EK-DR11W-UG-004

DR11-W
Direct Memory Access Interface User's Guide

Prepared by Educational Services
of
Digital Equipment Corporation

4th Edition, June 1987

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

Printed in U.S.A.

This document was set on a DIGITAL DECset Integrated Publishing System.

- Class A Computing Devices:

Notice: This equipment generates, uses, and may emit radio frequency energy. The equipment has been type tested and found to comply with the limits for a Class A computing device pursuant to Subpart J of Part 15 of FCC Rules, which are designed to provide reasonable protection against such radio frequency interference when operated in a commercial environment. Operation of this equipment in a residential area may cause interference in which case the user at his own expense may be required to take measures to correct the interference.

The following are trademarks of Digital Equipment Corporation:

digital™	DECwriter	RSX
	DIBOL	Scholar
DEC	MASSBUS	ULTRIX
DECmate	PDP	UNIBUS
DECset	P/OS	VAX
DECsystem-10	Professional	VMS
DECSYSTEM-20	Rainbow	VT
DECUS	RSTS	Work Processor

CONTENTS

CHAPTER 3 USER DEVICE INTERFACING

CHAPTER 4 PROGRAMMING THE DR11-W

FIGURES

TABLES

PREFACE

The *DR11-W Direct Memory Access Interface User's Guide* describes the installation, testing, and use of the DR11-W in both the PDP-11 and VAX-11 environments.

Chapter 1 describes the DR11-W in general and cites its specifications.

Chapter 2 describes the configuration, installation, and testing of the DR11-W.

Chapter 3 describes the physical interface presented to a user device connected to the DR11-W.

Chapter 4 describes DR11-W programming in general, and the specific programming considerations which exist when the DR11-W is connected to a user device.

Chapter 5 describes the additional hardware and software considerations which exist when two DR11-Ws are cross-connected to form a high-speed, computer-to-computer data link.

Chapter 6 describes in detail those inner workings of the DR11-W which may be of concern to the system implementer.

Appendix A provides a listing of the Macro-32 source for the VAX/VMS XADRIVER.

The manual assumes that you have a basic knowledge of the UNIBUS as well as PDP-11 or VAX-11 assembly language. All programming examples are given using PDP-11 code since that should be intelligible to both the PDP-11 and the VAX-11 programmer.

The UNIBUS is described in detail in the *1985 UNIBUS Processor Handbook*; other details of the processors may be found in either the *PDP-11 Architecture Handbook* (EB-23657-18) or the *VAX-11 Architecture Handbook* (EB-26115-46).

This, the fourth edition, is all new and supersedes the previous three editions. A few terms have been redefined. Most are self-evident but two changes should be noted in particular.

Operating Modes

The first through third editions referred to three operating modes for the DR11-W:

1. Word mode
2. Block mode
3. Burst mode (either two-cycle or n-cycle)

In the fourth edition, block mode refers to the various DMA modes: one-cycle burst (also known as non-burst), two-cycle burst, and n-cycle burst modes. The fourth edition defines the following operating modes:

1. Word mode

    a. Using programmed I/O
    b. Using interrupt I/O

2. Block Mode

    a. Using one-cycle bursts
    b. Using two-cycle bursts
    c. Using n-cycle bursts

This form is in keeping with the rest of the DR11-W related documentation, in particular, the VAX/VMS XADRIVER description contained in the *Guide to Writing a Device Driver for VAX/VMS* (AA-Y511A-TE).

Burst Data Late

Additionally, all references to burst data late have been changed to burst release. This is to eliminate confusion with the data-late error (which a burst release does not represent).

CHAPTER 1
GENERAL INFORMATION

1.1 INTRODUCTION

The DR11-W is a general-purpose module used whenever a UNIBUS PDP-11 or VAX-11 computer must be interfaced to another device or computer and transfer parallel binary information. The DR11-W provides the logic to interface this binary data to the UNIBUS.

Data is transferred on the UNIBUS as 8-bit bytes or 16-bit words. The user device may connect to between 1 and 16 bits of this data. Separate lines are used for data input and data output (see Figure 1-1).

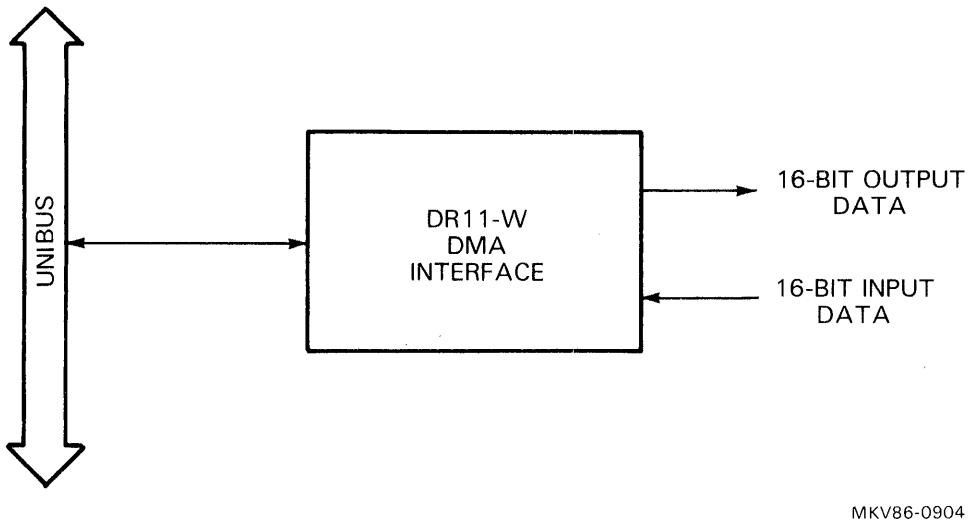

Figure 1-1 DR11-W Overview Block Diagram

Two DR11-Ws may be cross-connected to form a high-speed link between two UNIBUS computers. The DR11-W may also be cross-connected with the DRV11-B or DRV11-W, used on the Q-bus. Together, these devices allow any PDP-11, VAX-11, or MicroVAX systems to be linked together (see Figure 1-2).

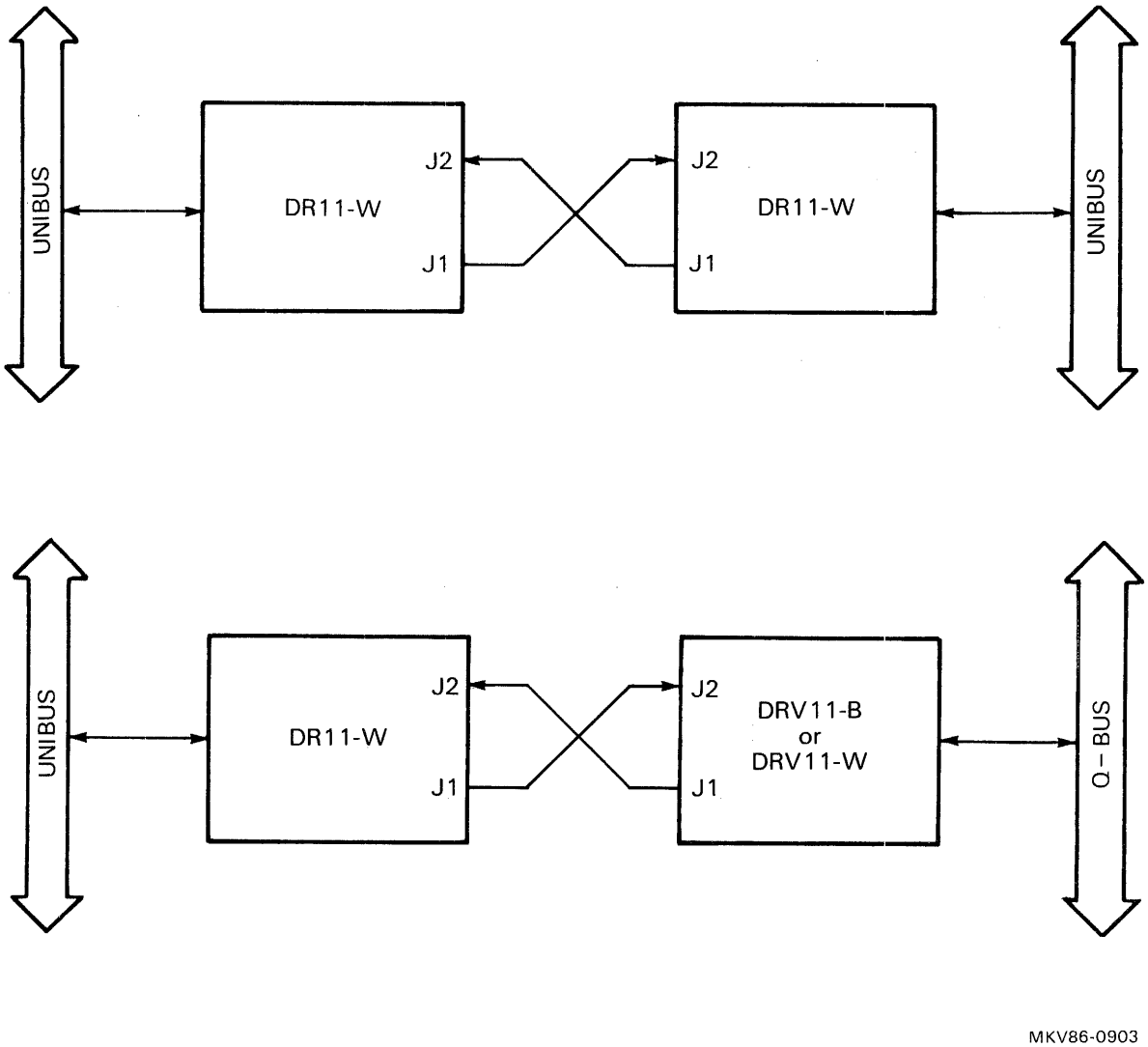

Figure 1-2 Overview Block Diagrams of DR11-W Links

1.2 MODULE LAYOUT

Figure 1-3 shows a DR11-W module. All of the user-interface features are called out.

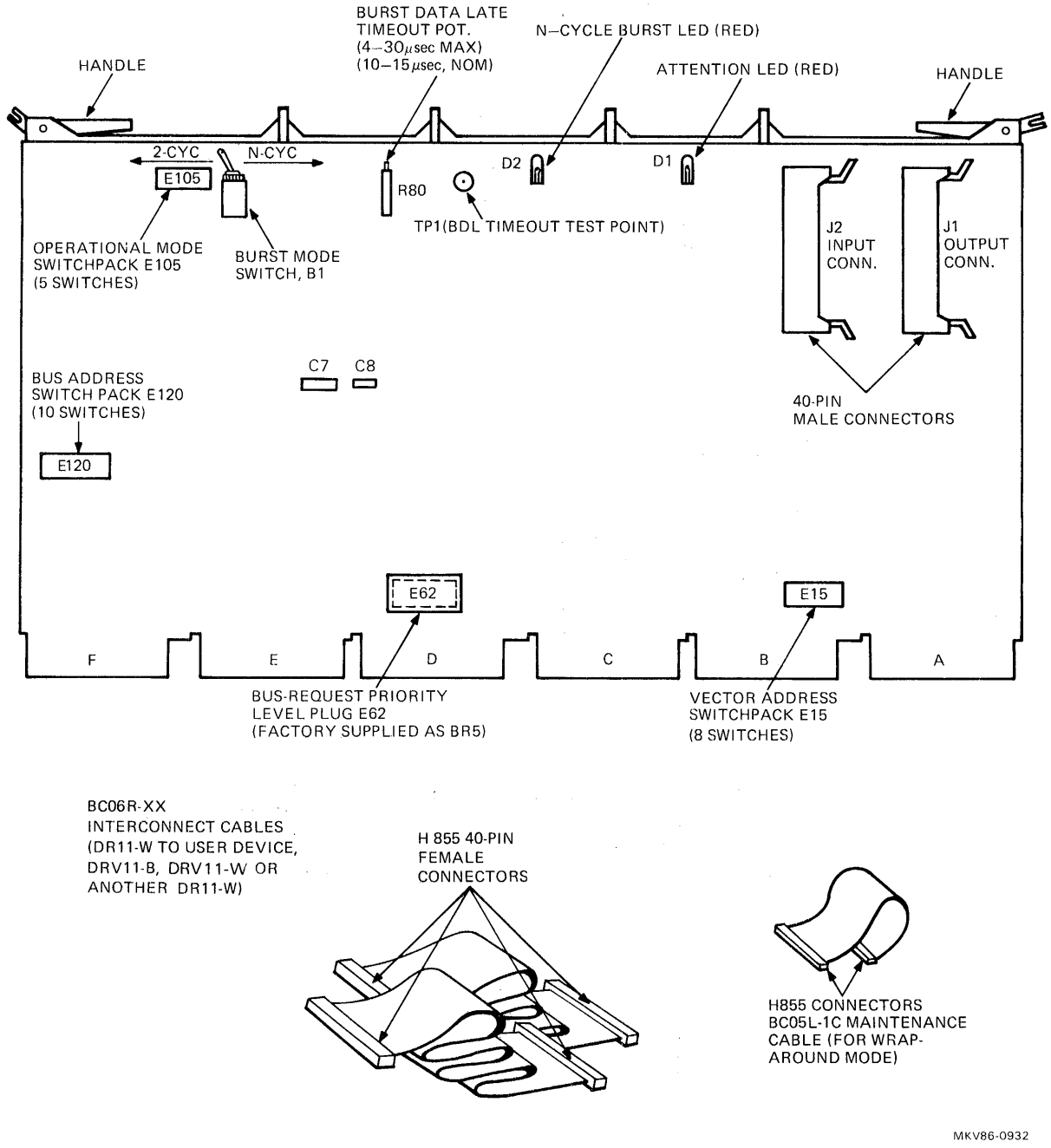

Figure 1-3 DR11-W Module Physical Layout

1.3 BASIC BLOCK DIAGRAM

The basic block diagram of the DR11-W is illustrated in Figure 1-4.

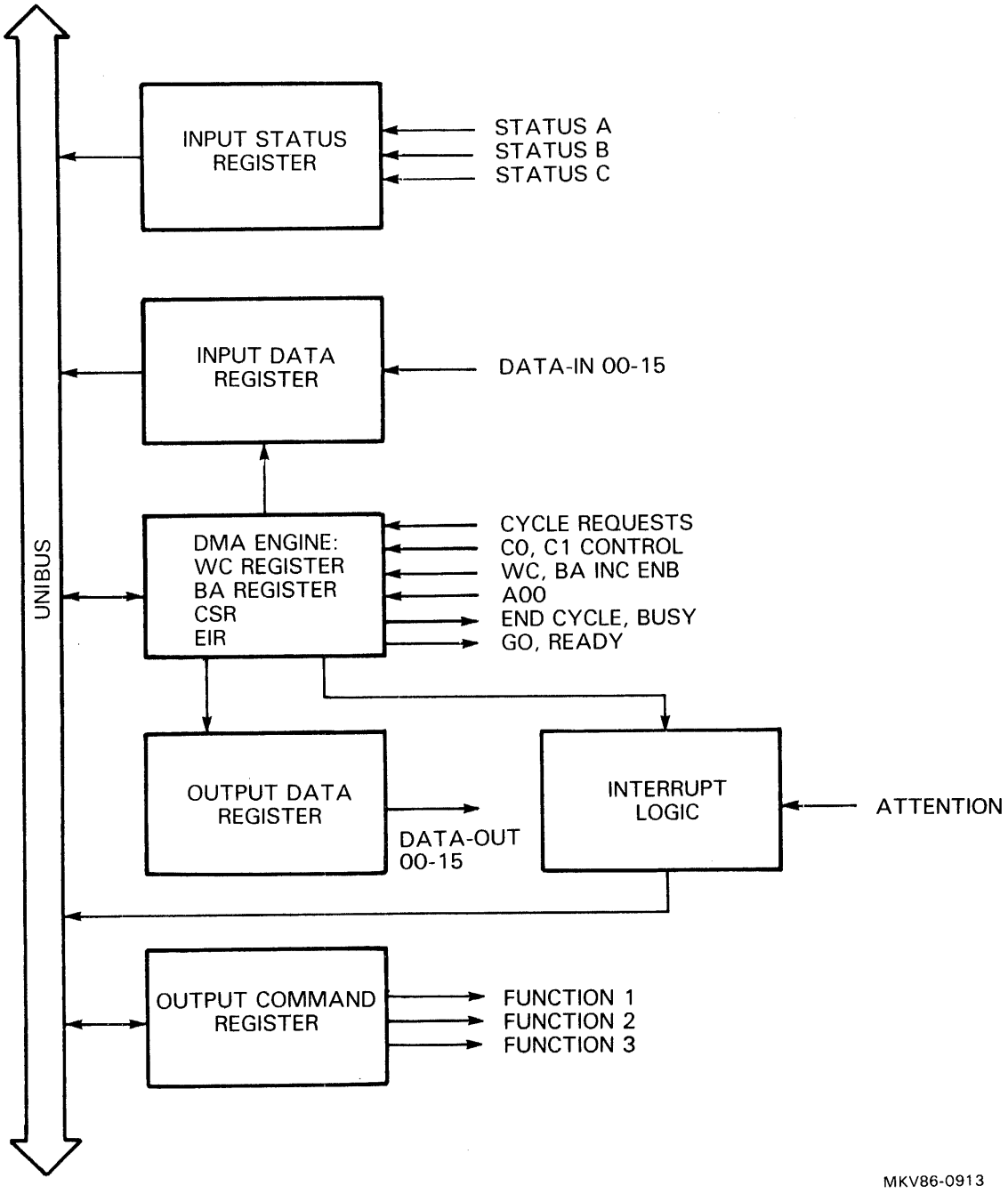

Figure 1-4 DR11-W Basic Block Diagram

1.4 DATA TRANSFER TECHNIQUES

The DR11-W may transfer data between the computer and the user device using:

- Programmed I/O
- Interrupt-driven I/O
- Direct memory access (DMA) I/O

The first two techniques are collectively referred to as word mode since the processor must participate in the transfer of every individual word. The third technique is referred to as block mode since the processor need only participate to set up the transfer of an entire block of data. These three techniques vary in speed and programming complexity. A typical application uses a mix of techniques.

1.4.1 Programmed I/O

Using this technique, a program simply moves data to the output data register or reads data from the input data register. Some means must be provided for synchronizing the movement of the data with the operation of the user device. This is often done using the function outputs and status inputs.

This technique can be quite fast and powerful, particularly for short transfers. However, it can consume up to 100 percent of the processor's time.

1.4.2 Interrupt-Driven I/O

Using this technique, the DR11-W transfers one word in or out each time an interrupt is generated by the user device. For relatively slow data rates, this causes much less processor overhead than programmed I/O, while maintaining the same flexibility.

Interrupt I/O typically has the lowest transfer rate of the three techniques.

1.4.3 Direct Memory Access I/O

This is the principal application of the DR11-W. Using this technique, the DR11-W is pointed at a block of the PDP-11 or VAX-11 memory. Then, without further interaction by the processor, the DR11-W can read words out of this memory and transfer them to the user device, or take words supplied by the user device and transfer them into memory. At the end of the block, an interrupt is usually generated.

For mid- to large-sized blocks of data, this is the fastest transfer method.

1.5 BURST MODES

The right to use the UNIBUS for a data transfer is decided by a priority-arbitration process. Before a DMA device (such as the DR11-W) can transfer data on the UNIBUS, it must first win this priority arbitration. While the UNIBUS arbitration process is concurrent with data transfers (so that the overall throughput of the UNIBUS is improved), the need to arbitrate does increase the latency for any one UNIBUS device.

Normally, a DMA device arbitrates for use of the UNIBUS, transfers one word of data, and then releases the UNIBUS (see Figure 1-5).

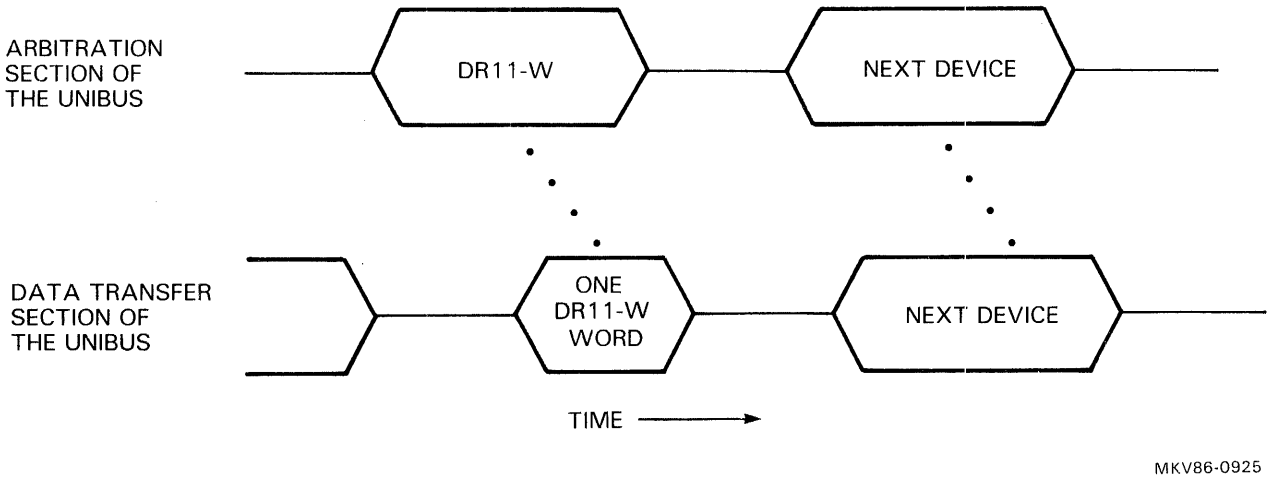

Figure 1-5 Single-Cycle UNIBUS Transfer Timing Block Diagram

If a device has a lot of data to move quickly, it often helps to transfer more than one word each time the device becomes the master of the UNIBUS. In DR11-W terminology, this is called burst mode. The DR11-W hardware supports the transferring of two words per UNIBUS mastership. This is called two-cycle burst mode (see Figure 1-6). The DR11-W can also be configured to transfer an unlimited number of words per UNIBUS mastership. This is referred to as n-cycle burst mode (see Figure 1-7). In n-cycle burst mode, the DR11-W can use the entire UNIBUS forever, to the exclusion of all other prospective users. This mode is sometimes referred to as bus hog mode.

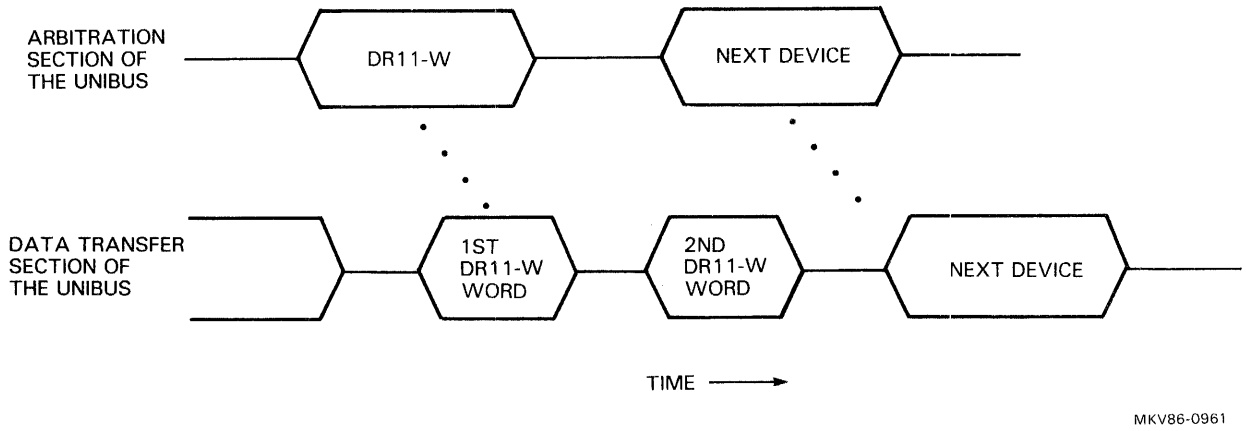

Figure 1-6 Two-Cycle UNIBUS Transfer Timing Block Diagram

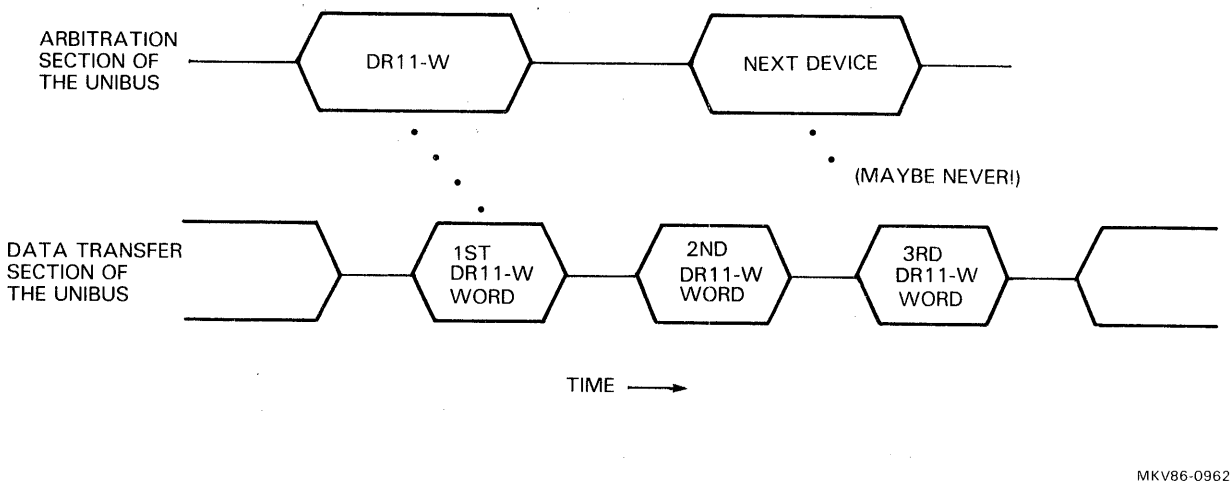

Figure 1-7 N-Cycle UNIBUS Transfer Timing Block Diagram

Table 1-1 DR11-W DMA Speed in Various Computers

Configuration	Bandwidth in Kilobytes/Second Burst Size 1	2
PDP-11/34 with MS11-L Memory:		
DATI (reading memory)	993	1192
DATO (writing memory words)	1394	1748
DATOB(writing memory bytes)	697	874
PDP-11/44 with MS11-P Memory:		
DATI (reading memory)	936	1057
DATO (writing memory words)	1192	1598
DATOB(writing memory bytes)	607	728
PDP-11/44 with MS11-M Memory:		
DATI (reading memory)	943	1032
DATO (writing memory words)	1202	1394
DATOB(writing memory bytes)	601	697
PDP-11/84 with MSV11-R Memory, UBMAP Off or DMA Cache Off:		
DATI (reading memory)	799	936
DATO (writing memory words)	1024	1394
DATOB(writing memory bytes)	512	697
PDP-11/84 with MSV11-R Memory, UBMAP and DMA Cache On:		
DATI (reading memory)	862	1024
DATO (writing memory words)	1024	1394
DATOB(writing memory bytes)	512	697
VAX-11/780, Direct Data Path (DDP):		
DATI (reading memory)	427	479
DATO (writing memory words)	528	602
DATOB(writing memory bytes)	264	301

1.6 DATA TRANSFER RATES TO USER DEVICES

Direct memory access programming techniques generally provide the fastest possible transfer rate. In this mode, the maximum speed of the DR11-W is largely determined by the speed of the memory that the DR11-W is accessing.

Table 1-1 documents the fastest observed rates on various UNIBUS processor and memory combinations, using optimum programming and the fastest possible user device.

NOTE

The speeds listed in Table 1-1 are actual speeds measured on hardware systems – your system may vary ±5% or more from the values shown in the table.

1.7 DATA TRANSFER RATES ACROSS A LINK

When operating as a cross-connected link with no additional buffering, the transfer rate is about one-half of the rate obtained by averaging the speed of each individual end. The exact formula is indicated below. For example, if the link is reading words from a PDP-11/24 and writing them to a PDP-11/44, the maximum transfer rate is:

$$R_{LINK} = \frac{1}{\frac{1}{R_{PDP\text{-}11/24\ READ}} + \frac{1}{R_{PDP\text{-}11/44\ WRITE}}}$$

MKV86-0963

1.8 SPECIFICATIONS

The general specifications for the DR11-W are listed below.

- Data format presented to the user:

  Binary parallel words of up to 16 bits

- Data format presented to the UNIBUS:

  Binary parallel 8-bit bytes or 16-bit words

- Transfer methods:

  Programmed I/O

  Interrupt-driven I/O

  Direct memory access I/O

- UNIBUS characteristics:

  Capable of DMA to the full 256K byte UNIBUS address range

  Maximum single transfer of 64K bytes or words

  All UNIBUS data cycles supported:

  - DATI (read)
  - DATO (write word)
  - DATOB (write byte)
  - DATIP (read with write intent)

  Three jumper (capacitor) selectable time-out values:

  - Twelve microseconds
  - Thirty-three microseconds
  - Forty-five microseconds (as shipped from factory)

  Four plug-selectable UNIBUS interrupt priority levels:

  - BR7
  - BR6
  - BR5 (as shipped from factory)
  - BR4

  Three user-selectable burst DMA modes:

  - One word transfers
  - Two word bursts
  - N word bursts

  Automatic release of UNIBUS if transfers stop in mid-burst

- User interface characteristics:

  Two, 40-pin, flat-cable connectors:

  - Twenty-five output lines, plus grounds
  - Twenty-eight input lines, plus grounds

  All signals driven with UNIBUS drivers or received with UNIBUS receivers for maximum noise immunity

  All signals terminated into 120 ohms and 3.3 volts

  Sixteen-bit parallel data inputs and outputs

  Three-bit parallel control outputs

  Three-bit parallel status inputs

  User-selectable logic-high or logic-low busy signal

- Physical characteristics:

  Single hex module – 21.6 cm × 38.1 cm (8.5 in × 15.0 in)

  Powered by 3.7 A (nominal) of +5 Vdc

  Temperature range:

  - 5° to 50°C (41° to 122°F) operating
  - −40° to 66°C (−40° to 151°F) storage

  Humidity range:

  - 10 to 90%, allowing no condensation

  Two indicator LEDs:

  - ATTN (interrupt)
  - N-cycle burst transfer in progress

- Programming characteristics:

  Six registers occupying four addresses in the I/O page:

  - Bus address register
  - Word (or byte) count register
  - Control and status register
  - Error and information register
  - Input data buffer register
  - Output data buffer register

  Some program compatibility with DR11-B

  Errors detected by the DR11-W:

  - UNIBUS time-out
  - Memory parity error
  - User data overrun/underrun error
  - Power failure during transfer

- Software available:

  PDP-11 stand-alone diagnostic program – CZDRL

  PDP-11 stand-alone link diagnostic – CZDRK

  VAX-11 stand-alone diagnostic program (includes link tests) – ESDRE

  VAX-11 on-line confidence check – ESDRB

  VAX/VMS sample device driver – XADRIVER

- Related documentation:

  DR11-W Field Maintenance Print Set (MP00693)

  PDP-11 UNIBUS Processor Handbook (EB-26077-41)

  PDP-11 Architecture Handbook (EB-23657-18)

  RSX11-M Guide to Writing an I/O Driver (AA-2600E-TC)

  RSX11-M+ Guide to Writing an I/O Driver (AA-H267B-TC)

  VAX-11 Architecture Handbook (EB-26115-46)

  Guide to Writing a Device Driver for VAX/VMS (AA-Y511A-TE)

CHAPTER 2
SETUP, INSTALLATION, AND TESTING

2.1 GENERAL

The DR11-W is installed and tested like most other modules. Off-line diagnostic programs are provided for both the PDP-11 and VAX-11 families of processors. In addition, the VAX-11 family provides an on-line confidence check program, which uses the VMS sample driver program (XADRIVER).

2.2 UNPACKING

The DR11-W may have been factory installed in your system. If so, go to Section 2.6. If the DR11-W has not already been installed in your system, it is shipped in a corrugated cardboard carton. To unpack the DR11-W, perform the following procedure.

- Check that the carton is sealed and undamaged. Report any damage to Digital Equipment Corporation or to the shipping carrier.
- Remove the sealing tape that holds the carton closed. Be careful not to damage the module as you do this.
- Remove the DR11-W module from the carton, keeping it in its anti-static bag. Do not remove the module from the anti-static bag until you are at an anti-static workstation.
- Inspect the DR11-W module for visible damage. Report any damage to Digital Equipment Corporation.

2.3 ANTI-STATIC PRECAUTIONS

Like most computer modules, the DR11-W contains components which can be damaged by static electricity. As long as the module remains inside its plastic bag, it is protected.

Outside of its protective plastic bag, the module should only be handled at an approved static-free workstation. A wrist strap connected to earth ground through a safety resistor should be worn by the user whenever handling the DR11-W.

2.4 SETUP

The DR11-W must be set up for each individual application. For each DR11-W module in the system, this setup consists of:

- Selecting the UNIBUS address of the registers.
- Selecting the interrupt vector address.
- Setting the E105 switches.
- Setting the burst-size toggle switch.

Three other setups may or may not matter, depending upon the application:

- Selecting the UNIBUS time-out value.
- Setting the burst release time-out value.
- Selecting the UNIBUS interrupt priority.

2.4.1 Selecting the UNIBUS Address

Each DR11-W in the system contains six registers which the software uses to control the operation of the DR11-W. Every DR11-W in the system must have a unique address for this group of registers and this address must match the address expected by the software. This address is selected on each DR11-W module by the E120 switchpack on that module. See Figure 1-3 for the location of E120.

The address of the DR11-W is actually the address of the first register in the group of six. The other registers follow in ascending order. By convention, the first DR11-W in the system is assigned to UNIBUS address 772 410. This is actually the address of the DR11-W word count register. For this DR11-W, the bus address register is found at address 772 412, and so forth.

Table 2-1 documents the relationship between the E120 switches and the address assigned to the DR11-W. Note that each switch selects one bit of the UNIBUS address that this particular DR11-W will respond to. A switch set to OFF corresponds to a 1 in the UNIBUS address. The conventional settings for the first two DR11-Ws are also shown. Additional DR11-Ws are assigned addresses from the floating address space (as shown in the PDP-11 and VAX-11 architecture handbooks).

Table 2-1 Address Selection

UNIBUS Address Bit	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
E120 Switch						**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**			
First DR11-W (772 410)						–	On	–	On	–	On	On	On	On	–			
Second DR11-W (772 430)						–	On	–	On	–	On	On	On	–	–			

Key: On = Switch ON
– = Switch OFF

2.4.2 Selecting the Interrupt Vector Address

Each DR11-W in the system also uses one unique address that contains a pointer to the interrupt service routine for that DR11-W, if any exists. By convention, these vector addresses are also unique to each DR11-W, although it is not mandatory that they be so. Once again, this vector address must match that expected by the software. The vector address is selected on each DR11-W module by the E15 switchpack on that module. See Figure 1-3 for the location of E15.

Only the first DR11-W in the system is assigned a fixed vector address. By convention, it is UNIBUS address 000 124. Any other DR11-Ws are assigned vectors from the floating vector space (also described in the PDP-11 and VAX-11 architecture handbooks).

Table 2-2 documents the relationship between the E15 switches and the interrupt vector address assigned to the DR11-W. Note that each switch selects one bit of the interrupt vector address assigned to the DR11-W. A switch set to OFF corresponds to a 1 in the interrupt vector address. Switch 1 of the E15 switchpack is used in link-mode applications. In link mode, switch 1 should be ON (closed); otherwise switch 1 should be OFF (open).

Table 2-2 Interrupt Vector Address Selection

Interrupt Vector Address Bit	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
E15 Switch								**8**	**7**	**6**	**5**	**4**	**3**	**2**		
First DR11-W (000 124)								On	On	–	On	–	On	–		

Key: On = Switch ON
– = Switch OFF

2.4.3 Setting the E105 Switchpack

The E105 switchpack selects three independent features:

- The polarity of the BUSY signal.
- Whether or not UNIBUS address bit ⟨00⟩ is suppressed.
- Whether or not the error and information register is suppressed.

Each of these features is described on subsequent pages.

2.4.3.1 Selecting the Correct BUSY Signal – Three switches on the E105 switchpack select from three different versions of the DR11-W BUSY signal. Figure 1-3 shows the location of E105 and Table 2-3 lists the settings of switches S1, S2, and S3.

For connection to a user device, either the BUSY H or BUSY L signal can be selected, as required by the particular user device.

For connection to another DR11-W, the BUSY L signal must be selected. Whenever the cable wrap-around test is performed, the BUSY L signal setting must also be used.

Finally, special timing is required for the DR11-W to participate as one end of a DR11-W to DRV11-B, or DR11-W to DRV11-W link. This special timing is provided by the third setting.

2.4.3.2 Setting UNIBUS Address Bit ⟨00⟩ Suppression – By itself, the DR11-W is capable of performing only word transfers on the UNIBUS. A small amount of additional logic within a user device allows the DR11-W to transfer bytes on the UNIBUS.

Table 2-3 Busy Signal Selection

Busy Signal	E105 switches 1	2	3	Use
BUSY H	On	Off	Off	Asserts the busy signal as a logic-high signal.
BUSY L	Off	On	Off	Asserts the busy signal as a logic-low signal. This setting is always used for DR11-W to DR11-W links and cable wrap-around testing.
D7 CYC INH(1) L	Off	Off	On	Asserts the busy signal using the particular timing required by the DRV11-B. This setting is always used for DR11-W to DRV11-B or DRV11-W links.

Memories attached to the UNIBUS are always word-oriented. During UNIBUS DATOB (write byte) operations, these memories use UNIBUS address line 00 (A00) to indicate which of the two bytes of the memory word should be updated. During all other UNIBUS operations, A00 should be ignored by the memory. A00 is directly supplied by the user device, not the DR11-W.

If you are using a memory which fails to ignore A00 for word operations, the A00 H input to the DR11-W may be disabled by placing switch 4 of the E105 switchpack in the ON position. See Figure 1-3 for the location of E105 and Table 2-4 for the switch settings. Normally, E105-4 is left in the OFF position.

Table 2-4 A00 Suppression

UNIBUS Address Bit <00>	E105 Switch 4
Enabled	Off
Forced to 0	On

If you are running the cable wrap-around diagnostics, E105-4 must be in the OFF position.

If you are connecting your DR11-W as part of a DR11-W to DR11-W (or DRV11-B or DRV11-W) link, you may leave E105-4 in either position. Placing the switch in the OFF position allows bus address bit ⟨00⟩ to follow the status of the ready bit within the DR11-W at the far end of the link. This allows the program at the near end of the link to tell whether the DR11-W at the far end of the link is ready simply by reading BAR ⟨00⟩. Placing the switch in the ON position disables this indication.

2.4.3.3 Selecting Error and Information Register (EIR) Suppression – Certain older software packages are not programmed to handle the error and information register of the DR11-W. Setting switch 5 of E105 to OFF allows this register to be turned off (that is, it becomes invisible to the program). See Figure 1-3 for the location of E105 and Table 2-5 for the switch settings.

Table 2-5 Error and Information Register Suppression

EIR	E105 Switch 5
Suppressed	Off
Enabled	On

Operation with switch 5 off is loosely named DR11-B mode, since the DR11-B does not contain an error and information register. This position, however, does not provide exact software compatibility with the DR11-B.

Setting switch 5 on E105 to ON allows this register to be used by the program. This is named DR11-W modes. It is recommended that all new software be developed to run in DR11-W mode.

2.4.4 Setting the Burst-Size Toggle Switch

Arbitrating for the use of the UNIBUS requires some finite time, so the UNIBUS specification allows a device to transfer more than one word each time it becomes the UNIBUS master. Without additional (external) logic, the DR11-W is capable of transferring one, two, or an unlimited number of words each time it becomes the master of the UNIBUS. This is referred to as the burst size.

When the DR11-W is transferring one word per UNIBUS mastership, it is said to be operating in one-cycle or non-burst mode.

When the DR11-W is transferring more than one word per UNIBUS mastership, it is said to be operating in burst mode. The DR11-W internal logic implements two distinct types of burst mode:

- Two-cycle burst mode (two transfers per UNIBUS mastership).
- N-cycle burst mode (unlimited transfers per UNIBUS mastership).

The user device selects between burst and non-burst mode. If burst mode is selected, a toggle switch on the DR11-W further selects between two-cycle and n-cycle modes. See Figure 1-3 for the switch location and Table 2-6 for the switch settings. Placing the switch handle towards the edge of the module selects two-cycle burst. Placing the handle towards the middle of the module selects n-cycle burst.

NOTE

Allowing the DR11-W to transfer an unlimited number of words may lock out all other users of the UNIBUS, so n-cycle mode is also known as bus hog mode. For this reason, n-cycle bursts are not supported on the VAX-11/780, VAX-11/782, VAX-11/785, VAX 8600, and VAX8650. However, n-cycle bursts will operate under certain restricted conditions (for example, if the DR11-W is the only device connected to the UNIBUS adapter and the UNIBUS).

Table 2-6 Burst Size Selection

Burst Size	Toggle Switch Handle Towards:
Two-Cycle	Edge of board
N-Cycle	Middle of board

2.4.5 Selecting the UNIBUS Time-Out Value

The UNIBUS is an asynchronous bus. This means that transactions on the UNIBUS are not synchronized by any master clock. Within certain limits, a device can operate as quickly or as slowly as necessary.

All UNIBUS interfaces which perform direct memory access must contain a feature known as UNIBUS slave-sync time-out. This feature sets the maximum time that the interface will wait for a slave device to respond on the UNIBUS. Without it, accessing a non-existent device could stall the UNIBUS indefinitely.

The DR11-W allows you to select from three different values of slave-sync time-out. This is done by optionally cutting out either capacitor C7 or capacitor C8. See Figure 1-3 for C7 and C8 locations and Table 2-7 for their values. The DR11-W is shipped set to the longest setting (that is, both capacitors installed). Normally, C8 should be removed (so as to use the shortest setting). Systems where the DR11-W is connected to a VAXBI bus through a DWBUA require the middle setting. Finally, if the system makes use of multiported memory, it may be necessary to select one of the longer time-out settings in order to avoid a false indication of a non-existent memory (NXM) error.

If the three supplied settings do not provide a long enough time-out, you can add a larger capacitor to the DR11-W module, parallel with either C7 or C8.

Table 2-7 Slave Sync Time-out Selection

Timeout in Microseconds	Used With	C8	C7
<Unused>	Nothing	Out	Out
12	Most systems	Out	In
33	VAXBI systems	In	Out
45	Multiport memory	In	In

NOTE

Use the shortest setting which provides error-free operation.

If the system is a VAX-11/780, VAX-11/782, VAX-11/785, VAX 8600, or VAX 8650 the time-out value must be less than 50 microseconds. Longer values will cause the VAX UNIBUS adapter to falsely report a hung UNIBUS.

2.4.6 Selecting the Burst Release Time-Out Value

The DR11-W is capable of transferring one, two, or n words each time it becomes the master of the UNIBUS. This is referred to as the burst size.

Burst mode is used to increase the overall UNIBUS throughput. It is only effective if the user device can deliver data rapidly (so that the UNIBUS does not sit idle while the DR11-W waits for the user's data).

If the DR11-W has become the UNIBUS master and has transferred at least one word of a two- or n-cycle burst, a timer is started. If the user device supplies the next word of data before the timer expires, the timer is started again. If, however, the user device fails to supply the data and the timer expires, then the DR11-W releases the UNIBUS.

This situation is harmless. The time-out ensures that the DR11-W does not continue to occupy the UNIBUS if the user device fails or momentarily stalls.

The burst release timer is adjustable by means of a trimpot, R80. See Figure 1-3 for the R80 location and Table 2-8 for timing adjustments. The minimum setting of the timer is 4 microseconds while the maximum setting is 30 microseconds. Like the slave-sync timer, additional capacitance can be added in parallel with the timing capacitor (C9), if needed.

Both the PDP-11 and VAX-11 off-line diagnostic programs contain a specific routine used to adjust the burst-release timer. For the VAX system, this is done using EVDRE Test 32. The correct command to invoke the test is:

```
DS> STA/SEC:BURST
```

For the PDP-11 system, this is done using CZDRL routine BRSTDL, invoked by answering YES to the question asking whether you wish to do burst data-late calibration.

In both cases, an oscilloscope should be connected to test point 1 (TP1). See Figure 1-3 for the location of TP1. The burst release time is the time during which the TP1 waveform is logic-low. Figure 2-1 shows the waveform viewed with an oscilloscope connected to TP1.

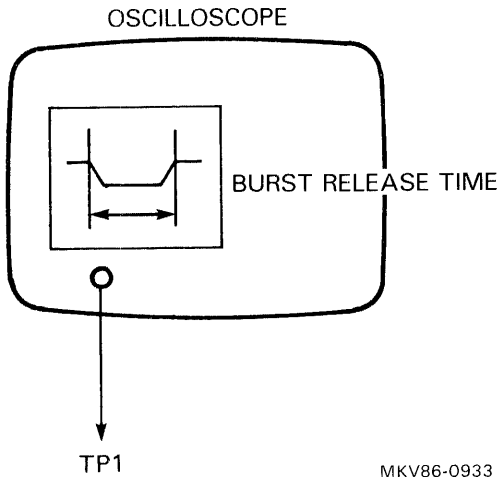

Figure 2-1 Burst Release Time Waveform

2.4.7 Selecting the UNIBUS Interrupt Priority

Every device on the UNIBUS that can generate interrupts has a specific interrupt priority level. The overall priority scheme is described in the UNIBUS chapter of the *PDP-11 Architecture Handbook*. In general, it is a combination of the device's bus request level (BR level) and the device's position along the UNIBUS.

The specific BR level of the DR11-W is set by a jumper plug installed at E62. See Figure 1-3 for the location of the BR plug. The DR11-W is shipped from the factory with a BR5 plug installed. Other plugs may be ordered and installed to select BR7, BR6, or BR4. BR7 is the highest priority; BR4 is the lowest priority (see Table 2-9).

2.5 INSTALLATION AND CHECKOUT

Once the module setup is complete, the DR11-W may be installed in the system. Before installing, check that:

- A hex SPC slot is available in which to mount the module.
- At least 3.7 A of +5 Vdc are available from the power supply powering that SPC slot.
- The NPG jumper wire has been removed from backplane pins CA1 and CB1 of that slot. This is usually a short blue wire. It may already have been removed.
- The grant continuity card (G727, G7270, or G7273) has been removed from that slot.

The system that you are installing the DR11-W in may or may not be FCC compliant. (The United States Federal Communications Commision has established rules and regulations regarding the amount of radio and television interference which a computer may legally emit.) The installation of the DR11-W into your system varies depending upon whether or not your system is FCC compliant.

Table 2-8 Burst Release Timer Adjustment

Timer (Microseconds)	R80 Position
4	Fully CCW
30	Fully CW

Table 2-9 BR Level Plug Part Numbers

BR Level	Plug Part Number	Other Devices Normally at This Level
BR7	54-08782	None
BR6	54-08780	Real-time clocks
BR5 (standard)	54-08778	Disks, tapes, multi-line synchronous comm
BR4	54-08776	Single-line asynchronous comm, printers

2.5.1 Installing the DR11-W in a Compliant System

If your system is already FCC compliant, you must install the DR11-W in such a fashion so as to maintain this compliance. This is done by routing the DR11-W user cables through a bulkhead panel at the rear of the mounting cabinet (see Figure 2-2). This bulkhead panel filters the DR11-W signal lines in order to contain the high frequencies within the computer cabinet. The bulkhead connector also provides a more convenient place to connect the user cables, resulting in a neater, more maintainable installation.

The interface presented at the outside of the bulkhead panel is identical to the interface presented at the DR11-W module itself.

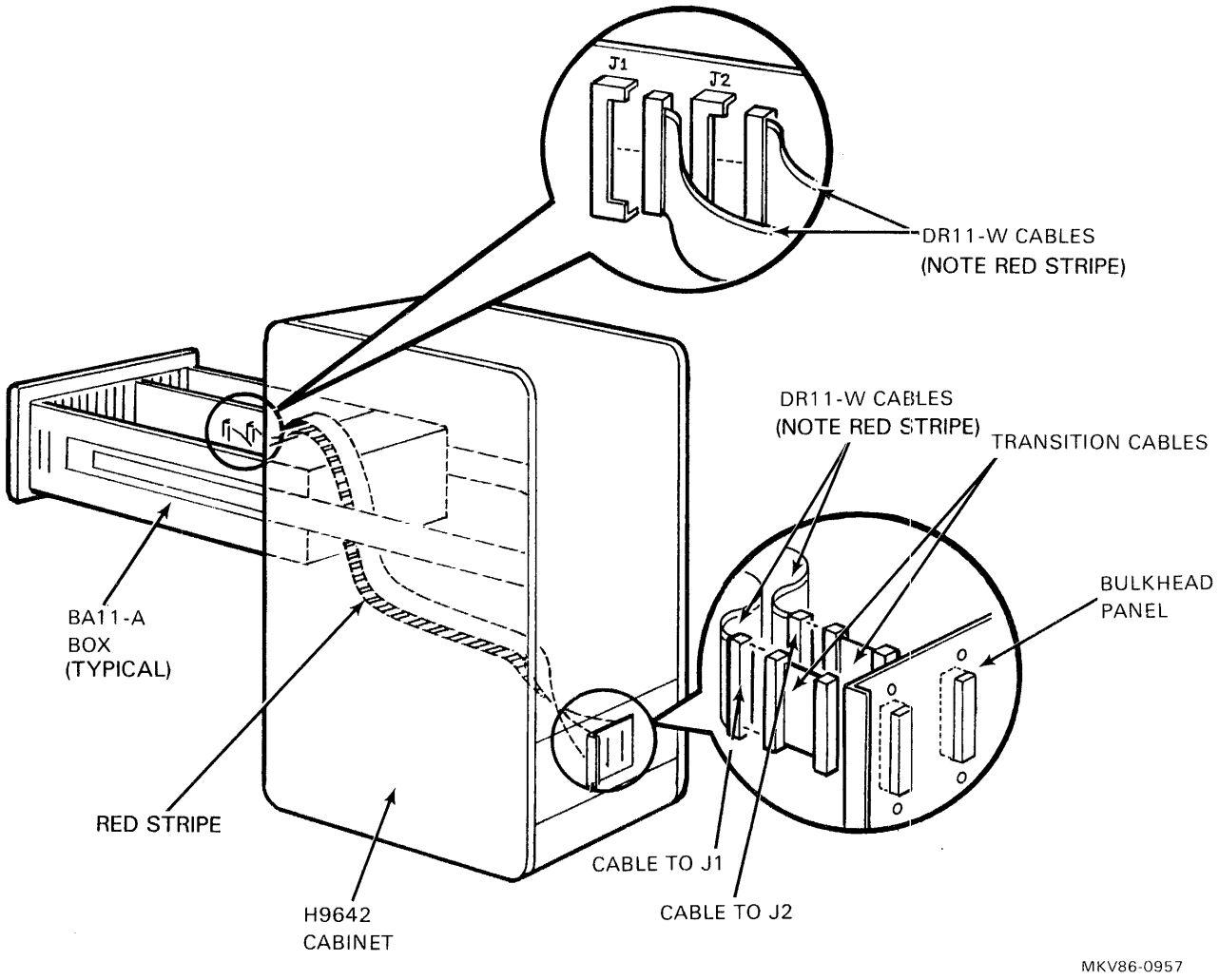

Figure 2-2 Installing the DR11-W in an FCC-Compliant System (Showing a PDP-11/24 or a PDP-11/44 and a BA11-A Box in an H9642 Cabinet)

2.5.2 Mounting the Bulkhead in the Existing Picture Frame

Locate an unused opening in the picture frame. Remove the screws which hold the cover plate in place, then remove the cover plate itself.

Ensure that both the picture frame opening and the bulkhead connector that you are about to install are each clean and free of corrosion.

Dress the BC06-R cables (which connect to the DR11-W) out through the opening in the picture frame. Connect each cable to an adapter (part number 70-21988), then connect the adapter to one of the connectors in the bulkhead panel.

When you have connected both cables and adapters to the bulkhead panel, and have verified that the connections are correct and secure, place the bulkhead panel in the opening and install and tighten the mounting screw. Ensure that the screws are tight (so that the best possible ground connection is made).

2.5.3 Installing the DR11-W in a Non-Compliant System

If your system is not FCC compliant, you can still take advantage of the convenience gained by use of the bulkhead panel (see Figure 2-3). In this case, the system may not already contain the picture frame which holds the bulkhead panels. Figure 2-3 shows the use of a single-hole picture frame which allows the bulkhead to be mounted. This **does not** make the system FCC-compliant.

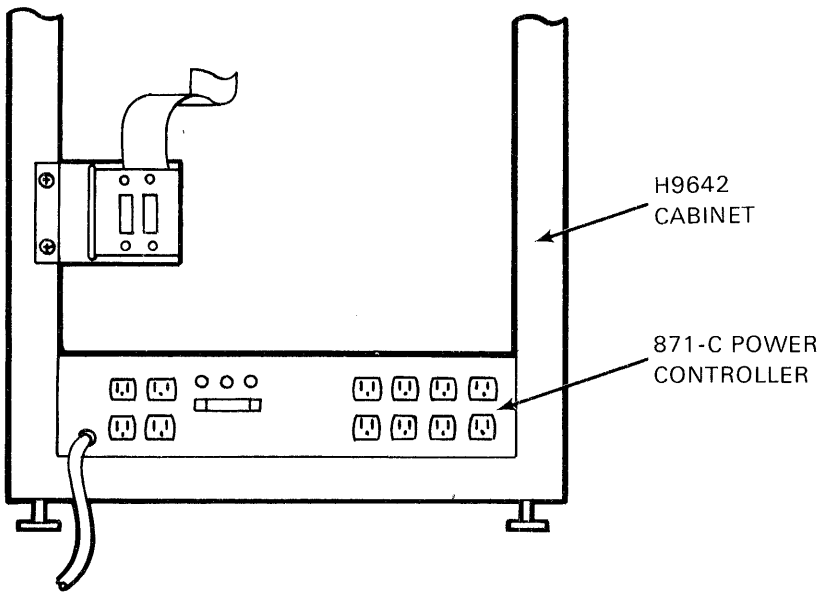

Figure 2-3 Installing the DR11-W in a Non-Compliant System (H9642 Cabinet, 871-C Power Controller, and Add-On Picture Frame)

Once again, the interface presented at the outside of the bulkhead panel is identical to the interface presented at the DR11-W module itself.

2.5.4 Installation Without the Bulkhead Panel

It is also possible to install the DR11-W without using the bulkhead panel. In this case, the cables are run directly from the connectors on the DR11-W module to the user device (see Figure 2-4).

This is only possible if your system is exempt from the FCC regulations.

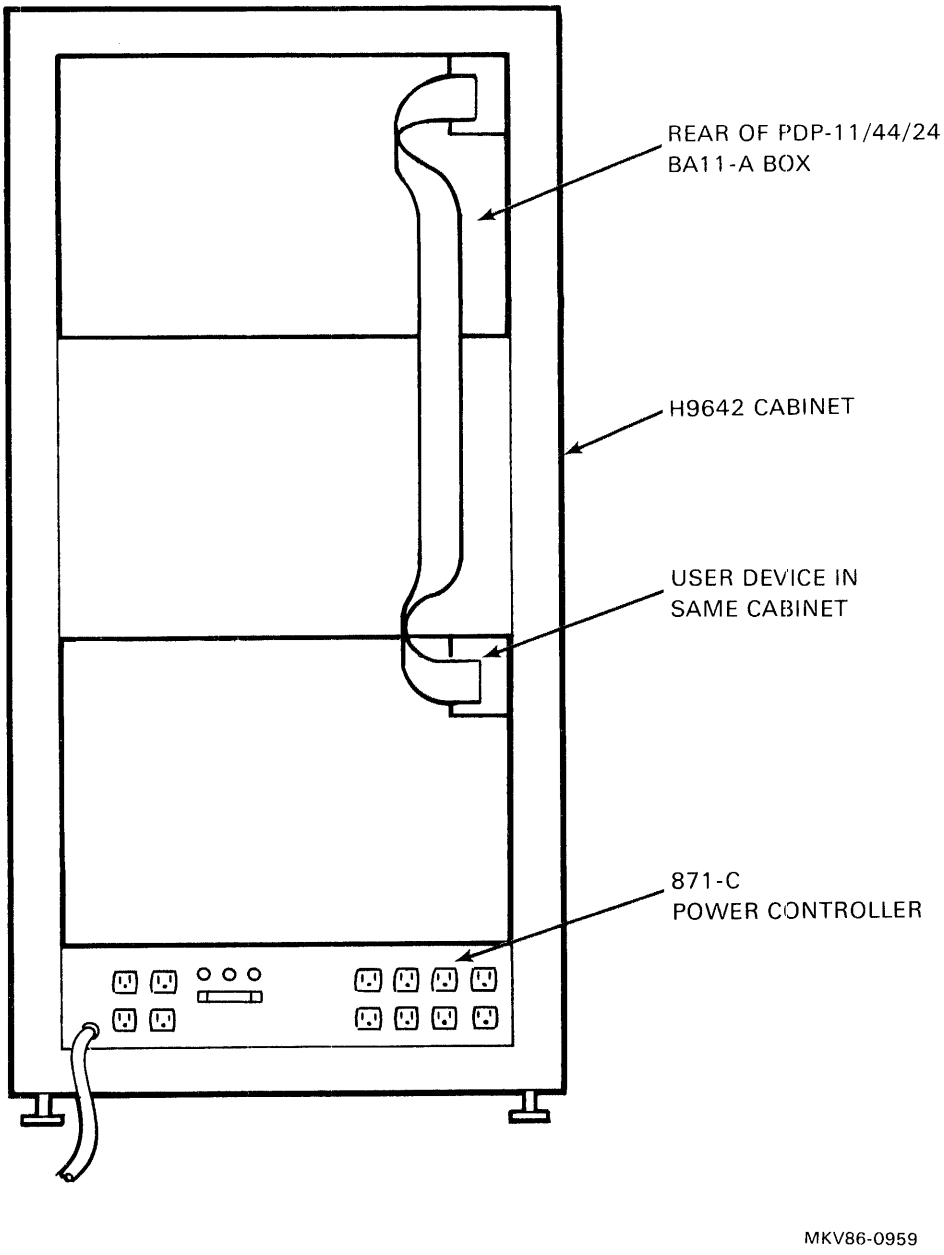

Figure 2-4 Installing the DR11-W Without the Bulkhead Panel (H9642 Cabinet, 871-C Power Controller, and User Device)

2.6 CABLE WRAP-AROUND TESTING

This testing essentially verifies all of the logic in the DR11-W. It should be performed upon initial installation and anytime the results of the logic wrap-around test are inconclusive.

If the cable wrap-around test is not to be performed, go to Section 2.9.

During this test, the user cables are disconnected from the DR11-W and a single cable is connected from the output connector on the DR11-W back to the input connector (see Figure 2-5). This covers the remainder of the DR11-W logic not tested in the logic wrap-around mode (primarily, the user-input receivers). This test also provides a convenient method of testing the user cables. Just connect them, one at a time, as the wrap-around cable.

If you are using the recommended BC06-R flat gray cables, be sure that the color stripe is oriented towards the module handle at both connectors. If you are using a short, unshielded cable (like the BC05-L), the cable orientation is unimportant so long as there are no twists in the cable.

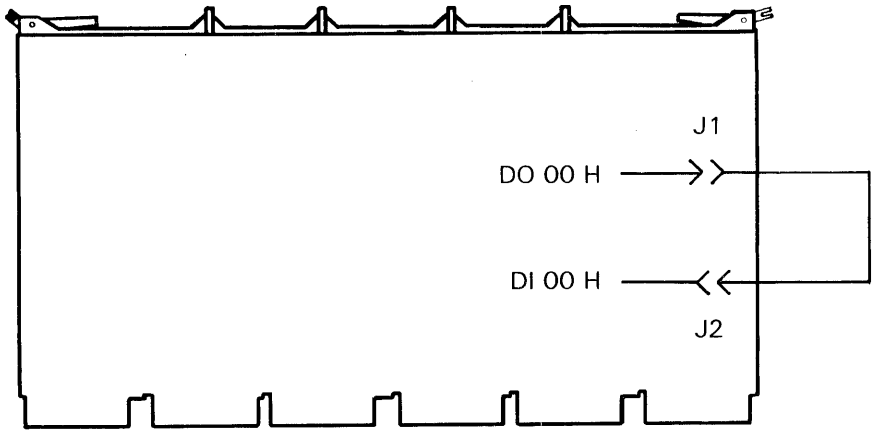

Figure 2-5 Cable Wrap-Around Testing Configuration

The DR11-W performs the cable wrap-around test by connecting the data from the output connector to the input connector. Figure 2-5 shows how one bit (D00 H) is connected. All data bits (D00 H through D15 H) and control signals are similarly connected. This connection method provides maximum diagnostic coverage, including input receivers untested by the logic wrap-around test.

To perform the test:

- Unplug the external cables from the two user connectors on the DR11-W.

NOTE

If the DR11-W is installed in a system using the FCC bulkhead, and you desire to test the cables between the DR11-W and the bulkhead, then unplug the user cables at the outside of the bulkhead rather than at the DR11-W module itself.

- Plug the maintenance wrap-around cable into both user connectors on the DR11-W or the FCC bulkhead. If the cable has a colored stripe, ensure that the stripe is facing the module handle. For the FCC bulkhead, ensure that the colored stripe is facing upwards or rightwards.

- Record the current settings of E105 switches 1, 2, 3, 4, and 5.

- Set the E105 switches:
  - 1 to OFF
  - 2 to ON
  - 3 to OFF
  - 4 to OFF
  - 5 to ON
- Carefully plug the DR11-W into the desired slot.
- Apply power to the system and run the appropriate stand-alone diagnostic program:
  - ESDRE for the VAX-11 family
  - CZDRL for the PDP-11 family
- If desired, perform the burst release time-out adjustment as directed by the diagnostic.
- Remove power from the system.
- Reset the E105 switchpack to the original settings.

2.7 LOGIC WRAP-AROUND TESTING

This testing verifies most of the logic in the DR11-W, excluding only a few user-interface drivers and receivers. It should be performed upon initial installation and anytime the operation of the DR11-W is in doubt.

With no cables attached to the DR11-W, the diagnostic internally disconnects the user inputs, connects the user outputs back to the user inputs, and runs the appropriate tests (see Figure 2-6).

The DR11-W performs the logical loopback by sampling the data appearing at the ouput connector (J1) of the DR11-W. Figure 2-6 shows how one bit (D00 H) is connected. All data bits (D00 H through D15 H) and the busy signal are similarly connected. This connection scheme was chosen to get maximum diagnostic coverage, up to and including the output drivers.

However, this connection scheme also means that if the user cables are left connected to the DR11-W, they may cause the diagnostic to fail if: the user cable is shorted, the user cable is excessively capacitive, or the user device is forcing the outputs to one state or the other.

CAUTION

Since the output signals are still driven while running the logic wrap-around test, the DR11-W produces unpredictable signals at the user connectors. You must take care that these signals do not cause damage to the user device (by commanding illegal or destructive operations, etc). It is recommended that the user device be disconnected during any DR11-W testing.

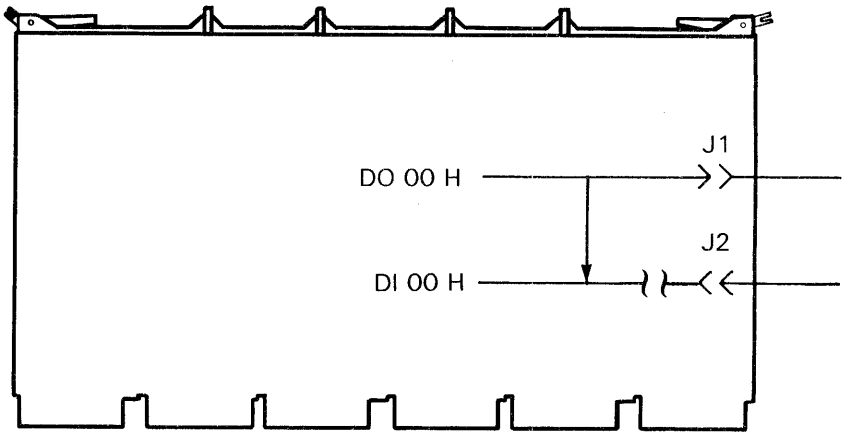

Figure 2-6 Logic Wrap-Around Testing Configuration

To perform the test:

- Disable, power-down, or disconnect the user device. See the caution above.
- Apply power to the system and run the appropriate diagnostic program:
  - ESDRE for the VAX-11 family running stand-alone
  - ESDRB for the VAX-11 family running on-line
  - CZDRL for the PDP-11 family

2.8 LINK-MODE TESTING

Additional diagnostics must be run on DR11-W links. If you are not using your DR11-W as part of a link, go to section 2.9.

When testing a DR11-W to DR11-W (or DR11-W to DRV11-B) link, each DR11 should first be individually tested. When each has individually passed, the two DR11s may be tested together as a link. Both ends of the link must be running cooperating software (such as ESDRE or CZDRK). Using DEC diagnostics, the link is cabled normally. One end is started as the slave, then the other end is started as the master (these definitions basically describe who speaks first; the definitions are exchanged at the end of each pass of the diagnostic). The full functionality of the link is tested. Figure 2-7 illustrates how one data bit is connected in each direction, forming a full link.

NOTE

If any of the tests report the dropping of data bit ⟨15⟩, suspect that either or both BC06-R cables are installed with the colored-stripe away from the module handle. This attempts to drive the cable's ground plane as bit ⟨15⟩. The ground plane is capacitive, and as result, bit ⟨15⟩ is slow. This causes it to be lost during high-speed transfers.

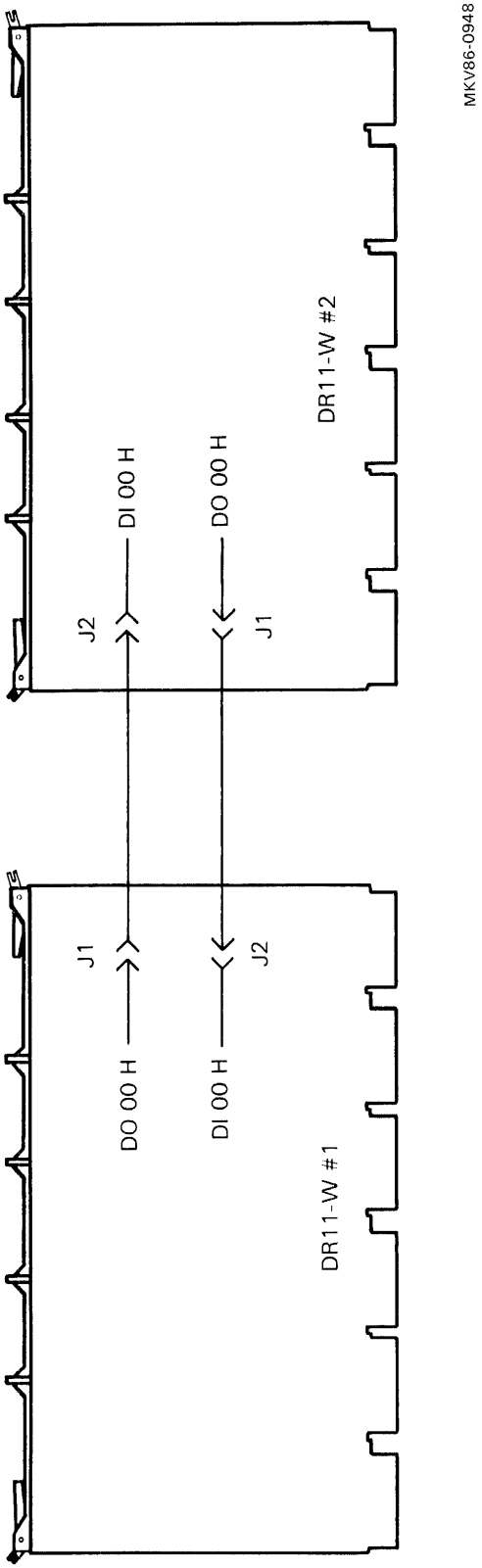

Figure 2-7 Link Testing Configuration

2.9 FINAL INSTALLATION

Once the DR11-W has been diagnosed as error free, you may perform the final installation of the module.

- Ensure that the E105 settings are correct.
- Connect the user or link cables. If the cables have a colored stripe, be sure that the stripe faces the module handle.
- Install the DR11-W in the desired slot.
- Apply power to the system and test the application, using your software.

CHAPTER 3
USER DEVICE INTERFACING

3.1 INTRODUCTION
This chapter describes the hardware interface between the DR11-W and your device. The case of a DR11-W linked to another DR11-type device is covered separately in Chapter 5.

3.2 PHYSICAL
All connections to your device are made through the two 40-pin connectors mounted on the DR11-W module.

If your system is FCC compliant, then the two connectors of the DR11-W are exactly duplicated at the bulkhead mounted at the rear of the cabinet.

3.3 ELECTRICAL
All user signals are received using standard UNIBUS receivers and are transmitted using standard UNIBUS open-collector drivers. All signals are terminated at the DR11-W module into 120 ohms and +3.3 Vdc (that is, the Thevenin resistance and voltage is 120 ohms and +3.3 Vdc). In order to minimize reflections, noise, and crosstalk, the signals should be connected to the user device using 120 ohm cable. The user device should use the same receivers, drivers, and terminators as are used on the DR11-W.

3.3.1 Output Circuit
Figure 3-1 illustrates a typical DR11-W output circuit. The driver used is an 8881 open-collector UNIBUS driver. Early DR11-Ws terminated each line with a pair of discrete resistors (as shown); more recent DR11-Ws use hybrid terminators. The user device should receive DR11-W output signals with a UNIBUS receiver: either an 8640 quad receiver chip or an 8641 quad transceiver (with the transmitter section disabled).

3.3.2 Input Circuit
Figure 3-2 illustrates a typical DR11-W input circuit. The receiver used is an 8640 low-leakage UNIBUS receiver. As in the output circuits, early DR11-Ws terminated each line with a pair of discrete resistors (as shown); more recent DR11-Ws use hybrid terminators. The user device should drive DR11-W input signals with a UNIBUS driver: either an 8881 quad driver chip or an 8641 quad transceiver.

3.3.3 Terminators
The terminators on the DR11-W module ensure that any input left completely unconnected is seen as a clean logic-high. However, this cannot be guaranteed if the line is run through the user cables and left unconnected and unterminated at the user device. An undriven, unterminated line is susceptible to the pickup of noise and crosstalk (see Figure 3-3).

3.3.4 Cable Characteristics
In order to minimize reflections, crosstalk, and noise pickup, the DR11-W user interface signals should be transmitted using cables with a characteristic impedance of 120 ohms to ground.

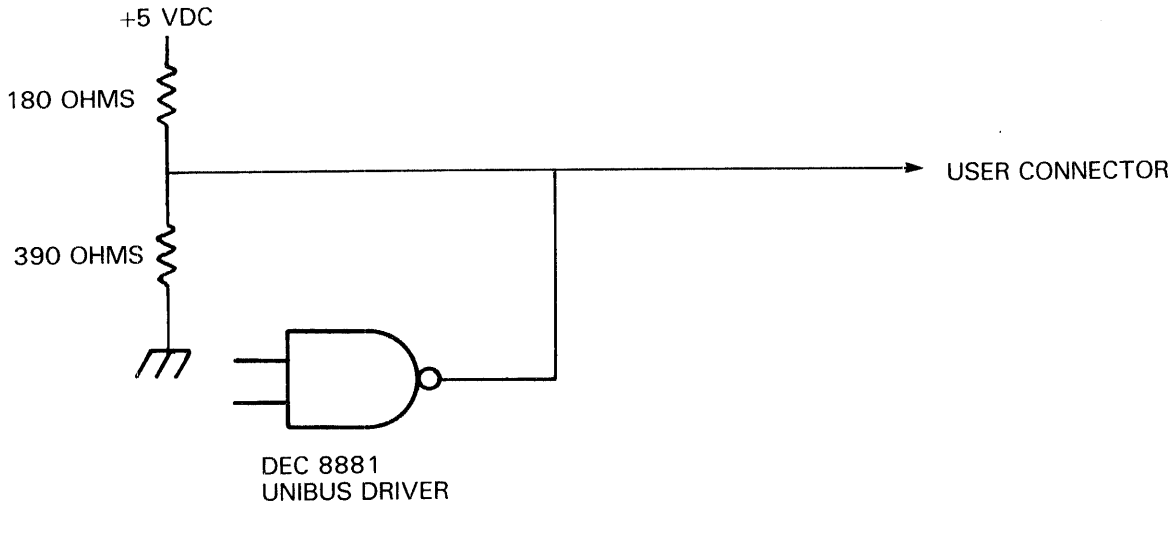

Figure 3-1 Typical DR11-W Output Circuit Diagram

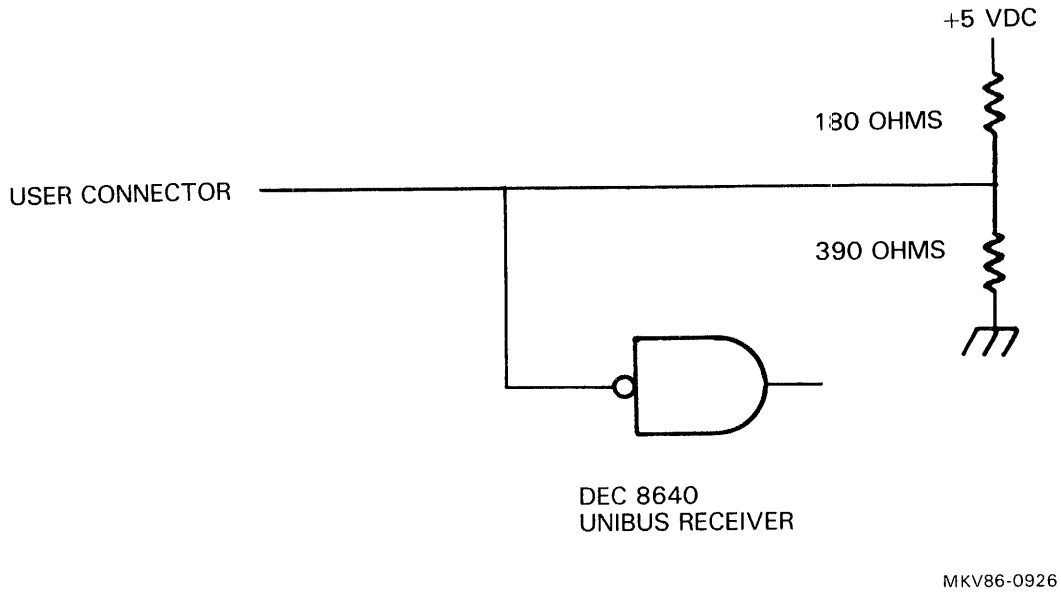

Figure 3-2 Typical DR11-W Input Circuit Diagram

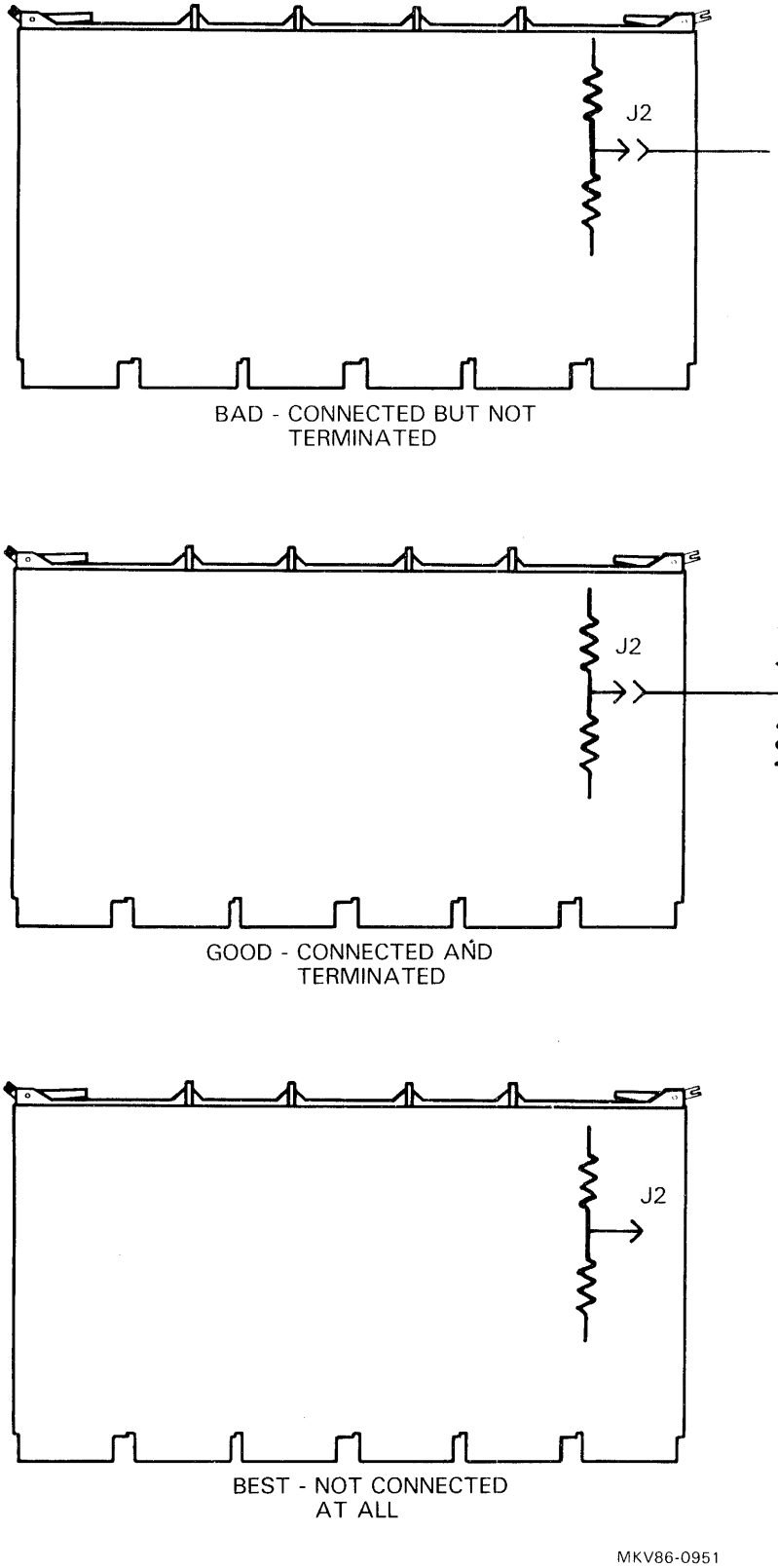

Figure 3-3 Unused Input Configurations

BC06-R flat grey MASSBUS cables meet this requirement and are available from Digital Equipment Corporation in a wide variety of lengths. You can also assemble your own custom cables using equivalent raw cable stock. Table 3-1 lists a few vendors and their part numbers for equivalent raw stock. All cables listed in Table 3-1 are AWM UL Style 2682.

NOTE

All of the cables listed in Table 3-1 contain an integral ground plane. This ground plane must connect to pin A on each of the DR11-W user connectors.

If cables other than those listed above are used, their characteristic impedance must be specified or measured to ensure best performance.

3.3.5 Logic Levels

Correct dc logic levels are essential for the correct operation of the DR11-W. Table 3-2 documents the logic level limits for both the drivers and receivers.

Table 3-1 Cable Stock Part Numbers

Vendor	Cable Stock Part Number
DIGITAL	17-00034-00
Tensolite	81-25-00-4000
Spectrastrip	N/A
3M	N/A
Brand Rex	N/A

Table 3-2 Logic Voltage Level Requirements

Parameter	Value
For drivers:	
Volts maximum	0.7 Vdc @ 70 mA[1]
Ioh maximum	25 μA @ 5 Vdc
For receivers:	
Vil range	0.0 to 1.3 Vdc
Vih range	1.7 to 5.0 Vdc
Iih max	80 μA

[1]Newer drivers also achieve 0.9 Vdc @ 100 mA

3.3.6 Logic Polarity

All inputs and outputs from the DR11-W are logic-high (that is, asserted when near +3 Vdc) except BUSY and BURST RQ L. The polarity of the busy signal is switch-selectable.

Since BURST RQ L is a logic-low signal, if no connection is made to this pin, then the DR11-W operates in non-burst mode. If the signal is hard grounded, then the DR11-W operates in two-cycle or n-cycle burst mode. Of course, like any other input signal, the user device can dynamically drive this pin.

3.3.7 Logic Reference

The DR11-W uses single-ended drivers and receivers. It is imperative for both the safety and proper operation of the equipment that the user device be at the same logic reference level (ground) as the DR11-W. More than a few millivolts of ground potential can cause improper operation. More than a few hundred millivolts can damage the equipment.

One way to ensure that the DR11-W and the user device share the same logic reference is to mount the user device and the DR11-W in the same cabinet, referenced to the same ground. It is recommended that the user device always be mounted in the same cabinet (or the same series of cabinets bolted together) as the DR11-W.

If the user device is mounted in a cabinet separate from the DR11-W, a solid logic reference must be provided. This may be accomplished by connecting the cabinet frames together with copper cable of number two gauge or larger. This is similar to the technique used with the MASSBUS (a differential bus).

Digital Equipment Corporation Computer Special Systems sells an adapter module (DR11-WC/WD) which converts the DR11-W's single-ended signals to differential signals. This adapter module can be used in difficult cases of ground noise and ground potential.

3.3.8 Transmission Distance

The UNIBUS drivers and receivers used in the DR11-W are specified for a maximum transmission distance of 15.25 m (50.0 ft). This distance includes all cabling, including the cable used between the DR11-W module and the FCC bulkhead. This limitation is imposed both by the maximum allowable dc voltage drop in the signal cables and by the maximum allowable signal skews.

As the signals propagate through the conductors of the cable, the resistance of the cable causes a loss of voltage (known as an IR drop). This voltage loss subtracts from the available noise margin. When an insufficient noise margin remains, transmission errors can occur.

Also, all signals in the two user cables do not propagate at exactly the same speed. The difference in arrival time between any two signals is referred to as skew. The DR11-W DMA engine accounts for a limited amount of skew by providing deskew time at the beginning of each transfer. If the skew in the data signals exceeds this deskew time, errors will occur. Increasing the deskew time reduces the maximum transfer rate of the DR11-W.

Neither of these limits is a hard and fast rule – bigger conductors in the cable can reduce IR losses while additional logic in the user device can accommodate additional skew. However, this is clearly a case where you must understand the rules before violating them.

The Computer Special Systems differential driver module (DR11-WC/WD) can be useful in extending the transmission distance of the DR11-W. The differential transmission technique used combined with increased deskew times allows distances of at least 300 m (1000 ft).

3.4 DATA FORMAT

The UNIBUS is a bus with 16 data lines and 18 address lines. The address space (as viewed from the UNIBUS) may be viewed as a linear series of 262144 (256K) bytes, each with its own unique address (see Figure 3-4). Addresses on the UNIBUS are always the address of a particular byte.

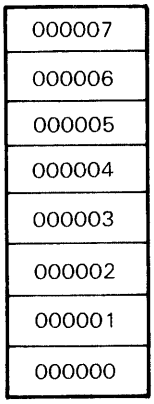

Figure 3-4 The UNIBUS Address Space Map for a Series of 8-Bit Bytes

The address space may also be viewed as a series of 131072 (128K) words, each composed of two bytes (see Figure 3-5). The address of the word can be specified by the address of either byte in the word, but the convention chosen for the UNIBUS is to use the address of the less significant byte. This means that for all word accesses on the UNIBUS, address bit ⟨00⟩ is equal to zero (A00=0). This use of address bit ⟨00⟩ leads to the less significant byte being referred to as the even byte (A00=0) while the more significant byte is referred to as the odd byte (A00=1).

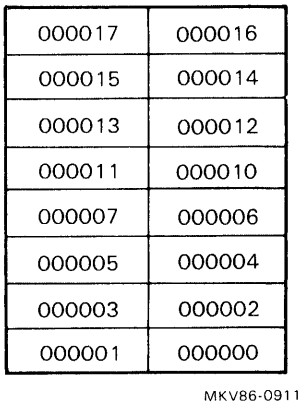

Figure 3-5 UNIBUS Address Space Map for a Series of 16-bit Words

Address bits on the UNIBUS are numbered from right to left in order of increasing significance (see Figure 3-6). The address bits are named A00 through A17. Remember, the address presented on the UNIBUS is always the address of a byte of memory.

In the DR11-W, address bits ⟨17⟩ through ⟨01⟩ are supplied by the software program via the control and status register (CSR) and the bus address register (BAR), while address bit ⟨00⟩ is provided by the user device via the A00 H input line.

Data bits on the UNIBUS are numbered from right to left in order of increasing significance. The data bits are named D00 through D15 (see Figure 3-7).

A UNIBUS data word can also be construed as representing two bytes (see Figure 3-8). When interpreting the data in this fashion, the bytes appear exactly as pictured in Figure 3-5. Data from bytes at even addresses (that is, with address bit ⟨00⟩=0) always appears on data lines 07-00. Data from bytes at odd addresses (that is, with address bit ⟨00⟩=1) always appears on data lines 15-08. Odd byte data never appears on D07-D00.

This representation of words and bytes allows for the easy processing of both, without much specialized hardware. Further discussion may be found in Section 3.12.

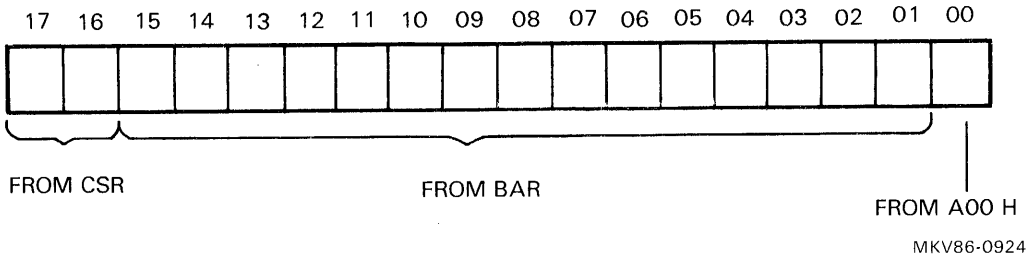

Figure 3-6 Eighteen-Bit UNIBUS Address Configuration

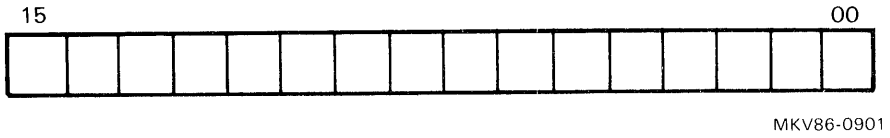

Figure 3-7 UNIBUS Data Interpreted as a Word

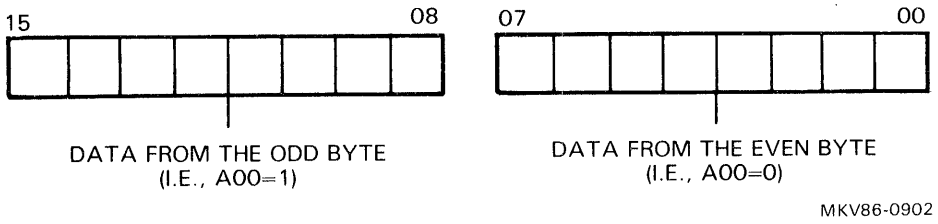

Figure 3-8 UNIBUS Data Interpreted as Two Bytes

3.5 DIRECTION OF DATAFLOW DURING PROGRAMMED I/O

During programmed I/O operations (or, by extension, interrupt-driven I/O), the processor is entirely in control of the data transfer (that is, the processor is the UNIBUS master) and the direction of dataflow is intuitive (see Figures 3-9 and 3-10). When the processor writes to the output data buffer, data appears on the data out wires and is presented to the user device. When the processor reads from the input data buffer, data from the user device is sampled from the data in wires.

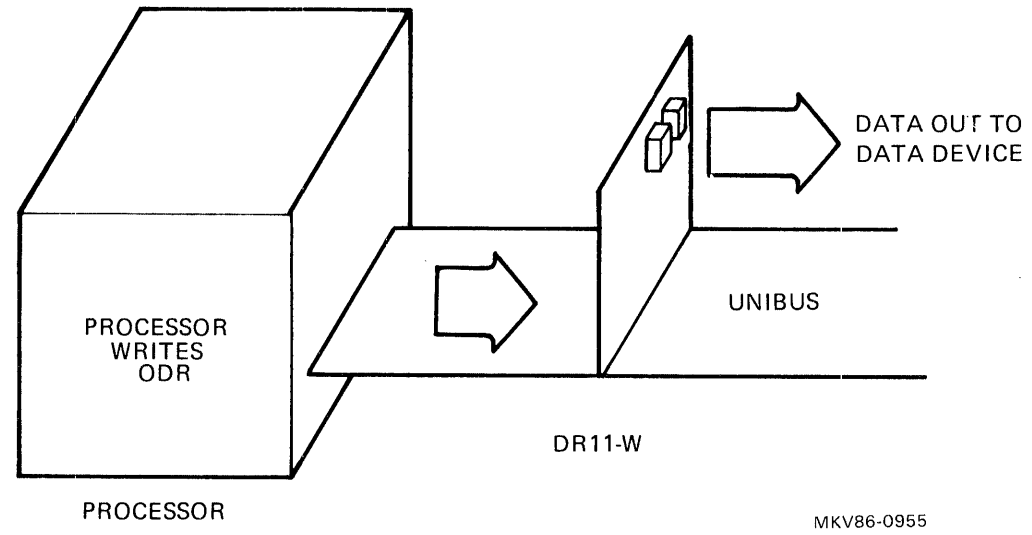

Figure 3-9 Processor Writing Output Data Register Diagram

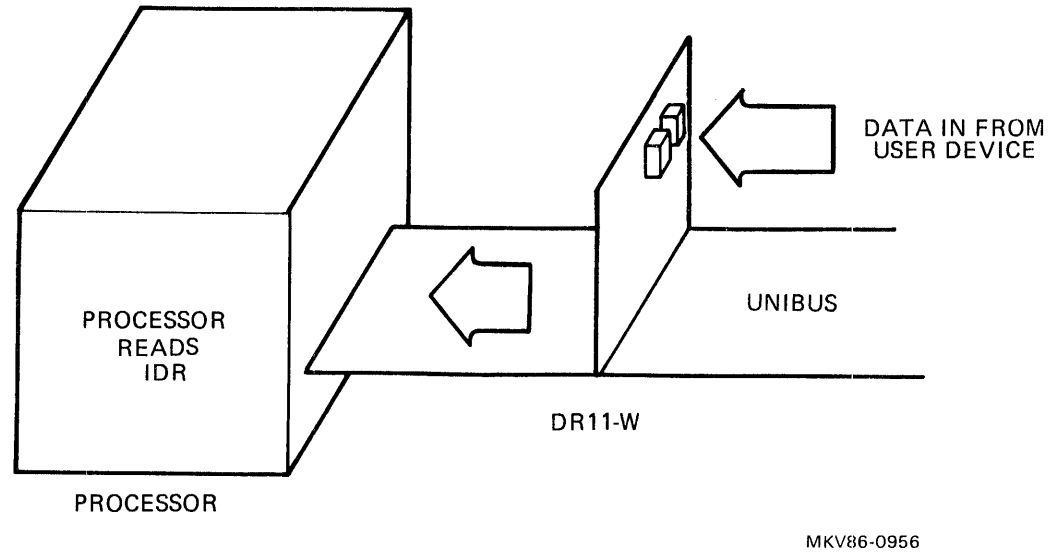

Figure 3-10 Processor Reading Input Data Register Diagram

3.6 DIRECTION OF DATAFLOW DURING DIRECT MEMORY ACCESS (DMA) I/O

During DMA I/O operations, the processor is not involved. Now, the DR11-W is the UNIBUS master and directly controls main memory (see Figures 3-11 and 3-12). The DR11-W has two ports – the UNIBUS port and the user port. While data is flowing in one port, it must be flowing out the other. This can lead to confusion in the naming of the dataflow direction. The direction of dataflow is always stated with regards to the DR11-W's UNIBUS port.

When data is flowing from memory to the user device, the data flows from memory in to the DR11-W's UNIBUS port, and then from the DR11-W's user port out to the user device. The operation that the DR11-W performs as the UNIBUS master is referred to as a data-in or DATI. This is the reverse of what intuition (and the programmed I/O case) would suggest.

When data is flowing from the user device to memory, the data flows from the user device in to the DR11-W's user port, and then from the DR11-W's UNIBUS port out to memory. The operation that the DR11-W performs as the UNIBUS master is referred to as a data-out or DATO.

Table 3-3 summarizes this description. Note that the table only makes reference to DATI and DATO. The DR11-W actually implements two additional operations: DATIP and DATOB. DATIP (data-in, pause) is a variant of DATI while DATOB (data-out, byte) is a variant of DATO.

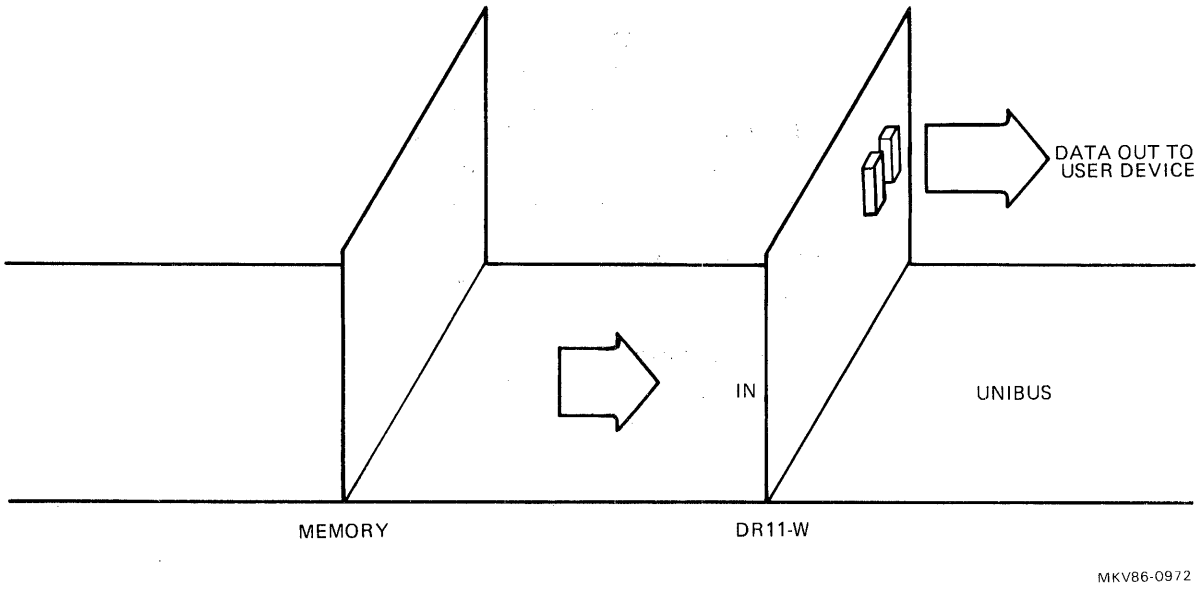

Figure 3-11 Data from Memory to User (DATI) Diagram

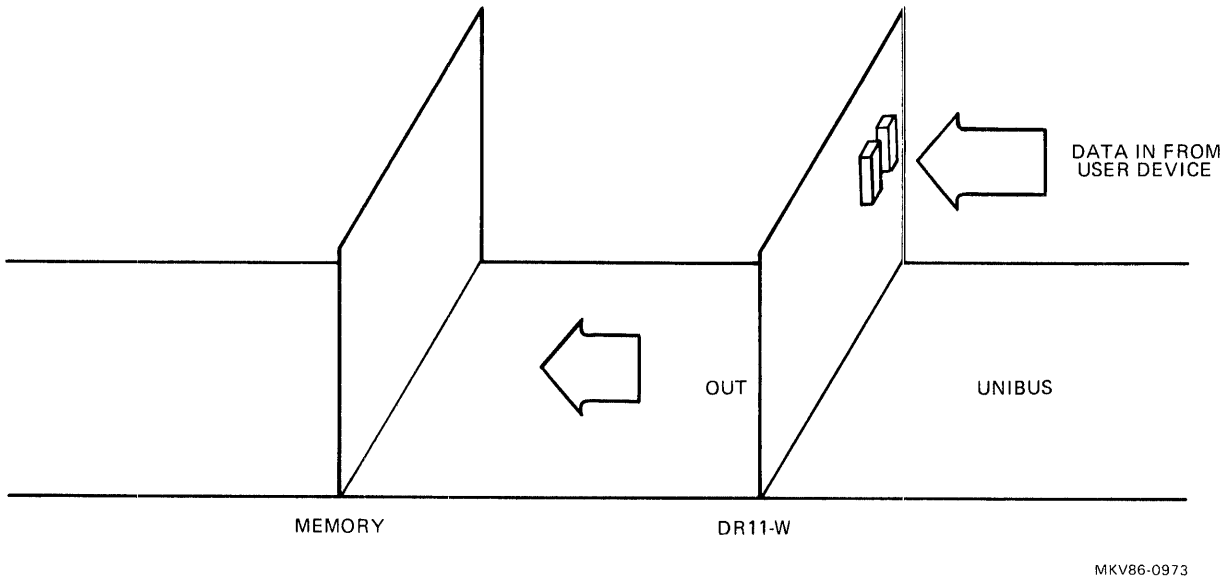

Figure 3-12 Data from User to Memory (DATO) Diagram

Table 3-3 DMA Data flow Directions

Direction of Data flow	Operation Name
From memory to user device	DATI
From user device to memory	DATO

3.7 INPUT SIGNAL FUNCTIONAL DESCRIPTION

The DR11-W takes 28 signals as inputs from the user device. They are listed in Table 3-4 and subsequently described in detail.

Table 3-4 DR11-W User Input Signals

Signal Name	Pin	Function
Data inputs:		
DI 00 H	J2-UU	Input data bit ⟨00⟩
DI 01 H	J2-SS	Input data bit ⟨01⟩
DI 02 H	J2-PP	Input data bit ⟨02⟩
DI 03 H	J2-MM	Input data bit ⟨03⟩
DI 04 H	J2-KK	Input data bit ⟨04⟩
DI 05 H	J2-HH	Input data bit ⟨05⟩
DI 06 H	J2-EE	Input data bit ⟨06⟩
DI 07 H	J2-CC	Input data bit ⟨07⟩
DI 08 H	J2-DD	Input data bit ⟨08⟩
DI 09 H	J2-FF	Input data bit ⟨09⟩
DI 10 H	J2-JJ	Input data bit ⟨10⟩
DI 11 H	J2-LL	Input data bit ⟨11⟩
DI 12 H	J2-NN	Input data bit ⟨12⟩
DI 13 H	J2-RR	Input data bit ⟨13⟩
DI 14 H	J2-TT	Input data bit ⟨14⟩
DI 15 H	J2-VV	Input data bit ⟨15⟩
Status Inputs:		
STATUS A H	J1-L	User device status bit ⟨1⟩
STATUS B H	J1-R	User device status bit ⟨2⟩
STATUS C H	J1-T, J1-V	User device status bit ⟨3⟩
Bus Control:		
CYCLE RQ A H	J1-B	DMA cycle start signal A
CYCLE RQ B H	J1-Z	DMA cycle start signal B
BURST RQ L	J1-K	Use burst mode
C0 CNTL H.	J2-N	UNIBUS cycle selection
C1 CNTL H	J2-T	UNIBUS cycle selection
A00 H	J2-F	Low-order UNIBUS address bit
BA INC ENB H	J2-J	Allow bus address register to increment
WC INC ENB H	J1-J	Allow word count register to increment
Interrupt:		
ATTN H	J2-D	Interrupt the processor

Most of the input signals are connected to the DR11-W via J2. The rest are connected via J1. Figure 1-3 shows the physical location of the two connectors and Figure 3-13 the individual pins and their designations for each connector.

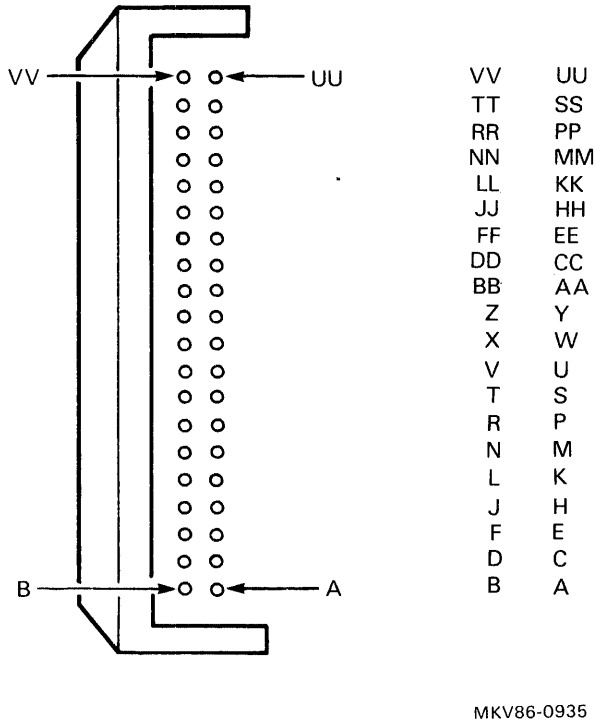

Figure 3-13 J1 and J2 Connector Pin Identification

3.7.1 Data Inputs

These 16 lines supply data to the DR11-W. They are logic-high lines. They are sampled and latched into the DR11-W each time the input data register is read, and approximately 200 nanoseconds after the rising edge of either CYCLE RQ A H or CYCLE RQ B H. Data must be stable at those times in order to be correctly read. Refer to Section 3.4.

3.7.2 Status Inputs

These three lines are used as general purpose inputs to the DR11-W, and can be read by the program at any time. They are sampled and latched into the DR11-W each time the control and status register is read, and approximately 200 nanoseconds after the rising edge of either CYCLE RQ A H or CYCLE RQ B H. Data must be stable at those times in order to be correctly read. Whatever presented at these inputs is immediately visible in the control and status register, regardless of the state of the DR11-W.

A common use of these lines is for handshaking between the user device and the software. They can also be used to provide the software with the status of the user device.

3.7.3 Bus Control

These signals are all used to initiate and control the DMA cycles performed by the DR11-W. All of these signals (except CYCLE RQ A H and CYCLE RQ B H) are latched into the DR11-W approximately 200 nanoseconds after the rising edge of CYCLE RQ A H or CYCLE RQ B H.

If the software has set up a DMA transfer, then the rising edge of either CYCLE RQ A H or CYCLE RQ B H sets the cycle flip-flop, starting a DMA cycle (assuming that the other input is deasserted and that the cycle flip-flop is not already set). The busy flip-flop is also set at this time. Since both CYCLE RQ A H and CYCLE RQ B H perform identical functions, throughout this manual they are frequently referred to collectively as CYCLE RQ x H (that is, either CYCLE RQ A H **or** CYCLE RQ B H).

C0 CNTL H and C1 CNTL H select the type of UNIBUS cycle to be performed (see Table 3-5). These signals must be driven by the user device or hard-wired to the desired logic state.

Table 3-5 UNIBUS Cycle Selection

C1 H	C0 H	Cycle Name	Description of Cycle
0	0	DATI	Read word or byte from memory
0	1	DATIP	Read with write intent
1	0	DATO	Write word to memory
1	1	DATOB	Write byte to memory

BURST RQ L is used to determine whether this cycle should be performed in burst or non-burst mode.

A00 H is used during DATOB (write byte) cycles as the least significant bit of the UNIBUS address. It determines which of the two bytes of the UNIBUS data word should be written. For read operations, word writes, and writes to even byte addresses, A00 H must remain unasserted. Only during writes to odd byte addresses should A00 be asserted by the user device.

BA INC ENB H controls whether or not the bus address register is incremented at the completion of the present DMA cycle. This signal is normally asserted (high), allowing the bus address register to increment with each transfer. If the cycle is a DATIP (read with write intent) or a DATOB (write byte) to the low byte of the word, then the bus address register should remain unchanged until after the next cycle. In this case, the user device should deassert BA INC ENB H by pulling it to ground.

WC INC ENB H performs the same function for the word count register. In the case of DATOB, however, it may be more convenient to treat the word count register as a byte count register. In this case, the register can be allowed to increment with every transfer.

More details on the use of A00 H, BA INC ENB H, and WC INC ENB H can be found in Section 3.12.

3.7.4 Interrupt

The rising edge of ATTN H causes the attention interrupt flip-flop in the DR11-W to be set. No further DMA cycles are started and the DR11-W becomes ready and interrupts the processor if the interrupt enable flip-flop is set.

This signal also sets the attention flip-flop. This flip-flop holds the attention indication for display in the control and status register.

3.8 OUTPUT SIGNAL FUNCTIONAL DESCRIPTION

The DR11-W provides 25 signals as outputs to the user device. They are listed in Table 3-6 and subsequently described in detail.

Table 3-6 DR11-W User Output Signals

Signal Name	Pin	Function
Data Outputs:		
DO 00 H	J1-UU	Output data bit ⟨00⟩
DO 01 H	J1-SS	Output data bit ⟨01⟩
DO 02 H	J1-PP	Output data bit ⟨02⟩
DO 03 H	J1-MM	Output data bit ⟨03⟩
DO 04 H	J1-KK	Output data bit ⟨04⟩
DO 05 H	J1-HH	Output data bit ⟨05⟩
DO 06 H	J1-EE	Output data bit ⟨06⟩
DO 07 H	J1-CC	Output data bit ⟨07⟩
DO 08 H	J1-DD	Output data bit ⟨08⟩
DO 09 H	J1-FF	Output data bit ⟨09⟩
DO 10 H	J1-JJ	Output data bit ⟨10⟩
DO 11 H	J1-LL	Output data bit ⟨11⟩
DO 12 H	J1-NN	Output data bit ⟨12⟩
DO 13 H	J1-RR	Output data bit ⟨13⟩
DO 14 H	J1-TT	Output data bit ⟨14⟩
DO 15 H	J1-VV	Output data bit ⟨15⟩
Function Outputs:		
FNCT 1 H	J2-V	User device command bit ⟨1⟩
FNCT 2 H	J2-R	User device command bit ⟨2⟩
FNCT 3 H	J2-K, J2-L	User device command bit ⟨3⟩
Bus Control:		
END CYCLE H	J1-X	DMA cycle about to end
BUSY	J2-B	DMA cycle in progress
Miscellaneous:		
UACLO FNCT 2 H	J1-D	FNCT 2 or'd with UNIBUS AC LO
INIT H	J1-N	The DR11-W is being initialized
READY H	J1-F	The DR11-W is ready
Go H	J2-X	The DR11-W go bit was just set

Most of the output signals are connected from the DR11-W via J1. The rest are connected via J2. Figure 1-3 shows the physical location of the two connectors while Figure 3-13 shows the individual pins on each connector.

3.8.1 Data Outputs

These 16 lines supply data to the user device. They are logic-high lines. They are driven from latches which are loaded each time the output data register is written or whenever a DMA read is finished. Refer to Section 3.4.

3.8.2 Function Outputs

These three lines are used as general purpose outputs from the DR11-W and can be set by the program at any time. Whatever is written into the control and status register function bits is immediately visible at the function outputs.

A common use of these lines is for handshaking between the user device and the software. They can also be used to provide functional commands to the user device.

3.8.3 Bus Control

END CYCLE H is a pulse which occurs just prior to the clearing of the DR11-W busy signal. Fast user devices can use this signal as a warning that the next word of data must shortly be presented to the DR11-W, or taken from the DR11-W. Slower user devices need not use the END CYCLE H signal.

BUSY is a level which indicates that a single DMA cycle is ongoing. The polarity of the busy signal is selected via the E105 switchpack. BUSY asserts when the cycle flip-flop is set; BUSY deasserts when the DMA cycle is completed. There is one complete cycle of the busy signal for each DMA transfer, regardless of the burst mode used. That is, the transfer of each word or byte is accompanied by a complete cycle of the busy signal.

3.8.4 Miscellaneous

UACLO FNCT 2 H is the logical OR of two signals: FNCT 2 H and UNIBUS ACLO. UNIBUS ACLO is the UNIBUS signal which indicates an impending power failure. Therefore, UACLO FNCT 2 H is asserted anytime that FNCT 2 is asserted or there is an impending power failure on the UNIBUS.

INIT H is asserted anytime the DR11-W is initialized. This can occur whenever UNIBUS INIT is asserted, or whenever the DR11-W maintenance flip-flop is cleared. This output may be used to signal that the user device should also be initialized, or to indicate to intelligent user devices that the logical connection with the processor should be re-established.

READY H is derived from the DR11-W ready flip-flop. This signal is asserted whenever the DMA engine of the DR11-W is idle (that is, asserting CYCLE RQ A H or CYCLE RQ B H does not start a transfer).

GO H is a 160 nanosecond pulse indicating that the DR11-W's DMA engine has just been armed. Simultaneously, READY H deasserts. Subsequent assertions of CYCLE RQ x H start a transfer.

3.9 DMA CYCLES IN GENERAL

The software must set up each block of DR11-W DMA cycles. This procedure is described in detail in Chapter 4. Once the go bit has been set, the DR11-W is ready to transfer data.

The fact that the go bit has been set is indicated to the user device in two ways. The ready line deasserts (ready is actually an indication to the software that the DR11-W is ready to accept the next software command). The go line also asserts for 160 nanoseconds.

Now, the user device controls the operation of the DR11-W. Each time the user device asserts either CYCLE RQ x H line, the DR11-W performs one DMA cycle. The type of DMA cycle (DATI, DATIP, DATO, or DATOB) is selected by the user device. The CPU does not control the direction of the data transfer.

As each DMA cycle begins, the DR11-W asserts the busy signal. This indicates that the DMA cycle is ongoing. Near the completion of the DMA cycle, END CYCLE H pulses. Shortly after that, BUSY deasserts, indicating that the DR11-W has completed the DMA cycle.

If the DR11-W is not yet ready (for the next software command), then another DMA cycle can be started using either CYCLE RQ x H input.

Each DMA cycle (except for a DATIP cycle) is independent of any other cycle. That is to say, the user device can freely intermix read cycles, write word cycles, and write byte cycles (although mixing cycle types within a block is not normally done).

DATIP (read with write intent) leaves the selected memory bank locked up. Only a DATO or DATOB (write word or write byte) to the same address can unlock the memory bank. Therefore, each DATIP must be followed by a DATO or DATOB to the same address as the DATIP.

3.10 A SINGLE CYCLE IN DETAIL

Assume that the DR11-W has been started by the software: the READY H line is deasserted and the GO H pulse has been seen.

Refer to Figure 3-14. The user device must set up the following lines:

- C0 CNTL H
- C1 CNTL H

In addition, if the transfer is writing a word **to** memory, the data inputs must also be set up:

- DI 15 H to DI 00 H (assuming this transfer writes memory)

Finally, if any of the following lines are in use, they must be set up as well:

- A00 CNTL H
- WC INC ENB H
- BA INC ENB H
- BURST RQ L

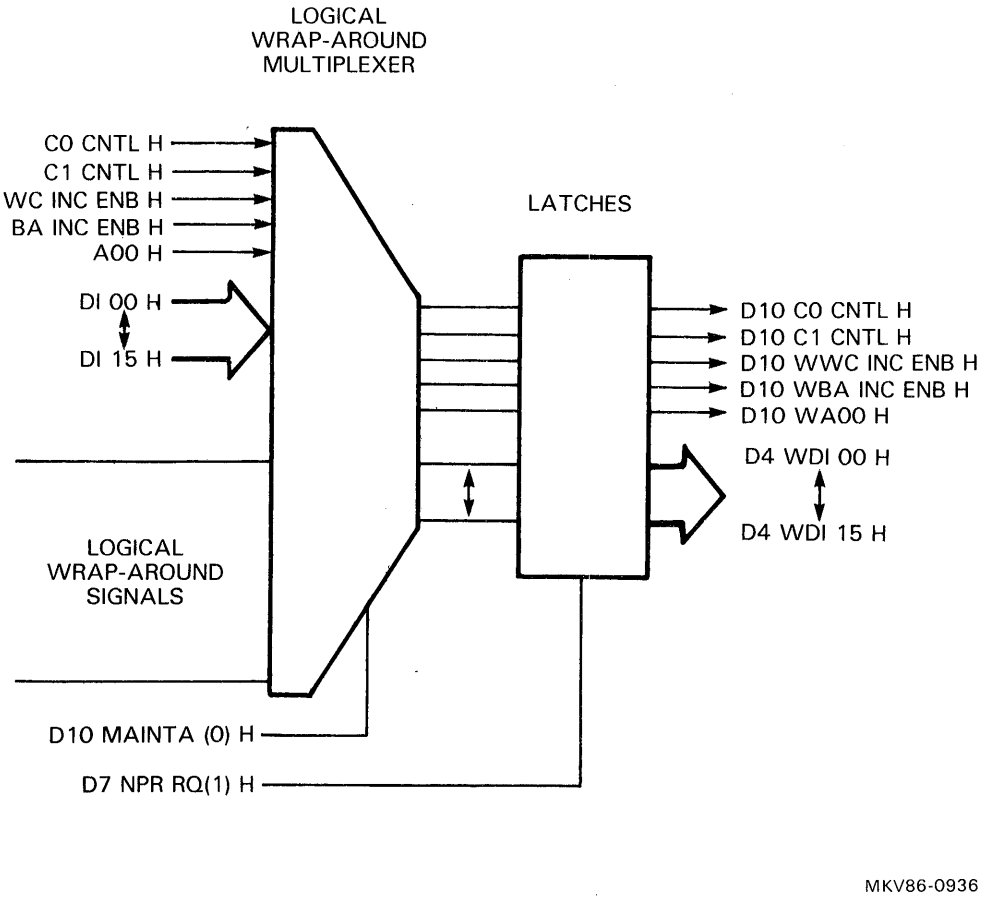

Figure 3-14 DR11-W Input MUX and Latch Block Diagram

Once these lines have been correctly set up (asserted or not asserted), then the user device should assert one of the two CYCLE RQ x H lines (see Figure 3-15). The other CYCLE RQ x H line must be held unasserted. Both lines perform identically and either may be used.

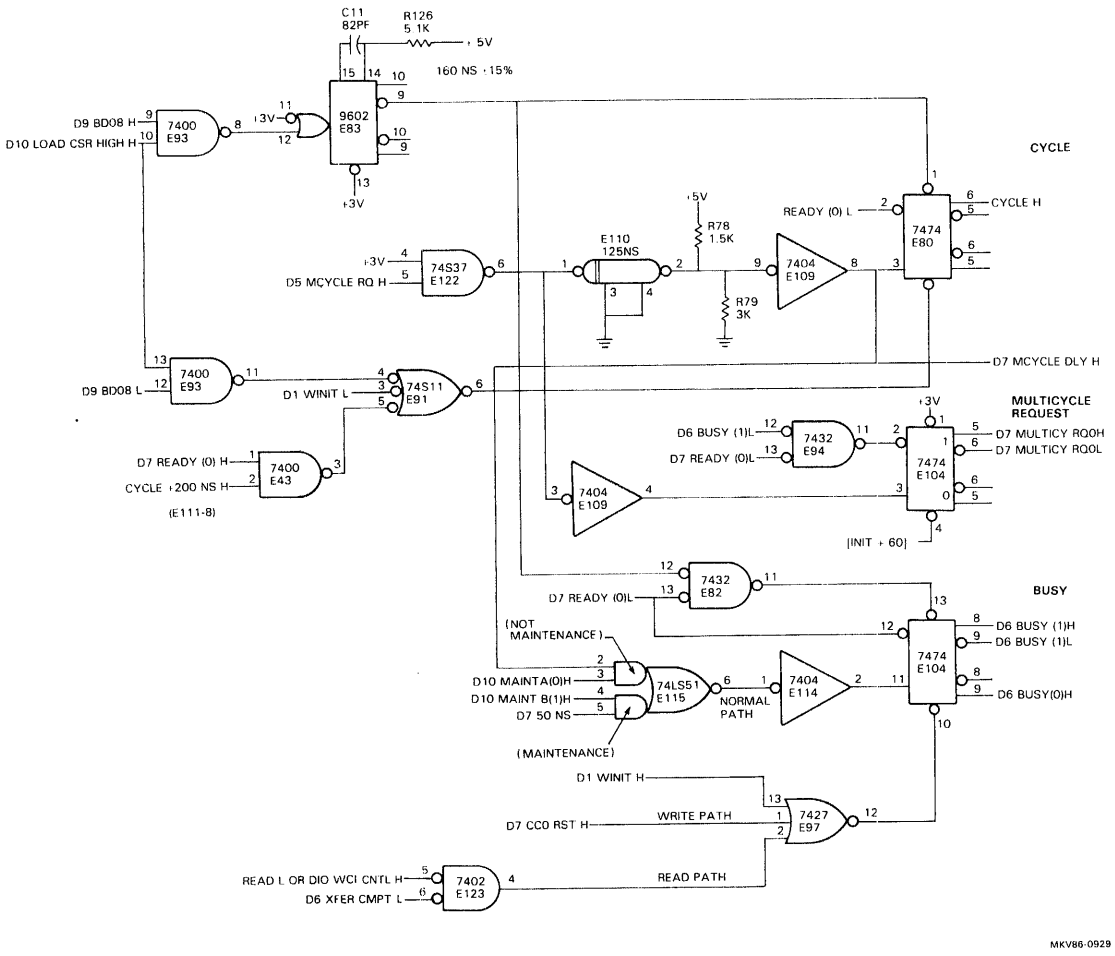

Figure 3-15 Cycle and Busy Logic Diagram

The DR11-W logic then allows approximately 200 nanoseconds for the deskew of all of the inputs mentioned above. The cycle flip-flop is then set and the DMA cycle starts (see Figures 3-16 and 3-17). CYCLE RQ x H need not be held until this occurs, but a minimum pulse width of 125 nanoseconds is recommended. There is no maximum time limit on CYCLE RQ x H (other than the fact that it must be deasserted before it can be reasserted).

CYCLE RQ x H also sets the busy flip-flop, driving the busy signal. This signal stays asserted until the DR11-W is ready to accept the next assertion of CYCLE RQ x H.

All of the user inputs mentioned above are latched approximately 200 nanoseconds after CYCLE RQ x H has been asserted. After this time, the user inputs may be removed or updated. If the cycle is a write to memory, this includes the user data inputs (DI 00 H to DI 15 H).

The DR11-W DMA engine begins the DMA cycle by asserting the UNIBUS signal BUS NPR L. This indicates that the DR11-W would like to become the UNIBUS master in order to perform a DMA data-transfer cycle. Eventually, the signal BUS NPG H is returned to the DR11-W, indicating that it is the next selected UNIBUS master. The interval between the requesting and the granting of the UNIBUS is dependent upon the DR11-W's position in the UNIBUS priority chain as well as the overall I/O load on the UNIBUS.

Once the DR11-W receives BUS NPG H, it asserts BUS SACK L, indicating that it acknowledges selection as the next UNIBUS master. The DR11-W then waits for the current UNIBUS master to complete its transactions. To do this, the DR11-W monitors the UNIBUS line BUS BBSY L. When BUS BBSY L deasserts, the current UNIBUS master is finished. The DR11-W then asserts BUS BBSY L for itself, indicating that it is the current UNIBUS master.

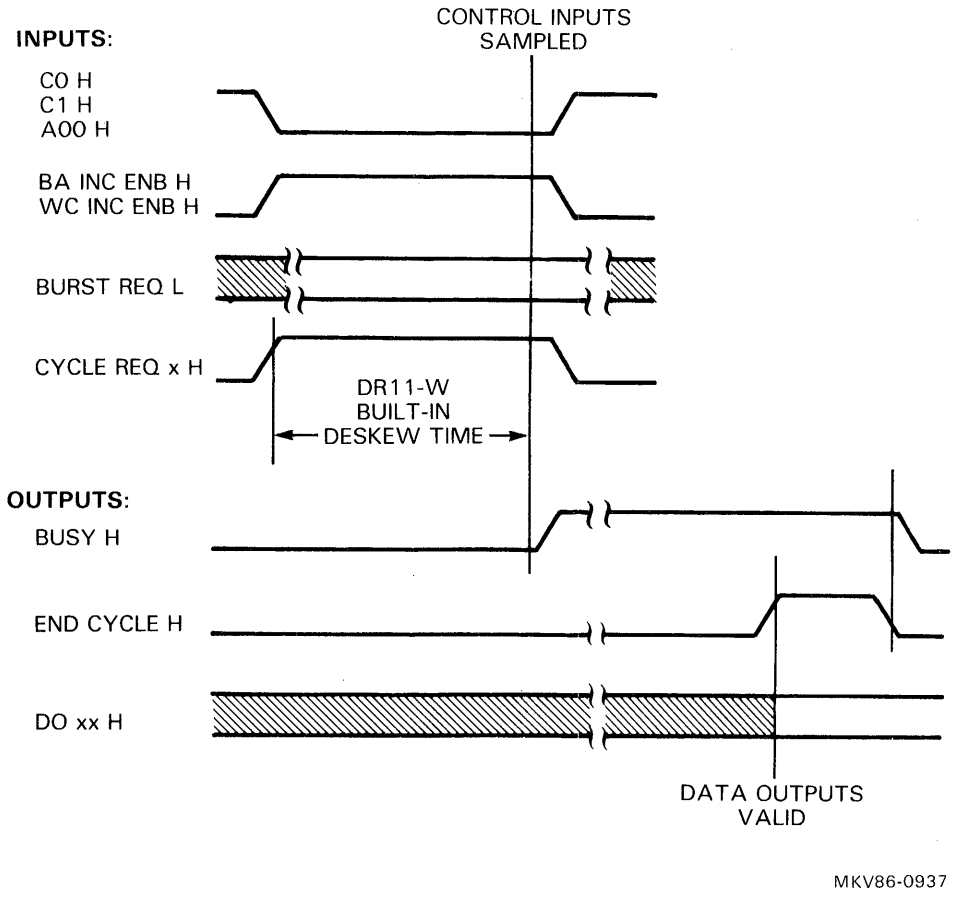

Figure 3-16 Simplified Read-Cycle Timing Diagram

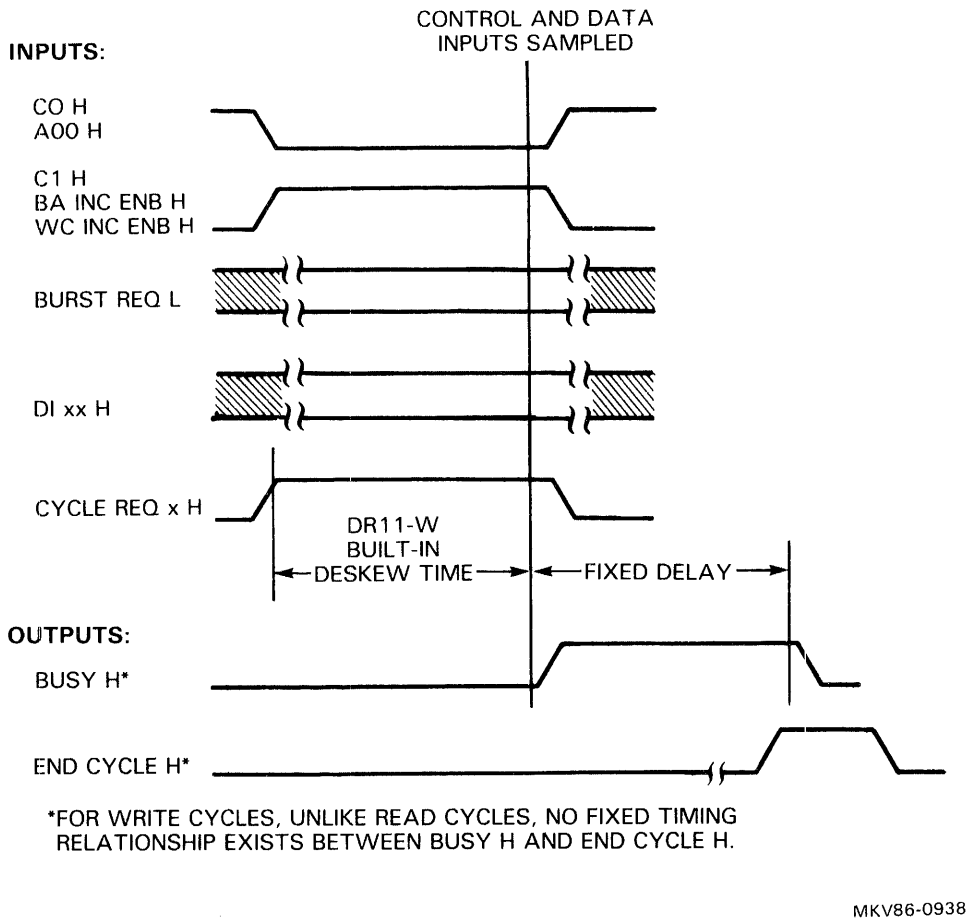

Figure 3-17 Simplified Write-Cycle Timing Diagram

The DR11-W DMA engine can now perform the actual DMA requested by the user.

If the DMA cycle is a read from memory, the completion of the DMA cycle clears the busy flip-flop, deasserting the busy signal to the user. It is also at this time that the valid data is present on the DR11-W output lines (DO 00 H to DO 15 H). The data remains valid until the next time that CYCLE RQ x H is asserted by the user.

If the DMA cycle is a write to memory, the busy signal clears prior to the completion of the cycle. In this case, the END CYCLE H pulse indicates that the actual DMA cycle has been completed. Note that for write cycles you may still reassert CYCLE RQ x H as soon as busy clears – the DR11-W holds your request until it is finished with the current DMA cycle.

3.11 POSSIBLE FAILURES

If the DR11-W is attempting to access memory which does not exist, the DMA engine detects that failure and causes the DR11-W to become ready and not busy. Once the DR11-W becomes ready, asserting CYCLE RQ x H does not start further DMA cycles. The software is explicitly told that a non-existent memory error occurred – the user device is not told.

Similarly, if the DR11-W reads from a memory location which contains bad parity, the DMA engine detects that failure as well. Once again, the DR11-W becomes ready and refuses further assertions of CYCLE RQ x H. The software is explicitly told that a parity error occurred – the user device is not told.

If the user device mistakingly reasserts CYCLE RQ x H while the DR11-W is still in the midst of a previous DMA cycle (that is, busy is still asserted), then a multi-cycle error is declared. This, like all other errors, stops the DR11-W and is indicated to the software. Multi-cycle error is roughly analagous to the data-late errors reported by disk and tape controllers.

If the user device asserts ATTN H, the DR11-W stops and an attention error is reported. This may or may not really represent an error, depending on your application.

Finally, if an impending power failure is detected on the UNIBUS (indicated by the assertion of BUS ACLO L), then the DR11-W is stopped and an AC LO error is reported. The occurence of this error **is** visible to the user device via the UACLO FNCT 2 H line.

The expiration of the burst release timer is not an error which will stop the DR11-W. It simply results in the release of the UNIBUS (until the next CYCLE RQ x H). If the software is polling the EIR, the fact that the timer timed out can be seen. The burst release timer is described further in Chapters 2 and 6.

3.12 BYTE TRANSFERS

By itself, the DR11-W is not capable of transferring single bytes of data to and from memory. This becomes possible with the addition of some hardware in the user device. Refer to Section 3.4.

3.12.1 Reading Bytes from Memory

The UNIBUS does not implement a specific read byte operation, since the read word operation is adequate for the job. Three approaches are possible for reading byte data.

3.12.1.1 Simplest and Slowest – The simplest approach to byte transfers is to ignore them (that is, treat all transfers as word transfers). If the data rate is low enough, the DR11-W can be allowed to transfer full words to or from memory, where only half of the word contains the useful data. This requires absolutely no hardware but does require that the processor software unpack the DR11-W's data buffer before starting the transfer, and that the data buffer be twice as large (see Figure 3-18).

3.12.1.2 A Faster Hardware-Based Approach – See Figure 3-19. This circuit enables the reading of bytes, one byte per UNIBUS cycle. Note that this circuit assumes that every block transfer starts on an even UNIBUS byte address and all transfers are byte transfers. The flip-flop presetting logic must be changed to accommodate starting at an odd byte.

3.12.1.3 More Complex and Fastest – The circuit shown in Figure 3-19 requires one UNIBUS cycle per byte transfered. This wastes the other byte that is produced by every UNIBUS cycle. A more complex circuit latches the unused byte, saving it for the next cycle. This cuts the DR11-W's UNIBUS traffic in half. Also, since the second byte is already in local storage, it can be produced very quickly.

When designing a circuit like this, caution must be taken to ensure that old data stored in the latches from a previous cycle is never presented, masquerading as new data. This requires careful attention to the handling of boundary conditions and any transfer errors which may arise.

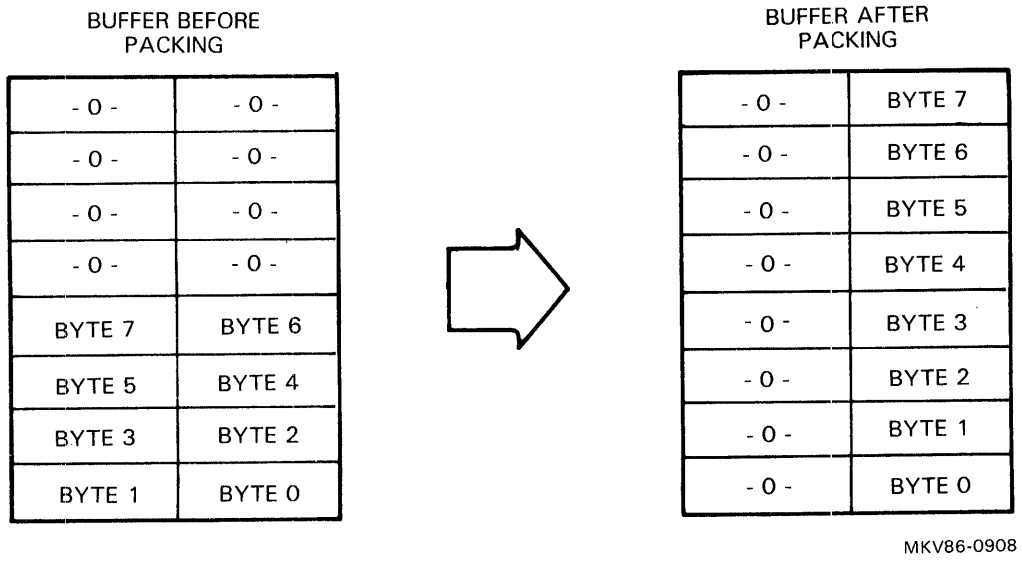

Figure 3-18 Unpacking a Buffer Prior to Reading Bytes from Memory

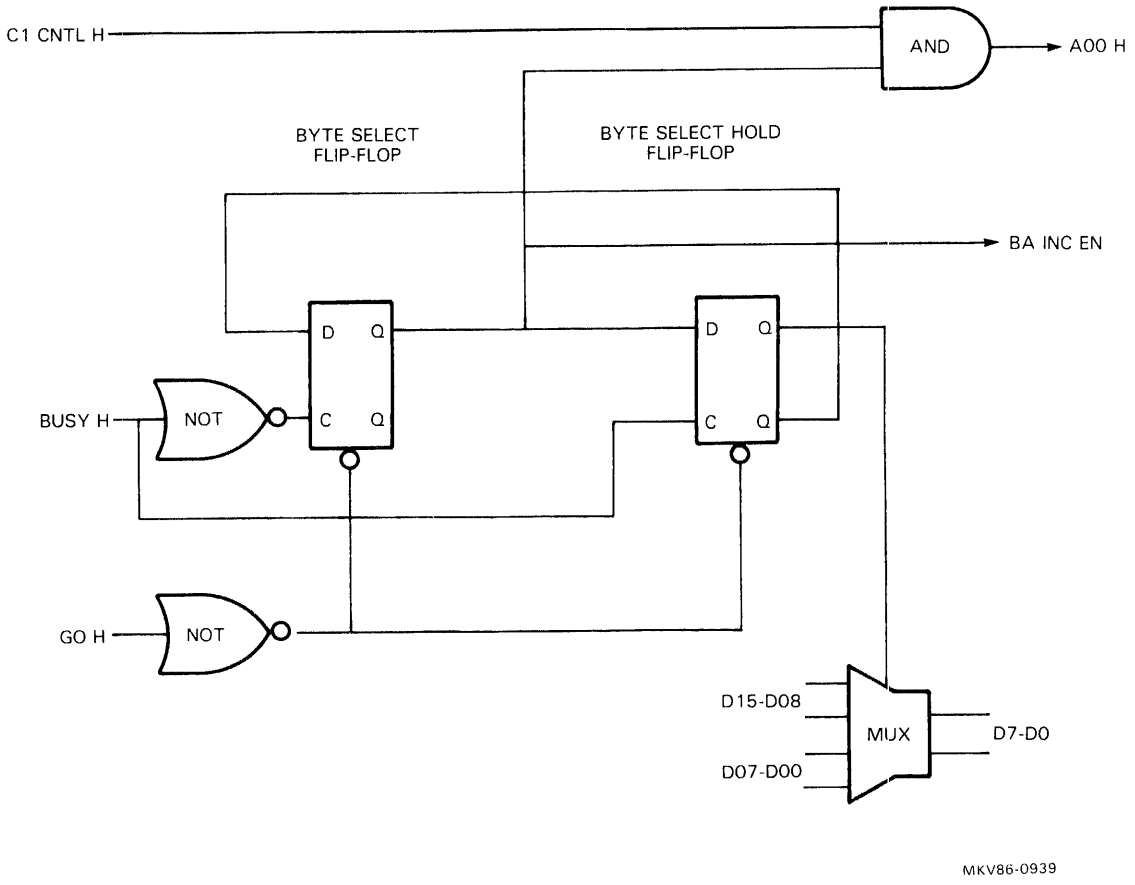

Figure 3-19 Logic Diagram of User-Supplied Hardware to Read One Byte per DMA Cycle

The circuit shown in Figure 3-20 implements such a design. Like the circuit shown in Figure 3-19, it uses the simplifying assumption that all transfers begin on an even byte address. The circuit also assumes that USER CYCLE RQ H lasts at least until the DR11-W busy signal asserts.

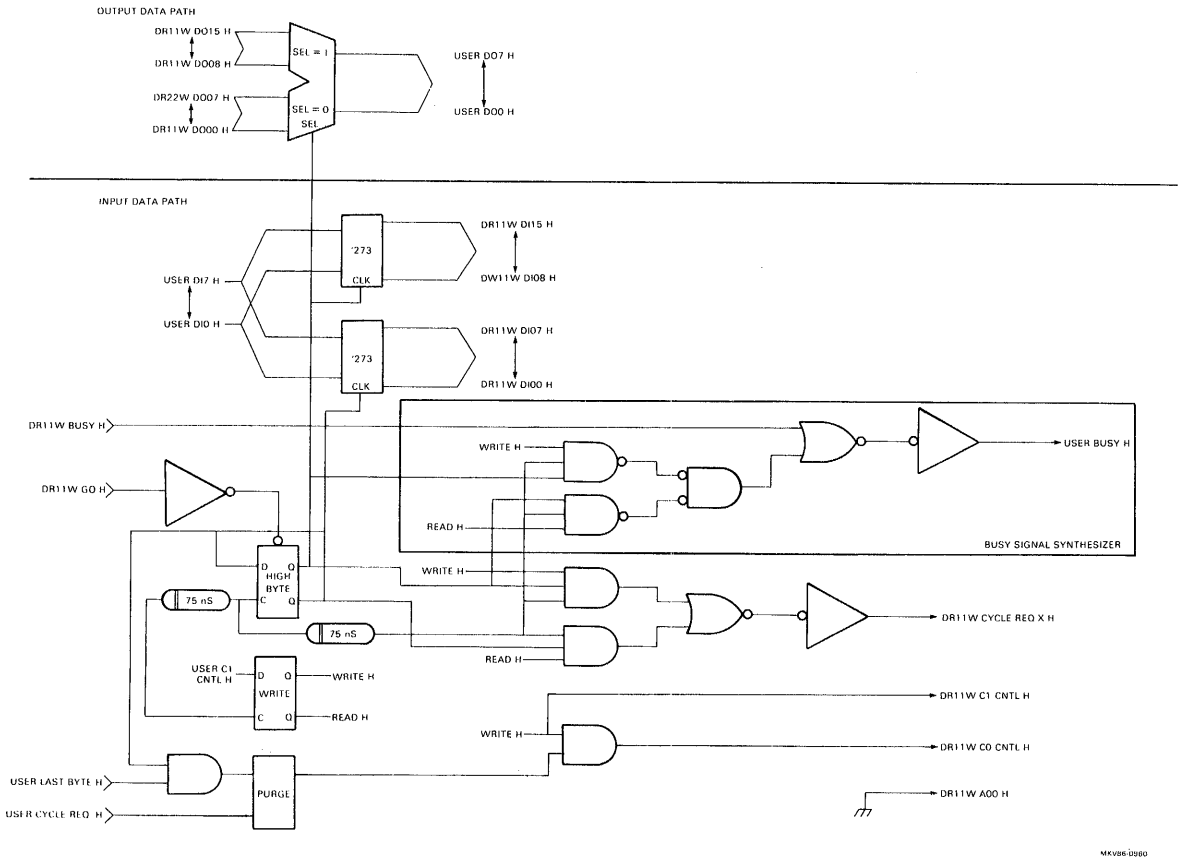

Figure 3-20 Logic Diagram of User-Supplied Hardware to Read Two Bytes per DMA Cycle

3.12.2 Writing Bytes

Writing bytes is similar, and the same three alternatives exist for writing bytes as for reading them.

3.12.2.1 Simplest and Slowest – Once again, the simplest approach to byte transfers is to ignore them. This time, the processor must pack the DR11-W's data buffer after the transfer completes, and the data buffer must still be twice as large (see Figure 3-21).

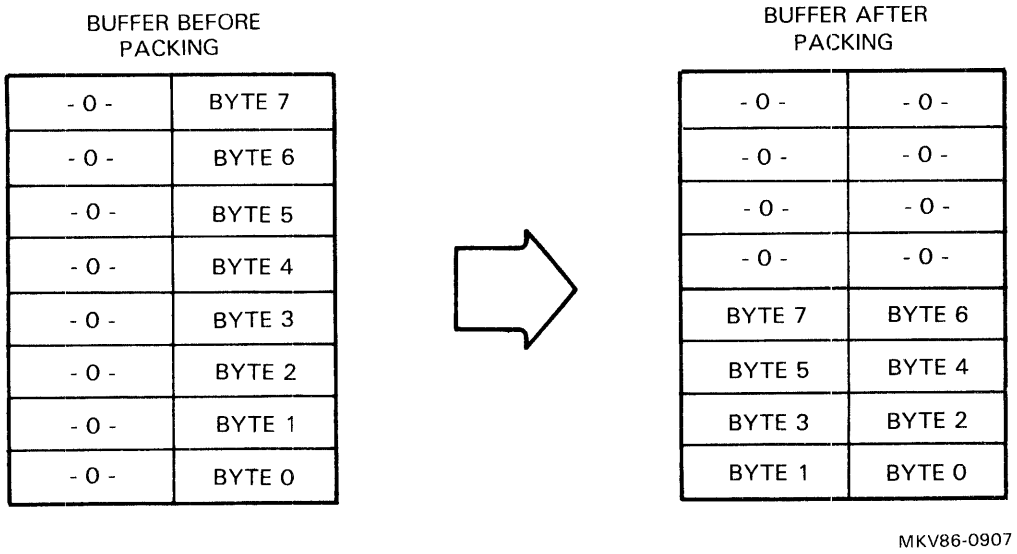

Figure 3-21 Packing a Buffer After Writing Bytes to Memory

3.12.2.2 A Faster Hardware-Based Approach – The same circuit illustrated in Figure 3-19 also works for writing bytes to memory. Here, no data demultiplexer is required – the byte of data from the user device may simply be duplicated onto both bytes of the DR11-W input data. That is, user data bit ⟨7⟩ should be presented on both DI 15 H and DI 7 H. Logic in the addressed UNIBUS slave device decides which byte to update, based on A00. The same simplifying assumptions still apply to the circuit: every transfer starts on an even UNIBUS byte address and all transfers are byte transfers.

3.12.2.3 More Complex and Fastest – A similar leap in complexity allows two byte writes to be compressed into one word write. Here, boundary and error conditions take on even more significance than when reading bytes since it is imperative that a latched byte actually get written to memory, not thrown away if the DMA happens to end on the wrong boundary.

3.13. PROCESSOR CONTROL OF TRANSFER DIRECTION

The UNIBUS signals BUS C0 L and BUS C1 L select which type of UNIBUS transfer will be performed (DATI, DATIP, DATO, or DATOB) and therefore which way the data will be moved. The DR11-W drives these signals based on the user inputs C0 CNTL H and C1 CNTL H. This allows the user device to select for each and every individual DMA cycle what type of cycle will occur. Sometimes this level of sophistication is not needed and it would be just as easy to let the processor make the choice. This can be done by choosing one of the FUNCT x H outputs and connecting it back to the C1 CNTL H line. See Figure 3-22. This FUNCT x H output then selects read or write cycles. If the FUNCT x H bit is 0, then the DR11-W reads words from memory. If the FUNCT x H bit is 1, then the DR11-W writes words to memory.

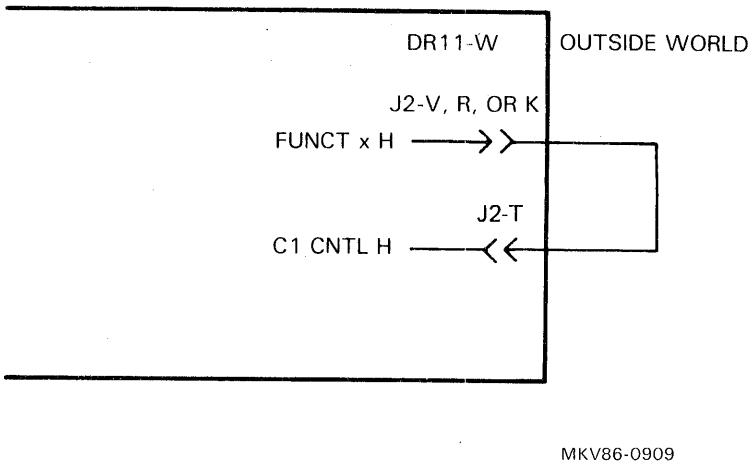

Figure 3-22 Connections for Processor Control of Transfer Direction

This will only work for the basic DATI (read) and DATO (write word) cycles. If C0 must be controlled, the user device must contain additional hardware.

3.14 PROCESSOR CONTROL OF BURST MODE

In a similar fashion, the processor can be made to control BURST RQ L. Once again, this can be done by choosing one of the FUNCT x H outputs and connecting it back to the burst RQ L line (see Figure 3-23). This FUNCT x H output then selects burst or non-burst cycles. (The toggle switch on the DR11-W module still selects whether the burst cycles are two-word or n-word bursts.) If the FUNCT x H bit is 0, then the DR11-W operates in burst mode. If the FUNCT x H bit is 1, then the DR11-W operates in non-burst mode.

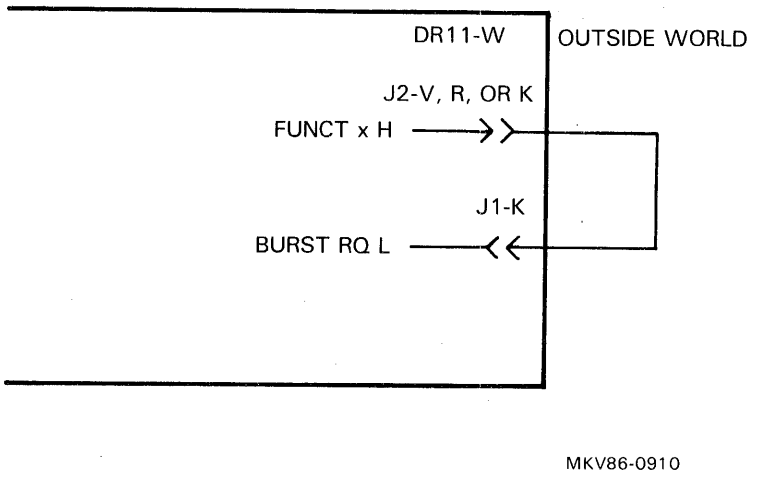

Figure 3-23 Connections for Processor Control of Burst Mode

3.15 USER DEVICE CONTROL OF BURST MODE

Burst mode is useful whenever the data rate is high or the UNIBUS is busy. Burst mode causes the DR11-W to transfer two or more words each time it becomes the UNIBUS master. This improves the system's overall throughput, since less time is spent arbitrating for, acquiring, and releasing the UNIBUS.

Burst mode should only be used if the user device is capable of delivering data to the DR11-W as quickly as the DR11-W can transfer that data on the UNIBUS. If the DR11-W idles while holding the UNIBUS, then burst mode hurts system performance.

If you decide to never use burst mode, then no connection need be made to the BURST RQ L line at the DR11-W user connectors. If you are using a flat cable (which must connect to the BURST RQ L pin), then the user device should provide a pull-up terminator for this line (see Figure 3-24). Do not let BURST RQ L float at the user device-end of the cable.

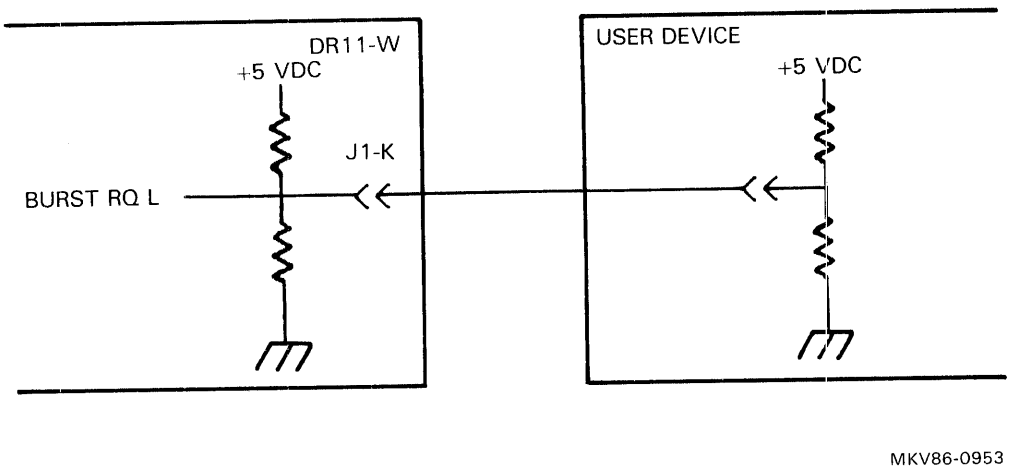

Figure 3-24 Circuit for a User Device Which Never Uses Burst Mode

If you decide to always use burst mode, then simply ground BURST RQ L at the user device (see Figure 3-25). The DR11-W contains sufficient logic to terminate bursts when the overall transfer ends (for example, if you are using two-cycle bursts and transfer an odd number of words, the DR11-W is smart enough to get off the UNIBUS, not wait forever for the next word to come along and even up the count).

If your device uses a first-in, first-out silo (FIFO), a useful approach is to control burst mode dynamically, based on how full your silo becomes (see Figure 3-26). As long as the silo remains relatively empty, the silo logic causes the transfers to be performed in non-burst mode. If the silo begins to fill, the silo logic asserts BURST RQ L so that more than one word is transferred each time the DR11-W becomes master of the UNIBUS. This allows the silo to empty faster.

Remember, in order to use burst mode effectively, your device must be able to respond quickly to the DR11-W each time the DR11-W completes a DMA cycle.

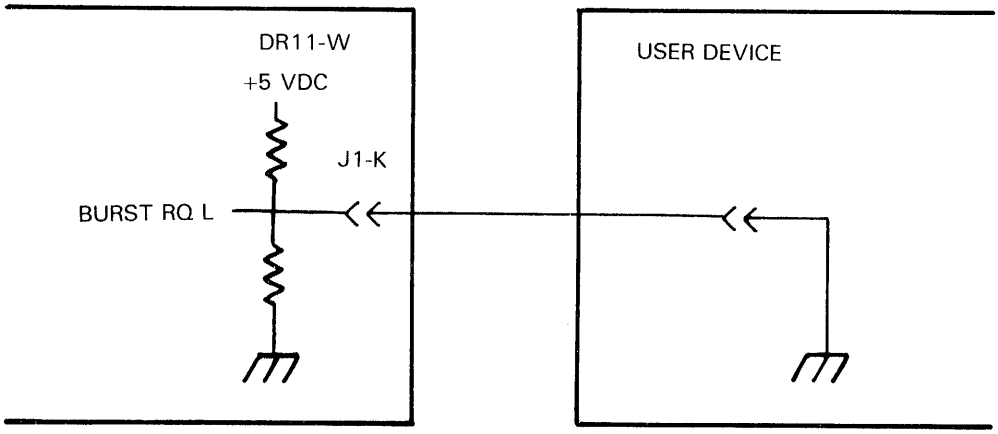

Figure 3-25 Circuit for a User Device Which Always Uses Burst Mode

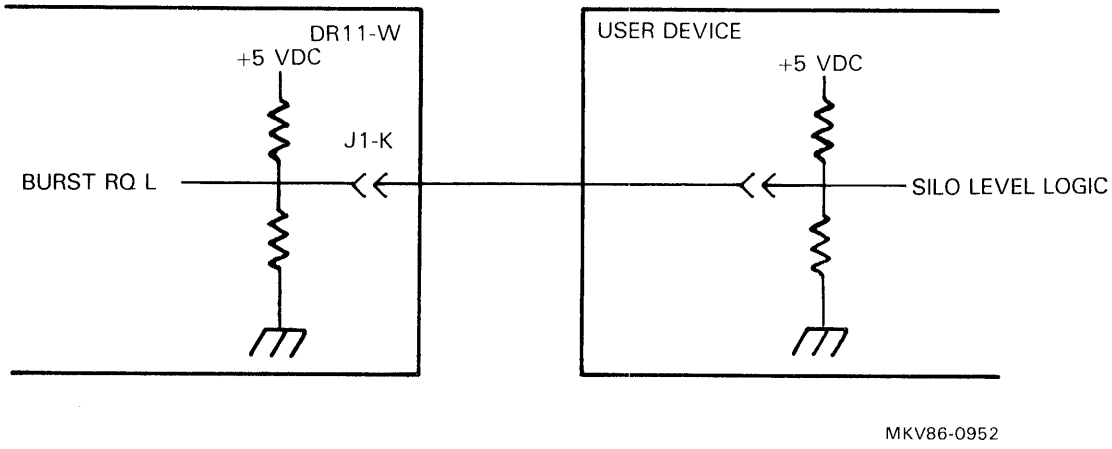

Figure 3-26 Circuit for a User Device Which Dynamically Uses Burst Mode

CHAPTER 4
PROGRAMMING THE DR11-W

4.1 INTRODUCTION

This chapter covers the programming of the DR11-W in general, and specifically when used to interface to a user device. Specific considerations for links are discussed in Chapter 5.

4.2 REGISTERS

The DR11-W is controlled by six registers occupying four UNIBUS addresses. These registers are:

- The word or byte count register.
- The bus address register.
- The control and status register.
- The error and information register.
- The input data register.
- The output data register.

4.2.1 Word Count Register

The word count register limits the number of DMA cycles to be performed (see Figure 4-1). Prior to setting the go bit, this register must be loaded with the twos-complement (negative) of the number of words or bytes to be transferred. If the DR11-W is transferring one word per UNIBUS DMA, then this register acts as a word-count register. If the DR11-W is transferring one byte per DMA, and the user device never drives WC INC ENB H, then the register acts as a byte count.

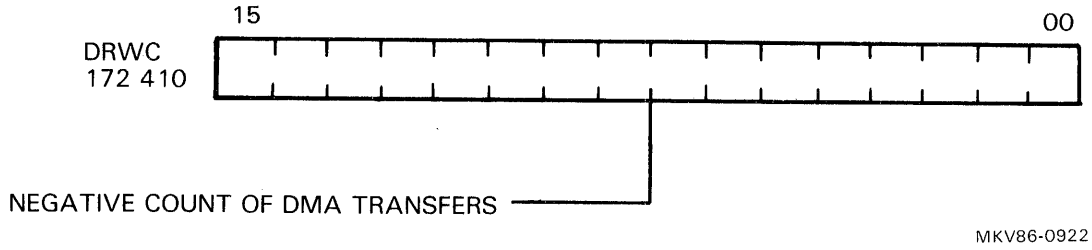

Figure 4-1 Word Count Register

This register is a read/write register located at 772 410 for the first DR11-W in the system. The word count register must be written to as an entire word using word instructions.

4.2.2 Bus Address Register

The bus address register provides UNIBUS address bits ⟨15⟩ through ⟨01⟩ (see Figure 4-2). These bits are combined with XBA17 and XBA16 (found in the control and status register) and A00 H (provided by the user device) to form the complete UNIBUS address of the next DMA transfer to be performed (see Figure 4-3). Prior to setting the go bit, this register must be loaded with the address of the first word to be transferred.

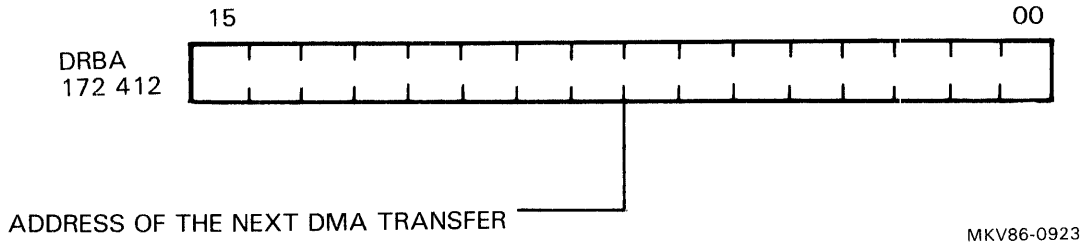

Figure 4-2 Bus Address Register

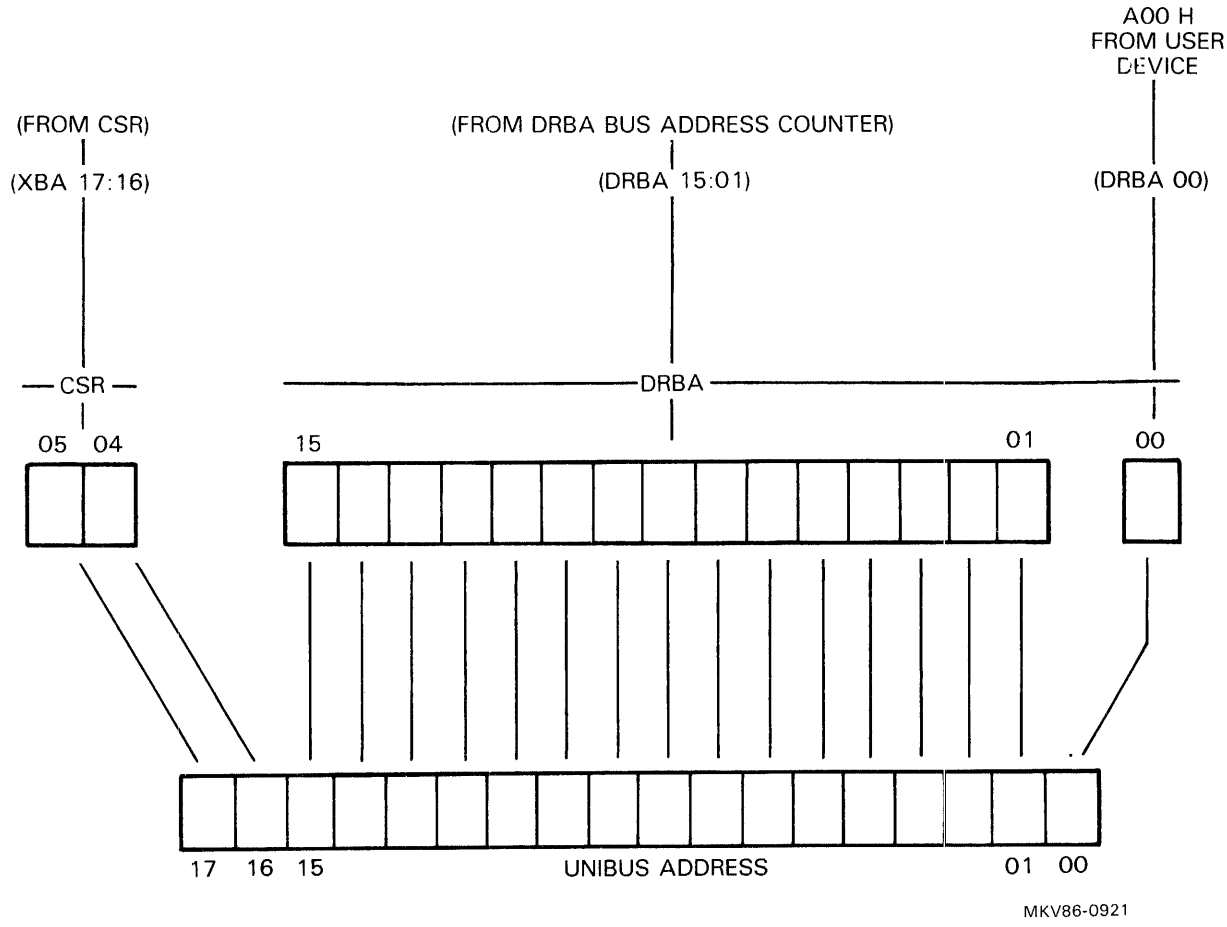

Figure 4-3 Formation of an 18-Bit UNIBUS Address

This register is located at 772 412 for the first DR11-W in the system. Bits ⟨15:01⟩ are read/write. Bit ⟨00⟩ is a read-only bit that may reflect the state of the A00 H input from the user connectors (see Section 2.4.3.2) . The program cannot directly control this bit.

The bus address register must be written to as an entire word using word instructions.

4.2.3 Shared Address 772 414

Two registers use offset address 4 (772 414 for the first DR11-W in the system). One register is the control and status register; the other is the error and information register.

Writing to address 772 414 always writes the control and status register. The bits in the error and information register cannot be directly written into.

What happens when the software reads from address 772 414 depends on the setting of E105 switch 5.

If E105 switch 5 is open, then reading from address 772 414 always reads the control and status register. It is just as though the error and information register did not exist at all.

However, if E105 switch 5 is closed, then reading from address 772 414 may read either the control and status register or the error and information register, depending on the state of an internal DR11-W flip-flop (see Figure 4-4). Writing bit ⟨15⟩ of the combined register sets or clears this flip-flop. The flip-flop's state is displayed by reading bit ⟨00⟩ of the combined register.

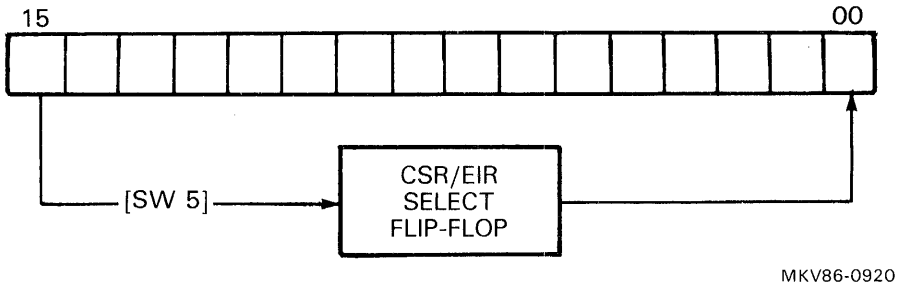

Figure 4-4 CSR/EIR Flip-Flop

4.2.4 Control and Status Register

The control and status register is the principal register used by the software to command the DR11-W (and the user device) and to obtain the status of the DR11-W (and the user device). See Figure 4-5.

The control and status register is located at offset address 4 (that is address 772 414 for the first DR11-W in your system). Writing to this address always writes the control and status register. Reading from this address may read the control and status register or the error and information register. Refer to Section 4.2.3 for details.

The CSR may be written to using either word or byte instructions.

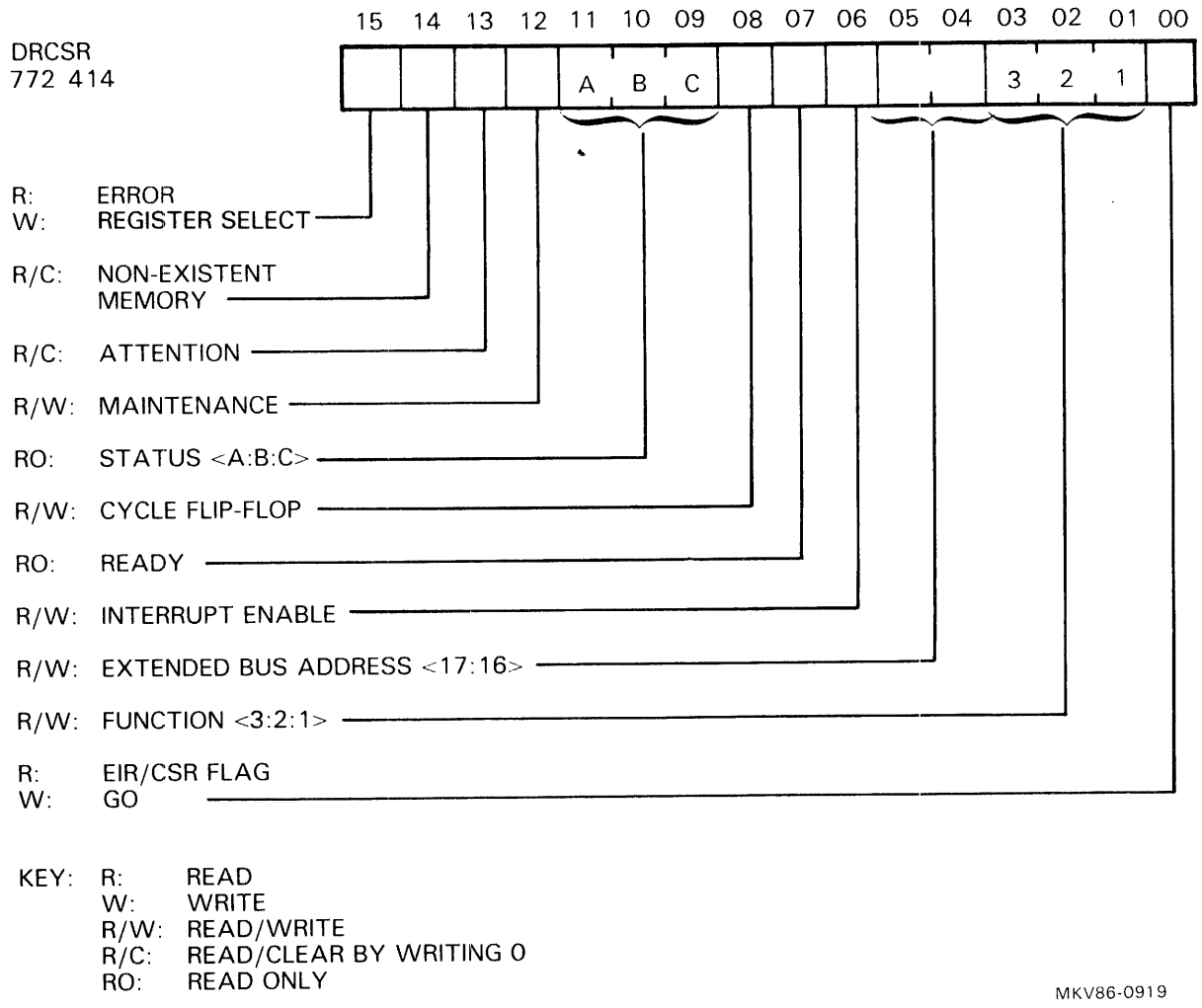

Figure 4-5 Control and Status Register

The description of the bits of the control and status register follows:

CSR Bit ⟨15⟩
Error

Reading this bit presents the logical OR of all possible error conditions.
If E105 switch 5 is open, then writing to this bit has no effect.
If E105 switch 5 closed, then writing to this bit selects whether the CSR or the EIR is displayed at this address.

[Reading bit 00 (the go bit) displays the contents of the flip-flop which selects CSR versus EIR.]
Writing a 1 to bit ⟨15⟩ causes the EIR to be read from address xxxxx4.
Writing a 0 to bit ⟨15⟩ causes the CSR to be read from address xxxxx4.
Partially for this reason, you must never use a read/modify/write instruction (that is, INC, BIS, BIC, etc) directly on the CSR. If the error bit (bit ⟨15⟩) is set, writing that 1 back to the CSR inadvertently switches to the EIR.

Similarly, if the EIR flag (bit ⟨00⟩) is set, writing that 1 back to the go bit causes the DR11-W to begin a DMA transfer. An example of bad code:

```
MOV     @#DRCSR, R0     ; Get the CSR
BIC     #100, R0        ; We'll clear the 'IE' bit
MOV     R0, @#DRCSR     ; Writeback the modified CSR
```

To avoid inadvertantly setting bits, the code must always write to the CSR with 0s in bit positions ⟨15⟩ and ⟨00⟩ (unless you really want to switch to the EIR or set go). An example of code that works:

```
MOV     @#DRCSR, R0     ; Get the CSR
BIC     #100101, R0     ; We'll clear the 'IE' bit,
                        ; and stay with the CSR
MOV     R0, @#DRCSR     ; Writeback the modified CSR
```

When E105 switch 5 is open, using R/M/W instructions does not cause this problem. Read the discussion under CSR bit ⟨13⟩ to understand a problem that R/M/W instructions can still cause, even with E105 switch 5 open.

CSR Bit ⟨14⟩
Non-Existent
Memory

This read/clear bit indicates that the DR11-W attempted to access non-existent memory during the last DMA cycle.

The bit can be cleared by writing a 0 to it, by asserting go, by the UNIBUS signal BUS INIT L, or by a DR11-W internal reset (caused by clearing the maintenance bit).
Writing a 1 to this bit has no effect.

CSR Bit ⟨13⟩
Attention

This read/clear bit indicates that the user device requires service from the software. Attention causes the DR11-W to interrupt the processor if the interrupt enable bit is set.
The bit can be cleared by writing a 0 to it, by asserting go, by the UNIBUS signal BUS INIT L, or by a DR11-W internal reset (caused by clearing the maintenance bit).
Writing a 1 to this bit has no effect.
Upon the rising edge of the ATTENTION H signal, the DR11-W latches two copies of the attention bit. One is used to generate the interrupt while the other is used for presentation in the CSR. Both are cleared by writing a 0 to bit ⟨13⟩ of the CSR. These flip-flops are thus read/clear. In addition, the CSR also receives the attention signal directly. This allows the user device to simply pulse the attention line (rather than having to hold it for handshaking). Unfortunately, it also allows the software to easily lose an attention interrupt by simply writing the CSR with a 0 in bit-position ⟨13⟩. Partially for this reason, you must never use a read/modify/write instruction (that is, INC, BIS, BIC, etc) directly on the CSR. In fact, you must be careful even while simulating a R/M/W instruction. An example of bad code:

```
MOV     @#DRCSR, R0     ; Get the CSR
BIS     #4, R0          ; We'll set the 'F2' bit
MOV     R0, @#DRCSR     ; Writeback the modified CSR
```

If attention asserted after we read the CSR, but before we wrote it back, the write-back would destroy the attention bit and attention interrupt. To eliminate this, the code must always write to the CSR with a 1 in bit position ⟨13⟩. One solution:

```
MOV     @#DRCSR, R0     ; Get the CSR
BIS     #010004, R0     ; We'll set the 'F2' bit,
                        ; but don't clear ATTN
MOV     R0, @#DRCSR     ; Writeback the modified CSR
```

Before you code this, read the discussion under CSR bit ⟨15⟩. If you are using the DR11-W with E105 switch 5 closed, the techniques explained under that heading must also be applied. An example of the combination:

```
MOV     @#DRCSR, R0     ; Get the CSR
BIS     #010004, R0     ; We'll set 'F2' without
                        ; killing ATTENTION
BIC     #100001, R0     ; Stay with the CSR, no 'GO'
MOV     R0, @#DRCSR     ; Writeback the modified CSR
```

Another alternative may be to write to the CSR with byte instructions directed at the low byte. Then, the only risk is from the EIR flag in bit ⟨00⟩. But since you are not going to inadvertantly select the EIR, the EIR flag should stay clear.

Note that additional attention interrupts can be generated prior to the software clearing the attention bit in the CSR. Each time a rising edge occurs on the attention H line, the DR11-W initiates an interrupt (unless an interrupt is already pending).

CSR Bit ⟨12⟩
Maintenance

Setting this read/write bit establishes an internal wrap-around of the output data to the data inputs. This allows testing without a turn-around cable. The function bits are also incremented with each word DMAed. This causes read and write cycles to alternate, as well as burst and non-burst cycles.

CAUTION

All of the DR11-W outputs remain enabled. Unpredictable data patterns are generated during testing.

Clearing the maintenance bit performs an internal DR11-W reset, analagous to the effects of the UNIBUS signal BUS INIT L. The internal reset lasts approximately 600 nanoseconds. A fast processor can clear the maintenance bit and attempt another access to the DR11-W during the internal reset. Do not expect reasonable results if you access the DR11-W during its reset. The maintenance bit need not be used for most cable wrap-around tests.

CSR Bits ⟨11:09⟩
Status ⟨A:B:C⟩
(for user devices)

Three general-purpose, read-only input bits. During link operation, these bits have specific uses. See Chapter 5 for details.

CSR Bit ⟨08⟩
Cycle Flip-Flop

This read/write bit controls the cycle flip-flop within the DR11-W.
Writing a 1 to this bit always sets the cycle flip-flop (similar to the user inputs cycle RQ x H, which set the cycle flip-flop if the DR11-W is not ready).
Writing a 0 to this bit always clears the cycle flip-flop.
Reading this bit indicates the state of the cycle flip-flop.
Rule: Because the user device is able to asynchronously set the cycle flip-flop (using CYCLE RQ x H), the software should never write to the CSR once the go bit is set. There is no fixed timing relationship between CYCLE RQ x H, XBA17 and XBA16, the software, and the DR11-W hardware so there is no way for the software to predict whether it should write a 1 or a 0 to the cycle bit.
Exception to the rule: For links, the sending end must set CYCLE once after the go bit has been set. See Chapter 5 for details.

CSR Bit ⟨07⟩
Ready

This read-only bit indicates that the software may set up the DR11-W registers for the next DMA transfer. Ready is also provided to the user device.

CSR Bit ⟨06⟩
Interrupt Enable

Setting this read/write bit allows the DR11-W to interrupt the processor each time the ready bit sets.

Setting interrupt enable while an attention interrupt is being requested (that is, while attention is set) immediately results in an interrupt.

CSR Bits ⟨05:04⟩
XBA17, XBA16

These two read/write bits are combined with the bits in DRBA and the A00 H line from the user device to form a full 18-bit UNIBUS address. These bits are incremented as the DRBA overflows. Refer to the description under cycle bit CSR bit ⟨08⟩.

CSR Bits ⟨03:01⟩
Function ⟨3:1⟩

These three read/write bits provide three general-purpose output lines to the user device. When used as part of a link, these bits have specific uses. See Chapter 5 for details.

CSR Bit ⟨00⟩
Go/EIR Flag

Writing a 1 to this bit while the error bit is clear causes the DR11-W DMA engine to be enabled, and the ready bit cleared. A 160 nanosecond pulse is provided to the user device. Subsequently setting software cycle (bit ⟨08⟩) or asserting CYCLE RQ x H causes one DMA cycle to begin.

Writing a 1 to this bit while the error bit is set clears the error bit, immediately causes the DR11-W to become ready again, and requests an interrupt if interrupt enable is set. The GO H pulse is still sent to the user device.

Writing a 0 to this bit has no effect.

Reading this bit indicates whether the CSR or EIR is being displayed at this address. If the CSR is being displayed, this bit is read as a zero. If the EIR is being displayed, this bit is read as a one.

Partially for this reason, you must never use read/modify/write instructions with the CSR/EIR. If the EIR is currently being displayed, you might inadvertantly set the go bit. Read the discussion under CSR Bit ⟨15⟩.

4.2.5 Error and Information Register

The alter ego of the control and status register is the error and information register. If E105 switch 5 is closed, the program can select this register to be read in place of the CSR. Writing to the address still writes the CSR (see Figure 4-6).

The description of the bits of the error and information register follow:

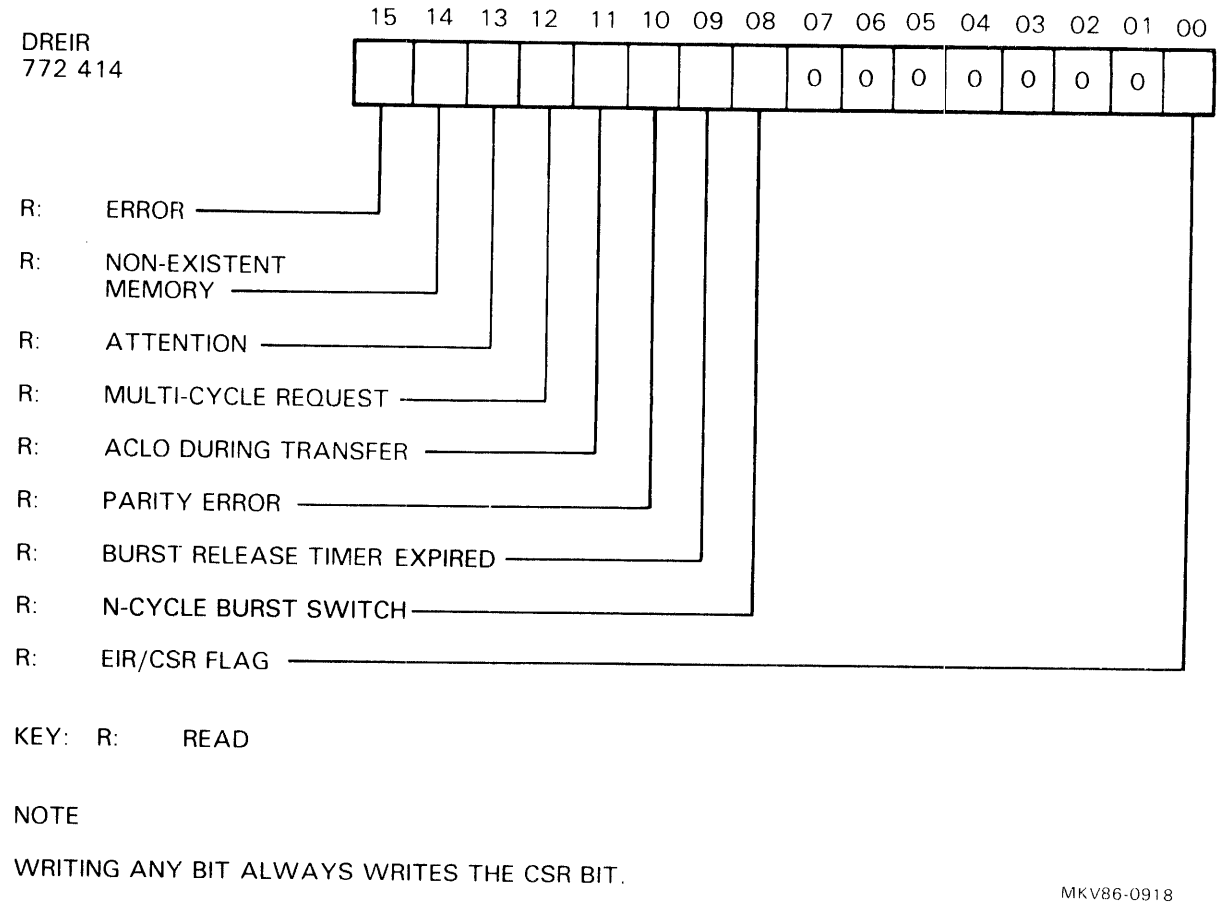

Figure 4-6 Error and Information Register

EIR Bit ⟨15⟩ Error	Same as CSR bit ⟨15⟩. It is duplicated in the EIR for easy access.
EIR Bit ⟨14⟩ Non-Existent Memory	Same as CSR bit ⟨14⟩. It is duplicated in the EIR for easy access.
EIR Bit ⟨13⟩	Same as CSR bit ⟨13⟩. It is duplicated in the EIR for easy access.
EIR Bit ⟨12⟩ Multicycle Request	This bit indicates if the user device requested that another transfer be started while the previous transfer was still in progress. This is analogous to the classical data-late error.
EIR Bit ⟨11⟩ AC LO	This bit indicates if a power failure occurred on the local UNIBUS during the last transfer.
EIR Bit ⟨10⟩ Parity Error	This bit indicates if the DR11-W read from memory a word with incorrect parity or an uncorrectable ECC error.
EIR Bit ⟨09⟩ Burst Release Timer Expired	This bit indicates if the user device is supplying data too slowly to justify the DR11-W staying the UNIBUS master. If set, the DR11-W has temporarily relinquished the bus. This is a non-fatal condition. The time-out value is adjustable on the module.
EIR Bit ⟨08⟩ N-Cycle Burst Switch	This bit indicates the position of the burst-size toggle switch. One indicates that the switch is in the n-cycle position. Zero indicates that the switch is in the two-cycle position.
EIR Bits ⟨07:01⟩ Unused	These bits read as 0 any time the EIR is being displayed.
EIR Bit ⟨00⟩ Register Flag	The DR11-W signals that the EIR is being displayed (rather than the CSR) by displaying a 1 in bit position ⟨00⟩. Any time the CSR is being displayed, bit ⟨00⟩ reads as a 0.

4.2.6 Input Data Register

The input data register is a read-only register sharing its address with the output data register (a write-only register) (see Figure 4-7).

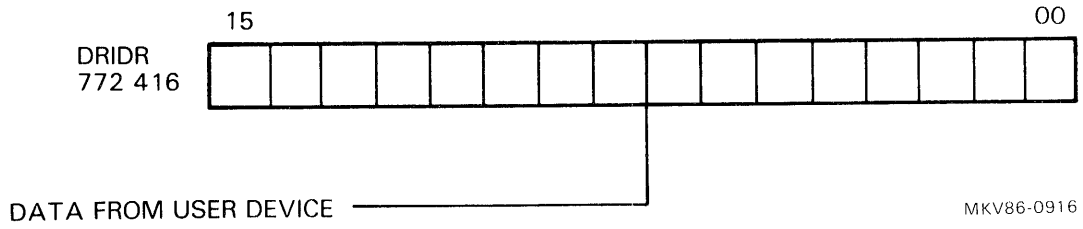

Figure 4-7 Input Data Register

Data is latched into the IDR any time a DMA write cycle is started or the IDR is explicitly read. If the CSR/EIR is currently displaying the EIR, explicitly reading the IDR does not clock in new data. This allows the program to read the last word transferred in by the DMA (ordinarily this data would be invisible to the program). If the EIR is disabled (via E105 switch 5), then the IDR always clocks in new data whenever explicitly read.

4.2.7 Output Data Register

The output data register is a write-only register sharing its address with the input data register (a read-only register). See Figure 4-8.

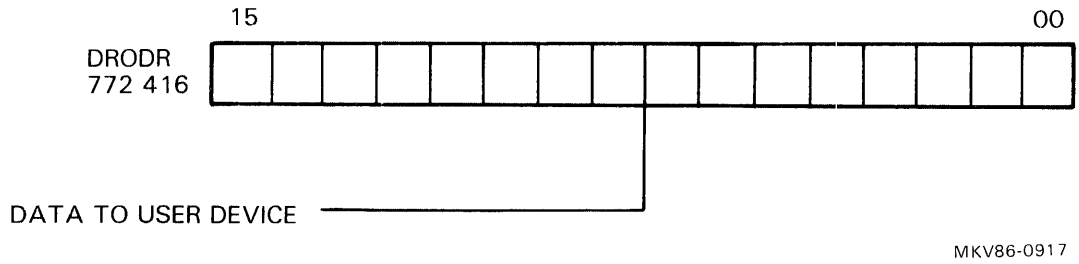

Figure 4-8 Output Data Register

The outputs of the register are driven directly onto DO 15 H through DO 00 H. The register is updated each time a DMA read cycle is concluded, as well as by explicit writes to the register.

The output data register may be written to using either byte or word instructions.

4.3 DATA DESKEW

Electrical signals do not all travel at a uniform speed. If you simultaneously write multiple bits (in the ODR) or multiple function bits (in the CSR), there is no guarantee that they will all appear simultaneously at the other end of the cables. Similarly, if the user device simultaneously sends multiple bits (to the IDR) or multiple status bits (to the CSR), there is no guarantee that they will all arrive at the DR11-W at the same time. Some bits may lead others by a few nanoseconds or tens of nanoseconds. This is called skewing (see Figure 4-9).

Normally, skew is of no consequence. The DMA engine contains hardware to compensate for skew during DMA transfers. If you are using an interrupt to signal data availability, the interrupt latency is many times the maximum possible skew. Even if you signal the presence of data in the ODR or IDR by means of the function or status bits, you will avoid skew problems so long as you write the data into the ODR prior to setting the function bit, which signals its availability. Conversely, the software should test for the data available flag in the status bits prior to reading the data from the IDR. The delay between the instructions is still more than adequate delay to deskew the data.

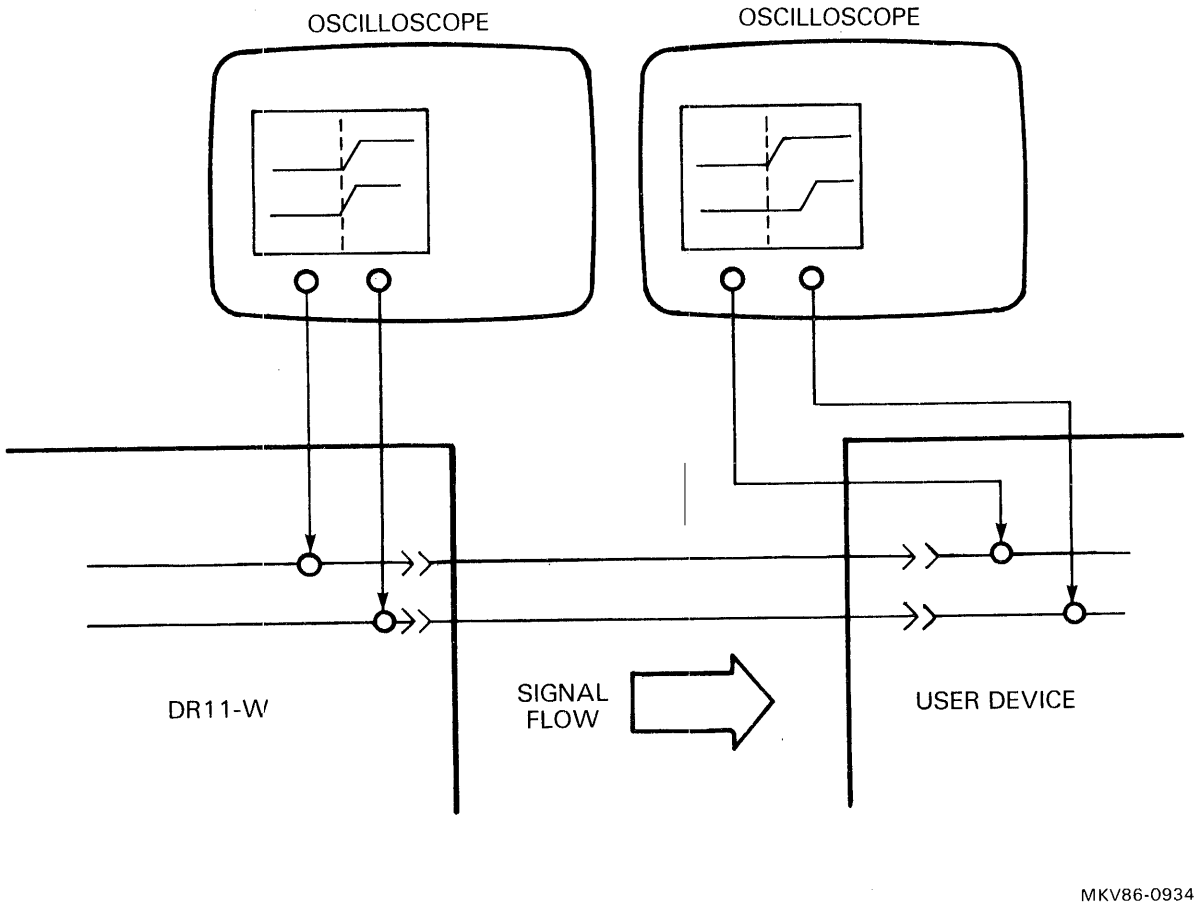

Figure 4-9 Skewing of Signals

Skew **can** be a problem if you use the same instruction to place the data and to signal its availability. Consider the following (bad) example. The user intends to move a byte to the user device via the ODR, and uses bit ⟨08⟩ of the ODR to signal that data is available to the user device:

```
CLR     @#ODR           ; Signal 'no data yet'
MOVB    DATA,    R0     ; Get the data from wherever
BIC     #177400, R0     ; Save just the data byte
BIS     #400,    R0     ; .OR. in the 'data-available'
                        ; flag
MOV     R0, @#ODR       ; Write that all to the ODR
                        ;
```

If bit ⟨08⟩ travels just slightly faster than any of the data bits, the user device may read the data just a few nanoseconds too soon. In doing so, it may miss a few of the data bits. Here, the programmer must consider and correct for the effects of skew. A technique that you can apply at the sending side:

```
        CLR     @#ODR           ; Signal 'no data yet'
        MOVB    DATA,    R0     ; Get the data from wherever
        BIC     #177400, R0     ; Save just the data byte
1$:     MOV     R0, @#ODR       ; Send the data to
                                ;   the user device
        BIS     #400,    R0     ; .OR. in the 'data
                                ;   available' flag
        MOV     R0, @#ODR       ; Write that all to the ODR
                                ;
```

Now, at 1$, we write the data two instructions before we write the data-available' flag. This allows plenty of time for all of the data bits to propogate to the user device.

A similar technique could be applied at the receiving end of the data. A non-deskewed example of a receiver:

```
1$:     MOV     @#IDR, R0       ; Get the IDR
        BIT     #400,  R0       ; Is the data 'available'?
        BEQ     1$              ; Branch back if not
        MOVB    R0, DATA        ; It's there --
                                ;   store the data
                                ;
```

That will not work unless the sender is deskewing. We read the entire IDR all at once. Bit ⟨08⟩ may have just arrived, but there is no guarantee that bits ⟨07:00⟩ have. A receiver with deskewing:

```
1$:     BIT     #400, @#IDR     ; Is data available yet ?
        BEQ     1$              ; Branch back if not
2$:     MOVB    @#IDR, DATA     ; Now get the data
                                ;
```

Now we again guarantee two instruction times for the data to all arrive. Note that only one end of the system (data sender or data receiver) needs to deskew, but if both ends deskew, the only harm done is slight loss of speed.

Remember that you may need to deskew either the ODR/IDR or the function/status bits.

4.4 USING PROGRAMMED I/O

The simplest way to use the DR11-W is via programmed I/O. Based on some synchronization method, data words are simply written by the processor to the output data register or read from the input data register.

The key to successful use of programmed I/O is the synchronization method. The DR11-W is optimized for DMA I/O, not programmed I/O (unlike some other DR11s such as the DR11-C), so no special hardware support for synchronization exists.

It is, however, quite easy to define a protocol using the function and status bits. The program fragment below is sending data to the user device using such a protocol. In this case, function 1 is used to indicate that the DR11-W has presented valid data in the ODR. Status A indicates that the device has accepted the data.

In this example, the DR11-W and the user device have a formal handshaking protocol. Each device takes only one step and then waits for an acknowledgement from the other end.

1. The DR11-W software asserts data in the output data register, followed by function 1.

2. The user device observes the assertion of function 1, strobes in the data, and returns status A.

3. The DR11-W software must hold the data and function 1 asserted until it sees status A come back from the user device. The DR11-W software then clears function 1.

4. Upon seeing function 1 clear, the user device must clear status A.

5. Upon seeing status A clear, the DR11-W software can assert new data, followed by function 1.

```
PIOOUT: BIT   #STATA,  @#DRCSR    ; Is user ready to begin?
        BNE   PIOOUT              ; Wait if not
                                  ;
1$:     MOV   DATA,    @#DRODR    ; Assert data to user
        BISB  #FUNCT1, @#DRCSR    ; Assert FUNCTION 1
                                  ;   (deskewing!)
2$:     BIT   #STATA,  @#DRCSR    ; Has user ACK'ed ?
        BEQ   2$                  ; Wait if not
3$:     BICB  #FUNCT1, @#DRCSR    ; Clear FUNCTION 1
4$:     BIT   #STATA,  @#DRCSR    ; Has user De-ACK'ed?
        BNE   4$                  ; Wait if not
        BR    1$ or <wherever>    ;
                                  ;
```

NOTE

The keen-eyed reader will note that bit sets and bit clears (read/modify/write instructions) are liberally sprinkled throughout this code. We can do this because we are using byte instructions directed at the low byte of the CSR. So long as the CSR rather than the EIR is selected, there is no danger here in using the R/M/W instructions.

This example might be more robust, but much less legible, if coded with the simulated BIS and BIC instructions shown earlier in this chapter.

4.5 USING INTERRUPT DRIVEN I/O

Here, the synchronization protocol is probably much simpler: every time an interrupt is generated, do something.

Often, an interrupt is used to catch the attention of the software, then one or more data words are exchanged quickly via programmed I/O.

4.6 STARTING A DMA TRANSFER

Once the software has determined that a large block of data is to move via DMA, it is an easy task to set up the DR11-W. The program fragment below illustrates such a setup.

```
DMAGO:  MOV    #-DMALEN, @#DRWC    ; Setup the word count
        MOV    #DMABUF,  @#DRBA    ; Setup the low bits
                                   ;   of the UNIBUS address
        MOV    #DMBFHI,  R0        ; Get BA17 and BA16
        BIS    #101,     R0        ; 'OR'-in 'IE' and 'GO'
        MOV    R0,       @#DRCSR   ; Load the CSR, clearing
                                   ;   all other bits
        BR     <wherever>          ; It's running . . .
                                   ;
```

Later, when the transfer ends and the completion interrupt is generated, a typical piece of service code might be:

```
DRINT:  TST    @#DRCSR             ; Did any error occur?
        BEQ    999$                ; We're done if not
        BIT    #020000, @#DRCSR    ; Is it ATTN 'error' ?
        BNE    DRATTN              ; Go service the ATTN
        MOV    @#DRCSR, R0         ; Hold the CSR for logging
        MOV    #100000, @#DRCSR    ; Switch to EIR, clear NXM + ATTN
        MOV    @#DREIR, R1         ; Hold EIR for logging
        MOV    @#DRIDR, R2         ; Hold last IDR for logging
        MOV    #010000, @#DRCSR    ; Set MAINTENANCE
        MOV    #000000, @#DRCSR    ; Clear MAINT, INITing DR
        BR     ERRLOG              ; Go log the error
                                   ;
999$:   RTI                        ; Dismiss the interrupt
                                   ;
```

4.7 DEVICE DRIVERS

A sample driver (the XADRIVER) is distributed with every VAX/VMS system. This driver is adequate to run the VAX/VMS on-line diagnostics, and serves to illustrate how the DR11-W may be programmed in a VAX/VMS system. You may use this driver as it stands, modify it to suit your needs, or simply use it as an example. The XADRIVER is documented in *Guide to Writing a Device Driver for VAX/VMS* (order code AA-Y511A-TE), part of the VAX/VMS documentation set. A source listing of this driver is also included in this manual in Appendix A.

No sample driver is available for any other VAX-11 or PDP-11 operating system.

CHAPTER 5
LINKS

5.1 INTRODUCTION

This chapter explains the use of the DR11-W module to create a high-speed, computer-to-computer link. The topics covered in this chapter are both the special hardware considerations and the special software considerations required for DR11-W links.

Be sure that you have read and understand the previous chapters before you begin this chapter.

NOTE
When operating the DR11-W in the link mode, switch 1 on switchpack E15 must be ON (closed). Refer to Section 2.4.2.

5.2 LINKING TWO DR11-Ws

A DR11-W can be cross-cabled with another DR11-W or a DR11-W compatible interface to create a high-speed, computer-to-computer link (see Figure 5-1). This link can move 16-bit words at rates approaching 500,000 words per second.

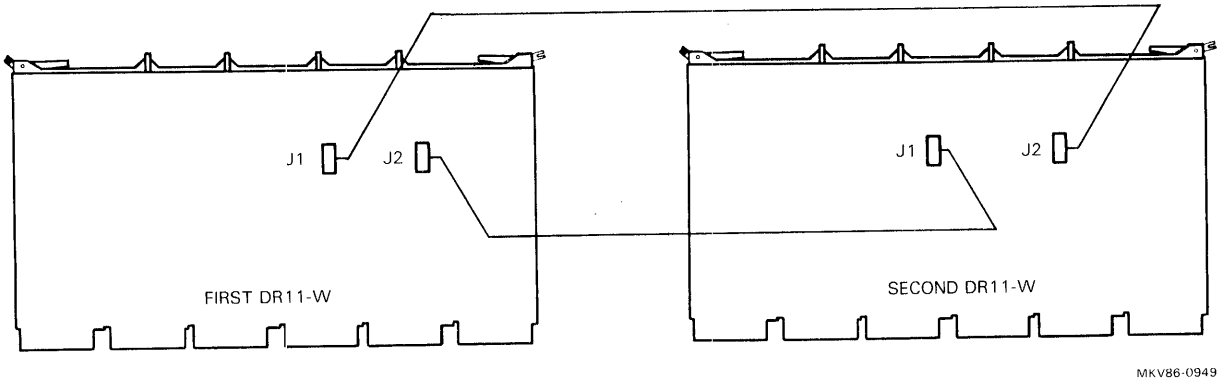

Figure 5-1 Configuration for Cross-Cabling Two DR11-Ws to Create a Link

For the rest of this chapter, the local DR11-W refers to the DR11-W plugged into your computer; the buddy refers to the interface plugged into the computer at the far end of the link.

Note that J1 on the first DR11-W is cabled to J2 on the second DR11-W and vice versa. The pin-out of the user connectors was specifically chosen so that this would connect the data outputs of each DR11-W to the data inputs of the other. All the necessary clocking, function, and status bits are also cross connected (see Figure 5-2). Remember that all of these connections are made in **each** direction.

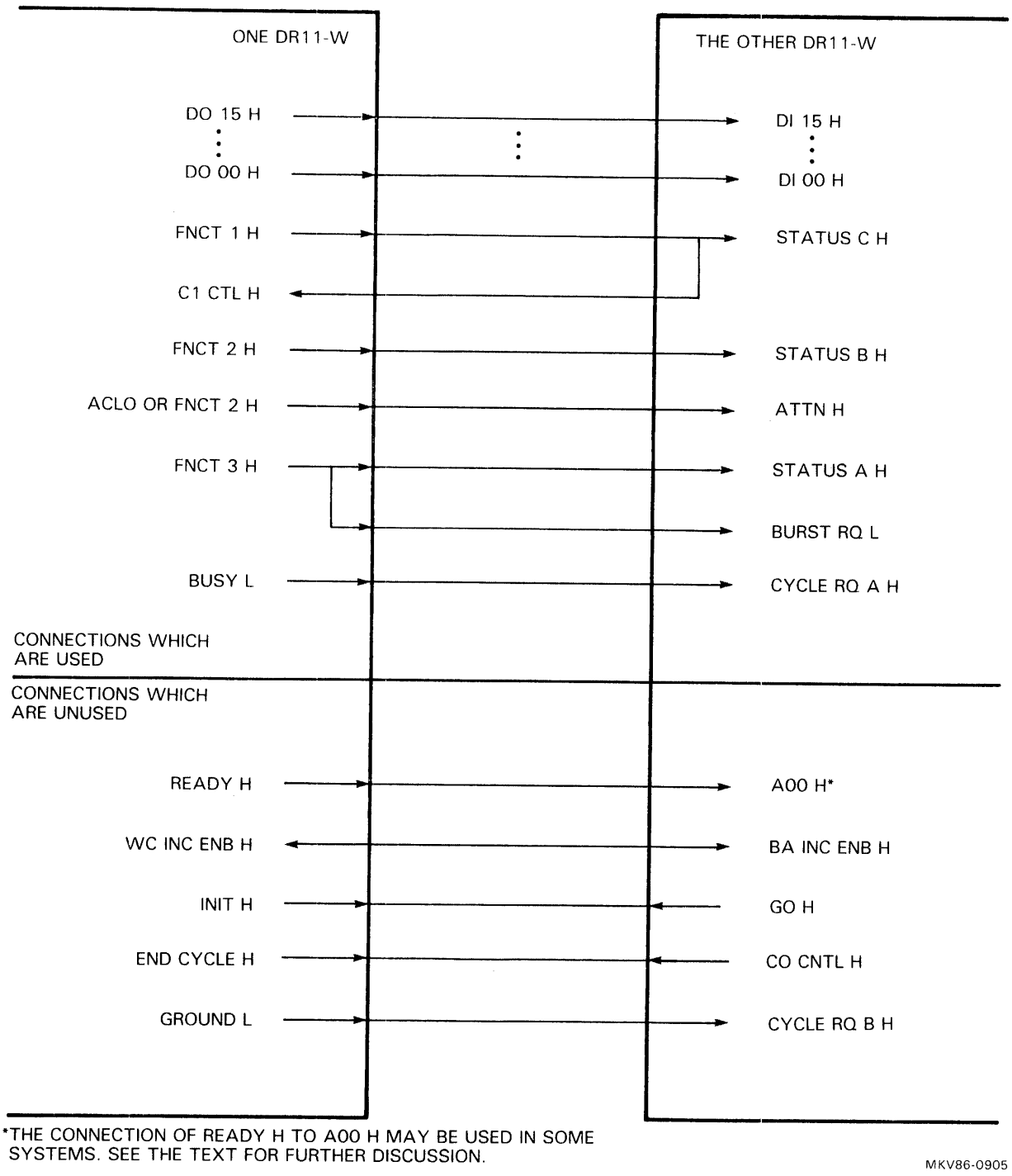

Figure 5-2 Connection Diagram for Half of a DR11-W Link

5.3 LINKING A DR11-W TO A DRV11-B OR DRV11-W

If the buddy is a Q-bus computer (such as a MicroPDP-11 or MicroVAX), a link can easily be constructed with the DR11-W installed in the UNIBUS computer and a DRV11-B or DRV11-W installed in the Q-bus computer.

Both the DRV11-B and DRV11-W are similar to the DR11-W, both in the user interface presented to the external world and in the programming interface presented to the software system.

The DRV11-W is a dual-height, Q-bus module implemented using programmed, logic-array technology, while the DRV11-B is an older, quad-height Q-bus module using standard TTL technology. Both program and cable identically with each other, and differ only in their size and level of integration.

The following differences and restrictions are important if you intend to build a DRV11-x to DR11-W link:

- Neither the DRV11-B nor the DRV11-W terminate all signals. This means that the transmission distance must be restricted. The maximum transmission distance is dependent upon the cabling used and must be determined by testing.

- The burst mode implemented by the DRV11-B and the DRV11-W is not compatible with the DR11-W. Therefore, the DR11-W must be programmed so as to keep the DRV11-B or DRV11-W in non-burst (one-cycle burst) mode. This is done by setting the function 3 bit in the DR11-W CSR. The DR11-W may be operated in any appropriate burst mode (as controlled by the function 3 bit in the DRV11-B or DRV11-W).

5.4 REDEFINITION OF SIGNALS AND CSR BITS

Cross coupling gives specific meaning to the function and status bits, which are undefined when the DR11-W is used to interface to a user device.

CSR Bit ⟨11⟩
Status A
(for links)

This bit indicates the contents of the buddy's function 3 bit (that is, whether the buddy is forcing us into burst-mode DMA). See the discussion under CSR bit ⟨03⟩.

CSR Bit ⟨10⟩
Status B
(for links)

This bit indicates the contents of the buddy's function 2 bit (that is, whether the buddy is asserting our attention input. See the discussion under CSR bit ⟨02⟩.

CSR Bit ⟨09⟩
Status C
(for links)

This bit indicates the contents of the buddy's function 1 bit (that is, whether the buddy is writing to memory on its UNIBUS or Q-bus). See the discussion under CSR bit ⟨01⟩.

CSR Bit ⟨03⟩
Function 3
(for links)

When configured as a link, the DR11-W uses function 3 as the signal for the buddy to operate using two- or n-cycle burst DMA. Function 3 must be held cleared for the duration of any transfer during which the buddy

should use burst-mode DMA. Function 3 must be held set for the duration of any transfer during which the buddy should use non-burst mode (single-cycle) DMA.

If the buddy is a DRV11-B or DRV11-W, this bit must be set in the DR11-W so that the DRV11-B or DRV11-W does not attempt to use burst mode. In the DRV11-B or DRV11-W, this bit may be set or cleared depending upon whether or not you want the DR11-W to operate in burst mode.

CSR Bit ⟨02⟩
Function 2
(for links)

When configured as a link, the DR11-W uses function 2 as the signal to interrupt the buddy. Function 2 may be asserted, then immediately removed, or held asserted until acknowledged somehow. If the buddy is a DR11-W, then attention will be latched at the buddy. Function 2 must be cleared prior to the time it is next used.

CSR Bit ⟨01⟩
Function 1
(for links)

When configured as a link, the DR11-W uses function 1 as the link direction control. Function 1 must be held asserted for the duration of any transfer where the local DR11-W should be writing to memory on its UNIBUS (in which case we hope that the buddy is reading from memory on its UNIBUS or Q-bus).

5.5 REDEFINITION OF BAR BIT ⟨00⟩

When the DR11-W is connected as a link and switch E105-4 is OFF, bus address register bit ⟨00⟩ has special meaning. This bit may be used to indicate the status of the buddy's ready bit.

Thus, when bar ⟨00⟩ is 1, the buddy is ready (that is, not armed for DMA). When bar ⟨00⟩ is a 0, the buddy is not ready (that is, is armed for DMA).

If Switch E105-4 is ON, this feature is disabled and bar ⟨00⟩ always reads as a 0.

5.6 SPECIAL INTERRUPT CONDITIONS

Two conditions can cause the local DR11-W to be interrupted by the buddy. These are:

- The buddy setting function 2.
- A power failure on the buddy's bus (UNIBUS or Q-bus).

Either of these conditions cause the local DR11-W to be interrupted. Assuming that the buddy holds function 2 set until acknowledged in some way, these two conditions can be distinguished by whether or not status B is set at the local DR11-W. If set, then the interrupt was explicitly requested by the buddy setting function 2. If status B is not set at the local DR11-W, then the attention interrupt must have been caused by a power failure at the buddy.

For systems where the buddy is connected to a UNIBUS or Q-bus, a power-failure interrupt is generated when the buddy powers down and when it powers up.

5.7 BASIC PROGRAMMING TECHNIQUES

The three basic programming techniques work for links as well as user devices.

5.7.1 Programmed I/O

Using function 1 (status A) and function 3 (status C), the two ends of the link can exchange data using programmed I/O techniques. In this case, function 2 (status B) is probably not a good choice since that bit is also wired to the attention interrupt logic on the other DR11-W.

Remember that either the transmitter or the receiver of the data must provide deskew of the data.

During programmed I/O, data may be exchanged on a full-duplex basis. That is, both ends of the link may be sending data simultaneously.

5.7.2 Interrupt Driven I/O

Since each end of the link can interrupt the other by asserting function 2, interrupt driven I/O is easy to use.

Interrupt driven I/O is also a full-duplex technique.

5.7.3 DMA I/O

Once the two ends of the link have established that a large transfer is to take place, then each end can be set up for direct memory access. The entire data block can then be sent without further action by either processor.

Since the DR11-W contains only one DMA engine, data can only be moving in one direction at a time during DMA I/O (that is, DMA I/O is a half-duplex technique). The link software must have some means of deciding on the data direction before starting the DMA. Both ends of the link must agree on the direction (since each end controls the direction of its own transfer). One DR11-W must read from memory while the other DR11-W writes to memory.

5.8 LINK ARBITRATION

If one end of the link is clearly the master, and the other end is clearly the slave, no arbitration for the DMA engine is required. The master end simply informs the slave (by the use of programmed I/O or interrupt-driven I/O) what transfer is next, and the slave must obey.

If, on the other hand, the two ends of the link are competing as equals, then there must be some method of resolving the situation where both ends desire use of the DMA engine. Three possible schemes (out of many) are listed below:

- Round-robin (alternate) priority
- First-come, first-served with deadlock resolution (tie-breaking)
- Message priority with deadlock resolution

Remember, all full-duplex use of the link must be ended prior to starting the DMA engine at either end of the link.

5.9 INITIATING-END VERSUS SENDING-END

In a symmetrical link, where both ends operate as equals, either end can initiate a transfer. This in no way implies the direction that the hardware will move data.

The initiating-end could ask that data be read from the buddy, or written to the buddy.

This chapter uses the phrase sending-end to indicate the end of the link that is reading data from memory. Receiving-end indicates the end of the link that is writing data into memory.

5.10 STARTING A DMA TRANSFER

Starting a DMA transfer on a link is not very different from starting a DMA transfer to a user device. As always, the word count and bus address registers are pointed to the memory buffer, and the go is bit set (see Figure 5-3).

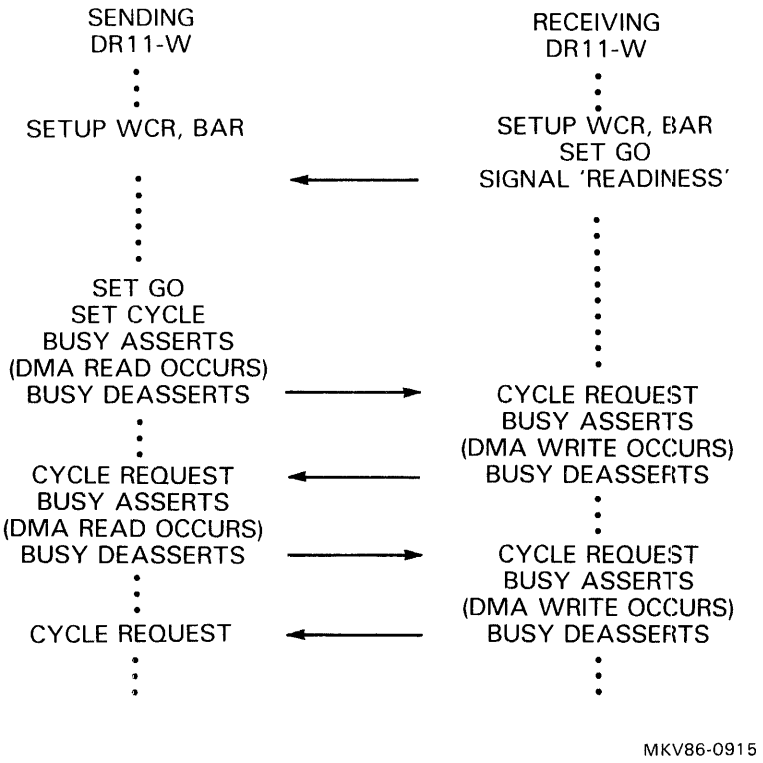

Figure 5-3 Link DMA Operation Flow Diagram

Now, however, there is no user device to issue cycle requests. Once the go bit has been set at both ends of the link, each DR11-W waits for the other to begin.

In order to get the transfer rolling, one additional step must be taken at the sending-end (that is, the end of the link which will be reading memory). Bit ⟨08⟩ in the CSR (the cycle bit) must be set. This causes the sending-end to perform one DMA cycle. As this cycle is performed, BUSY L asserts and then deasserts. When BUSY L deasserts (goes high), it acts as a CYCLE RQ A H at the receiving-end of the link. The receiving-end then performs a DMA cycle, asserting and releasing its busy signal. At the end of that cycle, the sending-end is triggered again.

This oscillation between sender and receiver continues until the word count expires at either or both ends. This oscillation between busy at one end and busy at the other end is shown again in Figure 5-4.

5.11 STARTING ORDER

In order for the receiving-end to not miss the first CYCLE RQ A H, the receiving-end's software must set go before the sending-end's software sets cycle. This means that some form of handshake (such as an attention interrupt) should be sent to the sending-end to indicate that it can now set cycle and start things

running. Alternatively, you can insist that the receiving-end take only a certain amount of time to set things up. Then, once this time has elapsed, the sending-end's software can set cycle, confident that the receiving-end is ready for the first word of data.

At the transmitting end, go and cycle may be set using the same instruction, or go may be set prior to cycle.

Once cycle has been set, neither end's software should write to its DR11-W CSR again until the DR11-W becomes ready. The cycle flip-flop could inadvertantly be set or cleared by the software, disrupting the correct operation of the link. Once go has been set, and the sending-end first sets cycle, there is no way to synchronize the operation of the software with the operation of the link hardware short of aborting the link transfer.

5.12 UNIBUS BANDWIDTH CONSUMPTION

The DR11-W does not contain any kind of hardware throttle regulating DMA cycles. Once the link is started, DMA cycles alternate between the two ends, as fast as the respective busses will allow (as previously shown in Figure 5-4). Typically, this means that 50% of the UNIBUS bandwidth will be consumed by each DR11-W while the link is running.

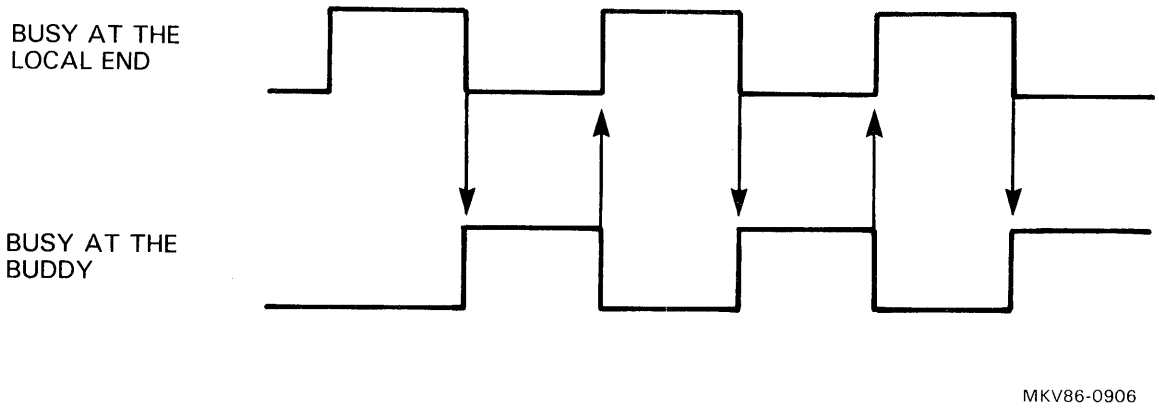

Figure 5-4 Busy Signal Relationship During Link DMA Operation

You can reduce the amount of bandwidth consumed by adding additional hardware to the link. This hardware should introduce delay between the deassertion of BUSY at one end of the link and the assertion of CYCLE RQ A H at the other end. This is shown in Figure 5-5. Usually, the same delay is added in each direction.

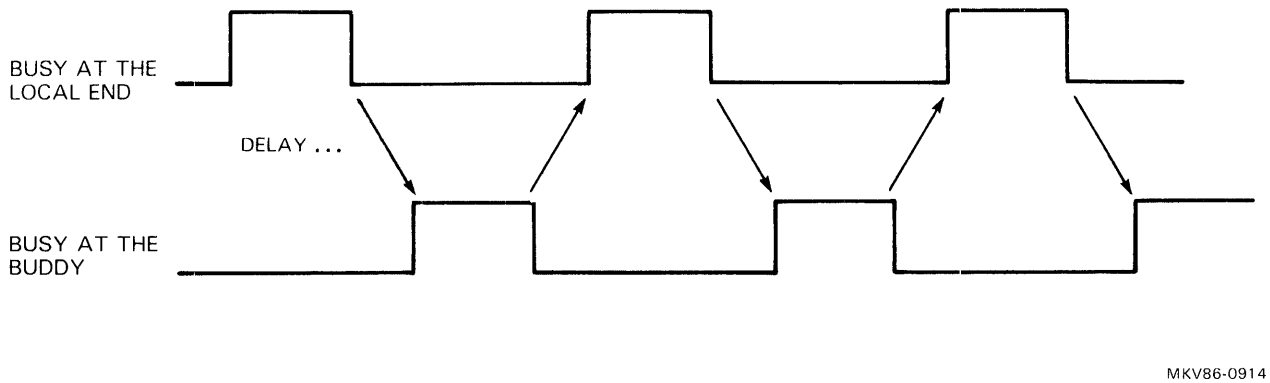

Figure 5-5 Busy Signal Relationship During Link Operation with Added Delays

5.13 BURST MODE

Burst mode is not generally useful during link operations, since, by definition, the ends of the link alternate DMA cycles and any given end of the link will be idle 50% of the time. Burst mode may be useful if:

- The UNIBUS would otherwise sit idle.
- There is a lot of contention for the UNIBUS, and the DR11-W data is of high priority.
- There is a large latency between DMA request and grant (as exists on many VAX-11 UNIBUS adapters).

Remember that it is the buddy that selects whether or not the local DR11-W will operate in burst mode.

CHAPTER 6
DMA OPERATION IN DETAIL

6.1 INTRODUCTION

This chapter explains in detail the direct memory access (DMA) operation of the DR11-W. This is done to aid you in understanding what the DR11-W can do for you and how you can optimize your design. This chapter is not a general discussion of DR11-W theory of operation. This manual does not cover that topic.

The drawings in this chapter are derived from the *DR11-W Maintenance Print Set* (MP00693) but are reorganized for improved clarity. In addition, some signals which are un-named in the maintenance print set have been given names in this manual.

Please note that the maintenance print set refers to the burst release logic as the burst data-late logic.

6.2 OVERVIEW

The DR11-W DMA mechanism can be visualized as having three layers (see Figure 6-1). Each lower level is dependent upon the levels above it.

The ready logic controls the cycle timing chain; the cycle timing chain controls the UNIBUS timing chain.

6.3 READY LOGIC

At the topmost level is the ready flip-flop (see Figure 6-2). When ready is set, the DR11-W is idle and ready to be set up by the software. When ready is clear, the DR11-W has been started by the software and the rest of the DMA logic is enabled.

The ready flip-flop is set whenever:

- The DR11-W is initialized.
- The word count register overflows to 000 000.
- An error occurs during DMA operation.
- ATTENTION H is asserted during DMA operation.

The ready flip-flop is cleared only by writing a 1 to the go bit.

Both ready and go are available to the user device. The go bit is first passed through a one-shot so that it always has a standard pulse width, regardless of the UNIBUS speed of the processor writing the bit.

6.4 WORD COUNT OVERFLOW

When the DR11-W word count register reaches 000 000, D3 WCOF L returns to the deasserted (that is, logic-high) state and the ready bit is set. No further DMA cycles are started, and CYCLE RQ x H cannot set the cycle or busy flip-flops. This is the normal way that the transfer of a data block completes.

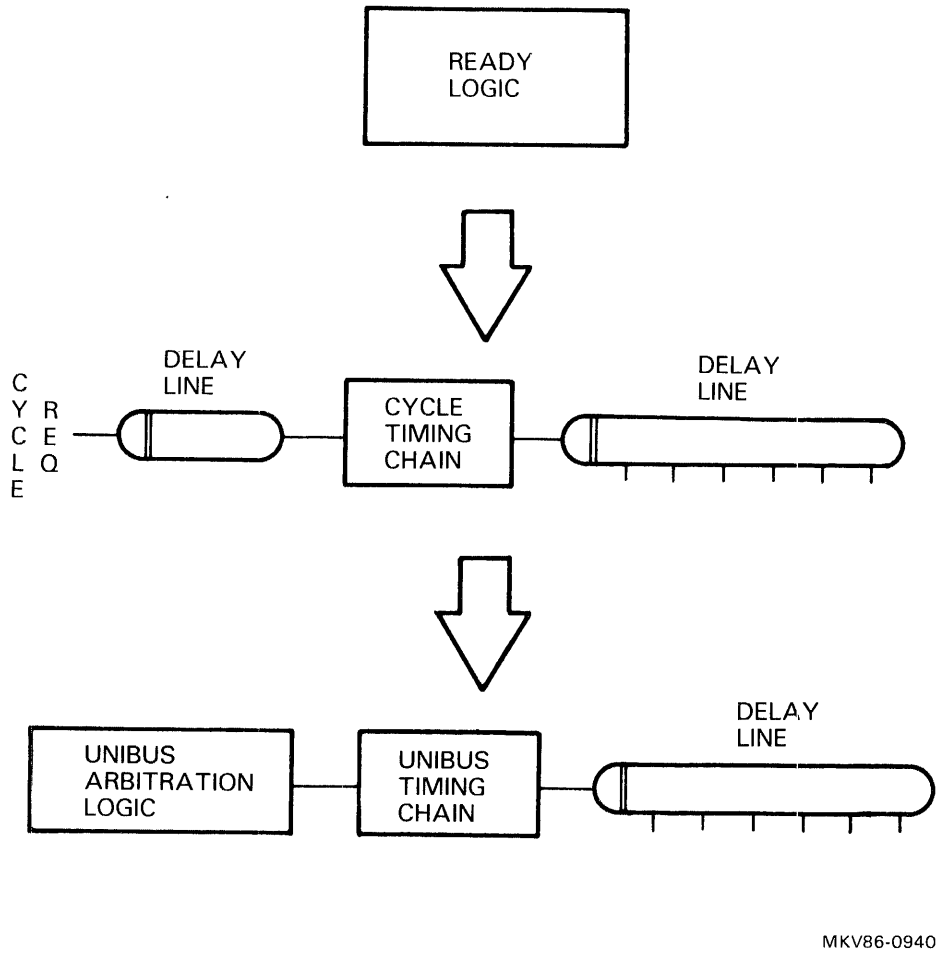

Figure 6-1 DMA Cycle Logic

6.5 CYCLE AND BUSY LOGIC

The clearing of ready (by the software asserting the go bit) enables the DMA logic. Ready is presented as the data input to both the cycle and busy flip-flops (see Figure 6-3).

Both CYCLE RQ A H and CYCLE RQ B H are first ORed together, then passed through the maintenance multiplexer (this logic is not shown in the figure). Then the combined cycle request is passed through a 125 nanosecond delay line. This delay line provides part of the delay required to deskew all of the user inputs to the DR11-W. Once the combined cycle request emerges from the delay line, it causes both the cycle and busy flip-flops to set.

The busy output of the DR11-W asserts at this time, indicating to the user device that a DMA cycle has begun.

The cycle flip-flop may also be direct-set or direct-cleared by the software writing to the CSR. During link operation, the software must direct-set the cycle flip-flop to start the first DMA in a block transfer. Generally, there is never any reason for the software to clear the cycle flip-flop, and doing so might cause improper operation of this circuitry. It is for this reason that the software must never write to the CSR once the go bit has been set (or the first DMA cycle of a link transfer has begun).

The cycle flip-flop may be read by the software, however, its operation is completely asynchronous with respect to the software so the results of reading the flip-flop usually are not meaningful.

6.6 MULTICYCLE REQUEST

Refer to Figure 6-3. If the user device requests another DMA cycle while busy is still asserted from the previous cycle, the multicycle flip-flop is set. This error is analagous to the data-late errors detected by classical disk and tape controllers. The user device is demanding words faster than the DR11-W can obtain them from memory, or the user device is presenting words faster than the DR11-W can write them to memory.

Like all of the error conditions described later, multicycle error stops the DR11-W at the end of the current DMA cycle and the DR11-W becomes ready.

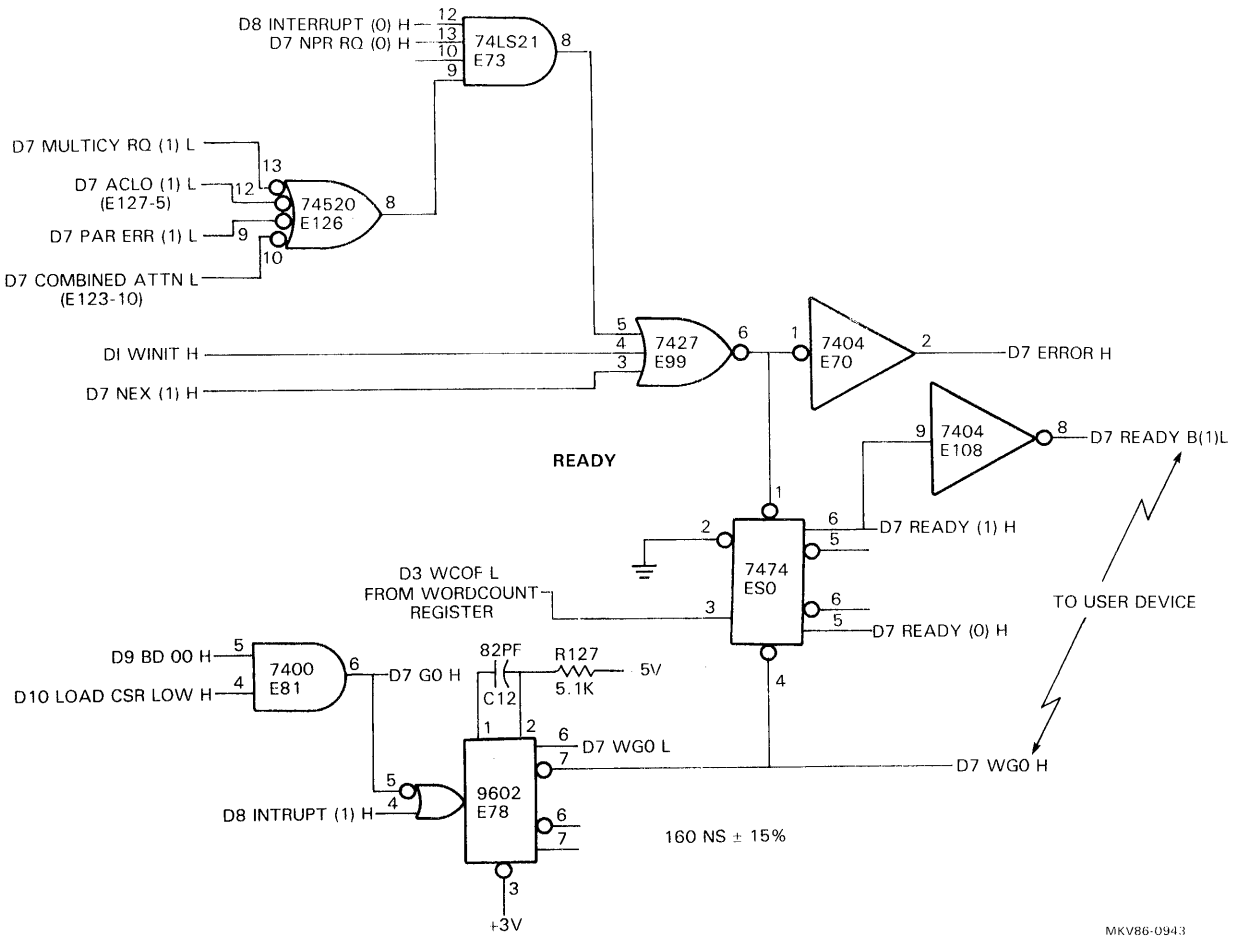

Figure 6-2 Ready Logic

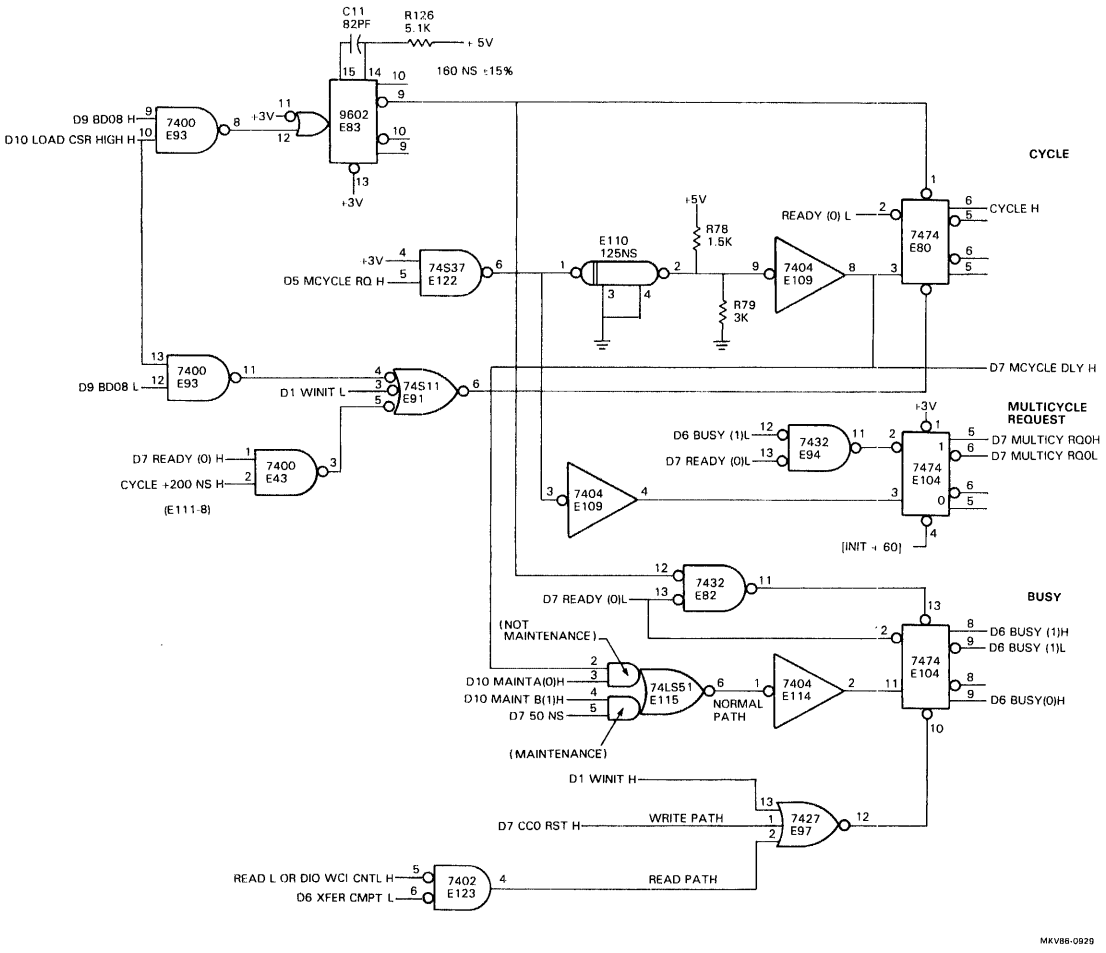

Figure 6-3 Cycle Request and Busy Logic Diagram

6.7 THE CYCLE DELAY LINE

Once the cycle flip-flop sets, a rising edge begins propagating down the cycle delay line (E100) (see Figure 6-4). Five outputs are generated:

- D7 GATED CYCLE L – This output is used to clear the burst release timed-out flip-flop.
- D7 50 nS – This output is used to set the NPR request flip-flop, causing the DR11-W to request use of the UNIBUS for a non-processor request (NPR) operation. It is also used to retrigger the burst release timer.
- D7 WC CLK H – This output is used to increment the word count register.
- D7 200 ns – This output sets the cycle inhibit flip-flop and is wrapped around to clear the cycle flip-flop. This means that the pulse traveling through the delay line is approximately 200 nanoseconds long.
- D7 CCO RST H – If the user has requested a write to memory, this signal clears busy.

6.8 CYCLE INHIBIT FLIP-FLOP

Once the cycle flip-flop has cleared, it could be set again by the user device reasserting CYCLE RQ x H. Cycle inhibit locks-out any new pulses from traveling down the cycle delay line.

This eliminates any errors which might arise if the user immediately sets cycle again while busy is still set (which, of course, causes a multicycle error). This also handles the case of write operations, where busy clears before the DR11-W is actually finished. In this case, the user device can present the new data and assert CYCLE RQ x H while the cycle inhibit flip-flop holds off the start of the new cycle until the DR11-W is ready to begin.

If the DR11-W is linked to a DRV11-B or DRV11-W, cycle inhibit is also used to provide a busy signal which has the particular timing that the DRV11-B or DRV11-W needs.

6.9 THE NPR REQUEST FLIP-FLOP

The NPR request flip-flop is set 50 nanoseconds after the cycle pulse begins traveling down the cycle delay line.

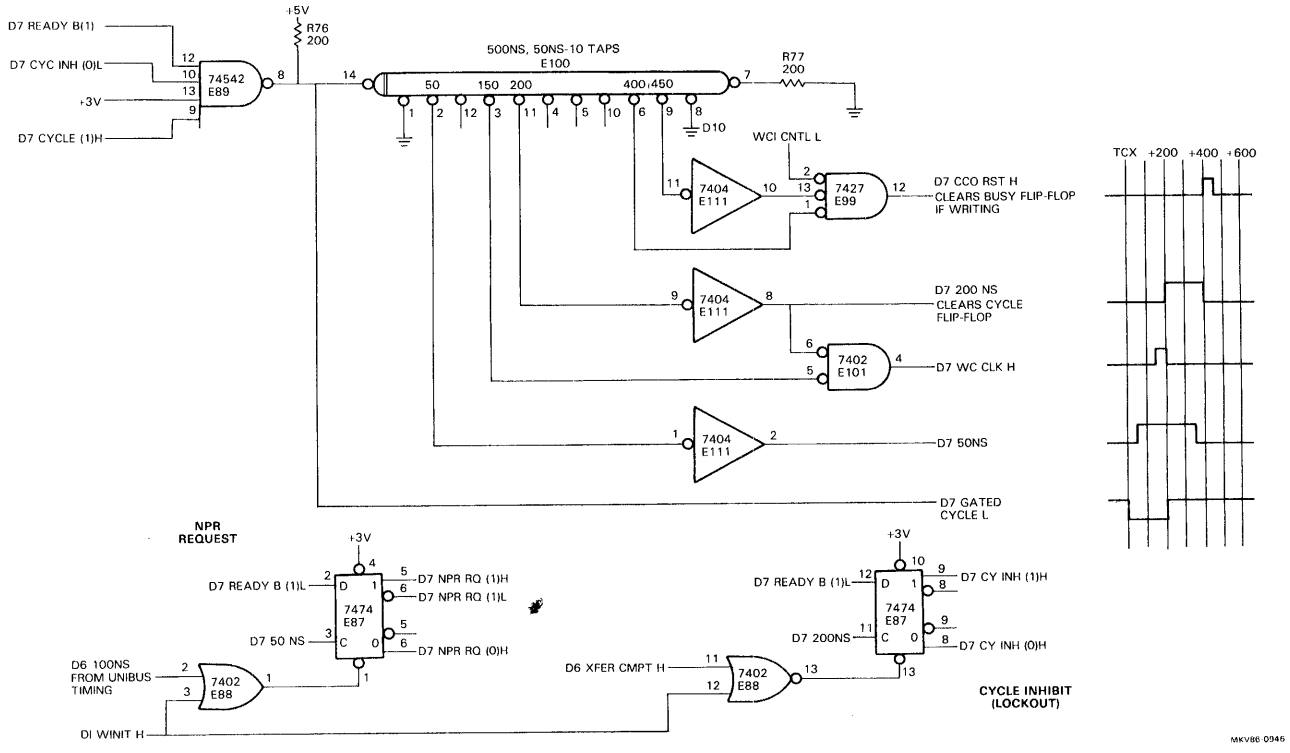

Figure 6-4 Cycle Delay Line Logic

This flip-flop notifies the UNIBUS NPR arbitrator (located at the front-end of the UNIBUS in the processor or UNIBUS adapter) that the DR11-W needs the UNIBUS. This flip-flop also provides the signal which latches the user inputs:

- DI00 to DI15
- BURST RQ L
- STATUS A H to STATUS C H
- C0 CNTL H and C1 CNTL H
- A00 H
- BA INC ENB H
- WC INC ENB H

The NPR request flip-flop is cleared 100 nanoseconds after the UNIBUS data cycle begins.

When reading the circuit schematics, take care not to confuse the timing signals coming from the cycle delay line (for example, D7 200 NS H) with the signals from the UNIBUS delay line (for example, D6 200 NS H). These are separate signals.

6.10 THE NPR ARBITRATOR

See Figure 6-5. The NPR arbitrator works with the UNIBUS arbitrator (at the front end of the UNIBUS) and the rest of the UNIBUS devices to establish the priority of the DR11-W.

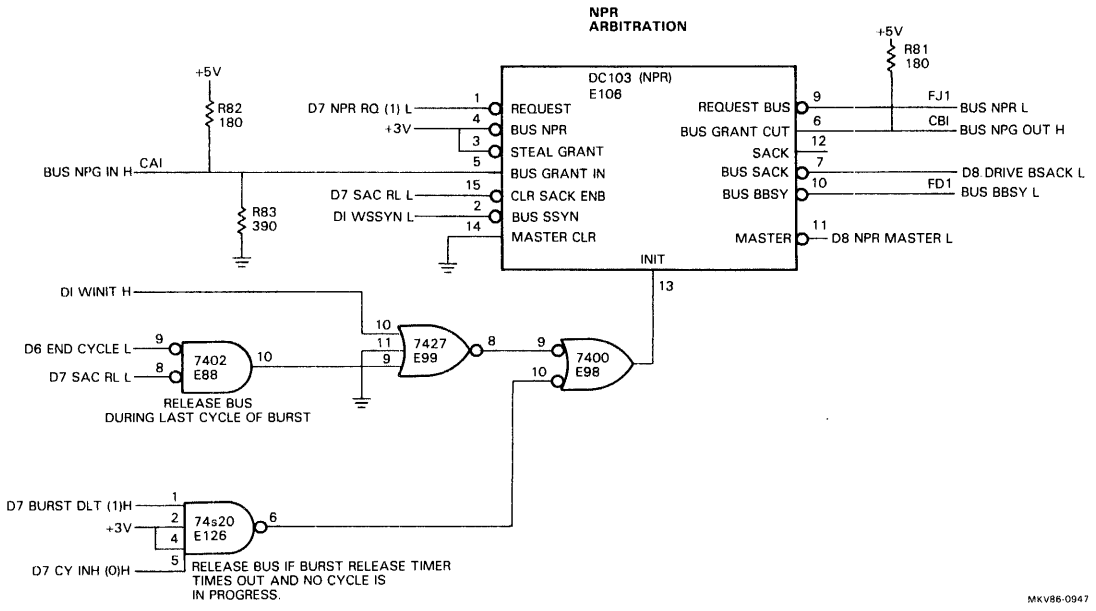

Figure 6-5 NPR Arbitration Logic

When the NPR request flip-flop sets, request is asserted at the DC013 chip. This causes the chip to assert BUS NPR L, the UNIBUS signal that indicates that the DR11-W would like to use the UNIBUS to perform an NPR (DMA) cycle.

The BUS NPR L signal is monitored by the UNIBUS arbitrator at the front end of the UNIBUS. When the UNIBUS arbitrator is willing to allow an NPR cycle, it issues BUS NPG H (non-processor grant).

NPG propagates back along the UNIBUS towards the DR11-W. Each device, in turn, receives the NPG and considers whether it needs to use the UNIBUS. If so, the device blocks the grant. If not, the device retransmits the grant towards the back of the UNIBUS. Eventually, it reaches the DR11-W.

The DC013 chip receives the NPG and asserts the signal D8 NPR MASTER L. This signals that the DR11-W has become the master of the UNIBUS and may perform one or more UNIBUS data cycles.

Any of three conditions causes the DC013 chip to release the UNIBUS:

- The DR11-W is initialized.
- The end cycle pulse occurs and the DR11-W has reached the end of a burst (of however many words).
- The burst release timer expires while the DR11-W is not presently in the middle of a UNIBUS data cycle.

6.11 UNIBUS TIMING LOGIC

See Figures 6-6, 6-7, and 6-8. Once the DR11-W has become the master of the UNIBUS, this logic produces the timing signals required to transfer one word of data.

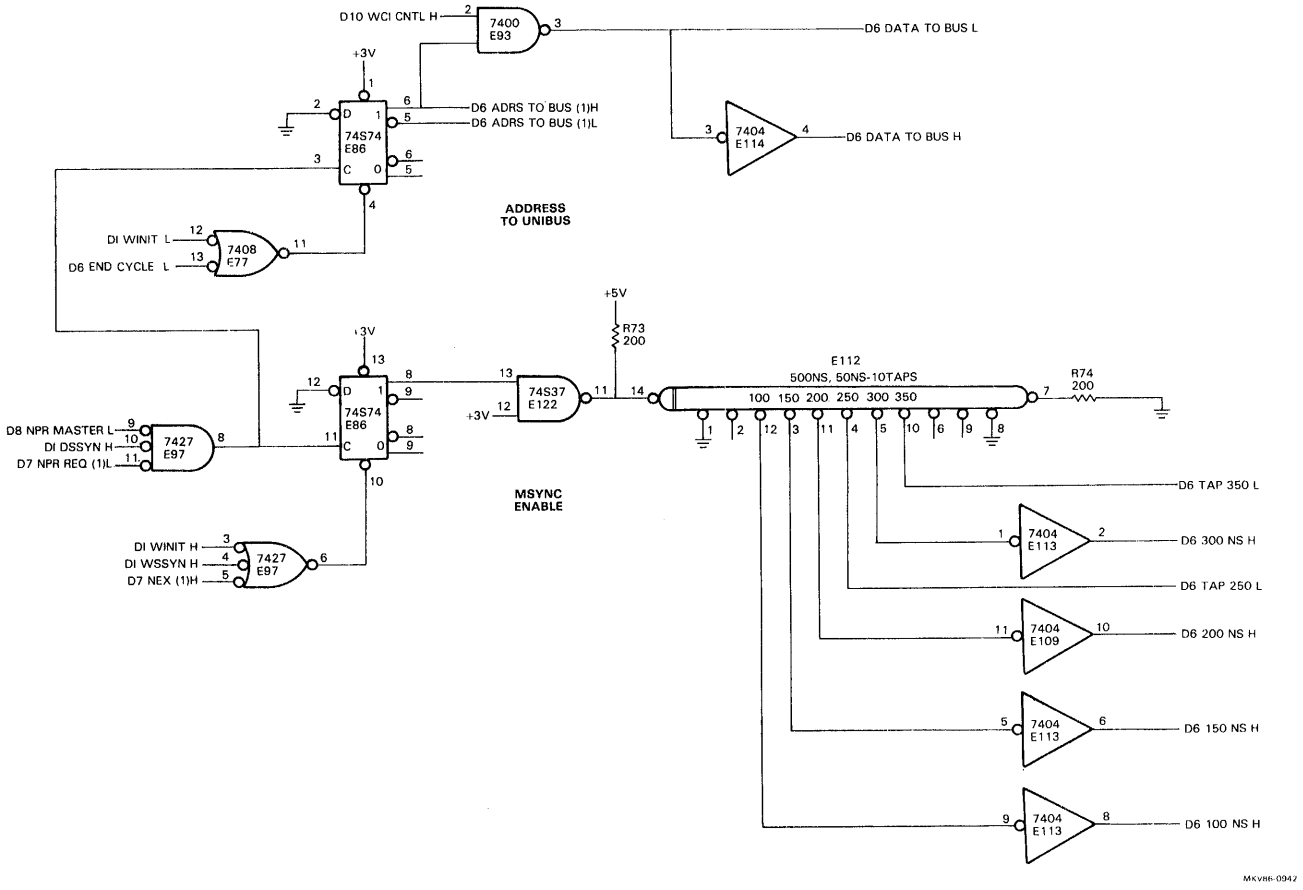

Figure 6-6 UNIBUS Timing Logic

The UNIBUS operation starts as both the address-to-UNIBUS and mastersync enable flip-flops set. This occurs if:

- The DR11-W is the UNIBUS master.
- The NPR request flip-flop is still set.
- There is no slave sync (BUS SSYN L) left over on the UNIBUS from a previous transfer.

As soon as address-to-UNIBUS sets, the DR11-W drive the contents of the bus address register (and the CSR's XBA17 and XBA16 and the user's A00H) onto the UNIBUS address lines. UNIBUS protocol then requires that the master wait 150 nanoseconds (minimum) for the address to reach all points on the UNIBUS and be decoded.

If the DR11-W is writing data to memory, D6 DATA TO BUS L is also asserted, causing the DR11-W to gate the contents of the input data register onto the UNIBUS data lines.

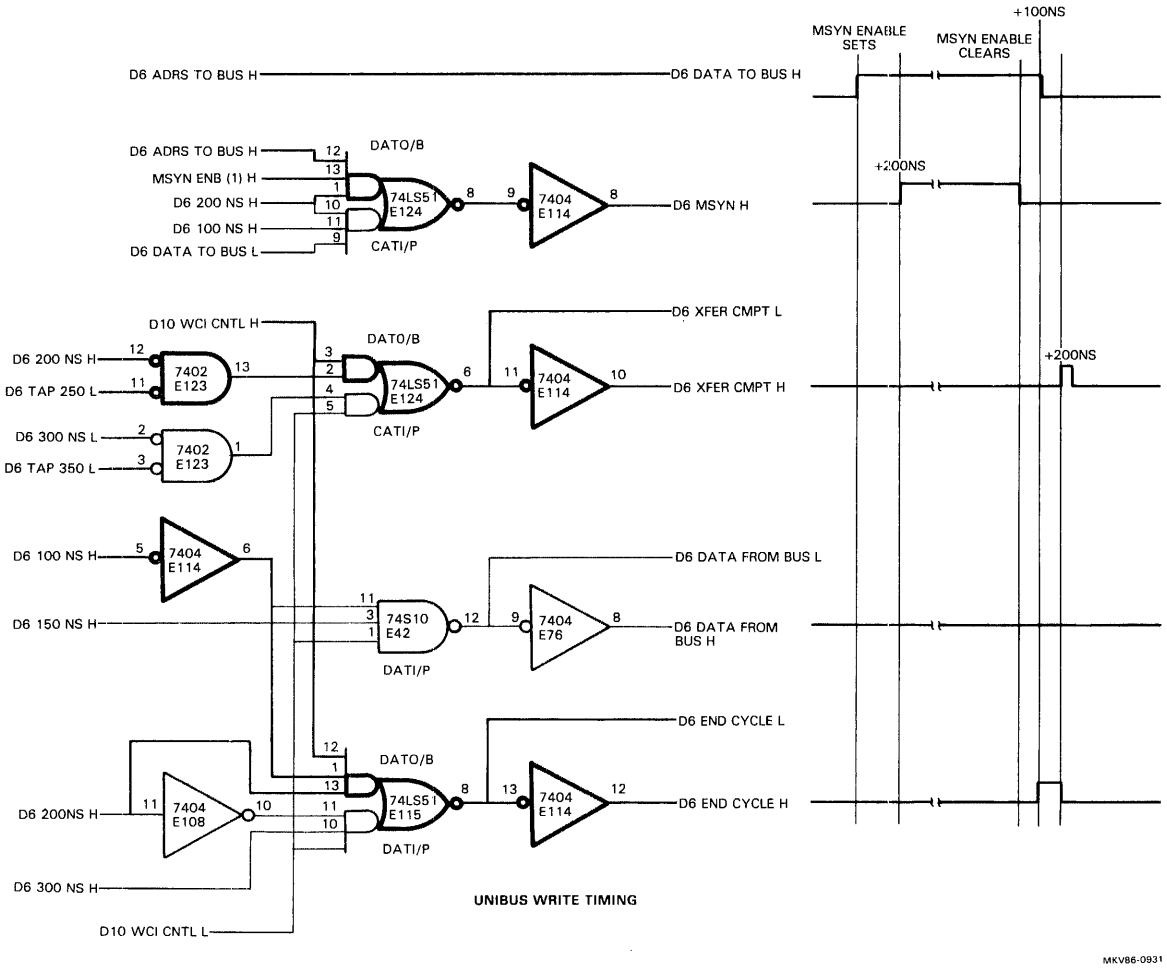

Figure 6-7 UNIBUS Write Logic and Timing Diagram

Meanwhile, the rising edge produced by the mastersync enable flip-flop is propagating down the UNIBUS timing delay line. When the rising edge reaches the 200 nanosecond tap, BUS MSYN L is asserted. This signal informs all other devices on the UNIBUS that the address on the UNIBUS is now valid. If this is a write cycle, BUS MSYN L also indicates that the data on the UNIBUS is valid. As each device receives BUS MSYN L, the address received by the device is compared against the set of addresses that that device is responsible for. If there is a match, then that device is referred to as the selected slave.

Once the DR11-W asserts BUS MSYN L, the DR11-W begins waiting. The rising edge of mastersync enable may or may not reach the end of the UNIBUS timing delay line before the selected device on the UNIBUS responds.

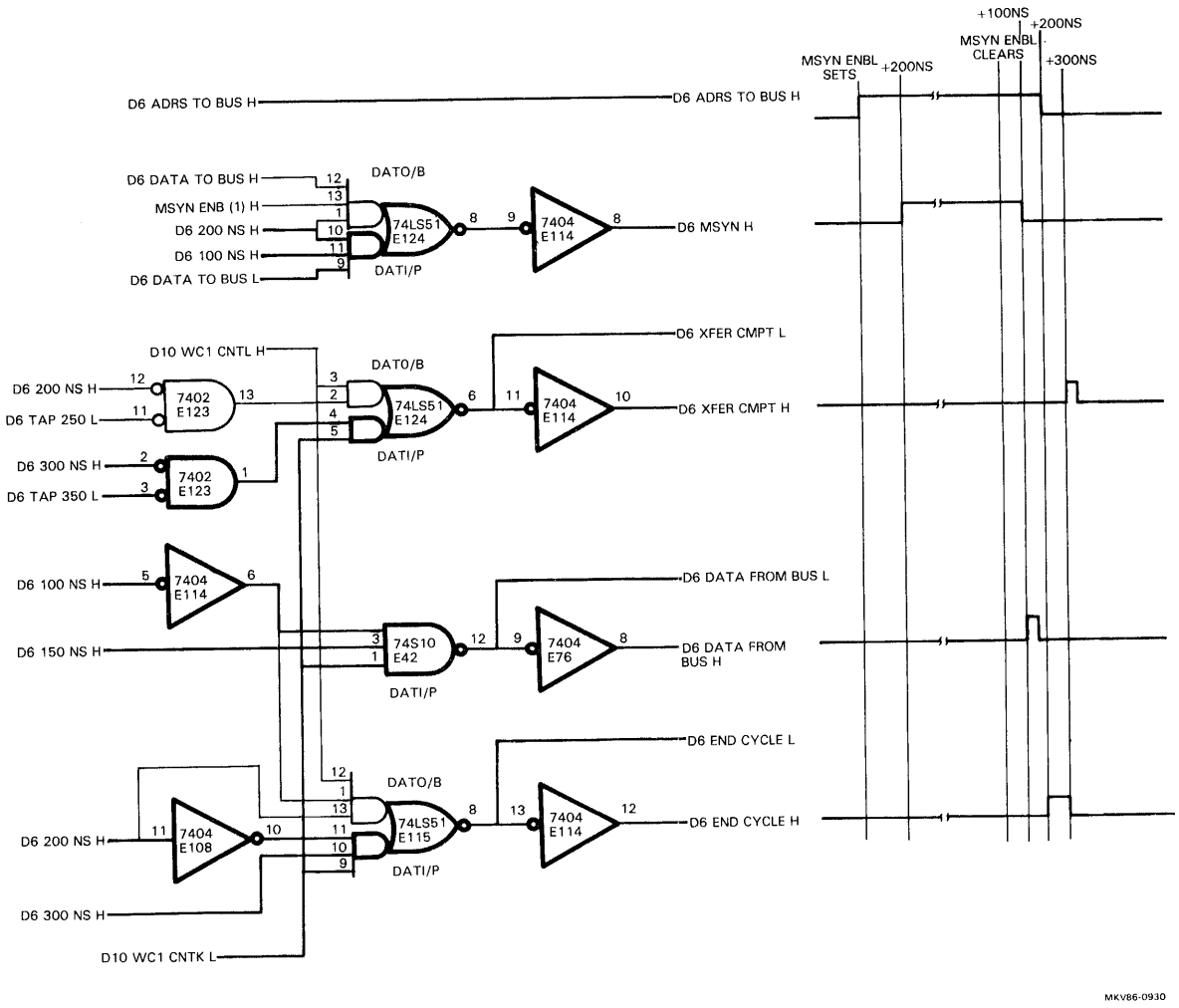

Figure 6-8 UNIBUS Read Logic and Timing Diagram

6.12 SLAVE SYNC FOR WRITE OPERATIONS

If the address issued by the DR11-W is recognized by another device on the UNIBUS, then that device latches in the data that the DR11-W has placed on the UNIBUS. Once the data is securely latched, the selected device asserts BUS SSYN L (slave sync).

When the DR11-W receives BUS SSYN L, the mastersync enable flip-flop is cleared. This causes a falling edge to begin propagating down the UNIBUS timing delay line.

The DR11-W immediately deasserts BUS MSYN L. This informs all devices on the UNIBUS that the address and data lines no longer contain valid information. However, if the DR11-W were to simultaneously remove the address from the UNIBUS address lines, some devices on the UNIBUS might see the address change before they received the negation of BUS MSYN L. Using this changed address, they might falsely be selected. To avoid this, the UNIBUS specification requires that the address lines be held valid for 75 nanoseconds (minimum) after the negation of BUS MSYN L. Therefore, the DR11-W does not clear address-to-UNIBUS until 100 nanoseconds after the deassertion of BUS MSYN L.

D6 END CYCLE H (at time T+100 nanoseconds) clears address-to-UNIBUS. The DR11-W now removes both the address and the data from the UNIBUS. D6 END CYCLE H is available to the user device.

The clearing of address-to-UNIBUS is also used to increment the bus address register (thus preparing it for the next transfer).

Finally, D6 XFER CMPT H (that is, transfer complete) is produced from T+200 to T+250 nanoseconds. This signal wraps around and clears the cycle inhibit flip-flop.

6.13 SLAVE SYNC FOR READ OPERATIONS

Read operations are similar to write operations. One additional step is required.

If the address issued by the DR11-W is recognized by another device on the UNIBUS, then that device reads the requested data word and places it on the UNIBUS data lines. At the same time, the selected device asserts BUS SSYN L (slave sync).

Once again, when the DR11-W receives BUS SSYN L, the mastersync enable flip-flop is cleared, causing a falling edge to begin propagating down the UNIBUS timing delay line.

Because the slave device asserted BUS SSYN L at the same time as it asserted the UNIBUS data, the DR11-W must allow some time for all of the data to reach the DR11-W (that is, the DR11-W must deskew the UNIBUS data). E42 is responsible for this delay. D6 DATA FROM BUS H is produced 100 nanoseconds after mastersync enable clears. This pulse stores the data in the output data register.

At the same time (T+100 nanoseconds), BUS MSYN L is deasserted by the DR11-W. The slave device is now free.

D6 END CYCLE H is produced from T+200 to T+300 nanoseconds.

Finally, D6 XFER CMPT H (that is, transfer complete) is produced from T+300 to T+350 nanoseconds. This signal wraps around and clears the busy and cycle inhibit flip-flops.

Note that relative to the write cycle, all of the operations are delayed by the 100 nanoseconds spent deskewing the received data.

6.14 BURST OPERATION

Two additional flip-flops and two one-shots are used to implement the various burst modes (see Figure 6-9).

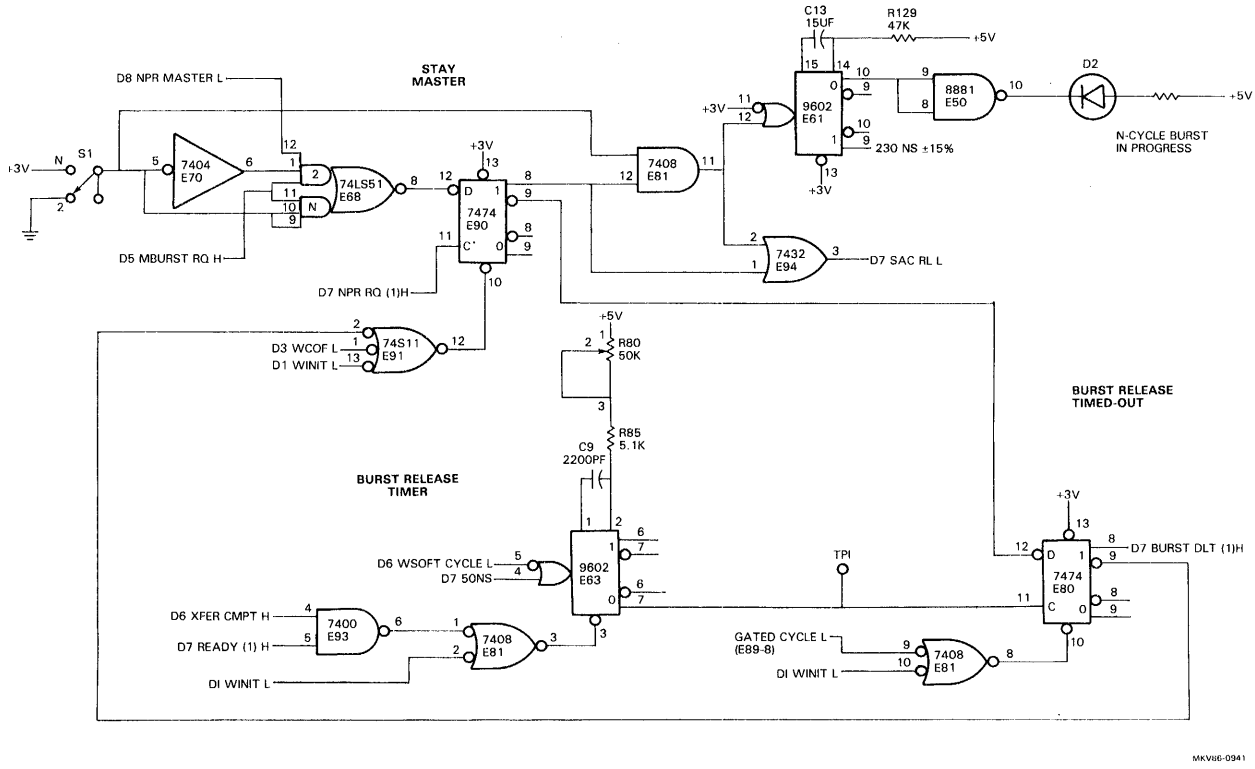

Figure 6-9 Burst Logic

The DC013 NPR arbitrator chip stays master of the UNIBUS until instructed to release it. This instruction normally comes in the form of D7 SAC RL L (that is, sack release L).

D7 SAC RL L is asserted if:

- The user device has not asserted BURST RQ L.
- The user device has asserted BURST RQ L, the two-cycle/n-cycle toggle switch is in the two-cycle position, an NPR cycle is requested, and the DR11-W is already UNIBUS master.

If neither of these conditions are met, the DR11-W stays UNIBUS master (n-cycle mode).

6.15 N-CYCLE LED

N-cycle mode hogs the UNIBUS, and requires you to design a system which can withstand this hogging. Just to alert you that you are using n-cycle mode, the DR11-W module contains a light-emitting diode (LED) which illuminates whenever n-cycle mode is in use.

If the DR11-W is operating in n-cycle mode, the E61 one-shot is fired each time the DR11-W stays master of the UNIBUS. This one-shot stretches the signal to 230 milliseconds (minimum) so that you can see it.

6.16 BURST RELEASE TIMER

A failure in the user device may cause the user device to stop issuing CYCLE RQ x H. Ordinarily, this would not cause a system problem. However, if the DR11-W is in the midst of a two-cycle or n-cycle burst, the DR11-W would continue to be master of the UNIBUS, possibly forever. This would certainly cause the rest of the system to fail.

The burst release timer (E83) eliminates this possibility. As long as DMA cycles are occuring at a rate faster than the time-out period of the one-shot, the one-shot stays set and the burst is allowed to continue (the normal two-cycle logic can still end the burst after two cycles).

If, however, the DMA cycles pause or stop, the one-shot may time out. In this case, the burst release timed-out flip-flop is set and the DR11-W releases the UNIBUS. No error has occurred – the DR11-W has simply vacated the UNIBUS since it had no useful work to perform. This allows other devices in the system to use the UNIBUS.

When the user device makes the next cycle request, the DR11-W once again arbitrates to use the UNIBUS.

The burst release timer is adjustable. Setting the burst release timer is described in Chapter 2.

6.17 ERRORS

The DR11-W detects five error conditions which can also terminate a data block:

- Multicycle error
- AC LO error
- Parity error
- Non-existent memory error
- Attention error

Your particular application may or may not define attention as an error, however, it does set the error bit and terminate the data block (see Figure 6-10).

Multicycle error has already been discussed.

6.17.1 AC LO Error

If an ac (main) power failure is detected by any of the devices connected to the UNIBUS, that device asserts the signal BUS AC LO L. This signals to all other devices on the UNIBUS that power is about to fail, and that action should be taken to save the state of the system.

The action taken by the DR11-W is to "get out of the way". It does this by setting the AC LO flip-flop, completing any DMA cycle already in progress, and then terminating the data block by setting ready.

The fact that a power failure has occurred is also signaled to the user device via the user AC LO flip-flop. This flip-flop can be set anytime (regardless of whether or not the DR11-W is ready).

Both flip-flops can be cleared by initializing the DR11-W or by setting the go bit.

6.17.2 Parity Error

Certain UNIBUS slaves (for example, memories) contain logic to generate and check parity upon the data that they have stored. Upon reading that data, they can use the UNIBUS line BUS PB L to signal whether or not a parity error was detected. Note that actual data traveling on the UNIBUS is not parity-checked.

If the DR11-W is reading data from memory, and the memory asserts BUS PB L (saying that the accessed word has bad parity), then the parity flip-flop is set. The current DMA cycle is completed and the DR11-W becomes ready. The bus address register points to the word beyond the word containing the parity error.

6.17.3 Non-Existent Memory Error

The UNIBUS is an asynchronous bus: although the order of events in a UNIBUS data cycle is fixed, there is no fixed schedule with which the events must occur. When the DR11-W issues a UNIBUS address accompanied by BUS MSYN L, the DR11-W begins waiting for some slave to become selected and reply with BUS SSYN L. If the DR11-W were to issue an address that did not correspond to any real slave device, then, in this scenario, the DR11-W would wait forever for BUS SSYN L.

Fortunately, this is not the way it works. The system designer establishes an absolute maximum waiting time for BUS SSYN L. This is generally referred to as the bus time-out value although it is more properly known as the slave sync time-out value.

When the DR11-W issues BUS MSYN L, it also fires the E78 one-shot. If this one-shot times-out before D6 END CYCLE L occurs, then the DR11-W concludes that there is no slave device at that address and the non-existent memory (NEX) flip-flop is set. The current DMA cycle is forced to completion and the DR11-W becomes ready. The bus address register points to the word beyond the word which did not respond.

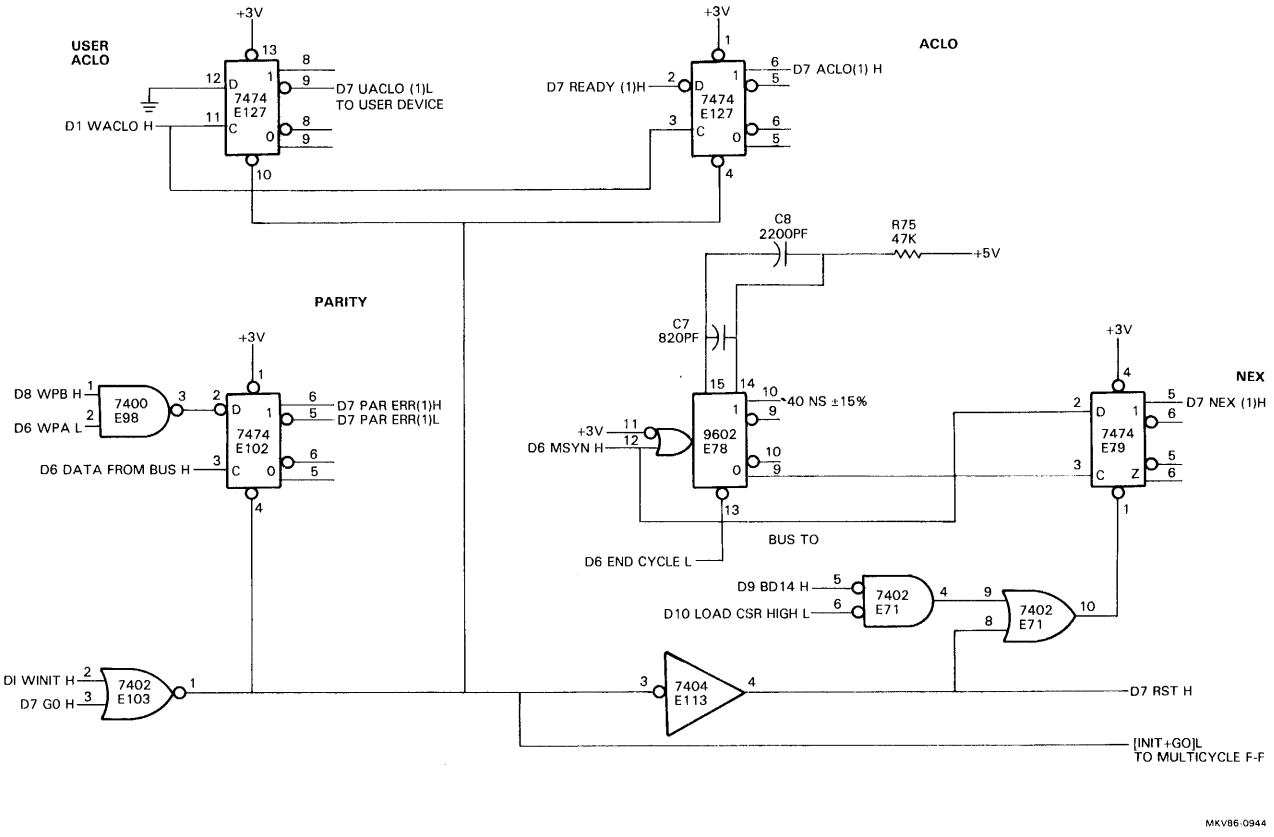

Figure 6-10 Error Logic

The value for the NEX timer is generally set at 12 microseconds. This is adequate for most systems. However, if the system contains multiported memory or other devices with potentially slow SSYN response times, this timer may need to be increased. An additional capacitor on the DR11-W module allows for longer settings. If this is insufficient, you can add additional capacitance. Setting this timer is described in Chapter 2.

6.17.4 Attention

Your application may or may not consider ATTENTION H to be an error indication, however, the DR11-W hardware treats it the same as the other error conditions so it is described as though it was an error (see Figure 6-11).

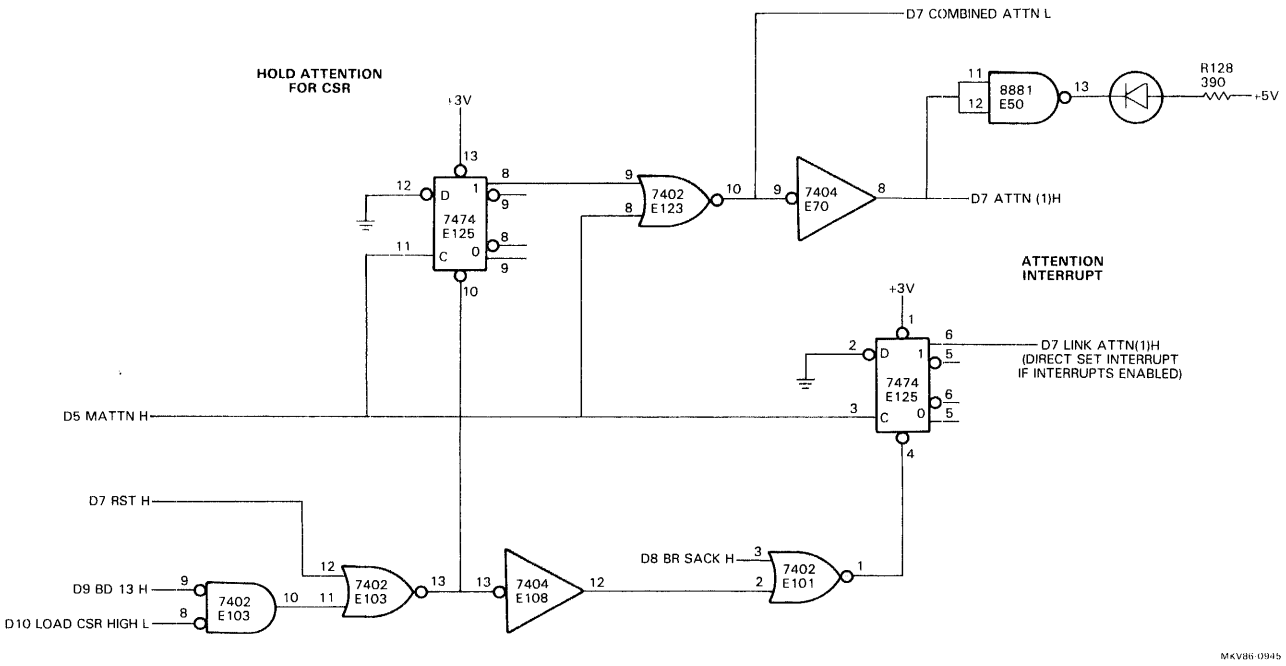

Figure 6-11 Attention Logic

The rising edge of D5 MATTN H (derived from the user signal ATTENTION H) sets both halves of the E125 flip-flop. The attention signal now travels through two separate paths: one path goes to the attention bit in the CSR while the other path causes an interrupt.

6.17.4.1 Attention to the CSR – The attention bit presented in the CSR is the logical OR of two sources:

- The D5 MATTN H signal, coming (almost) directly from the user device.
- The attention hold flip-flop (E125-8).

As long as the user device holds ATTENTION H asserted, the attention bit in the CSR reads as 1. If the user device merely pulses ATTENTION H, then the attention hold flip-flop holds that pulse until the flip-flop is cleared. As long as attention hold stays set, the attention bit in the CSR reads as a 1.

The attention hold flip-flop may be cleared by:

- Initializing the DR11-W.
- Setting the go bit.
- Writing a 0 to the CSR's attention bit.

If the user device is only pulsing ATTENTION H, then care must be taken in programming the DR11-W or the attention hold flip-flop may be accidentally cleared. See Chapter 4 for details.

6.17.4.2 Attention Causing an Interrupt – The attention interrupt flip-flop (the lower half of the E125 flip-flop) is used to hold ATTENTION H until an interrupt can be generated. Whether the user drives ATTENTION H with a pulse or a level, the action is unchanged.

Assuming that the flip-flop is not already being direct cleared, the attention interrupt flip-flop (E125-6) sets upon the rising edge of D5 MATTN H. If the interrupt enable flip-flop (not shown) is also set, the DR11-W requests that the processor be interrupted. The attention interrupt flip-flop remains set until one of four things happens:

- The interrupt actually occurs.
- The DR11-W is initialized.
- The go bit is set.
- The CSR's attention bit is written with a 0.

The first path is the method by which the attention interrupt flip-flop is normally cleared. When the DR11-W becomes the UNIBUS master and transmits its interrupt vector to the processor, the signal D8 BR SACK H is asserted. This signal clears the attention interrupt flip-flop.

The last method is the path by which the software can accidentally clear the attention interrupt flip-flop. The same cautions that apply to the attention hold flip-flop apply to the attention interrupt flip-flop as well.

6.18 INTERRUPT LOGIC

The final piece of logic is the interrupt logic. This logic is responsible for the DR11-W interrupting the PDP-11 or VAX-11 processor and passing the interrupt service vector on the UNIBUS (see Figure 6-12).

Three conditions may set the interrupt request flip-flop:

- The ready bit setting while interrupts are enabled.
- The software setting the go bit while an error condition still exists.
- The user device asserting ATTENTION H while the DR11-W is ready (sitting idle).

When the interrupt request flip-flop sets, the request is passed to the DC013 BR arbitration chip. All of the logic required by the DR11-W to arbitrate for the UNIBUS is contained in the DC013 chip.

The DC013 can only process one request at a time, so if the DC013 is presently the UNIBUS master (and finishing the previous request), E98-6 blocks the next request momentarily.

Once a request is made of the DC013, the DC013 asserts D8 BRx to PLUG L. This signal is routed to the BR-level selection plug, where it is connected to one of the four UNIBUS interrupt request lines (either BUS BR4 L, BUS BR5 L, BUS BR6 L, or BUS BR7 L).

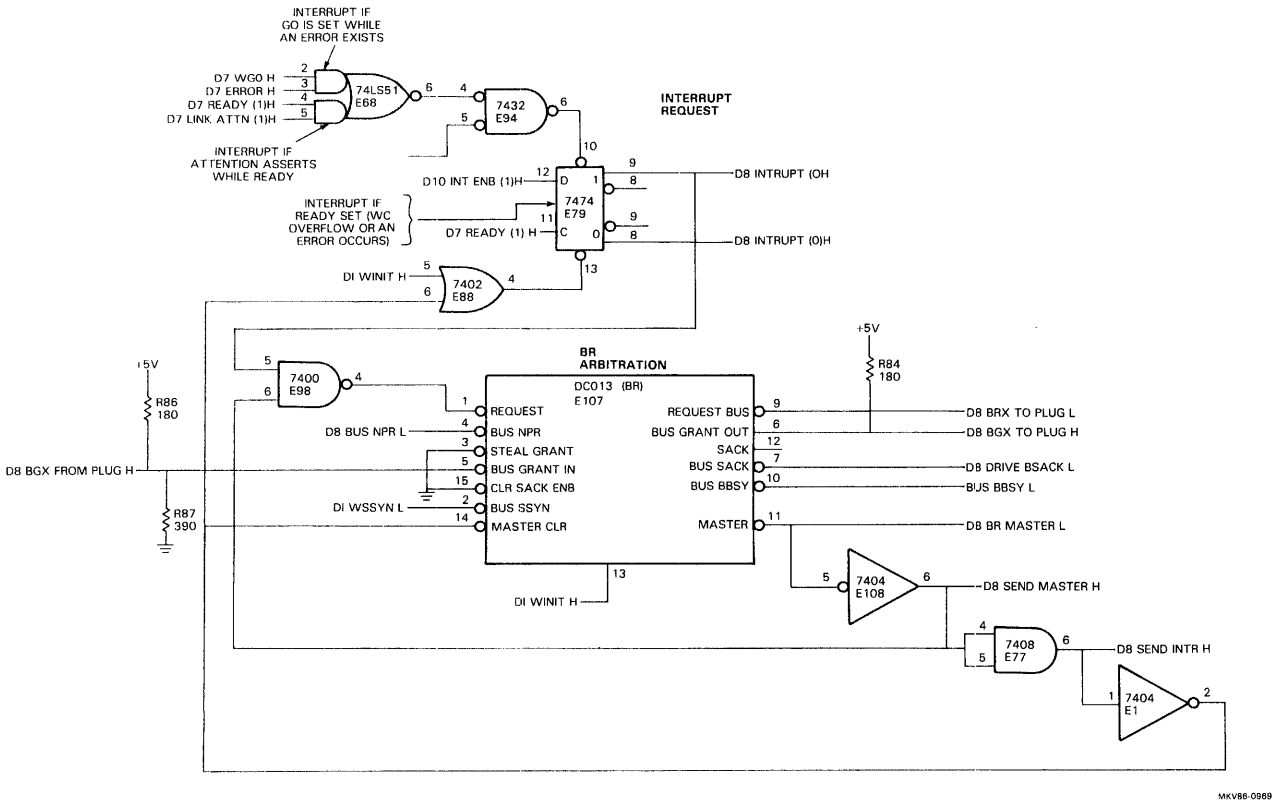

Figure 6-12 Interrupt Arbitration Logic

When the processor wishes to grant the interrupt, the processor sends the appropriate bus-grant signal (for example, BUS BG5 H in response to BUS BR5 L). The BR-level selection plug intercepts this signal and routes it to the DC013. Upon seeing the bus grant signal, the DC013 asserts BUS SACK L (via D8 DRIVE BSACK L). BUS SACK L (selection acknowledge) informs the central UNIBUS arbitrator that the grant was picked up (acknowledged) by an interrupting device. The DC013 then waits until the data section of the UNIBUS is idle. When the data section next becomes idle, the DC013 asserts BUS BBSY L and deasserts BUS SACK L. Asserting BUS BBSY L indicates that the data section of the UNIBUS is once again busy. Deasserting BUS SACK L indicates to the central arbitrator that the DR11-W is willing to let the arbitrator select the next UNIBUS master. The DR11-W is now master of the UNIBUS.

Once the DR11-W becomes the UNIBUS master, the signals D8 BR MASTER L, D8 BR MASTER H, and D8 SEND INTR H are all asserted, in that order. D8 BR MASTER L asserts the DR11-W's interrupt service vector address onto the UNIBUS, after which D8 SEND INTR H causes the UNIBUS signal BUS INTR L to be asserted. This signal strobes the vector into the processor. The processor then replies with BUS SSYN L, which causes the DC013 to end the transaction.

APPENDIX A
THE VAX/VMS XADRIVER

This Appendix contains a listing of the Macro-32 source for the VAX/VMS XADRIVER – the device driver for the DR11-W. This driver is included with all VAX/VMS systems.

This is the listing from VAX/VMS version 4.4 and is subject to change. It is provided in this manual as an example.

```
        .TITLE  XADRIVER - VAX/VMS DR11-W AND DRV11-WA DRIVER
        .IDENT  'X-5'

;
;*****************************************************************************
;*                                                                           *
;*  COPYRIGHT (c) 1978, 1980, 1982, 1984, 1985 BY                            *
;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
;*  ALL RIGHTS RESERVED.                                                     *
;*                                                                           *
;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
;*  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE           *
;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER      *
;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY       *
;*  TRANSFERRED.                                                             *
;*                                                                           *
;*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE     *
;*  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT         *
;*  CORPORATION.                                                             *
;*                                                                           *
;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS      *
;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
;*                                                                           *
;*                                                                           *
;*****************************************************************************
;
;++
;
; FACILITY:
;
;       VAX/VMS Executive, I/O Drivers
;
; ABSTRACT:
;
;       This module contains the driver for the DR11-W (Unibus) and
;       DRV11-WA (Q-bus).  Since the driver was originally written for
;       the DR11-W, many inline comments refer to the "DR11-W" and "Unibus"
;       but apply equally well to the DRV11-WA and the Q-bus.
;
```

```
;               Tables for loading and dispatching
;               Controller initialization routine
;               FDT routine
;               The start I/O routine
;               The interrupt service routine
;               Device specific Cancel I/O
;               Error logging register dump routine
;
; ENVIRONMENT:
;
;       Kernal Mode, Non-paged
;
; AUTHOR:
;
;       C. A. Sameulson 10-JAN-79
;
;
; MODIFIED BY:
;
;       V04-005 DGB0127         Donald G. Blair         19-Sep-1985
;               Clean up and document MicroVAX II support.
;
;       V04-004 DGB0124         Donald G. Blair         25-Jul-1985
;               Add support for the DRV11-WA on MicroVAX II.
;
;       V04-003 DGB0112         Donald G. Blair         31-Jan-1985
;               Move the IO$M_RESET bit to a new location so it no
;               longer coincides with the IO$M_INHERLOG bit.
;
;       V04-002 DGB0106         Donald G. Blair         07-Dec-1984
;               Fix synchronization problem which occurs in the
;               cancel routine if an I/O completes while we're trying
;               to cancel it.
;
;       V04-001 JLV0395         Jake VanNoy              6-SEP-1984
;               Add AVL bit to DEVCHAR.
;
;       V03-006 TMK0001         Todd M. Katz            07-Dec-1983
;               Fix a broken branch.
;
;       V03-005 JLV0304         Jake VanNoy             24-AUG-1983
;               Several bug fixes. All word writes to XA_CSR now have
;               ATTN set so as to prevent lost interrupts. Attention
;               AST list is synchronized at device IPL in DEL_ATTNAST.
;               Correct status is returned on a set mode ast that
;               is returns through EXE$FINISHIO. REQCOM's are always
;               done at FIPL.  Signed division that prevented full size
;               transfers has been fixed.
;
;
;       V03-004 KDM0059         Kathleen D. Morse       14-Jul-1983
;               Change time-wait loops to use new TIMEDWAIT macro.
;               Add $DEVDEF.
;
;       V03-003 KDM0002         Kathleen D. Morse       28-Jun-1982
;               Added $DYNDEF, $DCDEF, and $SSDEF.
;--
```

```
        .SBTTL  External and local symbol definitions

; External symbols

        $ACBDEF                         ; AST control block
        $CRBDEF                         ; Channel request block
        $DCDEF                          ; Device types
        $DDBDEF                         ; Device data block
        $DEVDEF                         ; Device characteristics
        $DPTDEF                         ; Driver prolog table
        $DYNDEF                         ; Dynamic data structure types
        $EMBDEF                         ; EMB offsets
        $IDBDEF                         ; Interrupt data block
        $IODEF                          ; I/O function codes
        $IPLDEF                         ; Hardware IPL definitions
        $IRPDEF                         ; I/O request packet
        $PRDEF                          ; Internal processor registers
        $PRIDEF                         ; Scheduler priority increments
        $SSDEF                          ; System status codes
        $UCBDEF                         ; Unit control block
        $VECDEF                         ; Interrupt vector block
        $XADEF                          ; Define device specific
                                        ;   characteristics

; Local symbols

; Argument list (AP) offsets for device-dependent QIO parameters

P1      = 0                             ; First QIO parameter
P2      = 4                             ; Second QIO parameter
P3      = 8                             ; Third QIO parameter
P4      = 12                            ; Fourth QIO parameter
P5      = 16                            ; Fifth QIO parameter
P6      = 20                            ; Sixth QIO parameter

; Other constants

XA_DEF_TIMEOUT  = 10                    ; 10 second default device timeout
XA_DEF_BUFSIZ   = 65535                 ; Default buffer size
XA_RESET_DELAY  = <<2+9>/10>            ; Delay N microseconds after RESET
                                        ;  (rounded up to 10 microsecond
                                        ;  intervals)

; DR11-W definitions that follow the standard UCB fields
; *** N O T E ***  ORDER OF THESE UCB FIELDS IS ASSUMED

        $DEFINI UCB
        .=UCB$L_DPC+4
$DEF    UCB$L_XA_ATTN                   ; Attention AST listhead
                .BLKL   1
$DEF    UCB$W_XA_CSRTMP                 ; Temporary storage of CSR image
                .BLKW   1
$DEF    UCB$W_XA_BARTMP                 ; Temporary storage of BAR image
                .BLKW   1
$DEF    UCB$W_XA_CSR                    ; Saved CSR on interrupt
                .BLKW   1
$DEF    UCB$W_XA_EIR                    ; Saved EIR on interrupt
                .BLKW   1
$DEF    UCB$W_XA_IDR                    ; Saved IDR on interrupt
                .BLKW   1
$DEF    UCB$W_XA_BAR                    ; Saved BAR register on interrupt
                .BLKW   1
```

```
$DEF    UCB$W_XA_WCR                    ; Saved WCR register on interrupt
                .BLKW   1
$DEF    UCB$W_XA_ERROR                  ; Saved device status flag
                .BLKW   1
$DEF    UCB$L_XA_DPR                    ; Data Path Register contents
                .BLKL   1
$DEF    UCB$L_XA_FMPR                   ; Final Map Register contents
                .BLKL   1
$DEF    UCB$L_XA_PMPR                   ; Previous Map Register contents
                .BLKL   1
$DEF    UCB$W_XA_DPRN                   ; Saved Datapath Register Number
                .BLKW   1               ; And Datapath Parity error flag
$DEF    UCB$W_XA_BAETMP                 ; Temporary storage of BAE (DRV11-WA
                .BLKW   1               ;   only)
$DEF    UCB$W_XA_BAE                    ; Saved BAE register (DRV11-WA only)
                .BLKW   1

; Bit positions for device-dependent status field in UCB

        $VIELD  UCB,0,<-                ; UCB device specific bit definitions
                <ATTNAST,,M>,-          ; ATTN AST requested
                <UNEXPT,,M>,-           ; Unexpected interrupt received
                <IGNORE_UNEXPT,,M>,-    ; Ignore initial interrupt on DRV11-WA
                >
UCB$K_SIZE=.
        $DEFEND UCB

; Device register offsets from CSR address

        $DEFINI XA                      ; Start of DR11-W definitions
$DEF    XA_WCR                          ; Word count
                        .BLKW   1
$DEF    XA_BAR                          ; Buffer address
$DEF    XA_BAE                          ; Buffer address extension (DRV11-WA)
                        .BLKW   1
$DEF    XA_CSR                          ; Control/status

; Bit positions for device control/status register

        $EQULST XA$K_,,0,1,<-           ; Define CSR FNCT bit values
                <FNCT1,2>-
                <FNCT2,4>-
                <FNCT3,8>-
                <STATUSA,2048>-         ; Define CSR STATUS bit values
                <STATUSB,1024>-
                <STATUSC,512>-
        >
        $VIELD  XA_CSR,0,<-             ; Control/status register
                <GO,,M>,-               ; Start device
                <FNCT,3,M>,-            ; CSR FNCT bits
                <XBA,2,M>,-             ; Extended address bits
                <IE,,M>,-               ; Enable interrupts
                <RDY,,M>,-              ; Device ready for command
                <CYCLE,,M>,-            ; Starts slave transmit
                <STATUS,3,M>,-          ; CSR STATUS bits
                <MAINT,,M>,-            ; Maintenance bit
                <ATTN,,M>,-             ; Status from other processor
                <NEX,,M>,-              ; Nonexistent memory flag
                <ERROR,,M>,-            ; Error or external interrupt
        >
$DEF    XA_EIR                          ; Error information register
```

```
; Bit positions for error information register

        $VIELD  XA_EIR,0,<-                     ; Error information register
                <REGFLG,,M>,-                   ; Flags whether EIR or CSR is accessed
                <SPARE,7,M>,-                   ; Unused - spare
                <BURST,,M>,-                    ; Burst mode transfer occured
                <DLT,,M>,-                      ; Time-out for successive burst xfer
                <PAR,,M>,-                      ; Parity error during DATI/P
                <ACLO,,M>,-                     ; Power fail on this processor
                <MULTI,,M>,-                    ; Multi-cycle request error
                <ATTN,,M>,-                     ; ATTN - same as in CSR
                <NEX,,M>,-                      ; NEX - same as in CSR
                <ERROR,,M>,-                    ; ERROR - same as in CSR
        >
                .BLKW   1
$DEF    XA_IDR                                  ; Input Data Buffer register
$DEF    XA_ODR                                  ; Output Data Buffer register
                .BLKW   1

        $DEFEND XA                              ; End of DR11-W definitions


        .SBTTL  Device Driver Tables

; Driver prologue table

        DPTAB   -                                       ; DPT-creation macro
                END=XA_END,-                            ; End of driver label
                ADAPTER=UBA,-                           ; Adapter type
                FLAGS=DPT$M_SVP,-                       ; Allocate system page table
                UCBSIZE=UCB$K_SIZE,-                    ; UCB size
                NAME=XADRIVER                           ; Driver name
        DPT_STORE INIT                                  ; Start of load
                                                        ; initialization table
        DPT_STORE UCB,UCB$B_FIPL,B,8                    ; Device fork IPL
        DPT_STORE UCB,UCB$B_DIPL,B,22                   ; Device interrupt IPL
        DPT_STORE UCB,UCB$L_DEVCHAR,L,<-                ; Device characteristics
                DEV$M_AVL!-                             ; Available
                DEV$M_RTM!-                             ; Real Time device
                DEV$M_ELG!-                             ; Error Logging enabled
                DEV$M_IDV!-                             ;   input device
                DEV$M_ODV>                              ;   output device
        DPT_STORE UCB,UCB$B_DEVCLASS,B,DC$_REALTIME     ; Device class
        DPT_STORE UCB,UCB$B_DEVTYPE,B,DT$_DR11W         ; Device Type
        DPT_STORE UCB,UCB$W_DEVBUFSIZ,W,-               ; Default buffer size
                XA_DEF_BUFSIZ
        DPT_STORE REINIT                                ; Start of reload
                                                        ; initialization table
        DPT_STORE DDB,DDB$L_DDT,D,XA$DDT                ; Address of DDT
        DPT_STORE CRB,CRB$L_INTD+4,D,-                  ; Address of interrupt
                XA_INTERRUPT                            ; service routine
        DPT_STORE CRB,CRB$L_INTD+VEC$L_INITIAL,-        ; Address of controller
                D,XA_CONTROL_INIT                       ; initialization routine
        DPT_STORE END                                   ; End of initialization
                                                        ; tables

; Driver dispatch table

        DDTAB   -                                       ; DDT-creation macro
                DEVNAM=XA,-                             ; Name of device
                START=XA_START,-                        ; Start I/O routine
                FUNCTB=XA_FUNCTABLE,-                   ; FDT address
                CANCEL=XA_CANCEL,-                      ; Cancel I/O routine
```

```
                REGDMP=XA_REGDUMP,-             ; Register dump routine
                DIAGBF=<<15*4>+<<3+5+1>*4>>,-    ; Diagnostic buffer size
                ERLGBF=<<15*4>+<1*4>+<EMB$L_DV_REGSAV>> ; Error log buffer size

;
; Function dispatch table
;
XA_FUNCTABLE:                                   ; FDT for driver
        FUNCTAB ,-                              ; Valid I/O functions
                <READPBLK,READLBLK,READVBLK,WRITEPBLK,WRITELBLK,WRITEVBLK,-
                SETMODE,SETCHAR,SENSEMODE,SENSECHAR>
        FUNCTAB ,                               ; No buffered functions
        FUNCTAB XA_READ_WRITE,-                 ; Device-specific FDT
                <READPBLK,READLBLK,READVBLK,WRITEPBLK,WRITELBLK,WRITEVBLK>
        FUNCTAB +EXE$READ,<READPBLK,READLBLK,READVBLK>
        FUNCTAB +EXE$WRITE,<WRITEPBLK,WRITELBLK,WRITEVBLK>
        FUNCTAB XA_SETMODE,<SETMODE,SETCHAR>
        FUNCTAB +EXE$SENSEMODE,<SENSEMODE,SENSECHAR>



        .SBTTL  XA_CONTROL_INIT, Controller initialization

;++
; XA_CONTROL_INIT, Called when driver is loaded, system is booted, or
; power failure recovery.
;
; Functional Description:
;
;       1) Allocates the direct data path permanently
;       2) Assigns the controller data channel permanently
;       3) Clears the Control and Status Register
;       4) If power recovery, requests device time-out
;
; Inputs:
;
;       R4 = address of CSR
;       R5 = address of IDB
;       R6 = address of DDB
;       R8 = address of CRB
;
; Outputs:
;
;       VEC$V_PATHLOCK bit set in CRB$L_INTD+VEC$B_DATAPATH
;       UCB address placed into IDB$L_OWNER
;
;
;--

XA_CONTROL_INIT:

        MOVL    IDB$L_UCBLST(R5),R0             ; Address of UCB
        MOVL    R0,IDB$L_OWNER(R5)              ; Make permanent controller owner
        BISW    #UCB$M_ONLINE,UCB$W_STS(R0)
                                                ; Set device status "on-line"
        CPUDISP <<UV1,3$>,-                     ; branch to handle MicroVAX I
                <UV2,5$>>,-                     ; branch to handle MicroVAX II
                CONTINUE=YES                    ; else continue for all other
                                                ;   processors
        BRB     9$
3$:     BUG_CHECK UNSUPRTCPU,FATAL              ; DRV11-WA not supported on MicroVAX I
5$:     MOVB    #DT$_XA_DRV11WA,-               ; If this is a Q-bus, then this is
                  UCB$B_DEVTYPE(R0)             ;   a DRV11-WA rather than a DR11-W.
```

```
; On the DRV11-WA, the interrupt enable bit normally remains set at all
; times since an interrupt is generated if the bit makes a low-to-high
; transition when there isn't a DMA transfer in progress.  Since the
; device has the IE bit clear at power-up, an interrupt will be generated
; when we set the IE bit.  Therefore, we tell the interrupt service
; routine to ignore the first unexpected interrupt that occurs.

        BBS     #XA_CSR$V_IE,-          ; Br if IE bit already set.
                  XA_CSR(R4),9$
        BBSS    #UCB$V_IGNORE_UNEXPT,-  ; Else interrupt will occur.
                  UCB$W_DEVSTS(R0),9$

; If powerfail has occured and device was active, force device time-out.
; The user can set his own time-out interval for each request.  Time-
; out is forced so a very long time-out period will be short circuited.

9$:     BBS     #UCB$V_POWER,UCB$W_STS(R0),10$
                                        ; Branch if powerfail
        BISB    #VEC$M_PATHLOCK,CRB$L_INTD+VEC$B_DATAPATH(R8)
                                        ; Permanently allocate direct datapath
10$:
        BSBW    XA_DEV_RESET            ; Reset DR11W
        RSB                             ; Done



        .SBTTL  XA_READ_WRITE, FDT for device data transfers

;++
; XA_READ_WRITE, FDT for READLBLK,READVBLK,READPBLK,WRITELBLK,WRITEVBLK,
;                        WRITEPBLK
;
; Functional description:
;
;       1) Rejects QUEUE I/O's with odd transfer count
;       2) Rejects QUEUE I/O's for BLOCK MODE request to UBA Direct Data
;          PATH on odd byte boundary
;       3) Stores request time-out count specified in P3 into IRP
;       4) Stores FNCT bits specified in P4 into IRP
;       5) Stores word to write into ODR from P5 into IRP
;       6) Checks block mode transfers for memory modify access
;
; Inputs:
;
;       R3 = Address of IRP
;       R4 = Address of PCB
;       R5 = Address of UCB
;       R6 = Address of CCB
;       R8 = Address of FDT routine
;       AP = Address of P1
;               P1 = Buffer Address
;               P2 = Buffer size in bytes
;               P3 = Request time-out period (conditional on IO$M_TIMED)
;               P4 = Value for CSR FNCT bits (conditional on IO$M_SETFNCT)
;               P5 = Value for ODR (conditional on IO$M_SETFNCT)
;               P6 = Address of Diagnostic Buffer
;
; Outputs:
;
;       R0 = Error status if odd transfer count
;       IRP$L_MEDIA = Time-out count for this request
;       IRP$L_SEGVBN = FNCT bits for DR11-W CSR and ODR image
;
;--
```

```
XA_READ_WRITE:

; The IO$M_INHERLOG ("inhibit error logging") function modifier was not
; intended to be used by this driver.  However, since the definition for
; the IO$M_RESET modifier used to be the same as that for IO$M_INHERLOG,
; the error logging routines incorrectly used the IO$M_RESET bit to
; determine whether it should log errors.  To solve this problem, the
; definition for IO$M_RESET was changed.  For the sake of old programs, we
; manually move the RESET bit to its new location.

        BBCC    #IO$V_INHERLOG,IRP$W_FUNC(R3),1$
                                        ; br if old reset bit not set
        BISW    #IO$M_RESET,IRP$W_FUNC(R3)
                                        ; set new reset bit
1$:     BLBC    P2(AP),10$              ; Branch if transfer count even
2$:     MOVZWL  #SS$_BADPARAM,R0        ; Set error status code
5$:     JMP     G^EXE$ABORTIO           ; Abort request
10$:    MOVZWL  IRP$W_FUNC(R3),R1       ; Fetch I/O Function code
        MOVL    P3(AP),IRP$L_MEDIA(R3)  ; Set request specific time-out count
        BBS     #IO$V_TIMED,R1,15$      ; Branch if time-out specified
        MOVL    #XA_DEF_TIMEOUT,IRP$L_MEDIA(R3)
                                        ; Else set default timeout value
15$:    BBC     #IO$V_DIAGNOSTIC,R1,20$ ; Branch if not maintenance reqeust
        EXTZV   #IO$V_FCODE,#IO$S_FCODE,R1,R1 ; AND out all function modifiers
        CMPB    #IO$_READPBLK,R1        ; If maintenance function, must be
                                        ; physical I/O read or write
        BEQL    20$
        CMPB    #IO$_WRITEPBLK,R1
        BEQL    20$
        MOVZWL  #SS$_NOPRIV,R0          ; No privilege for operation
        BRB     5$                      ; Abort request
20$:    EXTZV   #0,#3,P4(AP),R0         ; Get value for FNCT bits
        ASHL    #XA_CSR$V_FNCT,R0,IRP$L_SEGVBN(R3)
                                        ; Shift into position for CSR
        MOVW    P5(AP),IRP$L_SEGVBN+2(R3) ; Store ODR value for later

; If this is a block mode transfer, check buffer for modify access
; whether or not the function is read or write.  The DR11-W does
; not decide whether to read or write, the users device does.
; For word mode requests, return to read check or write check.
;
; If this is a BLOCK MODE request and the UBA Direct Data Path is
; in use, check the data buffer address for word alignment.  If buffer
; is not word aligned, reject the request.

        BBS     #IO$V_WORD,IRP$W_FUNC(R3),30$
                                        ; Branch if word mode transfer
        BBS     #XA$V_DATAPATH,UCB$L_DEVDEPEND(R5),25$
                                        ; Branch if Buffered Data Path in use
        BLBS    P1(AP),2$               ; DDP, branch on bad alignment
25$:    JMP     G^EXE$MODIFY            ; Checke buffer for modify access
30$:    RSB                             ; Return



        .SBTTL  XA_SETMODE, Set Mode, Set characteristics FDT

;++
; XA_SETMODE, FDT routine to process SET MODE and SET CHARACTERISTICS
;
```

```
; Functional description:
;
;       If IO$M_ATTNAST modifier is set, queue attention AST for device
;       If IO$M_DATAPATH modifier is set, queue packet.
;       Else, finish I/O.
;
; Inputs:
;
;       R3 = I/O packet address
;       R4 = PCB address
;       R5 = UCB address
;       R6 = CCB address
;       R7 = Function code
;       AP = QIO Paramater list address
;
; Outputs:
;
;       If IO$M_ATTNAST is specified, queue AST on UCB attention AST list.
;       If IO$M_DATAPATH is specified, queue packet to driver.
;       Else, use exec routine to update device characteristics
;
;--

XA_SETMODE:
        MOVZWL  IRP$W_FUNC(R3),R0       ; Get entire function code
        BBC     #IO$V_ATTNAST,R0,20$    ; Branch if not an ATTN AST

; Attention AST request

        PUSHR   #^M<R4,R7>
        MOVAB   UCB$L_XA_ATTN(R5),R7    ; Address of ATTN AST control block
                                        ;   list
        JSB     G^COM$SETATTNAST        ; Set up attention AST
        POPR    #^M<R4,R7>
        BLBC    R0,50$                  ; Branch if error
        BISW    #UCB$M_ATTNAST,UCB$W_DEVSTS(R5)
                                        ; Flag ATTN AST expected.
        BBC     #UCB$V_UNEXPT,UCB$W_DEVSTS(R5),10$
                                        ; Deliver AST if unsolicited interrupt
        BSBW    DEL_ATTNAST
10$:    MOVZBL  #SS$_NORMAL,R0          ; Set status
        JMP     G^EXE$FINISHIOC         ; Thats all for now (clears R1)

; If modifier IO$M_DATAPATH is set,
; queue packet.  The data path is changed at driver level to preserve
; order with other requests.

20$:    BBS     S^#IO$V_DATAPATH,R0,30$ ; If BDP modifier set, queue packet

        JMP     G^EXE$SETCHAR           ; Set device characteristics

; This is a request to change data path useage, queue packet

30$:    CMPL    #IO$_SETCHAR,R7         ; Set characteristics?
        BNEQ    45$                     ; No, must have the privelege
        JMP     G^EXE$SETMODE           ; Queue packet to start I/O

; Error, abort IO

45$:    MOVZWL  #SS$_NOPRIV,R0          ; No priv for operation
50$:    CLRL    R1
        JMP     G^EXE$ABORTIO           ; Abort IO on error
```

```
        .SBTTL  XA_START, Start I/O routines

;++
; XA_START - Start a data transfer, set characteristics, enable ATTN AST.
;
; Functional Description:
;
;       This routine has two major functions:
;
;       1) Start an I/O transfer.  This transfer can be in either word
;          or block mode.  The FNCTN bits in the DR11-W CSR are set.  If
;          the transfer count is zero, the STATUS bits in the DR11-W CSR
;          are read and the request completed.
;       2) Set Characteristics.  If the function is change data path, the
;          new data path flag is set in the UCB.
;
; Inputs:
;
;       R3 = Address of the I/O request packet
;       R5 = Address of the UCB
;
; Outputs:
;
;       R0 = final status and number of bytes transferred
;       R1 = value of CSR STATUS bits and value of input data buffer register
;       Device errors are logged
;       Diagnostic buffer is filled
;
;--
        .ENABL  LSB

XA_START:

; Retrieve the address of the device CSR

        ASSUME  IDB$L_CSR EQ 0
        MOVL    UCB$L_CRB(R5),R4        ; Address of CRB
        MOVL    @CRB$L_INTD+VEC$L_IDB(R4),R4
                                        ; Address of CSR

; Fetch the I/O function code

        MOVZWL  IRP$W_FUNC(R3),R1       ; Get entire function code
        MOVW    R1,UCB$W_FUNC(R5)       ; Save FUNC in UCB for Error Logging
        EXTZV   #IO$V_FCODE,#IO$S_FCODE,R1,R2 ; Extract function field

; Dispatch on function code.  If this is SET CHARACTERISTICS, we will
; select a data path for future use.
; If this is a transfer function, it will either be processed in word
; or block mode.

        CMPB    #IO$_SETCHAR,R2         ; Set characteristics?
        BNEQ    3$

;++
; SET CHARACTERISTICS - Process Set Characteristics QIO function
;
; INPUTS:
;
;       XA_DATAPATH bit in Device Characteristics specifies which data path
;       to use.  If bit is a one, use buffered data path.  If zero, use
;       direct datapath.
;
```

```
; OUTPUTS:
;
;       CRB is flagged as to which datapath to use.
;       DEVDEPEND bits in device characteristics is updated
;               XA_DATAPATH = 1 -> buffered data path in use
;               XA_DATAPATH = 0 -> direct data path in use
;--

        MOVL    UCB$L_CRB(R5),R0                ; Get CRB address
        MOVQ    IRP$L_MEDIA(R3),UCB$B_DEVCLASS(R5) ; Set device characteristics
        BISB    #VEC$M_PATHLOCK,CRB$L_INTD+VEC$B_DATAPATH(R0)
                                                ; Assume direct datapath
        BBC     #XA$V_DATAPATH,UCB$L_DEVDEPEND(R5),2$ ;  Were we right?
        BICB    #VEC$M_PATHLOCK,CRB$L_INTD+VEC$B_DATAPATH(R0)
                                                ; Set buffered datapath
2$:
        CLRL    R1                              ; Return Success
        MOVZWL  #SS$_NORMAL,R0
        REQCOM

; If subfunction modifier for device reset is set, do one here

3$:     BBC     S^#IO$V_RESET,R1,4$     ; Branch if not device reset
        BSBW    XA_DEV_RESET            ; Reset DR11-W

; This must be a data transfer function - i.e. READ OR WRITE
; Check to see if this is a zero length transfer.
; If so, only set CSR FNCT bits and return STATUS from CSR

4$:     TSTW    UCB$W_BCNT(R5)          ; Is transfer count zero?
        BNEQ    10$                     ; No, continue with data transfer
        BBC     S^#IO$V_SETFNCT,R1,6$   ; Set CSR FNCT specified?
        DSBINT
        MOVW    IRP$L_SEGVBN+2(R3),XA_ODR(R4)
                                        ; Store word in ODR
        MOVZWL  XA_CSR(R4),R0
        BICW    #<XA_CSR$M_FNCT!XA_CSR$M_ERROR>,R0
        BISW    IRP$L_SEGVBN(R3),R0
        BISW    #XA_CSR$M_ATTN,R0       ; Force ATTN on to prevent lost
                                        ;   interrupt
        MOVW    R0,XA_CSR(R4)
        BBC     #XA$V_LINK,UCB$L_DEVDEPEND(R5),5$ ; Link mode?
        BICW3   #XA$K_FNCT2,R0,XA_CSR(R4)         ; Make FNCT bit 2 a pulse
5$:
        ENBINT
6$:
        BSBW    XA_REGISTER             ; Fetch DR11-W registers
        BLBS    R0,7$                   ; If error, then log it
        JSB     G^ERL$DEVICERR          ; Log a device error
7$:     JSB     G^IOC$DIAGBUFILL        ; Fill diagnostic buffer if specified
        MOVL    UCB$W_XA_CSR(R5),R1     ; Return CSR and EIR in R1
        MOVZWL  UCB$W_XA_ERROR(R5),R0   ; Return status in R0
        BISB    #XA_CSR$M_IE,XA_CSR(R4) ; Enable device interrupts
        REQCOM                          ; Request done

; Build CSR image in R0 for later use in starting transfers

10$:
        MOVZWL  UCB$W_BCNT(R5),R0       ; Fetch byte count
        DIVL3   #2,R0,UCB$L_XA_DPR(R5)  ; Make byte count into word count
        ;
        ; Set up UCB$W_CSRTMP used for loading CSR later
        ;
        MOVZWL  XA_CSR(R4),R0
```

```
        BICW    #^C<XA_CSR$M_FNCT>,R0
        BISW    #XA_CSR$M_IE!XA_CSR$M_ATTN,R0    ; Set Interrupt Enable and ATTN
        BBC     S^#IO$V_SETFNCT,R1,20$  ; Set FNCT bits in CSR?
        BICW    #<XA_CSR$M_FNCT>,R0     ; Yes, Clear previous FNCT bits
        BISB    IRP$L_SEGVBN(R3),R0     ; OR in new value
20$:    BBC     S^#IO$V_DIAGNOSTIC,R1,23$
                                        ; Check for maintenance function
        BISW    #XA_CSR$M_MAINT,R0      ; Set maintenance bit in CSR image

; Is this a word mode or block mode request?

23$:    MOVW    R0,UCB$W_XA_CSRTMP(R5)  ; Save CSR image in UCB
        BBC     S^#IO$V_WORD,R1,BLOCK_MODE ; Check if word or block mode
        BRW     WORD_MODE               ; Branch to handle word mode
;++
; BLOCK MODE -- Process a Block Mode (DMA) transfer request
;
; FUNCTIONAL DESCRIPTION:
;
;       This routine takes the buffer address, buffer size, fucntion code,
;       and function modifier fields from the IRP.  It calculates the UNIBUS
;       address, allocates the UBA map registers, loads the DR11-W device
;       registers and starts the request.
;--
; Set up UBA
; Start transfer

BLOCK_MODE:

; If IO$M_CYCLE subfunction is specified, set CYCLE bit in CSR image
        BBC     #IO$V_CYCLE,R1,25$      ; Set CYCLE bit in CSR?
        BISW    #XA_CSR$M_CYCLE,UCB$W_XA_CSRTMP(R5) ; If yes, or into CSR image

; Allocate UBA data path and map registers

25$:
        REQDPR                          ; Request UBA data path
        REQMPR                          ; Request UBA map registers
        LOADUBA                         ; Load UBA map reqisters

; Calculate the UNIBUS transfer address for the DR11-W from the UBA
; map register address and byte offset.

        MOVZWL  UCB$W_BOFF(R5),R1       ; Byte offset in first page of xfer
        MOVL    UCB$L_CRB(R5),R2        ; Address of CRB
        INSV    CRB$L_INTD+VEC$W_MAPREG(R2),#9,#9,R1
                                        ; Insert page number
        EXTZV   #16,#2,R1,R2            ; Extract bits 17:16 of bus address
        CMPB    #DT$_DR11W,-            ; If this is a DR11-W,
                  UCB$B_DEVTYPE(R5)
        BEQL    100$                    ;   then branch.
        MOVW    R2,UCB$W_XA_BAETMP(R5)  ; Save value of BAE prior to transfer
        CLRL    R2                      ; Clear XBA bits
100$:   ASHL    #XA_CSR$V_XBA,R2,R2     ; Shift extended memory bits for CSR
        BISW    #XA_CSR$M_GO,R2         ; Set "GO" bit into CSR image
        BISW    R2,UCB$W_XA_CSRTMP(R5)  ; Set into CSR image we are building
        BICW3   #<XA_CSR$M_GO!XA_CSR$M_CYCLE>,UCB$W_XA_CSRTMP(R5),R0
                                        ; CSR image less "GO" and "CYCLE"
        BICW3   #XA$K_FNCT2,UCB$W_XA_CSRTMP(R5),R2 ; CSR image less FNCT bit 2
        MOVW    R1,UCB$W_XA_BARTMP(R5)  ; Save BAR for error logging
; At this juncture:
;       R0 = CSR image less "GO" and "CYCLE"
;       R1 = low 16 bits of transfer bus address
;       R2 = CSR image less FNCT bit 2
```

```
;       UCB$L_XA_DPR(R5) = transfer count in words
;       UCB$W_XA_CSRTMP(R5) = CSR image to start transfer with

; Set DR11-W registers and start transfer
; Note that read-modify-write cycles are NOT performed to the DR11-W CSR.
; The CSR is always written directly into.  This prevents inadvertently setting
; the EIR select flag (writing bit 15) if error happens to become true.

        DSBINT                              ; Disable interrupts (powerfail)
        MNEGW   UCB$L_XA_DPR(R5),XA_WCR(R4)
                                            ; Load negative of transfer count
        MOVW    R1,XA_BAR(R4)               ; Load low 16 bits of bus address
        CMPB    #DT$_DR11W,-                ; If this is a DR11-W,
                  UCB$B_DEVTYPE(R5)
        BEQL    200$                        ;   then branch.
        MOVW    UCB$W_XA_BAETMP(R5),-       ; Load high bits of bus address
                  XA_BAE(R4)
200$:   MOVW    R0,XA_CSR(R4)               ; Load CSR image less "GO" and "CYCLE"
        BBC     #XA$V_LINK,UCB$L_DEVDEPEND(R5),26$ ; Link mode?
        MOVW    R2,XA_CSR(R4)               ; Yes, load CSR image less "FNCT" bit 2
        BRB     126$            ; Only if link mode in dev characteristics
26$:
        MOVW    UCB$W_XA_CSRTMP(R5),XA_CSR(R4) ; Move all bits to CSR

; Wait for transfer complete interrupt, powerfail, or device time-out

126$:
        WFIKPCH XA_TIME_OUT,IRP$L_MEDIA(R3) ; Wait for interrupt

; Device has interrupted, FORK

        IOFORK                              ; FORK to lower IPL

; Handle request completion, release UBA resources, check for errors

        MOVZWL  #SS$_NORMAL,-(SP)           ; Assume success, store code on stack
        CLRW    UCB$W_XA_DPRN(R5)           ; Clear DPR number and DPR error flag
        PURDPR                              ; Purge UBA buffered data path
        BLBS    R0,27$                      ; Branch if no datapath error
        MOVZWL  #SS$_PARITY,(SP)            ; Flag parity error on device
        INCB    UCB$W_XA_DPRN+1(R5)         ; Flag PDR error for log
27$:    MOVL    R1,UCB$L_XA_DPR(R5)         ; Save data path register in UCB
        EXTZV   #VEC$V_DATAPATH,-           ; Get Datapath register no.
                #VEC$S_DATAPATH,-           ; For Error Log
                CRB$L_INTD+VEC$B_DATAPATH(R3),R0
        MOVB    R0,UCB$W_XA_DPRN(R5)        ; Save for later in UCB
        EXTZV   #9,#7,UCB$W_XA_BAR(R5),R0 ; Low bits, final map register no.
        CMPB    #DT$_DR11W,-                ; If this is a DR11-W,
                  UCB$B_DEVTYPE(R5)
        BEQL    300$                        ;   then branch.
        MOVZWL  UCB$W_XA_BAE(R5),R1         ; Fetch high bits of map register no.
        BRB     310$
300$:   EXTZV   #4,#2,UCB$W_XA_CSR(R5),R1 ; Hi bits of map register no.
310$:   INSV    R1,#7,#2,R0                 ; Entire map register number
        CMPW    R0,#496                     ; Is map register number in range?
        BGTR    28$                         ; No, forget it - compound error
        MOVL    (R2)[R0],UCB$L_XA_FMPR(R5) ; Save map register contents
        CLRL    UCB$L_XA_PMPR(R5)           ; Assume no previous map register
        DECL    R0                          ; Was there a previous map register?
        CMPV    #VEC$V_MAPREG,#VEC$S_MAPREG,-
                CRB$L_INTD+VEC$W_MAPREG(R3),R0
        BGTR    28$                         ; No if gtr
        MOVL    (R2)[R0],UCB$L_XA_FMPR(R5)
                                            ; Save previous map register contents
```

```
28$:    RELMPR                          ; Release UBA resources
        RELDPR

; Check for errors and return status

        TSTW    UCB$W_XA_WCR(R5)        ; All words transferred?
        BEQL    30$                     ; Yes
        MOVZWL  #SS$_OPINCOMPL,(SP)     ; No, flag operation not complete
30$:    BBC     #XA_CSR$V_ERROR,UCB$W_XA_CSR(R5),35$ ; Branch on CSR error bit
        MOVZWL  UCB$W_XA_ERROR(R5),(SP) ; Flag for controller/drive error
                                        ;   status
        BSBW    XA_DEV_RESET            ; Reset DR11-W
35$:    BLBS    (SP),40$                ; Any errors after all this?

        CMPW    (SP),#SS$_OPINCOMPL     ; Log the error, unless this is
        BNEQ    37$                     ;   a DRV11-WA running in link mode
        CMPB    #DT$_DR11W,-            ;   and the operation is incomplete,
                  UCB$B_DEVTYPE(R5)     ;   in which case it is an expected
        BEQL    37$                     ;   error and not worth logging.
        BBS     #XA$V_LINK,-            ;   ...
                  UCB$L_DEVDEPEND(R5),40$ ; ...
37$:    JSB     G^ERL$DEVICERR          ; Log the error.

40$:    BSBW    DEL_ATTNAST             ; Deliver outstanding ATTN AST's
        JSB     G^IOC$DIAGBUFILL        ; Fill diagnostic buffer
        MOVL    (SP)+,R0                ; Get final device status
        MULW3   #2,UCB$W_XA_WCR(R5),R1  ; Calculate final transfer count
        ADDW    UCB$W_BCNT(R5),R1
        INSV    R1,#16,#16,R0           ; Insert into high byte of IOSB
        MOVL    UCB$W_XA_CSR(R5),R1     ; Return CSR and EIR in IOSB
        BISB    #XA_CSR$M_IE,XA_CSR(R4) ; Enable interrupts
        REQCOM                          ; Finish request in exec
        .DSABL  LSB
;++
; WORD MODE -- Process word mode (interrupt per word) transfer
;
; FUNCTIONAL DESCRIPTION:
;
;       Data is transferred one word at a time with an interrupt for each word.

;       The request is handled separately for a write (from memory to DR11-W
;       and a read (from DR11-W to memory).
;       For a write, data is fetched from memory, loaded into the ODR of the
;       DR11-W and the system waits for an interrupt.  For a read, the system
;       waits for a DR11-W interrupt and the IDR is transferred into memory.
;       If the unsolicited interrupt flag is set, the first word is transferred
;       directly into memory withou waiting for an interrupt.
;--

        .ENABL  LSB
WORD_MODE:


; Dispatch to separate loops on READ or WRITE

        CMPB    #IO$_READPBLK,R2        ; Check for read function
        BEQL    30$

;++
; WORD MODE WRITE -- Write (output) in word mode
;
```

```
; FUNCTIONAL DESCRIPTION:
;
;       Transfer the requested number of words from user memory to
;       the DR11-W ODR one word at a time, wait for interrupt for each
;       word.
;--

10$:
        BSBW    MOVFRUSER                       ; Get two bytes from user buffer
        DSBINT                                  ; Lock out interrupts
                                                ; Flag interrupt expected
        MOVW    R1,XA_ODR(R4)                   ; Move data to DR11-W
        MOVW    UCB$W_XA_CSRTMP(R5),XA_CSR(R4) ; Set DR11-W CSR
        BBC     #XA$V_LINK,UCB$L_DEVDEPEND(R5),15$ ; Link mode?
        BICW3   #XA$K_FNCT2,UCB$W_XA_CSRTMP(R5),XA_CSR(R4)
                                                ; Clear interrupt FNCT bit 2
                                                ; Only if link mode specified
15$:

; Wait for interrupt, powerfail, or device time-out

        WFIKPCH XA_TIME_OUTW,IRP$L_MEDIA(R3)

; Check for errors, decrement transfer count, and loop til complete

        IOFORK                                  ; Fork to lower IPL
        CMPB    #DT$_DR11W,-                    ; Branch if this is a DR11-W
                  UCB$B_DEVTYPE(R5)
        BEQL    17$
        BBC     #XA_CSR$V_ERROR,-               ; DRV11-WA - check ERROR bit in CSR.
                  UCB$W_XA_CSR(R5),20$          ; Branch on success.
        BRW     40$                             ; Branch on error.
17$:    BITW    #XA_EIR$M_NEX!-
                XA_EIR$M_MULTI!-
                XA_EIR$M_ACLO!-
                XA_EIR$M_PAR!-
                XA_EIR$M_DLT,UCB$W_XA_EIR(R5) ; Any errors?
        BEQL    20$                             ; No, continue
        BRW     40$                             ; Yes, abort transfer.
20$:    DECW    UCB$L_XA_DPR(R5)                        ; All words trnasferred?
        BNEQ    10$                             ; No, loop until finished.

; Transfer is done, clear iterrupt expected flag and FORK
; All words read or written in WORD MODE.  Finish I/O.

RETURN_STATUS:

        JSB     G^IOC$DIAGBUFILL                ; Fill diagnostic buffer if present
        BSBW    DEL_ATTNAST                     ; Deliver outstanding ATTN AST's
        MOVZWL  #SS$_NORMAL,R0                  ; Complete success status
22$:    MULW3   #2,UCB$L_XA_DPR(R5),R1          ; Calculate actual bytes xfered
        SUBW3   R1,UCB$W_BCNT(R5),R1            ; From requested number of bytes
        INSV    R1,#16,#16,R0                   ; And place in high word of R0
        MOVL    UCB$W_XA_CSR(R5),R1             ; Return CSR and EIR status
        BISB    #XA_CSR$M_IE,XA_CSR(R4)         ; Enable device interrupts
        REQCOM                                  ; Finish request in exec
;++
; WORD MODE READ -- Read (input) in word mode
;
```

```
; FUNCTIONAL DESCRIPTION:
;
;       Transfer the requested number of words from the DR11-W IDR into
;       user memory one word at a time, wait for interrupt for each word.
;       If the unexpected (unsolicited) interrupt bit is set, transfer the
;       first (last received) word to memory without waiting for an
;       interrupt.
;--

30$:
        DSBINT  UCB$B_DIPL(R5)          ; Lock out interrupts

; If an unexpected (unsolicited) interrupt has occurred, assume it
; is for this READ request and return value to user buffer without
; waiting for an interrupt.

        BBCC    #UCB$V_UNEXPT,-
                UCB$W_DEVSTS(R5),32$    ; Branch if no unexpected interrupt
        ENBINT                          ; Enable interrupts
        BRB     37$                     ; continue

32$:
        SETIPL  #IPL$_POWER
35$:

; Wait for interrupt, powerfail, or device time-out

        WFIKPCH XA_TIME_OUTW,IRP$L_MEDIA(R3)

; Check for errors, decrement transfer count and loop until done

        IOFORK                          ; Fork to lower IPL
37$:
        CMPB    #DT$_DR11W,-            ; Branch if this is a DR11-W
                  UCB$B_DEVTYPE(R5)
        BEQL    1037$
        BBC     #XA_CSR$V_ERROR,-       ; DRV11-WA - check ERROR bit in CSR.
                  UCB$W_XA_CSR(R5),1038$ ; Branch on success.
        BRW     40$                     ; Branch on error.
1037$:  BITW    #XA_EIR$M_NEX!-
                XA_EIR$M_MULTI!-
                XA_EIR$M_ACLO!-
                XA_EIR$M_PAR!-
                XA_EIR$M_DLT,UCB$W_XA_EIR(R5) ; Any errors?
        BNEQ    40$                     ; Yes, abort transfer.
1038$:  BSBW    MOVTOUSER               ; Store two bytes into user buffer

; Send interrupt back to sender.  Acknowledge we got last word.

        DSBINT
        MOVW    UCB$W_XA_CSRTMP(R5),XA_CSR(R4)
        BBC     #XA$V_LINK,UCB$L_DEVDEPEND(R5),38$ ; Link mode?
        BICW3   #XA$K_FNCT2,UCB$W_XA_CSRTMP(R5),XA_CSR(R4) ; Yes, clear FNCT 2
38$:
        DECW    UCB$L_XA_DPR(R5)                ; Decrement transfer count
        BNEQ    35$                     ; Loop until all words transferred
        ENBINT
        BRW     RETURN_STATUS           ; Finish request in common code
; Error detected in word mode transfer

40$:
        BSBW    DEL_ATTNAST             ; Deliver ATTN AST's
        BSBW    XA_DEV_RESET            ; Error, reset DR11-W
        JSB     G^IOC$DIAGBUFILL        ; Fill diagnostic buffer if presetn
```

```
        JSB     G^ERL$DEVICERR          ; Log device error
        MOVZWL  UCB$W_XA_ERROR(R5),R0   ; Set controller/drive status in R0
        BRW     22$

        .DSABL  LSB
;
; MOVFRUSER - Routine to fetch two bytes from user buffer.
;
; INPUTS:
;
;       R5 = UCB address
;
; OUTPUTS:
;
;       R1 = Two bytes of data from users buffer
;       Buffer descriptor in UCB is updated.
;
        .ENABL  LSB
MOVFRUSER:
        MOVAL   -(SP),R1                ; Address of temporary stack loc
        MOVZBL  #2,R2                   ; Fetch two bytes
        JSB     G^IOC$MOVFRUSER         ; Call exec routine to do the deed
        MOVL    (SP)+,R1                ; Retreive the bytes
        BRB     20$                     ; Update UCB buffer pointers

;
; MOVTOUSER - Routine to store two bytes into users buffer.
;
; INPUTS:
;
;       R5 = UCB address
;       UCB$W_XA_IDR(R5) = Location where two bytes are saved
;
; OUTPUTS:
;
;       Two bytes are stored in user buffer and buffer descriptor in
;       UCB is updated.
;
MOVTOUSER:
        MOVAB   UCB$W_XA_IDR(R5),R1     ; Address of internal buffer
        MOVZBL  #2,R2
        JSB     G^IOC$MOVTOUSER         ; Call exec
20$:                                    ; Update buffer pointers in UCB
        ADDW    #2,UCB$W_BOFF(R5)       ; Add two to buffer descriptor
        BICW    #^C<^X01FF>,UCB$W_BOFF(R5) ; Modulo the page size
        BNEQ    30$                     ; If NEQ, no page boundary crossed
        ADDL    #4,UCB$L_SVAPTE(R5)     ; Point to next page
30$:
        RSB
;
        .DSABL  LSB


        .SBTTL  DR11-W DEVICE TIME-OUT

;++
; DR11-W device TIME-OUT
; If a DMA transfer was in progress, release UBA resources.
; For DMA or WORD mode, deliver ATTN AST's, log a device timeout error,
; and do a hard reset on the controller.
;
; Clear DR11-W CSR
```

```
; Return error status
;
; Power failure will appear as a device time-out
;--
        .ENABL  LSB
XA_TIME_OUT:                            ; Time-out for DMA transfer

        SETIPL  UCB$B_FIPL(R5)          ; Lower to FORK IPL
        PURDPR                          ; Purge buffered data path in UBA
        RELMPR                          ; Release UBA map registers
        RELDPR                          ; Release UBA data path
        BRB     10$                     ; continue

XA_TIME_OUTW:                           ; Time-out for WORD mode transfer

        SETIPL  UCB$B_FIPL(R5)          ; Lower to FORK IPL
10$:    MOVL    UCB$L_CRB(R5),R4        ; Fetch address of CSR
        MOVL    @CRB$L_INTD+VEC$L_IDB(R4),R4
        BSBW    XA_REGISTER             ; Read DR11-W registers
        JSB     G^IOC$DIAGBUFILL        ; Fill diagnostic buffer
        JSB     G^ERL$DEVICTMO          ; Log device time out
        BSBW    DEL_ATTNAST             ; And deliver the AST's
        BSBW    XA_DEV_RESET            ; Reset controller
        MOVZWL  #SS$_TIMEOUT,R0         ; Assume error status
        BBC     #UCB$V_CANCEL,-
                UCB$W_STS(R5),20$       ; Branch if not cancel
        MOVZWL  #SS$_CANCEL,R0          ; Set status
20$:    CLRL    R1
        BICW    #UCB$M_ATTNAST!UCB$M_UNEXPT,UCB$W_DEVSTS(R5)
                                        ; Clear unwanted flags.
        BICW    #<UCB$M_TIM!UCB$M_INT!UCB$M_TIMOUT!UCB$M_CANCEL!UCB$M_POWER>,-
                UCB$W_STS(R5)           ; Clear unit status flags
        REQCOM                          ; Complete I/O in exec
        .DSABL  LSB



        .SBTTL  XA_INTERRUPT, Interrupt service routine for DR11-W

;++
; XA_INTERRUPT, Handles interrupts generated by DR11-W
;
; Functional description:
;
;       This routine is entered whenever an interrupt is generated
;       by the DR11-W.  It checks that an interrupt was expected.
;       If not, it sets the unexpected (unsolicited) interrupt flag.
;       All device registers are read and stored into the UCB.
;       If an interrupt was expected, it calls the driver back at its Wait
;       For Interrupt point.
;       Deliver ATTN AST's if unexpected interrupt.
;
; Inputs:
;
;       00(SP) = Pointer to address of the device IDB
;       04(SP) = saved R0
;       08(SP) = saved R1
;       12(SP) = saved R2
;       16(SP) = saved R3
;       20(SP) = saved R4
;       24(SP) = saved R5
;       28(SP) = saved PSL
;       32(SP) = saved PC
;
```

```
; Outputs:
;
;       The driver is called at its Wait For Interrupt point if an
;       interrupt was expected.
;       The current value of the DR11-W CSR's are stored in the UCB.
;
;--
XA_INTERRUPT:                           ; Interrupt service for DR11-W
        MOVL    @(SP)+,R4               ; Address of IDB and pop SP
        MOVQ    (R4),R4                 ; CSR and UCB address from IDB

; Read the DR11-W device registers (WCR, BAR, CSR, EIR, IDR) and store
; into UCB.

        BSBW    XA_REGISTER             ; Read device registers

; Check to see if device transfer request active or not
; If so, call driver back at Wait for Interrupt point and
; Clear unexpected interrupt flag.

20$:    BBCC    #UCB$V_INT,UCB$W_STS(R5),25$
                                        ; If clear, no interrupt expected

; Interrupt expected, clear unexpected interrupt flag and call driver
; back.

        BICW    #UCB$M_UNEXPT,UCB$W_DEVSTS(R5)
                                        ; Clear unexpected interrupt flag
        MOVL    UCB$L_FR3(R5),R3        ; Restore drivers R3
        JSB     @UCB$L_FPC(R5)          ; Call driver back
        BRB     30$

; Deliver ATTN AST's if no interrupt expected and set unexpected
; interrupt flag.

25$:    BBSC    #UCB$V_IGNORE_UNEXPT,-  ; Ignore spurious interrupt -
                  UCB$W_DEVSTS(R5),30$  ;   (DRV11-WA only.)
        BISW    #UCB$M_UNEXPT,UCB$W_DEVSTS(R5) ; Set unexpected interrupt flag
        BSBW    DEL_ATTNAST             ; Deliver ATTN AST's
        BISB    #XA_CSR$M_IE,XA_CSR(R4) ; Enable device interrupts

; Restore registers and return from interrupt

30$:
        POPR    #^M<R0,R1,R2,R3,R4,R5>  ; Restore registers
        REI                             ; Return from interrupt


        .SBTTL  XA_REGISTER - Handle DR11-W CSR transfers

;++
; XA_REGISTER - Routine to handle DR11-W register transfers
;
; INPUTS:
;
;       R4 - DR11-W CSR address
;       R5 - UCB address of unit
;
```

```
; OUTPUTS:
;
;       CSR, EIR, WCR, BAR, BAE, IDR, and status are read and stored into UCB.
;       The DR11-W is placed in its initial state with interrupts enabled.
;       R0 - .true. if no hard error
;            .false. if hard error (cannot clear ATTN)
;
; If the CSR ERROR bit is set and the associated condition can be cleared, then
; the error is transient and recoverable.  The status returned is SS$_DRVERR.
; If the CSR ERROR bit is set and cannot be cleared by clearing the CSR, then
; this is a hard error and cannot be recovered.  The returned status is
; SS$_CTRLERR.
;
;       R0,R1 - destroyed, all other registers preserved.
;--

XA_REGISTER:

        MOVZWL  #SS$_NORMAL,R0          ; Assume success
        MOVZWL  XA_CSR(R4),R1           ; Read CSR
        MOVW    R1,UCB$W_XA_CSR(R5)     ; Save CSR in UCB
        BBC     #XA_CSR$V_ERROR,R1,55$  ; Branch if no error
        MOVZWL  #SS$_DRVERR,R0          ; Assume "drive" error
55$:    BICW    #^C<XA_CSR$M_FNCT>,R1   ; Clear all uninteresting bits for
                                        ;   later
        CMPB    #DT$_XA_DRV11WA,-       ; If this is a DRV11-WA,
                  UCB$B_DEVTYPE(R5)     ;
        BEQL    57$                     ;   then branch.
        BISB    #<XA_CSR$M_ERROR/256>,XA_CSR+1(R4) ; Set EIR flag
        MOVW    XA_EIR(R4),UCB$W_XA_EIR(R5) ; Save EIR in UCB
        BRB     59$
57$:    BISW    #XA_CSR$M_IE,R1         ; On the DRV11-WA, if the IE bit makes
                                        ; a 0->1 transition while READY=1, a
                                        ; spurious interrupt in generated.
                                        ; Therefore, we leave IE high at all
                                        ; times.
59$:    MOVW    R1,XA_CSR(R4)           ; Clear EIR flag and errors
        MOVW    XA_CSR(R4),R1           ; Read CSR back
        BBC     #XA_CSR$V_ATTN,R1,60$   ; If attention still set, hard error
        MOVZWL  #SS$_CTRLERR,R0         ; Flag hard controller error
60$:    MOVW    XA_IDR(R4),UCB$W_XA_IDR(R5) ; Save IDR in UCB
        MOVW    XA_BAR(R4),UCB$W_XA_BAR(R5)
        CMPB    #DT$_DR11W,-            ; If this is a DR11-W,
                  UCB$B_DEVTYPE(R5)     ;
        BEQL    70$                     ;   then branch.
        MOVW    XA_BAE(R4),UCB$W_XA_BAE(R5) ; Save BAE in UCB
70$:    MOVW    XA_WCR(R4),UCB$W_XA_WCR(R5)
        MOVW    R0,UCB$W_XA_ERROR(R5)   ; Save status in UCB
        RSB


        .SBTTL  XA_CANCEL, Cancel I/O routine

;++
; XA_CANCEL, Cancels an I/O operation in progress
;
; Functional description:
;
;       Flushes Attention AST queue for the user.
;       If transfer in progress, do a device reset to DR11-W and finish the
;       request.
;       Clear interrupt expected flag.
;
```

```
; Inputs:
;
;       R2 = negated value of channel index
;       R3 = address of current IRP
;       R4 = address of the PCB requesting the cancel
;       R5 = address of the device's UCB
;
; Outputs:
;
;--

XA_CANCEL:                                          ; Cancel I/O

        BBCC    #UCB$V_ATTNAST,-
                UCB$W_DEVSTS(R5),20$    ; ATTN AST enabled?

; Finish all ATTN AST's for this process.

        PUSHR   #^M<R2,R6,R7>
        MOVL    R2,R6                   ; Set up channel number
        MOVAB   UCB$L_XA_ATTN(R5),R7    ; Address of listhead
        JSB     G^COM$FLUSHATTNS        ; Flush ATTN AST's for process
        POPR    #^M<R2,R6,R7>

; Check to see if a data transfer request is in progress
; for this process on this channel

20$:
        DSBINT  UCB$B_DIPL(R5)          ; Lock out device interrupts
        BBC     #UCB$V_INT,-            ; br if I/O not in progress
                UCB$W_STS(R5),30$
        JSB     G^IOC$CANCELIO          ; Check if transfer going
        BBC     #UCB$V_CANCEL,-
                UCB$W_STS(R5),30$       ; Branch if not for this guy
;
; Force timeout
;
        CLRL    UCB$L_DUETIM(R5)        ; clear timer
        BISW    #UCB$M_TIM,UCB$W_STS(R5) ; set timed
        BICW    #UCB$M_TIMOUT,-
                UCB$W_STS(R5)           ; Clear timed out
30$:
        ENBINT                          ; Lower to FORK IPL
        RSB                             ; Return


        .SBTTL  DEL_ATTNAST, Deliver ATTN AST's

;++
; DEL_ATTNAST, Deliver all outstanding ATTN AST's
;
; Functional description:
;
;       This routine is used by the DR11-W driver to deliver all of the
;       outstanding attention AST's.  It is copied from COM$DELATTNAST in
;       the exec.  In addition, it places the saved value of the DR11-W CSR
;       and Input Data Buffer Register in the AST paramater.
;
; Inputs:
;
;       R5 = UCB of DR11-W unit
;
```

```
; Outputs:
;
;       R0,R1,R2 Destroyed
;       R3,R4,R5 Preserved
;--
DEL_ATTNAST:
        DSBINT  UCB$B_DIPL(R5)          ; Device IPL
        BBCC    #UCB$V_ATTNAST,UCB$W_DEVSTS(R5),30$
                                        ; Any ATTN AST's expected?
        PUSHR   #^M<R3,R4,R5>           ; Save R3,R4,R5
10$:    MOVL    8(SP),R1                ; Get address of UCB
        MOVAB   UCB$L_XA_ATTN(R1),R2    ; Address of ATTN AST listhead
        MOVL    (R2),R5                 ; Address of next entry on list
        BEQL    20$                     ; No next entry, end of loop
        BICW    #UCB$M_UNEXPT,UCB$W_DEVSTS(R1)
                                        ; Clear unexpected interrupt flag
        MOVL    (R5),(R2)               ; Close list
        MOVW    UCB$W_XA_IDR(R1),ACB$L_KAST+6(R5)
                                        ; Store IDR in AST paramater
        MOVW    UCB$W_XA_CSR(R1),ACB$L_KAST+4(R5)
                                        ; Store CSR in AST paramater
        PUSHAB  B^10$                   ; Set return address for FORK
        FORK                            ; FORK for this AST

; AST fork procedure

        MOVQ    ACB$L_KAST(R5),ACB$L_AST(R5)
                                        ; Re-arrange entries
        MOVB    ACB$L_KAST+8(R5),ACB$B_RMOD(R5)
        MOVL    ACB$L_KAST+12(R5),ACB$L_PID(R5)
        CLRL    ACB$L_KAST(R5)
        MOVZBL  #PRI$_IOCOM,R2          ; Set up priority increment
        JMP     G^SCH$QAST              ; Queue the AST

20$:    POPR    #^M<R3,R4,R5>           ; Restore registers
30$:    ENBINT                          ; Enable interrupts
        RSB                             ; Return


        .SBTTL  XA_REGDUMP - DR11-W register dump routine

;++
; XA_REGDUMP - DR11-W Register dump routine.
;
; This routine is called to save the controller registers in a specified
; buffer.  It is called from the device error logging routine and from the
; diagnostic buffer fill routine.
;
; Inputs:
;
;       R0 - Address of register save buffer
;       R4 - Address of Control and Status Register
;       R5 - Address of UCB
;
; Outputs:
;
;       The controller registers are saved in the specified buffer.
;
;               CSRTMP - The last command written to the DR11-W CSR by
;                        by the driver.
;               BARTMP - The last value written into the DR11-W BAR by
;                        the driver during a block mode transfer.
```

```
;               CSR - The CSR image at the last interrupt
;               EIR - The EIR image at the last interrupt
;               IDR - The IDR image at the last interrupt
;               BAR - The BAR image at the last interrupt
;               WCR - Word count register
;               ERROR - The system status at request completion
;               PDRN - UBA Datapath Register number
;               DPR - The contents of the UBA Data Path register
;               FMPR - The contents of the last UBA Map register
;               PMRP - The contents of the previous UBA Map register
;               DPRF - Flag for purge datapath error
;                       0 = no purger datapath error
;                       1 = parity error when datapath was purged
;               BAETMP - The last value written to the BAE by the
;                        driver during a block mode transfer (DRV11-WA only)
;               BAE - The BAE image at the last interrupt (DRV11-WA only)
;
;       Note that the values stored are from the last completed transfer
;       operation. If a zero transfer count is specified, then the
;       values are from the last operation with a non-zero transfer count.
;--

XA_REGDUMP:

        MOVZBL  #15,(R0)+                       ; 15 registers are stored.
        MOVAB   UCB$W_XA_CSRTMP(R5),R1          ; Get address of saved register images
        MOVZBL  #8,R2                           ; Return 8 registers here
10$:    MOVZWL  (R1)+,(R0)+
        SOBGTR  R2,10$                          ; Move them all
        MOVZBL  UCB$W_XA_DPRN(R5),(R0)+         ; Save Datapath Register number
        MOVZBL  #3,R2                           ; And 3 more here
20$:    MOVL    (R1)+,(R0)+                     ; Move UBA register contents
        SOBGTR  R2,20$
        MOVZBL  UCB$W_XA_DPRN+1(R5),(R0)+ ; Save Datapath Parity Error Flag
        MOVZWL  UCB$W_XA_BAETMP(R5),(R0)+ ; Save BAE stored prior to xfer
        MOVZWL  UCB$W_XA_BAE(R5),(R0)+  ; Save BAE store following xfer
        RSB


        .SBTTL  XA_DEV_RESET - Device reset DR11-W

;++
; XA_DEV_RESET - DR11-W Device reset routine
;
; This routine raises IPL to device IPL, performs a device reset to
; the required controler, and re-enables device interrupts.
;
; Inputs:
;
;       R4 - Address of Control and Status Register
;       R5 - Address of UCB
;
; Outputs:
;
;       Controller is reset, controller interrupts are enabled
;
;--
```

```
XA_DEV_RESET:

        PUSHR   #^M<R0,R1,R2>           ; Save some registers
        DSBINT                          ; Raise IPL to lock all interrupts
        CMPB    #DT$_DR11W,-            ; If this is a DR11-W,
                  UCB$B_DEVTYPE(R5)     ;
        BEQL    20$                     ;   then branch.
        MOVW    #XA_CSR$M_IE,XA_CSR(R4) ; Clear all writeable bits but IE.
        BITB    #XA_CSR$M_RDY,XA_CSR(R4); If not READY then no xfer in
                                        ;   progress,
        BNEQ    40$                     ; So no need to reset device
        MNEGW   #1,XA_WCR(R4)           ; Tell it only 1 byte left to xfer
        MOVB    #XA_CSR$M_CYCLE/256,-   ; and complete the transfer.
                  XA_CSR+1(R4)
        BRB     30$
        MOVB                               #<XA_CSR$M_MAINT/256>,XA_CSR+1(R4)
        CLRB    XA_CSR+1(R4)

; ***  Must delay here depending on reset interval

30$:    TIMEDWAIT TIME=#XA_RESET_DELAY  ; No. of 10 micro-sec intervals to wait

        MOVB    #XA_CSR$M_IE,XA_CSR(R4) ; Re-enable device interrupts
40$:    ENBINT                          ; Restore IPL
        POPR    #^M<R0,R1,R2>           ; Restore registers

        RSB

XA_END:                                 ; End of driver label
        .END
```

INDEX

READER'S COMMENTS

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

1. **How did you use this manual? (Circle your response.)**

   (a) Installation (c) Maintenance (e) Training
   (b) Operation/use (d) Programming (f) Other (Please specify.) _____

2. **Did the manual meet your needs? Yes ☐ No ☐ Why?** _____

3. **Please rate the manual on the following categories. (Circle your response.)**

	Excellent	Good	Fair	Poor	Unacceptable
Accuracy	5	4	3	2	1
Clarity	5	4	3	2	1
Completeness	5	4	3	2	1
Table of Contents, Index	5	4	3	2	1
Illustrations, examples	5	4	3	2	1
Overall ease of use	5	4	3	2	1

4. **What things did you like *most* about this manual?** _____

5. **What things did you like *least* about this manual?** _____

6. **Please list and describe any errors you found in the manual.**

Page	Description/Location of Error
_____	_____
_____	_____
_____	_____
_____	_____

Name _____ Job Title _____
Street _____ Company _____
City _____ Department _____
State/Country _____ Telephone Number _____
Postal (ZIP) Code _____ Date _____

THANK YOU FOR YOUR COMMENTS AND SUGGESTIONS.

Please do not use this form to order manuals. Contact your representative at Digital Equipment Corporation or (in the USA) call our DECdirect™ department at this toll-free number: 800-258-1710.

MKV

FOLD HERE AND TAPE. DO NOT STAPLE.

digital™

No Postage Necessary if Mailed in the United States

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 33 MAYNARD, MA

POSTAGE WILL BE PAID BY ADDRESSEE

DIGITAL EQUIPMENT CORPORATION
Educational Services/Quality Assurance
12 Crosby Drive BUO/E08
Bedford, MA 01730-1493
USA

FOLD HERE AND TAPE. DO NOT STAPLE.