address and channel control word as it should not be monitoring for data parity errors at this time.

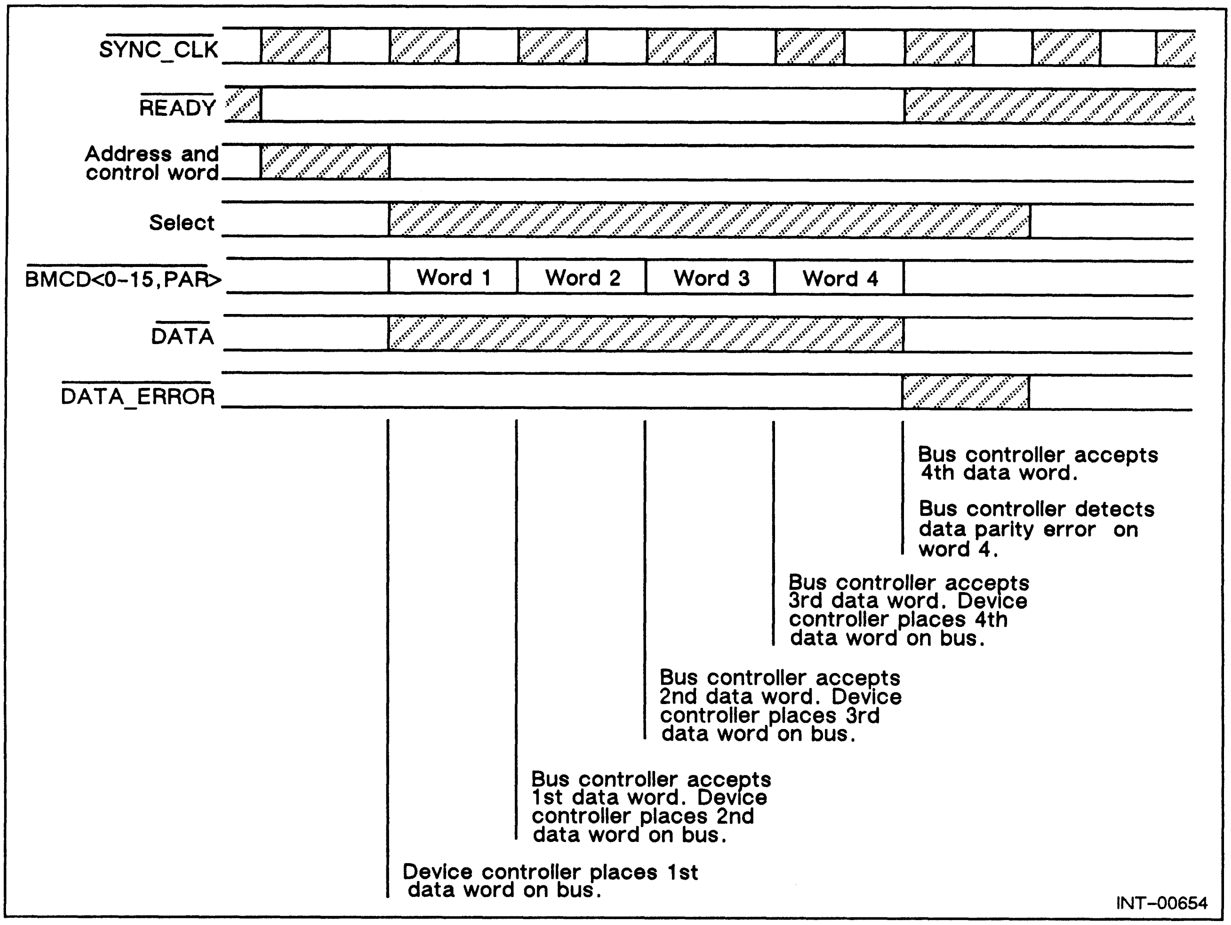

SYNC_CLK
READY
Address and control word
Select
BMCD<0-15,PAR>
DATA
DATA_ERROR
Word 1
Word 2
Word 3
Word 4
Bus controller accepts 4th data word.
Bus controller detects data parity error on word 4.
Bus controller accepts 3rd data word. Device controller places 4th data word on bus.
Bus controller accepts 2nd data word. Device controller places 3rd data word on bus.
Bus controller accepts 1st data word. Device controller places 2nd data word on bus.
Device controller places 1st data word on bus.
INT-00654

Figure 4–14 Data Parity Error Detected on Last Input Word Transferred

Data Parity Generation

The BMC bus controller always generates odd parity on data–out words, and the device controllers can either perform or disregard the parity check. If the device controller performs the parity check and an error is detected the controller must record the error and finish the data burst transfer before taking corrective action. The BMC bus controller is not notified of the data parity error and will continue to transfer the data burst as if no error occurred.

Protected Memory Check

If the Map Enable flag (^BMC_MAP) was specified in the channel control word received from the device controller, the BMC bus controller also checks for valid address translations (protected memory areas) during each memory access. If an attempt to access a protected memory area occurs during the course of a data burst transfer, the BMC bus controller records the error in its Channel Definition register, notifies the device controller by activating its ^**ADDR_ERROR** signal line as shown in Figure 4–15), and aborts the data transaction. If the bus controller flags a protected memory error, the device controller should neither transfer any more data words to the bus controller nor expect any more words from the bus controller for the current data transaction. The device controller is responsible for corrective action. If the bus controller flags a protected memory area error, it will allow the next data transaction to begin by asserting the ^**READY** signal line.

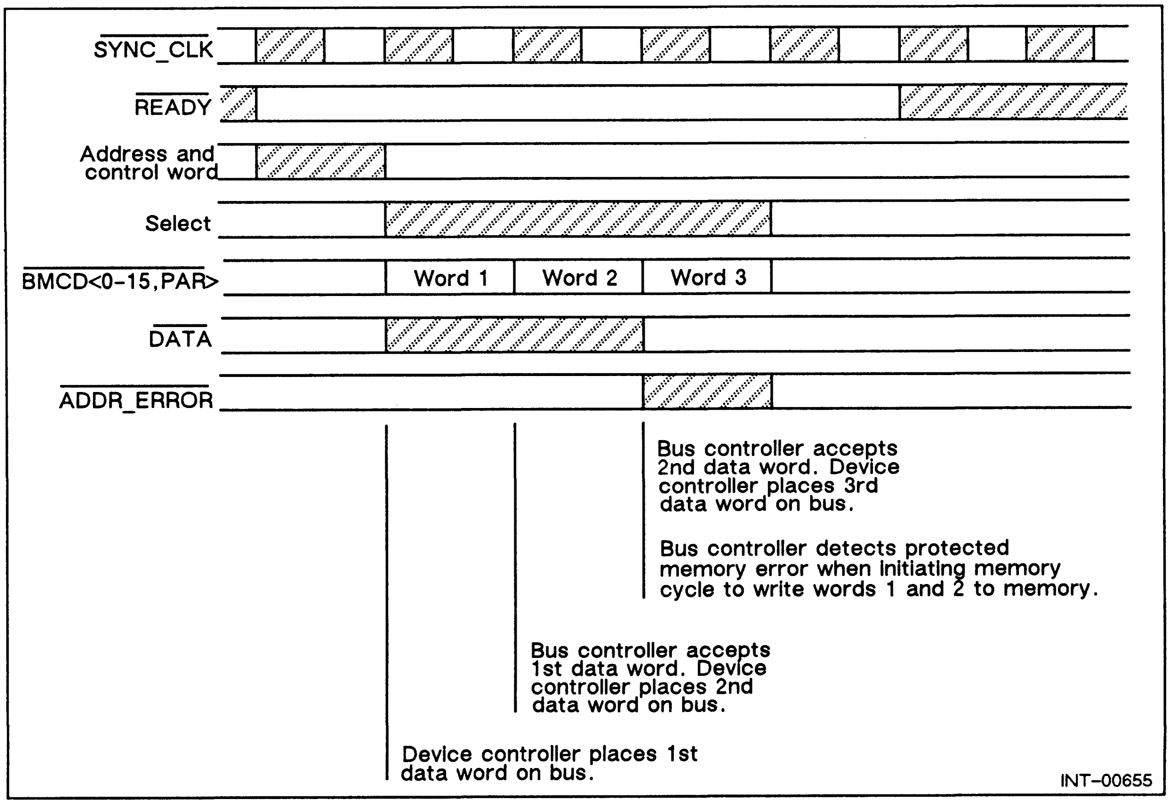

SYNC_CLK
READY
Address and control word
Select
BMCD<0-15,PAR>
Word 1
Word 2
Word 3
DATA
ADDR_ERROR
Bus controller accepts 2nd data word. Device controller places 3rd data word on bus.
Bus controller detects protected memory error when initiating memory cycle to write words 1 and 2 to memory.
Bus controller accepts 1st data word. Device controller places 2nd data word on bus.
Device controller places 1st data word on bus.
INT–00655

Figure 4–15 Protected Memory Error Detected On Input Data Transfer (Not On Last Word)

In a write–to–memory BMC operation, a protected memory error may occur as late as the BMC bus cycle that follows the cycle in which the last data word is transferred. If a protected memory error occurs on the last word transferred in the data burst, the device controller will be notified of the error in the bus cycle that follows the data burst transfer as shown in Figure 4–16. Therefore, a device controller must monitor for this type of error for one additional bus cycle after transferring data to the BMC bus controller. In this case the error may be reported during the address and channel control word transfer from another device controller using the BMC facility, but will not affect the controller transferring the address and channel control word as that controller should not be monitoring for protected memory errors until the next bus cycle.

In a BMC read–from–memory operation, the device controller would be notified of the protected memory error during a bus cycle in which a data word would have been transferred. Therefore, the device controller does not have to monitor for this type of error after the transfer completes.

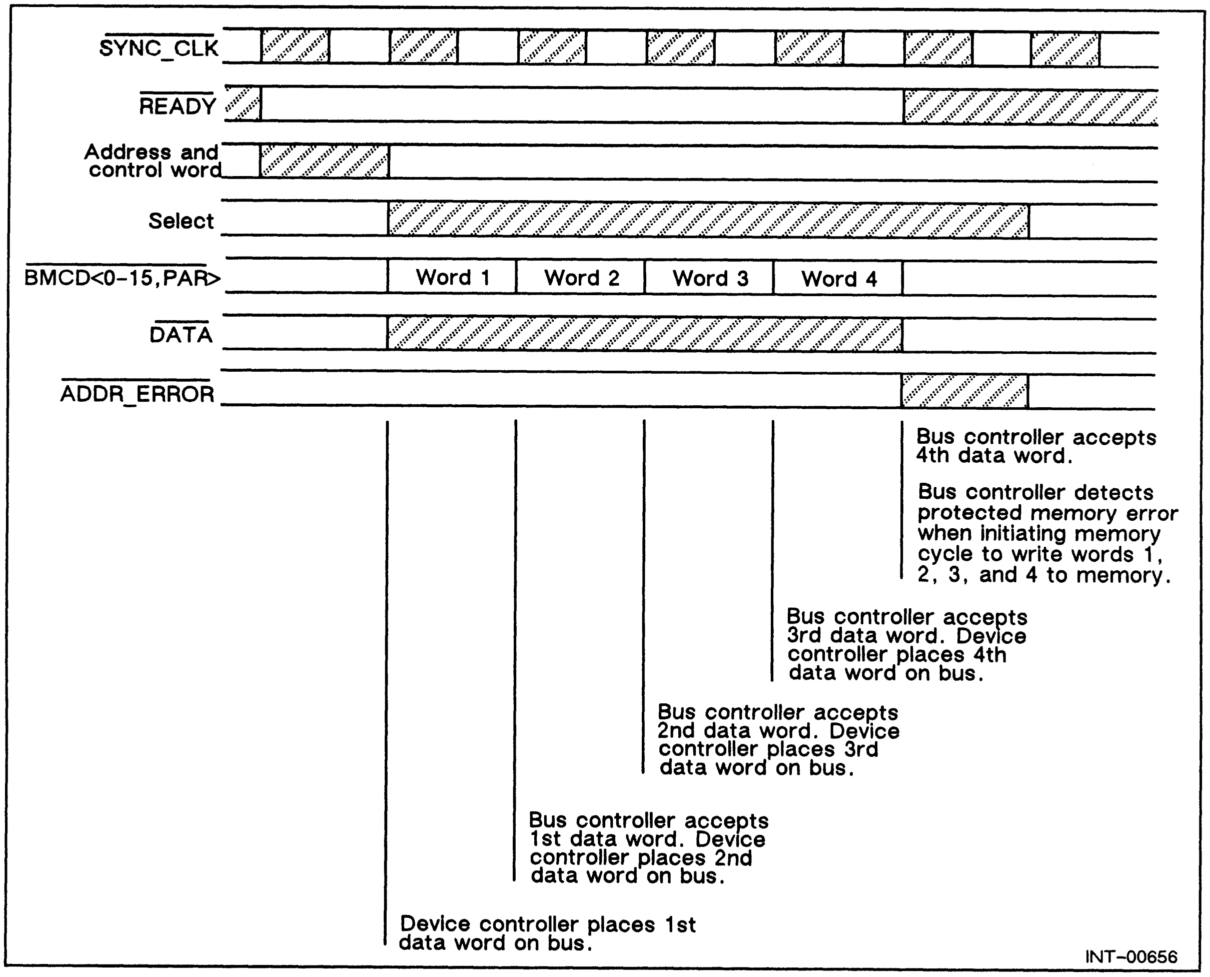

SYNC_CLK
READY
Address and control word
Select
BMCD<0-15,PAR>
Word 1
Word 2
Word 3
Word 4
DATA
ADDR_ERROR
Bus controller accepts 4th data word.
Bus controller detects protected memory error when initiating memory cycle to write words 1, 2, 3, and 4 to memory.
Bus controller accepts 3rd data word. Device controller places 4th data word on bus.
Bus controller accepts 2nd data word. Device controller places 3rd data word on bus.
Bus controller accepts 1st data word. Device controller places 2nd data word on bus.
Device controller places 1st data word on bus.
INT-00656

Figure 4–16 Protected Memory Error Detected Following Last Input Word Transferred

BMC Break Circuitry

Each device controller that implements the BMC facility should include circuitry to prevent back–to–back BMC facility transfer requests in order that lower priority BMC interfaces requiring service can periodically obtain use of the BMC facility. This can be accomplished by circuitry that counts a given number of ^SYNC_CLK pulses following each data burst transfer. The number of pulses can be selected at configuration time via a jumper or can be hardwired.

BMC Map Table

Because the BMC bus controller accesses memory directly, it must contain its own separate Map table that translates 20-bit logical memory addresses, received from device controllers, to physical memory addresses. The BMC map table contains 1,024 map slots, one map slot for each logical page. These map slots are located within the BMC map table portion of the Channel Input/Output (CIO) registers (numbered 0000 through 3777_8). Each map slot consists of an even-numbered register and its corresponding odd-numbered register as shown in Figure 4-17.

The BMC bus controller translates memory addresses by using the logical page number to select a map slot in the map table and then combining the physical page number, contained in that map slot, with the page offset address supplied by the device controller to form the physical memory address.

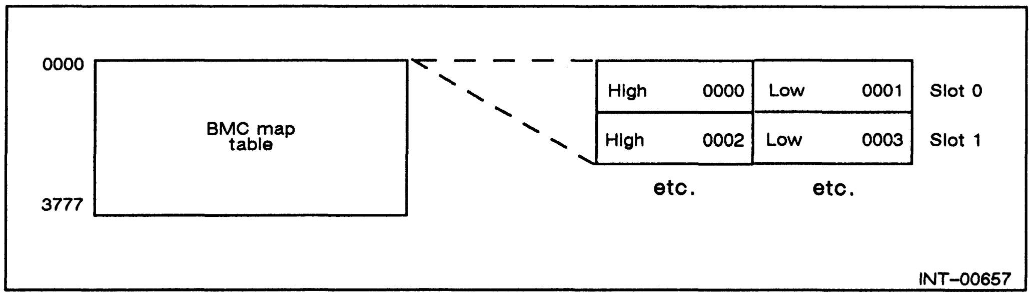


Figure 4-17 BMC Memory Map Register Locations

Loading BMC Map from I/O Device Controller

The BMC map is normally loaded with translation data transferred from main memory using programmed instructions (*WLMP, CIO,* and *CIOI*). However, some BMC device controllers have the ability to load their BMC map slots with translation data. (For programming details on loading the BMC Map, refer to the *ECLIPSE MV/Family (32-Bit) Systems Principles of Operation* manual as well as the *Principles of Operation Supplement* for your specific system.)

Note that the specified BMC map must be enabled for upstream loading to be successful. This is accomplished if the BMC upstream map loading function is implemented by the device controller and that operation is specified (**^EXT** and **^BMC_DATA_IN** signal lines are both active) in the channel control word received from the device controller.

Device controllers that implement the BMC map upstream load function can address any one of 1,024 map slots. They do this by asserting a 10-bit map slot number. Address word format for BMC map upstream loading by a device controller is shown in Figure 4-18. Note that **^BMC_ADDR20** of the address word defines which register (high-order or low-order) receives the first data word that is transferred.

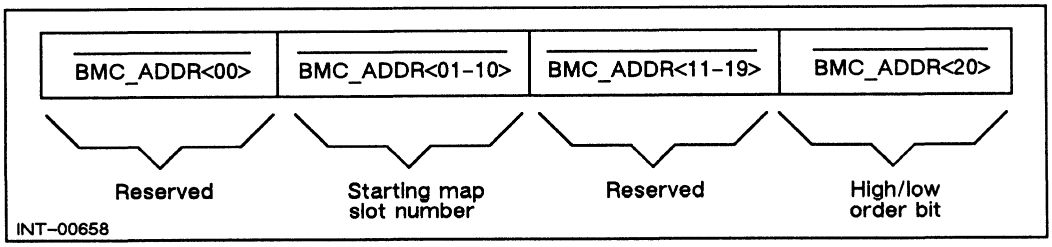


Figure 4-18 Address Word Format for BMC Map Upstream Loading

In the upstream load function, the device controller performs a BMC map write operation that writes a specified number of 16-bit words into map slot registers, beginning with a specified register. The peripheral controller specifies the number of map slot registers to write and the beginning register address. Note that two words are required to load each map slot: one for the high-order register, and one for the low-order register. To load a number of complete map slots, the beginning register address must be even and an even number of 16-bit words must be specified. The data word formats to be written are shown in Figure 4-19 and Figure 4-20.

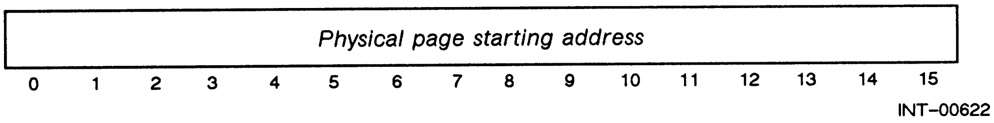
V	Z	*Reserved (Should be zeros)*													
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

INT-00659

Bits	Mnemonic	Field Name	Contents or Function
0	V	Valid page	Specifies whether a page can be accessed 0 = page access enabled 1 = page access denied [1]
1	Z	Data transfer	Specifies whether data or zeros are transferred 0 = data is transferred 1 = zeros are transferred
2-15	—	—	Hardware reserved; returns zeros when read, should be written with zeros.

[1] If access is attempted, a MAP validity fault condition is generated.

Figure 4-19 BMC Map Even-Numbered Register Format (High-Order Word)



Bits	Mnemonic	Field Name	Contents or Function
0-15	—	Physical page starting address	Starting physical memory address of logical page reference for the particular map slot.

Figure 4-20 BMC Map Odd-Numbered Register Format (Low-Order Word)

End of Chapter

Chapter 5
BMC Bus Specifications

This chapter presents functional timing characteristics and electrical specifications for the various BMC bus functions that were previously described in Chapter 4. The chapter also describes how to connect to and terminate the BMC bus.

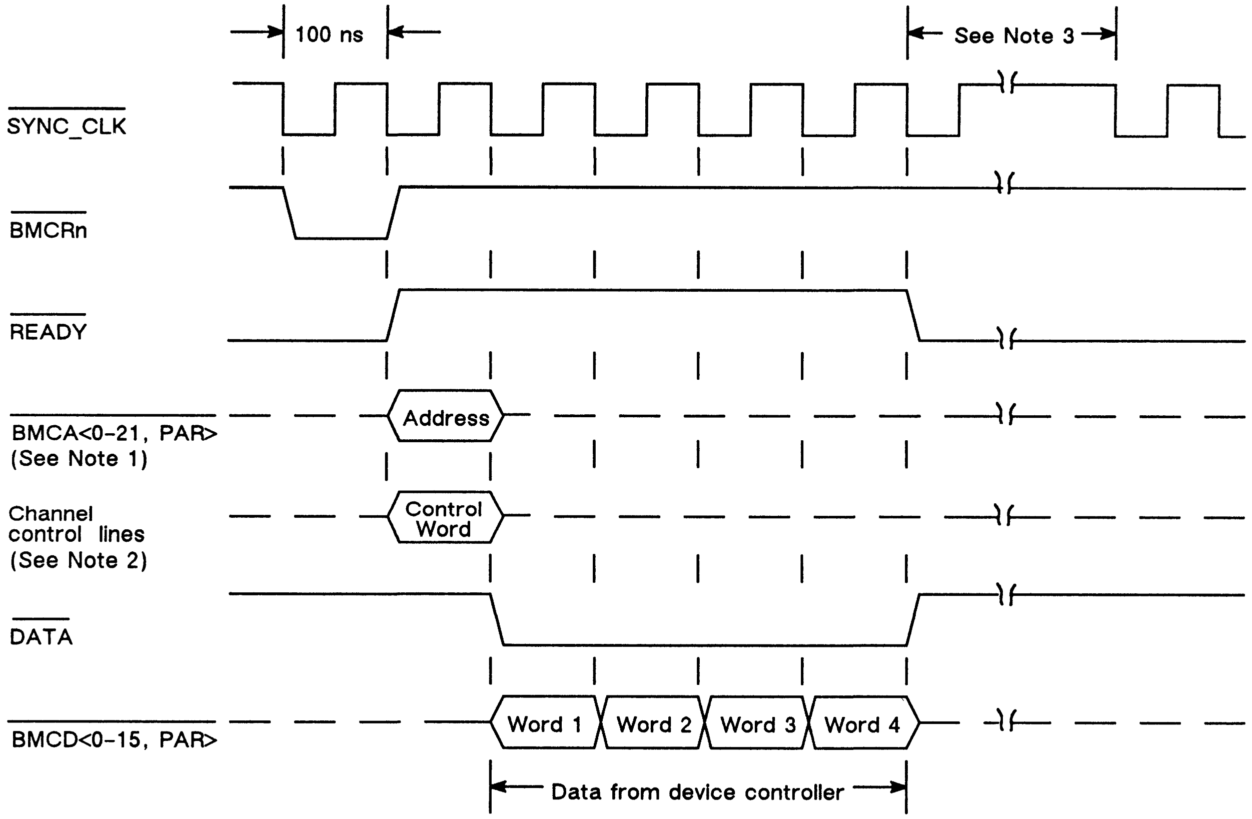
Timing

The following timing diagrams, Figures 5–1 through 5–19, show the characteristic timing for various BMC bus functions including those with specified error conditions. The period for ^**SYNC_CLK** in the timing diagrams is shown as 100 ns, which is the minimum time (95 ns when margined at 55°C) BMC device controllers must be designed to operate properly.

There are four timing specifications in the BMC facility, and they are relative to the channel synchronization clock signal (^**SYNC_CLK**).

1. All signals asserted by the BMC bus controller will arrive at the device controller's BMC interface at least 38 ns before the next clocking edge (transistion from high to low level) of ^**SYNC_CLK**.
2. Any signal asserted by the device controller's BMC interface must be asserted within 42 ns of the clocking edge (transistion from high to low level) of ^**SYNC_CLK**.
3. The nonclocking edge (transistion from low to high level) of ^**SYNC_CLK** will occur at a time greater than +/–40 ns relative to the clocking edge (transistion from high to low level).
4. Hold time on all signals should be 10 ns.

A device controller whose BMC interface was designed to operate with any ECLIPSE or ECLIPSE MV/Family computer will in most cases operate with another ECLIPSE MV/Family computer system.



Notes: 1. Address on 4-word boundary (2 LSBs = 00)

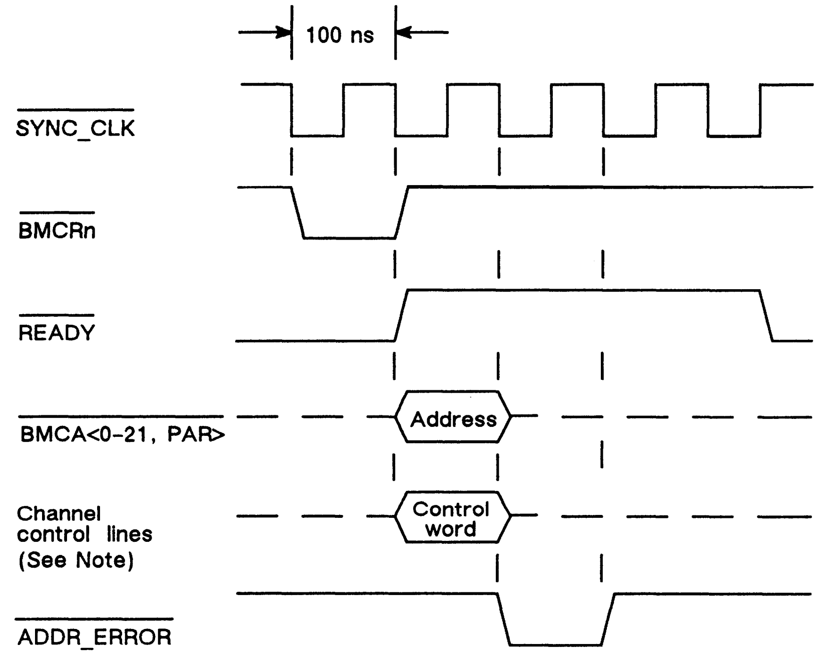
2. Channel control lines consist of the following signals from the device controller:

BMC DATA IN
WCNT<0-7>
BMC ENB PAR
BMC MAP
EXTEND

3. Time required for BMC bus controller to access memory and store 4 words.

INT-00661

Figure 5-1 Data Input (1 Request, 4 Words, No Errors)

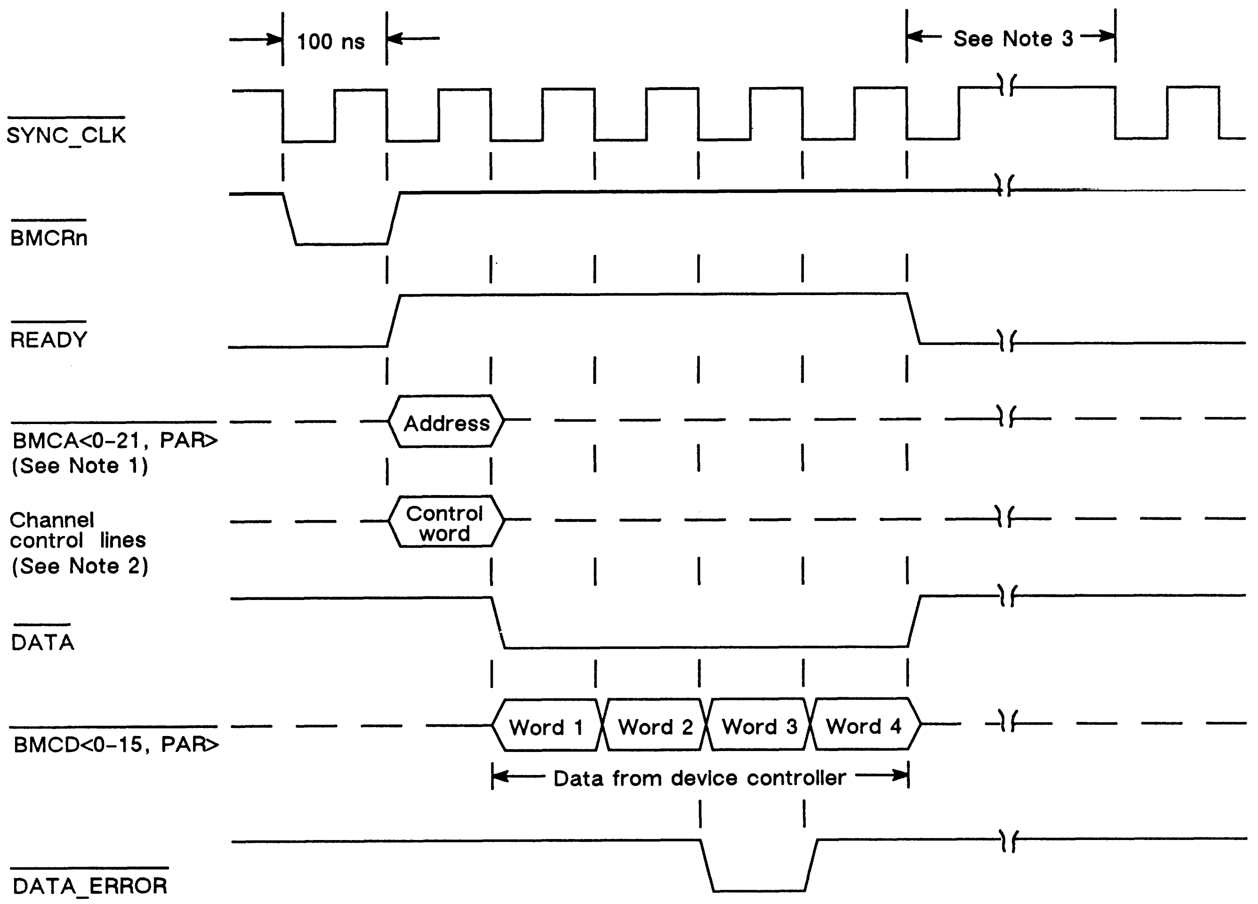


Note: Channel control lines consist of the following signals from the device controller:

$\overline{\text{BMC DATA IN}}$
$\overline{\text{WCNT<0-7>}}$
$\overline{\text{BMC ENB PAR}}$
$\overline{\text{BMC MAP}}$
$\overline{\text{EXTEND}}$

INT-00662

Figure 5-2 Address Parity Error (Input or Output)



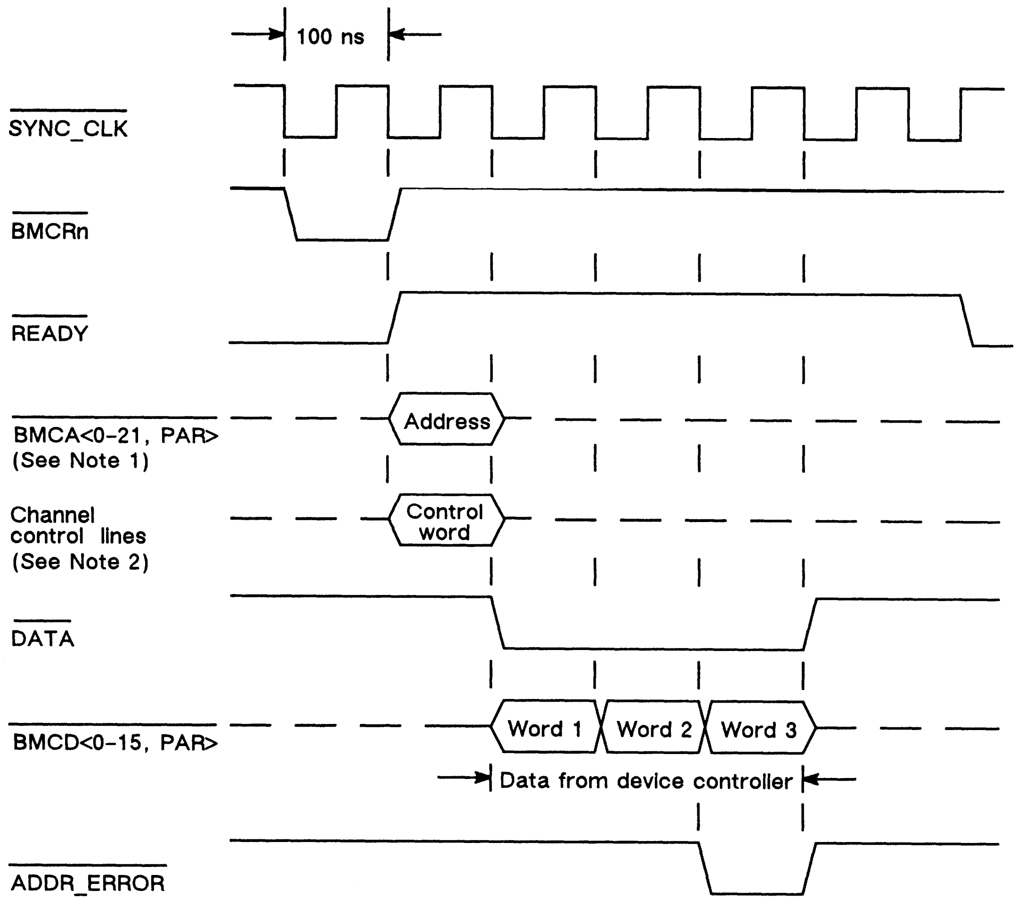
Notes:
1. Address on 4-word boundary (2 LSBs = 00)
2. Channel control lines consist of the following signals from the device controller:

   BMC DATA IN
   WCNT<0-7>
   BMC ENB PAR
   BMC MAP
   EXTEND

3. Time required for BMC bus controller to access memory and store 4 words.

INT-00663

Figure 5-3 Data Input (1 Request, 4 Words, Data Parity Error)



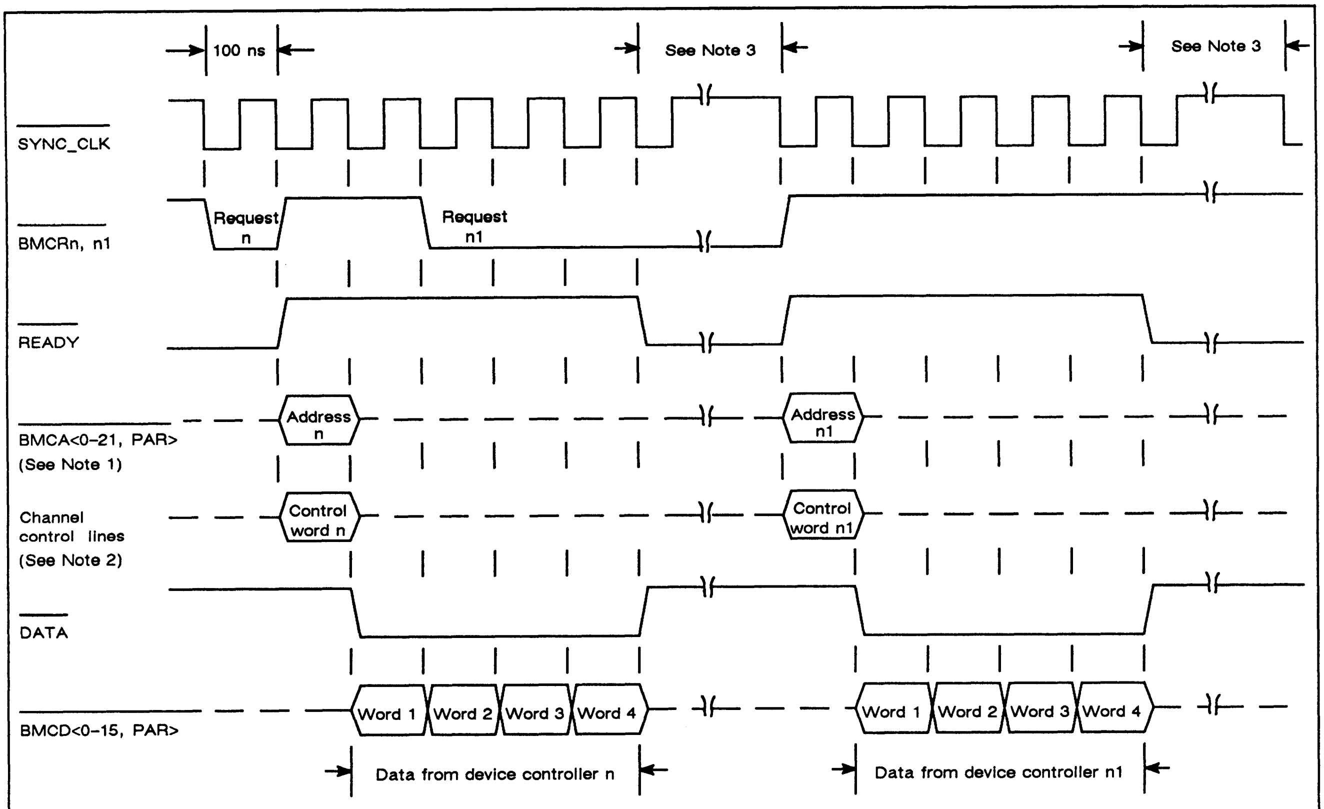
Notes:
1. Address on 2-word boundary (2 LSBs = 00)
2. Channel control lines consist of the following signals from the device controller:

   $\overline{\text{BMC DATA IN}}$
   $\overline{\text{WCNT<0-7>}}$
   $\overline{\text{BMC ENB PAR}}$
   $\overline{\text{BMC MAP}}$
   $\overline{\text{EXTEND}}$

3. Data is not stored in memory.

INT-00664

Figure 5-4 Data Input (1 Request, 4 Words, Protected Memory Error)



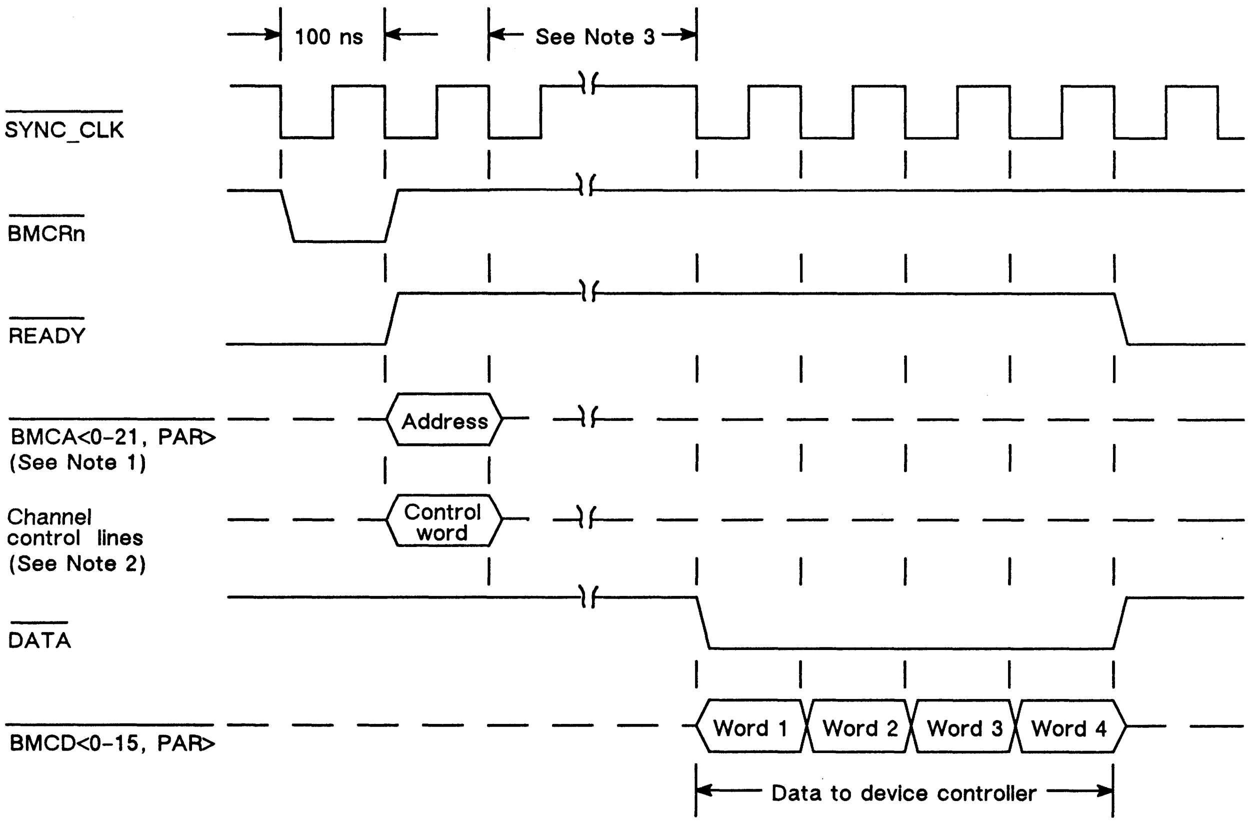
Notes:

1. Both addresses on 4-word boundary (2 LSBs = 00)
2. Channel control lines consist of the following signals from the device controller:

   BMC DATA IN
   WCNT<0-7>
   BMC ENB PAR
   BMC MAP
   EXTEND

3. Time required for BMC bus controller to access memory and store 4 words.

Figure 5-5 Data Input (2 Requests, 4 Words Each, No Errors)



Notes: 1. Address on 4-word boundary (2 LSBs = 00)

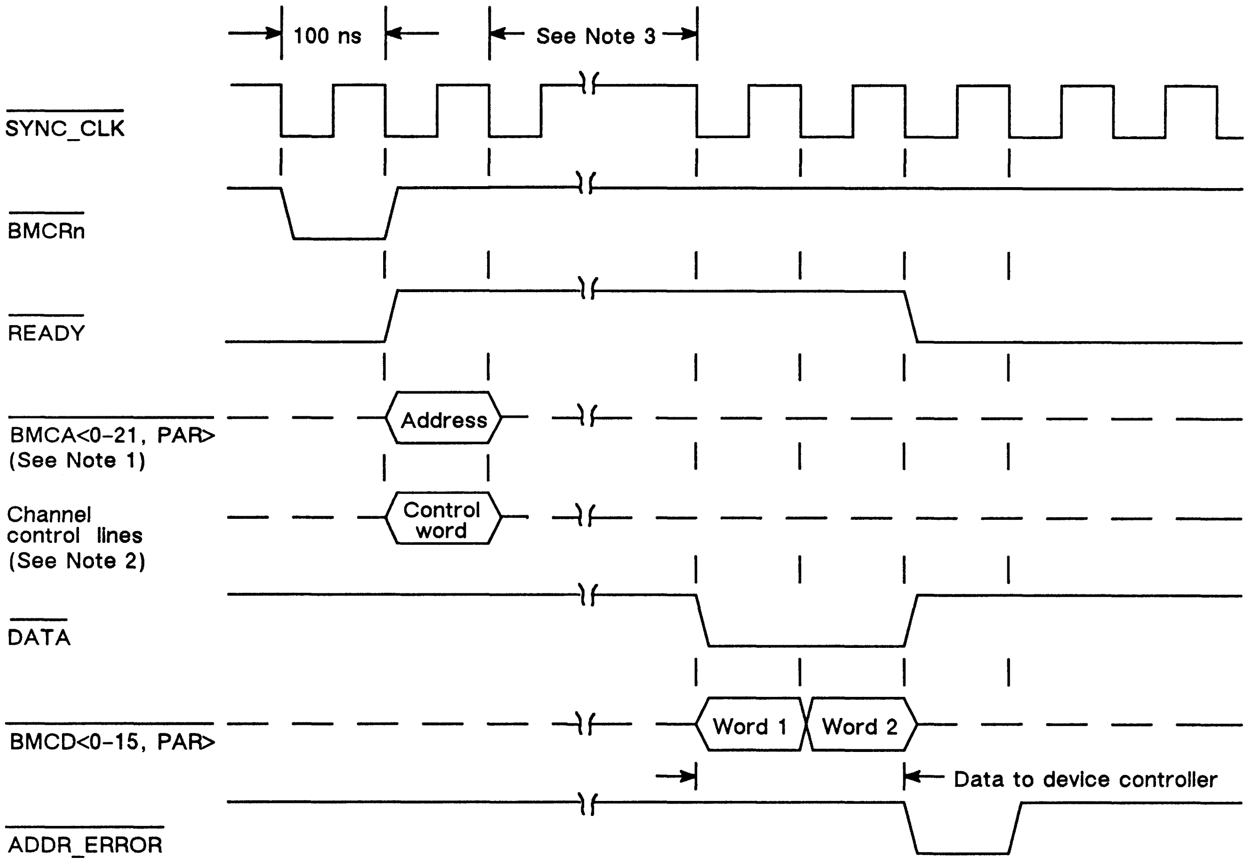
2. Channel control lines consist of the following signals from the device controller:

BMC DATA IN
WCNT<0-7>
BMC ENB PAR
BMC MAP
EXTEND

3. Time required for BMC bus controller to access memory and retrieve 4 words.

INT-00666

Figure 5-6 Data Output (1 Request, 4 Words, No Errors)



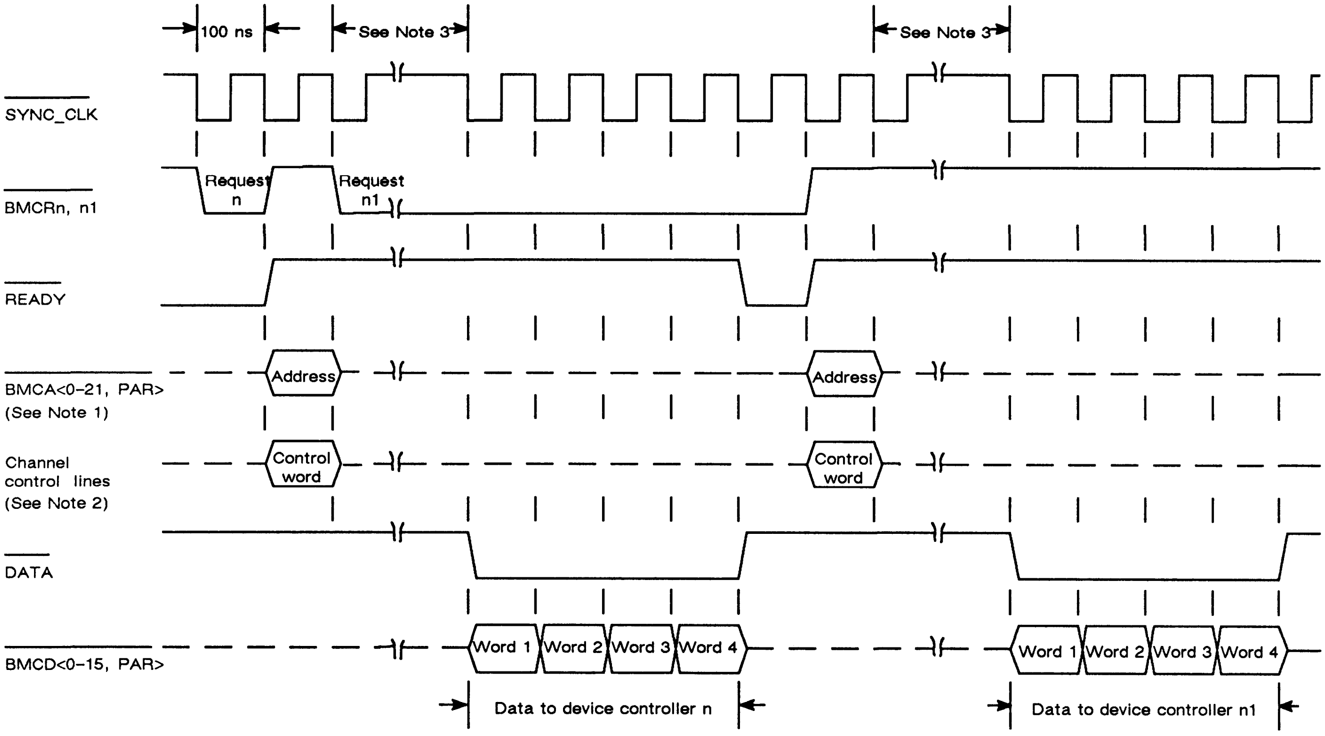
Notes:
1. Address on 2-word boundary (2 LSBs = 00)
2. Channel control lines consist of the following signals from the device controller:

   $\overline{\text{BMC DATA IN}}$
   $\overline{\text{WCNT<0-7>}}$
   $\overline{\text{BMC ENB PAR}}$
   $\overline{\text{BMC MAP}}$
   $\overline{\text{EXTEND}}$

3. Memory is not accessed for words 3 and 4.

INT-00667

Figure 5-7 Data Output (1 Request, 4 Words, Protected Memory Error)



Notes:

1. Both addresses on 4-word boundary (2 LSBs = 00)

2. Channel control lines consist of the following signals from the device controller:

   BMC DATA IN

   WCNT<0-7>

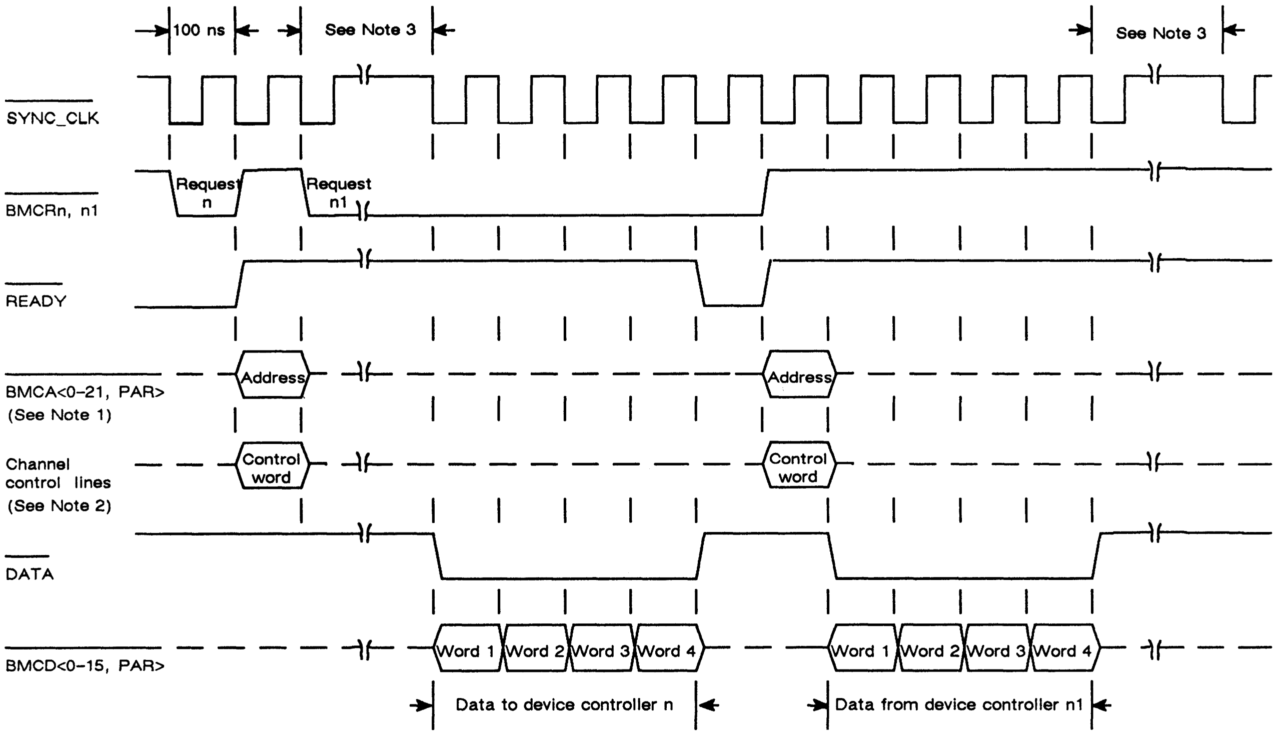
   BMC ENB PAR

   BMC MAP

   EXTEND

3. Time required for BMC bus controller to access memory and retrieve 4 words.

Figure 5–8 Data Output (2 Requests, 4 Words Each, No Errors)



Notes:

1. Both addresses on 4-word boundary (2 LSBs = 00)
2. Channel control lines consist of the following signals from the device controller:

   BMC DATA IN
   WCNT<0-7>
   BMC ENB PAR
   BMC MAP
   EXTEND
3. Time required for BMC bus controller to access memory and retrieve 4 words.

Figure 5-9 Two Requests (1st Output; 2nd Input, 4 Words Each, No Errors)

Signal Levels

All of the signals on the ECLIPSE BMC bus are digital transistor–transistor logic signals and thus have two electrical states. As shown in Figure 5–10, any voltage between 0 and +0.7 volts is considered low while any voltage greater than +2.0 volts is considered high. The normal voltage for low is 0.5 volt. The normal voltage for high ranges from +2.7 to +3 volts. The relation between the electrical level of a signal and its logical value depends on the particular signal in question. In the following figures, signals that are asserted when low are identified by a bar over the signal name (for example, $\overline{\text{CLR}}$ as explained in the Preface of this manual.

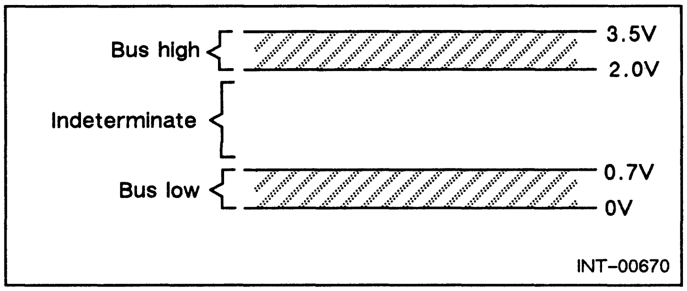


Figure 5–10 Bus Electrical Levels

Electrical Specifications

The BMC bus will support one receiver and one driver per bus signal for up to eight BMC device controllers. The receiver can be a Schottky or other TTL load; but it must not be the high capacity PNP input type. The driver must be an open collector type and must be capable of sinking at least 64 ma at 0.5 volts.

Follow the suggestions listed below when designing a device controller that connects to the BMC bus:

- Keep all BMC bus signals very short. That is, locate the receivers, drivers, and bus terminators for BMC bus signals physically close to the printed circuit boards (PCBs) front edge J–connectors where the signals connect to the the BMC bus ribbon cables.
- Lay out all BMC bus signal etch to minimize capacitance and crosstalk.
- Use only one receiver for each signal. That is, if a signal is used at more than one point on a single board or device controller interface assembly, that board should nevertheless cause only one load on the bus for that signal.
- Treat these signals as transmission lines; their physical as well as electrical characteristics are important.
- Consider the effect on signal delays and pulse widths caused by bus receivers, transmitters, repeaters, and filters.

BMC Bus Connections and Termination

The BMC bus signals are carried by two flat-ribbon cables that connect the BMC bus to device controllers that use the BMC facility. These cables originate from two 40-pin connectors on the front edge of the printed circuit board containing the BMC bus controller and daisy chain through two 40-pin connectors on the front edge of each device controller connected to a specific BMC bus controller. The BMC bus must be terminated at the device controller that is farthest from the BMC bus controller on the chain.

BMC Bus Connectors

Figure 5-11 shows the pin layout of the two 40-pin front edge connectors used to connect the BMC bus to the device controller printed circuit board.

BMC Bus Termination

All BMC bus bidirectional signals and signals received by the device controllers must be terminated by the device controller located at the end of the BMC bus farthest from the BMC bus controller. The following signals must be terminated:

^SYNC_CLK

^BMCR<0-7>

^READY

^DATA

^ADDR_ERROR

^DATA_ERROR

^BMC_DATA<0-15>

^BMC_DPAR

Bus termination can be accomplished by generating a +3 volt mass termination bus on each device controller that contains a BMC interface and connecting the termination bus to a third 40-pin front-edge J-connector. Each BMC bus signal that requires termination is also connected through a 150 ohm resistor to the same J-connector. Then a 40-pin plug can be wired that connects all of the BMC bus signals that are to be terminated to the +3 volt termination bus. This plug would then be installed only on the front-edge J-connector of the device controller that is located farthest from the BMC bus controller on the BMC bus. Data General Corporation makes available a 40-pin plug for this purpose that connects all odd-numbered pins and all even-numbered pins together (DGC part number 005-013419).

An alternative method of terminating the BMC bus can be accomplished by connecting each BMC bus signal that must be terminated by a jumper through a 150 ohm resistor to the +3 volt termination bus. The jumpers would then be installed only on the device controller that is located farthest from the BMC bus controller on the BMC bus.

J1 Connector

Pin	Signal	Signal	Pin
1	GND	BMCR0	2
3	BMCR1	BMC_DATA3	4
5	BMCR2	GND	6
7	BMCR3	BMC_DATA2	8
9	BMCR4	BMC_DATA9	10
11	BMCR5	GND	12
13	BMCR6	BMC_DATA8	14
15	BMCR7	BMC_DATA1	16
17	BMC_DATA7	GND	18
19	BMC_DATA11	BMC_DATA4	20
21	BMC_DATA10	BMC_DATA5	22
23	BMC_DATA6	GND	24
25	BMC_DPAR	BMC_DATA0	26
27	DATA_ERROR	GND	28
29	ADDR_ERROR	GND	30
31	SYNC_CLK	GND	32
33	DATA	GND	34
35	BMC_DATA15	GND	36
37	READY	BMC_DATA14	38
39	BMC_DATA12	BMC_DATA13	40

J4 Connector

Pin	Signal	Signal	Pin
1	GND	BMC_ADDR2	2
3	BMC_ADDR5	BMC_ADDR1	4
5	BMC_ADDR6	GND	6
7	BMC_ADDR8	BMC_ADDR3	8
9	BMC_ADDR7	BMC_ADDR4	10
11	WCNT4	EXTEND	12
13	WCNT5	BMC_DATA_IN	14
15	WCNT6	BMC_ADDR0	16
17	WCNT7	GND	18
19	BMC_ADDR12	BMC_MAP	20
21	BMC_ADDR11	BMC_ENAB_PAR	22
23	BMC_ADDR10	GND	24
25	BMC_ADDR9	BMC_APAR	26
27	BMC_ADDR17	WCNT1	28
29	BMC_ADDR18	GND	30
31	BMC_ADDR20	WCNT2	32
33	BMC_ADDR19	WCNT3	34
35	BMC_ADDR16	GND	36
37	BMC_ADDR15	WCNT0	38
39	BMC_ADDR14	BMC_ADDR13	40

INT-00575

Figure 5-11 BMC Bus Connector Pin Layout

End of Chapter

Chapter 6
Device Connections and Connectors

This chapter describes how to connect device controllers in an ECLIPSE MV/Family computer chassis to adapters and devices outside the chassis and how to install internal and external I/O cables.

Cables and Connections

Device controllers must be connected both to the I/O bus and to the device it controls. In addition, device controllers that interface to the BMC bus must also connect to that bus.

The I/O bus transfers command and status information and data between the I/O bus controller and the device controller. The information or data that is transferred can originate at the central processor unit, memory, or the device controller. The I/O bus signals are carried by the etching on the chassis' printed circuit backpanel (backplane). The ECLIPSE I/O bus can be connected to an expansion chassis when system configurations require expansion capability. A bus repeater printed circuit board is required when you connect the I/O bus to an expansion chassis. The I/O bus signals are terminated by resistor networks solder-mounted on the chassis' backpanel.

The BMC bus transfers data between the BMC bus controller and device controllers. The data that is transferred can originate at the memory or the device controller. The BMC bus signals are carried by two flat-ribbon cables that connect the BMC bus controller to device controllers that use the BMC facility. These cables originate from two 40-pin connectors on the front edge of the board containing the BMC bus controller and daisy chain through two 40-pin connectors on the front edge of each device controller connected to a specific BMC bus controller. The BMC bus chain must be terminated at the device controller that is farthest from the BMC bus controller on the chain.

The device controller connects to a device or an adapter through a series of cables and connectors. These connections are made in two parts: first through an internal cable, and then through an external (or device) cable. The internal cable brings the signals from the device controller to the rear connector panel. The external cable takes the device signals from the connector panel directly to the device or, in some cases, to a connection junction box and then to the device(s).

There are several styles of internal device cables. These cables have one or two 100-pin backpanel connectors on one end that push onto pins of the backpanel. These pins are extensions of the contacts in the two 100-pin female edge connectors into which the device controller printed circuit board is inserted. The other end of the internal device cables have 1, 2, or 4 sub-D connectors that fasten to the rear connector panel of the chassis. The sub-D connectors have 25, 37, or 50 pins. The sub-D connectors of internal device cables interconnect with sub-D connectors at one end of external device cables. Figure 6-1 shows a typical internal device cable. (See the section "Cabling to User Designed and Built Interfaces" later in this chapter for a list of internal device cables available from Data General Corporation.)

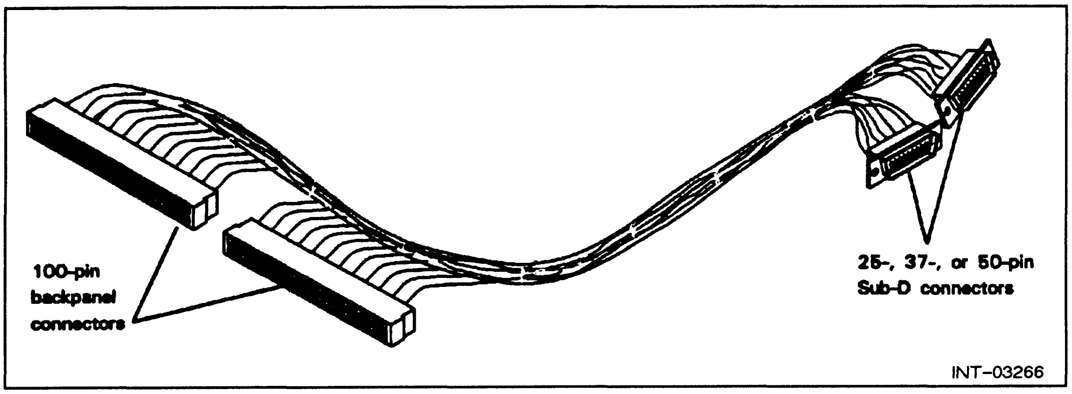


Figure 6–1 Typical Internal Device Cable

The external device cable is typically a multiwire round (or bundled) cable. This cable directly connects to the device (for example, a disk subsystem). One end of external device cables contains a sub–D connector that interconnects with the internal device cable at the chassis' rear connector panel; the other end contains the appropriate connector for the device or adapter.

Backpanel Connections

Input/output connections are made to a device controller board in an ECLIPSE chassis via pins on the backpanel. These pins are extensions of the contacts in the two 100–pin female edge connectors into which the device controller printed circuit board is inserted. The pins in each connector are arranged in two rows of 50 pins each, one row making contact with fingers on the component side of the board and the other row making contact with fingers on the opposite side of the board.

Backpanel pins within a slot are designated by a letter indicating one of the two female edge connectors and a two–digit, decimal number indicating the position of the pin within the connector. The letters A and B designate the female connectors. The odd–numbered pins make contact with the component side of the board, and the even–numbered pins with the opposite side.

I/O Bus Connections

The I/O bus carries signals that contain very high frequency components in their rising and falling edges. Any cable carrying these signals must be treated as a high–frequency transmission line. Within the computer chassis itself, the I/O bus is carried via etching on the backpanel; and line lengths are short enough so that the propagation times are low compared to the rise time of the signals. If it is necessary to bring the I/O bus out of the chassis via a cable, then line lengths are extended so that reflections, settling times, and crosstalk become significant problems.

Whenever possible, a device controller should be inserted within the main or expansion chassis. The next two chapters describe prefabricated boards available from Data General Corporation that allow the user to design and build custom device controllers that insert into the chassis. When the device controller must be mounted outside these chassis, however, it should be connected to the I/O bus via the 95 ohm, twisted pair I/O bus cable sold by Data General Corporation. The total length of the I/O bus must never exceed 50 feet, including etching and wires within chassis.

I/O Bus Connections Within a Chassis

Minimal I/O bus cabling is required when a device controller is installed in an ECLIPSE chassis because most I/O bus signals are carried to every slot available for I/O device controllers via etching on the backpanel. Some connections, however, may be required to maintain the integrity of the bus priority signals **^INTP** and **^DCHP**.

The bus priority signals **^INTP** and **^DCHP** are chained from one device controller to the next, and jumpers are required to carry these signals across any unused input/output printed circuit board slot or user-designed and built board that does not properly pass these signals along the bus. (See Chapter 7 for bus priority signal details.) The signal **^INTP** is jumpered across a slot by connecting pins A95 and A96 of that slot. The signal **^DCHP** is jumpered across a slot by connecting pins A93 and A94 of that slot.

I/O Bus Connections Outside a Chassis

Device controllers that are not mounted in an ECLIPSE system chassis must be connected to the I/O bus by an internal and external I/O bus cable in the same fashion as device controllers connect to devices. For detail see the preceding sections in this chapter. Figure 6-2 shows which backpanel pins are used to connect to the I/O bus.

Cabling to an Adapter or Device

A device controller inserted in the ECLIPSE chassis connects to an adapter or device outside the chassis by an internal device cable and an external device cable. The internal device cable connects pins in one slot on the backpanel with pins in a sub-D connector that mounts on the rear connector panel of the chassis. One end of the external device cable is then plugged into that sub-D connector, and the other end is plugged into the adapter or device. Figure 6-2 shows which backpanel pins are used to connect to the I/O bus and which pins are available to connect to external devices or adapters.

Cabling to User-Designed and Built Interfaces

When an internal device cable must be installed for a user-designed interface, it is recommended that one of the cables listed in Table 6-1 be used. These cables are general-purpose, internal device cables. The cables are compatible with many standard Data General Corporation interfaces but they can also be used for user-designed interfaces. They are supplied with 100-pin backpanel connector(s) and sub-D connector(s) as specified in the table. The correlation between backpanel pins and sub-D connector pins for these cables can be found in the engineering drawing specified in the table.

A Connector

Pin	Signal	Signal	Pin
2	GND	GND	1
4	+5V	+5V	3
6	-5V	User	5
8	User	User	7
10	Reserved	User	9
12	User	User	11
14	Reserved	User	13
16	User	User	15
18	User	User	17
20	User	User	19
22	User	User	21
24	User	User	23
26	User	User	25
28	User	User	27
30	User	User	29
32	User	User	31
34	GND	User	33
36	User	User	35
38	$\overline{\text{MSKO}}$	User	37
40	INTA	User	39
42	DATIB	User	41
44	DATIA	User	43
46	$\overline{\text{DS3}}$	User	45
48	DATOC	User	47
50	CLR	User	49
52	STRT	User	51
54	DATIC	User	53
56	DATOB	User	55
58	DATOA	User	57
60	$\overline{\text{DCHA}}$	User	59
62	$\overline{\text{DS4}}$	User	61
64	$\overline{\text{DS5}}$	User	63
66	$\overline{\text{DS2}}$	User	65
68	$\overline{\text{DS1}}$	User	67
70	IORST	User	69
72	$\overline{\text{DS0}}$	User	71
74	IOPLS	User	73
76	User	User	75
78	User	User	77
80	$\overline{\text{SELD}}$	User	79
82	$\overline{\text{SELB}}$	User	81
84	User	User	83
86	User	User	85
88	User	User	87
90	User	User	89
92	User	User	91
94	$\overline{\text{DCHPIN}}$	$\overline{\text{DCHPOUT}}$	93
96	$\overline{\text{INTPIN}}$	$\overline{\text{INTPOUT}}$	95
98	+5V	+5V	97
100	GND	GND	99

B Connector

Pin	Signal	Signal	Pin
2	GND	GND	1
4	+5V	+5V	3
6	User	User	5
8	Reserved	Reserved	7
10	Reserved	Reserved	9
12	User	User	11
14	User	User	13
16	User	User	15
18	User	$\overline{\text{DCHMO}}$	17
20	User	User	19
22	User	$\overline{\text{PWRFAIL}}$	21
24	User	User	23
26	User	User	25
28	User	User	27
30	User	$\overline{\text{INTR}}$	29
32	User	User	31
34	User	DCHO	33
36	User	$\overline{\text{DCHR}}$	35
38	User	DCHI	37
40	User	PWROK	39
42	User	$\overline{\text{RQENB}}$	41
44	User	User	43
46	User	User	45
48	User	User	47
50	User	User	49
52	User	User	51
54	User	User	53
56	$\overline{\text{DATA14}}$	$\overline{\text{DATA7}}$	55
58	$\overline{\text{DATA11}}$	$\overline{\text{DATA5}}$	57
60	$\overline{\text{DATA8}}$	$\overline{\text{DATA12}}$	59
62	$\overline{\text{DATA0}}$	$\overline{\text{DATA4}}$	61
64	$\overline{\text{DATA13}}$	$\overline{\text{DATA9}}$	63
66	$\overline{\text{DATA15}}$	$\overline{\text{DATA1}}$	65
68	User	User	67
70	User	User	69
72	User	User	71
74	$\overline{\text{EXTDCH}}$	$\overline{\text{DATA3}}$	73
76	User	$\overline{\text{DATA10}}$	75
78	User	User	77
80	User	User	79
82	$\overline{\text{DATA2}}$	-5V	81
84	User	User	83
86	User	User	85
88	+12V	+12V	87
90	+12V	GND	89
92	GND	Reserved	91
94	GND	Reserved	93
96	Reserved	$\overline{\text{DATA6}}$	95
98	+5V	+5V	97
100	GND	GND	99

INT-00671

Figure 6-2 Input/Output Backpanel Connector Layout (Viewed from Pin Side)

Table 6-1 General-Purpose Internal Device Cables

DGC Part Number	Reference Drawing	Description
005-018382	001-003197	Contains two 100-pin backpanel connectors and one 50-pin sub-D connector
005-018631	001-003204	Contains two 100-pin backpanel connectors and two 50-pin sub-D connectors
005-019499	001-003321	Contains one 100-pin backpanel connector and one 50-pin sub-D connector
005-018500	001-003320	Contains one 100-pin backpanel connector and one 50-pin sub-D connector
005-020307	001-003660	Contains two 100-pin backpanel connectors and two 50-pin sub-D connectors

End of Chapter

Chapter 7
Device Controller Boards

Device controller printed circuit boards designed by the user must meet certain specifications to be installed in the ECLIPSE MV/Family computer and expansion chassis. Controller circuit boards can be completely fabricated by the user. And Data General Corporation also makes available three types of boards that will aid the user in constructing custom device controller boards. One type of board available, *a general–purpose wiring board*, requires the user to design and build the entire controller circuitry. Information on these boards is found in this chapter.

The second type of board available, *a general–purpose I/O interface board*, provides designed and built basic I/O bus interface circuitry; the user only designs and builds the necessary controller circuitry. The user–designed circuitry is built on available space on the board and connected to the existing I/O bus interface circuitry with wiring. This board performs data transfers using the programmed I/O and data channel facilities of the host computer system. Information on the general–purpose I/O interface board can be found in the next chapter.

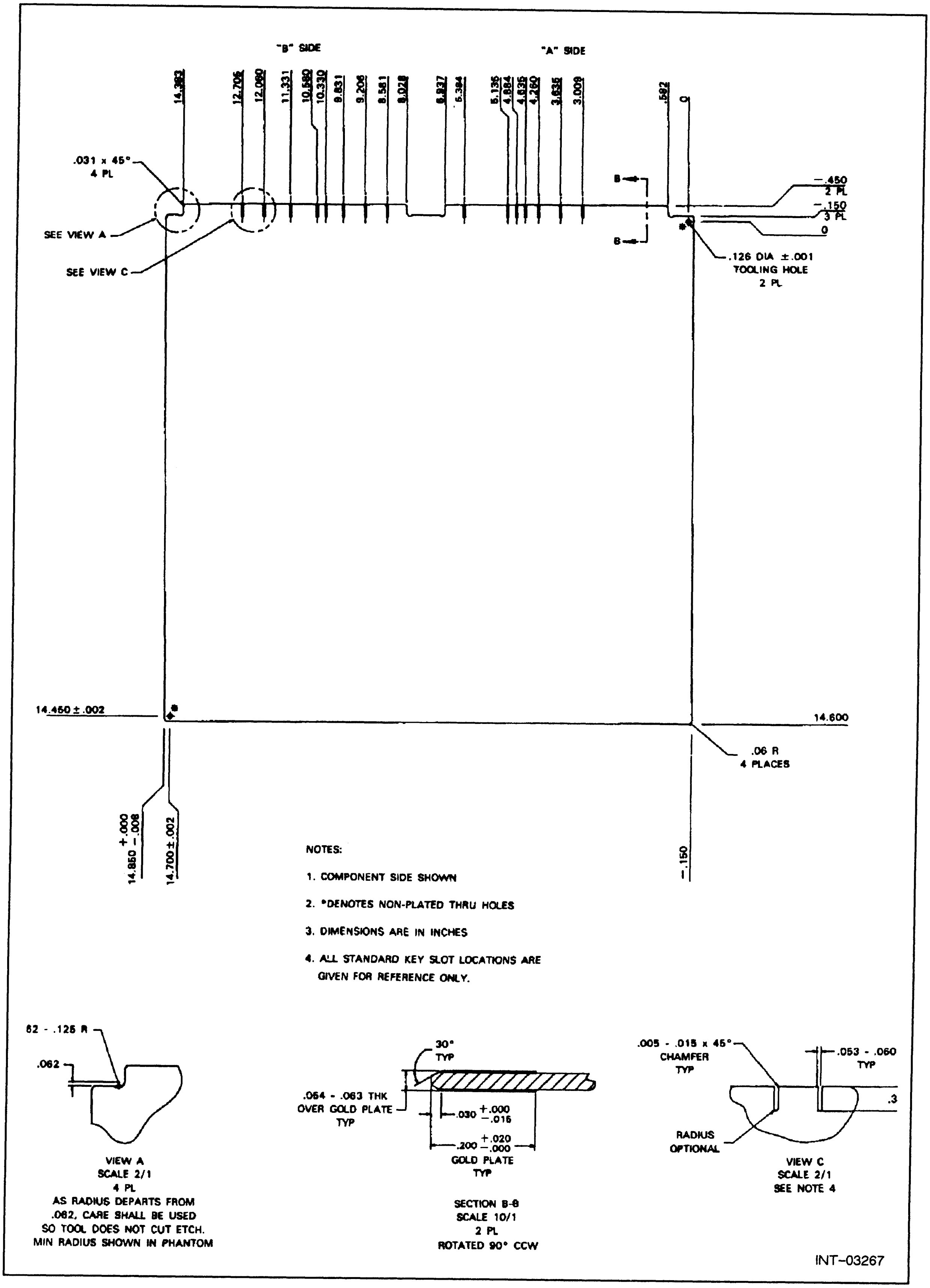
The third type of board available, *a high–speed, general–purpose BMC/DCH interface board*, provides a completely fabricated board that interfaces to the I/O bus and the BMC bus. This board makes available several general–purpose input and output signal lines for user definition. The board performs data transfers using the programmed I/O, data channel, or burst multiplexor channel facilities of the ECLIPSE MV/Family computer systems. Data transfers using either the data channel or burst multiplexor channel are program selectable. Information on the Model 5400GP General–Purpose BMC/DCH Interface can be obtained by contacting Data General Corporation's Special Systems Group.

Printed Circuit Board Specifications

The ECLIPSE computer and expansion chassis impose certain restrictions on the dimensions, vertical clearances, power consumption and heat dissipation of device controller printed circuit boards that can be installed in them.

Dimensions

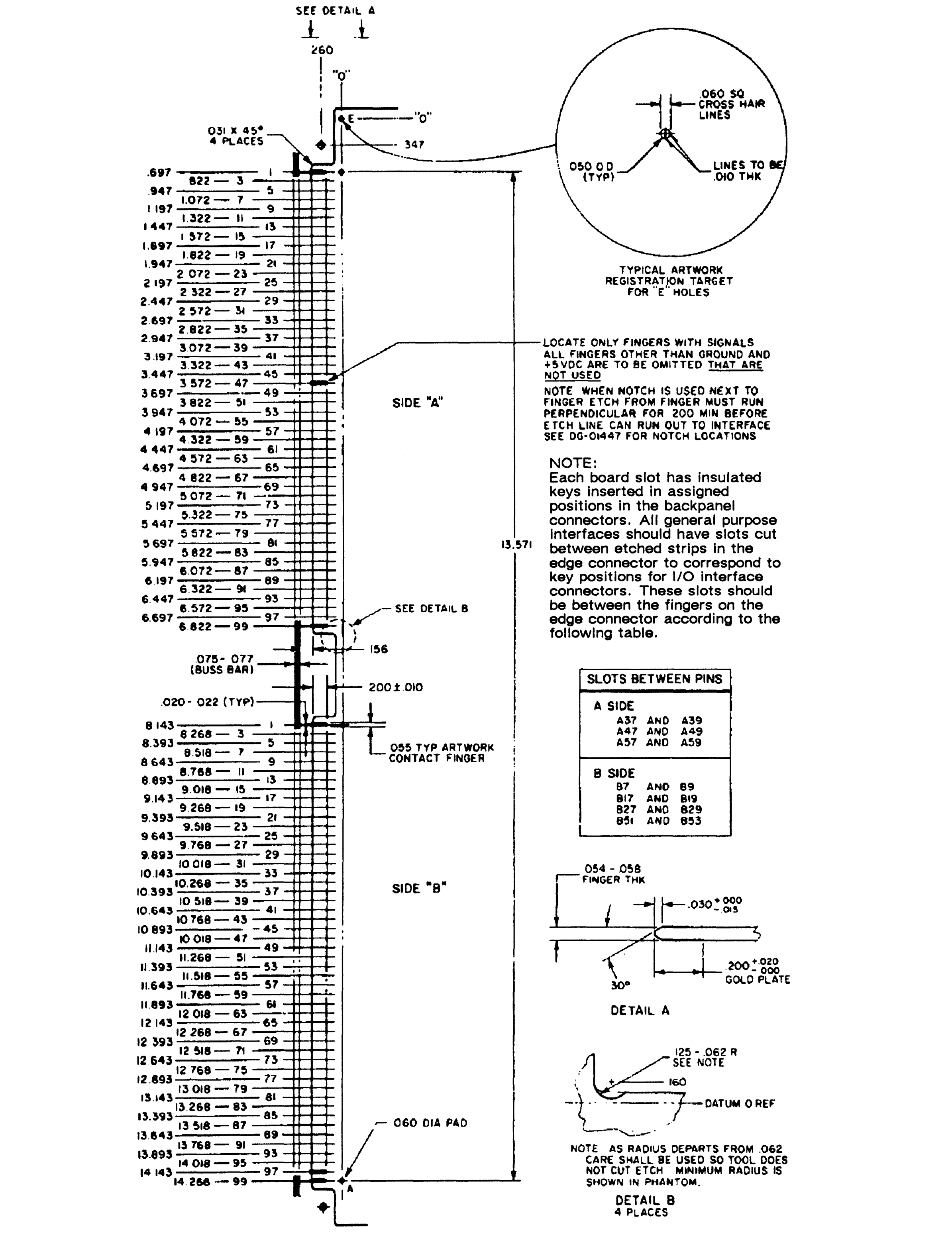
Figures 7–1 and 7–2 show the dimensions of printed circuit boards that can be installed into ECLIPSE MV/Family computers and expansion chassis.



NOTES:

1. COMPONENT SIDE SHOWN
2. *DENOTES NON-PLATED THRU HOLES
3. DIMENSIONS ARE IN INCHES
4. ALL STANDARD KEY SLOT LOCATIONS ARE GIVEN FOR REFERENCE ONLY.

Figure 7–1 Mechanical Dimensions



TYPICAL ARTWORK REGISTRATION TARGET FOR "E" HOLES

LOCATE ONLY FINGERS WITH SIGNALS ALL FINGERS OTHER THAN GROUND AND +5VDC ARE TO BE OMITTED THAT ARE NOT USED

NOTE WHEN NOTCH IS USED NEXT TO FINGER ETCH FROM FINGER MUST RUN PERPENDICULAR FOR 200 MIN BEFORE ETCH LINE CAN RUN OUT TO INTERFACE SEE DG-01447 FOR NOTCH LOCATIONS

NOTE:
Each board slot has insulated keys inserted in assigned positions in the backpanel connectors. All general purpose interfaces should have slots cut between etched strips in the edge connector to correspond to key positions for I/O interface connectors. These slots should be between the fingers on the edge connector according to the following table.

SLOTS BETWEEN PINS
A SIDE
A37 AND A39
A47 AND A49
A57 AND A59
B SIDE
B7 AND B9
B17 AND B19
B27 AND B29
B51 AND B53

DETAIL A

NOTE AS RADIUS DEPARTS FROM .062 CARE SHALL BE USED SO TOOL DOES NOT CUT ETCH MINIMUM RADIUS IS SHOWN IN PHANTOM.

DETAIL B
4 PLACES

Figure 7–2 Connector Specifications

Clearances

The clearance between circuit boards in an ECLIPSE MV/Family computer or expansion chassis, or between a circuit board and either the top of the chassis (for horizontally inserted boards) or the side of the chassis (for vertically inserted boards) is 0.375 inches. To maintain installation compatibility with other boards in the chassis, components must be mounted only on the top surface of the board. Components must not project more than 0.312 inches above the board, and no protrusion below the board may be greater than 0.062 inches. In general, a board should be constructed with a low profile so that air flow is restricted as little as possible.

Power Requirements

Device controller boards installed in an ECLIPSE MV/Family computer chassis or expansion chassis receive dc power, through the pins of their backpanel (backplane) edge connectors, from the chassis dc power supply. This immediately presents two considerations for the designer. The first is the capacity of the dc power supply, and the second is the quality of the voltages supplied. Power supply capacities and backpanel print numbers can be found in the appropriate ECLIPSE computer hardware configuration guide.

NOTE: It is important that a device controller board added to a computer or expansion chassis not overload the dc power supply of that chassis. Thus, the designer should know what the capacity of the machine is and design accordingly. Note that the capacity here is the total capacity of the dc power supply minus the load from all the boards already installed in the chassis.

The ECLIPSE computer chassis provides +5 V dc, +12 V dc, and –5 V dc to the I/O board slots of the backpanel. Installation data sheets, provided with each chassis, contain the dc capacities of the chassis as well as the current draw on the dc supplies by various circuit boards. The designer should use this information to calculate the actual capacity of the computer system. Current draw for the various circuit boards can also be found in the appropriate ECLIPSE computer hardware configuration guide; however, the appropriate installation data sheets may be more up–to–date than the reference manuals.

If there is insufficient capacity to drive the planned device controller board, you may wish to configure the system somewhat differently. The following are some possibilities:

- If possible, add an additional power regulator board to the ECLIPSE computer system where power configuration is below maximum (possible in some ECLIPSE computer systems).
- Use an expansion chassis for the planned device controller board or other device controller boards.
- Build most (or all) of the controller board in its own chassis, with its own power supply, especially if the planned device controller board is fairly large.

In addition to power supply capacity, the device controller board designer may need to be concerned with the quality of the available voltage. A characteristic of TTL logic is the tendency to superimpose switching transients and other noise on the power supply line. This noise may be particularly troublesome to analog circuitry, especially where, for instance, high–gain amplifiers are being used to deal with low–level signals.

If power supply noise is a problem, it may be desirable to isolate the device controller circuitry from the supply either by incorporating additional regulation on the controller assembly or by using a dc/dc converter.

Heat Dissipation of Device Controller Boards

The problem of heat dissipation has two facets. The first, a local effect, is the way in which the heat produced by hot components will affect nearby components; the second is the degree to which the heat produced by a device controller board assembly will raise the ambient temperature of adjacent boards in the chassis. Both of these considerations involve analyses that will be discussed only in the most basic terms here; but they are mentioned so that the designer will remain aware of possible restrictions.

The principal problem involved with localized heating is caused by concentrated high dissipation devices that create hot spots. Every ECLIPSE MV/Family computer and expansion chassis includes fans for forced air movement over the boards. Sufficient air moves past the heat-producing components to keep the air temperature rise within reason, thereby partially limiting the temperature level. Spreading the heat source over a wider area will improve the heat transfer across the board. Whenever possible, heat sensitive circuits and components should be mounted as far as possible from high-temperature components.

Effects on internal temperature rise can be minimized principally by limiting the total heat dissipation of the device controller board. If excessive heat is dissipated on the board, the result will be a rise in the ambient temperature for nearby boards. Take care that the profile of the board is low enough not to interfere with the air flow across the board. In general, if the total dissipation of a device controller board is less than 50 watts, there will be no problem of overheating other boards.

Prefabricated Wiring Boards

To aid the designer in constructing custom device controller assemblies, Data General makes available two series of wiring boards. These are particularly useful for building limited quantities of a special device controller; that is, where it would be economically unsuitable to lay out and etch a printed circuit board. The 1000- and 1020-series general-purpose wiring boards are blank component boards that allow the designer great flexibility in the design configuration.

1000 Series General-Purpose Wiring Boards

The 1000 series wiring board, shown in Figure 7-3, is a package consisting of a 15-inch square wiring frame with edge connectors (type 1001) on which wiring module boards can be mounted. This frame has two 100-pin edge connectors with solder pads, which are compatible with the female backpanel edge connectors. There is space on the frame for up to eight 6.5-inch x 3.25-inch wiring modules. The wiring modules themselves are available in three varieties:

- The 1002 is a basic module board with hole patterns for twelve 14- or 16-pin dual in-line integrated circuits (ICs). Discrete components can also be mounted in these holes. Solder pads are provided for the larger 24- and 36-pin ICs, but the holes are not predrilled through the module board and the solder pads.

Interconnections on the 1002 module board are made by soldering interconnecting wires or #30 AWG insulated solid wire to solder pads provided for each integrated circuit terminal.

- The 1003 module board is the same as the 1002 board except that it provides wire-wrap pins for the interconnecting wires, as shown in Figure 7-4.
- The 1004 module board is the same as the 1003 board except that, in addition to the wire-wrap pins that are provided, twelve 16-pin, low-profile sockets are also provided for added convenience in mounting dual in-line ICs.

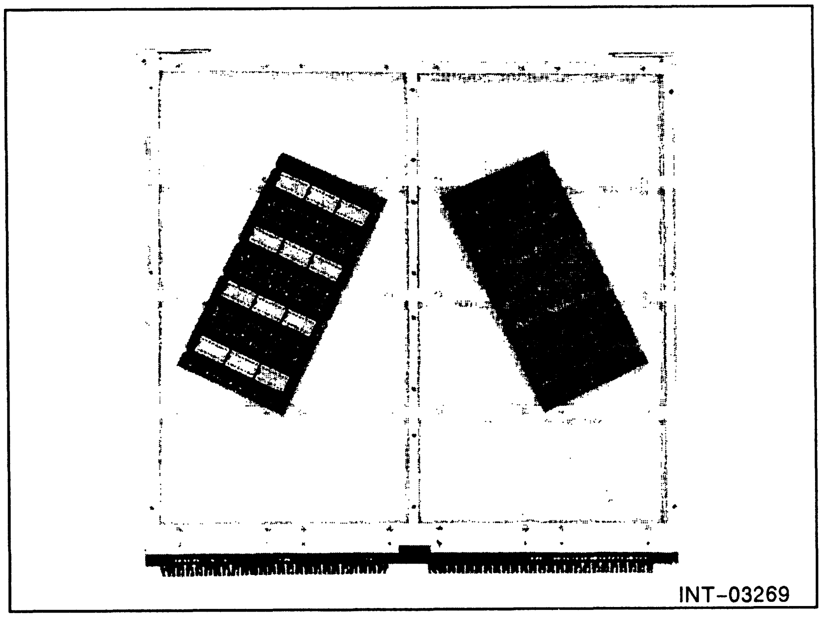


Figure 7-3 1000 Series Wiring Board

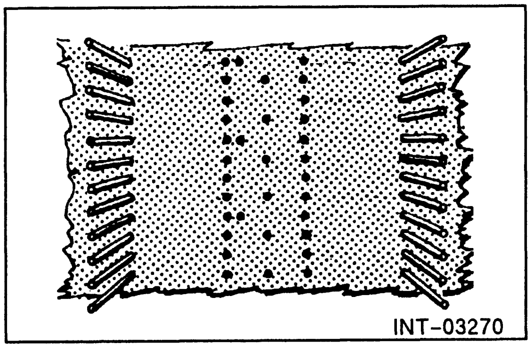


Figure 7-4 1000 Series Module Board Hole Pattern

There is a protective cover, type 1014, that fits over the 1001 wiring frame.

1020 Series General–Purpose Wiring Boards

The 1020 series wiring boards, shown in Figure 7–5, are 15–inch square printed circuit boards with a hole pattern for 14–, 16–, 24–, and 36–pin integrated circuits, as well as discrete components. The board has two 100–pin edge connectors, which are compatible with the chassis female backpanel connectors, and includes etched buses for dc power and ground throughout the board, including decoupling capacitors.

This series of wiring boards can hold 155 14– or 16–pin dual in–line IC packages. A pair of 24–pin packages replaces three 14– or 16–pin packages, while each 36–pin package replaces two 14– or 16–pin packages. This series of wiring boards is available in three varieties:

- The 1021 board, on which interconnections are made by soldering interconnecting wires or #30 AWG insulated solid wire to solder pads provided for each IC terminal.
- The 1022 board is the same as the 1021 board except that it provides wire–wrap pins for the interconnecting wires.
- The 1023 board is the same as the 1022 board except that, in addition to the wire–wrap pins, 155 16–pin, low–profile sockets provide added convenience in mounting dual in–line ICs.

There is a protective cover, type 1024, that fits over the 1020 series wiring boards.

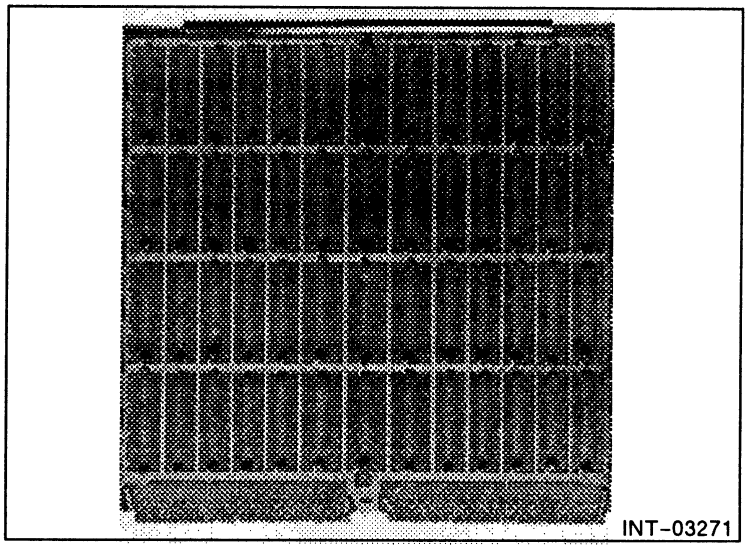


Figure 7–5 1020 Series Wiring Board

End of Chapter

Chapter 8
Model 4040A General–Purpose I/O Interface Board

The Model 4040A General–Purpose I/O Interface is a 15–inch, square printed–circuit board that provides basic I/O bus interface circuitry on approximately half the board, while the remainder of the board provides drilled hole patterns for a variety of integrated circuit chips as well as for discrete components.

Functionally, the board design accommodates a variety of interface applications. As shown in Figure 8–1, the existing circuitry provides the interface to the ECLIPSE I/O bus and the unpopulated area of the board, allowing for the custom design of peripheral controller circuitry for interfacing with specific peripheral devices.

Additionally, the board features:

- Two edge connectors, A and B, that plug into I/O slot female connectors of the computer or expansion chassis backpanel. (The odd numbered pins of the A and B connectors are located on the component side of the board; the even numbered pins are located on the opposite side.) (See Figure 8–2.)
- Two 78–pin, wire–wrap arrays, J5 and J6, used by the designer to connect the custom–designed circuitry to the basic I/O bus interface logic.
- Four 40–pin connectors with corresponding wire–wrap arrays — J1, J2, J3, and J4 — to be used at the designer's discretion.

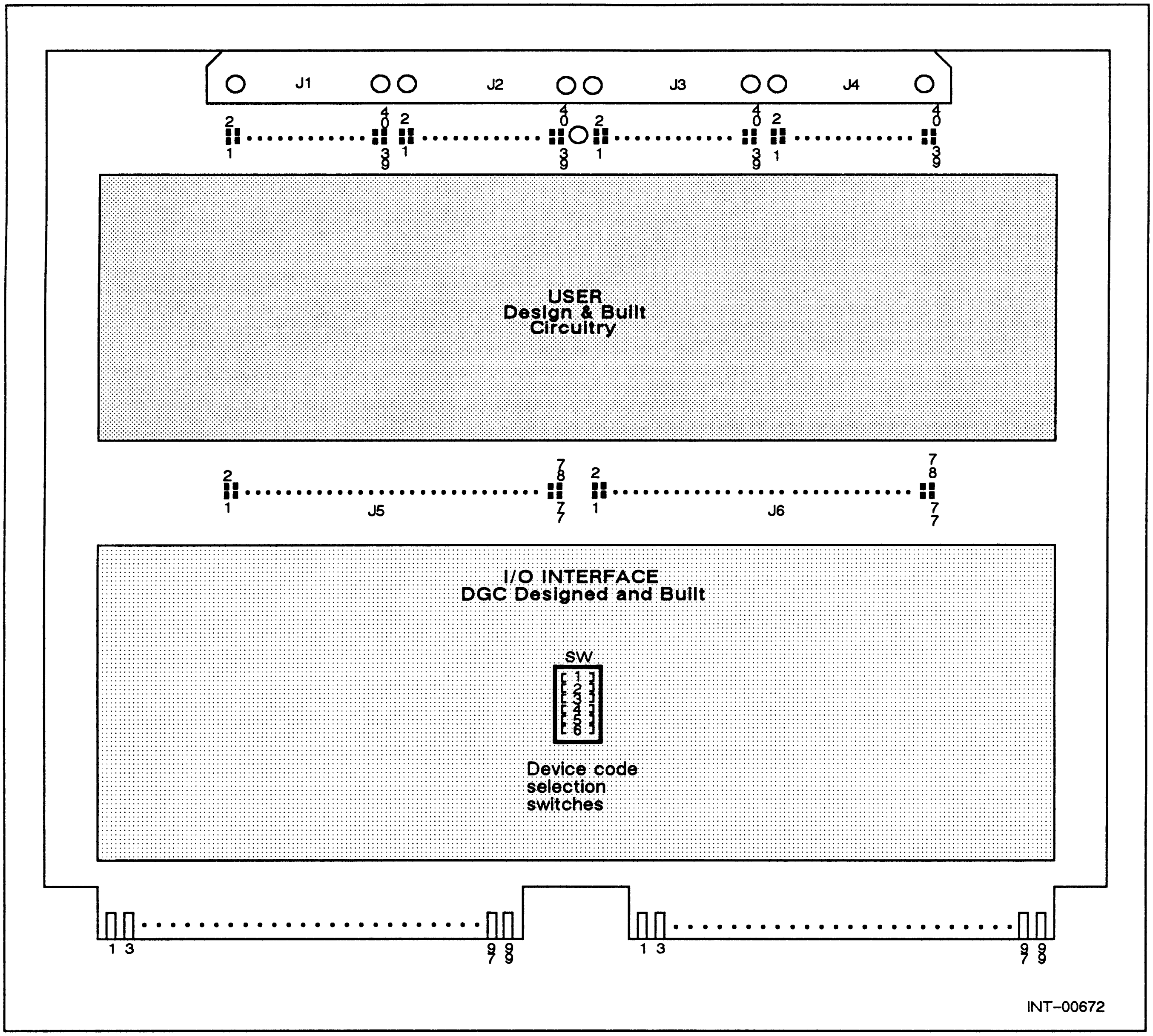
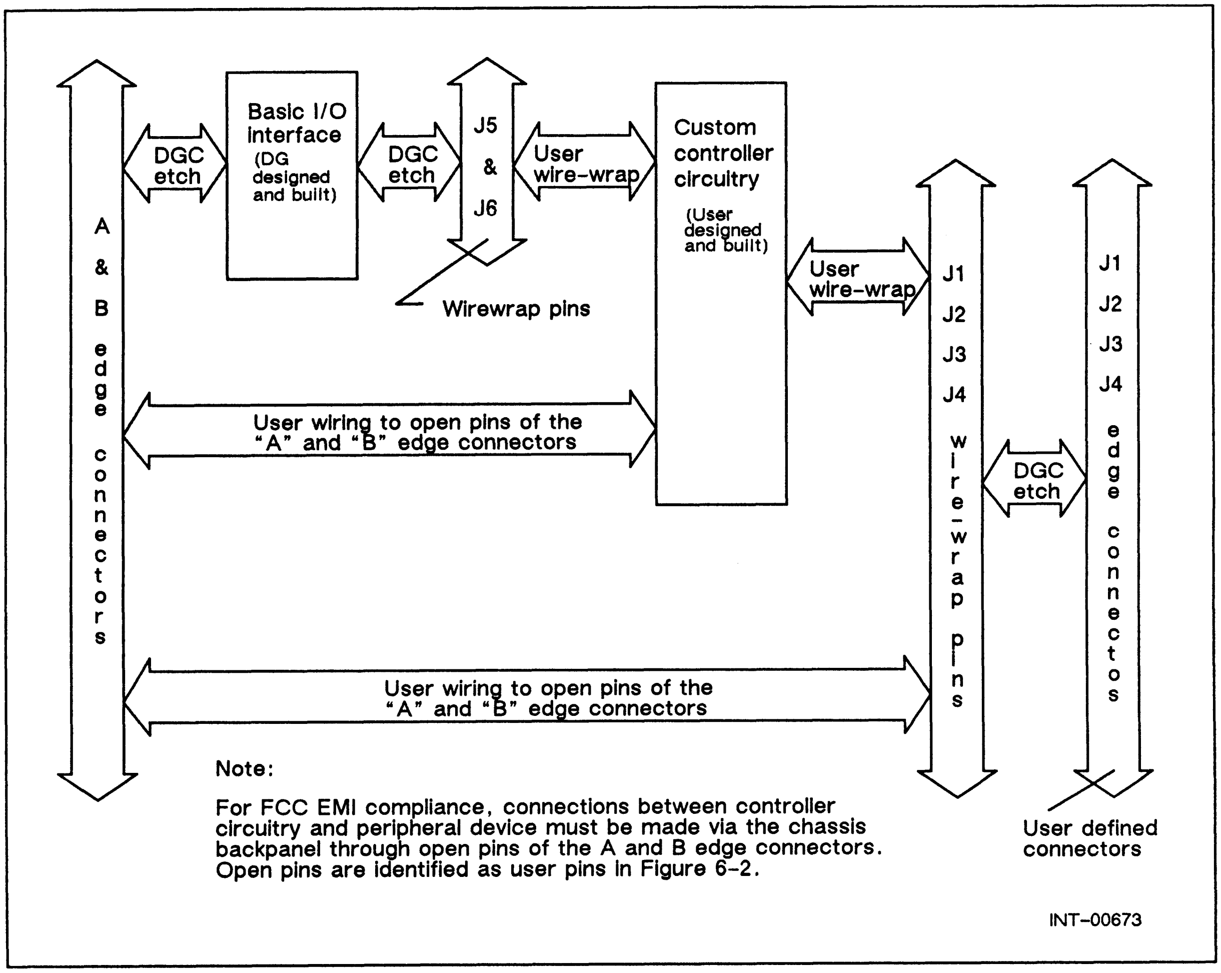


Figure 8–1 Model 4040A Printed Circuit Board Layout


A & B edge connectors
DGC etch
Basic I/O Interface (DG designed and built)
DGC etch
J5 & J6
User wire-wrap
Custom controller circuitry (User designed and built)
Wirewrap pins
User wire-wrap
J1 J2 J3 J4 wire-wrap pins
DGC etch
J1 J2 J3 J4 edge conectos
User wiring to open pins of the "A" and "B" edge connectors
User wiring to open pins of the "A" and "B" edge connectors
User defined connectors
INT-00673

Note:

For FCC EMI compliance, connections between controller circuitry and peripheral device must be made via the chassis backpanel through open pins of the A and B edge connectors. Open pins are identified as user pins in Figure 6-2.

Figure 8-2 Model 4040A Printed Circuit Board Features

Functional Description

The following description concerns only the existing I/O bus interface circuitry on the board. Essentially, the functions of the circuitry can be categorized under three basic groupings: I/O bus interface control, data handling, and data channel control. A description for these functional groups and their interaction is provided in the following sections.

Interface Control

The interface control circuits provide device select decoding, I/O control signal gating, Busy and Done flagging, as well as interrupt request and interrupt priority functions.

Device Select

The CPU selects the desired peripheral interface by including a specific device code within the programmed I/O instruction. This code is asserted on the I/O device select lines (^**DS<0-5>**) and is compared with switch settings on the Model 4040A board (Figure 8-3). When the code matches the switch settings, a ^**PDEVCODE** signal derived from a primary, even-numbered octal code (for example, 22_8) or a

^**SDEVCODE** signal derived from a secondary, odd-numbered octal code (for example, 23_8, as the case may be, will be asserted to the peripheral controller portion of the board).

Note that ^**DS5** is not switch selectable; thus the controller circuitry must be designed to respond to either one or both of the primary and secondary device code signals.

Assertion of the ^**PDEVCODE** signal or J5-17 (can be wired to ^**SDEVCODE**) also places the condition of the board's Done and Busy flags onto the respective ^**SELD** and ^**SELB** lines of the I/O bus.

In the case of a CPU response to an interrupt request, assertion of the Interrupt Acknowledge signal (**INTA**) causes the device select switches, for the highest priority interface requesting interrupt service, to be transferred to the CPU on the I/O bus data lines ^**DATA<10-15>**.

I/O Control Signals

Various I/O control signals from the I/O bus are gated through the I/O bus interface (Figure 8-3) to the peripheral controller portion of the board by assertion of the ^**PDEVCODE** signal or of J5-18 (which can be wired to ^**SDEVCODE**). Some of these I/O control signals are also used by the I/O bus interface circuitry.

Busy and Done Flags

The Busy and Done flags are used to provide status information to the CPU regarding operation of the peripheral device. In general, the Busy flag indicates that the peripheral device is busy and unable to respond to a CPU command, and the Done flag signifies that the peripheral has completed a task and is ready for a new command from the CPU. Figure 8-4 shows the Busy and Done circuitry.

The Busy flag is set by either a *Start* (^**START**) pulse from the CPU or a command from the peripheral controller, on J5-27 (also resets the Done flag). The Busy flag resets when the Done flag sets. It is also reset by a *Clear* (^**CLEAR**) or *IO Reset* (^**IORESET**) from the CPU.

The Done flag is set by either a direct setting with assertion of the J5-36 by the peripheral controller or by clocking from the peripheral controller on the J5-31 line in conjunction with the Busy flag set. (Clock setting from the device controller when the Busy flag is set can be disabled by a low assertion of the J5-4 line.) The Done flag resets by either a *Start* (^**START**) pulse from the CPU or a command from the peripheral controller, on J5-27 (when the Busy flag sets). It is also reset by a *Clear* (^**CLEAR**) or *IO Reset* (^**IORESET**) from the CPU.

A CPU programmed I/O instruction can test the condition of either flag by asserting the primary device code for the interface (the secondary device code can be used by wiring J5-6 to J5-17). Alternatively, the peripheral controller can be wired to accomplish the same thing by asserting J5-17.

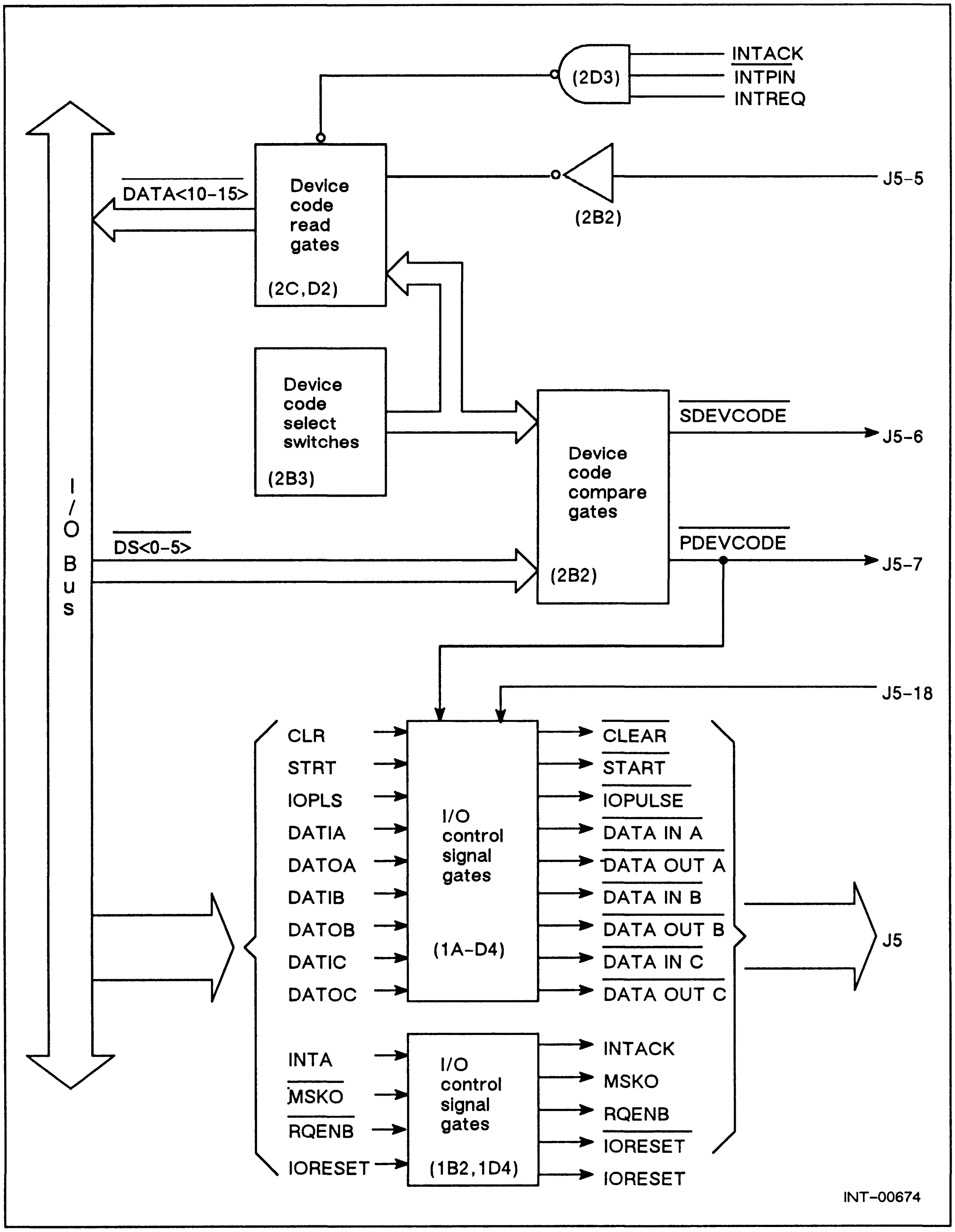


Figure 8-3 Device Selection and I/O Control Signals

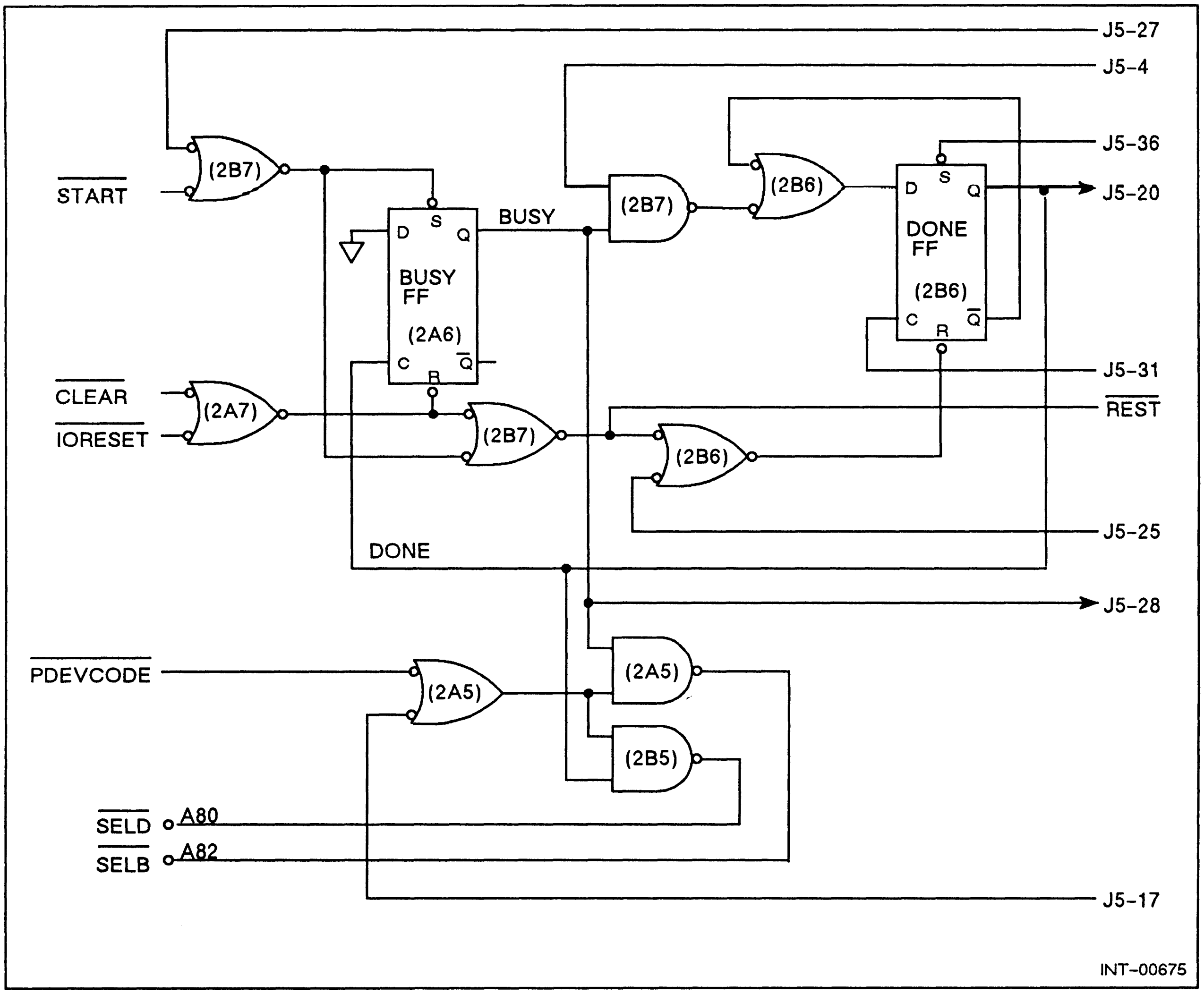


Figure 8–4 Model 4040A Busy and Done Circuitry

Interrupt Control

The peripheral controller signifies its readiness to communicate with the CPU by setting the Done flag. When the Done flag is set, the next assertion of the I/O bus controller-generated ^**RQENB** signal causes the Interrupt Request flip–flop to set as shown in Figure 8–5. This in turn, asserts an Interrupt Request signal (^**INTR**) to the CPU. Note that, with the Interrupt Request flip–flop set, the interrupt priority chain is broken. This inhibits the ^**INTPOUT** signal to I/O interfaces farther from the I/O bus controller on the I/O bus.

When the I/O bus controller responds to the interrupt request with an Interrupt Acknowledge (**INTA**) signal and the Model 4040A interface is the highest priority interface requesting interrupt service (**INTREQ** and **INTPIN** signals), the I/O bus interface asserts its device code on the ^**DATA<10–15>** signal lines. Note that the least significant bit of the device code must be jumpered when the secondary device code (odd-numbered) is used.

The CPU can selectively inhibit interrupts on any of the interface boards by asserting the ^**MSKO** command along with the mask-out bit pattern on the I/O bus data lines

(^**DATA<0-15>**. In this case, assertion of the mask-out bit must be felt at J5-3, which in turn causes the Interrupt Disable flip-flop to set when clocked by the ^**MSKO** signal. While the Interrupt Disable flip-flop is set, the Interrupt Request flip-flop cannot be set.

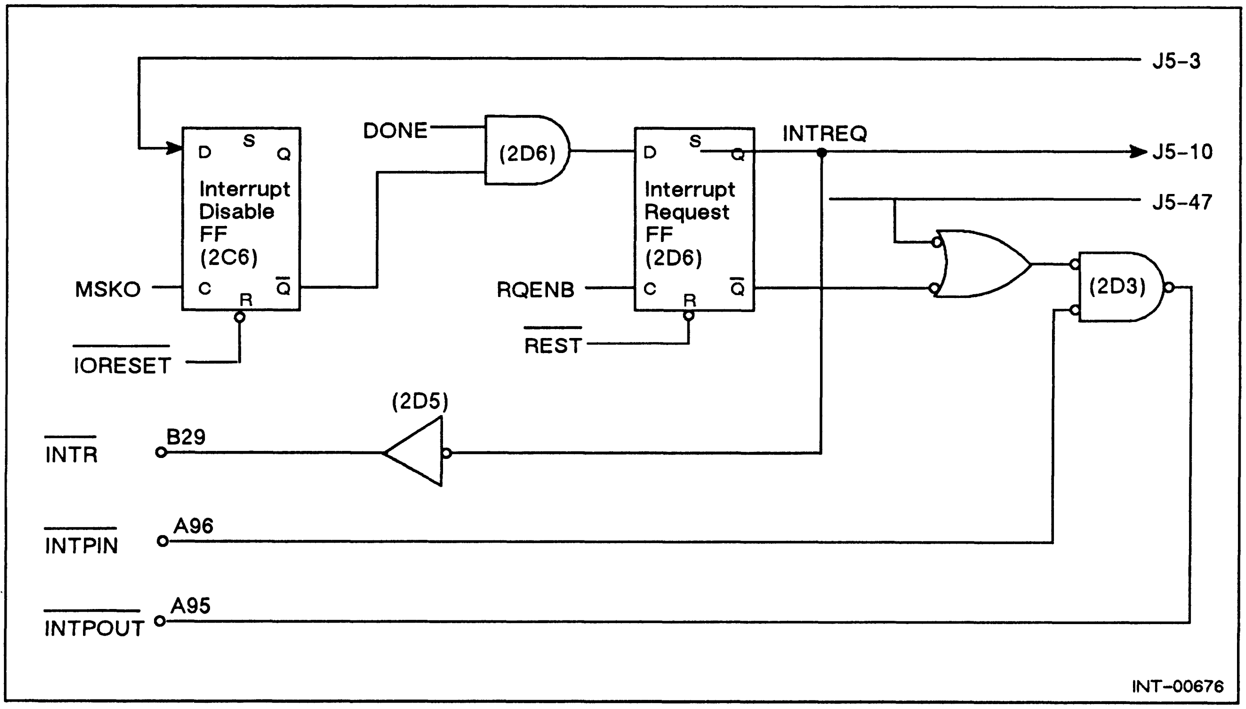


Figure 8-5 Model 4040A Interrupt Control Circuitry

Data Handling

The data handling circuits, shown in Figure 8-6, provide the data path between the ECLIPSE I/O bus and the peripheral controller logic. They also provide the memory addressing and word count facilities for data channel operation.

The data handling circuits provide facilities for both serial and parallel transfer of data or the conversions of serial to parallel and parallel to serial. Input gates 1A7 through 1D7 provide the parallel transfer from the J5/J6 connectors to the I/O bus. Conversely, inverters 1A6-1D6 provide the parallel transfer from the I/O bus to the J6 connector. Serial transfers of data occur using either shift registers 3A6-3D6 and input gates 1A7-1D7 (from peripheral controller to I/O bus) or inverters 1A6-1D6 and shift registers 3A3-3D3 (from I/O bus to peripheral controller). Additionally, the shift registers may be used to provide the serial/parallel conversions on data received from the peripheral controller and returned again to the peripheral controller. When performing serial operations, shift left moves bits from bit position 15 towards bit position 0; shift right moves bits in the opposite direction.

The data handling circuits also incorporate two counters, the data channel Word Counter (5A6 through 5D6) and Address Counter (5A3 through 5D3) that are used during data channel operations.

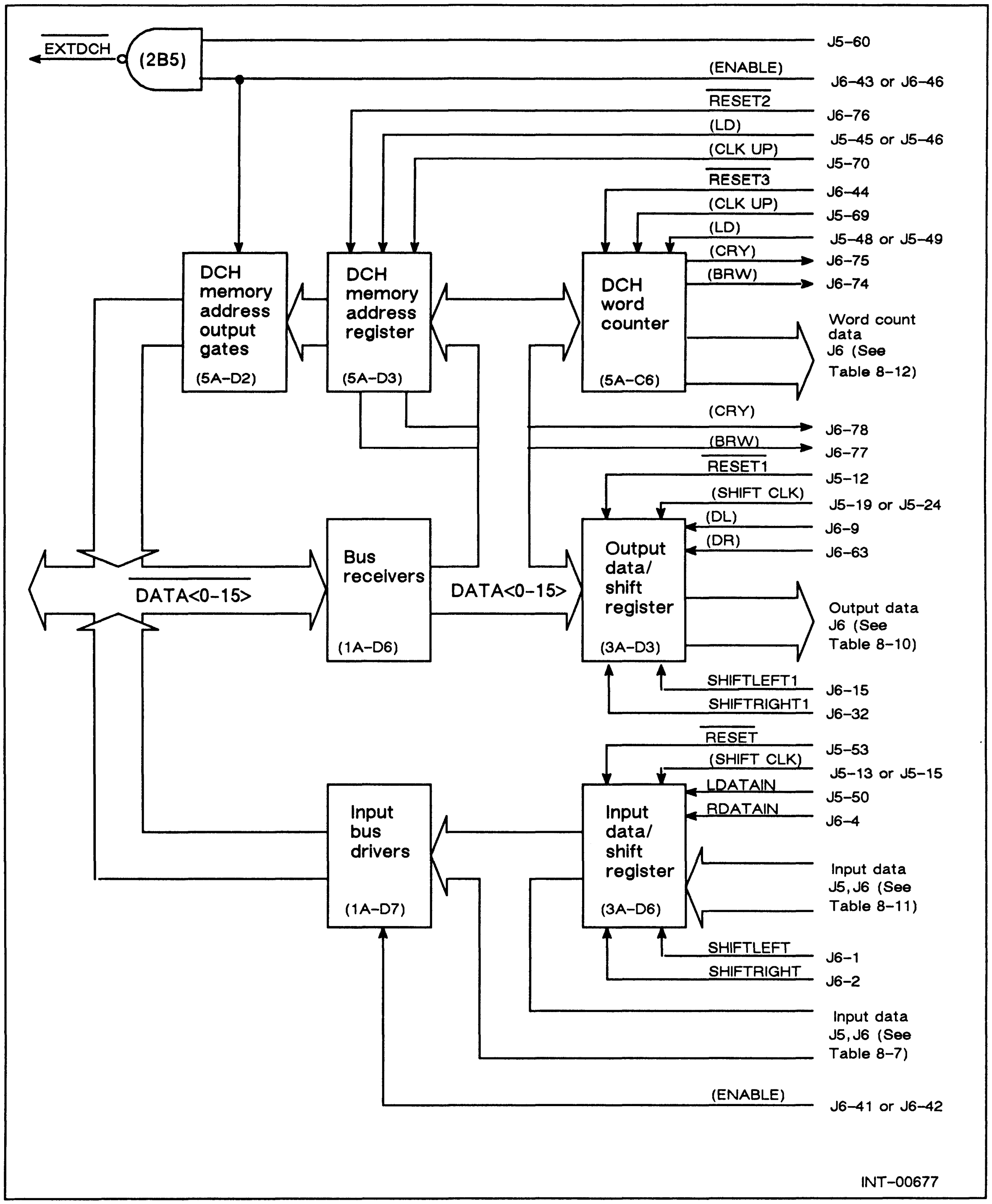


Figure 8-6 Model 4040A Data Handling Circuitry

Data Channel Control

The data channel control circuits, shown in Figure 8-7, provide the various control functions required for data channel operation between memory and the interfacing peripheral controller. These functions include data channel request, priority, data transfer mode select, input data transfer, and output data transfer.

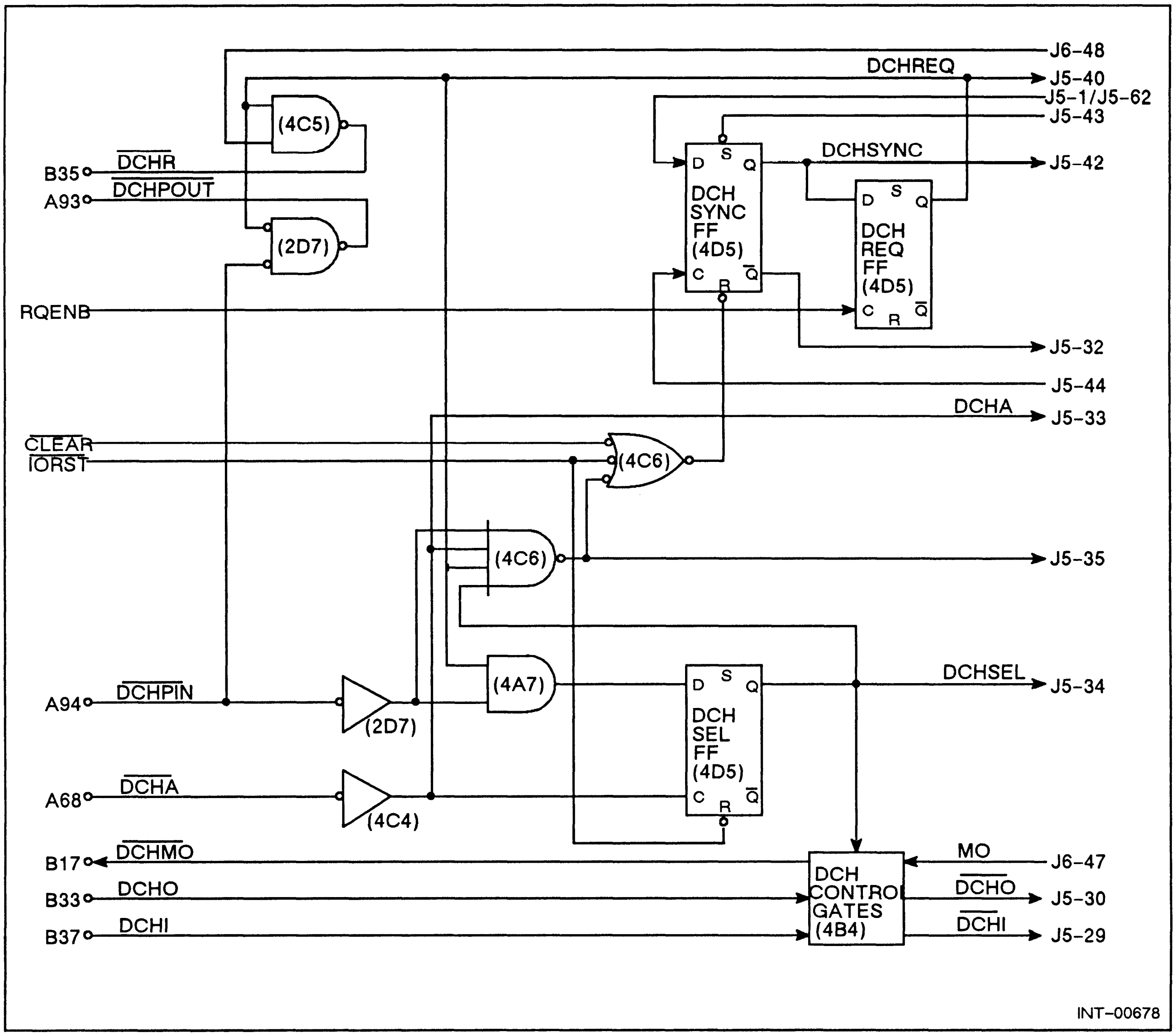


Figure 8-7 Model 4040A Data Channel Control Circuitry

The peripheral controller initiates a data channel request by setting the DCH Sync flip-flop (the designer must provide the circuitry and signal jumper(s) required to set DCH Sync). In turn, the next ^**RQENB** from the I/O bus controller sets the DCH Request flip-flop, asserting ^**DCHR** signal on the I/O bus to the I/O bus controller. Note that, with the DCH Request flip-flop set, the ^**DCHPOUT** signal is disabled. Therefore, the serial data channel priority chain is broken to I/O interfaces farther from the I/O bus controller on the I/O bus priority chain.

When the I/O bus controller responds to the data channel request with an acknowledge (^**DCHA**) signal, and the Model 4040A interface is the highest priority interface requesting data channel service (DCHREQ and DCHPIN signals), the Data Channel Select flip-flop is set and the DCH Sync flip-flop is reset.

The Data Channel Select flip-flop enables the DCH control gates to transfer the mode-select (**M0**) signal from the peripheral controller to the I/O bus controller via the ^**DCHMO** signal line. (The designer must provide the circuitry and signal jumper, J6–47, required for **MO**.) **^DCHMO** defines whether the request to the I/O bus controller is for an input or an output operation.

In addition to transferring the mode select signal while **^DCHA** is asserted, the memory address must also be transferred to the I/O bus controller. This is accomplished by the designer providing the circuitry and signal jumper (J6–43 or J6–46) to enable the address output gates shown in the data handling section (Figure 8–6).

Another concern for the controller designer is that driving of the **^EXTDCH**, **^DATA<0>**, and **^DS<4–5>** signal lines may be required for DCH memory addressing (refer to "Data Channel Map Slot Selection" and "Loading Data Channel Map from I/O Device" in Chapter 7). With the exception of **^EXTDCH**, when driving of these signal lines is required, open–collector controller circuitry must be incorporated by the designer to drive the respective **^DS** signal lines. The I/O bus interface circuitry includes a bus driver for **^EXTDCH**; thus, the designer needs to provide only a control circuit and jumper, J5–60, when using this signal.

When the I/O bus controller performs the data transfer, it asserts either the **DCHO** (output) or **DCHI** (input) signal. The Data Channel Select flip–flop enables the DCH Control gates to transfer the respective signal to the controller circuitry. Using the respective data transfer signal, the designer must provide the necessary circuitry and jumpering to either place data onto the I/O bus from the interface–provided input register or load data from the I/O bus into the interface–provided output data register. Also the designer must provide circuitry and jumpering to update both the Memory Address register and Word Counter during the data channel cycle.

Near the completion of the data transfer the I/O bus controller will clock the **^RQENB** signal line until another data channel request is received. Since the DCH Sync flip–flop is now reset (unless it had once again been set by the controller circuitry if back–to–back data transfers are required), the DCH Request flip–flop is reset. This in turn enables data channel priority to be passed to lower priority peripherals when the Model 4040A board is receiving data channel priority input. The DCH Select flip–flop remains set until the next data channel operation begins (**^DCHA** asserted) for any data channel peripheral controller.

Switch Settings

A six–switch array (SW1) defines the device code for the board. Its configuration is the only tailoring required of the basic I/O interface logic on the board. Note that the individual switches of the array, 1 through 5, are in respective correspondence with device select lines **^DS<0–4>**. The open (ON) position of the switch equals a "1" for the respective device code bit. (For the location of the switch array on the board, refer back to Figure 8–1.)

Set the individual switches so that the octal value of the device code is defined by **^DS<0–4>** (see Figure 8–8).

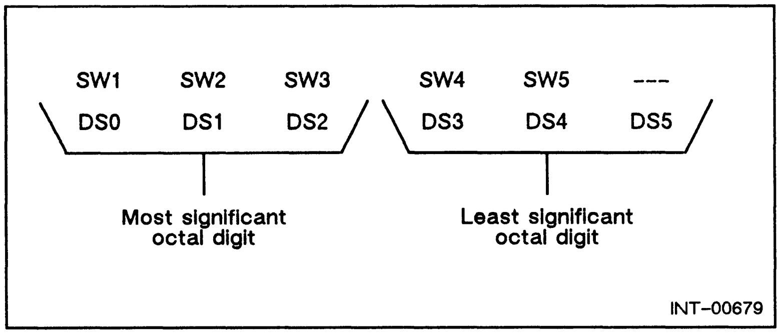


Figure 8-8 Switch Settings

NOTE: ^**DS5** is not switch selectable; thus, the controller circuitry must be designed to respond to one (or both) of the primary device codes (even-numbered; ^**DS5** not asserted) and secondary device codes (odd-numbered; ^**DS5** asserted).

Technical Specifications

Tables 8-1 and 8-2 list electrical characteristics and operating environmental specifications for the Model 4040A General Purpose I/O Interface Board.

Table 8-3 lists general specifications for interfacing the Model 4040A basic I/O bus interface circuitry to the peripheral controller. Tables 8-4 and 8-5 list the specific jumpering required to program the Input and Output Shift/Data registers. These tables define all the facilities available to the designer; however, only those features required for a particular controller need be used. Refer to previous chapters in this manual for a detailed description of input/output programming, I/O bus functions, I/O bus timing, and electrical specifications. For detailed schematic information, refer to engineering drawing 001-002864.

Table 8-1 Electrical Characteristics

Item	Description
Interfacing wire-wrap pins	J5 and J6; see Table 8-6 through Table 8-15 for pin assignments
Signal voltage/current/rise time	Standard TTL logic characteristics

Table 8-2 Operating Environmental Specifications

Item	Operating Range
Temperature	0°C - 55°C 32°F - 131° F
Relative humidity	10% - 90%, noncondensing
Altitude	-305 - +2430 m -1000 - +8000 ft

Table 8-3 General Specifications

Function	Description
Data Transfer	
Parallel Input/Parallel Output	Data bits of the Input and Output data registers (16-bits each register) at wire-wrap pins of connectors J5 and J6 (see Table 8-7 and Table 8-8). Used for programmed input/output (PIO) and data channel (DCH) transfers. PIO instructions and data channel transfer must be discriminated by the peripheral controller. Outputs of the input data register or the contents of the input data wire-wrap pins of J5/J6 are applied onto the I/O bus (B connector) by assertion of either J6-41 or J6-42. Controller circuitry uses the outputs of both data registers by wiring from the respective wire-wrap pins of J5/J6. Respective controlling connection points for both data registers are shown in Table 8-4 and Table 8-5.
Serial Input/Serial Output	The Input and/or Output data register can be controlled by the designedcontroll er circuitry to accommodate serial input and serial output of data. Respective controlling connection points for both data registers are shown in Table 8-4 and Table 8-5.
Serial Input/Parallel Output	The Input and/or Output data register can be controlled by the designed controller circuitry to accommodate serial input and parallel output of data. Outputs of the input data register or the contents of the input data wire-wrap pins of J5/J6 are applied onto the I/O bus (B connector) by assertion of either J6-41 or J6-42. Respective controlling connection points are shown in Table 8-4 and Table 8-5.
Parallel Input/Serial Output	The Input and/or Output data register can be controlled by the designed controller circuitry to accommodate parallel input and serial output of data. Respective controlling connection points are shown in Table 8-4 and Table 8-5.
Data Channel Control	
Requests	Data channel requests are initiated by the peripheral controller with the setting of DCH Sync flip-flop initiated as follows: Direct set by negative assertion of J5-43 Clock pulse on J5-44 in conjunction with assertion of either J5-1 or J5-62. DCH Request flip-flop sets with the next ^**RQENB** following setting of the DCH Sync flip-flop. ^**DCHR** is gated to the I/O bus controller by assertion of pin J6-48 from peripheral controller. The status of DCH Sync FF is presented back to the controller on either pin J5-42 (**Q**) or pin J5-32 (^**Q**). Status of DCH Request FF is presented on pin J5-40.
Mode (Input/Output)	Must be defined by peripheral controller's M0 signal, on pin J6-47. M0 asserted high = input; M0 asserted low = output. I/O bus controller responds with either ^**DCHO** asserted on pin J5-30 or ^**DCHI** asserted on pin J5-29.
Starting Memory Address	Must be defined by programmed I/O command from I/O bus controller (or supplied by peripheral controller on the output data transfer bus, J6 (see Table 8-8); address is stored in the DCH Address Counter by assertion of either J5-45 or J5-46; address is incremented by clock pulse asserted on J5-70; address is gated onto I/O bus (B connector) by assertion of J6-43 or J6-46. Address count overflow can be monitored at J6-78.

(continued)

Table 8-3 General Specifications

Function	Description
Transfer Word Count	Must be defined by programmed I/O command from I/O bus controller (or supplied by peripheral controller on the output data transfer bus, J6 (see Table 8-8); count is stored in the DCH Word Counter by assertion of either pin J5-48 or J5-49; count is incremented by clock pulse asserted on J5-69. The count status can be monitored at wire-wrap array J6 (see Table 8-12) or at pin J6-75 (Carry — trickle count up). Contents of the Word Counter can be transferred to the CPU by wiring the word counter output wire-wrap pins of J6 to the respective input data wire-wrap pins of J5/J6.
Interrupt and Status Control	
Device Select	Decoded primary and secondary Device Select signals provided to the peripheral controller on pins J5-7 and J5-6, respectively.
Device Code	Device code is returned to the computer when Interrupt Acknowledge (INTA) is received and the Model 4040A is the highest priority peripheral controller requesting interrupt service. Note that the least significant bit of the device code must be jumpered at J5-5 when the secondary device code (odd-numbered) is used.
Busy Flip-flop	Setting is initiated by either a *Start* flag control command received from the I/O bus controller or by assertion of pin J5-27 from the peripheral controller. Status of the flip-flop is gated to CPU by either the primary device code received from the I/O bus controller or by the assertion of pin J5-17 from the peripheral controller. Status is also presented back to controller on pin J5-28.
Done Flip-flop	Setting is initiated by peripheral interface as follows: Direct set by negative assertion of J5-36. Clock pulse on J5-31 in conjunction with Busy flip-flop set. Clock pulse on J5-31 in conjunction with assertion of J5-4. Status of the flip-flop is gated to the CPU by either the primary device code received from the I/O bus controller or by the assertion of pin J5-17 from the peripheral controller. Status is also presented back to controller on pin J5-20.
Interrupt Request	Initiated by the I/O bus interface control circuits with the first RQENB signal received from the I/O bus controller following setting of the Done flip-flop; status is returned back to the peripheral controller on pin J5-10.
I/O Control Signals	Provided to the peripheral controller at connector J5 (see Table 8-6).

(concluded)

Table 8-4 Jumper Connections for Output Register

Mode	Input	Output	Clock
Serial Input/Output - left aligned data			
J6-32 = L; J6-15 = H	J6-9	J6-67	J5-19/J5-24
Serial Input/Output - right aligned data			
J6-32 = H; J6-15 = L	J6-63	J6-11	J5-19/J5-24
Parallel Input			
J6-32 = H; J6-15 = H	DATA<0-15>	—	J5-19/J5-24
Parallel Output			
J6-32 = L; J6-15 = L	—	J6*	—

* See Table 8-10

Table 8-5 Jumper Connections for Input Register

Mode	Input	Output	Clock
Serial Input/Output - left aligned data			
J6-2 = L; J6-1 = H	J5-50	J6-10	J5-13/J5-15
Serial Input/Output - right aligned data			
J6-2 = H; J6-1 = L	J6-4	J5-57	J5-13/J5-15
Parallel Input			
J6-2 = H; J6-1 = H	J5,J6 [1]	—	J5-13/J5-15
Parallel Output			
J6-2 = L; J6-1 = L	—	IDATA<0-15> J5,J6 [2]	—

[1] See Table 8-9

[2] See Table 8-7

Controller Interface Signal Names/Pin Assignments

Table 8–6 through Table 8–13 provide the signal names and pin assignments that are available to the designer for implementation in the interface circuitry. Table 8–14 and Table 8–15 provide the signals and their functions for the J5 and J6 wire-wrap pins in numerical sequence. Information for fanout and loading of signals by the interface circuitry is also included in these tables. Fanout and loading shown in these tables are expressed in milliamperes. Input and output data, described in Table 8–7 through Table 8–10, refers to the central processing unit.

Table 8–6 I/O Control

Signal	Pin	Fanout	Load
^IO RESET	J5-41	16.0	—
^START	J5-8	16.0	—
^CLEAR	J5-9	16.0	—
^IOPULSE	J5-2	16.0	—
^DATA IN A	J5-11	16.0	—
^DATA OUT A	J5-14	16.0	—
^DATA IN B	J5-21	16.0	—
^DATA OUT B	J5-12	16.0	—
^DATA IN C	J5-16	16.0	—
^DATA OUT C	J5-22	16.0	—
INT ACK	J5-38	16.0	—
MSKO	J5-37	16.0	—
RQENB	J5-39	16.0	—
Enable Control Gates	J5-18	—	1.6
INT REQ	J5-10	14.0	—
Enable ^INTPOUT	J5-47	—	0.4
Enable Interrupt Disable FF	J5-3	—	3.5
DONE	J5-20	10.0	—
Set DONE FF	J5-36	—	1.6
Clk DONE FF	J5-31	—	3.2
Disable DONE FF	J5-4	—	1.6
Reset DONE FF	J5-25	—	0.7
Set BUSY, Reset DONE	J5-27	—	0.7
BUSY	J5-28	14.0	—
Enable ^SELD, ^SELB	J5-17	—	2.0
Assert ^DATA15 on Device Code Read	J5-5	—	2.0
^SDEVCODE	J5-6	16.0	—
^PDEVCODE	J5-7	16.0	—

Table 8-7 Input Data

Signal	Pin	Fanout	Load
IDATA0	J6-10	6.0	—
IDATA1	J6-7	6.0	—
IDATA2	J6-39	6.0	—
IDATA3	J6-45	6.0	—
IDATA4	J5-76	6.0	—
IDATA5	J5-75	6.0	—
IDATA6	J5-78	6.0	—
IDATA7	J5-77	6.0	—
IDATA8	J5-68	6.0	—
IDATA9	J5-67	6.0	—
IDATA10	J6-40	6.0	—
IDATA11	J6-37	6.0	—
IDATA12	J5-59	6.0	—
IDATA13	J6-58	6.0	—
IDATA14	J6-56	6.0	—
IDATA15	J6-57	6.0	—
Enable Input Data	J6-41	—	4.0
Enable Input Data	J6-42	—	4.0

Table 8-8 Output Data

Signal	Pin	Fanout	Load
DATA0	J6-64	16.0	—
DATA1	J6-62	16.0	—
DATA2	J6-69	16.0	—
DATA3	J6-61	16.0	—
DATA4	J6-51	16.0	—
DATA5	J6-52	16.0	—
DATA6	J6-49	16.0	—
DATA7	J6-50	16.0	—
DATA8	J6-25	16.0	—
DATA9	J6-35	16.0	—
DATA10	J6-26	16.0	—
DATA11	J6-38	16.0	—
DATA12	J6-21	16.0	—
DATA13	J6-24	16.0	—
DATA14	J6-18	16.0	—
DATA15	J6-23	16.0	—

Table 8-9 Input Data Register

Signal	Pin	Fanout	Load
SHIFT LEFT	J6-1	—	0.7
SHIFT RIGHT	J6-2	—	0.7
^RESET	J5-53	—	0.7
LDATAIN (Left Data In)	J5-50	—	0.4
RDATAIN (Right Data IN)	J6-4	—	0.4
Shift CLK	J5-13	—	2.0
Shift CLK	J5-15	—	2.0
D Input - IDATA15	J5-55	—	0.4
C Input - IDATA14	J5-54	—	0.4
B Input - IDATA13	J5-51	—	0.4
A Input - IDATA12	J5-52	—	0.4
D Input - IDATA11	J5-66	—	0.4
C Input - IDATA10	J5-65	—	0.4
B Input - IDATA9	J5-64	—	0.4
A Input - IDATA8	J5-63	—	0.4
D Input - IDATA7	J5-74	—	0.4
C Input - IDATA6	J5-73	—	0.4
B Input - IDATA5	J5-72	—	0.4
A Input - IDATA4	J5-71	—	0.4
D Input - IDATA3	J6-8	—	0.4
C Input - IDATA2	J6-5	—	0.4
B Input - IDATA1	J6-6	—	0.4
A Input - IDATA0	J6-3	—	0.4

NOTE: The pin assignments shown in Table 8-7 can also be used as outputs of the Input Data register for the respective data bits 0-15 shown in Table 8-9.

Table 8-10 Output Shift/Data Registers

Signal	Pin	Fanout	Load
Shift Left	J6-15	—	0.7
Shift Right	J6-32	—	0.7
^RESET	J6-12	—	0.7
Data Left Input	J6-9	—	0.4
Data Right Input	J6-63	—	0.4
Shift CLK	J5-19	—	2.0
Shift CLK	J5-24	—	2.0
D Output - DATA15	J6-11	8.0	—
C Output - DATA14	J6-14	8.0	—
B Output - DATA13	J6-13	8.0	—
A Output - DATA12	J6-16	7.6	—
D Output - DATA11	J6-28	7.6	—
C Output - DATA10	J6-27	8.0	—
B Output - DATA9	J6-29	8.0	—
A Output - DATA8	J6-30	7.6	—
D Output - DATA7	J6-54	7.6	—
C Output - DATA6	J6-53	8.0	—
B Output - DATA5	J6-55	8.0	—
A Output - DATA4	J6-56	7.6	—
D Output - DATA3	J6-65	7.6	—
C Output - DATA2	J6-66	8.0	—
B Output - DATA1	J6-68	8.0	—
A Output - DATA0	J6-67	8.0	—

Table 8-11 Data Channel Control

Signal	Pin	Fanout	Load
Set DCH SYNC FF	J5-43	—	2.0
Clock DCH SYNC FF	J5-44	—	3.5
Enable DCH SYNC FF	J5-1	—	2.0
Enable DCH SYNC FF	J5-62	—	2.0
DCH SYNC FF - Q	J5-42	16.0	—
DCH SYNC FF - ^Q	J5-32	16.0	—
Clear DCH SYNC	J5-35	15.0	—
DCH REQ	J5-40	13.0	—
Enable ^DCHR	J6-48	—	2.0
DCHSEL	J5-34	10.0	—
DCHI	J5-29	16.0	—
DCHO	J5-30	16.0	—
DCHA	J5-33	16.0	—
M0	J6-47		2.0
Assert ^EXTDCH	J6-60	—	2.0
Enable Address Gates	J6-43	—	4.0
Enable Address Gates	J6-46	—	4.0

Table 8-12 DCH Word Counter*

Signal	Pin	Fanout	Load
Clock Word Counter	J5-69	—	2.0
LD Word Counter	J5-48	—	0.6
LD Word Counter	J5-49	—	0.6
^**RESET3** (Word Counter)	J6-44	—	2.0
Output Data Bit 15	J6-17	8.0	—
Output Data Bit 14	J6-20	8.0	—
Output Data Bit 13	J6-19	8.0	—
Output Data Bit 12	J6-22	8.0	—
Output Data Bit 11	J6-33	8.0	—
Output Data Bit 10	J6-31	8.0	—
Output Data Bit 9	J6-34	8.0	—
Output Data Bit 8	J6-36	8.0	—
Output Data Bit 7	J6-57	8.0	—
Output Data Bit 6	J6-58	8.0	—
Output Data Bit 5	J6-60	8.0	—
Output Data Bit 4	J6-59	8.0	—
Output Data Bit 3	J6-71	8.0	—
Output Data Bit 2	J6-70	8.0	—
Output Data Bit 1	J6-72	8.0	—
Output Data Bit 0	J6-73	8.0	—
CRY (Trickle Count Up)	J6-75	8.0	—
BRW (Trickle Count Down)	J6-74	8.0	—

* For input bit information to the DCH Word Counter, see Table 8-8.

NOTE: Word Counter output can be gated onto the I/O bus via the board's B connector by wire-wrapping the counter outputs to the **IDATA<0-15>** signals (see Table 8-7) and asserting J6-41 or J6-42 at the correct time.

Table 8-13 DCH Address Counter*

Signal	Pin	Fanout	Load
Clock Address Counter	J5-70	—	2.0
Load Address Counter	J5-45	—	0.6
Load Address Counter	J5-46	—	0.6
^**RESET2** (Address Counter)	J6-76	—	2.0
Enable Address Output	J6-43	—	4.0
Enable Address Output	J6-46	—	4.0
CRY (Trickle Count Up)	J6-78	8.0	—
BRW (Trickle Count Down)	J6-77	8.0	—

* For input bit information to the DCH Address Counter, see Table 8-8.

NOTE: Counter output is gated onto the I/O bus via the boards B connector by asserting J6-43 or J6-46 at the correct time.

Table 8-14 J5 Wire-Wrap Pins

Pin	Function	Signal	Fanout	Load
J5-1	DCH Control	Enable DCH SYNC FF	—	2.0
J5-2	I/O Control	^IOPULSE	16.0	—
J5-3	I/O Control	Mask Bit to Interrupt Disable FF	—	3.5
J5-4	I/O Control	Disable DONE FF	—	1.6
J5-5	I/O Control	Assert ^DATA15 on Interrupt Acknowledge	—	2.0
J5-6	I/O Control	^SDEVCODE	16.0	—
J5-7	I/O Control	^PDEVCODE	16.0	—
J5-8	I/O Control	^START	16.0	—
J5-9	I/O Control	^CLEAR	16.0	—
J5-10	I/O Control	INT REQ	14.0	—
J5-11	I/O Control	^DATA IN A	16.0	—
J5-12	I/O Control	^DATA OUT B	16.0	—
J5-13	Input Data Register	Shift CLK	—	2.0
J5-14	I/O Control	^DATA OUT A	16.0	—
J5-15	Input Data Register	Shift CLK	—	2.0
J5-16	I/O Control	^DATA IN C	16.0	—
J5-17	I/O Control	Enable ^SELD, ^SELB	—	2.0
J5-18	I/O Control	Enable Control Gates	—	1.6
J5-19	Output Data Register	Shift CLK	—	2.0
J5-20	I/O Control	DONE	10.0	—
J5-21	I/O Control	^DATA IN B	16.0	—
J5-22	I/O Control	^DATA OUT C	16.0	—
J5-24	Output Data Register	Shift CLK	—	2.0
J5-25	I/O Control	Reset DONE FF	—	0.7
J5-27	I/O Control	Set BUSY, Reset DONE	—	0.7
J5-28	I/O Control	BUSY	14.0	—
J5-29	DCH Control	DCHI	16.0	—
J5-30	DCH Control	DCHO	16.0	—
J5-31	I/O Control	Clock DONE FF	—	3.2
J5-32	DCH Control	DCH SYNC FF - ^Q	16.0	—
J5-33	DCH Control	DCHA	16.0	—
J5-34	DCH Control	DCHSEL	10.0	—
J5-35	DCH Control	Clear DCH SYNC	15.0	—
J5-36	I/O Control	Set DONE FF	—	1.6
J5-37	I/O Control	MSKO	16.0	—
J5-38	I/O Control	INT ACK	16.0	—
J5-39	I/O Control	RQENB	16.0	—
J5-40	DCH Control	DCH REQ	13.0	—
J5-41	I/O Control	^IO RESET	16.0	—
J5-42	DCH Control	DCH SYNC FF - Q	16.0	—
J5-43	DCH Control	Set DCH SYNC FF	—	2.0
J5-44	DCH Control	Clock DCH SYNC FF	—	3.5
J5-45	DCH Address Counter	Load Address Counter	—	0.6
J5-46	DCH Address Counter	Load Address Counter	—	0.6
J5-47	I/O Control	Enable ^INTPOUT	—	0.4
J5-48	DCH Word Counter	Load Word Counter	—	0.6
J5-49	DCH Word Counter	Load Word Counter	—	0.6
J5-50	Input Data Register	LDATAIN (Left Data In)	—	0.4

(continued)

Table 8-14 J5 Wire-Wrap Pins

Pin	Function	Signal	Fanout	Load
J5-51	Input Data Register	**Data bit 13**	—	0.4
J5-52	Input Data Register	**Data bit 12**	—	0.4
J5-53	Input Data Register	**^RESET**	—	0.7
J5-54	Input Data Register	**Data bit 14**	—	0.4
J5-55	Input Data Register	**Data bit 15**	—	0.4
J5-56	Input Data	**IDATA14**	6.0	—
J5-57	Input Data	**IDATA15**	6.0	—
J5-58	Input Data	**IDATA13**	6.0	—
J5-59	Input Data	**IDATA12**	6.0	—
J5-60	DCH Control	Assert **^EXTDCH**	—	2.0
J5-62	DCH Control	Enable DCH SYNC FF	—	2.0
J5-63	Input Data Register	**Data bit 8**	—	0.4
J5-64	Input Data Register	**Data bit 9**	—	0.4
J5-65	Input Data Register	**Data bit 10**	—	0.4
J5-66	Input Data Register	**Data bit 11**	—	0.4
J5-67	Input Data	**IDATA9**	6.0	—
J5-68	Input Data	**IDATA8**	6.0	—
J5-69	DCH Word Counter	Clock Word Counter	—	2.0
J5-70	DCH Address Counter	Clock Address Counter	—	2.0
J5-71	Input Data Register	**Data bit 4**	—	0.4
J5-72	Input Data Register	**Data bit 5**	—	0.4
J5-73	Input Data Register	**Data bit 6**	—	0.4
J5-74	Input Data Register	**Data bit 7**	—	0.4
J5-75	Input Data	**IDATA5**	6.0	—
J5-76	Input Data	**IDATA4**	6.0	—
J5-77	Input Data	**IDATA7**	6.0	—
J5-78	Input Data	**IDATA6**	6.0	—

(concluded)

Table 8-15 J6 Wire-Wrap Pins

Pin	Function	Signal	Fanout	Load
J6-1	Input Data Register	SHIFT LEFT	—	0.7
J6-2	Input Data Register	SHIFT RIGHT	—	0.7
J6-3	Input Data Register	Data bit 0	—	0.4
J6-4	Input Data Register	RDATAIN (Right Data IN)	—	0.4
J6-5	Input Data Register	Data bit 2	—	0.4
J6-6	Input Data Register	Data bit 1	—	0.4
J6-7	Input Data	IDATA1	6.0	—
J6-8	Input Data Register	Data bit 3	—	0.4
J6-9	Output Data Register	Data Left Input	—	0.4
J6-10	Input Data	IDATA0	6.0	—
J6-11	Output Data Register	Data bit 15	8.0	—
J6-12	Output Data Register	^RESET	—	0.7
J6-13	Output Data Register	Data bit 13	8.0	—
J6-14	Output Data Register	Data bit 14	8.0	—
J6-15	Output Data Register	Shift Left	—	0.7
J6-16	Output Data Register	Data bit 12	7.6	—
J6-17	DCH Word Counter	Output Data Bit 15	8.0	—
J6-18	Output Data	DATA14	16.0	—
J6-19	DCH Word Counter	Output Data Bit 13	8.0	—
J6-20	DCH Word Counter	Output Data Bit 14	8.0	—
J6-21	Output Data	DATA12	16.0	—
J6-22	DCH Word Counter	Output Data Bit 12	8.0	—
J6-23	Output Data	DATA15	16.0	—
J6-24	Output Data	DATA13	16.0	—
J6-25	Output Data	DATA8	16.0	—
J6-26	Output Data	DATA10	16.0	—
J6-27	Output Data Register	Data bit 10	8.0	—
J6-28	Output Data Register	Data bit 11	7.6	—
J6-29	Output Data Register	Data bit 9	8.0	—
J6-30	Output Data Register	Data bit 8	7.6	—
J6-31	DCH Word Counter	Output Data Bit 10	8.0	—
J6-32	Output Data Register	Shift Right	—	0.7
J6-33	DCH Word Counter	Output Data Bit 11	8.0	—
J6-34	DCH Word Counter	Output Data Bit 9	8.0	—
J6-35	Output Data	DATA9	16.0	—
J6-36	DCH Word Counter	Output Data Bit 8	8.0	—
J6-37	Input Data	IDATA11	6.0	—
J6-38	Output Data	DATA11	16.0	—
J6-39	Input Data	IDATA2	6.0	—
J6-40	Input Data	IDATA10	6.0	—
J6-41	Input Data	Enable Input Data	—	4.0
J6-42	Input Data	Enable Input Data	—	4.0
J6-43	DCH Control	Enable Address Gates	—	4.0
J6-44	DCH Word Counter	^RESET3 (Word Counter)	—	2.0
J6-45	Input Data	IDATA3	6.0	—
J6-46	DCH Control	Enable Address Gates	—	4.0
J6-47	DCH Control	M0	—	2.0
J6-48	DCH Control	Enable ^DCHR	—	2.0

(continued)

Table 8-15 J6 Wire-Wrap Pins

Pin	Function	Signal	Fanout	Load
J6-49	Output Data	DATA6	16.0	—
J6-50	Output Data	DATA7	16.0	—
J6-51	Output Data	DATA4	16.0	—
J6-52	Output Data	DATA5	16.0	—
J6-53	Output Data Register	Data bit 6	8.8	—
J6-54	Output Data Register	Data bit 7	7.6	—
J6-55	Output Data Register	Data bit 5	8.0	—
J6-56	Output Data Register	Data bit 4	7.6	—
J6-57	DCH Word Counter	Output Data Bit 7	8.0	—
J6-58	DCH Word Counter	Output Data Bit 6	8.0	—
J6-59	DCH Word Counter	Output Data Bit 4	8.0	—
J6-60	DCH Word Counter	Output Data Bit 5	8.0	—
J6-61	Output Data	DATA3	16.0	—
J6-62	Output Data	DATA1	16.0	—
J6-63	Output Data Register	Data Right Input	—	0.4
J6-64	Output Data	DATA0	16.0	—
J6-65	Output Data Register	Data bit 3	7.6	—
J6-66	Output Data Register	Data bit 2	8.0	—
J6-67	Output Data Register	Data bit 0	8.0	—
J6-68	Output Data Register	Data bit 1	8.0	—
J6-69	Output Data	DATA2	16.0	—
J6-70	DCH Word Counter	Output Data Bit 2	8.0	—
J6-71	DCH Word Counter	Output Data Bit 3	8.0	—
J6-72	DCH Word Counter	Output Data Bit 1	8.0	—
J6-73	DCH Word Counter	Output Data Bit 0	8.0	—
J6-74	DCH Word Counter	BRW (Trickle Count Down)	8.0	—
J6-75	DCH Word Counter	CRY (Trickle Count Up)	8.0	—
J6-76	DCH Address Counter	^RESET2 (Address Counter)	—	2.0
J6-77	DCH Address Counter	BRW (Trickle Count Down)	8.0	—
J6-78	DCH Address Counter	CRY (Trickle Count Up)	8.0	—

(concluded)

End of Chapter

Index

TIPS ORDERING PROCEDURES

TO ORDER

1. An order can be placed with the TIPS group in two ways:

   a) MAIL ORDER – Use the order form on the opposite page and fill in all requested information. Be sure to include shipping charges and local sales tax. If applicable, write in your tax exempt number in the space provided on the order form.

   Send your order form with payment to:

   Data General Corporation
   ATTN: Educational Services/TIPS G155
   4400 Computer Drive
   Westboro, MA 01581–9973

   b) TELEPHONE – Call TIPS at (508) 870–1600 for all orders that will be charged by credit card or paid for by purchase orders over $50.00. Operators are available from 8:30 AM to 5:00 PM EST.

METHOD OF PAYMENT

2. As a customer, you have several payment options:

   a) Purchase Order – Minimum of $50. If ordering by mail, a hard copy of the purchase order must accompany order.

   b) Check or Money Order – Make payable to Data General Corporation.

   c) Credit Card – A minimum order of $20 is required for Mastercard or Visa orders.

SHIPPING

3. To determine the charge for UPS shipping and handling, check the total quantity of units in your order and refer to the following chart:

Total Quantity	Shipping & Handling Charge
1–4 Units	$5.00
5–10 Units	$8.00
11–40 Units	$10.00
41–200 Units	$30.00
Over 200 Units	$100.00

If overnight or second day shipment is desired, this information should be indicated on the order form. A separate charge will be determined at time of shipment and added to your bill.

VOLUME DISCOUNTS

4. The TIPS discount schedule is based upon the total value of the order.

Order Amount	Discount
$1–$149.99	0%
$150–$499.99	10%
Over $500	20%

TERMS AND CONDITIONS

5. Read the TIPS terms and conditions on the reverse side of the order form carefully. These must be adhered to at all times.

DELIVERY

6. Allow at least two weeks for delivery.

RETURNS

7. Items ordered through the TIPS catalog may not be returned for credit.

8. Order discrepancies must be reported within 15 days of shipment date. Contact your TIPS Administrator at (508) 870–1600 to notify the TIPS department of any problems.

INTERNATIONAL ORDERS

9. Customers outside of the United States must obtain documentation from their local Data General Subsidiary or Representative. Any TIPS orders received by Data General U.S. Headquarters will be forwarded to the appropriate DG Subsidiary or Representative for processing.

TIPS ORDER FORM

Mail To: Data General Corporation
Attn: Educational Services/TIPS G155
4400 Computer Drive
Westboro, MA 01581 - 9973

BILL TO:	SHIP TO: (No P.O. Boxes - Complete Only If Different Address)
COMPANY NAME__________	COMPANY NAME__________
ATTN:__________	ATTN:__________
ADDRESS__________	ADDRESS (NO PO BOXES)__________
CITY__________	CITY__________
STATE__________ ZIP______	STATE__________ ZIP______

Priority Code __________ (See label on back of catalog)

__________ __________ ______ __________ ______

Authorized Signature of Buyer (Agrees to terms & conditions on reverse side) | Title | Date | Phone (Area Code) | Ext.

ORDER #	QTY	DESCRIPTION	UNIT PRICE	TOTAL PRICE

A SHIPPING & HANDLING

□ UPS	ADD
1-4 Items	$ 5.00
5-10 Items	$ 8.00
11-40 Items	$ 10.00
41-200 Items	$ 30.00
200+ Items	$100.00

Check for faster delivery

Additional charge to be determined at time of shipment and added to your bill.

□ UPS Blue Label (2 day shipping)

□ Red Label (overnight shipping)

B VOLUME DISCOUNTS

Order Amount	Save
$0 - $149.99	0%
$150 - $499.99	10%
Over $500.00	20%

Tax Exempt #
or Sales Tax
(If applicable)
____

ORDER TOTAL	
Less Discount See B	-
SUB TOTAL	
Your local* sales tax	+
Shipping and handling - See A	+
TOTAL - See C	

C PAYMENT METHOD

□ Purchase Order Attached ($50 minimum)
P.O. number is________. (Include hardcopy P.O.)

□ Check or Money Order Enclosed

□ Visa □ MasterCard ($20 minimum on credit cards)

Account Number □□□□□□□□□□□□□□□□

Expiration Date □□□□

____

Authorized Signature
(Credit card orders without signature and expiration date cannot be processed.)

THANK YOU FOR YOUR ORDER

PRICES SUBJECT TO CHANGE WITHOUT PRIOR NOTICE.
PLEASE ALLOW 2 WEEKS FOR DELIVERY.
NO REFUNDS NO RETURNS.

* Data General is required by law to collect applicable sales or use tax on all purchases shipped to states where DG maintains a place of business, which covers all 50 states. Please include your local taxes when determining the total value of your order. If you are uncertain about the correct tax amount, please call 508-870-1600.

Form 702
Rev. 8/87

DATA GENERAL CORPORATION TECHNICAL INFORMATION AND PUBLICATIONS SERVICE TERMS AND CONDITIONS

Data General Corporation ("DGC") provides its Technical Information and Publications Service (TIPS) solely in accordance with the following terms and conditions and more specifically to the Customer signing the Educational Services TIPS Order Form. These terms and conditions apply to all orders, telephone, telex, or mail. By accepting these products the Customer accepts and agrees to be bound by these terms and conditions.

1. CUSTOMER CERTIFICATION

Customer hereby certifies that it is the owner or lessee of the DGC equipment and/or licensee/sub-licensee of the software which is the subject matter of the publication(s) ordered hereunder.

2. TAXES

Customer shall be responsible for all taxes, including taxes paid or payable by DGC for products or services supplied under this Agreement, exclusive of taxes based on DGC's net income, unless Customer provides written proof of exemption.

3. DATA AND PROPRIETARY RIGHTS

Portions of the publications and materials supplied under this Agreement are proprietary and will be so marked. Customer shall abide by such markings. DGC retains for itself exclusively all proprietary rights (including manufacturing rights) in and to all designs, engineering details and other data pertaining to the products described in such publication. Licensed software materials are provided pursuant to the terms and conditions of the Program License Agreement (PLA) between the Customer and DGC and such PLA is made a part of and incorporated into this Agreement by reference. A copyright notice on any data by itself does not constitute or evidence a publication or public disclosure.

4. LIMITED MEDIA WARRANTY

DGC warrants the CLI Macros media, provided by DGC to the Customer under this Agreement, against physical defects for a period of ninety (90) days from the date of shipment by DGC. DGC will replace defective media at no charge to you, provided it is returned postage prepaid to DGC within the ninety (90) day warranty period. This shall be your exclusive remedy and DGC's sole obligation and liability for defective media. This limited media warranty does not apply if the media has been damaged by accident, abuse or misuse.

5. DISCLAIMER OF WARRANTY

EXCEPT FOR THE LIMITED MEDIA WARRANTY NOTED ABOVE, DGC MAKES NO WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY AND FITNESS FOR PARTICULAR PURPOSE ON ANY OF THE PUBLICATIONS, CLI MACROS OR MATERIALS SUPPLIED HEREUNDER.

6. LIMITATION OF LIABILITY

A. CUSTOMER AGREES THAT DGC'S LIABILITY, IF ANY, FOR DAMAGES, INCLUDING BUT NOT LIMITED TO LIABILITY ARISING OUT OF CONTRACT, NEGLIGENCE, STRICT LIABILITY IN TORT OR WARRANTY SHALL NOT EXCEED THE CHARGES PAID BY CUSTOMER FOR THE PARTICULAR PUBLICATION OR CLI MACRO INVOLVED. THIS LIMITATION OF LIABILITY SHALL NOT APPLY TO CLAIMS FOR PERSONAL INJURY CAUSED SOLELY BY DGC'S NEGLIGENCE. OTHER THAN THE CHARGES REFERENCED HEREIN, IN NO EVENT SHALL DGC BE LIABLE FOR ANY INCIDENTAL, INDIRECT, SPECIAL OR CONSEQUENTIAL DAMAGES WHATSOEVER, INCLUDING BUT NOT LIMITED TO LOST PROFITS AND DAMAGES RESULTING FROM LOSS OF USE, OR LOST DATA, OR DELIVERY DELAYS, EVEN IF DGC HAS BEEN ADVISED, KNEW OR SHOULD HAVE KNOWN OF THE POSSIBILITY THEREOF; OR FOR ANY CLAIM BY ANY THIRD PARTY.

B. ANY ACTION AGAINST DGC MUST BE COMMENCED WITHIN ONE (1) YEAR AFTER THE CAUSE OF ACTION ACCRUES.

7. GENERAL

A valid contract binding upon DGC will come into being only at the time of DGC's acceptance of the referenced Educational Services Order Form. Such contract is governed by the laws of the Commonwealth of Massachusetts, excluding its conflict of law rules. Such contract is not assignable. These terms and conditions constitute the entire agreement between the parties with respect to the subject matter hereof and supersedes all prior oral or written communications, agreements and understandings. These terms and conditions shall prevail notwithstanding any different, conflicting or additional terms and conditions which may appear on any order submitted by Customer. DGC hereby rejects all such different, conflicting, or additional terms.

8. IMPORTANT NOTICE REGARDING AOS/VS INTERNALS SERIES (ORDER #1865 & #1875)

Customer understands that information and material presented in the AOS/VS Internals Series documents may be specific to a particular revision of the product. Consequently user programs or systems based on this information and material may be revision-locked and may not function properly with prior or future revisions of the product. Therefore, Data General makes no representations as to the utility of this information and material beyond the current revision level which is the subject of the manual. Any use thereof by you or your company is at your own risk. Data General disclaims any liability arising from any such use and I and my company (Customer) hold Data General completely harmless therefrom.

Cut here and insert in binder spine pocket

Data General

Data General Corporation, Westboro, Massachusetts 01580