

ARTISAN®
TECHNOLOGY GROUP

Your **definitive** source for quality pre-owned equipment.

Artisan Technology Group
(217) 352-9330 | sales@artisantg.com | artisantg.com

Full-service, independent repair center
with experienced engineers and technicians on staff.

We buy your excess, underutilized, and idle equipment
along with credit for buybacks and trade-ins.

Custom engineering
so your equipment works exactly as you specify.

- Critical and expedited services
- In stock / Ready-to-ship
- Leasing / Rentals / Demos
- ITAR-certified secure asset solutions

Expert team | Trust guarantee | 100% satisfaction

All trademarks, brand names, and brands appearing herein are the property of their respective owners.

Visit our website - *Click HERE*

VMIVME-DR11W

VMEbus-TO-DR11-W INTERFACE

INSTRUCTION MANUAL

DOCUMENT NO. 500-00DR11-000

Revised September 17, 1992

VME MICROSYSTEMS INTERNATIONAL CORPORATION
12090 SOUTH MEMORIAL PARKWAY
HUNTSVILLE, AL 35803-3308
(205) 880-0444
1-800-322-3616

IMPORTANT NOTICE

CABLE REQUIREMENTS:

The VMIVME-DR11W requires that the following cabling guidelines are followed for proper operation and data integrity:

1. Shielded flat cable (or superior cable) must be used. Mass-terminated standard flat cable ***will*** cause data errors.

2. The cable shield must be grounded at both ends of the cable if a shielded cable is used. The cable grounds should have low impedance at high frequencies.

3. Macrolink interfaces for the Concurrent/Perkin Elmer require a pinout reversal on the cables. When ordering specify macrolink cables.

DR11W COMPATIBILITY:

The VMIVME-DR11W is pinout compatible with DR11-W and DRV11-B type devices. It is ***not*** pinout compatible with DR11-B type devices and damage may result if it is cabled directly to a DR11-B type device.

NOTICE

The information in this document has been carefully checked and is believed to be entirely reliable. While all reasonable efforts to ensure accuracy have been taken in the preparation of this manual, VMIC assumes no responsibility resulting from omissions or errors in this manual, or from the use of information contained herein.

VMIC reserves the right to make any changes, without notice, to this or any of VMIC's products to improve reliability, performance, function, or design.

VMIC does not assume any liability arising out of the application or use of any product or circuit described herein; nor does VMIC convey any license under its patent rights or the rights of others.

VME Microsystems International Corporation

All Rights Reserved

This document shall not be duplicated, nor its contents used for any purpose, unless granted express written permission from VMIC.

Copyright © January 1986 by
VME Microsystems International Corporation

VMIC

RECORD OF REVISIONS

REVISION LETTER	DATE	PAGES INVOLVED	CHANGE NUMBER
A	03/06/86	Entire Manual	Manual - 01
B	07/07/86	Entire Manual	DR11-08
C	02/05/87	Entire Manual	DR11-10
D	04/22/88	Entire Manual	88-0081
E	08/18/88	Section 5	DR11-24
F	08/18/88	Appendix A	88-0118
G	08/18/88	Sections 2, 4, and Appendix A	88-0128
H	08/18/88	Notice, Sections 1, 4, and 5	88-0161
J	10/12/88	Appendix A	88-0219
K	11/17/88	Appendix A	88-0254
L	11/17/88	Appendix A	88-0317
M	12/27/88	Appendix A	88-0348
N	04/11/89	Cover, pages iii, vii, Section 6, and Appendix A	89-0014
P	06/14/89	Release ZZZ and Appendix A	89-0047
R	09/20/89	Cover, pages ii, 1-5, and 4-14	89-0143
T	11/06/89	Cover, page ii, and Appendix A	89-0155
U	01/18/90	Cover, pages ii, 5-8, C1-1, C2-1, C3-1, and C4-1	90-0006
V	05/25/90	Cover, pages ii, 3-10, 4-9, 4-22, and Appendix A	90-0102
W	10/17/90	Cover, page ii, and Appendix A	90-0114
Y	10/29/90	Page ii and Appendix A	90-0164
AA	02/27/91	Cover, page ii, and Appendix A	90-0236
BB	04/22/91	Cover, pages ii, and 2-3	91-0122
CC	08/23/91	Cover, page ii, and Appendix A	91-0216
DD	12/09/91	Convert from ZZZ to 500.	92-0327
EE	09/17/92	Cover, pages v, vi, vii, Sections 2, 6, and Appendix A	92-0327

VMIC
SAFETY SUMMARY

THE FOLLOWING GENERAL SAFETY PRECAUTIONS MUST BE OBSERVED DURING ALL PHASES OF THIS OPERATION, SERVICE, AND REPAIR OF THIS PRODUCT. FAILURE TO COMPLY WITH THESE PRECAUTIONS OR WITH SPECIFIC WARNINGS ELSEWHERE IN THIS MANUAL VIOLATES SAFETY STANDARDS OF DESIGN, MANUFACTURE, AND INTENDED USE OF THE PRODUCT. VME MICROSYSTEMS INTERNATIONAL CORPORATION ASSUMES NO LIABILITY FOR THE CUSTOMER'S FAILURE TO COMPLY WITH THESE REQUIREMENTS.

GROUND THE SYSTEM

To minimize shock hazard, the chassis and system cabinet must be connected to an electrical ground. A three-conductor AC power cable should be used. The power cable must either be plugged into an approved three-contact electrical outlet or used with a three-contact to two-contact adapter with the grounding wire (green) firmly connected to an electrical ground (safety ground) at the power outlet.

DO NOT OPERATE IN AN EXPLOSIVE ATMOSPHERE

Do not operate the system in the presence of flammable gases or fumes. Operation of any electrical system in such an environment constitutes a definite safety hazard.

KEEP AWAY FROM LIVE CIRCUITS

Operating personnel must not remove product covers. Component replacement and internal adjustments must be made by qualified maintenance personnel. Do not replace components with power cable connected. Under certain conditions, dangerous voltages may exist even with the power cable removed. To avoid injuries, always disconnect power and discharge circuits before touching them.

DO NOT SERVICE OR ADJUST ALONE

Do not attempt internal service or adjustment unless another person, capable of rendering first aid and resuscitation, is present.

DO NOT SUBSTITUTE PARTS OR MODIFY SYSTEM

Because of the danger of introducing additional hazards, do not install substitute parts or perform any unauthorized modification to the product. Return the product to VME Microsystems International Corporation for service and repair to ensure that safety features are maintained.

DANGEROUS PROCEDURE WARNINGS

Warnings, such as the example below, precede only potentially dangerous procedures throughout this manual. Instructions contained in the warnings must be followed.

WARNING

DANGEROUS VOLTAGES, CAPABLE OF CAUSING DEATH, ARE PRESENT IN THIS SYSTEM. USE EXTREME CAUTION WHEN HANDLING, TESTING, AND ADJUSTING.

SAFETY SYMBOLS

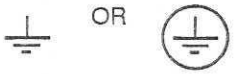
GENERAL DEFINITIONS OF SAFETY SYMBOLS USED IN THIS MANUAL



Instruction manual symbol: the product is marked with this symbol when it is necessary for the user to refer to the instruction manual in order to protect against damage to the system.



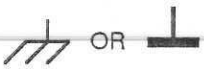
Indicates dangerous voltage (terminals fed from the interior by voltage exceeding 1000 volts are so marked).



Protective conductor terminal. For protection against electrical shock in case of a fault. Used with field wiring terminals to indicate the terminal which must be connected to ground before operating equipment.



Low-noise or noiseless, clean ground (earth) terminal. Used for a signal common, as well as providing protection against electrical shock in case of a fault. Before operating the equipment, terminal marked with this symbol must be connected to ground in the manner described in the installation (operation) manual.



Frame or chassis terminal. A connection to the frame (chassis) of the equipment which normally includes all exposed metal structures.



Alternating current (power line).



Direct current (power line).



Alternating or direct current (power line).

WARNING

The WARNING sign denotes a hazard. It calls attention to a procedure, a practice, a condition, or the like, which, if not correctly performed or adhered to, could result in injury or death to personnel.

CAUTION

The CAUTION sign denotes a hazard. It calls attention to an operating procedure, a practice, a condition, or the like, which, if not correctly performed or adhered to, could result in damage to or destruction of part or all of the system.

NOTE:

The NOTE sign denotes important information. It calls attention to a procedure, a practice, a condition or the like, which is essential to highlight.

VMIVME-DR11W
VMEbus-TO-DR11W INTERFACE

TABLE OF CONTENTS

Page

SECTION 1. INTRODUCTION

1.1 FUNCTIONS 1-1
1.2 FUNCTIONAL DESCRIPTION 1-1
1.3 COMPATIBILITY 1-1
1.4 REFERENCE MATERIAL LIST 1-6

SECTION 2. PHYSICAL DESCRIPTION AND SPECIFICATIONS

SECTION 3. THEORY OF OPERATION

3.1 BLOCK DIAGRAMS 3-1
3.2 OPERATIONAL OVERVIEW 3-1
3.3 TRANSFER PROTOCOL 3-1
3.4 DATA TRANSFER DESCRIPTION 3-8
3.5 VME-TO-VME LINK 3-8
3.6 USER INTERFACE SIGNAL FUNCTION DESCRIPTION 3-9

SECTION 4. PROGRAMMING

4.1 PROGRAMMING OVERVIEW 4-1
4.2 REGISTER BIT DEFINITIONS 4-1
4.3 SEQUENCE OF REGISTER LOADING 4-1
4.4 PROGRAMMING THE SCB68430 DMAI 4-11
4.4.1 Channel Status Register/Channel Error Register 4-11
4.4.1.1 Channel Status Register (CSR) 4-11
4.4.1.2 Channel Error Register (CER) 4-16
4.4.2 Device Control Register/Operation Control Register 4-16
4.4.2.1 Device Control Register (DCR) 4-16
4.4.2.2 Operation Control Register (OCR) 4-17
4.4.3 Sequence Control Register (SCR) 4-17

TABLE OF CONTENTS (Continued)

Page

SECTION 4. PROGRAMMING (Concluded)

4.4.4 Channel Control Register (CCR) 4-17
4.4.5 Memory Transfer Count Registers 4-19
4.4.6 Memory Address Counter Registers 4-19
4.4.7 DMA Interrupt Vector Register 4-19
4.4.8 Channel Priority Register 4-19
4.5 PROGRAMMING THE MC68153 BIM 4-19
4.5.1 Bus Error Interrupt Control Register (BERRICR), Attention Interrupt Control Register (AICR), and DMA Interrupt Control Register (DICR) 4-20
4.5.2 BERR Interrupt Vector Register (BERRIVR) 4-20
4.5.3 Attention Interrupt Vector Register (AIVR) 4-20
4.6 PROGRAMMING THE ON-BOARD REGISTERS 4-21
4.6.1 Board Control Register HIGH/LOW 4-21
4.6.1.1 Board Control Register HIGH (BCRH) 4-21
4.6.1.2 Board Control Register LOW (BCRL) 4-22
4.6.2 I/O Data Register HIGH/LOW (IODRH, IODRL) 4-23
4.6.3 Address Modifier Register 4-23
4.7 SAMPLE SOFTWARE LISTINGS 4-23

SECTION 5. CONFIGURATION AND INSTALLATION

5.1 UNPACKING PROCEDURES 5-1
5.2 PHYSICAL INSTALLATION 5-1
5.3 JUMPER INSTALLATIONS 5-1
5.3.1 VMEbus Priority Jumpers 5-1
5.3.2 Busy Polarity 5-4
5.4 BOARD BASE ADDRESS 5-4
5.5 ADDRESS MODIFIERS 5-4
5.6 I/O CABLES 5-4
5.6.1 Macrolink Pin Number Reversal 5-7
5.7 PIN NUMBER REVERSAL THROUGH BULKHEAD CONNECTOR 5-7
5.8 CABLE CONNECTOR 5-7
5.9 USER CONNECTOR PIN ASSIGNMENTS 5-7

SECTION 6. MAINTENANCE AND WARRANTY

6.1 MAINTENANCE 6-1
6.2 MAINTENANCE PRINTS 6-1
6.3 WARRANTY 6-1

TABLE OF CONTENTS (Continued)

Page

SECTION 6. MAINTENANCE AND WARRANTY (Concluded)

6.4 OUT-OF-WARRANTY REPAIR POLICY 6-2
6.4.1 Repair Category 6-3
6.4.2 Repair Pricing 6-3
6.4.3 Payment 6-3
6.4.4 Shipping Charges 6-4
6.4.5 Shipping Instructions 6-4
6.4.6 Warranty on Repairs 6-4
6.4.7 Exclusions 6-4

LIST OF FIGURES

Figure Page

1.1-1 Printed Circuit Assembly Photograph 1-2
1.2-1 Connection-to-User Device 1-3
1.2-2 VMIVME-DR11W CPU-to-CPU Link 1-4
3.1-1 Functional Block Diagram 3-2
3.1-2 Interface Interconnection for Two DR11Ws in Back-to-Back Configuration 3-3
3.1-3 Interface Connection to a DEC DRV11-WA 3-4
3.1-4 Timing/Handshake Sequencing Data Transfer from the VMIVME-DR11W to the User Device 3-5
3.1-5 Back-to-Back Configuration (Interprocessor Mode) 3-6
3.1-6 Diagram for Data Transfer from the User Device to the VMIVME-DR11W 3-7
5.3-1 VMIVME-DR11W Configuration Information (Rev. H Boards) 5-2
5.3.1-1 Jumper Installation for Selection of VMEbus Priority Level (Rev. H and Subsequent Revisions of the Board) 5-3
5.3.2-1 Busy Signal Polarity Selection 5-5
5.4-1 VMIVME-DR11W Base Address Selection Switch 5-6

TABLE OF CONTENTS (Concluded)

LIST OF TABLES

Table		Page
1.3-1	VMIVME-DR11W Compatible Host Computer Interface Sources	1-5
4.1-1	VMIVME-DR11W Register Address Map	4-2
4.2-1	Register Bit Definitions	4-3
4.2-2	Bus Interrupter Module No. MC68153 Register Bit Definitions	4-7
4.2-3	Discrete On-Board Register Bit Formats	4-9
4.3-1	Register Initialization Sequence for Transmitting a 4 Kword Block Starting Data Address $40000	4-12
4.3-2	Register Initialization Sequence for Transmitting a Second 4 Kword Block Starting Data Address $20000	4-13
4.3-3	Register Initialization Sequence for Receiving a 4 Kword Block Starting Data Address $40000	4-14
4.3-4	Register Initialization Sequence for Receiving a Second 4 Kword Block Starting Data Address $20000	4-15
4.6.3-1	Address Modifier Codes	4-24
5.9-1	Connector Pin Assignments	5-8

APPENDICES

A Assembly Drawing, Parts List, and Schematic
B Integrated Circuit Specifications
C VMIVME-DR11W Test Software Listing

SECTION 1

INTRODUCTION

1.1 FUNCTIONS

The VMIVME-DR11W is a 16-bit high performance DMA interface which is compatible with the DEC DR11-W and DRV11-WA type devices, and with the VMIVME-DR11W CPU links. A photograph of the VMIVME-DR11W printed circuit assembly is shown in Figure 1.1-1.

Features of the VMIVME-DR11W include the following:

a. DEC DR11-W compatible I/O connection

b. VMEbus (IEEE P1014) compatible

c. 16-bit DMA transfers

d. Switch-selectable board address

e. Programmable interrupt vectors and levels

f. Programmable address modifiers

g. Uses VME short I/O address space for on-board registers

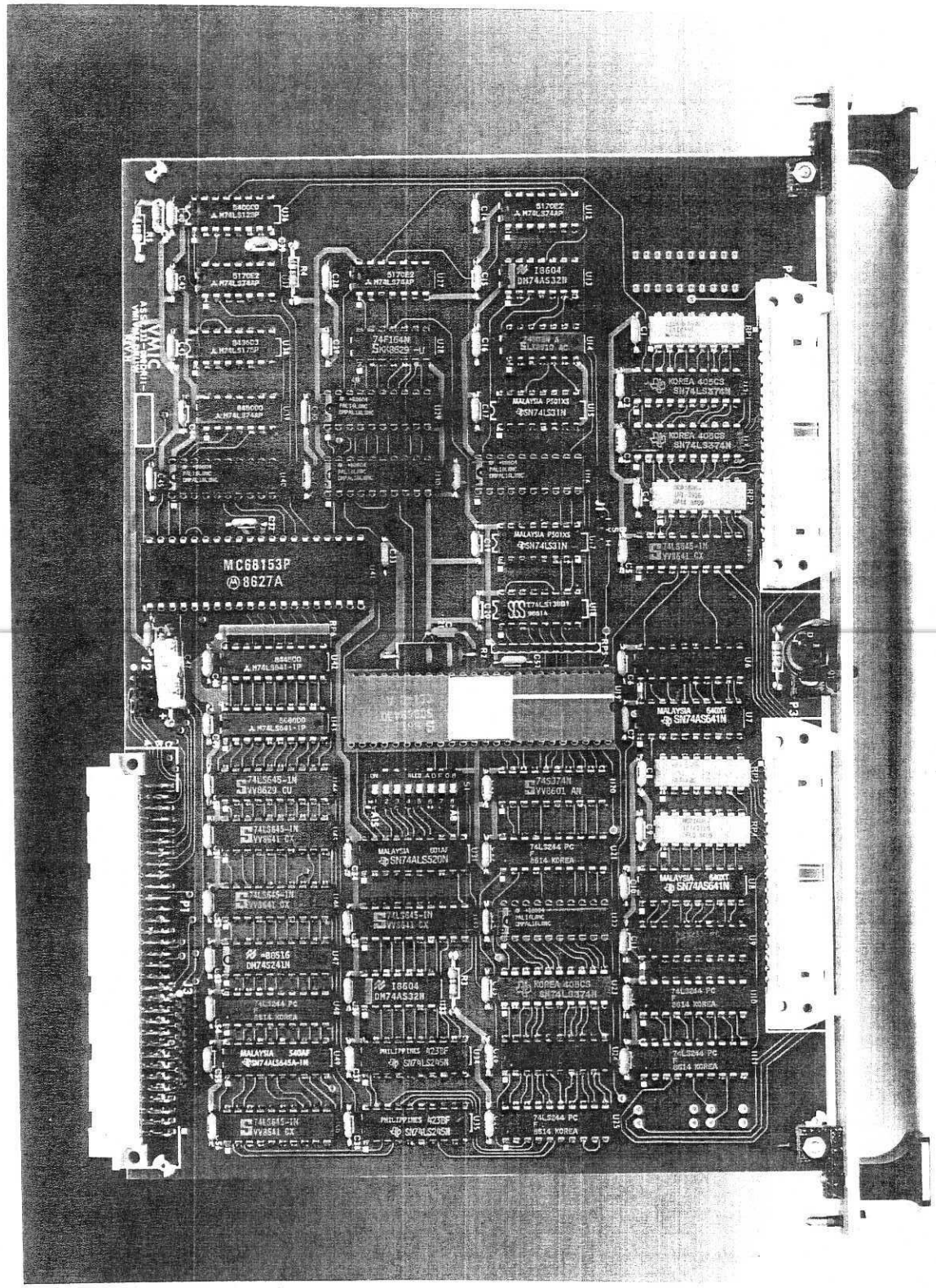
1.2 FUNCTIONAL DESCRIPTION

The VMIVME-DR11W provides a high performance, 16-bit DMA Link between the VMEbus and DEC DR11-W compatible devices. The VMIVME-DR11W interfaces may also be connected back-to-back to form a VMEbus-to-VMEbus DMA Link (see Figures 1.2-1 and 1.2-2).

The VMIVME-DR11W incorporates a SCB68430 DMA controller device to perform its DMA transfers, a MC68153 BIM IC to implement a VMEbus interrupter (for the attention and DMA completion interrupts), and it also contains logic to allow the VMIVME-DR11W to generate an attention interrupt to the external device.

1.3 COMPATIBILITY

VMIVME-DR11W compatible interfaces for the computers listed in Table 1.3-1 are available from the companies indicated for each computer.



MDR11/F1.1-1

Figure 1.1-1. Printed Circuit Assembly Photograph

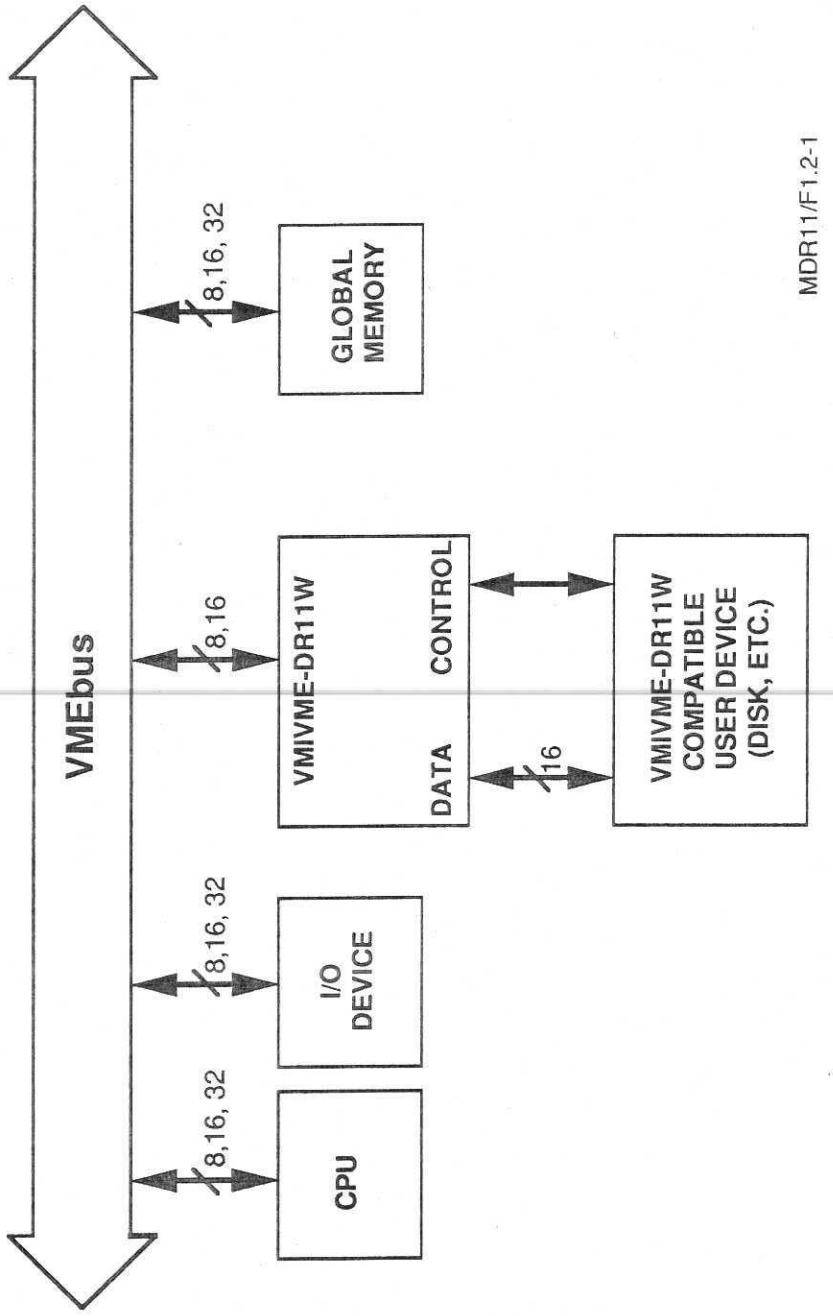
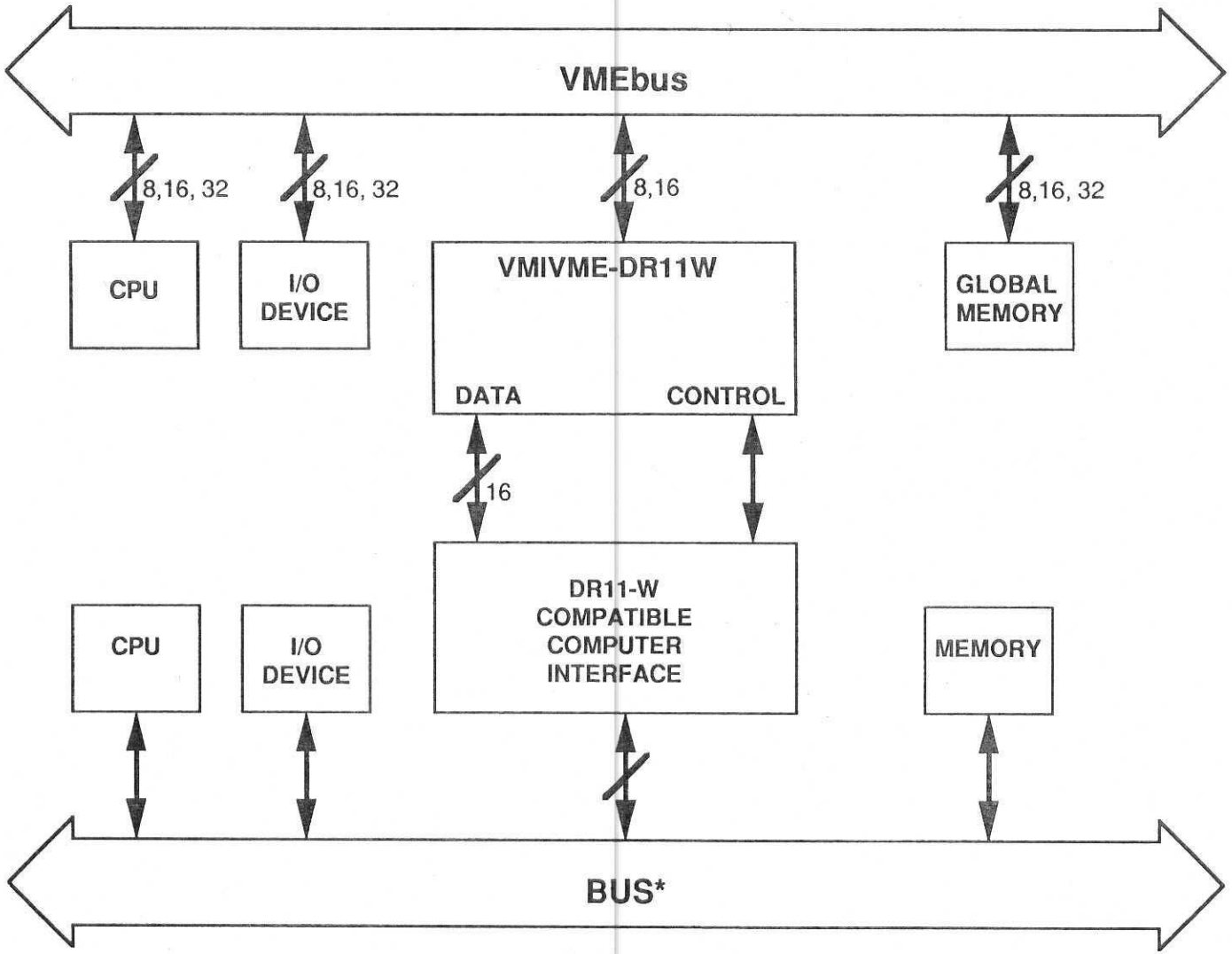


Figure 1.2-1. Connection-to-User Device



*DEC, Perkin-Elmer, Data General, or another VMEbus.

MDR11/F1.2-2

Figure 1.2-2. VMIVME-DR11W CPU-to-CPU Link

Table 1.3-1. VMIVME-DR11W Compatible Host Computer Interface Sources

COMPUTER	COMPANY	PHONE NUMBER
DEC VAX, PDP-11, LSI-11	Digital Equipment Corp.	1-800-DIGITAL Continental Blvd. Merrimack, NH 03054
CONCURRENT/ PERKIN-ELMER	Macrolink** Bill Goodale	(714) 634-8080
HARRIS H60-H100	Harris L. Dean Rice	(404) 256-4000
GOULD/SEL 32 Series*	Gould/Sel	(305) 587-2900
IBM PC/AT	Omnicomp Graphics*** Skip Parker	(713) 464-2990 Houston, TX

*Not directly VMIVME-DR11W compatible - requires simple modification of MPCI model 8020 IOP interface board (contact Gould for VMIVME-DR11W modification kit). VMIC has a 32-bit Gould HSD-to-VMEbus interface.

**Macrolink interfaces require a cable pinout reversal. When ordering cables specify the macrolink cables.

***Requires Omnicomp interprocessor link driver with soft start.

MDR11/T1.3-1

NOTICE

THE INFORMATION IN THIS TABLE IS OFFERED AS A HELPFUL SUGGESTION. IT IS INTENDED AS A GUIDE. VMIC DOES NOT GUARANTEE FAVORABLE RESULTS NOR DOES IT ASSUME ANY LIABILITY IN CONNECTION WITH ITS USE. NO STATEMENT OR RECOMMENDATION IS INTENDED BY VMIC FOR THE USE OF THESE PRODUCTS. BEFORE USING THE PRODUCTS LISTED, THE USER SHALL DETERMINE THE SUITABILITY OF THE PRODUCT(S) FOR HIS INTENDED USE, AND THE USER ASSUMES ALL RISK AND LIABILITY IN CONNECTION WITH THEIR USE.

1.4 REFERENCE MATERIAL LIST

The reader should refer to "The VMEbus Specification" for a detailed explanation of the VMEbus. "The VMEbus Specification" is available from the following source:

VITA
VMEbus International Trade Association
10229 N. Scottsdale Road
Scottsdale, AZ 85253
(602) 951-8866

Additional information concerning the VMIVME-DR11W protocol is contained in the "DR11-W Direct Memory Interface Module (No. EK-DR11W-UG-OO2)". This document is available from the following source:

Digital Equipment Corporation
146 Main St.
Maynard, MA 01754-2751
(617) 897-5111

Specific information concerning the SCB68430 may be obtained from the following source:

Signetics Corporation
811 E. Arques Ave.
P.O. Box 3409
Sunnyvale, CA 94088-3409
(408) 991-2000

A detailed specification is provided in Appendix B.

SECTION 2

PHYSICAL DESCRIPTION AND SPECIFICATIONS

REFER TO 800-00DR11-000 SPECIFICATION

SECTION 3

THEORY OF OPERATION

3.1 BLOCK DIAGRAMS

As shown in the functional block diagram (see Figure 3.1-1) and in the interconnection diagrams (Figures 3.1-2 and 3.1-3) the VMIVME-DR11W interface provides for the exchange of program controlled interrupts and status as well as two 16-bit data buses with handshake signals. The function of these interface signals are described in Section 3.4; whereas, the handshake sequencing of the signals are depicted in Figures 3.1-4, 3.1-5, and 3.1-6.

Although the VMIVME-DR11W interface was developed for interconnection to DEC DR11-W compatible devices, it also performs DMA transfers in conjunction with the DEC DRV11-WA interface (see Figure 3.1-3).

3.2 OPERATIONAL OVERVIEW

The VMIVME-DR11W has two modes of operation, INTERPROCESSOR LINK MODE (back-to-back configuration), and EXTERNAL DEVICE MODE. In the INTERPROCESSOR LINK MODE, the VMIVME-DR11W communicates with another DEC DR11-W compatible computer interface. In the EXTERNAL DEVICE MODE, the VMIVME-DR11W communicates with a user's device such as a disk drive system, tape drive, etc.

To operate in either mode, a transfer protocol (see Section 3.3) must be established. Once this is determined, the VMIVME-DR11W may be programmed as detailed in Section 4.1. One difference in the operation of this device versus other DEC DR11-W implementations is that the bus transfer type (word/byte and direction) is determined before the start of any data transfers and not on a cycle-by-cycle basis.

When two VMIVME-DR11W compatible boards are connected back-to-back, the two cables are crossed over so that one board drives a control signal and the other board receives it. Thus, there is no need for a hardware "master/slave mode".

3.3 TRANSFER PROTOCOL

In order for DMA transfers to take place, the DMA interfaces must be set up at both ends of the link. Therefore, if a CPU on one bus wants to transfer data, a

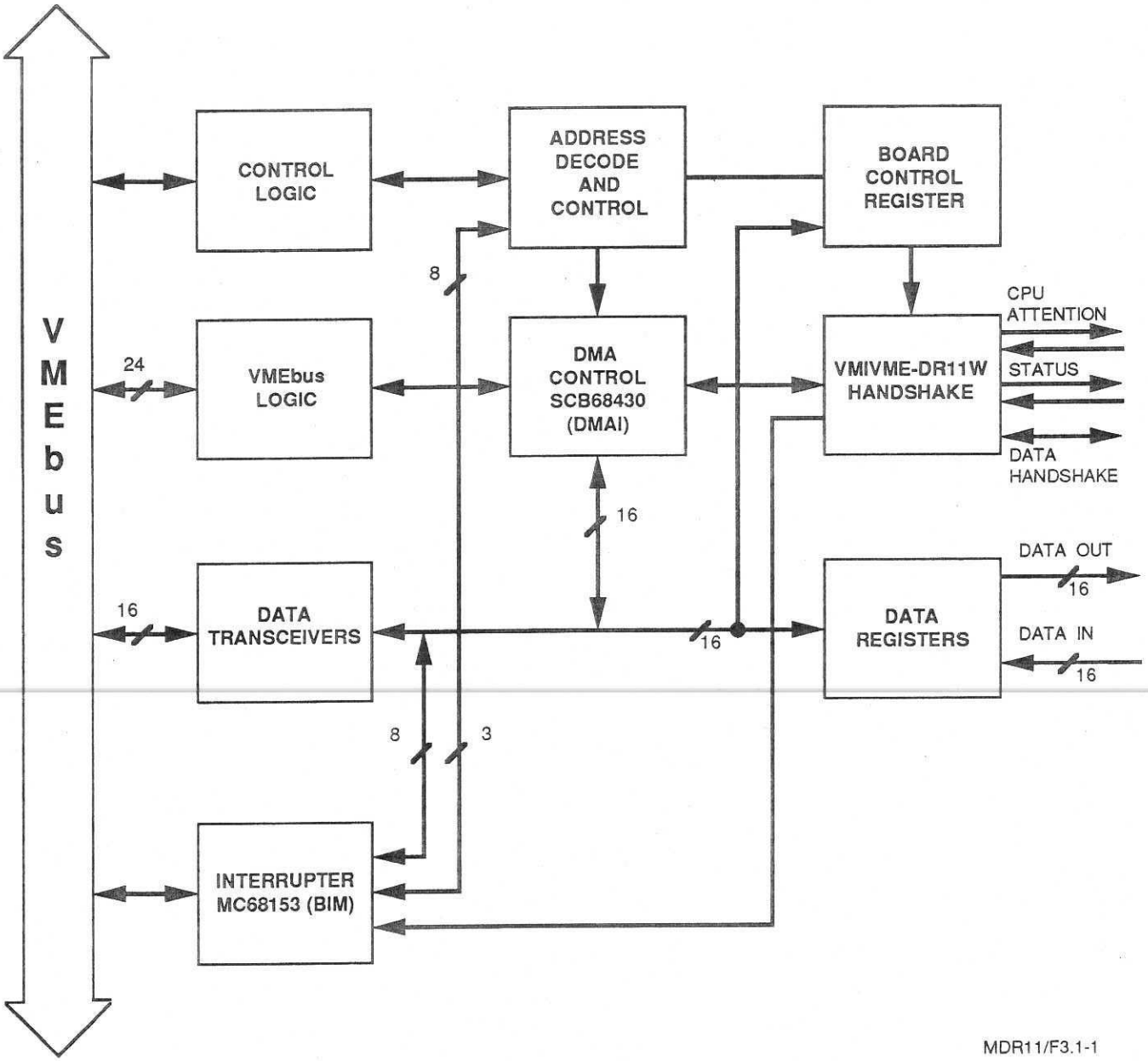


Figure 3.1-1. Functional Block Diagram

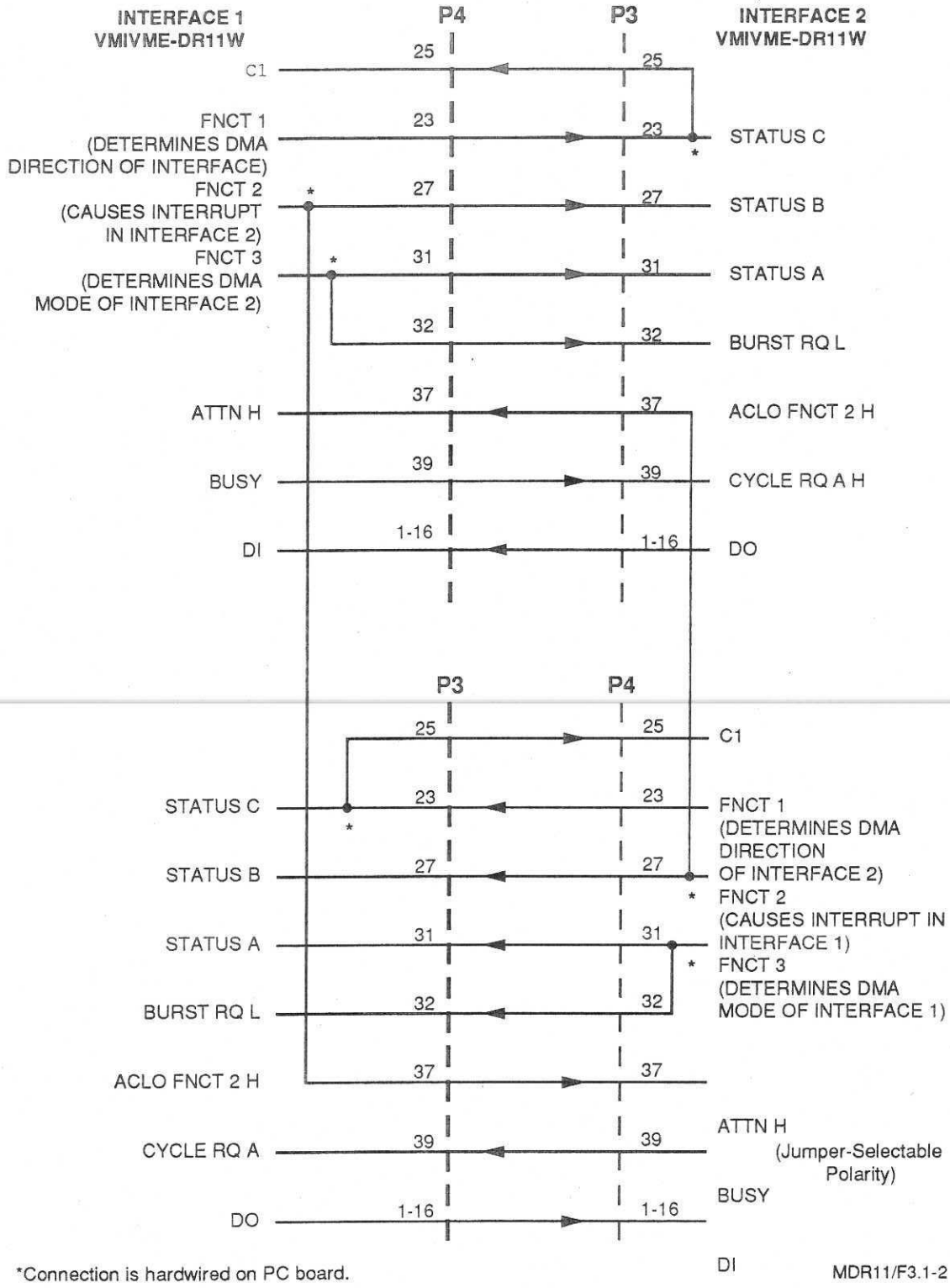
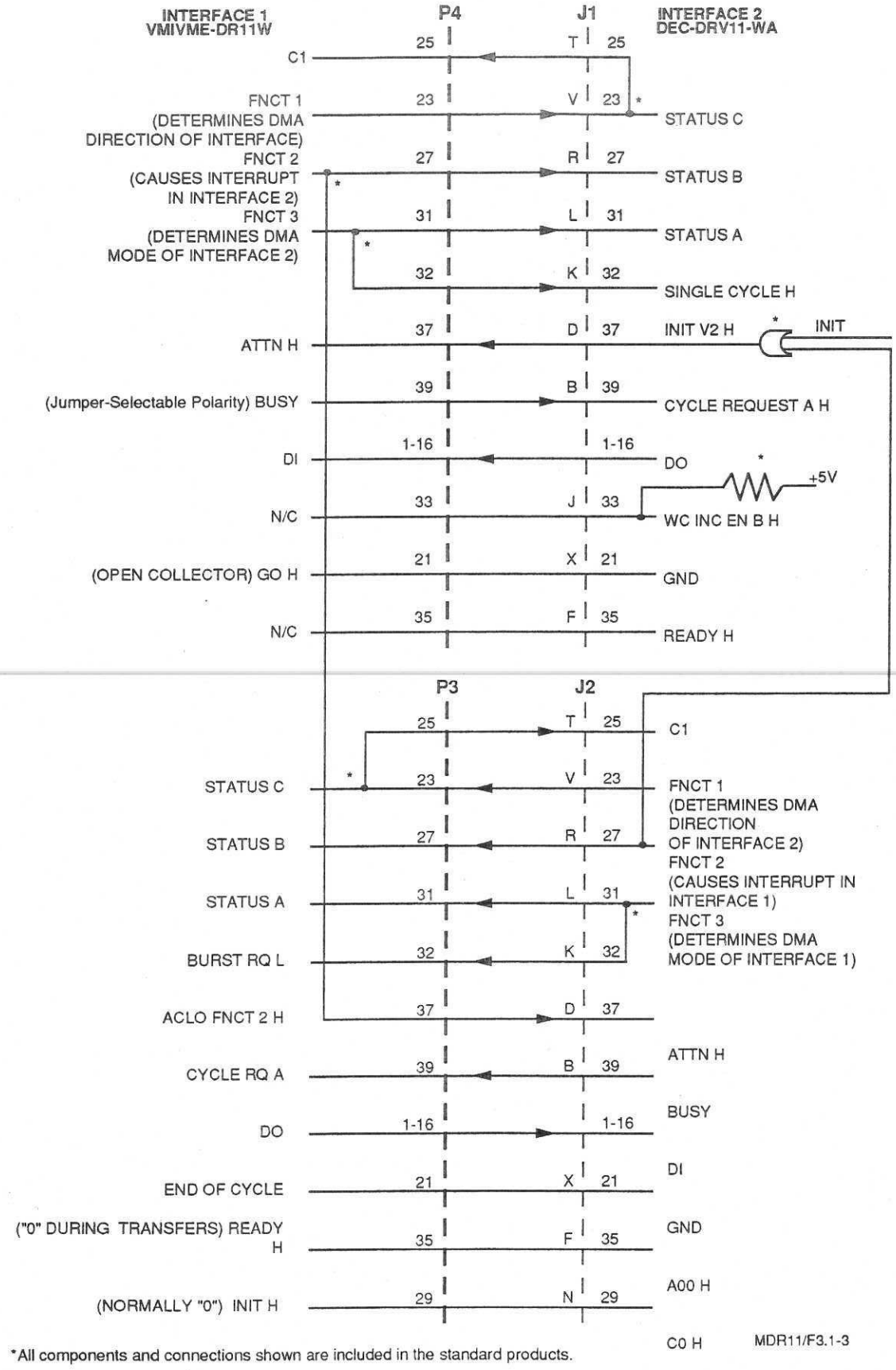


Figure 3.1-2. Interface Interconnection for Two DR11Ws in Back-to-Back Configuration



*All components and connections shown are included in the standard products.

Figure 3.1-3. Interface Connection to a DEC DRV11-WA

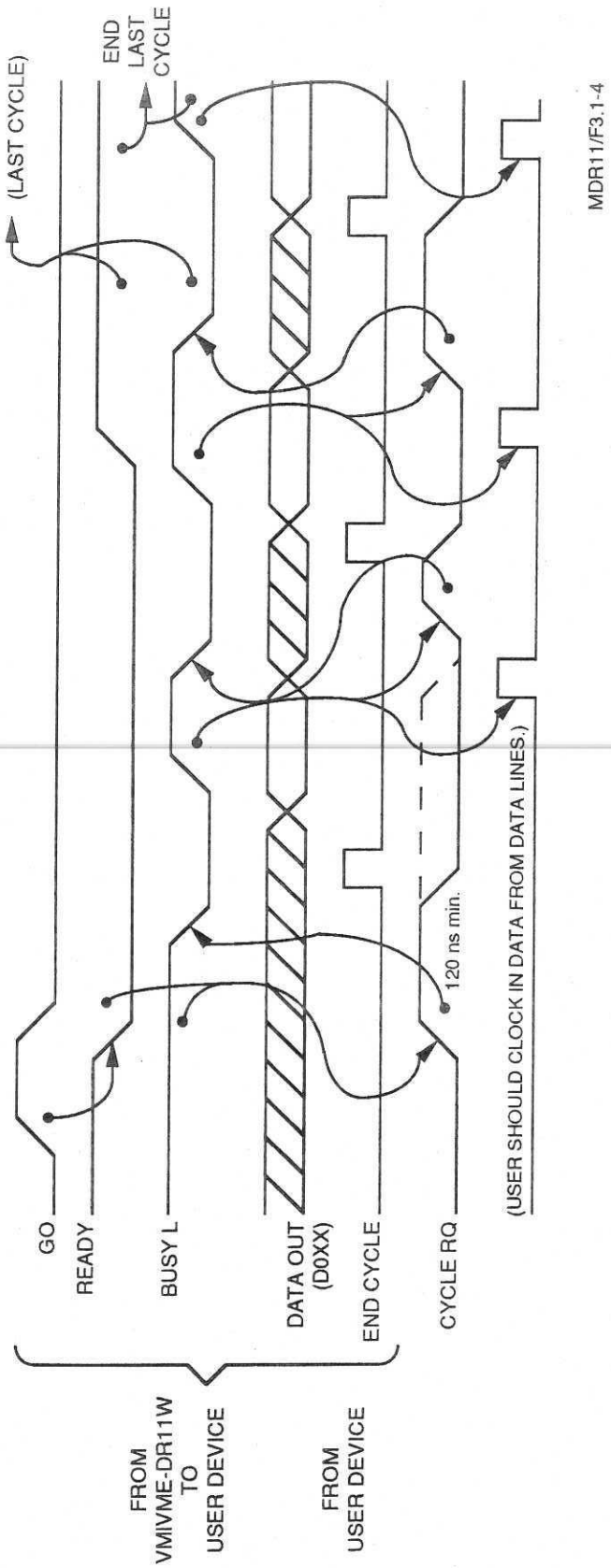


Figure 3.1-4. Timing/Handshake Sequencing Data Transfer from the VMIVME-DR11W to the User Device

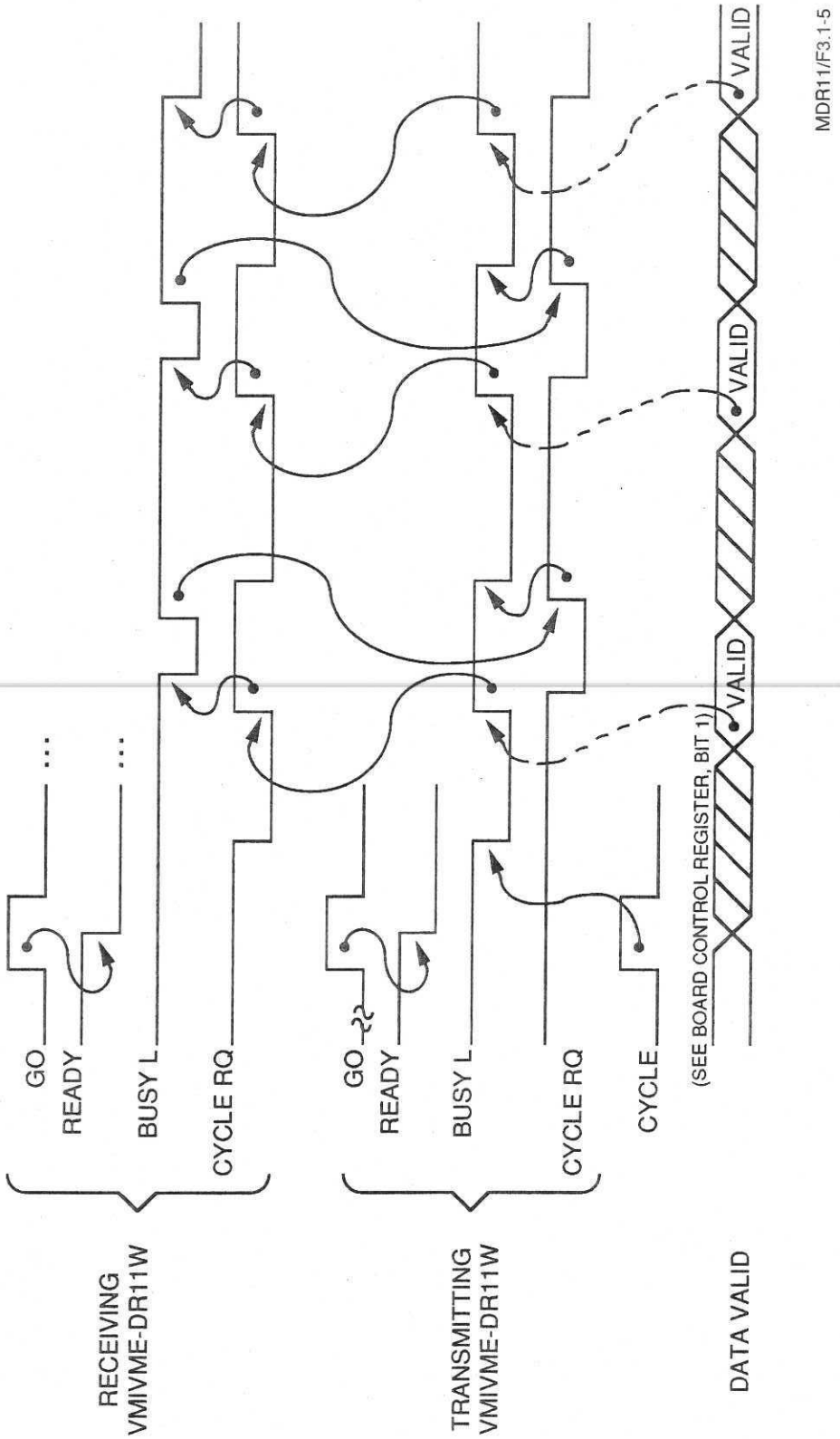


Figure 3.1-5. Back-to-Back Configuration (Interprocessor Mode)

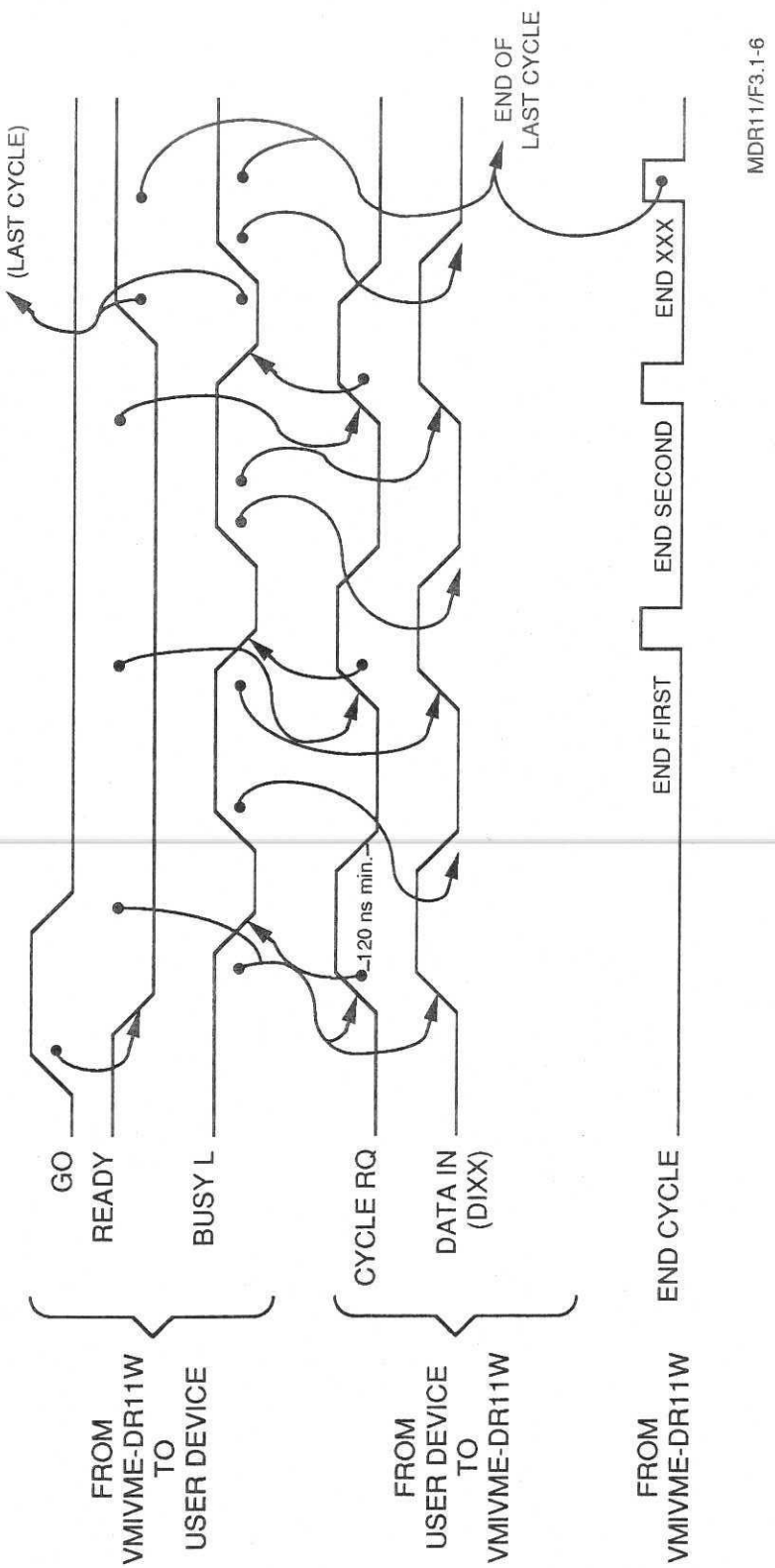


Figure 3.1-6. Diagram for Data Transfer from the User Device to the VMIVME-DR11W

second CPU on the second bus must set up the DMA interface on that second bus to the selected memory address, word count, and direction of transfer.

The sequence of operations necessary for two CPUs on two separate buses to control data transfers is referred to as the transfer protocol. There are several types of protocols that may be used with the VMIVME-DR11W.

One protocol type consists of the following sequences:

a. The CPU in bus one interrupts the CPU on the other bus via the "ATTENTION" interrupt.

b. Both DMA interfaces are set up to transfer a control data block. The control data block contains information required by the second system to set up the other VMIVME-DR11W (such as transfer counts, transfer direction, etc.)

c. The control data block is then used to set up the DMA interface in the second bus. This protocol preselects the size and direction of transfer of the control block.

A second protocol type consists of using predetermined transfer block sizes and sequences.

A third protocol type consists of sending the DMA set-up data via a serial communication link.

3.4 DATA TRANSFER DESCRIPTION

When the VMIVME-DR11W transfers individual data words to or from an external device, it becomes the VMEbus master and transfers the data via DMA transfers to or from memory. It asserts the VMEbus address, control lines, address modifier, and sends or receives data. The data transfers are controlled by a SCB68430 DMA interface chip. Handshake signals to/from the SCB68430 are converted to/from VMIVME-DR11W compatible handshake signals.

3.5 VME-TO-VME LINK

Two VMIVME-DR11W Boards may be connected back-to-back to create a DMA link between two VMEbus chassis assemblies. In this configuration, the two boards are connected so that their data buses and control signals are cross coupled which allows them to transfer data back and forth under handshake control. The cross coupling is shown in Figure 3.1-2.

3.6 USER INTERFACE SIGNAL FUNCTION DESCRIPTION

The following paragraphs describe the standard VMIVME-DR11W signals used by the VMIVME-DR11W.

DO00 to DO15 - DATA OUTPUT BUS. The data is positive true and is transmitted by bus transceivers. Each line is terminated by a 180/390 Ω network.

CYCLE REQUEST A - CYCLE REQUEST B. These two inputs are used to request a DMA cycle from the user's device. These two signals are "ORed" together.

END CYCLE. This signal indicates the end of a DMA cycle to the user's device.

STATUS A - STATUS BIT A. This bit is a user-defined status bit from the user's device to the VMIVME-DR11W. In the back-to-back configuration, this bit indicates that single cycle transfers should be performed when it is asserted.

STATUS B - STATUS BIT B. This bit is a user-defined status bit from the user's device to the VMIVME-DR11W. In the back-to-back configuration, STATUS B indicates the state of the ATTN interrupt bit.

STATUS C - STATUS BIT C. This signal is a user-defined status bit from the user's device to the VMIVME-DR11W. In the back-to-back configuration, STATUS C indicates the direction of the DMA transfer.

INIT - INITIALIZE. This signal is output from the VMIVME-DR11W to indicate a power ON reset or other reset has occurred.

BURST RQ - BURST REQUEST. This signal is an input to the VMIVME-DR11W and when asserted, indicates that burst transfers should be performed. This bit is negative true.

READY. This output signal indicates that the board is ready and does not have transfers in progress. This signal is deasserted by GO.

ACLO FUNCT2 - AC LOW. OR FUNCTION 2. This bit is the same as Function 2 on the VMIVME-DR11W.

DI00 to DI15 - DATA INPUT BUS. The data is positive true and is received by bus transceivers. Each line is terminated by a 180/390 Ω network.

GO. This bit is output by the VMIVME-DR11W to indicate the start of a DMA transfer block.

FNCT1 - FUNCTION 1. Board Control Register Bit 05 (user-defined) is output by the VMIVME-DR11W. In the back-to-back configuration this bit controls the DMA direction of the VMIVME-DR11W and drives C1.

FNCT2 - FUNCTION 2. Board Control Register Bit 06 (user-defined) is output from the VMIVME-DR11W. In the back-to-back configuration, this bit drives STATUS B (and ATTN) on the other board.

FNCT3 - FUNCTION 3. Board Control Register Bit 07 (user-defined) is output from the VMIVME-DR11W. In the back-to-back configuration, this bit drives STATUS A.

C1 - CONTROL 1. This signal is an input to the VMIVME-DR11W and controls the direction of DMA transfer. In the back-to-back configuration, it is driven by FNCT1. A logic "one" on C1 causes the VMIVME-DR11W to receive data while a logic "zero" causes it to transmit.

ATTN - ATTENTION. This signal is an input to the VMIVME-DR11W. It indicates that the user device needs attention and causes an interrupt to the VMEbus system on the rising edge.

BUSY. This signal is an output from the VMIVME-DR11W and indicates that a VMEbus cycle is in progress. The polarity of this signal is jumper-selectable (Busy H or Busy L). In the back-to-back mode, Busy L must be selected since it is connected to the cycle request on the other VMIVME-DR11W Board.

OTHER BOARD READY. (-100 Option Only). This signal is an input to the VMIVME-DR11W-100 that receives the other board's Ready signal.

SECTION 4

PROGRAMMING

4.1 PROGRAMMING OVERVIEW

The VMIVME-DR11W is designed with three types of devices that must be programmed for proper operation. These are the DMA controller chip (SCB68430), the Bus Interrupter Module (MC68153), and on-board registers. The on-board registers are used to provide the address modifier, to allow control of the DMA controller and the BIM, and to communicate with the external VMIVME-DR11W compatible device. Other programming aspects of the VMIVME-DR11W include executing the transfer protocol (see Section 3.3), handling interrupts, and handling errors. See Table 4.1-1 for names and addresses or registers utilized by the VMIVME-DR11W. Data sheets for the MC68153 and the SCB68430 are located in Appendix B.

4.2 REGISTER BIT DEFINITIONS

For Register Bit Definitions see Tables 4.2-1 through 4.2-3.

4.3 SEQUENCE OF REGISTER LOADING

The receiving VMIVME-DR11W interface must be set up before the transmitting interface to avoid losing the first word transferred.

To transfer a block of data, the programming sequence of the VMIVME-DR11W should be as follows:

a. Load the registers in the MC68153 and SCB68430.

b. The Channel Control Register (CCR) should be the last register loaded in the 68430.

c. Load the Address Modifier Register (AMR)(AMR - on-board).

d. Load the Board Contol Register (BCR)(BCR - on-board) with the GO and CYCLE bits (Bits 0 and 1) set to "zero".

e. Reload the BCR with the GO and CYCLE bits set to "one".

Table 4.1-1. VMIVME-DR11W Register Address Map

ADDRESS	ACRONYM	REGISTER NAME	PHYSICAL LOCATION
00	CSR	Channel Status Register	SCB68430 DMAI
01	CER	Channel Error Register	SCB68430 DMAI
04	DCR	Device Control Register	SCB68430 DMAI
05	OCR	Operation Control Register	SCB68430 DMAI
06	SCR*	Sequence Control Register	SCB68430 DMAI
07	CCR	Channel Control Register	SCB68430 DMAI
0A	MTCH	Memory Transfer Count High	SCB68430 DMAI
0B	MTCL	Memory Transfer Count Low	SCB68430 DMAI
0C	MACH**	Memory Address Count High	SCB68430 DMAI
0D	MACMH	Memory Address Count Mid-High	SCB68430 DMAI
0E	MACML	Memory Address Count Mid-Low	SCB68430 DMAI
0F	MACL	Memory Address Count Low	SCB68430 DMAI
25	DIVR	DMA Interrupt Vector Register	SCB68430 DMAI
27	DIVR***	DMA Interrupt Vector Register	SCB68430 DMAI
2D	CPR*	Channel Priority Register	SCB68430 DMAI
41	CR0**	Channel Register 0	MC68153 BIM
43	BERRICR	BERR Interrupt Control Register	MC68153 BIM
45	AICR (CR2)	Attention Interrupt Control Register	MC68153 BIM
47	DICR (CR3)	DMA Interrupt Control Register	MC68153 BIM
49	VR0**	Vector Register 0	MC68153 BIM
4B	BERRIVR	BERR Interrupt Vector Register	MC68153 BIM
4D	AIVR (CR3)	Attention Interrupt Vector Register	MC68153 BIM
4F	VR3**	Vector Register 3	MC68153 BIM
60	BCRH	Board Control Register HIGH	(On-Board)
61	BCRL	Board Control Register LOW	(On-Board)
62	IODRH	I/O Data Register HIGH	(On-Board)
63	IODRL	I/O Data Register LOW	(On-Board)
65	AMR	Address Modifier Register	(On-Board)

*Included for software compatibility.

**Register is not used.

***The IVR has two addresses for software compatibility (see specification sheet).

MDR11/T4.1-1

Table 4.2-1. Register Bit Definitions

DMA INTERFACE REGISTER BIT DEFINITIONS (DMAI, SIGNETICS NO. SCB68430)

CHANNEL STATUS REGISTER (ADDRESS: $XX00)

CSR

BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8
CHANNEL OPERATION COMPLETE	NOT USED	NORMAL DEVICE TERMINATE	ERROR	CHANNEL ACTIVE	NOT USED	NOT USED	READY INPUT* STATE
0=NO 1=YES	(0)	0=NO 1=YES	0=NO 1=YES	0=NO 1=YES	(0)	(0)	0=LOW 1=HIGH

* This bit monitors the status of the DMAI Ready input pin and in no way reflects the status of the VMIVME-DR11W's ready output pin. A logic "zero" indicates that a DMA cycle request has been received.

CHANNEL ERROR REGISTER ($XX01)

CER

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
NOT USED	NOT USED	NOT USED	ERROR CODE				
(0)	(0)	(0)	00000=NO ERROR 01001=BUS ERROR 10001=SOFTWARE ABORT				

DEVICE CONTROL REGISTER ($XX04)

DCR

BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8
EXTERNAL REQUEST MODE	NOT USED	NOT USED	NOT USED	NOT USED	NOT USED	NOT USED	NOT USED
0=BURST 1=CYCLE STEAL	(0)	(1)	(1)	*	(0)	(0)	(0)

*Should be programmed as a "zero" if the operand size (see Operation Control Register, Bits 4 and 5) is BYTE. It should be programmed as a "one" otherwise.

OPERATION CONTROL REGISTER ($XX05)

OCR

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
DIRECTION	NOT USED	OPERAND SIZE		NOT USED	NOT USED	NOT USED	NOT USED
0=MEM TO DEV 1=DEV TO MEM	(0)	00=BYTE 01=WORD 10=LONGWORD* 11=32-bit WORD*		(0)	(0)	(1)	(0)

*Not applicable to the VMIVME-DR11W.

MDR11/T4.2-1/1

Table 4.2-1. Register Bit Definitions (Continued)

SEQUENCE CONTROL REGISTER ($XX06)

	BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8
SCR**	NOT USED*							
	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)

*Dummy Register.

CHANNEL CONTROL REGISTER ($XX07)

	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
CCR	START	NOT USED	NOT USED	SOFTWARE ABORT	INTERRUPT ENABLE	NOT USED	NOT USED	NOT USED
	0=NO 1=YES	(0)	(0)	0=NO 1=YES	0=NO 1=YES	(0)	(0)	(0)

MEMORY TRANSFER COUNT HIGH ($XX0A)

	BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8
MTCH	MEMORY TRANSFER COUNT MSB							
	BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8

MEMORY TRANSFER COUNT LOW ($XX0B)

	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
MTCL	MEMORY TRANSFER COUNT LSB							
	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0

MDR11/T4.2-1/2

Table 4.2-1. Register Bit Definitions (Continued)

MEMORY ADDRESS COUNT HIGH ($XX0C)

MACH

BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8
NOT USED*							
(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)

*Dummy Register.

MEMORY ADDRESS COUNT MID-HIGH ($XX0D)

MACMH

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
MEMORY ADDRESS COUNTER							
BIT 23	BIT 22	BIT 21	BIT 20	BIT 19	BIT 18	BIT 17	BIT 16

MEMORY ADDRESS COUNT MID-LOW ($XX0E)

MACML

BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8
MEMORY ADDRESS COUNTER							
BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8

MEMORY ADDRESS COUNT LOW ($XX0F)

MACL

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
MEMORY ADDRESS COUNTER							
BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0

MDR11/T4.2-1/3

Table 4.2-1. Register Bit Definitions (Concluded)

DMA INTERRUPT VECTOR REGISTER ($XX25, $XX27)

	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
DIVR	DONE INTERRUPT VECTOR							
	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	NOTE 1

NOTE 1: This bit is automatically set if an error occurred. This register is mapped into two locations to provide compatibility with other controllers.

CHANNEL PRIORITY REGISTER ($XX2D)

	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
CPR	NOT USED*							
	(0)	(0)	(0)	(0)	(0)	(0)	(0)	(0)

*Dummy Register.

MDR11/T4.2-1/4

Table 4.2-2. Bus Interrupter Module No. MC68153 Register Bit Definitions

BERR INTERRUPT CONTROL REGISTER ($XX43)

	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
BERRICR	FLAG BIT	FLAG BIT AUTO CLEAR	VECTOR 0=INT 1=EXT	INTERRUPT ENABLE	INTERRUPT AUTO CLEAR	IRQ LEVEL 2	IRQ LEVEL 1	IRQ LEVEL 0

ATTENTION INTERRUPT CONTROL REGISTER ($XX45)

	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
AICR	FLAG BIT	FLAG BIT AUTO CLEAR	VECTOR 0=INT 1=EXT	INTERRUPT ENABLE	INTERRUPT AUTO CLEAR	IRQ LEVEL 2	IRQ LEVEL 1	IRQ LEVEL 0

DMA INTERRUPT CONTROL REGISTER ($XX47)

	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
DICR	FLAG BIT	FLAG BIT AUTO CLEAR	VECTOR 0=INT 1=EXT	INTERRUPT ENABLE	INTERRUPT AUTO CLEAR	IRQ LEVEL 2	IRQ LEVEL 1	IRQ LEVEL 0

BERR INTERRUPT VECTOR REGISTER ($XX4B)

	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
BERRIVR	BUS ERROR VECTOR							
	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0

MDR11/T4.2-2/1

Table 4.2-2. Bus Interrupter Module No. MC68153 Register Bit Definitions (Concluded)

ATTENTION INTERRUPT VECTOR REGISTER ($XX4D)

	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
AIVR	ATTENTION INTERRUPT VECTOR							
	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0

MDR11/T4.2-2/2

Table 4.2-3. Discrete On-Board Register Bit Formats

BOARD CONTROL REGISTER HIGH ($XX60) - STATUS

BCRH

BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8
STATUS C (READ ONLY)	STATUS B (READ ONLY)	STATUS A (READ ONLY)	BURST R1 (READ ONLY)	C1 (READ ONLY)	ATTN (READ ONLY)	RESERVED EXCEPT-100 OPTION **	RESERVED
						(0)	(0)

BOARD CONTROL REGISTER LOW ($XX61) - COMMAND*

BCRL

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
FNCT 3 (R/W)	FNCT 2 (R/W)	FNCT 1 (R/W)	RESERVE (R/W)	LED CONTROL 0=ON 1=OFF	ENABLE ATTN INTERRUPT (R/W)	CYCLE (WRITE ONLY)	GO (WRITE ONLY)

*Do not use a read-modify-write instruction (e.g., bit clear) when writing to the BCR.

I/O DATA REGISTER HIGH ($XX62)*

IODRH

BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8
I/O DATA REGISTER HIGH							
BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8

I/O DATA REGISTER LOW ($XX63)*

IODRL

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
I/O DATA REGISTER LOW							
BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0

*When writing to IODRH and IODRL, only word write operations should be used.

**For -100 option, Bit 9 is Other Board Ready (state of other board's GO bit).

MDR11/T4.2-3/1

Table 4.2-3. Discrete On-Board Register Bit Formats (Concluded)

ADDRESS MODIFIER REGISTER ($XX65) (WRITE ONLY)

	NOT USED	NOT USED	ADDRESS MODIFIER					
AMR			AM 5	AM 4	AM 3	AM 2	AM 1	AM 0
	(0)	(0)						

MDR11/T4.2-3/2

Note that the start bit (CCR Bit 07) must be set inside the SCB68430 before the GO and CYCLE bits are set in the BCR. The start bit in the SCB68430 is independent of the CYCLE (start) bit in the BCR. The C1 interface signal input (BCR - Bit 10) controls the direction of the I/O Data Register, while the direction bit in the SCB68430 (OCR - Bit 07) controls the transfer direction of the SCB68430. These two bits must be in the same logic state for correct operation.

An example of register initialization sequence for transmitting data (from memory) is shown in Table 4.3-1. After the first DMA transfer is completed, the VMIVME-DR11W can be reinitialized with only seven steps to transmit a second block of data (see Table 4.3-2).

An example of a register initialization sequence for receiving data (transfer to memory) is shown in Table 4.3-3. After the first block transfer is completed, the VMIVME-DR11W can be reinitialized with only seven steps to receive a second block of data (see Table 4.3-4).

4.4 PROGRAMMING THE SCB68430 DMAI

4.4.1 Channel Status Register/Channel Error Register

These two registers are used together to determine status and error conditions related to the DMA controller.

4.4.1.1 Channel Status Register (CSR)

BIT 15 - CHANNEL OPERATION COMPLETE*. This bit is set when the DMA operation is completed. If interrupts are enabled, an interrupt will be generated at this time.

BIT 14. Not used.

BIT 13 - NORMAL DEVICE TERMINATE - NOT USED.

BIT 12 - ERROR*. This bit will be set if the DMA operation was terminated due to an error condition.

*CSR BIT 15 and CSR BIT 12 must be cleared if set, before another DMA operation can be started. These bits may be cleared by writing a "one" to each bit that is set. The following 68000 instruction will do this without testing each of the bits.

```
MOVE.B    CSR, CSR
```

The other bits in the CSR are unaffected by a write.

Table 4.3-1. Register Initialization Sequence for Transmitting a 4 Kword Block Starting Data Address $40000

STEP	REGISTER LOCATION	REGISTER NAME	REGISTER ADDRESS OFFSET	DATA LOADED (HEXADECIMAL)	COMMENT
1	SCB68430	DCR	04	B8	Cycle Steal
2	SCB68430	OCR	05	12	Word from memory
3	SCB68430	MTCH	0A	1000	16-bit transfer count
4	SCB68430	MACMH	0D	04	Upper 8 bits of memory address
5	SCB68430	MACML	0E	0000	Lower 16 bits of memory address
6	SCB68430	DIVR	25	42	Done interrupt vector
7	MC68153	AIVR	4D	40	Attention interrupt vector
8	MC68153	AICR	45	1F	Enable internal vector with auto clear
9	MC68153	DICR	47	3F	Enable external vector with auto clear
10	On-Board	AMR	65	39	ADDR MOD=39
11	SCB68430	CSR	00	B9	Clear Channel Status Register
12	SCB68430	CCR	07	88	Start DMA chip
13	On-Board	BCR	61	8F	Go, enable ATTN, LED OFF

MDR11/T4.3-1

Table 4.3-2. Register Initialization Sequence for Transmitting a Second 4 Kword Block Starting Data Address $20000

STEP	REGISTER LOCATION	REGISTER NAME	REGISTER ADDRESS OFFSET	DATA LOADED (HEXADECIMAL)	COMMENT
1	SCB68430	MTCH	0A	1000	16-bit transfer count
2	SCB68430	MACMH	0D	02	Upper 8 bits of memory address
3	SCB68430	MACML	0E	0000	Lower 16 bits of memory address
4	MC68153	DICR	47	3F	Enable interrupts with auto clear
5	SCB68430	CSR	00	B9	Clear Channel Status Register
6	SCB68430	CCR	07	88	Start DMA chip
7	On-Board	BCR	61	8F	Go, enable ATTN, LED OFF

MDR11/T4.3-2

Table 4.3-3. Register Initialization Sequence for Receiving a 4 Kword Block Starting Data Address $40000

STEP	REGISTER LOCATION	REGISTER NAME	REGISTER ADDRESS OFFSET	DATA LOADED (HEXADECIMAL)	COMMENT
1	SCB68430	DCR	04	B8	Cycle Steal
2	SCB68430	OCR	05	92	Word to memory
3	SCB68430	MTCH	0A	1000	16-bit transfer count
4	SCB68430	MACMH	0D	04	Upper 8 bits of memory address
5	SCB68430	MACML	0E	0000	Lower 16 bits of memory address
6	SCB68430	DIVR	25	42	Done interrupt vector
7	MC68153	AIVR	4D	40	Attention interrupt vector
8	MC68153	AICR	45	1F	Enable internal vector with auto clear
9	MC68153	DICR	47	3F	Enable external vector with auto clear
10	On-Board	AMR	65	39	ADDR MOD=39
11	SCB68430	CSR	00	B9	Clear Channel Status Register
12	SCB68430	CCR	07	88	Start DMA chip
13	On-Board	BCR	61	CC	Go=0, enable ATTN, LED OFF, receive
14	On-Board	BCR	61	AD	Go, enable ATTN, LED OFF, receive

MDR11/T4.3-3

Table 4.3-4. Register Initialization Sequence for Receiving a Second 4 Kword Block Starting Data Address $20000

STEP	REGISTER LOCATION	REGISTER NAME	REGISTER ADDRESS OFFSET	DATA LOADED (HEXADECIMAL)	COMMENT
1	SCB68430	MTCH	0A	1000	16-bit transfer count
2	SCB68430	MACMH	0D	02	Upper 8 bits of memory address
3	SCB68430	MACML	0E	0000	Lower 16 bits of memory address
4	MC68153	DICR	47	3F	Enable interrupts with auto clear
5	SCB68430	CSR	00	B9	Clear Channel Status Register
6	SCB68430	CCR	07	88	Start DMA chip
7	On-Board	BCR	61	AD	Go, enable ATTN, LED OFF, receive

MDR11/T4.3-4

BIT 11 - CHANNEL ACTIVE. This bit indicates that a DMA operation is in progress and is automatically cleared upon termination of the operation.

BITS 10, 9. Not used.

BIT 8 - READY INPUT. Indicates the state of the RDY input signal, i.e., a "zero" indicates zero volts and that data is ready.

4.4.1.2 Channel Error Register (CER)

BITS 7, 6, 5. Not used.

BITS 4, 3, 2, 1, 0 - ERROR CODE. These 5 bits indicate the error type when an error occurs. This register is cleared when the error bit (CSR BIT 12) is cleared.

ERROR CODE	ERROR TYPE
00000	No Error
01001	Bus Error
10001	Software Abort

4.4.2 Device Control Register/Operation Control Register

These two registers control the data direction, data size, and request mode of the DMA transfers.

4.4.2.1 Device Control Register (DCR)

BIT 15 - EXTERNAL REQUEST MODE. This bit is set to perform single cycle transfers and cleared to perform burst transfers.

BITS 14, 13, 12. Not used.

BIT 11 - SIZE. This bit is read and written as the "Logical OR" of bits 4 and 5 in the OCR.

BITS 10, 9, 8. Not used.

4.4.2.2 Operation Control Register (OCR)

BIT 7 - DIRECTION. This bit contains the DMA transfer direction.

0 = Read from Memory
Write to External Device
1 = Write to Memory
Read from External Device

BIT 6. Not used.

BITS 5, 4 - OPERAND SIZE. These bits control the operand size.

BIT 5	BIT 4	OPERAND SIZE
0	0	BYTE
0	1	WORD

BITS 3, 2, 1, 0. Not used.

4.4.3 Sequence Control Register (SCR)

The SCR is a dummy register provided for compatibility with other DMA controllers.

BITS 15 - 8. Not used (dummy register).

4.4.4 Channel Control Register (CCR)

BIT 7 - START. This bit causes the DMA controller to start its operation. This bit must be set before the GO bit is set in the Board Control Register (BCR).

BITS 6, 5. Not used.

BIT 4 - SOFTWARE ABORT. This bit (when set to "one") aborts the current DMA operation. It sets the error bit in the CSR and the error code in the CER. The software abort bit must be cleared to clear the error bit in the CSR and to clear the error code in the CER.

SOFTWARE ABORT PROCEDURE:

To abort an in-process DMA, perform the following steps:

STEP 1: Set BIT 4 of the Channel Control Register (CCR, $0007) to a logic "one". Maintain BITS 3 and 7 according to the desired control state.

STEP 2: Read the Channel Status Register (CSR, $0000) until BIT 15 (operation complete) and BIT 12 (error) are both equal to a logic "one", indicating a DMA complete with error.

STEP 3: Read the Channel Error Register to obtain the error condition for the DMA complete with error. Reset the software abort bit (BIT 4), (CCR, $0007) to zero.

NOTE: The Channel Error Register must be read while the abort bit (BIT 4, CCR, $0007) is active (set) or the Channel Error Register will indicate the default error status (bus error).

STEP 4: Perform normal DMA termination processing clean-up, resetting all of the CSR status bits to logic "zero". (CSR bit reset requires writing a logic "one" to each bit which is set to a logic "one".)

NOTE

USE OF "BURST" MODE PRECLUDES THE USE OF SOFTWARE ABORT, SINCE THE BUS WILL BE RETAINED BY THE VMIVME-DR11W UNTIL THE DMA IS COMPLETED ONCE IT HAS BEEN INITIATED.

BIT 3 - INTERRUPT ENABLE. This bit enables the DMA interrupt request when a DMA operation is completed.

BITS 2, 1, 0. Not used.

4.4.5 Memory Transfer Count Registers

These registers contain the number of desired transfers for the current operation. The count is a 16-bit integer, allowing transfers of up to 64 K 16-bit words in a single transfer block.

4.4.6 Memory Address Count Registers

These four registers contain the memory address for the DMA operation. The MACH Register is a dummy register and is provided for compatibility with other DMA controllers. The three implemented registers provide a 24-bit memory address for the DMA transfers. These registers should be loaded with the starting memory address before an operation is started. The registers are incremented during operation and may be read while an operation is in process.

4.4.7 DMA Interrupt Vector Register

This register is used to store the interrupt vector used for the DMA complete interrupt and is resident inside the SCB68430. It is mapped to two locations to provide compatibility with other DMA controllers. Also, note that Bit 0 is automatically set when an error occurs and cannot be written. To generate a bus error interrupt, the Bus Error Interrupt Control Register and the Bus Error Interrupt Vector Register must be programmed.

Thus, a DMA interrupt vector for normal conditions must be even and the error vector is automatically the next higher odd vector. Note that although the SCB68430 supplies the interrupt vector, the MC68153 provides the interrupt handshake; therefore, the MC68153 must be programmed to handle the external vector from the SCB68430 (i.e., BIT 5 of the DICR must be set to a "one").

4.4.8 Channel Priority Register

This register is a dummy register provided for compatibility with other DMA controllers.

4.5 PROGRAMMING THE MC68153 BIM

The MC68153 contains eight registers, but only five are used in the VMIVME-DR11W. Three of these registers are Control Registers and two are Vector Registers. Data sheets for the MC68153 are provided in Appendix B.

4.5.1 Bus Error Interrupt Control Register (BERRICR), Attention Interrupt Control Register (AICR), and DMA Interrupt Control Register (DICR)

BITS 7, 6 - FLAG CONTROL - NOT USED. These two bits do not affect operation of the MC68153.

BIT 5 - VECTOR LOCATION. The user should clear this bit to a logic "zero" to cause the MC68153 to use its Internal Vector Register for an interrupt acknowledge and it should be set to a logic "one" to use an external interrupt vector. For the VMIVME-DR11W, this bit should be cleared in the BERRICR, AICR, and set in the DICR since an external vector is obtained from the SCB68430 DMA Controller Vector Register (DIVR).

BIT 4 - INTERRUPT ENABLE. This bit should be set to enable the corresponding interrupt.

BIT 3 - INTERRUPT AUTO CLEAR. This bit automatically clears the interrupt enable bit and the interrupt request output whenever the interrupt is acknowledged. This bit must be set in both registers to provide VMEbus compatible timing. This requires that the interrupt enable bit be set after each interrupt.

BITS 2, 1, 0 - IRQ LEVEL BITS. These bits determine the interrupt level at which an interrupt request is made.

LEVEL	BIT 2	BIT 1	BIT 0
7	1	1	1
6	1	1	0
5	1	0	1
4	1	0	0
3	0	1	1
2	0	1	0
1	0	0	1
DISABLED	0	0	0

4.5.2 BERR Interrupt Vector Register (BERRIVR)

This register holds the vector used for the bus error interrupt, any 8-bit integer may be used as determined by the system configuration.

4.5.3 Attention Interrupt Vector Register (AIVR)

This register holds the vector used for the attention interrupt. Any 8-bit integer may be used as determined by the system configuration.

4.6 PROGRAMMING THE ON-BOARD REGISTERS

4.6.1 Board Control Register HIGH/LOW

4.6.1.1 Board Control Register HIGH (BCRH)

BIT 15 - STATUS C. This bit is user-defined. It indicates the state of the FNCT1 bit of the other VMIVME-DR11W device in back-to-back mode.

STATUS C	OTHER VMIVME-DR11W
0	Transmitter
1	Receiver

BIT 14 - STATUS B. This bit is user-defined. It indicates the state of the other VMIVME-DR11W's FNCT2 bit in back-to-back mode.

BIT 13 - STATUS A. This bit is user-defined. It indicates the state of the other VMIVME-DR11W's FNCT3 bit in back-to-back mode.

STATUS A	OTHER VMIVME-DR11W
0	Burst Transfers
1	Normal (Single Cycle) Transfers

BIT 12 - BURST REQUEST. This bit indicates the state of the request line. In the back-to-back mode, it is the same as STATUS A.

BIT 11 - C1. This bit indicates the state of the C1 line. In the back-to-back mode, C1 is controlled by FUNCTION 1 of the same board.

C1	PROGRAM OCR BIT 7	THIS VMIVME-DR11W
0	0	Transmitter
1	1	Receiver

BIT 10 - ATTN. This bit indicates the state of the ATTN line. In back-to-back mode it is the same as Status B. A positive edge on the ATTN input will generate an interrupt (if enabled); therefore, the other VMIVME-DR11W can generate an interrupt by pulsing the ATTN line.

BIT 8 - RESERVED.

BIT 9 - RESERVED. (Except -100 Option).

BIT 9 - OTHER BOARD READY. (-100 Option only). This bit indicates the status of the other board's Go bit.

4.6.1.2 Board Control Register LOW (BCRL)

BIT 7 - FUNCTION 3. This bit is user-defined. It drives the STATUS A and BURST RQ lines of the other VMIVME-DR11W in back-to-back mode.

FUNCTION 3	TELLS OTHER VMIVME-DR11W TO USE
0	Burst Transfers
1	Normal (Single Cycle) Transfers

BIT 6 - FUNCTION 2. This bit is user-defined and drives the Status B and ATTN lines of the other DR11W in back-to-back mode. This bit can also be pulsed high to generate an attention interrupt.

BIT 5 - FUNCTION 1. This bit is user-defined, and in the back-to-back mode it drives its own C1 line and the Status C line of the other VMIVME-DR11W.

FNCT1 BIT	THIS VMIVME-DR11W
0	Transmitter
1	Receiver

This bit should be the complement of STATUS C.

BIT 4 - RESERVED.

BIT 3 - LED CONTROL. This bit controls the front panel Fail LED.

0 = LED ON
1 = LED OFF

BIT 2 - ENABLE ATTN INTERRUPT. This bit is used to control the ATTN interrupt flip-flop. This bit must be set in addition to AICR - BIT 4, to enable external attention interrupts.

BIT 1 - CYCLE. This bit is used to start the first handshake cycle when the VMIVME-DR11W is the transmitter. It may be set at the same time as the GO bit.

BIT 0 - GO. This bit sets up the handshake logic to perform transfers and causes a pulse to be generated on the VMIVME-DR11W's GO output. It must be set before the start of each block of transfers. The DMA start bit (BIT 7 of CCR) must be set before the GO bit is set.

4.6.2 I/O Data Register HIGH/LOW (IODRH, IODRL)

These two registers are used to read and write data via program control to or from the VMIVME-DR11W's data bus. The output drivers will be disabled when C1 is set to read. In the back-to-back mode the C1 input line is controlled by FNCT1 of the same board. Note that the IODR should always be accessed as a word when writing.

FNCT1	C1	OUTPUT DRIVERS
0	0	Enabled
1	1	Disabled

During program controlled transfers, the ATTN interrupt can be used to inform the other VMIVME-DR11W that data is ready to be read.

4.6.3 Address Modifier Register

This write only register contains the address modifier which is asserted when the VMIVME-DR11W is bus master.

BITS 7, 6. Not used.

BITS 5, 4, 3, 2, 1, 0 - ADDRESS MODIFIER BITS. Bits 5, 4, 3, 2, 1, 0 drive address modifier bits 5, 4, 3, 2, 1, 0, respectively, on the VMEbus during a DMA transfer. A description of the VMEbus address modifier codes is shown in Table 4.6.3-1.

4.7 SAMPLE SOFTWARE LISTINGS

VMIC has provided sample software listings in Appendix C to assist the user in programming the VMIVME-DR11W. Refer to Appendix C for an itemized list of software listings provided.

Table 4.6.3-1. Address Modifier Codes

HEXADECIMAL CODE	ADDRESS MODIFIER 5	4	3	2	1	0	FUNCTION	DEFINED BY
3F	H	H	H	H	H	H	Standard Supervisory Ascending Access	VMEbus Spec.
3E	H	H	H	H	H	L	Standard Supervisory Program Access	VMEbus Spec.
3D	H	H	H	H	L	H	Standard Supervisory Data Access	VMEbus Spec.
3C	H	H	H	H	L	L	Undefined	Reserved
3B	H	H	H	L	H	H	Standard Nonprivileged Ascending Access	VMEbus Spec.
3A	H	H	H	L	H	L	Standard Nonprivileged Program Access	VMEbus Spec.
39	H	H	H	L	L	H	Standard Nonprivileged Data Access	VMEbus Spec.
38	H	H	H	L	L	L	Undefined	Reserved
30-37	H	H	L	X	X	X	Undefined	Reserved
2F	H	L	H	H	H	H	Undefined	Reserved
2E	H	L	H	H	H	L	Undefined	Reserved
2D	H	L	H	H	L	H	Short Supervisory I/O Access	VMEbus Spec.
2C	H	L	H	H	L	L	Undefined	Reserved
2B	H	L	H	L	H	H	Undefined	Reserved
2A	H	L	H	L	H	L	Undefined	Reserved
29	H	L	H	L	L	H	Short Nonprivileged Access	VMEbus Spec.
28	H	L	H	L	L	L	Undefined	Reserved
20-27	H	L	L	X	X	X	Undefined	Reserved
10-1F	L	H	X	X	X	X	Undefined	User
0F	L	L	H	H	H	H	Extended Supervisory Ascending Access	VMEbus Spec.
0E	L	L	H	H	H	L	Extended Supervisory Program Access	VMEbus Spec.
0D	L	L	H	H	L	H	Extended Supervisory Data Access	VMEbus Spec.
0C	L	L	H	H	L	L	Undefined	Reserved
0B	L	L	H	L	H	H	Extended Nonprivileged Ascending Access	VMEbus Spec.
0A	L	L	H	L	H	L	Extended Nonprivileged Program Access	VMEbus Spec.
09	L	L	H	L	L	H	Extended Nonprivileged Data Access	VMEbus Spec.
08	L	L	H	L	L	L	Undefined	Reserved
00-07	L	L	L	X	X	X	Undefined	Reserved

MDR11/T4.6.3-1

SECTION 5

CONFIGURATION AND INSTALLATION

5.1 UNPACKING PROCEDURES

```
*****************
*    CAUTION    *
*****************
```

SOME OF THE COMPONENTS ASSEMBLED ON VMIC's PRODUCTS MAY BE SENSITIVE TO ELECTROSTATIC DISCHARGE AND DAMAGE MAY OCCUR ON BOARDS THAT ARE SUBJECTED TO A HIGH ENERGY ELECTROSTATIC FIELD. WHEN THE BOARD IS TO BE LAID ON A BENCH FOR CONFIGURING, etc., IT IS SUGGESTED THAT CONDUCTIVE MATERIAL BE INSERTED UNDER THE BOARD TO PROVIDE A CONDUCTIVE SHUNT. UNUSED BOARDS SHOULD BE STORED IN THE SAME PROTECTIVE BOXES IN WHICH THEY WERE SHIPPED.

Upon receipt, any precautions found in the shipping container should be observed. All items should be carefully unpacked and thoroughly inspected for damage that might have occurred during shipment. The board(s) should be checked for broken components, damaged printed circuit board(s), heat damage, and other visible contamination. All claims arising from shipping damage should be filed with the carrier and a complete report sent to VMIC together with a request for advice concerning disposition of the damaged item(s).

5.2 PHYSICAL INSTALLATION

```
*****************
*    CAUTION    *
*****************
```

DO NOT INSTALL OR REMOVE BOARD WHILE POWER IS APPLIED.

De-energize the equipment and insert the board into an appropriate slot of the chassis. While ensuring that the board is properly aligned and oriented in the supporting card guides, slide the board smoothly forward against the mating connector until firmly seated.

5.3 JUMPER INSTALLATIONS

Revision H and subsequent revisions of the board provide for jumper selection of the VMEbus priority level and for the polarity of the BUSY handshake signal. The reader should refer to Figure 5.3-1 for jumper locations.

5.3.1 VMEbus Priority Jumpers

Jumpers J2 and J3 provide for the selection of bus request level and grant level, respectively. The reader should refer to Figure 5.3.1-1 for selection of bus priority level (3, 2, 1, or 0). The factory configuration is level 3.

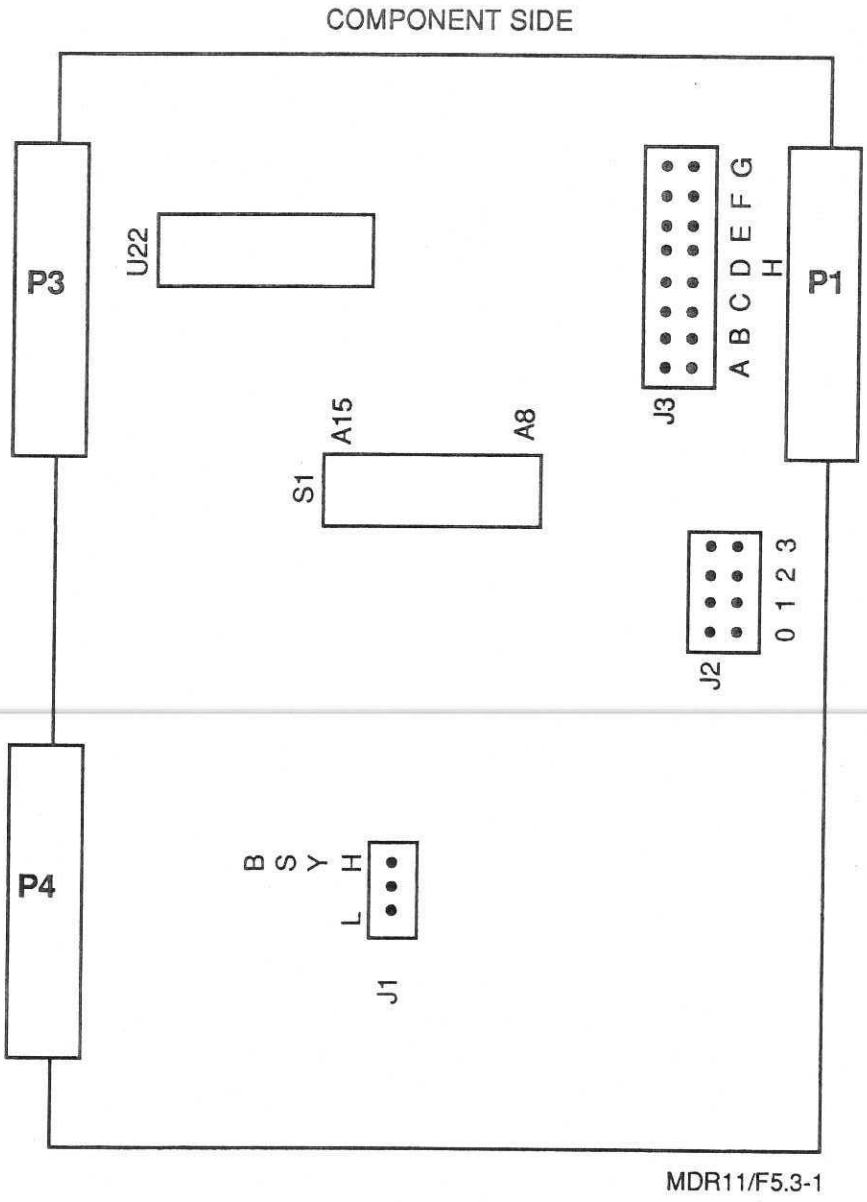


Figure 5.3-1. VMIVME-DR11W Configuration Information (Rev. H Boards)

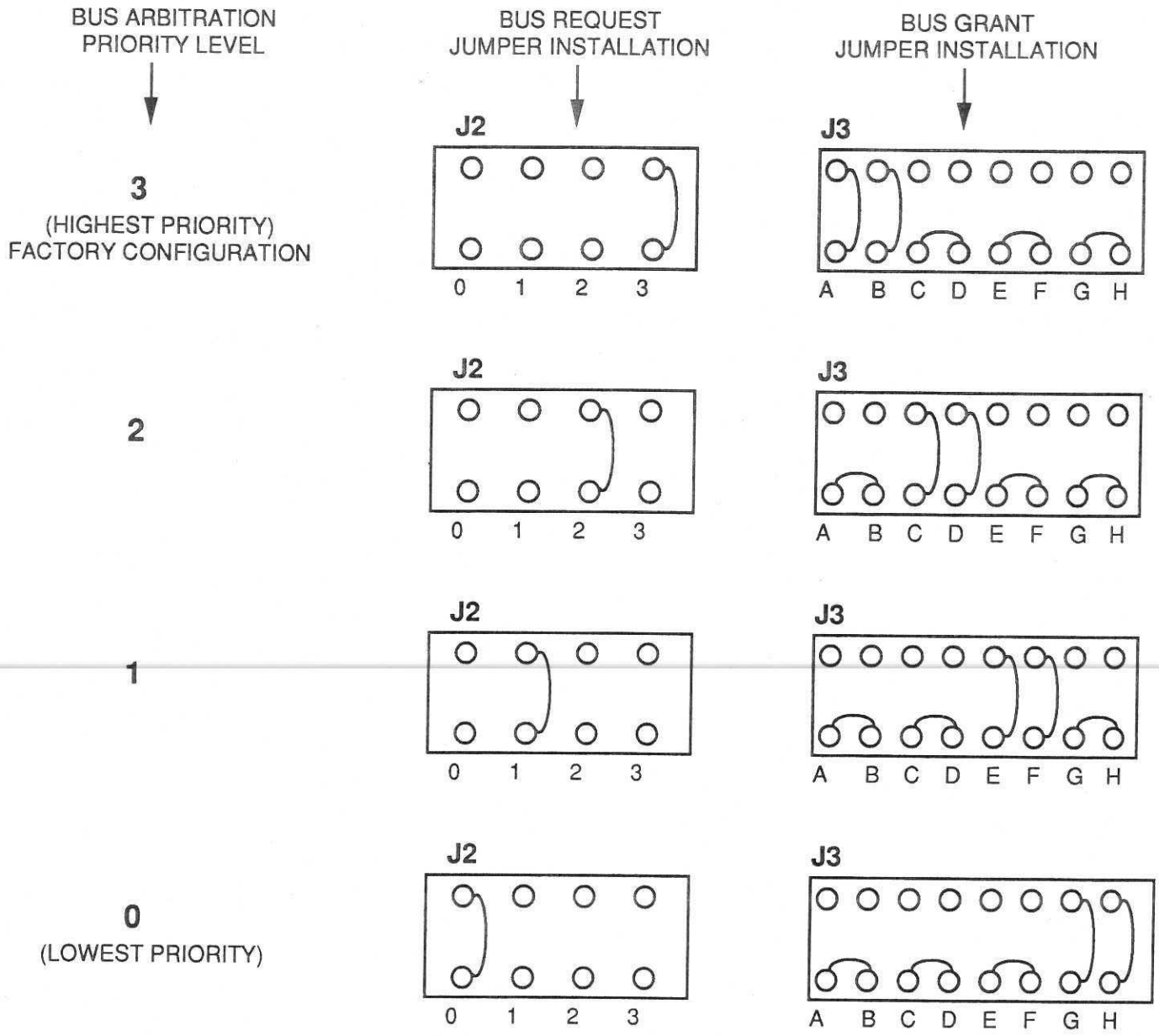


Figure 5.3.1-1. Jumper Installation for Selection of VMEbus Priority Level (Rev. H and Subsequent Revisions of the Board)

5.3.2 Busy Polarity

The polarity of the BUSY handshake signal can be selected to be active H or L. As shown in Figure 5.3.2-1 BUSY L must be selected for back-to-back operation of two VMIVME-DR11W Boards. The factory configuration is Busy L.

5.4 BOARD BASE ADDRESS

The VMIVME-DR11W occupies 256 bytes of the VMEbus short I/O space. The upper eight bits of the address are DIP switch selectable. Refer to Figure 5.4-1 for configuration of this address selection and to Figure 5.3-1 for location of the DIP switch.

5.5 ADDRESS MODIFIERS

The board is configured at the factory to respond to short I/O address (16 bits) via a programmed PAL. See Figure 5.3-1 for location of the address modifier PAL.

5.6 I/O CABLES

The VMIVME-DR11W provides a TTL compatible user interface that is terminated in 120 Ω via 180 Ω/390 Ω terminating resistors. The VMIVME-DR11W may be connected to external devices or to another VMIVME-DR11W using up to 50-foot (15.2 m) cables, if the following guidelines are followed:

a. Shielded flat cable (or superior cable) must be used. Mass-terminated unshielded flat cable will cause data errors. A cable with individually shielded conductors or a signal/ground wire pair for each signal (40 pairs) should provide best results.

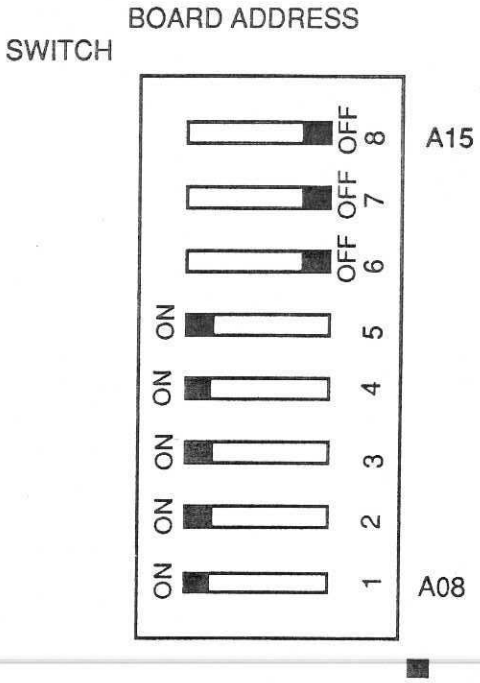
b. The cable shield must be grounded at both ends of the interface cable if a shielded cable is used. The cable grounds should have low impedance at high frequencies.

c. Shielded flat cables typically have the shield connected to pin 1; therefore, Pin 1 of the cable must connect to Pin 40 of the connector (because the standard VMIVME-DR11W pinout has data bit 15 tied to Pin 1.)

d. A special cable is required for the macrolink interface. The interface on the macrolink side requires that Pin 1 on the connector connects to Pin 1 of the cable and Pin 40 of the connector connects to Pin 40 of the cable.

e. Pin 40 of the board connectors for the cable (P3 and P4) is located at the top of the connector. For back-to-back connection of the VMIVME-DR11W units using VMIC I/O cables, the cable should be connected with the black stripe (which designates ground) toward the top of the connector (and thus toward the top of board).

Figure 5.3.2-1. Busy Signal Polarity Selection



ON, CLOSED = 0
OFF, OPEN = 1

EXAMPLE: BASE ADDRESS = $E000; $E0 = 1110 0000

MDR11/F5.4-1

Figure 5.4-1. VMIVME-DR11W Base Address Selection Switch

Refer to Figures 2.1-1, 3.1-2, and 3.1-3 for proper cable connection to the user device. To connect two VMIVME-DR11Ws back-to-back, P3 should be connected to P4 on the other device, and P4 should be connected to P3 on the other device.

The VMIVME-DR11W is pinout compatible with DEC DR11-W and DEC DRV-11B type devices. It is ***NOT*** pinout compatible with DEC DR11-B type devices and damage may result if it is cabled directly to a DEC DR11-B type device.

5.6.1 Macrolink Pin Number Reversal

The macrolink VMIVME-DR11W compatible interface has Pin 1 of the connector connected to ground and Pin 40 connected to data bit 15. The special cable required can be ordered from VMIC.

5.7 PIN NUMBER REVERSAL THROUGH BULKHEAD CONNECTOR

If two flat cables are mated together by back-to-back male connectors a pin number reversal will occur, i.e, the odd-numbered conductors will be swapped with the even numbered conductors. Therefore, if the Digital Equipment Corporation DC06 cable kit is utilized, the short pin number reversing cable (in the kit) must be installed to correct the pin number reversal.

5.8 CABLE CONNECTOR

A compatible cable connector for the VMIVME-DR11W is Panduit No. 050-040-455. Strain relief Panduit No. is 100-000-058.

5.9 USER CONNECTOR PIN ASSIGNMENTS

User connector pin assignments and signal names are shown in Table 5.9-1.

Table 5.9-1. Connector Pin Assignments

CONNECTOR P3			
PIN	**SIGNAL**	**PIN**	**SIGNAL**
1 VV	D0 15 H	21 X	END CYCLE H
2 UU	D0 00 H	22 W	GND
3 TT	D0 14 H	23 V	STATUS C H
4 SS	D0 01 H	24 U	GND
5 RR	D0 13 H	25 T	STATUS C H
6 PP	D0 02 H	26 S	GND
7 NN	D0 12 H	27 R	STATUS B H
8 MM	D0 03 H	28 P	GND
9 LL	D0 11 H	29 N	INIT H
10 KK	D0 04 H	30 M	GND
11 JJ	D0 10 H	31 L	STATUS A H
12 HH	D0 05 H	32 K	BURST RQ L
13 FF	D0 09 H	33 J	WC INC ENB H*
14 EE	D0 06 H	34 H	GND
15 DD	D0 08 H	35 F	READY H
16 CC	D0 07 H	36 E	GND
17 BB	GND	37 D	ACLO FNCT 2 H
18 AA	GND	38 C	GND
19 Z	CYCLE RQB H	39 B	CYCLE RQA H
20 Y	GND	40A	GND

CONNECTOR P4			
PIN	**SIGNAL**	**PIN**	**SIGNAL**
1 VV	DI 15 H	21 X	GO H
2 UU	DI 00 H	22 W	GND
3 TT	DI 14 H	23 V	FNCT 1 H
4 SS	DI 01 H	24 U	GND
5 RR	DI 13 H	25 T	C1 H
6 PP	DI 02 H	26 S	GND
7 NN	DI 12 H	27 R	FNCT 2 H
8 MM	DI 03 H	28 P	GND
9 LL	DI 11 H	29 N	CO H*
10 KK	DI 04 H	30 M	GND
11 JJ	DI 10 H	31 L	FNCT 3 H
12 HH	DI 05 H	32 K	FNCT 3 H
13 FF	DI 09 H	33 J	BA IND ENB H*
14 EE	DI 06 H	34 H	GND
15 DD	DI 08 H	35 F	AOO H*
16 CC	DI 07 H	36 E	GND
17 BB	GND	37 D	ATTN H
18 AA	GND	38 C	GND
19 Z	GND	39 B	BUSY**
20 Y	GND	40A	GND

*Not utilized by or necessary for operation of VMIVME-DR11W link.
**See Section 5.3.2 for polarity selection.

MDR11/T5.9-1

SECTION 6

MAINTENANCE AND WARRANTY

6.1 MAINTENANCE

This section of the technical manual provides information relative to the care and maintenance of VMIC's products. Should the products malfunction, the user should verify the following:

a. Software
b. System configuration
c. Electrical connections
d. Jumper or configuration options
e. Boards fully inserted into their proper connector location
f. Connector pins are clean and free from contamination
g. No components of adjacent boards are disturbed when inserting or removing the board from the VMEbus card cage
h. Quality of cables and I/O connections

User level repairs are not recommended. Contact VMIC for a Return Material Authorization (RMA) Number. This RMA Number must be obtained prior to any return.

6.2 MAINTENANCE PRINTS

The appendix(ices) to this manual contain(s) drawings and diagrams for reference purposes.

6.3 WARRANTY

VMIC's Standard Products are warranted to be free from defects in material and workmanship for a period of two years (24 months) from the date of shipment. In discharge of this warranty, VMIC, at its option, agrees to either repair or replace, at VMIC's facility and at VMIC's discretion, any part, component, subassembly accessory, or any hardware, software, or system product, which under proper and normal use proves defective in material and workmanship.

The customer shall provide notice to VMIC of each such defect within a reasonable time after the customer's discovery of such defect.

In order to return the defective product(s) or part(s), the customer must contact VMIC's Customer Service Department to obtain a Call Ticket Number. The

defective product(s) or part(s) must also be properly boxed and weighed. After a VMIC Call Ticket Number and RMA Number have been obtained, the defective product(s) or part(s) may be returned (transportation collect for surface UPS) to VMIC. Any replaced or repaired product(s) or part(s) will be shipped back to the customer's at the expense of VMIC (also UPS surface).

The customer should be aware that the above process can sometimes take up to eight (8) days for the shipment to reach VMIC. The customer has the option to ship the defective product(s) or part(s) at the customer's own expense if the customer cannot afford this possible delay.

There shall be no warranty or liability on any VMIC product(s) or part(s) that is (are) damaged or subjected to accident(s), perils of nature, negligence, overtemperature, overvoltage, misapplication of electrical power, insertion or removal of boards from backplanes and/or I/O connectors with power applied by the customer(s), appointee(s), or any other person(s) without the expressed approval of VMIC.

Final determination of warranty eligibility shall be made by VMIC, and if a warranty claim is considered invalid for any reason, the customer will be charged for services performed and expenses incurred by VMIC in repair, handling and shipping the returned product or part. Determination as to whether the item is within warranty, coverage shall not be unreasonably withheld.

The warranty period of the replacement or repaired product(s) or part(s) shall terminate with the termination of the warranty period with respect to the original product(s) or part(s) for all replacement parts supplied or repairs made during the original warranty period.

THE FOREGOING WARRANTY AND REMEDY ARE EXCLUSIVE AND VMIC SHALL HAVE NO OTHER OR ADDITIONAL LIABILITY TO BUYER OR TO ANYONE CLAIMING UNDER BUYER (THIRD PARTY) UNDER ANY OTHER AGREEMENT OR WARRANTY, EXPRESS OR IMPLIED EITHER IN FACT OR BY OPERATION OF THE LAW, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY OR FITNESS, STATUTORY, OR OTHERWISE. VMIC SHALL HAVE NO LIABILITY FOR SPECIAL OR CONSEQUENTIAL DAMAGES OF ANY KIND OR FROM ANY CAUSE ARISING OUT OF THE INSTALLATION OR USE OF ANY PRODUCT FURNISHED HEREUNDER.

6.4 OUT-OF-WARRANTY REPAIR POLICY

The following sections describe VMIC's policy on repairs and warranties on repaired products.

6.4.1 Repair Category

VMIC's repair policy of standard products is divided into two categories, depending on the item to be repaired. These categories are:

a. Product Exchange
b. Fixed Price Repair

Category 1 (product exchange) represents the fastest turn around of the two categories. In this case, the customer sends the malfunctioning product to VMIC. VMIC will return an operational product to the customer within 72 hours of receipt provided VMIC has the product in stock.

Provided that the returned product is repairable customers should contact VMIC prior to returning products for repair to determine stocking status.

Category 2 (Fixed Price Repair) applies to products returned to VMIC for repair and subsequent return to the customer.

Return authorizations are required on all product repairs, and all purchase orders should refer to VMIC's RMA Number which is assigned by VMIC's Customer Service Department.

6.4.2 Repair Pricing

Contact your factory representative for repair pricing. Current pricing can be found in the Repair and Replacement Policy in the most current Standard Conditions of Sales Document (F0109-91). Refer to exclusions (Section 6.4.7).

6.4.3 Payment

Payment is due upon delivery or at VMIC's option, net thirty (30) days from the date of delivery. Payment should be made to:

VME Microsystems International Corporation
12090 South Memorial Parkway
Huntsville, Alabama 35803-3308
Attention: Accounts Receivable

VMIC allows a one (1) percent discount for payment made within ten (10) days of invoice date or a two (2) percent discount on payment made prior to shipment of order. This payment discount, however, does not apply to freight.

6.4.4 Shipping Charges

Shipping charges are the customer's responsibility, with the exception of warranty repairs, whereby VMIC will pay the return to customer shipping charges.

6.4.5 Shipping Instructions

The type of packaging used to ship the product depends on whether the product is shipped singly, in a chassis, or packaged with other boards. The shipper should carefully pack the product(s), using the same precautions listed in the "unpacking procedures". The user should utilize the same (or equivalent) protective packaging container for re-shipment as provided by VMIC. Approved ESD procedures are recommended when handling VMIC's products.

6.4.6 Warranty on Repairs

Products repaired by VMIC are warranted against defects in workmanship and material for a period of ninety (90) days from date of shipment to the customer for all products that were repaired out of warranty. See Standard Conditions of Sale for products repaired within the warranty.

6.4.7 Exclusions

Repair rates may not apply to products which have received unusual physical or electrical damage. In such cases, VMIC will provide an estimated price for product repair or replacement. The customer may then choose to have the product repaired at the estimated price, returned unrepaired at no charge, or replaced at VMIC's current list price.

APPENDIX A

ASSEMBLY DRAWING, PARTS LIST, AND SCHEMATIC

APPENDIX B

INTEGRATED CIRCUIT SPECIFICATIONS

DESCRIPTION	PART NO.
Bus Interrupt Module	MC68153
Direct Memory Access Interface (DMAI)	SCB68430

MOTOROLA

SEMICONDUCTORS

PO BOX 20912 • PHOENIX ARIZONA 85036

MC68153

Advance Information

BUS INTERRUPTER MODULE

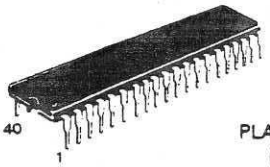
The bipolar LSI MC68153 Bus Interrupter interfaces a microcomputer system bus to multiple slave devices requiring interrupt capabilities. It handles up to 4 independent sources of interrupt requests and is fully programmable.

- VERSAbus/VMEbus Compatible
- MC68000 Compatible
- Handles 4 Independent Interrupt Sources
- 8 Programmable Read/Write Registers
- Programmable Interrupt Request Levels
- Programmable Interrupt Vectors
- Supports Interrupt Acknowledge Daisy Chain
- Control Registers Contain Flag Bits
- Single +5.0 Volt Supply
- Total Power Dissipation = 1.5 W Typical
- Temperature Range of 0°C to 70°C
- Chip Access Time = 200 ns Typical with 16 MHz Clock
- 40-Pin Dual-In-Line Package

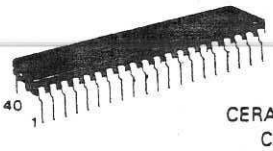
TTL

BUS INTERRUPTER MODULE

ADVANCED LOW POWER SCHOTTKY

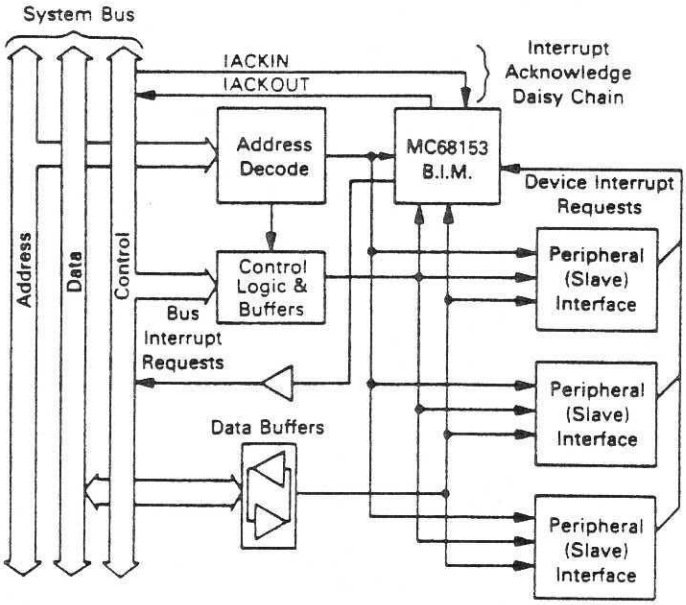


P SUFFIX
PLASTIC PACKAGE
CASE 711-03



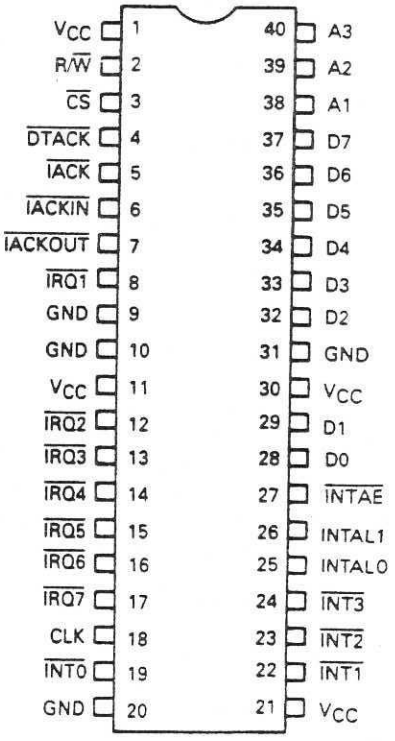
L SUFFIX
CERAMIC PACKAGE
CASE 734-04

FIGURE 1 — MC68153 SYSTEM BLOCK DIAGRAM



VERSAbus is a trademark of Motorola.

PIN ASSIGNMENTS



This document contains information on a new product. Specifications and information herein are subject to change without notice.

© MOTOROLA INC. 1984

ADI-1057

ABSOLUTE MAXIMUM RATINGS (Beyond which useful life may be impaired.)

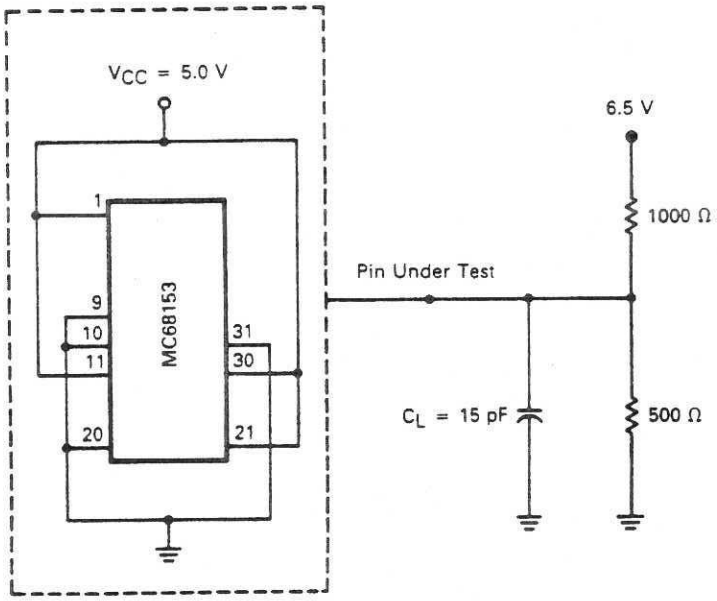
Parameter	Symbol	Value	Unit
Supply Voltage	V_{CC}	−0.5 to +7.0	V
Input Voltage	V_{in}	−0.5 to +7.0	V
Input Current	I_{in}	−30 to +5.0	mA
Output Voltage	V_{out}	−0.5 to +5.5	V
Output Current	I_{OL}	Twice Rated I_{OL}	mA
Storage Temperature	T_{stg}	−65 to +140	°C
Junction Operating Temperature	T_J	−55 to +140	°C

BURN-IN LIMITS: A maximum T_J of +175°C may be used for periods not to exceed 250 hours.

DC ELECTRICAL SPECIFICATIONS (V_{CC} = 5.0 V ±5%, T_A = 0°C to 70°C)

Parameter	Symbol	Min	Max	Unit	Test Conditions
High Level Input Voltage	V_{IH}	2.0	—	V	
Low Level Input Voltage	V_{IL}	—	0.8	V	
Input Clamp Voltage	V_{IK}	—	−1.5	V	V_{CC} = MIN, I_{IN} = −18 mA
High Level Output Voltage(1)	V_{OH}	2.7	—	V	V_{CC} = MIN, I_{OH} = −400 μA
Low Level Output Voltage	V_{OL}	—	0.4	V	V_{CC} = MIN, I_{OL} = 8.0 mA
Output Short Circuit Current(2)	I_{OS}	−15	−130	mA	V_{CC} = MAX, V_{OUT} = 0 V
High Level Input Current	I_{IH}	—	20	μA	V_{CC} = MAX, V_{IN} = 2.7 V
Low Level Input Current	I_{IL}	—	−0.4	mA	V_{CC} = MAX, V_{IN} = 0.4 V
Supply Current	I_{CC}	225	385	mA	V_{CC} = MAX
Output Off Current (High)	I_{OZH}	—	20	μA	V_{CC} = MAX, V_{OUT} = 2.4 V
Output Off Current (Low)	I_{OZL}	—	−20	μA	V_{CC} = MAX, V_{OUT} = 0.4 V

AC TEST CIRCUIT — AC Testing of All Outputs



NOTES:
1. Not applicable to open-collector outputs.
2. Not more than one output should be shorted at a time for longer than one second.
3. $\overline{CS}$ Low to CLK High (Setup Time) of 15 ns Min must be observed.
4. $\overline{IACK}$ Low to CLK High and $\overline{IACKIN}$ Low to CLK High (Setup Times) of 15 ns Min must be observed.
5. See Table 1 for additional performance guidelines.

AC ELECTRICAL SPECIFICATIONS (V_{CC} = 5.0 V ±5%, T_A = 0°C to 70°C)

Parameter	Test Number(5)	Max (ns)
CLK High to Data Out Valid (Delay)(3)	1	55
CLK High to $\overline{DTACK}$ Low (Delay)(3)	2	40
$\overline{CS}$ High to $\overline{DTACK}$ High (Delay)	3	35
CLK High to Data Out Valid (Delay)(4)	4	55
CLK High to $\overline{INTAE}$ Low (Delay)(4)	5	40
$\overline{IACK}$ High to Data Out High Impedance (Delay)	6	60
$\overline{IACK}$ High to $\overline{DTACK}$ High (Delay)	7	45
$\overline{CS}$ High to Data Out High (Delay)	8	45
CS High to IRQ High (Delay)	9	60
$\overline{IACK}$ High to $\overline{INTAE}$ High (Delay)	10	35

GENERAL DESCRIPTION

The MC68153 Bus Interrupter Module (BIM) is designed to serve as an interrupt requester for peripheral devices in a microcomputer system. Up to 4 independent devices can be interfaced to the system bus by the MC68153. Intended for asynchronous master/slave bus operation, the BIM is compatible with VERSAbus, VMEbus, MC68000 device bus, and other system buses. Figure 1 shows a block diagram of a typical configuration. In this example, three peripheral devices (bus slaves) are connected to the system data bus. Each of these devices could be parallel I/O, serial I/O, or some other function. An interrupt request from any device is routed to the MC68153, and the BIM handles all interface to the system bus. It generates a bus interrupt request as a result of the device interrupt request. When the system interrupt handler or processor responds with an interrupt acknowledge cycle, the MC68153 can answer supplying an interrupt vector and handling all timing.

The functional block diagram of the MC68153 is shown in Figure 2. The device contains circuitry to accept four separate interrupt sources ($\overline{INT0}$ – $\overline{INT3}$). Interface to the system bus includes generation of bus interrupt requests ($\overline{IRQ1}$ – $\overline{IRQ7}$), response to a bus interrupt acknowledge cycle (either supplying a vector or passing on a daisy chain signal), and releasing the bus interrupt request signal at the proper time. The BIM has flexibility provided by eight programmable read/write registers. Four 8-bit vector registers (VR0 – VR3) contain status/address information and supply a byte vector in response to an interrupt acknowledge cycle for the corresponding interrupt source. Four other 8-bit control registers (CR0 – CR3) contain information that oversees operation of the interrupt circuitry. The control information is programmable and includes interrupt request level and interrupt enable and disable. Also contained in the control registers are flag-bits. These flags are useful for task coordination, resource management, and interprocessor communication.

SIGNAL DESCRIPTION

Throughout the data sheet, signals are presented using the terms asserted and negated independent of whether the signal is asserted in the high voltage or low voltage state. Active low signals are denoted by a superscript bar.

BIDIRECTIONAL DATA BUS — D0 – D7

Pins D0 – D7 form an 8-bit bidirectional data bus to/from the system bus. These are active high, 3-state pins. D7 is the most significant bit.

ADDRESS INPUTS — A1 – A3

These active high inputs serve two functions. One function is to select one of the eight possible registers during a read or write cycle. Secondly, during an interrupt acknowledge A1 – A3 show the level of interrupt being acknowledged, and the BIM uses these to determine if a match exists with an internal level.

CHIP SELECT — $\overline{CS}$

$\overline{CS}$ is an active low input used to select the BIM's registers for the current bus cycle. Address strobe, data strobe, and appropriate address bits must be included in the chip select equation.

READ/WRITE — R/$\overline{W}$

The R/$\overline{W}$ input is a signal from the system bus used to determine if the current bus cycle is a read (high) or write (low).

DATA TRANSFER ACKNOWLEDGE — $\overline{DTACK}$

$\overline{DTACK}$ is an open-collector, active low output that signals the completion of a read, write, or interrupt acknowledge cycle. During read or interrupt acknowledge cycles, $\overline{DTACK}$ is asserted by the MC68153 after data has been provided on the data bus; during write cycles it is asserted after data has been accepted from the data bus. A pullup resistor is required to maintain $\overline{DTACK}$ high between bus cycles.

FIGURE 2 — MC68153 FUNCTIONAL BLOCK DIAGRAM

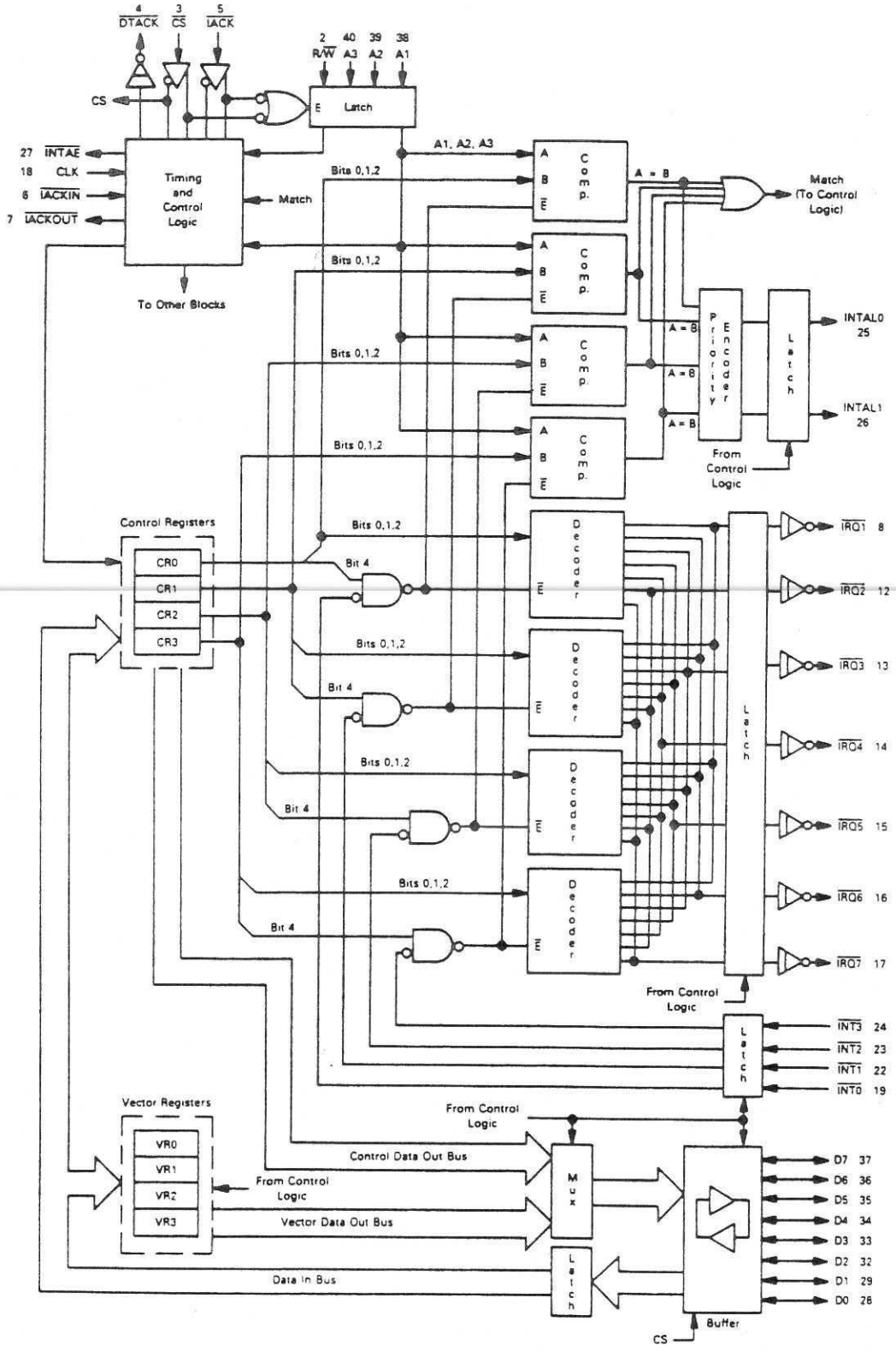
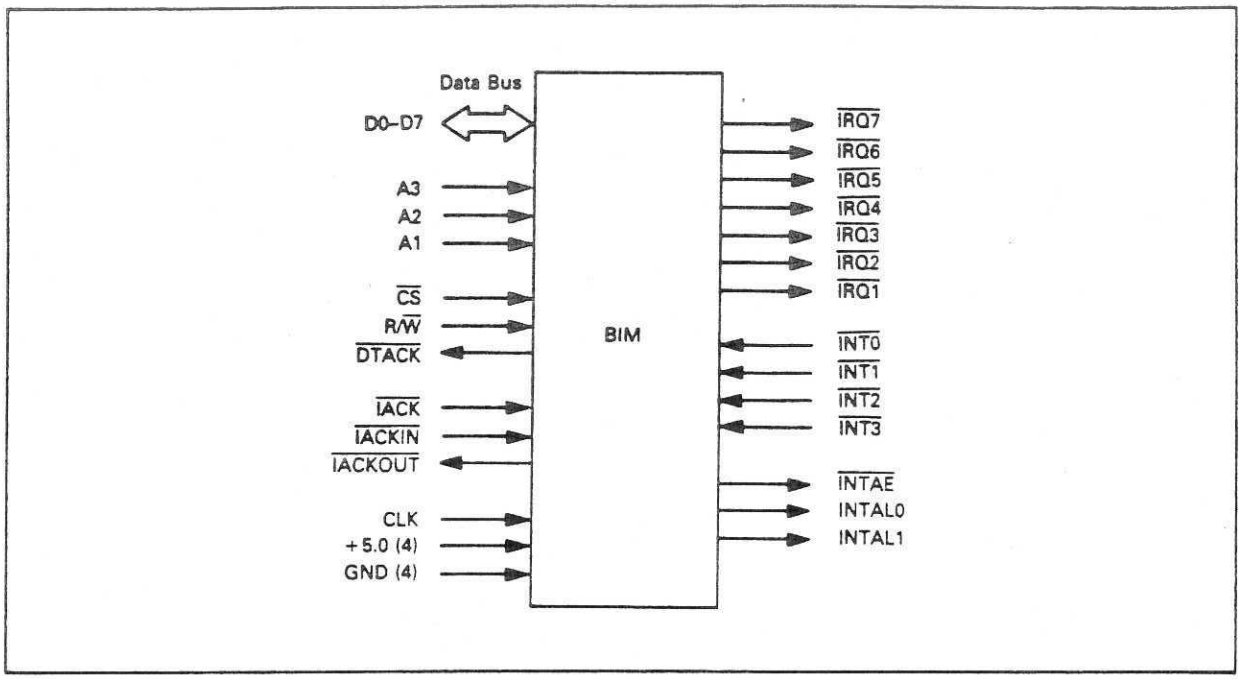


FIGURE 3 — LOGICAL PIN ASSIGNMENT



INTERRUPT ACKNOWLEDGE SIGNALS — IACK, IACKIN, IACKOUT

These three pins support the interrupt acknowledge cycle. A low level on the IACK input indicates an interrupt acknowledge cycle has been initiated. This signal is conditioned externally with Address Strobe and the lower data strobe of an MC68000 type bus. After IACK is asserted the BIM compares the interrupt level presented on address lines A1, A2, and A3 with the current levels generated internally and determines if a match exists. Then, if input IACKIN is asserted (driven low), the BIM will either complete the interrupt acknowledge cycle if a match exists or assert output IACKOUT if *no* match exists.

IACKIN and IACKOUT form part of a prioritized interrupt acknowledge daisy chain. The daisy chain prioritizes interrupters and guarantees that two or more devices requesting an interrupt on the same level will not respond to the same cycle. The requesting device (or interrupter) must wait until IACKIN is asserted and not pass the signal on (assert IACKOUT) if it is to complete the interrupt acknowledge cycle.

BUS INTERRUPT REQUEST SIGNALS — IRQ1 – IRQ7

These open-collector outputs are low when asserted, indicating a bus interrupt is requested at the corresponding level. An open-collector buffer is normally required for sufficient drive when interfacing to a system bus. A pullup resistor is required to maintain IRQ1 – IRQ7 high between interrupt requests.

DEVICE INTERRUPT REQUEST SIGNALS — INT0 – INT3

INT0 – INT3 are active low inputs used to indicate to the BIM that a device wants a bus interrupt.

INTERRUPT ACKNOWLEDGE ENABLE — INTAE

During an interrupt acknowledge cycle, this output pin is asserted low to indicate that outputs INTAL0 and INTAL1 are valid. These two outputs contain an encoded number (x) corresponding to the interrupt (INTx) being acknowledged. This feature can be used to signal interrupting devices, which supply their own vector, when to respond to the interrupt acknowledge cycle with the vector and a DTACK signal.

INTERRUPT ACKNOWLEDGE LEVEL — INTAL0, INTAL1

These active high outputs contain an encoded number corresponding to the interrupt level being acknowledged. They are valid only when INTAE is asserted low.

CLOCK — CLK

The CLK input is used to supply the clock for internal operations of the MC68153.

RESET — CS, IACK

Although a reset input is not supplied, an on-board reset is performed if CS and IACK are asserted simultaneously.

FIGURE 4 — MC68153 REGISTER MODEL

ADDRESS BIT A3	A2	A1	FLAG	FLAG AUTO-CLEAR	EXTERNAL/INTERNAL	INTERRUPT ENABLE	INTERRUPT AUTO-CLEAR	INTERRUPT LEVEL			REGISTER NAME
0	0	0	F	FAC	X/$\overline{IN}$	IRE	IRAC	L2	L1	L0	CONTROL REGISTER 0
0	0	1	F	FAC	X/$\overline{IN}$	IRE	IRAC	L2	L1	L0	CONTROL REGISTER 1
0	1	0	F	FAC	X/$\overline{IN}$	IRE	IRAC	L2	L1	L0	CONTROL REGISTER 2
1	1	1	F	FAC	X/$\overline{IN}$	IRE	IRAC	L2	L1	L0	CONTROL REGISTER 3
1	0	0	V7	V6	V5	V4	V3	V2	V1	V0	VECTOR REGISTER 0
1	0	1	V7	V6	V5	V4	V3	V2	V1	V0	VECTOR REGISTER 1
1	1	0	V7	V6	V5	V4	V3	V2	V1	V0	VECTOR REGISTER 2
1	1	1	V7	V6	V5	V4	V3	V2	V1	V0	VECTOR REGISTER 3
			7	6	5	4	3	2	1	0	REGISTER NAME
			REGISTER BIT								

REGISTER DESCRIPTION

The MC68153 contains 8 programmable read/write registers. There are four control registers (CR0 – CR3) that govern operation of the device. The other four (VR0 – VR3) are vector registers that contain the vector data used during an interrupt acknowledge cycle. Figure 4 illustrates the device register model.

CONTROL REGISTERS

There is a control register for each interrupt source, i.e., CR0 controls $\overline{INT0}$, CR1 controls $\overline{INT1}$, etc. The control registers are divided into several fields:

1. Interrupt level (L2, L1, L0) — The least significant 3-bit field of the register determines the level at which an interrupt will be generated:

L2	L1	L0	IRQ LEVEL
0	0	0	DISABLED
0	0	1	IRQ1
0	1	0	IRQ2
0	1	1	IRQ3
1	0	0	IRQ4
1	0	1	IRQ5
1	1	0	IRQ6
1	1	1	IRQ7

   A value of zero in the field disables the interrupt.

2. Interrupt Enable (IRE) — This field (Bit 4) must be set (high level) to enable the bus interrupt request associated with the control register. Thus, if the $\overline{INTX}$ line is asserted and IRE is cleared, *no* interrupt request ($\overline{IRQX}$) will be asserted.
3. Interrupt Auto-Clear (IRAC) — If the IRAC is set (Bit 3), IRE (Bit 4) is cleared during an interrupt acknowledge cycle responding to this request. This action of clearing IRE disables the interrupt request. To re-enable the interrupt associated with this register, IRE must be set again by writing to the control register.
4. External/Internal (X/$\overline{IN}$) — Bit 5 of the control register determines the response of the MC68153 during an interrupt acknowledge cycle. If the X/$\overline{IN}$ bit is clear (low level) the BIM will respond with vector data and a $\overline{DTACK}$ signal, i.e., an internal response. If X/$\overline{IN}$ is set, the vector is not supplied and no $\overline{DTACK}$ is given by the BIM, i.e., an external device should respond.
5. Flag (F) — Bit 7 is a flag that can be used in conjunction with the test and set instruction of the MC68000. It can be changed without affecting chip operation. It is useful for processor-to-processor communication and resource allocation.
6. Flag Auto-Clear (FAC) — If FAC (Bit 6) is set, the Flag bit is automatically cleared during an interrupt acknowledge cycle.

VECTOR REGISTERS

Each interrupt input has its own associated vector register. Each register is 8 bits wide and supplies a data byte during its interrupt acknowledge cycle if the associated External/Internal (X/$\overline{IN}$) control register bit is clear (zero). This data can be status, identification, or address information depending on system usage. The information is programmed by the system user.

DEVICE RESET

When the MC68153 is reset, the registers are set to a known condition. The control registers are set to all zeros (low). The vector registers are set to $0F. This value is the MC68000 vector for an uninitialized interrupt vector.

FUNCTIONAL DESCRIPTION

SYSTEM OVERVIEW

The MC68153 can be used with many system buses, however, it is primarily intended for VMEbus, VERSAbus and MC68000 applications. Figure 5 shows a system configuration similar to VMEbus. In the figure only one system Data Transfer Bus (DTB) master is used. The Priority Interrupt structure provides a means for peripheral slave devices to ask for an interrupt of other processor (DTB master) activity and receive service from the processor. The MC68153 BIM acts as an interface device requesting and responding to interrupt acknowledge cycles for up to 4 independent slaves.

In Figure 5, functional modules are identified as Interrupters and an Interrupt Handler. An Interrupter (such as the MC68153) receives slave requests for an interrupt and handles all interface to the system bus required to ask for and respond to interrupt requests. The Interrupt Handler receives the bus interrupt requests, determines when an interrupt acknowledge will occur and at which level, and finally either performs the interrupt acknowledge (IACK) cycle or tells the DTB master to execute the IACK cycle.

The signal lines in the Priority Interrupt structure include (* — indicates active low):

1. IRQ1*–IRQ7* — seven prioritized interrupt request lines.
2. IACK* — signal line that indicates an interrupt acknowledge cycle is occurring.
3. IACKIN*/IACKOUT* — two signals that form part of a daisy chain that prioritizes interrupters.

In addition Data Transfer Bus control signals are involved in the IACK bus cycle:

1. AS* — the Address Strobe asserted low indicates a valid address is on the bus.
2. DSO* — the lower Data Strobe asserted low indicates a data transfer will occur on bus bits D00–D07.
3. WRITE* — the Read/Write is negated indicating the data is to be read from the Interrupter.
4. A01–A03 — Address lines A01–A03 contain the encoded priority level of the IACK cycle.
5. D00–D07 — Data bus lines D00–D07 are used to pass the interrupt vector from the responding Interrupter to the Interrupt Handler.
6. DTACK* — Data Transfer Acknowledge asserted low signals that the Interrupter has put the vector on the data bus.

FIGURE 5 — SIMPLE VMEbus CONFIGURATION

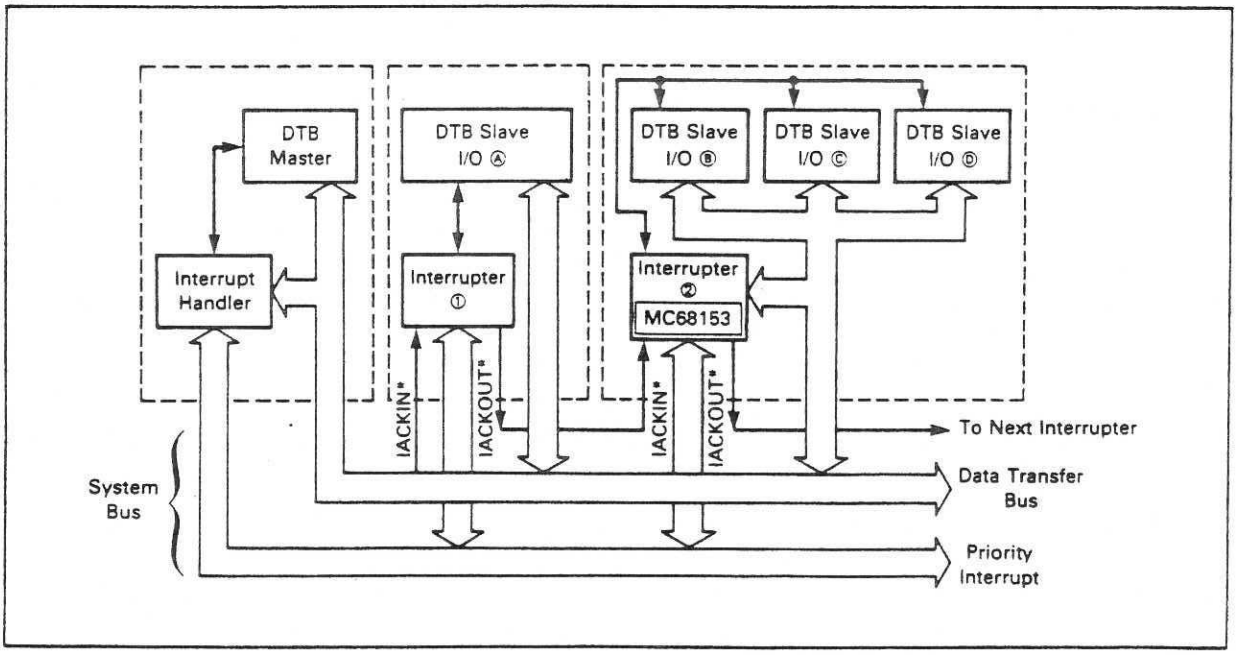
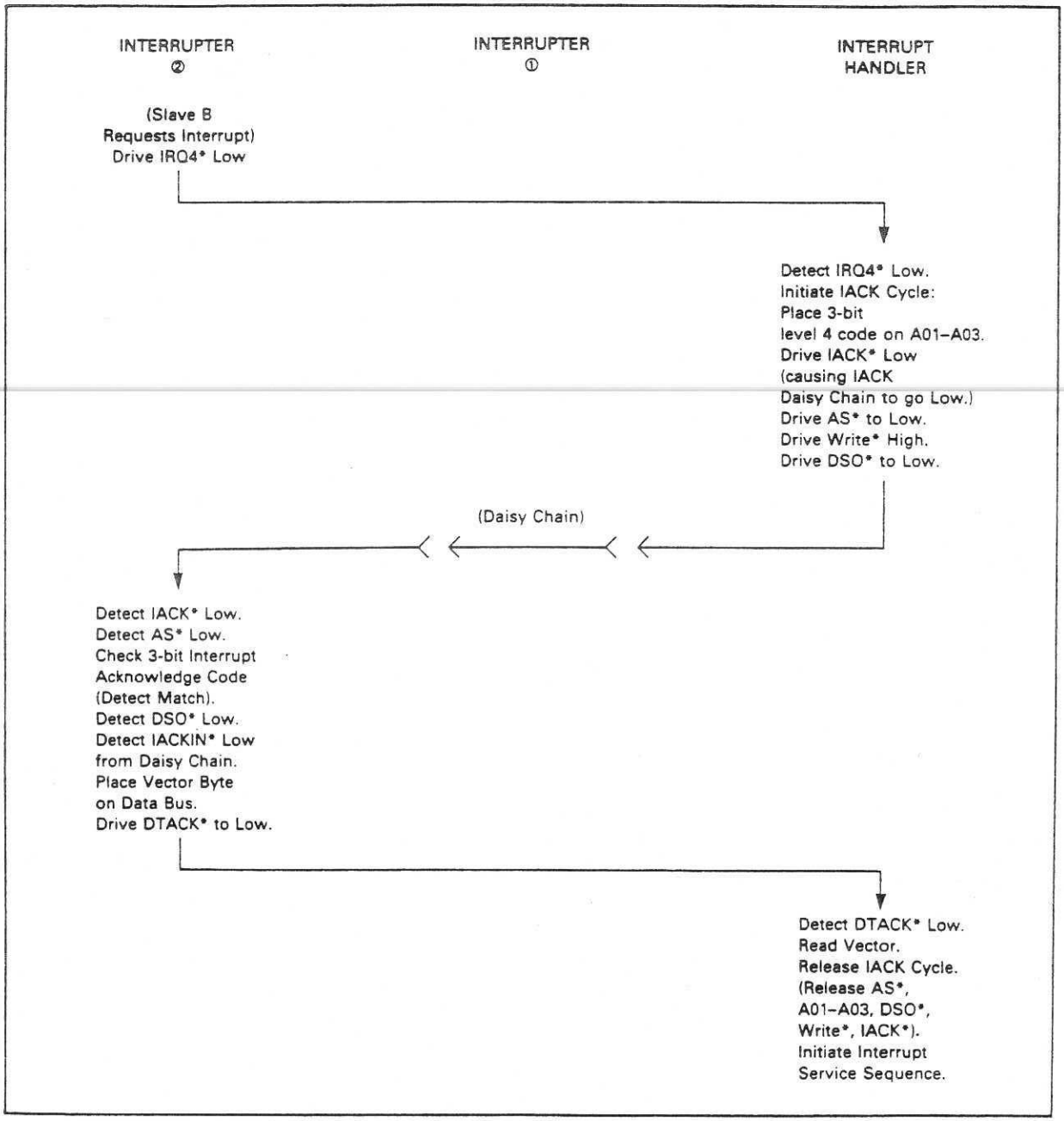


Figure 6 shows a flow diagram of a typical interrupt request and acknowledge operation. Briefly, the sequence of events is first, an Interrupter makes a request, next the Handler responds with an IACK cycle, then the Interrupter passes a vector to the Handler completing the IACK cycle, and finally the Handler uses the vector to determine additional action. Typically, an interrupt service routine is stored in software and the vector determines where its starting address is stored.

Note the daisy chain operation. If the IACK level (on A01–A03) does not match the Interrupter's request level or if no request is pending, the Interrupter passes the IACKIN* signal on and asserts IACKOUT*. This sequential action automatically prioritizes Requesters on the same level (first one in line with a request pending gets serviced) and prevents two or more Interrupters from responding simultaneously.

FIGURE 6 — INTERRUPT REQUEST AND ACKNOWLEDGE OPERATION FLOW DIAGRAM



This discussion is a very cursory look at the bus operation. For more details including situations with multiple bus masters, the user is directed to the VMEbus Specification MVMEBS or VERSAbus Specification M68KVBS. Also, the MC68153 can be used with other buses having similar interrupt structures.

BIM BUS INTERFACE

Figure 7 shows a simplified block diagram of the MC68153 interface to VERSAbus or VMEbus. Address Decode and Control Logic are dependent on the application and must be designed to guarantee BIM ac specifications. It is possible in most cases that the decode logic can be shared with the slave devices. Buffers are provided where shown to comply with bus loading and drive specifications. It is also possible that buffers can be shared with the slave bus interface.

READ/WRITE OPERATION

All eight BIM registers can be accessed from the system bus in both read and write modes. The BIM has an asynchronous bus interface, primarily designed for MC68000-like buses. The following BIM signals generate read and write cycles: Chip Select ($\overline{CS}$), Read/Write (R/$\overline{W}$), Address Inputs (A1–A3), Data Bus (D0–D7), and Data Transfer Acknowledge ($\overline{DTACK}$). During read and write cycles the internal registers are selected by A1, A2, and A3 in compliance with the Figure 4 Truth Table.

Figure 8 shows the device timing for a read cycle. R/$\overline{W}$ and A1–A3 are latched on the falling edge of $\overline{CS}$ and must meet specified setup and hold times. Chip access time for valid data and $\overline{DTACK}$ are dependent on the clock frequency as shown in the figure.

Figure 9 shows the device timing for a write cycle. R/$\overline{W}$, A1–A3, and D0–D7 are latched on the falling edge of $\overline{CS}$ and must meet specified setup and hold times. Chip access time for $\overline{DTACK}$ is dependent on the clock frequency as shown in the figure.

FIGURE 7 — VMEbus/VERSAbus INTERFACE BLOCK DIAGRAM

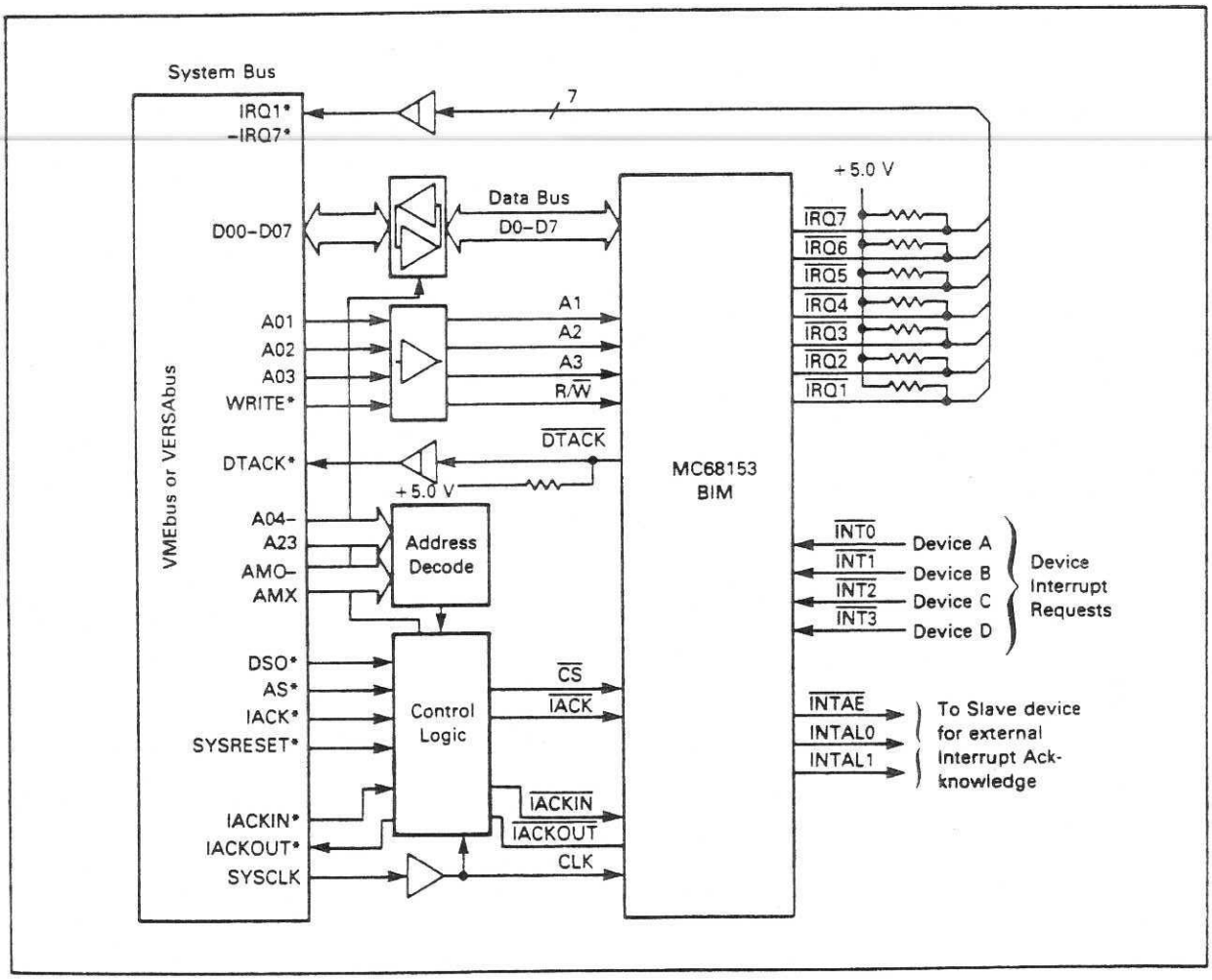


FIGURE 8 — READ CYCLE

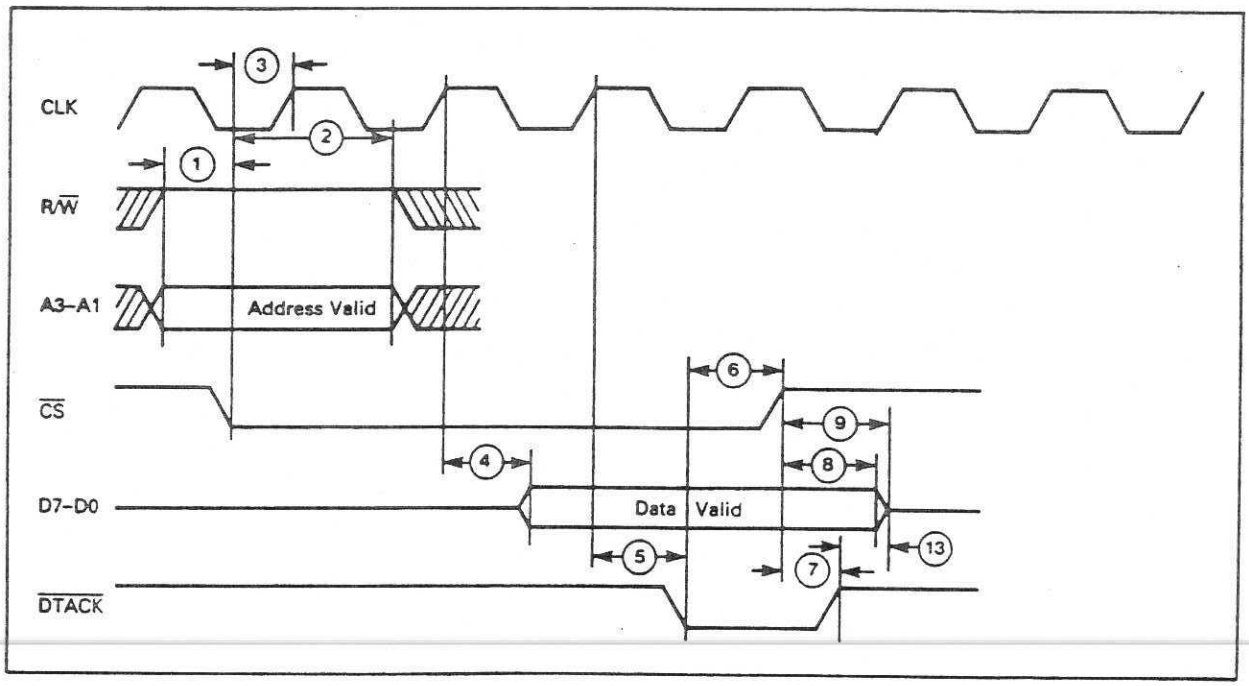
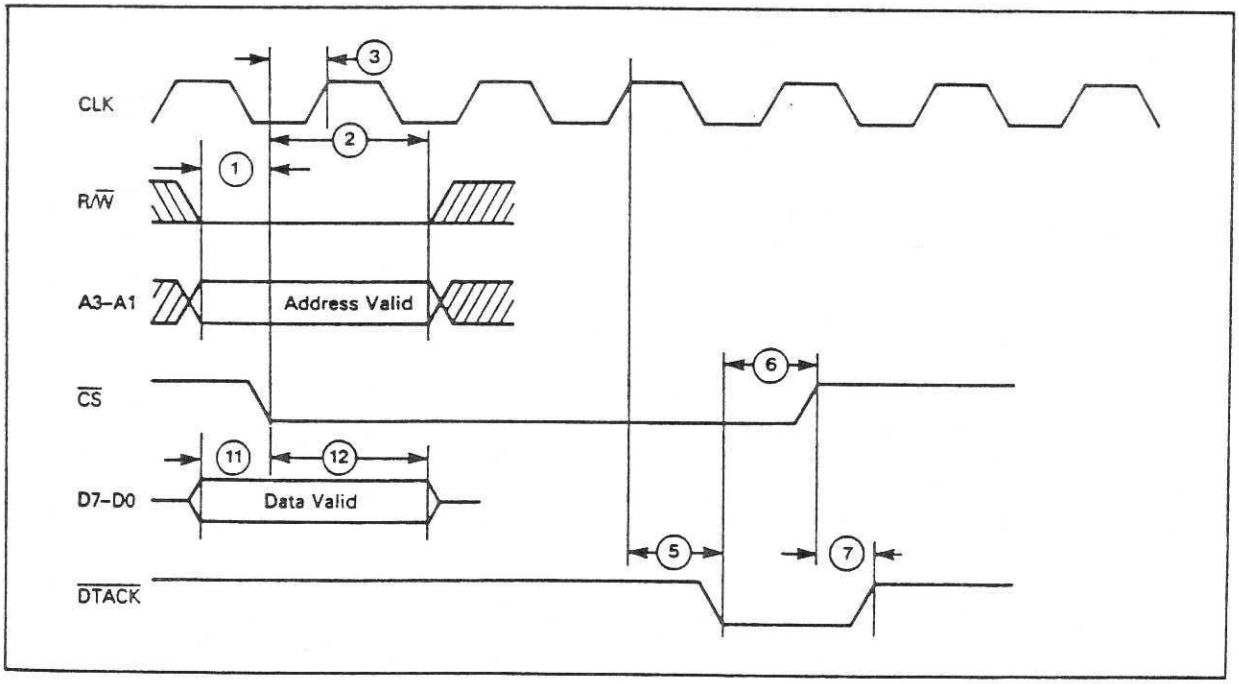


FIGURE 9 — WRITE CYCLE



INTERRUPT REQUESTS

The MC68153 accepts device interrupt requests on inputs $\overline{INT0}$, $\overline{INT1}$, $\overline{INT2}$, and $\overline{INT3}$. Each input is regulated by Bit 4 (IRE) of the associated control register (CR0 controls $\overline{INT0}$, CR1 controls $\overline{INT1}$, etc). If IRE (Interrupt Enable) is set and a device input is asserted, an Interrupt Request open-collector output ($\overline{IRQ1}$–$\overline{IRQ7}$) is asserted. The asserted $\overline{IRQX}$ output is selected by the value programmed in Bits 0, 1, and 2 of the control register (L0, L1, and L2). This 3-bit field determines the interrupt request level as set by software.

Two or more interrupt sources can be programmed to the same request level. The corresponding $\overline{IRQX}$ output will remain asserted until multiple interrupt acknowledge cycles respond to all requests.

If the interrupt request level is set to zero, the interrupt is disabled because there is no corresponding IRQ output.

INTERRUPT ACKNOWLEDGE

The response of an Interrupt Handler to a bus interrupt request is an interrupt acknowledge cycle. The IACK cycle is initiated in the MC68153 by receiving $\overline{IACK}$ low. R/$\overline{W}$, A1, A2, A3 are latched, and the interrupt level on line A1–A3 is compared with any interrupt requests pending in the chip. Further activity can be one of four cases:

1. No further action required — This occurs if $\overline{IACKIN}$ is not asserted. Asserting $\overline{IACK}$ only starts the BIM activity. If the daisy chain signal never reaches the MC68153 ($\overline{IACKIN}$ is not asserted), another Interrupter has responded to the $\overline{IACK}$ cycle. The cycle will end, the chip $\overline{IACK}$ is negated, and no additional action is required.

2. Pass on the interrupt acknowledge daisy chain — For this case, $\overline{IACKIN}$ input is asserted by the preceding daisy chain Interrupter, and $\overline{IACKOUT}$ output is in turn asserted. The daisy chain signal is passed on when no interrupts are pending on a matching level or when any possible interrupts are disabled. The Interrupt Enable (IRE) bit of a control register can disable any interrupt requests, and in turn, any possible matches.

3. Respond internally — For this case, $\overline{IACKIN}$ is asserted and a match is found. The MC68153 completes the IACK cycle by supplying an interrupt vector from the proper vector register followed by a $\overline{DTACK}$ signal asserted. $\overline{IACKOUT}$ is not asserted because the interrupt acknowledge cycle is completed by this device.

   For the MC68153 to respond in this mode of operation, the EXTERNAL/$\overline{INTERNAL}$ control register bit (X/$\overline{IN}$) must be zero. For each source of interrupt request, the associated control register determines the BIM response to an IACK cycle, and the X/$\overline{IN}$ bit sets this response either internally (X/$\overline{IN}$ = 0) or externally (X/$\overline{IN}$ = 1).

4. Respond externally — For the final case, $\overline{IACKIN}$ is also asserted, a match is found and the associated control register has X/$\overline{IN}$ bit set to one. The MC68153 does not assert $\overline{IACKOUT}$ and does assert $\overline{INTAE}$ low. $\overline{INTAE}$ signals that the requesting device must complete the IACK cycle (supplying a vector and $\overline{DTACK}$) and that the 2-bit code contained on outputs INTAL0 and INTAL1 shows which interrupt source is being acknowledged.

These cases are discussed in more detail in the following paragraphs.

Internal Interrupt Acknowledge

For an internal interrupt acknowledge to occur, the following conditions must be met:

1. One or more device interrupt inputs ($\overline{INT0}$–$\overline{INT3}$) has been asserted and corresponding control bit IRE value is one.

2. $\overline{IACK}$ asserted.

3. A match exists between [A3, A2, A1] and the [L2, L1, L0] field of an enabled, requesting control register. If two or more devices are requesting at the same interrupt level, preference is given to the highest number requester, that is, $\overline{INT3}$ has highest priority and $\overline{INT0}$ has lowest.

4. Control register bit X/$\overline{IN}$ of matching interrupt source must be zero.

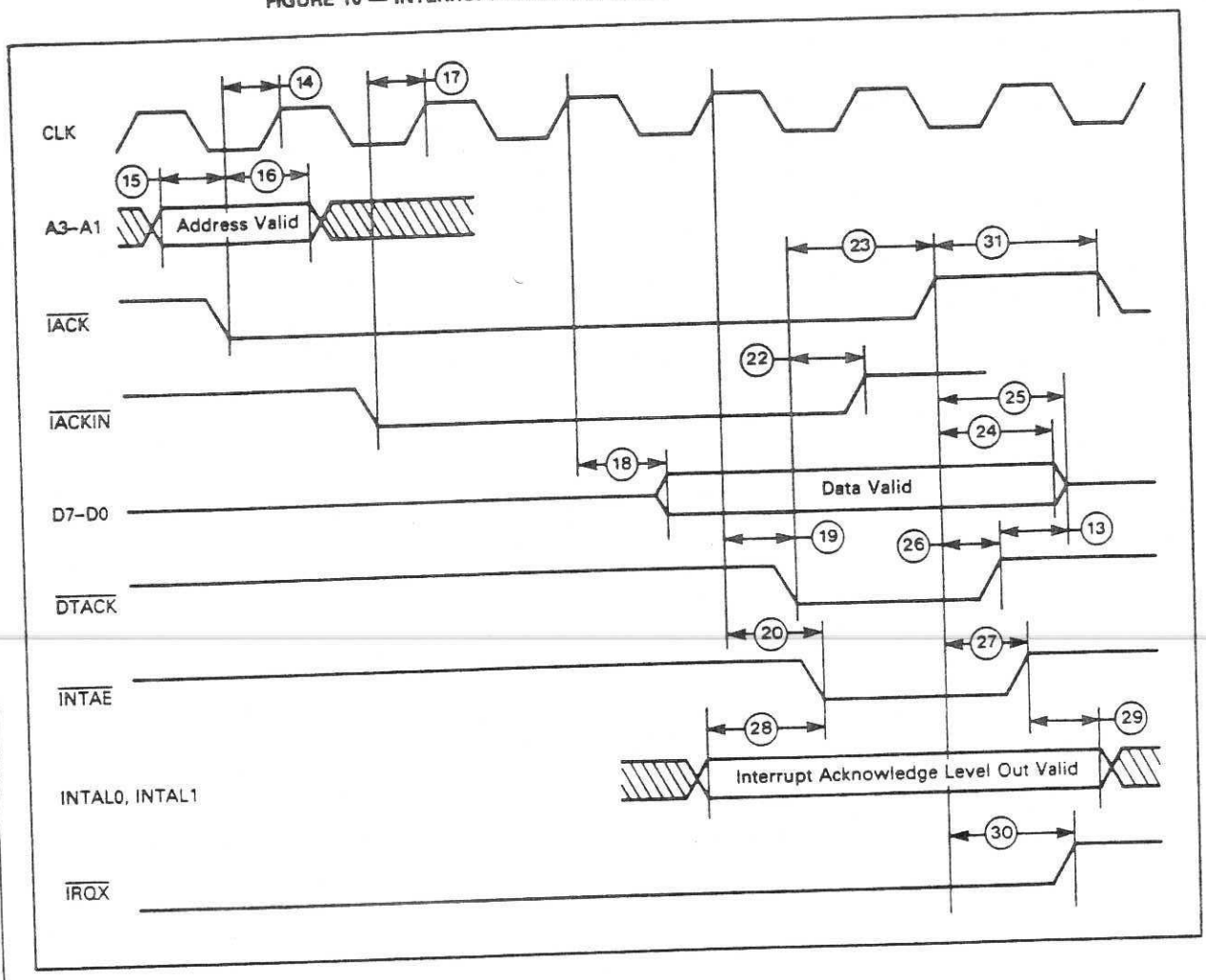
5. $\overline{IACKIN}$ asserted.

The internal interrupt acknowledge cycle timing is shown in Figure 10. The 8-bit interrupt acknowledge vector is presented to the data bus and $\overline{DTACK}$ is asserted. Note also that INTAL0 and INTAL1 are valid and $\overline{INTAE}$ is asserted during this cycle although they would normally not be used. The cycle is terminated (data and $\overline{DTACK}$ released) after $\overline{IACK}$ is negated.

During the IACK cycle, the INTERRUPT AUTO-CLEAR control bit (IRAC) comes into play. If the IRAC = one for the responding interrupt source, the INTERRUPT ENABLE (IRE) bit is automatically cleared during the IACK cycle, thus disabling the associated interrupt input and any $\overline{IRQX}$ output asserted due to this interrupt input. Before another interrupt can be requested from this source, IRE must be set to one by writing to the control register.

Note that $\overline{IACKOUT}$ is not asserted because this device is responding to the IACK and does not pass the daisy chain signal on. Also, new device interrupt requests occurring on $\overline{INT0}$–$\overline{INT3}$ after $\overline{IACK}$ is asserted are locked out to prevent any race conditions on the daisy chain.

FIGURE 10 — INTERRUPT ACKNOWLEDGE CYCLE — INTERNAL VECTOR



External Interrupt Acknowledge

For an external interrupt acknowledge, the same conditions as listed above are met with one exception. Control register bit X/$\overline{IN}$ of matching interrupt source must be set to one. The timing is shown in Figure 11. For this cycle, the interrupt vector and $\overline{DTACK}$ must be supplied by an external device. $\overline{INTAE}$ is asserted indicating that INTAL0 and INTAL1 are valid. The external device can use these signals to enable the vector and $\overline{DTACK}$. The cycle is terminated after $\overline{IACK}$ is negated.

The IRAC control bit acts in the external interrupt acknowledge the same as described for the internal response (see above). Also, $\overline{IACKOUT}$ is not asserted and new device interrupts are disabled for reasons discussed above.

Pass On IACK Daisy Chain

If the MC68153 has no interrupt request pending at the same level as the interrupt acknowledge, the IACK daisy chain signal is passed on to the next device if $\overline{IACKIN}$ is asserted. The following conditions are thus met:

1. $\overline{IACK}$ asserted.
2. No match exists between [A3, A2, A1] and the [L2, L1, L0] field of an enabled, requesting control register.
3. $\overline{IACKIN}$ is asserted.

$\overline{IACKOUT}$ is asserted if these conditions are valid. This output drives $\overline{IACKIN}$ of the next Interrupter on the daisy chain, passing the signal along. Figure 12 shows the timing for this case. $\overline{IACKOUT}$ is negated after $\overline{IACK}$ is negated.

FIGURE 11 — INTERRUPT ACKNOWLEDGE CYCLE — EXTERNAL VECTOR

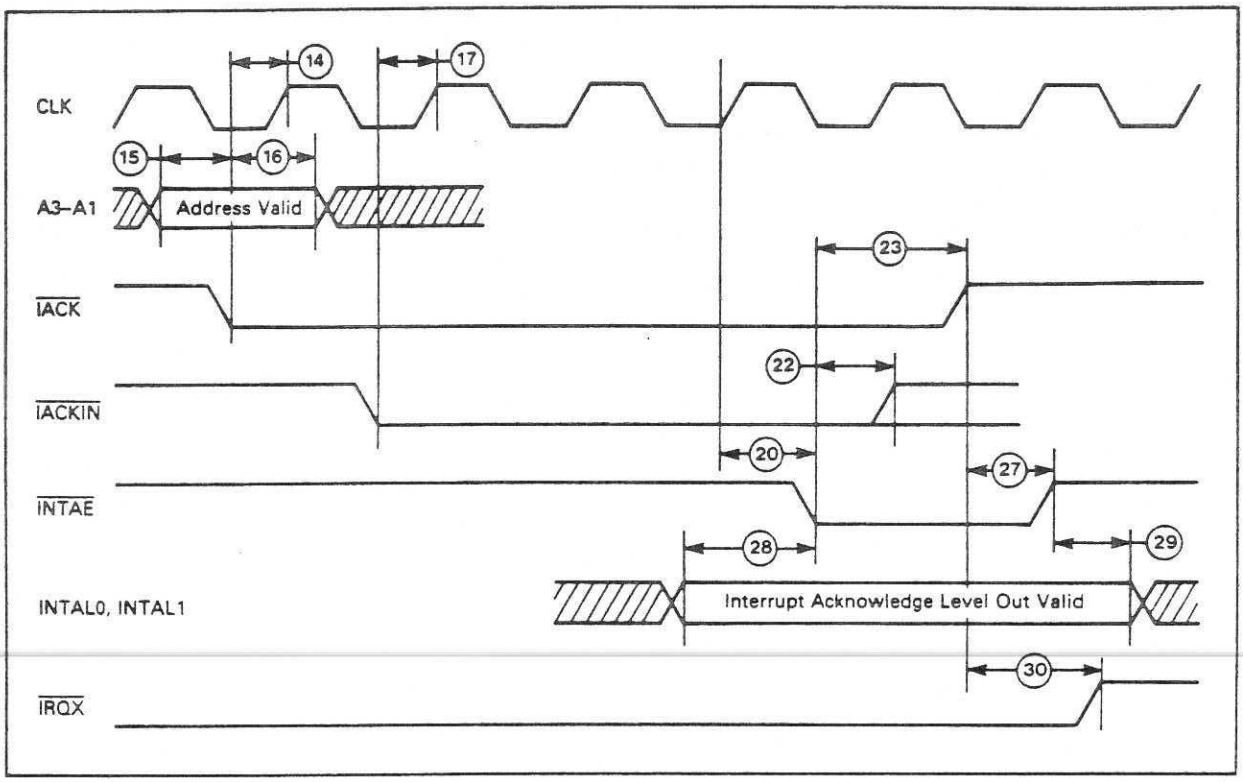
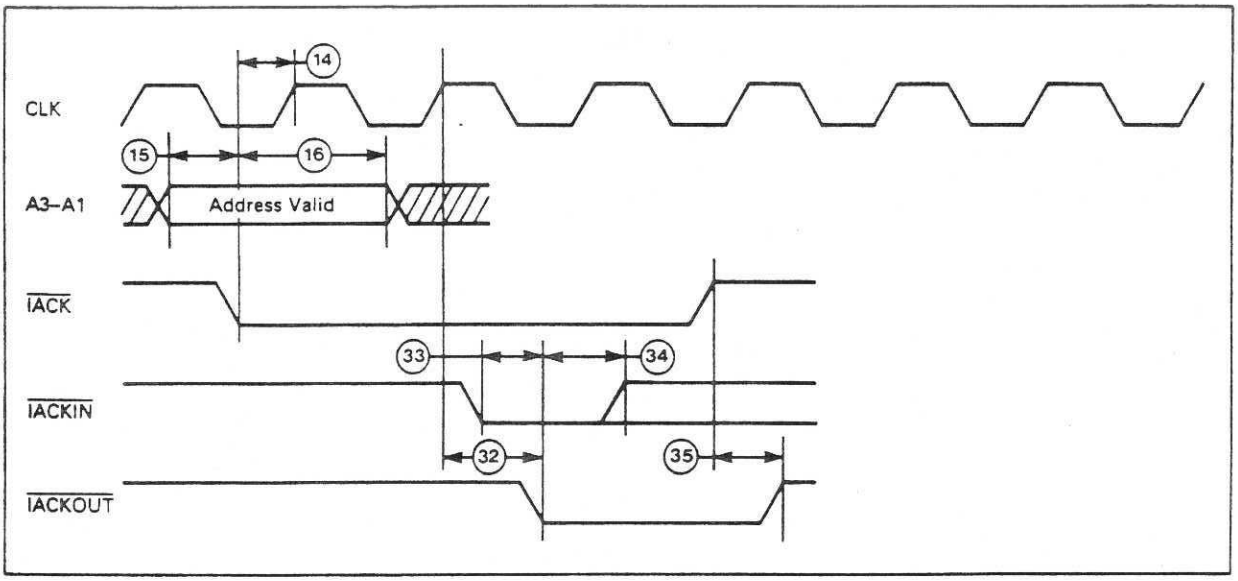


FIGURE 12 — INTERRUPT ACKNOWLEDGE CYCLE — $\overline{\text{IACKOUT}}$



CONTROL REGISTER FLAGS

Each control register contains a Flag bit (F) and a Flag Auto-Clear bit (FAC). Both bits can be read or altered via a register write without affecting the interrupt operation of the device. The Flag is useful as a status indicator for resource management and as a semaphor in multitasking or multiprocessor systems. Flag (F) is located in bit position 7 and can be used with the MC68000 Test and Set (TAS) instruction.

The Flag Auto-Clear (FAC) is used to manipulate the Flag bit. If the Flag is set to one and the FAC is also one, an interrupt acknowledge cycle to the associated interrupt source clears the Flag bit. This feature is useful in determining the interrupt status and passing messages.

RESET

There is no reset input, however, a chip reset is activated by asserting both $\overline{CS}$ and $\overline{IACK}$ simultaneously (Figure 13). These inputs should be held low for a minimum of two clock cycles for a full reset function. The control registers are reset to all zeroes and the Vector Registers are set to a value of $0F. This vector value is the uninitialized vector for the MC68000. See the MC68000 Users Manual for more details on this vector.

CLOCK

The chip clock is required for internal operation to occur. Typical frequency is 16 MHz in VMEbus and VERSAbus applications derived from the system clock. Any frequency can be used, however, up to 25 MHz (Figure 14).

FIGURE 13 — RESET

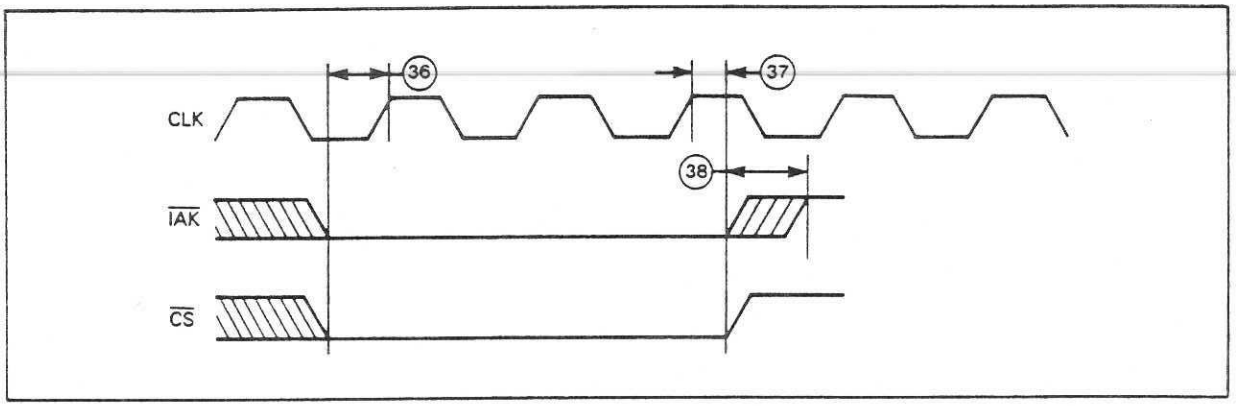


FIGURE 14 — CLOCK WAVEFORM

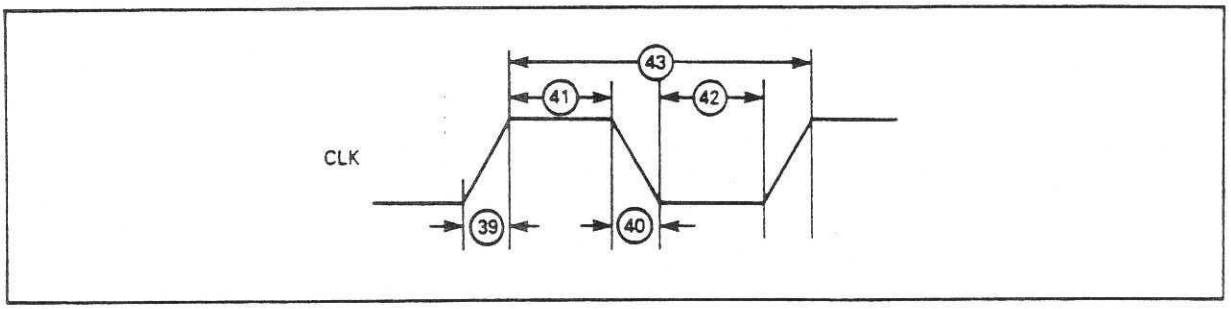


TABLE 1
AC PERFORMANCE SPECIFICATIONS
(V_{CC} = 5.0 V ±5%, T_A = 0°C to 70°C)

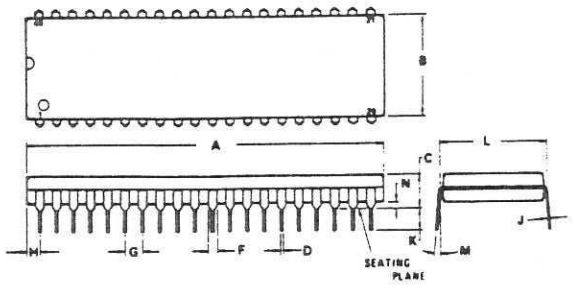
Number	Characteristic	Min	Max	Units	Notes
1	R/$\overline{W}$, A1–A3 Valid to $\overline{CS}$ Low (Setup Time)	10	—	ns	
2	$\overline{CS}$ Low to R/$\overline{W}$, A1–A3 Invalid (Hold Time)	5.0	—	ns	
3	$\overline{CS}$ Low to CLK High (Setup Time)	15	—	ns	1
4	CLK High to Data Out Valid (Delay)	—	55	ns	2
5	CLK High to $\overline{DTACK}$ Low (Delay)	—	40	ns	2
6	$\overline{DTACK}$ Low to $\overline{CS}$ High	0		ns	
7	$\overline{CS}$ High to $\overline{DTACK}$ High (Delay)	—	35	ns	10
8	$\overline{CS}$ High to Data Out Invalid (Hold Time)	0	—	ns	
9	$\overline{CS}$ High to Data Out High-Impedance (Hold Time)	—	50	ns	
10	$\overline{CS}$ High to $\overline{CS}$ or $\overline{IACK}$ Low	20	—	ns	
11	Data In Valid to $\overline{CS}$ Low (Setup Time)	10	—	ns	
12	$\overline{CS}$ Low to Data In Invalid (Hold Time)	5.0	—	ns	
13	$\overline{DTACK}$ High to Data Out High-Impedance	—	25	ns	10
14	$\overline{IACK}$ Low to CLK High (Setup Time)	15	—	ns	1
15	A1–A3 Valid to $\overline{IACK}$ Low (Setup Time)	10	—	ns	
16	$\overline{IACK}$ Low to A1–A3 Invalid (Hold Time)	5.0	—	ns	
17	$\overline{IACKIN}$ Low to CLK High (Setup Time)	15	—	ns	1, 8
18	CLK High to Data Out Valid (Delay)	—	55	ns	3
19	CLK High to $\overline{DTACK}$ Low (Delay)	—	40	ns	3
20	CLK High to $\overline{INTAE}$ Low (Delay)	—	40	ns	3
22	$\overline{DTACK}$ Low to $\overline{IACKIN}$ High	0	—	ns	8
23	$\overline{DTACK}$ Low to $\overline{IACK}$ High	0	—	ns	
24	$\overline{IACK}$ High to Data Out Invalid (Hold Time)	0	—	ns	
25	$\overline{IACK}$ High to Data Out High Impedance (Delay)	—	60	ns	
26	$\overline{IACK}$ High to $\overline{DTACK}$ High (Delay)	—	45	ns	10
27	$\overline{IACK}$ High to $\overline{INTAE}$ High (Delay)	—	35	ns	
28	INTAL0, INTAL1 Valid to $\overline{INTAE}$ Low (Setup Time)	1.0	2.0	CLK Per	
29	$\overline{INTAE}$ High to INTAL0, INTAL1 Invalid (Hold Time)	1.0	2.0	CLK Per	
30	$\overline{IACK}$ High to $\overline{IRQx}$ High (Delay)	—	50	ns	7, 10
31	$\overline{IACK}$ High to $\overline{IACK}$ or $\overline{CS}$ Low	20	—	ns	
32	CLK High to $\overline{IACKOUT}$ Low (Delay)	—	40	ns	5
33	$\overline{IACKIN}$ Low to $\overline{IACKOUT}$ Low (Delay)	—	30	ns	4, 8
34	$\overline{IACKOUT}$ Low to $\overline{IACKIN}$, $\overline{IACK}$ High	0	—	ns	8
35	$\overline{IACK}$ High to $\overline{IACKOUT}$ High (Delay)	—	35	ns	
36	$\overline{IACK}$ and $\overline{CS}$ both Low to CLK High (Setup Time)	15	—	ns	9
37	CLK High to $\overline{IACK}$ or $\overline{CS}$ High (Hold Time)	0	—	ns	
38	$\overline{IACK}$ or $\overline{CS}$ High to $\overline{IACK}$ and $\overline{CS}$ High (Skew)	—	1.0	CLK Per	6
39	Clock Rise Time	—	10	ns	
40	Clock Fall Time	—	10	ns	
41	Clock High Time	20	—	ns	
42	Clock Low Time	20	—	ns	
43	Clock Period	40	—	ns	

NOTES:

1. This specification only applies if the VBIM had completed all operations initiated by the previous bus cycle when $\overline{CS}$ or $\overline{IACK}$ was asserted. Following a normal bus cycle, all operations are completed within 2 clock cycles after $\overline{CS}$ or $\overline{IACK}$ have been negated. If $\overline{IACK}$ or $\overline{CS}$ is asserted prior to completion of these operations, the new cycle, and hence, $\overline{DTACK}$ is postponed.

   If the $\overline{IACK}$, $\overline{IACKIN}$ or $\overline{CS}$ setup time is violated, $\overline{DTACK}$ may be asserted as shown, or may be asserted one clock cycle later (i.e. $\overline{IACK}$ will not be recognized until the next rising edge of the clock).
2. Assumes that 3 has been met.
3. Assumes that 14 and 17 have both been met.
4. Assumes that 14 has been met. ($\overline{IACKOUT}$ cannot go low prior to $\overline{IACKIN}$ going low).
5. Assumes that 14 has been met and $\overline{IACKIN}$ has been low for at least the amount of time specified by 33.
6. 38 is the minimum skew between the last moment when both $\overline{IACK}$ and $\overline{CS}$ are asserted to when both are negated, to insure that an access cycle is not unintentionally started.
7. Assumes no other $\overline{INTx}$ input is causing $\overline{IRQx}$ to be driven low.
8. In non-daisy chain systems, $\overline{IACKIN}$ may be tied low.
9. Failure to meet this spec. causes RESET to be ignored for 1 clock period. It is then necessary to keep these signals low for 3 clock periods instead of 2.
10. Delay time is specified from Input signal to Open-Collector Output pulled High thru 1.0 kΩ resistor to +6.5 V.

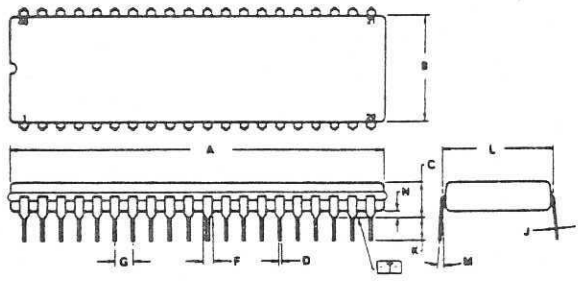
OUTLINE DIMENSIONS



DIM	MILLIMETERS MIN	MAX	INCHES MIN	MAX
A	51.69	52.45	2.035	2.065
B	13.72	14.22	0.540	0.560
C	3.94	5.08	0.155	0.200
D	0.36	0.56	0.014	0.022
F	1.02	1.52	0.040	0.060
G	2.54 BSC		0.100 BSC	
H	1.65	2.16	0.065	0.085
J	0.20	0.38	0.008	0.015
K	2.92	3.43	0.115	0.135
L	15.24 BSC		0.600 BSC	
M	0°	15°	0°	15°
N	0.51	1.02	0.020	0.040

NOTES
1. POSITIONAL TOLERANCE OF LEADS (D), SHALL BE WITHIN 0.25 mm (0.010) AT MAXIMUM MATERIAL CONDITION, IN RELATION TO SEATING PLANE AND EACH OTHER.
2. DIMENSION L TO CENTER OF LEADS WHEN FORMED PARALLEL.
3. DIMENSION B DOES NOT INCLUDE MOLD FLASH.

CASE 711-03
PLASTIC PACKAGE



DIM	MILLIMETERS MIN	MAX	INCHES MIN	MAX
A	51.31	53.24	2.020	2.096
B	12.70	15.49	0.500	0.610
C	4.06	5.84	0.160	0.230
D	0.38	0.56	0.015	0.022
F	1.27	1.65	0.050	0.065
G	2.54 BSC		0.100 BSC	
J	0.20	0.30	0.008	0.012
K	3.18	4.06	0.125	0.160
L	15.24 BSC		0.600 BSC	
M	5°	15°	5°	15°
N	0.51	1.27	0.020	0.050

NOTES
1. DIM A IS DATUM
2. POSITIONAL TOLERANCE FOR LEADS ⌖ Ø 0.25(0.010) Ⓜ T A Ⓜ
3. T IS SEATING PLANE
4. DIM L TO CENTER OF LEADS WHEN FORMED PARALLEL
5. DIMENSIONS A AND B INCLUDE MENISCUS
6. DIMENSIONING AND TOLERANCING PER ANSI Y14.5, 1973

CASE 734-04
CERAMIC PACKAGE

TYPICAL THERMAL CHARACTERISTICS

Package	θJA (Junction to Ambient) Still Air	Junction Temperature Still Air @ 70°C Ambient
L Suffix	40°C/W	147°C
P Suffix[1]	35°C/W	137°C

NOTES:
1. For reliable system operation the maximum allowable junction temperature (T_J) for plastic encapsulated packages has been limited to +140°C. Exceeding this limit will accellerate "wear-out" mechanisms associated with industry standard assembly methods using thermosonic ball bonds to attach gold (Aμ) bond wire to aluminum (Al) bond pads on the die surface.
2. At T_J = 140°C, time to 0.1% failure due to Aμ/Al interconnect = 8,920 Hours.

Motorola reserves the right to make changes without further notice to any products herein to improve reliability, function or design. Motorola does not assume any liability arising out of the application or use of any product or circuit described herein; neither does it convey any license under its patent rights nor the rights of others. Motorola and Ⓜ are registered trademarks of Motorola, Inc. Motorola, Inc. is an Equal Employment Opportunity/Affirmative Action Employer.

MOTOROLA *Semiconductor Products Inc.*

BOX 20912 • PHOENIX, ARIZONA 85036 • A SUBSIDIARY OF MOTOROLA INC.

Signetics

SCB68430
Direct Memory Access Interface (DMAI)

Preliminary Specification

Microprocessor Products

DESCRIPTION

The SCB68430 Direct Memory Access Interface (DMAI) is a single channel interface circuit which is intended to complement the performance and architectural capabilities of the SCN68000 microprocessor. The DMAI functions by transferring a series of operands (data) between memory and a device: operand sizes may be byte, word, or long word. A block is a sequence of operands: the number of operands in the block is determined by a transfer count stored within the DMAI. The SCB68430 can be programmed to utilize single cycle (cycle stealing) or burst data transfers.

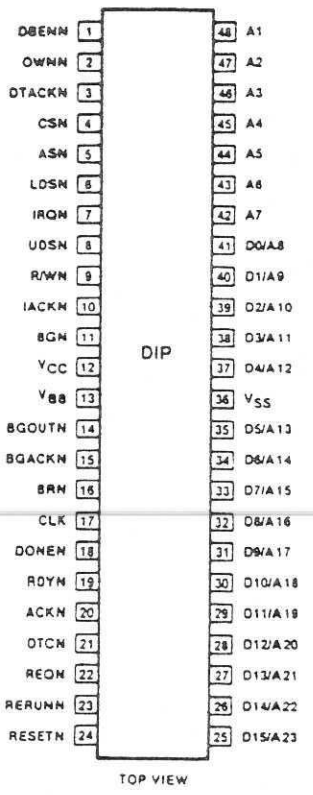
The DMAI provides two interfaces. The microprocessor interface is fully compatible with the SCN68000 microprocessor. The device interface includes lines for requesting, acknowledging, controlling, and timing the data transfers. The DMAI is a single-channel subset of the other 68000 family DMA controllers (68440 and 68450). It is software compatible with these devices and provides similar interfacing signals to both the system bus and the device.

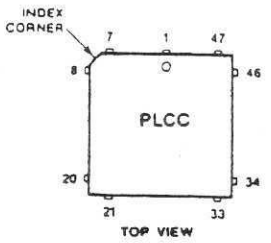
The SCB68430 is constructed using Signetics ISL bipolar technology.

FEATURES

- **Bus compatible with SCN68000 microprocessor**
- **Software compatible with other 68K family DMA controllers**
- **Single address transfers**
- **Cycle steal and burst mode operation**
- **Bus arbitration daisy chain**
- **Automatic rerun on bus error**
- **Supports 32-bit transfers for VME bus**
- **Supports SCN68000 vectored interrupts**
- **24-bit address counter**
- **16-bit transfer counter**
- **Maximum transfer rate of 5 Mbytes per second**
- **Signetics ISL bipolar technology**

PIN CONFIGURATION





Pin	Function	Pin	Function	Pin	Function
1	NC	19	CLK	36	D7/A15
2	DBENN	20	DONEN	37	D6/A14
3	OWNN	21	RDYN	38	D5/A13
4	DTACKN	22	ACKN	39	V_{SS}
5	CSN	23	DTCN	40	NC
6	ASN	24	REQN	41	D4/A12
7	LDSN	25	RERUNN	42	D3/A11
8	IRQN	26	RESETN	43	D2/A10
9	UDSN	27	NC	44	D1/A9
10	R/WN	28	D15/A23	45	D0/A8
11	IACKN	29	D14/A22	46	A7
12	BGN	30	D13/A21	47	A6
13	V_{CC}	31	D12/A20	48	A5
14	NC	32	D11/A19	49	A4
15	V_{BB}	33	D10/A18	50	A3
16	BGOUTN	34	D9/A17	51	A2
17	BGACKN	35	D8/A16	52	A1
18	BRN				

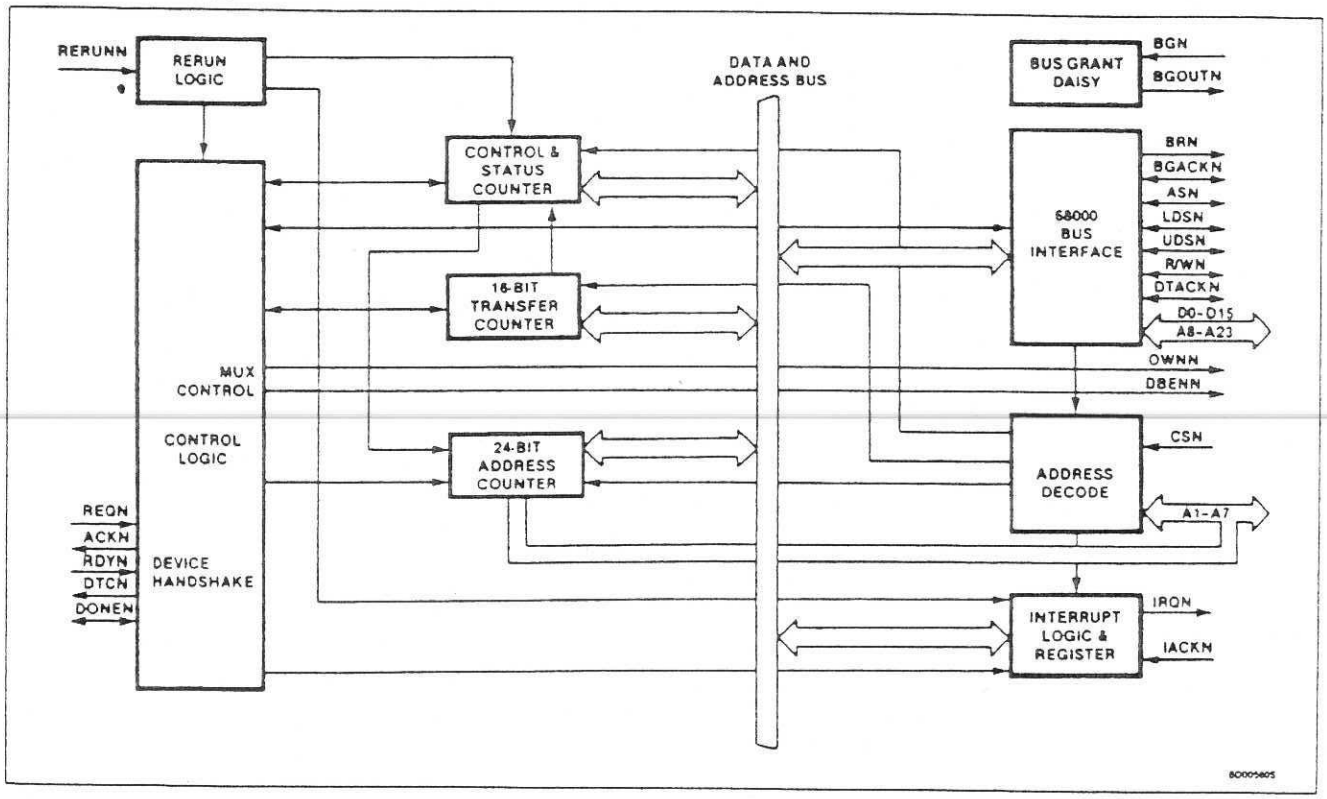
Direct Memory Access Interface (DMAI)

SCB68430

ORDERING CODE

PACKAGES	V_{CC} = 5V ± 5%, T_A = 0°C to 70°C	
	10MHz	12.5MHz
Ceramic DIP	SCB68430CAI48	SCB68430CCI48
Plastic DIP	SCB68430CAN48	SCB68430CCN48
Plastic LCC	SCB68430CAA52	SCB68430CCA52

BLOCK DIAGRAM



Direct Memory Access Interface (DMAI) SCB68430

PIN DESCRIPTION

MNEMONIC	PIN NO.	TYPE	DESCRIPTION
A1 – A7	48 – 42	I/O	**Address Lines:** Active high, three-state. In the MPU mode, these low order address lines specify which internal register of the DMAI is being accessed. In DMA mode, A1 – A7 are ouputs which provide the low order address bits of the location being accessed. Three-stated in IDLE Mode.
A8 – A23/ D0 – D15	41 – 37 35 – 25	I/O	**Address/Data Lines:** Active high, three-state. These lines are time multiplexed for data and address leads. The lines OWNN, RWN, CSN, and DBENN are used to control the demultiplexing of the address and data using external circuitry. In MPU mode, the bidirectional data lines (D0 – D15) are used to transfer data between the MPU and the DMAI. In the DMA mode, A8 – A23 provide the high order address bits of the location being accessed. Three-stated in IDLE mode.
ASN	5	I/O	**Address Strobe:** Active low, three-state. In MPU and IDLE modes, ASN is an input which indicates that the current bus master has placed a valid address on the bus. It is monitored by the DMAI during bus arbitration to ascertain that the previous bus master has completed the current bus cycle. In DMA mode, it is an output indicating that the DMAI has placed a valid address on the bus.
UDSN	8	I/O	**Upper Data Stobe:** Active low, three-state. In MPU and IDLE modes, UDSN is an input which indicates that the upper data byte of the addressed word is being addressed. In DMA mode, it is an output with the same meaning.
LDSN	6	I/O	**Lower Data Strobe:** Active low, three-state. In MPU and IDLE modes, LDSN is an input which indicates that the lower data byte of the addressed word is being addressed. In DMA mode, it is an output with the same meaning.
R/WN	9	I/O	**Read/Write:** Active high for read, low for write, three-state. In MPU mode, R/WN is an input which controls the direction of data flow through the DMAI's input/output data bus interface and, if required, through an external data bus buffer. R/WN high causes the DMAI to place the data from the addressed register on the data bus, while R/WN low causes the DMAI to accept data from the data bus. In DMA mode, R/WN is an output to memory and I/O controllers indicating the type of bus cycle. It is held three-stated during IDLE mode.
CSN	4	I	**Chip Select:** Active low. When low, places the DMAI into the MPU mode. This input signal is used to select the DMAI for programmed data transfers. These transfers take place over D0 – D15 as controlled by the R/WN and A1 – A7 inputs. The DMAI is deselected when CSN is high. CSN is ignored during DMA mode.
DTACKN	3	I/O	**Data Transfer Acknowledge:** Active low, three-state. In MPU mode, DTACKN is asserted on a write cycle to indicate that the data on the bus has been latched, and on a read cycle or interrupt acknowledge cycle to indicate that valid data is present on the bus. The signal is negated (driven high) when completion of the cycle is indicated by negation of the CSN or IACKN input, and returns to the inactive third state a short time after it is negated. In DMA Mode, DTACKN is an input monitored by the DMAI to determine when the addressed device (memory) has latched the data (write cycle) or put valid data on the bus (read cycle).
RESETN	24	I	**Master Reset:** Active low. Assertion of this pin clears internal control registers (See table 1), initializes the interrupt vector register to H'0F', and sets the status register to the default value B'0000 000X', where X is the state of RDYN. All bidirectional I/O lines are three-stated and the DMAI is placed in the IDLE mode.
CLK	17	I	**Clock:** Active high. Usually the system clock, but may be any clock meeting the electrical specifications. Used by the DMAI to synchronize device functions and external control lines, and may not be gated off at any time.
IRQN	7	O	**Interrupt Request:** Active low, open collector. This output is asserted, if interrupts are enabled, upon end of transfer, on occurrence of a bus error, and on receipt of an abort from the MPU. The CPU can read the status register to determine the interrupting condition(s), or can respond with an interrupt acknowledge cycle to cause the DMAI to output an interrupt vector on the data bus.
IACKN	10	I	**Interrupt Acknowledge:** Active low. When asserted, indicates that the current cycle is an interrupt acknowledge cycle. The DMAI normally responds by placing the contents of the interrupt vector register on the data bus and asserting DTACKN. IACKN is not serviced if the DMAI has not generated an interrupt request.
BRN	16	O	**Bus Request:** Active low, open collector. BRN is asserted by the DMAI to request ownership of the bus after a DMA request is sensed on the REQN input from the I/O device. It is negated when the bus has been granted (BGN low) and BGACKN has been asserted, or, in burst DMA request mode, if the I/O device negates its request at least one clock cycle before BGACKN is asserted.
BGN	11	I	**Bus Grant:** Active low. BGN indicates to the DMAI that it is to be the next bus master. This signal is originated by the MPU and propagated via a daisy chain or other prioritization mechanism. After BGN is asserted, the DMAI waits until DTACKN, ASN, and BGACKN have become inactive before assuming ownership of the bus by asserting BGACKN.

PIN DESCRIPTION (Continued)

MNEMONIC	PIN NO.	TYPE	DESCRIPTION
BGOUTN	14	O	**Bus Grant Output:** Active low. Daisy chain output which is asserted by the DMAI when BGN is asserted and the DMAI does not have a bus request pending.
BGACKN	15	I/O	**Bus Grant Acknowledge:** Active low, three-state. As an input, BGACKN is monitored by the DMAI during the bus arbitration cycle to determine when it can assume ownership of the bus (BGACKN negated). In DMA mode, it is asserted by the DMAI to indicate that it is the bus master. Three-stated in MPU and IDLE modes.
RERUNN	23	I	**Rerun:** Active low. This input is asserted by external error detect logic to indicate a bus error. In DMA mode, the DMAI stops operation and three-states the data, address, and control lines, except BGACKN. It remains halted until RERUNN becomes inactive, and then re-tries the last bus cycle. If RERUNN is asserted again, the DMAI sets the ERR bit in the status register, stops DMA operation, releases the bus, and interrupts the CPU, if interrupts are enabled, responding with a special interrupt vector when IACKN is asserted. Not monitored in MPU and IDLE modes.
REQN	22	I	**DMA Request:** Active low. This input from the I/O device requests service from the DMAI and causes the DMAI to request control of the bus. In burst mode, the input is level sensitive, and the DMAI releases the bus after REQN is negated and the current DMA cycle is completed. In cycle steal mode, the REQN input is negative edge triggered. A negative going edge must occur at least one clock cycle before DTCN is asserted to accomplish continuous transfer cycles but not earlier than beginning of master cycle.
ACKN	20	O	**DMA Request Acknowledge:** Active low. ACKN is asserted by the DMAI to indicate that it has gained the bus and the requested bus cycle is now beginning. It is asserted at the beginning of every bus cycle after ASN has been asserted, and is negated at the end of every bus cycle.
RDYN	19	I	**Device Ready:** Active low. RDYN is asserted by the requesting device to indicate to the DMAI that valid data has either been stored or put on the bus. If negated, it indicates that the data has not been stored or presented, causing the DMAI to enter wait states. RDYN can be held low continuously if the device is fast enough so that wait states are not required.
DTCN	21	O	**Device Transfer Complete:** Active low. In DMA mode, DTCN is asserted by the DMAI to indicate to the device that the requested data transfer is complete. On a write to memory operation, it indicates that the data provided by the device has been successfully stored. On a read from memory operation, it indicates to the device that the data from memory is present on the data bus and should be latched.
DONEN	18	I/O	**Done:** Active low, open collector. As an output, DONEN is asserted by the DMAI concurrent with the ACKN output to indicate to the device that the transfer count is exhausted and that the DMAI's operation is completed as a result of that transfer. As an input, if asserted by the device before the transfer count became zero, it causes the DMAI to abort service and generate an interrupt request, if interrupts are enabled.
OWNN	2	O	**Own:** Active low, open collector. This output is asserted by DMAI during the DMA mode to indicate bus mastership. It can be used to enable external address/data and control buffers. Inactive in MPU and IDLE modes.
DBENN	1	O	**Data Bus Enable:** Active low, open collector. Asserted by the DMAI when CSN is asserted or when IACKN is asserted and the DMAI has an interrupt request pending. Can be used to enable bidirectional data buffers for D0 – D15. Inactive in IDLE and DMA mode.
V_{CC}	12	I	**Power Supply:** +5 volt power input.
V_{BB}	13	I	**Power Supply:** +1.5 volt power input.
V_{SS}	36	I	**Ground:** Signal and power ground input.

PIN DESCRIPTION

The Pin Description table describes the function of each of the pins of the DMAI. Signal names ending in 'N' are active low. All other signals are active high. In the descriptions, 'MPU mode' refers to the state when the DMAI is chip selected. The term 'DMA mode' refers to the state when the DMAI assumes ownership of the bus. The DMAI is in the 'IDLE mode' at all other times.

In this data sheet signals are discussed using the terms 'active' and 'inactive' or 'asserted' and 'negated' independent of whether the signal is active in the high (logic one) state or the low (logic zero) state. Refer to the individual pin descriptions for the definition of the active level of each signal.

REGISTERS AND COUNTERS

Register Map

The internal accessible register organization of the DMAI is shown in table 1. The following rules apply to all registers:

1. A read from a reserved location in the map results in a read from the 'null register'. The null register returns all ones for data and results in a normal bus cycle. A write to one of these locations results in a normal bus cycle but no write occurs.
2. Unused bits of a defined register are read as indicated in the register descriptions.
3. All registers are addressable as 8-bit quantities. To facilitate operation with the 68K MOVEP instruction, addresses are ordered such that certain sets of registers may also be accessed as words or long words.

The operation of the DMAI is programmed by writing control words into the appropriate registers. Operational feedback is provided via status registers which can be read by the CPU. The contents of certain control and status registers are initialized on RESET.

To provide compatibility with the other 68K family DMA controllers, control and status bits are mapped in bit positions equivalent to where they are located in the register map of the other devices. Bits which are used in the other devices but not in the DMAI are assigned default values. If upward compatibility to the other controllers is required, the programmer should use these default values when writing the control words to the registers, although they have no effect in the DMAI. When a register is read, the default value is returned regardless of the value used when the register is programmed. The default value is indicated by '(x)' in unused bit positions in the register formats, which are illustrated in table 2.

Table 1. DMAI ADDRESS MAP

ADDRESS BITS[1,2] 7 6 5 4 3 2 1 0	ACRONYM	REGISTER NAME	MODE	AFFECTED BY RESET
d d 0 0 0 0 0 0	CSR	Channel Status Register	R/W[3]	Yes
d d 0 0 0 0 0 1	CER	Channel Error Register	R	Yes
d d 0 0 0 0 1 0		Reserved		
d d 0 0 0 0 1 1		Reserved		
d d 0 0 0 1 0 0	DCR	Device Control Register	R/W	Yes
d d 0 0 0 1 0 1	OCR	Operation Control Register	R/W	Yes
d d 0 0 0 1 1 0	SCR	Sequence Control Register	R/W[4]	No
d d 0 0 0 1 1 1	CCR	Channel Control Register	R/W	Yes
d d 0 0 1 0 0 0		Reserved		
d d 0 0 1 0 0 1		Reserved		
d d 0 0 1 0 1 0	MTCH	Memory Transfer Counter High	R/W	No
d d 0 0 1 0 1 1	MTCL	Memory Transfer Counter Low	R/W	No
d d 0 0 1 1 0 0	MACH	Memory Address Counter High	R/W[4]	No
d d 0 0 1 1 0 1	MACMH	Memory Address Counter Middle High	R/W	No
d d 0 0 1 1 1 0	MACML	Memory Address Counter Middle Low	R/W	No
d d 0 0 1 1 1 1	MACL	Memory Address Counter Low	R/W	No
d d 0 1 d d d d		Reserved		
d d 1 0 0 0 d d		Reserved		
d d 1 0 0 1 0 0		Reserved		
d d 1 0 0 1 0 1	IVR	Interrupt Vector Register – Normal	R/W	Yes
d d 1 0 0 1 1 0		Reserved		
d d 1 0 0 1 1 1	IVR	Interrupt Vector Register – Error	R/W	Yes
d d 1 0 1 0 d d		Reserved		
d d 1 0 1 1 0 0		Reserved		
d d 1 0 1 1 0 1	CPR	Channel Priority Register	R/W[4]	No
d d 1 0 1 1 1 0		Reserved		
d d 1 0 1 1 1 1		Reserved		
d d 1 1 d d d d		Reserved		

NOTES:
1. A0 = 0 for UDSN asserted, A0 = 1 for LDSN asserted.
2. 'd' designates don't care.
3. A write to this register may perform a status resetting operation.
4. This register is a dummy register present only to provide compatibility with other 68K family DMA controllers. A write to this register has no effect on the DMAI.

Table 2. REGISTER BIT FORMATS

DEVICE CONTROL REGISTER

	BIT15	BIT14	BIT13	BIT12	BIT11	BIT10	BIT09	BIT08
DCR	EXTERNAL REQUEST MODE 0 = BURST 1 = CYCLE STEAL	NOT USED (0)	NOT USED (1)	NOT USED (1)	NOT USED (*)	NOT USED (0)	NOT USED (0)	NOT USED (0)

*Should be programmed as '0' for SIZE (OCR[5:4]) = 00 and as '1' otherwise. When read, the value of this bit is OCR[5]. .OR.OCR[4].

OPERATION CONTROL REGISTER (OCR)

	BIT07	BIT06	BIT05	BIT04	BIT03	BIT02	BIT01	BIT00
OCR	DIRECTION 0 = MEM TO DEV 1 = DEV TO MEM	NOT USED (0)	OPERAND SIZE 00 = BYTE 01 = WORD (16 BIT) 10 = LONG WORD* 11 = WORD (32-BIT)*		NOT USED (0)	NOT USED (0)	NOT USED (1)	NOT USED (0)

*Long word and 32-bit word modes are not supported by 68440. 32-bit word mode is not supported by 68450.

SEQUENCE CONTROL REGISTER (SCR)

	BIT15	BIT14	BIT13	BIT12	BIT11	BIT10	BIT09	BIT08
SCR	NOT USED (0)	NOT USED (0)	NOT USED (0)	NOT USED (0)	NOT USED (0)	NOT USED (1)	NOT USED (0)	NOT USED (0)

CHANNEL CONTROL REGISTER (CCR)

	BIT07	BIT06	BIT05	BIT04	BIT03	BIT02	BIT01	BIT00
CCR	START 0 = NO 1 = YES	NOT USED (0)	NOT USED (0)	SOFTWARE ABORT 0 = NO 1 = YES	INTERRUPT ENABLE 0 = NO 1 = YES	NOT USED (0)	NOT USED (0)	NOT USED (0)

CHANNEL STATUS REGISTER (CSR)

	BIT15	BIT14	BIT13	BIT12	BIT11	BIT10	BIT09	BIT08
CSR	CHANNEL OPERATION COMPLETE 0 = NO 1 = YES	NOT USED (0)	NORMAL DEVICE TERMINATE 0 = NO 1 = YES	ERROR 0 = NO 1 = YES	CHANNEL ACTIVE 0 = NO 1 = YES	NOT USED (0)	NOT USED (0)	READY INPUT STATE 0 = LOW 1 = HIGH

CHANNEL ERROR REGISTER (CER)

	BIT07	BIT06	BIT05	BIT04	BIT03	BIT02	BIT01	BIT00
CER	NOT USED (0)	NOT USED (0)	NOT USED (0)	ERROR CODE 00000 = NO ERROR 01001 = BUS ERROR 10001 = SOFTWARE ABORT				

CHANNEL PRIORITY REGISTER (CPR)

	BIT07	BIT06	BIT05	BIT04	BIT03	BIT02	BIT01	BIT00
CPR	NOT USED (0)	NOT USED (0)	NOT USED (0)	NOT USED (0)	NOT USED (0)	NOT USED (0)	NOT USED (0)	NOT USED (0)

Device Control Register (DCR)

[15] External Request Mode

This bit selects whether the DMAI operates in burst or cycle steal mode.

0 Burst mode. This mode allows a device to request the transfer of multiple operands using consecutive bus cycles. In this mode the request (REQN) line is an active low input which is asserted by the device to request an operand transfer. The DMAI services the request by arbitrating for the bus, obtaining the bus, and notifying the peripheral by asserting the acknowledge (ACKN) output. If the request line is active when the DMAI asserts ACKN, and remains active at least until the DMAI asserts device transfer complete (DTCN), the DMAI recognizes a valid request for another operand, which will be transferred during the next bus cycle. If the request line is negated before the DMAI asserts DTCN, the DMAI relinquishes the bus and waits for the next request, but the current transfer will be completed.

1 Cycle steal mode. In this mode, the device requests an operand transfer by generating a falling edge on the request (REQN) line. The DMAI services the request by arbitrating for the bus, obtaining the bus, and notifying the peripheral by asserting the acknowledge (ACKN)

output. The request line must be in the inactive state for at least one clock cycle before a request is made. After a request has been asserted, it must remain at the assertion level for at least one clock cycle. If another request is received before the first operand part of a former request is acknowledged, the second request is not recognized. Normally, the DMAI will relinquish the bus after servicing a valid request. However, if the device generates a new request before the DMAI asserts DTCN for the last operand part, the DMAI will retain ownership of the bus and that request will be serviced before the bus is relinquished.

Operation Control Register (OCR)

[7] Direction

0 Transfer is from memory to device.

1 Transfer is from device to memory.

[5:4] Operand Size

The programming of these bits determine whether UDSN, LDSN, or both are generated during the transfer cycle and the increment by which the memory address counter (MAC) is changed in each transfer cycle.

00 Byte. The operand size is 8 bits. The MAC is incremented by one after each operand transfer. If the LSB of the MAC is a '0', UDSN is asserted during the transfer. If the LSB of a MAC is a '1', LDSN is asserted during the transfer. The transfer counter decrements by one before each byte is transferred.

01 Word. The operand size is 16 bits. The MAC is incremented by two after each operand transfer. The value of the LSB of the MAC is ignored and both UDSN and LDSN are asserted during the transfer. The transfer counter decrements by one before each word is transferred.

10 Long word. The operand size is 32 bits. The operand is transferred as two 16-bit words. The MAC is incremented by two after each 16-bit word is transferred. The value of the LSB of the MAC is ignored and both UDSN and LDSN are asserted during the transfer. The transfer counter decrements by one before the entire long word is transferred. Note that this mode is not implemented in the 68440.

11 Double word. The operand size is 32 bits. The operand is transferred as a single 32-bit word. The MAC is incremented by four after each operand transfer. The value of the two LSBs of the MAC is ignored (the A1 output will always be a zero in this mode) and both UDSN and LDSN are asserted during the transfer. The transfer counter decrements by one before the double word is transferred. Note that this mode is not implemented in the 68440 or 68450; it is included in the DMAI to support VME bus operations.

Sequence Control Register (SCR)

This register serves no function in the DMAI. It is included only to provide compatibility with the programming for the 68440 and 68450 DMA controllers.

Channel Control Register (CCR)

[7] Start Operation

0 No start pending.

1 Start operation. The start bit is set to initiate operation of the DMAI. The memory address counter and the memory transfer counter should have been previously initialized, and all bits of the channel status register (CSR) should have previously been reset. The DMAI initiates operation by clearing any pending requests, clearing the start bit, and setting the channel active bit in the CSR. The DMAI is then ready to receive requests for an operation. The channel cannot be started if any of the internal status bits in the CSR (CSR[15:11]) have not been cleared.

A pending start cannot be reset by a write to the register. START can be cleared only by the DMAI when it starts operation or by setting the software abort bit (CCR[4]).

[4] Software Abort

0 Do not abort.

1 Abort operation. Setting this bit terminates the current operation of the DMAI and places it in the IDLE state. The channel operation complete and error bits in the CSR are set, the channel active bit in the CSR is reset, and an ABORT ERROR condition is signaled in the CER. Setting this bit causes a pending start to be reset.

[3] Interrupt Enable

0 Interrupts not enabled.

1 Enable interrupts. An interrupt request is generated if the channel operation complete bit in the CSR is set. When the IACKN input is asserted, the DMAI returns the normal interrupt vector if the error bit in the CSR is not set, or the error interrupt vector if error is set.

Channel Status Register (CSR)

A read of this register provides the status of the DMAI. The COC, NDT, and ERR bits can be cleared by writing a '1' to the bit positions of the register which are to be cleared. Those bit positions which are written with a '0' remain unaffected.

[15] Channel Operation Complete

This bit is set following the termination, whether successful or not, of any DMAI operation and indicates that the DMA transfer has completed. This bit must be cleared to start another channel operation.

[13] Normal Device Termination

This bit is set when the device terminates the DMAI operation by asserting the DONEN line while the device was being acknowledged. This bit must be cleared to start another channel operation.

[12] Error

This bit is used to report that the DMAI's operation was terminated due to the occurrence of an error. The condition which caused the error can be determined by reading the channel error register (CER). This bit must be cleared to start another channel operation. When this bit is cleared, the CER is also cleared.

[11] Channel Active

This bit is set after the channel has been started and remains set until the channel operation terminates. It is then automatically cleared by the DMAI. The bit is unaffected by the write operations.

[8] Ready Input State

This bit reflects the state of the RDYN input at the time the CSR is read. The bit is a '0' if RDYN is low and a '1' if RDYN is high. This bit is unaffected by write or reset operations.

Channel Error Register (CER)

[4:0] Error Code

This field indicates the source of error when an error is indicated in CER[12]. The contents of this register are cleared when CER[12] is cleared.

00000 No error.

01001 Bus error. A bus error occurred during the last bus cycle generated by the DMAI. See rerun description in OPERATION section.

10001 Software abort. The channel operation was terminated by a software abort command. See CCR[4].

Channel Priority Register (CPR)

This register serves no function in the DMAI. It is included only to provide compatibility with the programming for the other 68K family DMA controllers.

Memory Address Counter (MACH, MACMH, MACML, MACL)

The 32-bit memory address counter is used to program the memory location where the first operand to be transferred is located or is to be transferred to, depending on the direction of transfer. The counter must be initialized prior to beginning the transfer of a block of data and then increments automatically depending on the operand length, as de-

scribed in the Operation Control Register description.

Only the least significant 24 bits of the counter (MACMH, MACML, and MACL) are implemented in the DMAI. The most significant byte of the counter, MACH, is provided only to allow compatibility with programming of the 68440 and 68450. Writing to MACH has no effect on the DMAI operation. Reading MACH always returns H'00'.

Memory Transfer Counter (MTCH, MTCL)

The 16-bit memory transfer counter programs the number of operands to be transferred by the DMAI. The counter must be initialized prior to beginning the transfer of a block of data and then decrements once per operand transfer (regardless of operand size) until it reaches the terminal value of zero. Channel operation then terminates and the COC bit in the CSR will be asserted.

Interrupt Vector Register (IVR)

The IVR contains the value to be placed on the data bus upon receipt of an interrupt acknowledge from the MPU. Only the seven most significant bits of the programmed value are used by the DMAI. The output vector from the DMAI contains a zero in the least significant bit position if a normal termination occurred (error bit not set) and contains a one in the least significant bit position if termination was due to an error (error bit set).

The contents of this register are initialized to H'0F' by a reset. The value returned will be H'0F', regardless of the error state, until the register is programmed by the MPU.

To provide compatibility with the other 68K family DMA controllers, the IVR has two addresses (see table 1). If program compatibility is required, the value written at the normal IVR address should have a zero as its LSB, and the value written at the error IVR address should be the same but with the LSB equal to one.

OPERATION

A DMAI operation proceeds in three principal phases. During the initialization phase, the MPU configures the channel control registers, loads the initial memory address and transfer count, and starts the channel. During the transfer phase, the DMAI accepts requests for transfers from the device, arbitrates for and acquires ownership of the bus, and provides for addressing and bus controls for the transfers. The termination phase occurs after the operation is complete, when the DMAI reports the status of the operation.

Operation Initiation

After having programmed the control registers, the memory address counter, and the memory transfer counter, the MPU sets the start bit (CCR[7]). The DMAI initiates the operation by clearing any pending requests, clearing the start bit, and setting the channel active bit in the CSR. The DMAI is then ready to receive valid requests for an operation.

The channel cannot be started if any of the internal status bits in the CSR (CSR[15:11]) have not been cleared. An error is not signaled if this condition occurs. The only indication of this state is that the start bit remains set in the CCR. A pending start cannot be reset by a write to the register. START can be cleared only by the DMAI when it starts operation or by setting the software abort bit (CCR[4]).

Device/DMAI Communication

Communication between the peripheral device and the DMAI is accommodated by five signal lines:

Request (REQN)

The device makes a request for service by asserting the request line. The DMAI can operate in either the burst request mode or the cycle stealing request mode, as programmed by the external request mode bit (DCR[15]).

The burst mode allows a device to request the transfer of multiple operands using consecutive bus cycles. In this mode the request line is an active low input. The DMAI services the request by arbitrating for the bus, obtaining the bus, and notifying the peripheral by asserting the acknowledge (ACKN) output. If the request line is active when the DMAI asserts ACKN, and remains active at least until the DMAI asserts device transfer complete (DTCN), the DMAI recognizes a valid request for another operand, which will be transferred during the next bus cycle. If the request line is negated before the DMAI asserts DTCN, the DMAI relinquishes the bus and waits for the next request. For long word transfers (2 x 16), the request must be asserted at least until the acknowledge for the second part of the operand has been asserted.

In the cycle steal mode, the device requests an operand transfer by generating a falling edge on the request line. The DMAI services the request by arbitrating for the bus, obtaining the bus, and notifying the peripheral by asserting the acknowledge (ACKN) output. The request line must be in the inactive state for at least one clock cycle before a request is made. After a request has been asserted, it must remain at the assertion level for at least one clock cycle. If another request is received before the first operand part of a former request is acknowledged, the second request is not recognized. Normally, the DMAI will relinquish the bus after servicing a valid request. However, if the device generates a new request before the DMAI asserts DTCN for the last operand part, the DMAI will retain ownership of the bus and that request will be serviced before the bus is relinquished.

Acknowledge (ACKN)

The DMAI asserts the acknowledge line, which implicitly addresses the device making the request, during transfers to and from the device. The line may be used to control buffering circuits between the data bus and the MPU bus.

During burst mode, REQN must not be disasserted for less than one CLK period plus four RC time constants, where R is the value of the resistor used for the pullup on BRN and C has a typical value of 20pF.

Ready (RDYN)

Ready is an active low input which is asserted by the requesting device to indicate to the DMAI that valid data has either been stored or put on the bus. If negated, it indicates that the data has not been stored or presented, causing the DMAI to enter wait states until RDYN is asserted. RDYN can be held low continuously if the device is fast enough so that wait states are not required. The current state of the ready input is reflected in CSR[8].

Done (DONEN)

Done is a bidirectional active low signal. As an output, it is asserted and negated by the DMAI concurrent with the ACKN output of the last operand part to indicate to the device that the memory transfer count is exhausted and that the DMAI's operation is completed as a result of that transfer.

The DMAI also monitors the state of the line while acknowledging a device. If the device asserts DONEN, the DMAI will terminate operation after the transfer of the current operand. In this case the DMAI clears the channel active bit and sets the channel operation complete and normal device termination bits in the CSR. If both the DMAI and the device assert DONEN, the device termination is not recognized, but the operation does terminate.

Device Transfer Complete (DTCN)

DTCN is an active low output which is asserted by the DMAI to indicate to the device that the requested data transfer is complete. On a write to memory operation, it indicates that the data provided by the device has been successfully stored. On a read from memory operation, it indicates to the device that the data from memory is present on the data bus and should be latched. DTCN is not asserted if assertion of the RERUNN input terminates the bus cycle.

Bus Arbitration

Upon receiving a valid request for a transfer from the device, the DMAI will arbitrate for and obtain ownership of the system bus.

The DMAI indicates that it wishes to become the bus master by asserting its bus request (BRN) output. This is a wire-ORed signal that indicates to the MPU that some external device requires control of the bus. The processor is effectively at a lower priority level than external devices and will relinquish the bus after it has completed the last bus cycle it has started. The processor puts the bus up for external arbitration by asserting its bus grant (BGN) output. This signal may be routed through a daisy chain (such as provided by the DMAI) or through some other priority-encoded network. When the DMAI making the bus request receives the bus grant (indicated by its BGN input being asserted), it is to be the next bus master. It waits until address strobe (ASN), data transfer acknowledge (DTACKN) and bus grant acknowledge (BGACKN) become inactive and then assumes ownership of the bus by asserting its own BGACKN output. The DMAI then negates the BRN output and proceeds with the data transfer phase. After this phase is completed, the DMAI relinquishes bus ownership by negating the BGACKN output.

In burst DMA mode, detection of an active low request input after the DMAI operation has been started will begin the bus arbitration cycle. However, if the device negates its request at least one clock cycle before the DMAI asserts BGACKN, the DMAI will negate its bus request and will not assume ownership of the bus.

Data Transfers

The actual transfer of data between the memory and the device occurs during the data transfer phase. All transfers occur during a single cycle except in the case of long word operands, in which case two cycles are used to transfer the operand as two 16-bit words. The transfers take place using a 'single address' protocol; the DMAI addresses the memory via the bus address lines, while the device is implicitly addressed via the acknowledge output.

When a request is generated using the request method programmed in the control register, the DMAI obtains the bus and asserts acknowledge to notify the device that a transfer is to take place. The DMAI asserts all S68000 bus control signals needed for the transfer and holds them until the device responds with ready. The bus cycle then terminates normally. Ready may be tied low (asserted) if the device is fast enough.

When the transfer is from memory to the device, data is valid when DTACKN is asserted by the memory and remains valid until the data strobe(s) are negated. The assertion of DTCN from the DMAI can be used to latch the data, as the data strobes are not removed until one-half clock after the assertion of DTCN.

When the transfer is from device to memory, the data must be valid on the bus before the DMAI asserts the data strobe(s). The device indicates valid data by asserting ready. The DMAI then asserts the strobes and holds them asserted until the memory accepts the data, indicated by the assertion of DTACKN. The DMAI then asserts DTCN and negates the data strobes.

Flow charts for these operations are shown in figures 1 and 2. Refer to the timing section for the equivalent timing diagrams.

Operation Termination

Termination of the block transfer occurs under the conditions detailed below.

Terminal Count

As part of each transfer of an operand, the DMAI decrements the memory transfer counter. If this counter is decremented to zero, the operand is the last operand of the block. The DMAI operation is complete and it notifies the device of completion by asserting the DONEN output during the last operand transfer cycle. When the transfer has been completed, the channel active bit in the CSR is cleared and the COC bit is set, unless an error occurs.

Device Termination

The DMAI monitors the state of the DONEN line while acknowledging a device transfer request. If the device asserts DONEN, the DMAI will terminate operation after the transfer of the current operand. When the transfer has been completed, the DMAI clears the channel active bit and sets the COC and normal device termination bits in the CSR. If both the DMAI and the device assert DONEN, the device termination is not recognized, but the operation does terminate.

Software Abort

The software abort bit (CCR[4]) allows the MPU to abort the current operation of the DMAI. The COC and error bits in the CSR are set, the channel active bit in the CSR is cleared, and an abort error condition is signaled in the CER.

Rerun Error

The DMAI provides a rerun input (RERUNN) to indicate a bus exception condition. RERUNN must arrive prior to or in coincidence with DTACKN in order to be recognized, and the DMAI verifies that the line has been stable for two clock cycles before acting on it. The occurrence of a rerun during a DMAI bus cycle forces it to terminate the bus cycle in an orderly manner.

When the assertion of rerun is verified, the DMAI stops operation and three-states the data, address, and control lines, except BGACKN, so that it retains ownership of the bus. It remains halted until rerun becomes inactive, and then re-tries the last bus cycle. If rerun is asserted again, the DMAI stops DMA operation, releases the bus, sets the error and COC bits in the CSR, clears the active bit in the CSR, and sets the error code in the CER to indicate a bus error.

While stopped due to assertion of rerun, the DMAI does not generate any bus cycles and will not honor any requests until it is removed. However, the DMAI still recognizes requests.

Error Recovery Procedure

If an error occurs during a DMA transfer such that the DMAI stops the DMA operation, information is available to the operating system for an error recovery routine.

The information available to the operating system consists of the memory address counter, the memory transfer counter, and the control, status, and error registers. The DMAI decrements the memory transfer counter before attempting a DMA operation, so the register will contain the count minus one of the attempted transfer. The memory address counter will contain the address at which the DMA operation was attempted.

Reset

The reset input (RESETN) provides a means of resetting and initializing the DMAI from an external source. If the DMAI is a bus master when reset is received, the DMAI relinquishes the bus. Reset clears the control and error registers, sets all bits of the status register except CSR[8] to zero, and initializes the interrupt vector register to H'0'F.

Interrupts

The interrupt enable bit (CCR[3]) determines whether the DMAI generates interrupt requests. When the bit is set, an interrupt request is generated if the channel operation complete bit in the CSR is set. When the IACKN input is asserted, and the DMAI has an interrupt request pending, the DMAI returns an interrupt vector on the data bus.

The interrupt vector issued is the contents of the IVR. Only the seven most significant bits of the programmed value are used by the DMAI. The vector from the DMAI contains a zero in the LSB position if a normal termination occurred (error bit not set) and contains a one in the LSB position if termination was due to an error (error bit set).

The contents of this register are initialized to H'0F' by a reset. The value returned will be H'0F', regardless of the error state, until the register is programmed by the MPU.

To provide compatibility with the other 68K family DMA controllers, the IVR has two addresses (see table 1). If program compatibility is required, the value written at the

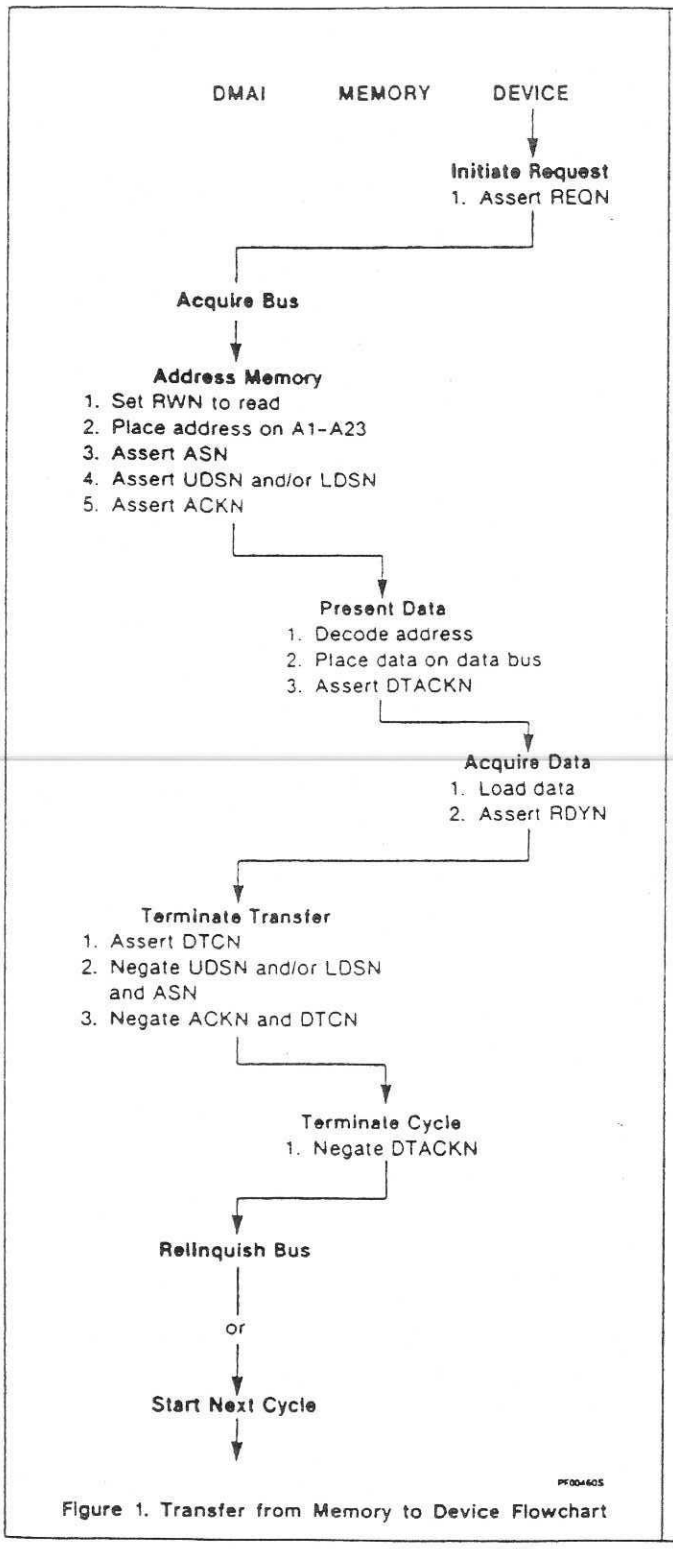


Figure 1. Transfer from Memory to Device Flowchart

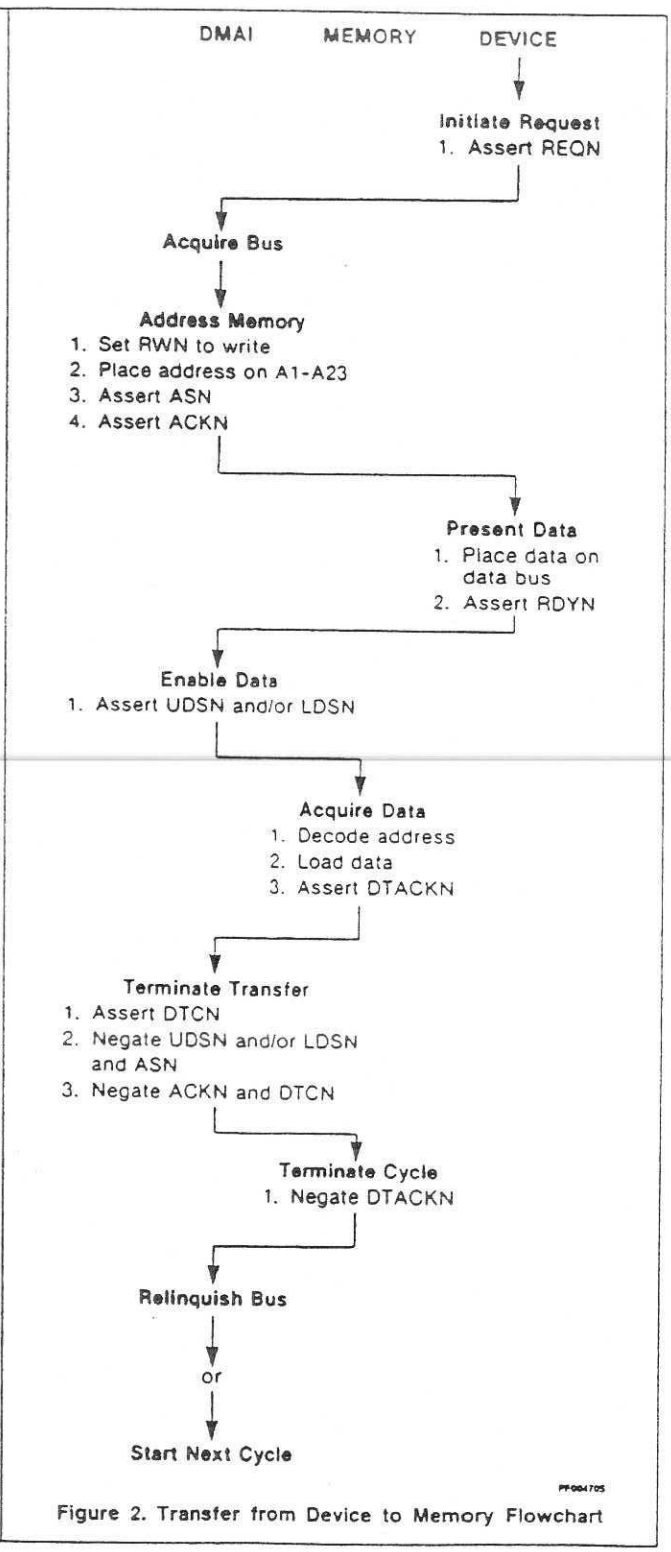


Figure 2. Transfer from Device to Memory Flowchart

normal IVR address should have a zero as its LSB, and the value written at the error IVR address should be the same but with the LSB equal to one.

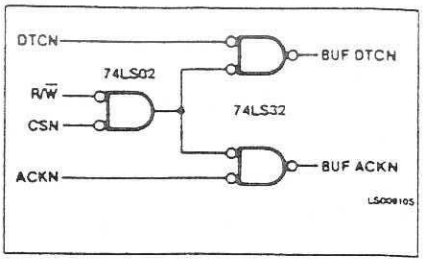
APPLICATIONS

Figure 3 illustrates a typical interconnection of the DMAI in a 68000 based system.

DESIGN NOTE

When clearing the error bit in CSR (bit 12) after a DMAI abort due to a double RERUNN, ACKN and DTCN will both go low concurrent with CSN and DTACKN for one CLK cycle.

To prevent the possibility that the device may misinterpret these signals, it is suggested that these signals be ANDed with CSN (see figure below).



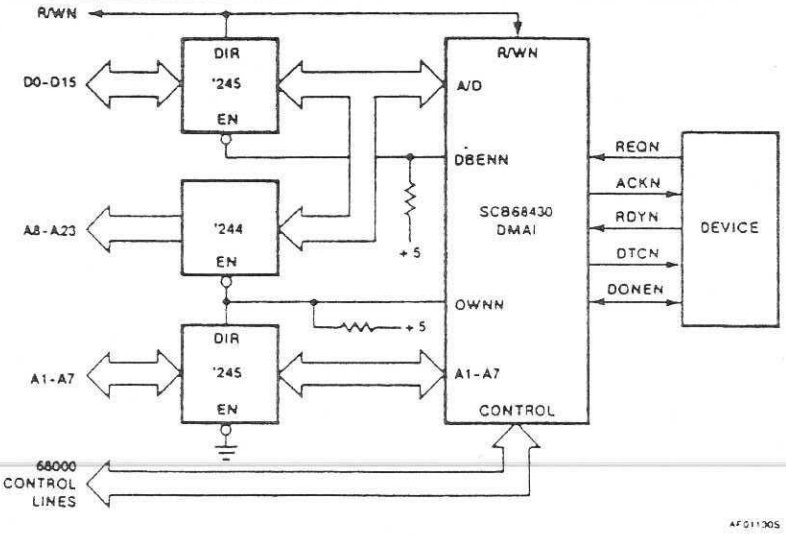


Figure 3. DMAI Application

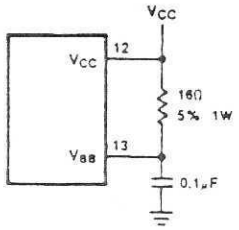


Figure 4. Recommended V_{BB} Test Circuit

ABSOLUTE MAXIMUM RATINGS[1]

PARAMETER	RATING	UNIT
Supply voltages V_{CC} and V_{BB}	-0.5 to +7.0	V
Input voltage	-0.5 to +5.5	V
Operating temperature range[2]	0 to +70	°C
Storage temperature	-65 to +150	°C

DC ELECTRICAL CHARACTERISTICS V_{CC} = 5.0V ±5%, V_{BB} = Figure 4, T_A = 0°C to +70°C[3,4,7]

PARAMETER		TEST CONDITIONS	LIMITS		UNIT
			Min	Max	
V_{IL}	Input low voltage			0.8	V
V_{IH}	Input high voltage		2.0		V
V_{OL}	Output low voltage	I_{OUT} = 5.3mA		0.5	V
V_{OH}	Output high voltage, all outputs except open collector outputs[5]	I_{OUT} = -400µA	2.5		V
I_{IL}	Input low current	V_{IN} = 0.4V		-400	µA
I_{IH}	Input high current	V_{IN} = 2.7V		20	µA
I_{OC}	Open collector off state current[5]	V_{OUT} = 2.4V		20	µA
I_{SC}	Output short circuit current[6]	V_{CC} = max	-40	-100	mA
I_{CC}	V_{CC} supply current	V_{CC} = max		130	mA
I_{BB}	V_{BB} supply current			275	mA

NOTES:

1. Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is stress rating only and functional operation of the device at these or at any other conditions other than those indicated in the Electrical Characteristics section of this data sheet is not implied.
2. For operating at elevated temperatures, the device must be derated based on +150°C maximum junction temperature.
3. Parameters are valid over specified temperature range.
4. All voltage measurements are referenced to ground (V_{SS}). For testing, all signals swing between 0.4V and 2.4V with a transition time of 20ns maximum. All time measurements are referenced at input voltages of 0.8V and 2.0V as appropriate.
5. IRQN, BRN, DONEN, and OWNN are open collector outputs.
6. No more than one output should be connected to ground at one time.
7. Capacitive test load is 100pF for all pins except DTCN which has a 35pF capacitive test load.

AC ELECTRICAL CHARACTERISTICS V_{CC} = 5.0V ±5%, V_{BB} = Figure 4, T_A = 0°C to +70°C[3,4,7]

NO.	FIGURE	CHARACTERISTICS	TENTATIVE LIMITS				UNIT
			10MHz		12.5MHz		
			Min	Max	Min	Max	
1	5	A1 - A7, ASN, RWN, set-up to UDSN, LDSN low	0		0		ns
2	5	D0 - D15 3-state to invalid data from ASN, CSN, and UDSN or LDSN low	10		10		ns
3	5	DTACKN 3-state to high from ASN, CSN, and UDSN or LDSN low	10		10		ns
4	5	CSN low after UDSN or LDSN low		25		25	ns
5	5, 6	DBENN low after ASN and CSN low		60		60	ns
6	5	D0 - D15 valid data from ASN, CSN, and UDSN or LDSN low		100		100	ns
7	5	DTACKN low after D0 - D15 valid data	-15	30	-15	30	ns
8	5	A1 - A7, ASN, RWN or CSN hold after UDSN and LDSN high	0		0		ns
9	5, 6	DBENN high from either ASN or CSN high		45		45	ns
10	5	D0 - D15 to 3-state from UDSN and LDSN high		80		80	ns
11	5	D0 - D15 to invalid data from UDSN and LDSN high	10		10		ns
12	5, 6	DTACKN high from UDSN and LDSN high		55		55	ns
13	5, 6	DTACK 3-state from either CSN or ASN high		85		85	ns
14	6	A1 - A7, ASN, RWN set-up to UDSN, LDSN low	50		50		ns/s
15	6	CSN set-up before UDSN or LDSN low	20		20		ns
16	6	DTACKN 3-state to high after CSN and ASN low	10		10		ns
17	6	D0 - D15 valid after UDSN or LDSN low		0		0	ns
18	6	DTACKN low from UDSN or LDSN low		100		100	ns
19	6	UDSN and LDSN low time	115		100		ns
20	6	A1 - A7 hold after UDSN and LDSN high	0		0		ns
21	6	ASN, RWN and CSN hold after UDSN and LDSN high	0		0		ns

AC ELECTRICAL CHARACTERISTICS (Continued)

NO.	FIGURE	CHARACTERISTICS	TENTATIVE LIMITS				UNIT
			10MHz		12.5MHz		
			Min	Max	Min	Max	
22	6	D0 – D15 hold after UDSN and LDSN high	0		0		ns
23	7	DBENN low from last low of ASN, IACKN, LDSN		65		65	ns
24	7	D0 – D7 valid after last low of ASN, IACKN, LDSN		105		105	ns
25	7	DTACKN 3-state to high after last low of ASN, IACKN, LDSN		100		100	ns
26	7	DTACKN low after last low of ASN, IACKN, LDSN		110		110	ns
27	7	DBENN high after first high of ASN, IACKN, LDSN		55		55	ns
28	7	D0 – D7 hold after first high of ASN, IACKN, LDSN		60		60	ns
29	7	D0 – D7 3-state after first high of ASN, IACKN, LDSN		80		80	ns
30	7	DTACKN high after first high of ASN, IACKN, LDSN		60		60	ns
31	7	DTACKN 3-state after first high of ASN, IACKN, LDSN		95		95	ns
32	8	BRN high from CLK high		65		65	ns
33	8, 11, 12	BGACKN low from CLK low		75		75	ns
34	8, 11, 12	OWNN low from CLK high		75		75	ns
35	8	BGACKN high from CLK low		75		75	ns
36	8,11,12	OWNN high from CLK high (load dependent)		50		50	ns
37	10	REQN set-up before CLK low	30		30		ns
38	10	REQN hold after CLK high	20		20		ns
39	10	BRN low from CLK high		80		80	ns
41	11, 12	ASN, UDSN, LDSN, RWN 3-state to high from CLK low		75		75	ns
43	11, 12	A1 – A23 to valid ASN	0		0		ns
44	11, 12	ASN low from CLK high		65		65	ns
45	11, 12	LDSN, UDSN low from CLK high		90		90	ns
46	11, 12	ACKN low from CLK high		65		65	ns
47	11, 12	DTACKN set-up to CLK high	30		30		ns
48	11, 12	RDYN set-up to CLK low	30		30		ns
49	11, 12	DTCN low from CLK high		70		70	ns
50	11, 12	ASN high from CLK high		75		75	ns
51	11, 12	LDSN, UDSN, high from CLK high		90		90	ns
52	11, 12	DTACKN, RDYN hold after CLK high	0		0		ns
–	11, 12	ASN, LDSN, UDSN, high from DTCN low	–20		–20		ns
53	11, 12	ACKN high from CLK high		50		50	ns
54	11, 12	DTCN high from CLK high		50		50	ns
55	11, 12	Address valid after CLK low	10		10		ns
–	11, 12	Address valid after ASN high	0		10		ns
56	11, 12	DONEN (output) low from CLK low		120		120	ns
57	11, 12	DONEN (output) high from CLK high		50		50	ns
58	11, 12	DONEN (input) set-up low before CLK low	30		30		ns
59	11, 12	DONEN (input) hold low after CLK high	0		0		ns
60	11, 12	BGACKN, ASN, UDSN, LDSN, RWN to 3-state from CLK low		75		75	ns
62	11, 12	A1 – A23 valid to 3-state from CLK high		100		100	ns
63	12	R/WN low from CLK high		65		65	ns
64	12	R/WN high from CLK high		75		75	ns
65	13	RERUNN set-up low before CLK high	30		30		ns
66	13	RERUNN hold low from CLK high	20		20		ns
67	13	A1 – A23 to idle state from CLK low		100		100	ns
68	13	A1 – A23 to valid after CLK low		85		85	ns

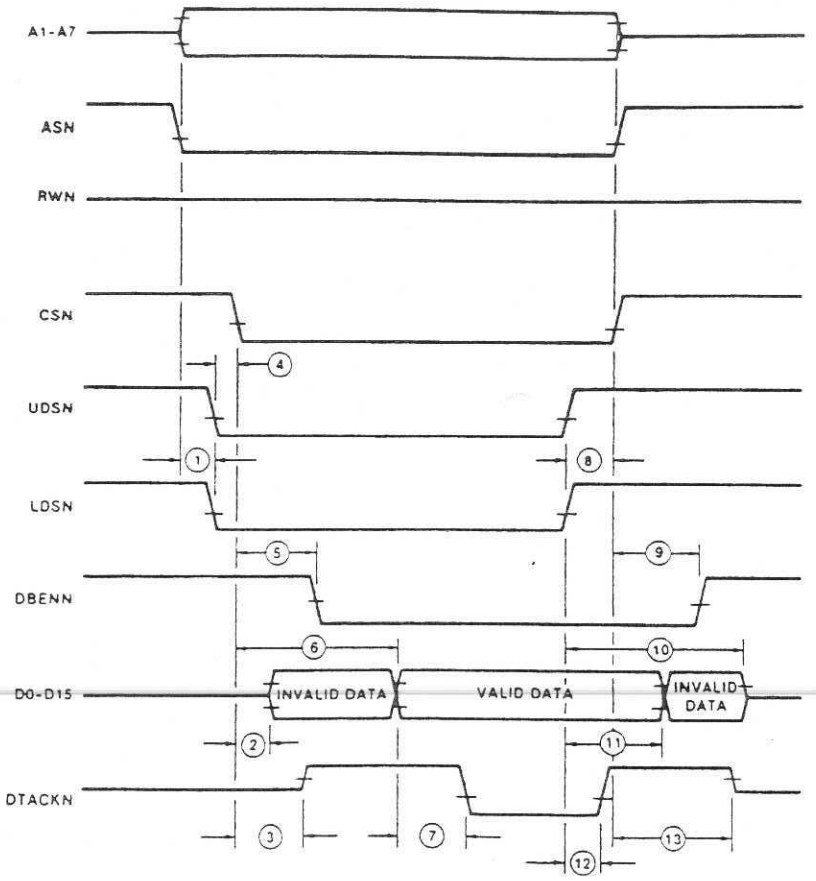


Figure 5. DMAI Read Timing

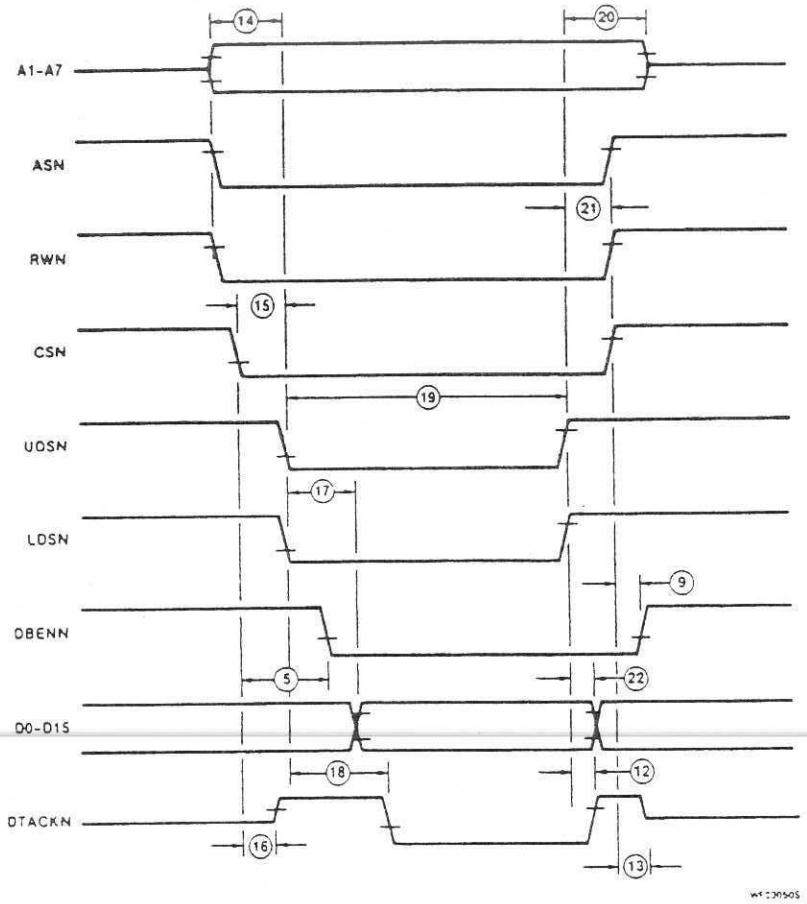


Figure 6. DMAI Write Timing

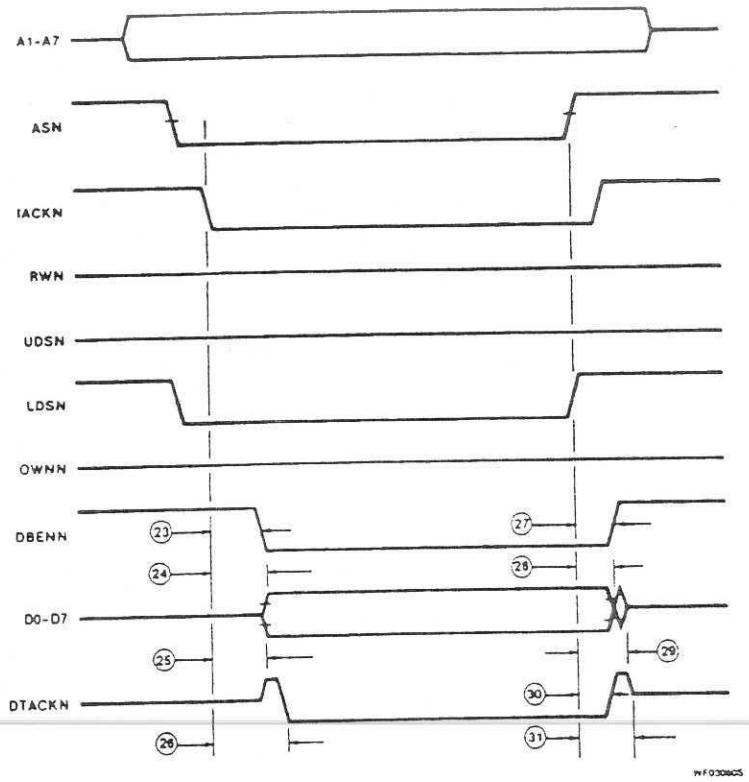
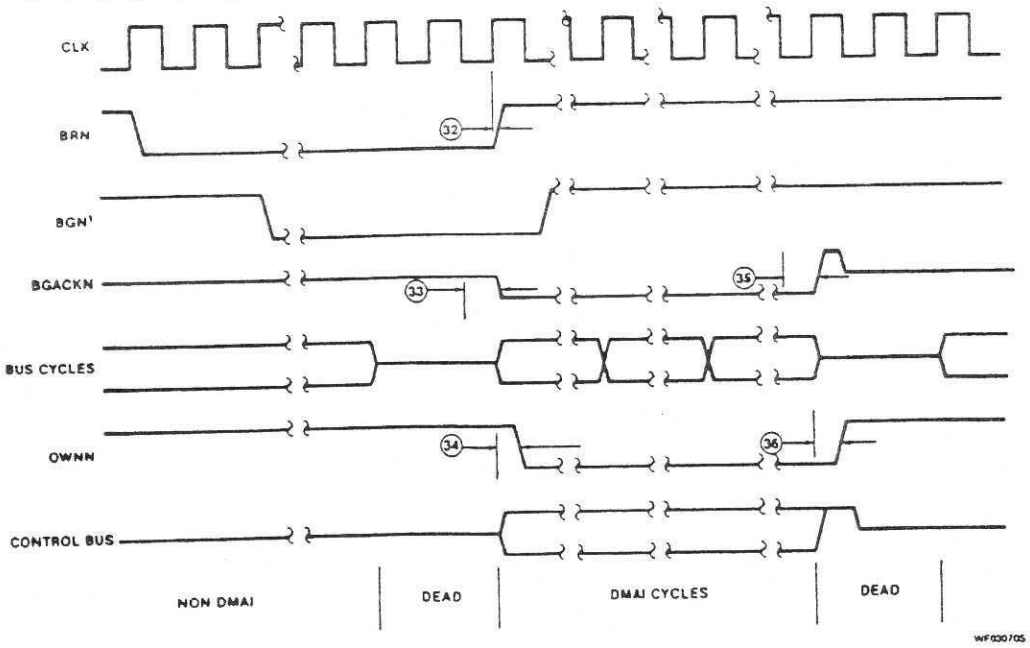
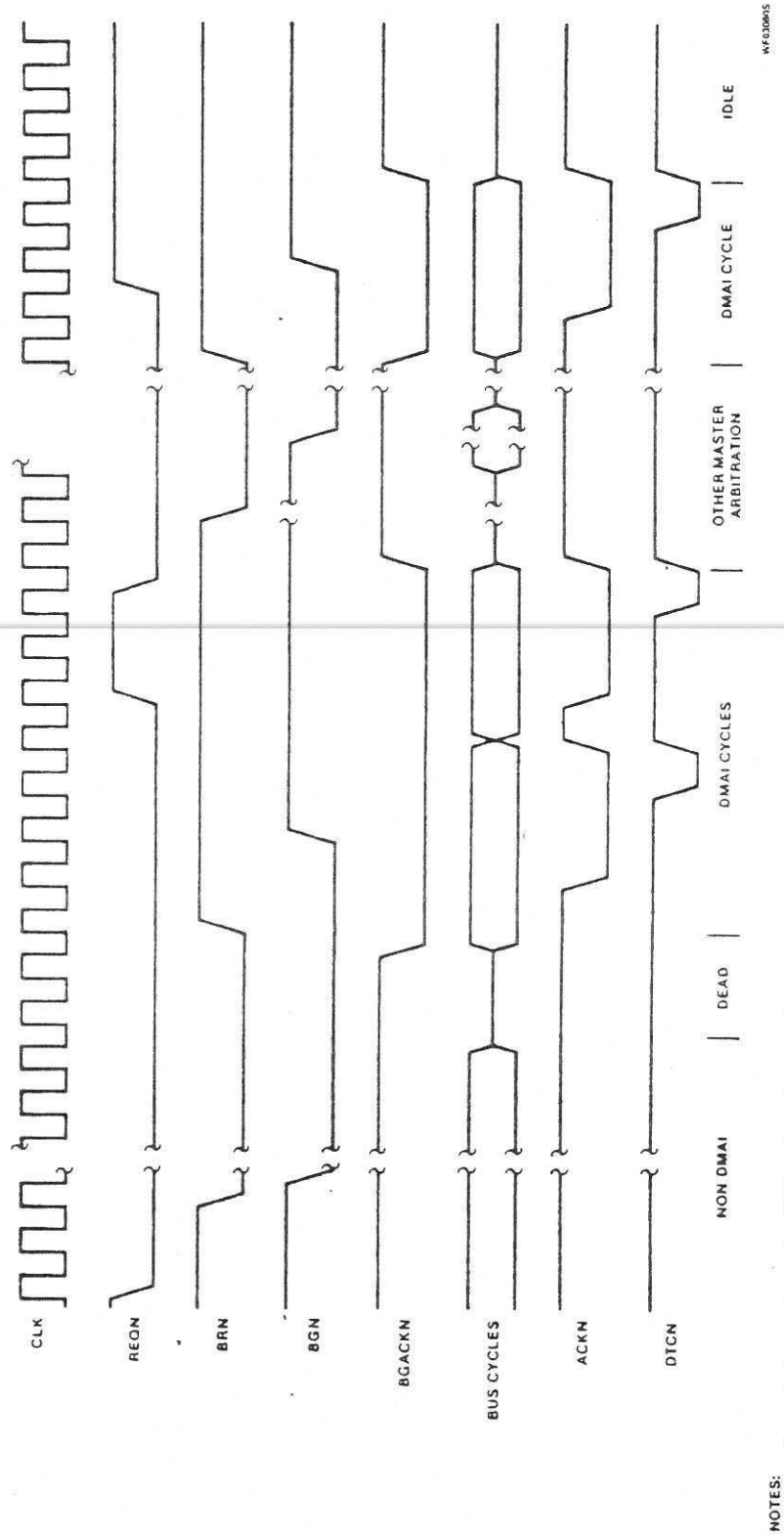


Figure 7. CPU IACK Cycle to DMAI



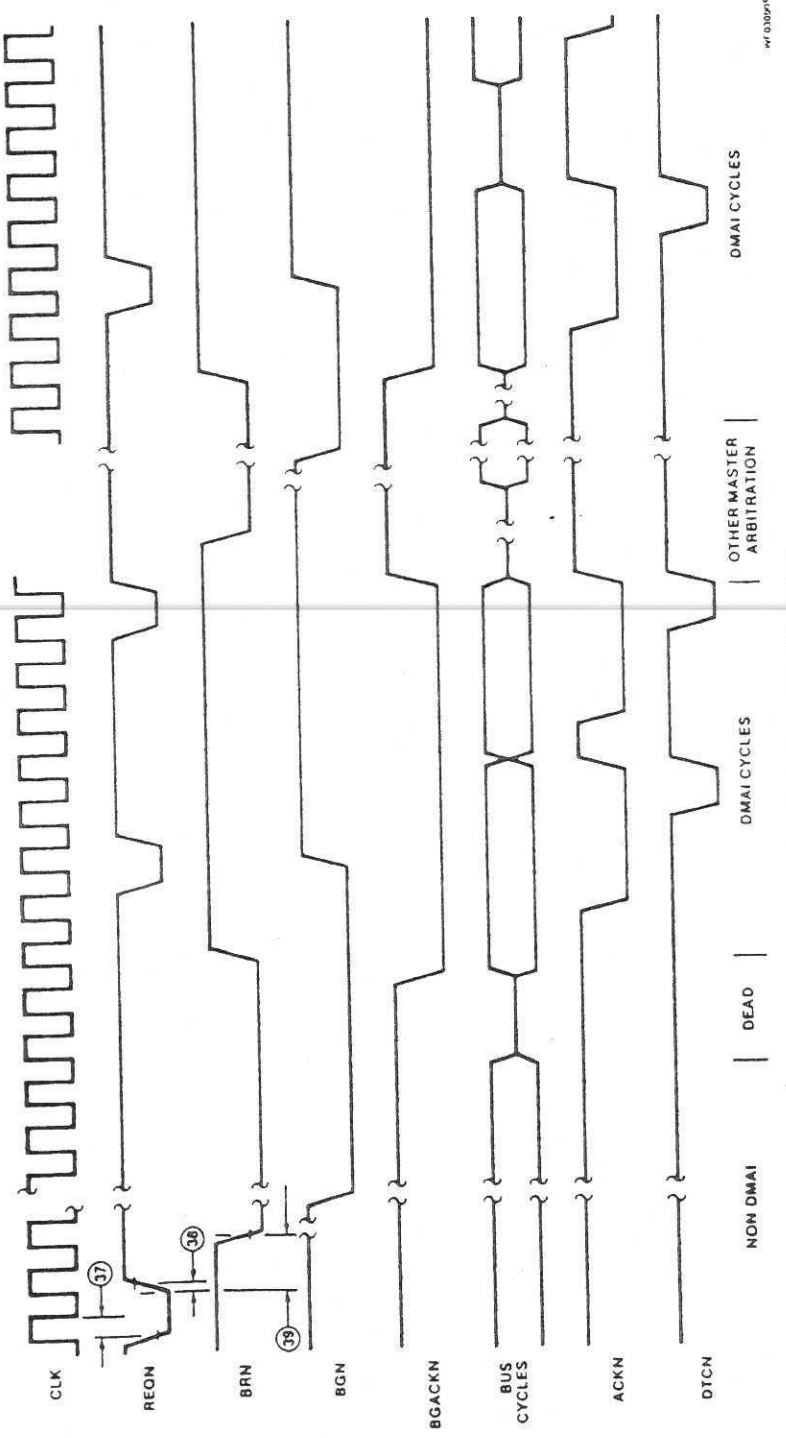
NOTES:
1. Device will become master if BGN is asserted concurrent with or later than REQN (same clock edge or later).
2. ASN, DTACKN and BGACKN must be negated in order for DMAI to become master. Timing assumes all these happen concurrent with BGN — if not, it is from the latest signal which is negated.

Figure 8. DMAI Bus Arbitration Timing



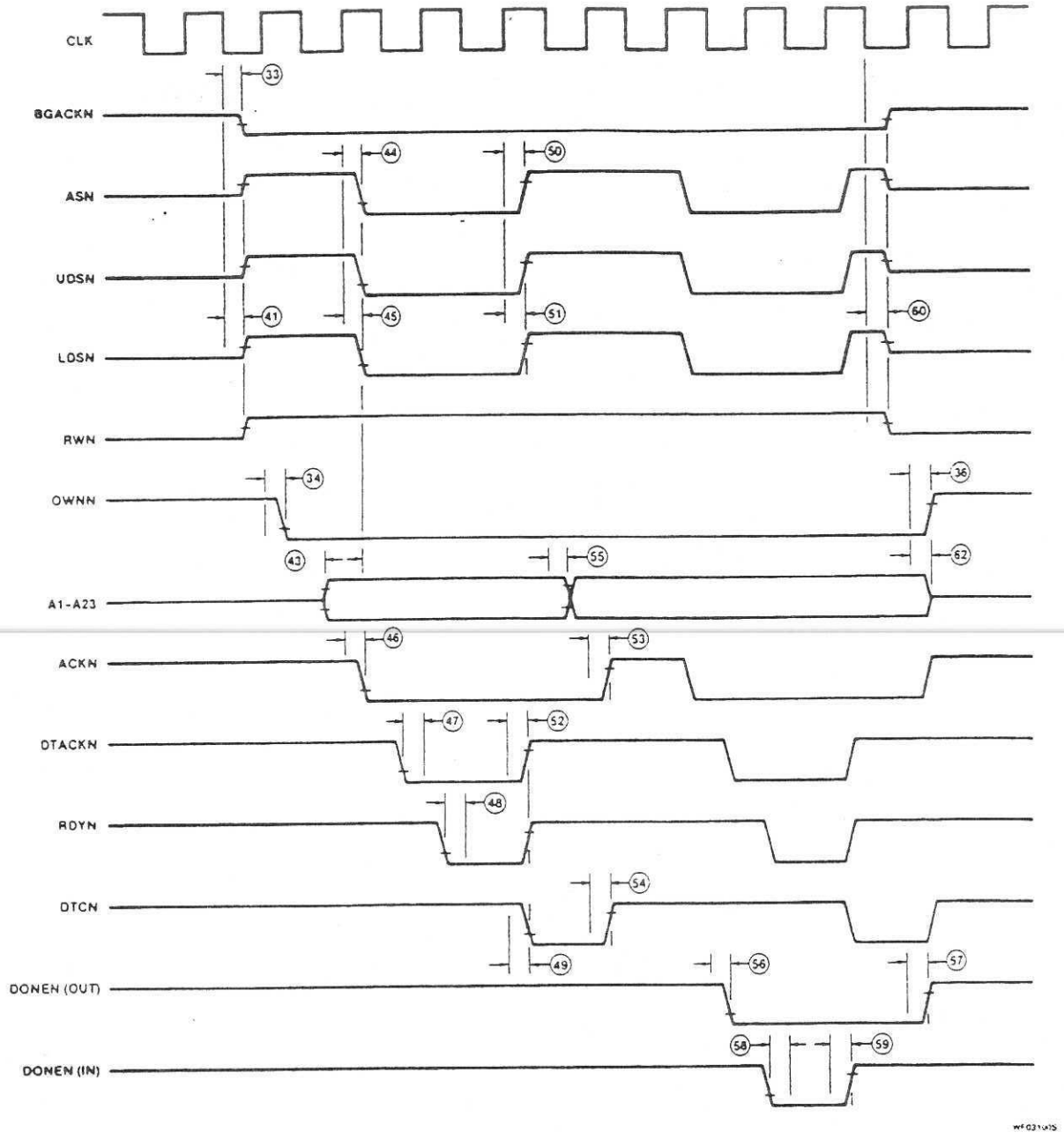
NOTES:
1. To maintain mastership next bus cycle, REQN must remain asserted at least until 1/2 clock cycle after DTCN is asserted.
2. To guarantee that mastership is relinquished next cycle, REQN must be negated no later than 1/2 clock cycle prior to DTCN.

Figure 9. DMAI Burst Mode Request Timing



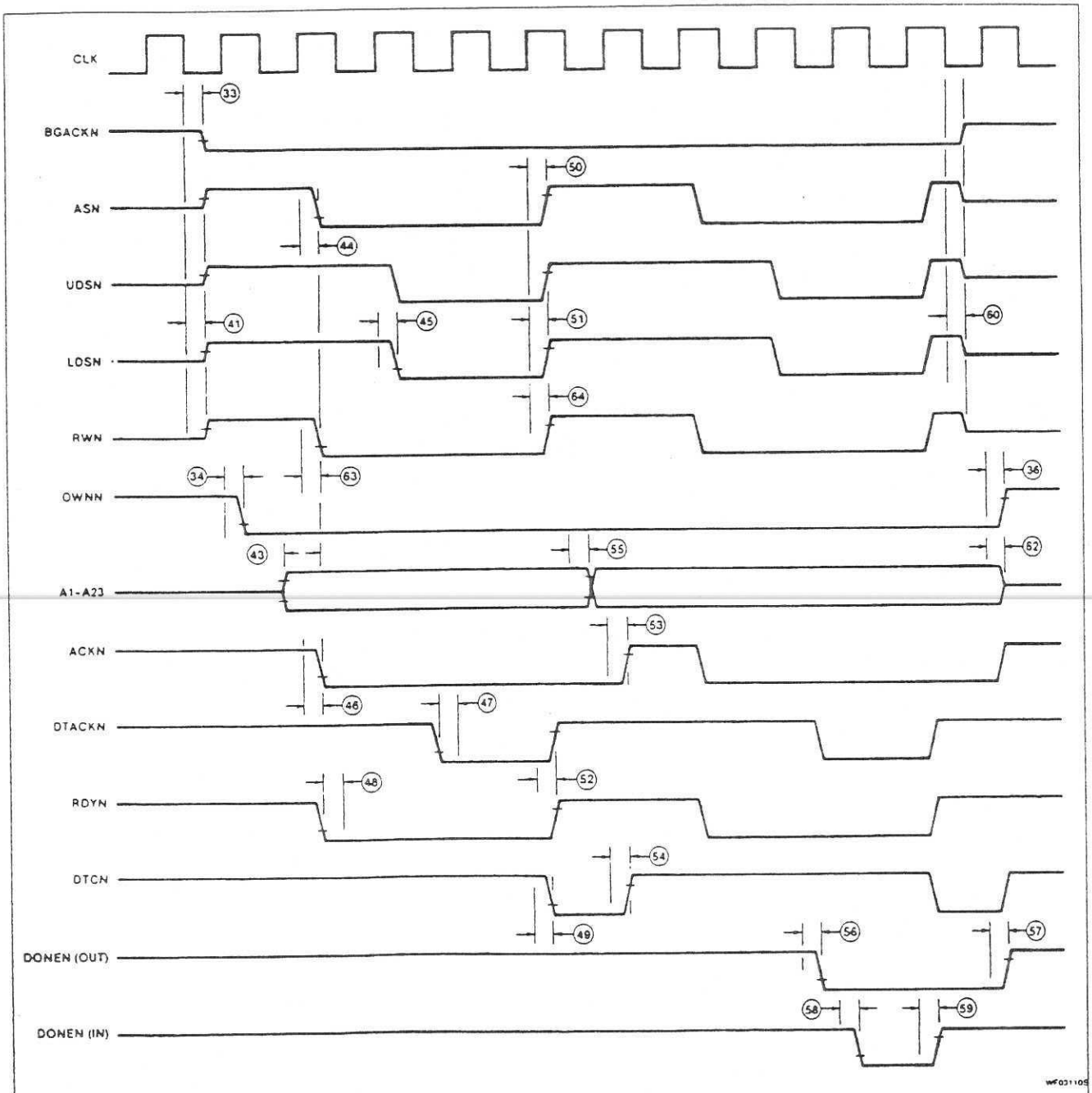
1. In order to keep the bus, REON must come no later than the 1/2 clock minus the set-up time (37) prior to assertion edge of DTCN.

Figure 10. DMAI Cycle Steal Mode Request Timing



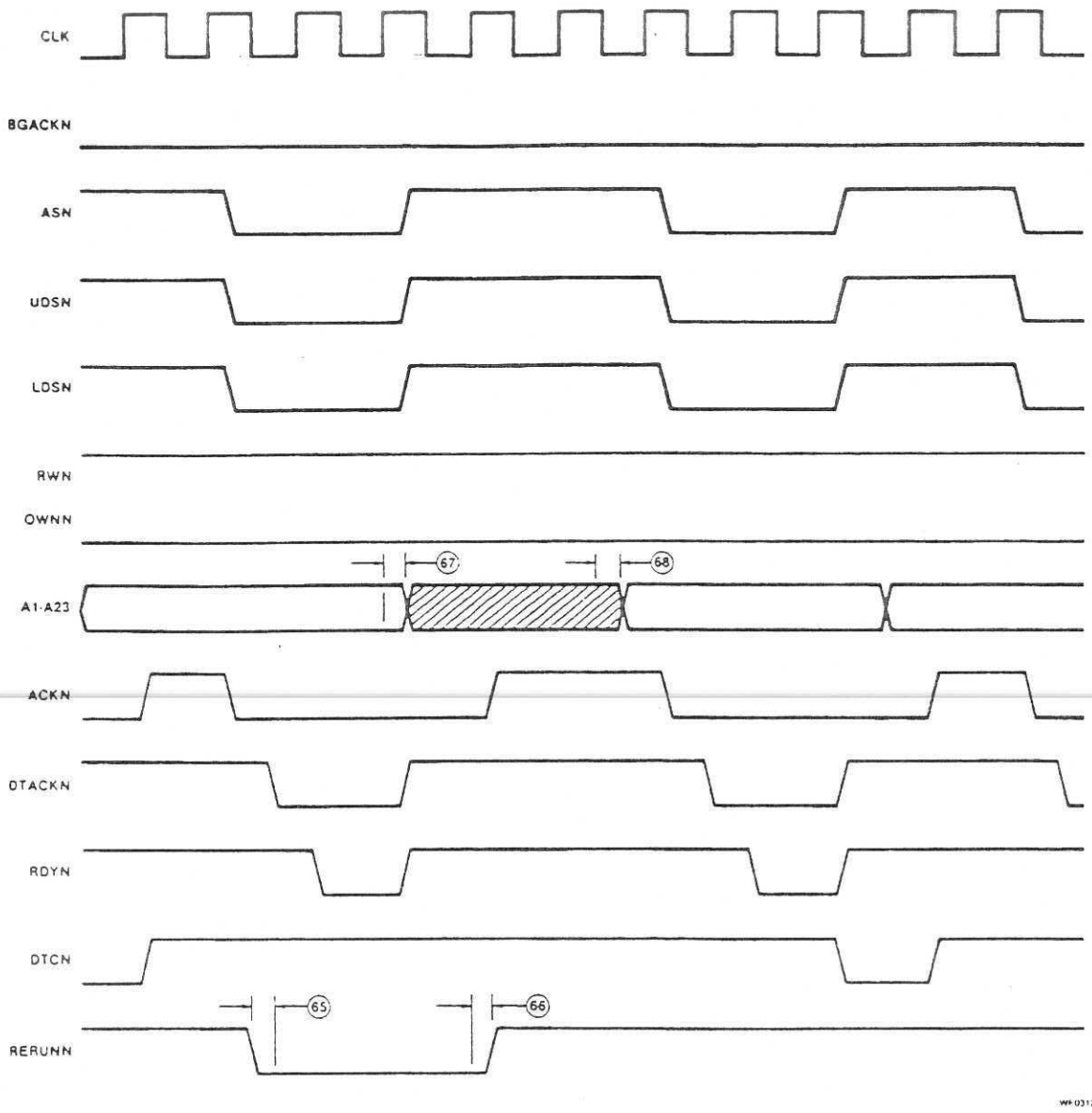
NOTE:
1. 16-bit transfer illustrated. For 8-bit transfer either LDSN or UDSN, but not both, will be asserted each cycle, depending on byte address

Figure 11. Read from Memory, Write to Device



NOTE:
1. 16-bit transfer illustrated. For 8-bit transfer either LDSN or UDSN, but not both, will be asserted each cycle, depending on byte address.

Figure 12. Write to Memory, Read from Device



NOTES:
1. 16-bit transfer illustrated. For 8-bit transfer either LDSN or UDSN, but not both, will be asserted each cycle, depending on byte address.
2. DMAI will release the bus after a RERUNN if there is no valid request. The next request will then retry the cycle which was terminated by the RERUNN signal.
3. RERUNN must be asserted no later than DTACKN and RDYN.
4. If a cycle is terminated by RERUNN, the transfer count will be one less than the actual data transferred correctly. The double RERUNN signal on the same cycle will terminate the DMAI operation with a status bit set and an interrupt generated (if enabled).

Figure 13. Rerun Asserted During Read from Memory, Write to Device

APPENDIX C

VMIVME-DR11W TEST SOFTWARE LISTING

src=dr11w4.sa

```
*************************************************************************
*                                                                       *
*       DR11W4:   DR11W  DATA TRANSFER TESTS                            *
*       ==========================================================      *
*       MODE_M:   Start test processing sequence as a master.           *
*       MODE_S:   Start test processing sequence as a slave.            *
*       ----------------------------------------------------------      *
*                                                                       *
*************************************************************************
*   (C)COPYRIGHT 1990   VME MICROSYSTEMS INTERNATIONAL CORPORATION      *
*   Version: 2.00.043090:CCG   --+--   CPU: Force cpu with Vmeprom       *
*************************************************************************

*       This is the main entry point to transfer tests

        XDEF    DR11W4              dr11w Dma (+) transfer tests

*       These entry points are provided for setup modification

        XDEF    SETMTC4             set the transfer counter
        XDEF    SETMAC4             set the address counter
        XDEF    SETAMR4             set the address modifier
        XDEF    SETDEF4             set MTC,MAC,AMR defaults
        XDEF    ACS                 set attn control+level

*       Reference external subroutines and variables

        XREF    DSPBYTE             display byte subr
        XREF    DSPWORD             display word subr
        XREF    DSPLONG             display long subr
        XREF    FLAGS               program (test) flags
        XREF    PCOUNT              loop on test pass count
        XREF    FCOUNT              loop on test fail count

        INCLUDE   VMEPROM.SA
*************************************************************************
*                                                                       *
*               VMEPROM INTERFACE SUPPORT CODE                          *
*       ==========================================================      *
*       Copyright 1989 Vme Microsystems International Corp.             *
*       ==========================================================      *
*       VMEPROM.SA  contains  the  assembler  options,                  *
*       constants  and macros  for  assembling  Development             *
*       Test  code that is to be ran on a Force  Cpu  board             *
*       under  Vmeprom/Pdos. This code is  presently  being             *
*       assembled on the motorola Vme/10 development system             *
*       or  compatable  cross-assembler  on  the  IBM  PC               *
*       (msdos).  This code is included in the test  source             *
*       at assembly time in each module as it is needed. It             *
*       is usually listed only once at the beginning in the             *
*       primary or 'main' module of the listing print file.             *
*       Recently included  are parameters for four commonly             *
*       used terminals and the macro to install them in the             *
*       TCB of the current task.  (this must be done before             *
```

```
                        *       changing register A6 in the test code software).          *
                        *       ------------------------------------------------         *
                        *       Version 2.00.02.14.90    by   Clark Coe Gwathney Jr.     *
                        *                                                                 *
                        *******************************************************************
                        *
                        *******[ TYPICAL ASSEMBLER OPTIONS ]*********************************
                        *
                                OPT     P=68020             use 68020 instruction coding
                                OPT     NOCEX               do not print dc.x definitions
                                OPT     NOMEX               do not print macro expansions
                        *
                        *******[ ASCII CONTROL EQUATES ]*************************************
                        *
<00000000>      NUL     EQU     00                  null
<00000001>      SOH     EQU     01                  start of header
<00000002>      STX     EQU     02                  start of text
<00000003>      ETX     EQU     03                  end of text
<00000004>      EOT     EQU     04                  end of transmission
<00000005>      ENQ     EQU     05                  inquiry
<00000006>      ACK     EQU     06                  acknowledge
<00000007>      BEL     EQU     07                  bell (beep)
<00000008>      BS      EQU     08                  backspace
<00000009>      HT      EQU     09                  horizontal tab
<0000000A>      LF      EQU     10                  line feed
<0000000B>      VT      EQU     11                  vertical tab
<0000000C>      FF      EQU     12                  form feed
<0000000D>      CR      EQU     13                  carraige return
<0000000E>      SO      EQU     14
<0000000F>      SI      EQU     15
<00000010>      DLE     EQU     16
<00000011>      DC1     EQU     17                  device control 1 (xon)
<00000012>      DC2     EQU     18                  device control 2
<00000013>      DC3     EQU     19                  device control 3 (xoff)
<00000014>      DC4     EQU     20                  device control 4
<00000015>      NAK     EQU     21                  not acknowledge
<00000016>      SYN     EQU     22
<00000017>      ETB     EQU     23                  end of transmission block
<00000018>      CAN     EQU     24                  cancel
<00000019>      EM      EQU     25
<0000001A>      SUB     EQU     26
<0000001B>      ESC     EQU     27                  escape
<0000001C>      FS      EQU     28
<0000001D>      GS      EQU     29
<0000001E>      RS      EQU     30
<0000001F>      US      EQU     31
<0000007F>      DEL     EQU     127                 delete
                        *
                        *******[ VMEPROM / PDOS SYSTEM CALL EQUATES ]************************
                        *
<0000A006>      X881    EQU     $A006               save 68881 flags on switch
<0000A0AA>      XAPF    EQU     $A0AA               append file to file
<0000A070>      XBCP    EQU     $A070               set baud on console port
<0000A072>      XCBC    EQU     $A072               check for break character
<0000A050>      XCBD    EQU     $A050               convert binary to decimal
```

```
<0000A052>      XCBH    EQU     $A052           convert binary to hex
<0000A054>      XCBM    EQU     $A054           convert to decimal w/message
<0000A074>      XCBP    EQU     $A074           check for break or pause
<0000A06A>      XCBX    EQU     $A06A           convert to decimal in buffer
<0000A056>      XCDB    EQU     $A056           convert ascii to binary
<0000A0D0>      XCFA    EQU     $A0D0           close file with attribute
<0000A068>      XCHX    EQU     $A068           convert binary to hex in buffer
<0000A0D2>      XCLF    EQU     $A0D2           close opened file
<0000A076>      XCLS    EQU     $A076           clear screen (TCB parms)
<0000A0AE>      XCPY    EQU     $A0AE           copy file
<0000A026>      XCTB    EQU     $A026           create task block
<0000A032>      XDEV    EQU     $A032           delay set/reset event
<0000A0D4>      XDFL    EQU     $A0D4           define file
<0000A0D6>      XDLF    EQU     $A0D6           delete file
<0000A04A>      XDMP    EQU     $A04A           dump memory from stack
<0000A114>      XDPE    EQU     $A114           delay physical event
<0000A024>      XDTV    EQU     $A024           define trap vectors
<0000A00C>      XERR    EQU     $A00C           return error to vmeprom
<0000A030>      XEXC    EQU     $A030           execute pdos call in d7.w
<0000A00E>      XEXT    EQU     $A00E           exit to vmeprom
<0000A0CE>      XFAC    EQU     $A0CE           file altered check
<0000A0F8>      XFBF    EQU     $A0F8           flush buffers
<0000A0A0>      XFFN    EQU     $A0A0           fix file name
<0000A058>      XFTD    EQU     $A058           fix time & date
<0000A040>      XFUM    EQU     $A040           free user memory
<0000A048>      XGCB    EQU     $A048           conditional get char
<0000A078>      XGCC    EQU     $A078           get char conditional
<0000A09E>      XGCP    EQU     $A09E           get char from port
<0000A07A>      XGCR    EQU     $A07A           get character raw
<0000A07C>      XGLB    EQU     $A07C           get line in buffer
<0000A07E>      XGLM    EQU     $A07E           get line in mon. buff
<0000A080>      XGLU    EQU     $A080           get line in user buff
<0000A010>      XGML    EQU     $A010           get memory limits
<0000A004>      XGMP    EQU     $A004           get message pointer
<0000A05A>      XGNP    EQU     $A05A           get next parameter
<0000A01E>      XGTM    EQU     $A01E           get task message
<0000A03E>      XGUM    EQU     $A03E           get user memory
<0000A0C0>      XISE    EQU     $A0C0           initialize sector
<0000A0FA>      XKTB    EQU     $A0FA           kill task
<0000A028>      XKTM    EQU     $A028           kill task message
<0000A0B0>      XLDF    EQU     $A0B0           load file
<0000A03A>      XLER    EQU     $A03A           load error register
<0000A0A2>      XLFN    EQU     $A0A2           look for file name
<0000A0D8>      XLKF    EQU     $A0D8           lock file
<0000A014>      XLKT    EQU     $A014           lock task
<0000A02E>      XLSR    EQU     $A02E           load status register
<0000A0DA>      XNOP    EQU     $A0DA           open random shared file
<0000A00A>      XPAD    EQU     $A00A           pack ascii date
<0000A084>      XPBC    EQU     $A084           put buffer to console
<0000A086>      XPCC    EQU     $A086           put char's to console
<0000A088>      XPCL    EQU     $A088           put cr,lf to console
<0000A0BC>      XPCP    EQU     $A0BC           place char in port buffer
<0000A0BA>      XPCR    EQU     $A0BA           put character raw
<0000A096>      XPDC    EQU     $A096           put data to console
<0000A09C>      XPEM    EQU     $A09C           put encoded msg to console
```

```
<0000A08A>      XPLC    EQU     $A08A           put line to console
<0000A08C>      XPMC    EQU     $A08C           put message to console
<0000A08E>      XPSC    EQU     $A08E           put cursor @ row,col
<0000A0DC>      XPSF    EQU     $A0DC           position file
<0000A098>      XPSP    EQU     $A098           put space to console
<0000A0DE>      XRBF    EQU     $A0DE           read bytes from file
<0000A0B2>      XRCN    EQU     $A0B2           reset console inputs
<0000A092>      XRCP    EQU     $A092           read port cursor position
<0000A0A6>      XRDE    EQU     $A0A6           read next directory entry
<0000A02A>      XRDM    EQU     $A02A           dump processor registers
<0000A0A8>      XRDN    EQU     $A0A8           read dir entry by name
<0000A05C>      XRDT    EQU     $A05C           read date
<0000A0E0>      XRFA    EQU     $A0E0           read file attributes
<0000A0FE>      XRFP    EQU     $A0FE           read file position
<0000A0E2>      XRLF    EQU     $A0E2           read line from file
<0000A0E4>      XRNF    EQU     $A0E4           rename file
<0000A0E6>      XROO    EQU     $A0E6           open random read only file
<0000A0E8>      XROP    EQU     $A0E8           open random file
<0000A094>      XRPS    EQU     $A094           read port status
<0000A0C2>      XRSE    EQU     $A0C2           read sector
<0000A042>      XRSR    EQU     $A042           read status register
<0000A0B4>      XRST    EQU     $A0B4           reset disk
<0000A0C4>      XRSZ    EQU     $A0C4           read sector zero
<0000A044>      XRTE    EQU     $A044           return from interrupt
<0000A05E>      XRTM    EQU     $A05E           read time
<0000A034>      XRTP    EQU     $A034           read time parameters
<0000A012>      XRTS    EQU     $A012           read task status
<0000A0EA>      XRWF    EQU     $A0EA           rewind file
<0000A018>      XSEF    EQU     $A018           set event flag w/swap
<0000A046>      XSEV    EQU     $A046           set event flag
<0000A002>      XSMP    EQU     $A002           send message pointer
<0000A112>      XSOE    EQU     $A112           suspend on physical event
<0000A0EC>      XSOP    EQU     $A0EC           open sequential file
<0000A09A>      XSPF    EQU     $A09A           set port flag
<0000A020>      XSTM    EQU     $A020           send task message
<0000A03C>      XSTP    EQU     $A03C           set/read task priority
<0000A01C>      XSUI    EQU     $A01C           suspend until event intr
<0000A02C>      XSUP    EQU     $A02C           enter supervisor mode
<0000A000>      XSWP    EQU     $A000           swap to next task
<0000A0B6>      XSZF    EQU     $A0B6           get disk size
<0000A090>      XTAB    EQU     $A090           tab to column
<0000A01A>      XTEF    EQU     $A01A           test event flag
<0000A110>      XTLP    EQU     $A110           xlate logical to phys event
<0000A036>      XUAD    EQU     $A036           unpack ascii date
<0000A060>      XUDT    EQU     $A060           unpack date
<0000A0EE>      XULF    EQU     $A0EE           unlock file
<0000A016>      XULT    EQU     $A016           unlock task
<0000A008>      XUSP    EQU     $A008           return to user mode
<0000A062>      XUTM    EQU     $A062           unpack time
<0000A116>      XVEC    EQU     $A116           set/read exception vector
<0000A0F0>      XWBF    EQU     $A0F0           write bytes to file
<0000A064>      XWDT    EQU     $A064           write date
<0000A0F2>      XWFA    EQU     $A0F2           write file attributes
<0000A0FC>      XWFP    EQU     $A0FC           write file parameters
<0000A0F4>      XWLF    EQU     $A0F4           write line to file
```

```
<0000A0C6>      XWSE    EQU     $A0C6           write sector
<0000A066>      XWTM    EQU     $A066           write time
<0000A0F6>      XZFL    EQU     $A0F6           zero file (wipe)
                *
                *******[ TERMINAL SETUP PARAMETERS ]*****************************
                *
                *   Vmeprom TCB terminal setups (see ST cmd in vmeprom)
                *
<C8CA9B59>      V52     EQU     $C8CA9B59       DEC V52
<1A009B3D>      ADM3A   EQU     $1A009B3D       LEER SEIGLER ADM3A
<00000000>      VT100   EQU     $00000000       DEC VT100
<1E1A9B3D>      TVI950  EQU     $1E1A9B3D       TELE VIDIO TVI 950
                *
                *******[ FORCE CPU DEVICE ADDRESS EQUATES ]**********************
                *
                *   PI/T 2 Timer address equates
                *
<FF800E00>      PIT2    EQU  $FF800E00      Force cpu pit2 base address
                *
<FF800E10>      TCR     EQU  PIT2+$10       timer control register
<FF800E1A>      TSR     EQU  PIT2+$1A       timer status register
<FF800E12>      CPR     EQU  PIT2+$12       counter preload register
<FF800E16>      CTR     EQU  PIT2+$16       counter register
                *
                *******[ TEST MACRO DEFINITIONS ]********************************
                *
                PUSH    MACRO
                        MOVEM.L   \1,-(A7)          Push register(s) on stack
                        ENDM
                POP     MACRO
                        MOVEM.L   (A7)+,\1          pop register(s) off stack
                        ENDM
                CLC     MACRO
                        ANDI.B    #$FE,CCR          clear carry flag in CCR
                        ENDM
                STC     MACRO
                        ORI.B     #$1,CCR           set carry flag in CCR
                        ENDM
                SYSCALL MACRO
                        DC.W      \1                Vmeprom line a opcode emulation
                        ENDM
                DISPLAY MACRO
                        LEA       \1,A1             point to start of message
                        SYSCALL   XPLC              put line to console
                        ENDM
                PROMPT  MACRO
                        LEA       \1,A1             point to prompt msg
                        SYSCALL   XPLC              put line to console
                        SYSCALL   XGCR              get char from console
                        SYSCALL   XPCR              put char to console
                        ENDM
                TERM    MACRO
                        MOVE.L    \1,($43A,A6)      set terminal parms in TCB
                        ENDM
                WAIT    MACRO
```

```
                        MOVE.L   \1,D0              load do with delay value
                        BSR      DELAY              and call pit2 delay subr
                        ENDM
                        INCLUDE  DR11W.EQ

                *       Base pointer for VME short I/O

<FBFF0000>      VMEIO   EQU      $FBFF0000          Short I/O on Force Cpu 29

                *       Base Address for Vnet board registers

<FBFFE000>      DR11    EQU      VMEIO+$E000        Base address of DR11W

                *       Offset Address Equates for the Dmai module

<FBFFE000>      CHSR    EQU      DR11+$00           Channel Status Register
<FBFFE001>      CHER    EQU      DR11+$01           Channel Error Register
<FBFFE004>      DCR     EQU      DR11+$04           Device Control Register
<FBFFE005>      OCR     EQU      DR11+$05           Operation Ctrl Register
<FBFFE007>      CHCR    EQU      DR11+$07           Channel Ctrl Register
<FBFFE00A>      MTC     EQU      DR11+$0A           Mem. Transfer Count
<FBFFE00C>      MACH    EQU      DR11+$0C           Mem. Address Count High
<FBFFE00E>      MACL    EQU      DR11+$0E           Mem. Address Count Low
<FBFFE025>      DIVR    EQU      DR11+$25           Done Intr Vector Register

                *       Offset Address Equates for the Bim module

<FBFFE043>      BICR    EQU      DR11+$43           Berr Intr Ctrl Register
<FBFFE045>      AICR    EQU      DR11+$45           Attention Intr Ctrl Reg
<FBFFE047>      DICR    EQU      DR11+$47           Dma Done Intr Ctrl Reg
<FBFFE04B>      BIVR    EQU      DR11+$4B           Berr Intr Vect Register
<FBFFE04D>      AIVR    EQU      DR11+$4D           Attention Intr Vect Reg

                *       Offset Address Equates for Vnet board registers

<FBFFE060>      BSR     EQU      DR11+$60           Board Status Register
<FBFFE061>      BCR     EQU      DR11+$61           Board Control Register
<FBFFE062>      IODR    EQU      DR11+$62           I/O Data Register
<FBFFE065>      AMR     EQU      DR11+$65           Address Modifier Reg



                *       DR11W  command equates

<00000000>      NULL    EQU      $0000              null (sync) cmd
<00000001>      ACKN    EQU      $0001              acknowledge cmd
<00000002>      LREQ    EQU      $0002              link request cmd
<00000003>      BYTE    EQU      $0003              set byte dma mode
<00000004>      WORD    EQU      $0004              set word dma mode
<00000005>      SEND    EQU      $0005              init dma Tx
<00000006>      RCVE    EQU      $0006              init dma rx
<00000007>      BRST    EQU      $0007              set burst dma
<00000008>      CYCL    EQU      $0008              set cycle dma
<00000009>      XFER    EQU      $0009              start transfers
<0000000A>      EXIT    EQU      $000A              set exit loop flag
```

```
      <0000000B>      ABRT    EQU     $000B           set abort mode flag

                      ********************************************************************
                      *
                      *       Swap to supervisor to get VBR then back ...

/00
000000                DR11W4  SYSCALL XSUP            Supervisor mode on 29
000002 4E7A8801               MOVEC   VBR,A0          get Vector Base pointer
000006                        SYSCALL XUSP            back to User mode
000008 23C800000000           MOVE.L  A0,SVBR         save vector base register

                      *       Init Interrupt Vector Table   (@ VBR)

00000E 217C00000000+  INITS   MOVE.L  #DONE,($150,A0) Init Done Vector Addr ($54)
000016 217C00000000+          MOVE.L  #DERR,($154,A0) Init Derr Vector Addr ($55)
00001E 217C00000000+          MOVE.L  #ATTN,($158,A0) Init Attn Vector Addr ($56)
000026 217C00000000+          MOVE.L  #BERR,($15C,A0) Init Berr Vector Addr ($57)

                      *       Init Interrupt vector registers

00002E 13FC0054+              MOVE.B  #$54,DIVR       Init Done Intr vector reg
000036 13FC0056+              MOVE.B  #$56,AIVR       Init Attn Intr vector reg
00003E 13FC0057+              MOVE.B  #$57,BIVR       Init Berr Intr vector reg

                      *       Init program variables

000046 23FC00000000+          MOVE.L  #DTBL,DPTR      init test data pointer

                      ********************************************************************
                      *                                                                  *
                      *       SHOW WE'RE HERE AND PROMPT FOR START MODE                  *
                      *                                                                  *
                      ********************************************************************

000050                DR11W   DISPLAY DR11_M          show dr11 dma testing
000058                D_MODE  PROMPT  SLCT_M          display select mode msg
000064 0C000058               CMP.B   #'X',D0         select 'X' for exit this ..
000068 6718                   BEQ.S   DR11W_X         then exit to main menu
00006A 0C00004D               CMP.B   #'M',D0         start in Master mode?
00006E 67000056               BEQ     M_MODE          then start as master
000072 0C000053               CMP.B   #'S',D0         start in Slave  mode?
000076 670C                   BEQ.S   S_MODE          then start as slave
000078                        DISPLAY ARGH_M          for an invalid selection
000080 60D6                   BRA.S   D_MODE          and repeat selection
000082 4E75           DR11W_X RTS                     return to main menu

                      ********************************************************************
                      *                                                                  *
                      *       S: START THIS NODE IN SLAVE MODE TRANSFER TESTS             *
                      *                                                                  *
                      ********************************************************************

000084 13FC002C+      S_MODE  MOVE.B  #$2C,BCR        Fled_off,dir_in,en_attn
00008C 13FC0000+              MOVE.B  #$00,BICR       disable Berr interrupt
```

```
000094 13F900000000+            MOVE.B    ACS,AICR            enable  Attn interrupt

                       *        Enter Slave mode until exit cmd from Master

00009E 61000068        S_LOOP   BSR       SLAVE               slave/responder mode
0000A2 08B90003+                BCLR      #3,FLAGS            is the abort bit set
0000AA 6618                     BNE.S     S_EXIT              then exit from S_MODE

                       *        Enter Master mode and sequence tests to Slave

0000AC 610000A2                 BSR       MASTER              master/initiator mode

                       *        Then exit via console input or repeat test

0000B0 08390001+                BTST      #1,FLAGS            check loop on test flag
0000B8 670A                     BEQ.S     S_EXIT              if not set then exit else
0000BA 08B90003+                BCLR      #3,FLAGS            check for test loop abort
0000C2 67DA                     BEQ.S     S_LOOP              if not set then continue
0000C4 4E75            S_EXIT   RTS                           return from module

                       ********************************************************************
                       *                                                                  *
                       *        M: START THIS NODE IN MASTER MODE TRANSFER TESTS           *
                       *                                                                  *
                       ********************************************************************

0000C6 13FC002C+       M_MODE   MOVE.B    #$2C,BCR            Fled_off,dir_in,en_attn
0000CE 13FC0000+                MOVE.B    #$00,BICR           disable BERR interrupt
0000D6 13F900000000+            MOVE.B    ACS,AICR             enable ATTN interrupt

                       *        Master initiating commands to the Slave

0000E0 6100006E        M_LOOP   BSR       MASTER              master/initiator mode
0000E4 08B90003+                BCLR      #3,FLAGS            is the abort bit set
0000EC 6618                     BNE.S     M_EXIT              then exit from M_MODE

                       *        Enter Slave mode until exit cmd from Master

0000EE 61000018                 BSR       SLAVE               slave/responder mode

                       *        Then exit via console input or repeat test

0000F2 08390001+                BTST      #1,FLAGS            check loop on test flag
0000FA 670A                     BEQ.S     M_EXIT              if not set then exit else
0000FC 08B90003+                BCLR      #3,FLAGS            check for test loop abort
000104 67DA                     BEQ.S     M_LOOP              if not set then continue
000106 4E75            M_EXIT   RTS                           return from module

                       ********************************************************************
                       *                                                                  *
                       *        SLAVE or RESPONDER Sequence                               *
                       *                                                                  *
                       ********************************************************************

000108                 SLAVE    DISPLAY   SLAV_M              show we are in slave mode
```

```
000110 61000280          S0      BSR       GET_C              waiting for command input
000114 02F90000+                 CMP2.W    VALID,D0           for a valid command word
00011C 651C                      BCS.S     S1                 show received invalid cmd
00011E 41F900000000              LEA       SCMDS,A0           point to slave cmd table
000124 20700400                  MOVE.L    (A0,D0.W*4),A0     get cmd handler address
000128 4E90                      JSR       (A0)               jump subr to handler
00012A 08B90002+                 BCLR      #2,FLAGS           test skip acknowledge
000132 66DC                      BNE       S0                 if set skip sending ack
000134 61000278                  BSR       SLACK              send SLave ACKnowledge
000138 60D6                      BRA       S0                 continue slave mode
00013A                   S1      DISPLAY   CERR_M             show command error msg
000142 323900000000              MOVE.W    COMMAND,D1         get command word to dsp
000148 61000000                  BSR       DSPWORD            show bad command word
00014C                           SYSCALL   XPCL               next line on the crt
00014E 60C0                      BRA       S0

                         ********************************************************************
                         *                                                                  *
                         *        MASTER or INITIATOR Sequence                              *
                         *                                                                  *
                         ********************************************************************

000150                   MASTER  DISPLAY   MAST_M             show we are in master mode
000158                   M_CONT  DISPLAY   LREQ_M             show master making link req
000160 303C0002                  MOVE.W    #LREQ,D0           to other node for transfers
000164 61000222                  BSR       PUT_C              put link request command
000168                           DISPLAY   LACK_M             show link request ackn'd

                         *------------------------------------------------------------------*
                         *        Test Burst and Cycle in main test loop                    *
                         *------------------------------------------------------------------*

                         *   I.   Execute all dma tests in BURST mode

000170 6100034C          T_BURST BSR       SETBRST            set burst mode
000174 49F900000000              LEA       BRST_M,A4          burst mode msg
00017A 303C0007                  MOVE.W    #BRST,D0           burst command
00017E 61000208                  BSR       PUT_C              put command
000182 61000040                  BSR       T_MAIN             test loop

                         *   II.  Execute all dma tests in CYCLE mode

000186 61000358          T_CYCLE BSR       SETCYCL            set cycle mode
00018A 49F900000000              LEA       CYCL_M,A4          cycle mode msg
000190 303C0008                  MOVE.W    #CYCL,D0           cycle command
000194 610001F2                  BSR       PUT_C              put command
000198 6100002A                  BSR       T_MAIN             test loop

                         *   III. Send EXIT loop command to slave

00019C                   T_EXIT  DISPLAY   RELL_M             show releasing link
0001A4 08390001+                 BTST      #1,FLAGS           loop on test flag
0001AC 6704                      BEQ.S     T_LESS             then skip loop info
0001AE 6100040A                  BSR       L_INFO             else display loop info
0001B2 08B90002+         T_LESS  BCLR      #2,FLAGS           clr omit ackn flag for slave
```

```
0001BA 303C000A              MOVE.W   #EXIT,D0          exit command to slave
0001BE 610001C8              BSR      PUT_C             put command wait ackn
0001C2 4E75                  RTS                        return from master mode

                     *-------------------------------------------------------------*
                     *       Main Test loop (longs words and bytes)                *
                     *-------------------------------------------------------------*

      <0001C4/00>    T_MAIN   EQU      *                 main test loop entry point

                     *   A.   Test Word Dma Transfers    (16 bits)

0001C4 610003A0      T_WORD   BSR      SETWORD           set up for words
0001C8 4BF900000000          LEA      WORD_M,A5         word message
0001CE 303C0004              MOVE.W   #WORD,D0          word setup command
0001D2 610001B4              BSR      PUT_C             put command, get ackn
0001D6 6100001A              BSR      T_TEST            data test loop

                     *   B.   Test Byte Dma Transfers    (8 bits)

0001DA 610003B4      T_BYTE   BSR      SETBYTE           set up for bytes
0001DE 4BF900000000          LEA      BYTE_M,A5         byte message
0001E4 303C0003              MOVE.W   #BYTE,D0          byte setup command
0001E8 6100019E              BSR      PUT_C             put command, get ackn
0001EC 61000004              BSR      T_TEST            data test loop

                     *   C.   Then we return for cycle or exit testing

0001F0 4E75                  RTS                        return to caller

                     *-------------------------------------------------------------*
                     *       Main Dma I/O output,input compare test data loop      *
                     *-------------------------------------------------------------*

                     *   1.   Display Test mode (Burst/Cycle) and Size (B/W)

0001F2 224C          T_TEST   MOVE.L   A4,A1             get mode message
0001F4                        SYSCALL  XPLC              put line to console
0001F6 224D                   MOVE.L   A5,A1             get data size msg
0001F8                        SYSCALL  XPLC              put line to console

                     *   2.   Fill test buffer with TDATA (test data var)

0001FA 610003E4      T_FILL   BSR      FILLBUFF          with t_data

                     *   3.   Transmit the buffer to the Slave board

0001FE 303C0006      T_TX     MOVE.W   #RCVE,D0          Slave Rx cmd
000202 61000184              BSR      PUT_C             put command
000206 61000216              BSR      SET_TX            Transmit
00020A 303C0009              MOVE.W   #XFER,D0          Go (xfer) cmd
00020E 61000178              BSR      PUT_C             put command
000212 6100024A              BSR      SET_GO            start dma
000216 08B90005+     D_TX     BCLR     #5,FLAGS          test done in flags
00021E 67F6                   BEQ.S    D_TX              wait for done flag
```

```
000220 08B90007+                BCLR      #7,FLAGS            tx error
000228 66000066                 BNE       D_FAIL              then dma error

                        *  4.   Receive the buffer from the Slave board

00022C 303C0005         T_RX    MOVE.W    #SEND,D0            Slave Tx cmd
000230 61000156                 BSR       PUT_C               put command
000234 61000208                 BSR       SET_RX              set Receive
000238 61000224                 BSR       SET_GO              and set go
00023C 6100013E                 BSR       PUT_GO              & dont wait on ack
000240 08B90005+        D_RX    BCLR      #5,FLAGS            test done flag
000248 67F6                     BEQ.S     D_RX                wait for done
00024A 08B90007+                BCLR      #7,FLAGS            test error flag
000252 6600003C                 BNE       D_FAIL              then dma error

                        *  5.   Test the buffer for correct TDATA

000256 610003CA         T_BUFF  BSR       TESTBUFF            for t_data
00025A 08B90007+                BCLR      #7,FLAGS            buffer err?
000262 66000040                 BNE       B_FAIL              then fail

                        *  6.   Test for last data in test data table

000266 0CB900000000+    T_NEXT  CMPI.L    #0,TDATA            last data ?
000270 6688                     BNE       T_FILL              then refill & test

                        *  7.   Show passed and exit

000272                  T_PASS  DISPLAY   PASS_M              else show passed
00027A 23FC00000000+            MOVE.L    #DTBL,DPTR          reset data pointer
000284 06B900000001+            ADDI.L    #1,PCOUNT           increment pass count
00028E 4E75                     RTS

                        *  8.   Handle dmai error from slave board

000290                  D_FAIL  DISPLAY   BADD_M              bad dma from slave
000298 607E                     BRA.S     X_FAIL              common exit

                        *  9.   Handle acknowledge error from slave board

00029A                  A_FAIL  DISPLAY   BADA_M              bad ackn from slave
0002A2 6074                     BRA.S     X_FAIL              common exit

                        *  10.  Handle buffer data error from slave board

0002A4                  B_FAIL  DISPLAY   BADB_M              bad data from slave
0002AC                          DISPLAY   ADDR_M              show address msg
0002B4 2208                     MOVE.L    A0,D1               get fixed addr
0002B6 61000000                 BSR       DSPLONG             show it
0002BA                          DISPLAY   READ_M              show wrote data msg
0002C2 0C390001+                CMPI.B    #1,DSIZE            test data size
0002CA 6D1E                     BLT.S     B_LONG              < 1 is long
0002CC 6EFF00000034             BGT.L     B_BYTE              > 1 is byte
0002D2 3210             B_WORD  MOVE.W    (A0),D1             = 1 is word
0002D4 61000000                 BSR       DSPWORD             show word
```

```
0002D8                          DISPLAY   WROT_M            show read data msg
0002E0 3200                     MOVE.W    D0,D1
0002E2 61000000                 BSR       DSPWORD
0002E6                          SYSCALL   XPCL
0002E8 602E                     BRA.S     X_FAIL            exit
0002EA 2210             B_LONG  MOVE.L    (A0),D1           = 0 is long
0002EC 61000000                 BSR       DSPLONG           show long
0002F0                          DISPLAY   READ_M
0002F8 2200                     MOVE.L    D0,D1
0002FA 61000000                 BSR       DSPLONG
0002FE                          SYSCALL   XPCL
000300 6016                     BRA.S     X_FAIL            exit
000302 1210             B_BYTE  MOVE.B    (A0),D1           = 2 is byte
000304 61000000         B_DATA  BSR       DSPBYTE           show byte
000308                          DISPLAY   READ_M
000310 1200                     MOVE.B    D0,D1
000312 61000000                 BSR       DSPBYTE
000316                          SYSCALL   XPCL

                        *       Common exit for transfer tests

000318 06B900000001+    X_FAIL  ADDI.L    #1,FCOUNT         increment fail count
000322                          DISPLAY   CHSR_M            Common exit
00032A 1239FBFFE000             MOVE.B    CHSR,D1
000330 61000000                 BSR       DSPBYTE
000334                          DISPLAY   STAT_M
00033C 123900000000             MOVE.B    STATS,D1
000342 61000000                 BSR       DSPBYTE
000346                          DISPLAY   FLAG_M
00034E 123900000000             MOVE.B    FLAGS,D1
000354 61000000                 BSR       DSPBYTE
000358 23FC00000000+            MOVE.L    #DTBL,DPTR
000362                          SYSCALL   XPCL
000364 08390000+                BTST      #0,FLAGS          set this in MAIN
00036C 670C                     BEQ.S     NO_BR             break flag not set
00036E                  DO_BR   PROMPT    ANYK_M            break flag set
00037A 4E75             NO_BR   RTS                         return errors

                        ************************************************************************
                        ***                                                                  ***
                        **                    DR11W Board Subroutines                       **
                        ***                                                                  ***
                        ************************************************************************

                        *-----------------------------------------------------------------------*
                        *       put go command to slave without waiting for ackn                *
                        *-----------------------------------------------------------------------*
                        *       this is for the master mode when he is to Rx

00037C 33FC0009+         PUT_GO MOVE.W    #XFER,IODR        put go command to slave
000384 60000030                 BRA       ATTEN             & return via atten toggle

                        *-----------------------------------------------------------------------*
                        *       PUT/GET commands to/from other board (master & slave)           *
                        *-----------------------------------------------------------------------*
```

```
000388 33C0FBFFE062     PUT_C   MOVE.W  D0,IODR          write commmand via IODR
00038E 61000026                 BSR     ATTEN            set attn to other board
000392 08B90006+        GET_C   BCLR    #6,FLAGS         clear and test attn flag
00039A 67F6                     BEQ.S   GET_C            wait for attn from other brd
00039C 303900000000             MOVE.W  COMMAND,D0       and get command from buffer
0003A2 13F900000000+            MOVE.B  ACS,AICR          re-enable attn interrupt
0003AC 4E75                     RTS

                        *-------------------------------------------------------------*
                        *       SLave ACKn to other board with Atten (fall thru)      *
                        *-------------------------------------------------------------*

0003AE 33FC0001+        SLACK   MOVE.W  #ACKN,IODR       put acknowledge in IODR

                        *-------------------------------------------------------------*
                        *       Toggle attn input to other board (either mode)        *
                        *-------------------------------------------------------------*

0003B6 1039FBFFE061     ATTEN   MOVE.B  BCR,D0           get board control status
0003BC 024000FC                 AND.W   #$FC,D0          mask cycle and go bits
0003C0 08C00006                 BSET    #6,D0            set attention bit high
0003C4 13C0FBFFE061             MOVE.B  D0,BCR           write to other board
0003CA 08800006                 BCLR    #6,D0            set attention bit low
0003CE 13C0FBFFE061             MOVE.B  D0,BCR           write to other board
0003D4 4E75                     RTS                      return

                        *-------------------------------------------------------------*
                        *             NULL command handler (slave)                    *
                        *-------------------------------------------------------------*

0003D6                  NULLCMD DISPLAY NULL_M           show doing null command
0003DE 4E75                     RTS

                        *-------------------------------------------------------------*
                        *             LINK request command handler (slave)            *
                        *-------------------------------------------------------------*

0003E0                  LINKCMD DISPLAY LINK_M           show doing link request cmd
0003E8 4E75                     RTS

                        *-------------------------------------------------------------*
                        *             ACKN command handler (slave)                    *
                        *-------------------------------------------------------------*

0003EA                  ACKNCMD DISPLAY ACKN_M           show doing ackn command
0003F2 4E75                     RTS

                        *-------------------------------------------------------------*
                        *       Abort Mode handler (looping slave)                    *
                        *-------------------------------------------------------------*

0003F4 08F90003+        ABRTCMD BSET    #3,FLAGS         set test mode abort flag (loop)
                        *                                fall through slave exit cmd
                        *-------------------------------------------------------------*
```

```
                        *       Set Exit current loop flag (slave)                      *
                        *-------------------------------------------------------------*

0003FC                  EXITCMD DISPLAY EXIT_M          show exit message
000404 0239000B+                ANDI.B  #$0B,FLAGS      clr flags-(abort,loop,break)
00040C 33FC0001+                MOVE.W  #ACKN,IODR      put ackn in iodr
000414 61A0                     BSR     ATTEN           and acknowledge exit
000416 DFFC00000004             ADDA.L  #4,A7           to return to S_MODE handler
00041C 4E75                     RTS                     & return

                        *-------------------------------------------------------------*
                        *       Init Dmai and BCR to Transmit data (both)             *
                        *-------------------------------------------------------------*

00041E 1039FBFFE061     SET_TX  MOVE.B  BCR,D0          Get BCR status
000424 024000FC                 AND.W   #$FC,D0         mask cycle and go bits
000428 08800005                 BCLR    #5,D0           set direction to in
00042C 13C0FBFFE061             MOVE.B  D0,BCR          set direction in BCR
000432 207900000000             MOVE.L  TXS,A0          get pointer to setup
000438 6100004C                 BSR     SETDMA          do dmai module setup
00043C 4E75                     RTS                     return to caller

                        *-------------------------------------------------------------*
                        *       Init Dmai and BCR to Receive data (both)              *
                        *-------------------------------------------------------------*

00043E 1039FBFFE061     SET_RX  MOVE.B  BCR,D0          Get BCR status
000444 024000FC                 AND.W   #$FC,D0         mask cycle and go bits
000448 08C00005                 BSET    #5,D0           set direction to out
00044C 13C0FBFFE061             MOVE.B  D0,BCR          set direction in BCR
000452 207900000000             MOVE.L  RXS,A0          point to Rx setup table
000458 6100002C                 BSR     SETDMA          and go set the dmai
00045C 4E75                     RTS                     return to caller

                        *-------------------------------------------------------------*
                        *       Set the go bit in BCR to start dma transfers (both)   *
                        *-------------------------------------------------------------*

00045E 1039FBFFE061     SET_GO  MOVE.B  BCR,D0          get BCR status
000464 024000FC                 AND.W   #$FC,D0         mask cycle and go bits
000468 08000005                 BTST    #5,D0           test this direction bit
00046C 660C                     BNE.S   NOTX            skip cycle set for receive
00046E 08F90002+                BSET    #2,FLAGS        set skip acknowledge flag
000476 08C00001                 BSET    #1,D0           set cycle bit for transmit
00047A 08C00000         NOTX    BSET    #0,D0           set go bit to start dma
00047E 13C0FBFFE061             MOVE.B  D0,BCR          start transfers via BCR
000484 4E75                     RTS                     and return

                        *-------------------------------------------------------------*
                        *       Setup Dmai for transfers from table @ (A0) (both)     *
                        *-------------------------------------------------------------*
                        * registers:  csr|dcr|ocr|amr| mtc.w | mac.h | mac.l |ccr|dicr
                        *             0 | 1 | 2 | 3 | 4   5 | 6   7 | 8   9 | A | B

000486 13D8FBFFE000     SETDMA  MOVE.B  (A0)+,CHSR       Channel Status Register
```

```
00048C 13D8FBFFE004             MOVE.B    (A0)+,DCR          Device Control Registr
000492 13D8FBFFE005             MOVE.B    (A0)+,OCR          Operation Control Reg
000498 13D8FBFFE065             MOVE.B    (A0)+,AMR          Address Modifier reg
00049E 33D8FBFFE00A             MOVE.W    (A0)+,MTC          Mem Transfer Count
0004A4 33D8FBFFE00C             MOVE.W    (A0)+,MACH         Mem Address Count (hi)
0004AA 33D8FBFFE00E             MOVE.W    (A0)+,MACL         Mem Address Count (lo)
0004B0 13D8FBFFE007             MOVE.B    (A0)+,CHCR         enable Dmai Start & intrs
0004B6 13D8FBFFE047             MOVE.B    (A0)+,DICR         Done Intr Control Reg
0004BC 4E75                     RTS                          return to caller

                    *-------------------------------------------------------------------*
                    *       Set BURST mode in Transfers tables (both)                   *
                    *-------------------------------------------------------------------*

0004BE 08B90007+    SETBRST BCLR      #7,TX.W+1          for
0004C6 08B90007+            BCLR      #7,RX.W+1          all
0004CE 08B90007+            BCLR      #7,TX.B+1          xfer
0004D6 08B90007+            BCLR      #7,RX.B+1          modes
0004DE 4E75                 RTS                          return

                    *-------------------------------------------------------------------*
                    *       Set CYCLE mode in Transfer tables (both)                    *
                    *-------------------------------------------------------------------*

0004E0 08F90007+    SETCYCL BSET      #7,TX.W+1          for
0004E8 08F90007+            BSET      #7,RX.W+1          all
0004F0 08F90007+            BSET      #7,TX.B+1          xfer
0004F8 08F90007+            BSET      #7,RX.B+1          modes
000500 4E75                 RTS                          return

                    *-------------------------------------------------------------------*
                    *       Set Memory Transfer Count in tables (MTC.W) (both)          *
                    *-------------------------------------------------------------------*

000502 33C100000000 SETMTC4 MOVE.W    D1,TX.W+4          and set all
000508 33C100000000         MOVE.W    D1,RX.W+4          table MTC's
00050E 33C100000000         MOVE.W    D1,TX.B+4          the same
000514 33C100000000         MOVE.W    D1,RX.B+4          length
00051A 4E75                 RTS                          and return

                    *-------------------------------------------------------------------*
                    *       Set Address Modifier in Transfer tables (both)              *
                    *-------------------------------------------------------------------*

00051C 13C100000000 SETAMR4 MOVE.B    D1,TX.W+3          set AMR in tables for
000522 13C100000000         MOVE.B    D1,RX.W+3
000528 13C100000000         MOVE.B    D1,TX.B+3
00052E 13C100000000         MOVE.B    D1,RX.B+3
000534 4E75                 RTS

                    *-------------------------------------------------------------------*
                    *       Set Memory Address Counter in Transfer tables (both)        *
                    *-------------------------------------------------------------------*

000536 23C100000000 SETMAC4 MOVE.L    D1,TX.W+6          set Mac in tables
```

```
00053C 23C100000000             MOVE.L    D1,RX.W+6
000542 23C100000000             MOVE.L    D1,TX.B+6
000548 23C100000000             MOVE.L    D1,RX.B+6
00054E 4E75                     RTS

                        *-------------------------------------------------------------------*
                        *       Set Transfer defaults for MTC, MAC, AMR, EMA (menu)         *
                        *-------------------------------------------------------------------*

000550 223CFB000000     SETDEF4 MOVE.L    #$FB000000,D1       default transfer buffer
000556 61DE                     BSR       SETMAC4              set MAC
000558 323C1000                 MOVE.W    #$1000,D1           default transfer count
00055C 61A4                     BSR       SETMTC4              set MTC
00055E 123C0039                 MOVE.B    #$39,D1             default address modifier
000562 61B8                     BSR       SETAMR4              set AMR
000564 4E75                     RTS                           return

                        *-------------------------------------------------------------------*
                        *       Set Transfer mode pointers to WORD mode (both)              *
                        *-------------------------------------------------------------------*

000566 23FC00000000+    SETWORD MOVE.L    #FILL_W,FBS         setup fill buffer subr
000570 23FC00000000+            MOVE.L    #BUFF_W,TBS         setup test buffer subr
00057A 23FC00000000+            MOVE.L    #TX.W,TXS           setup word xfer table
000584 23FC00000000+            MOVE.L    #RX.W,RXS           setup word xfer table
00058E 4E75                     RTS                           and return

                        *-------------------------------------------------------------------*
                        *       Set Transfer mode pointers to BYTE mode (both)              *
                        *-------------------------------------------------------------------*

000590 23FC00000000+    SETBYTE MOVE.L    #FILL_B,FBS         setup fill buffer subr
00059A 23FC00000000+            MOVE.L    #BUFF_B,TBS         setup test buffer subr
0005A4 23FC00000000+            MOVE.L    #TX.B,TXS           setup byte xfer table
0005AE 23FC00000000+            MOVE.L    #RX.B,RXS           setup byte xfer table
0005B8 4E75                     RTS                           and return

                        *-------------------------------------------------------------------*
                        *       Display loop on test information                            *
                        *-------------------------------------------------------------------*

0005BA                  L_INFO  DISPLAY   PCNT_M              show pass message
0005C2 223900000000             MOVE.L    PCOUNT,D1           get passed count and
0005C8 61000000                 BSR       DSPLONG             show it as hex longword
0005CC                          DISPLAY   FCNT_M              show fail message
0005D4 223900000000             MOVE.L    FCOUNT,D1           get failed count and
0005DA 61000000                 BSR       DSPLONG             show it as hex longword
0005DE 4E75                     RTS                           and return

                        *-------------------------------------------------------------------*
                        *       Fill Buffer with Test DATA & incr pointer (master)          *
                        *-------------------------------------------------------------------*
                        *       Fill with TDATA update

0005E0 207900000000     FILLBUFF MOVE.L   DPTR,A0             get test data pointer
```

```
0005E6 23D800000000             MOVE.L    (A0)+,TDATA         get test data from table
0005EC 23C800000000             MOVE.L    A0,DPTR             set test data pointer

                        *       Fill only (no tdata update)

0005F2 207900000000     FILLMEM MOVE.L    TX.W+6,A0           point to transfer buffer
0005F8 323900000000             MOVE.W    TX.W+4,D1           Get buffer length in D1
0005FE 5341                     SUB.W     #1,D1               Less 1 for DBF instruction
000600 203900000000             MOVE.L    TDATA,D0            get current test data
000606 227900000000             MOVE.L    FBS,A1              pointer to fill block subr
00060C 4ED1                     JMP       (A1)                use B/W/L subr & return

                        *       Byte buffer fill subroutine

00060E 10C0             FILL_B  MOVE.B    D0,(A0)+            put data into buffer
000610 4600                     NOT.B     D0                  invert every other data
000612 51C9FFFA                 DBF       D1,FILL_B           do the whole buffer
000616 4E75                     RTS                           return to caller

                        *       Word Buffer fill subroutine

000618 30C0             FILL_W  MOVE.W    D0,(A0)+            put data into buffer
00061A 4640                     NOT.W     D0                  invert every other data
00061C 51C9FFFA                 DBF       D1,FILL_W           do the whole buffer
000620 4E75                     RTS                           return to caller

                        *-------------------------------------------------------------------*
                        *       Test buffer Data to current test data: TDATA  (master)      *
                        *-------------------------------------------------------------------*

000622 207900000000     TESTBUFF MOVE.L   TX.W+6,A0           point to transfer buffer
000628 323900000000             MOVE.W    TX.W+4,D1           get buffer length in D1
00062E 5341                     SUB.W     #1,D1               Less 1 for DBF instruction
000630 203900000000             MOVE.L    TDATA,D0            get current test data and
000636 227900000000             MOVE.L    TBS,A1              pointer to test buffer subr
00063C 4ED1                     JMP       (A1)                use B/W/L subr & return

                        *       Byte Buffer Test subroutine

00063E B018             BUFF_B  CMP.B     (A0)+,D0            comp with data in buffer
000640 6608                     BNE.S     BBX                 Error if not equal
000642 4600                     NOT.B     D0                  invert for testing
000644 51C9FFF8                 DBF       D1,BUFF_B           test the whole buffer
000648 4E75                     RTS                           return passing
00064A 91FC00000001     BBX     SUBA.L    #1,A0               calc error address
000650 13FC0002+                MOVE.B    #2,DSIZE            error size 2 = byte
000658 601A                     BRA.S     BUFF_X              and exit w/error

                        *       Word Buffer Test subroutine

00065A B058             BUFF_W  CMP.W     (A0)+,D0            comp with data in buffer
00065C 6608                     BNE.S     BWX                 Error if not equal
00065E 4640                     NOT.W     D0                  invert for testing
000660 51C9FFF8                 DBF       D1,BUFF_W           test the whole buffer
000664 4E75                     RTS                           return passing
```

```
000666 91FC00000002     BWX     SUBA.L   #2,A0              calc error addrsess
00066C 13FC0001+                MOVE.B   #1,DSIZE           error size 1 = word

                        *       Test buffer error exit

000674 08F90007+        BUFF_X  BSET     #7,FLAGS           set error bit in flags
00067C 4E75                     RTS                         return test buffer err

                        ********************************************************************
                        ***                                                              ***
                        **                 DRB32E Interrupt Handlers                     **
                        ***                                                              ***
                        ********************************************************************

                        *----------------------------------------------------------------*
                        *       Attention Handler (Command input interrupt) (both)       *
                        *----------------------------------------------------------------*

00067E 08F90006+        ATTN    BSET     #6,FLAGS           set attn bit in flags
000686 33F9FBFFE062+            MOVE.W   IODR,COMMAND       read command via IODR
000690 4E73                     RTE                         return from exception

                        *----------------------------------------------------------------*
                        *       Dma DONE & Derr Handlers (both)                          *
                        *----------------------------------------------------------------*

000692 08F90007+        DERR    BSET     #7,FLAGS           Set error bit in flags
00069A 08F90005+        DONE    BSET     #5,FLAGS           Set done bit in flags
0006A2                          PUSH     D0                 preserve registers
0006A6 1039FBFFE061             MOVE.B   BCR,D0             read board control
0006AC 020000FC                 AND.B    #$FC,D0            clear go and cycle bits
0006B0 13C0FBFFE061             MOVE.B   D0,BCR             and set board status
0006B6                          POP      D0                 restore registers
0006BA 13F9FBFFE060+            MOVE.B   BSR,STATS          read bsr into stats
0006C4 4E73                     RTE                         Return from Exception

                        *----------------------------------------------------------------*
                        *       BUS Error (board) Handler (both)                         *
                        *----------------------------------------------------------------*

0006C6 08F90007+        BERR    BSET     #7,FLAGS           set error bit in flags
0006CE 08F90000+                BSET     #0,FLAGS           set bus error bit in flags
0006D6 4E73                     RTE                         and return from exception

                        ********************************************************************
                        ***                                                              ***
                        **                    DRB32E Test messages                      **
                        ***                                                              ***
                        ********************************************************************

0006D8                  DR11_M
0006D8 0D0A0A            DC.B CR,LF,LF
0006DB 2A2A2A2A2A2A2A+   DC.B '**********************************************************',CR,LF
000719 2A202020202020+   DC.B '*                                                          *',CR,LF
000757 2A202020202020+   DC.B '*       ((((((  VMIVME DR11W TRANSFER TESTS  ))))))        *',CR,LF
```

```
000795 2A2020202020+   DC.B '*                                                    *',CR,LF
0007D3 2A2A2A2A2A2A+   DC.B '************************************************************',CR,LF
000811 2A2020496E7374+ DC.B '*  Install dr11w cable from P3 of UUT to P4 of Gold board  *',CR,LF
00084F 2A2020496E7374+ DC.B '*  Install dr11w cable from P4 of UUT to P3 of Gold board  *',CR,LF
00088D 2A2A2A2A2A2A+   DC.B '************************************************************',CR,LF,NUL
0008CC                 SLCT_M
0008CC 53656C65637420+  DC.B 'Select Transfer Test Startup mode M=Master, S=Slave (M/S): ',NUL
000908 203F0D0700      ARGH_M DC.B ' ?',CR,BEL,NUL

00090D 0D0A5072657373+ ANYK_M DC.B CR,LF,'Press any key to continue: ',nul

00092B 0D0A0A3C204D41+ MAST_M DC.B CR,LF,LF,'< MASTER MODE >',NUL
00093E 0D0A4D61737465+ MOTH_M DC.B CR,LF,'Master doing other tasks.',NUL
00095A 0D0A5265717565+ LREQ_M DC.B CR,LF,'Requesting link as Master.',NUL
000977 0D0A4C696E6B20+ LACK_M DC.B CR,LF,'Link request acknowledged.',CR,LF,NUL
000996 52656C65617369+ RELL_M DC.B 'Releasing link request.',NUL

0009AE 0D0A0A3C20534C+ SLAV_M DC.B CR,LF,LF,'< SLAVE  MODE >',NUL
0009C1 0D0A5265636569+ LINK_M DC.B CR,LF,'Received Link Request.',NUL
0009DA 0D0A4772616E74+ GRNT_M DC.B CR,LF,'Granting link to master.',nul
0009F5 0D0A5265636569+ EXIT_M DC.B CR,LF,'Received Exit command.',nul

000A0E 42757273742044+ BRST_M DC.B 'Burst Dma....',NUL
000A1C 4379636C652044+ CYCL_M DC.B 'Cycle Dma....',NUL

000A2A 576F72642E2E2E+ WORD_M DC.B 'Word....',nul
000A33 427974652E2E2E+ BYTE_M DC.B 'Byte....',nul

000A3C 5061737365640D+ PASS_M DC.B 'Passed',CR,LF,nul
000A45 4661696C65640D+ FAIL_M DC.B 'Failed',CR,LF,NUL

000A4E 0D0A0A50617373+ PCNT_M DC.B CR,LF,LF,'Passed ',NUL
000A59 2C204661696C65+ FCNT_M DC.B ', Failed ',NUL

000A63 0D0A5265636569+ NULL_M DC.B CR,LF,'Received NULL command.',NUL
000A7C 0D0A5265636569+ ACKN_M DC.B CR,LF,'Received ACKN command.',NUL
000A95 0D0A5265636569+ CERR_M DC.B CR,LF,'Received Invalid command word = ',nul

000AB8 4661696C656420+ BADB_M DC.B 'Failed ==> Bad DATA from "slave"!',CR,LF,NUL
000ADC 4661696C656420+ BADA_M DC.B 'Failed ==> Bad ACKN from "slave"!',CR,LF,NUL
000B00 4661696C656420+ BADD_M DC.B 'Failed ==> Bad DMAx from "slave"!',CR,LF,NUL

000B24 20416464726573+ ADDR_M DC.B ' Address: ',NUL
000B2F 2057726F74653A+ WROT_M DC.B ' Wrote: ',NUL
000B38 20526561643A20+ READ_M DC.B ' Read: ',NUL
000B40 204353523A2000  CHSR_M DC.B ' CSR: ',NUL
000B47 204253523A2000  BSR_M  DC.B ' BSR: ',NUL
000B4E 2053544154533A+ STAT_M DC.B ' STATS: ',NUL
000B57 20464C4147533A+ FLAG_M DC.B ' FLAGS: ',NUL

                       ********************************************************************
                       ***                                                              ***
                       **          DRB32E Test Constants, Variables, & Storage           **
                       ***                                                              ***
                       ********************************************************************
```

```
000B60 00000000          SVBR     DC.L      0                  save system vector base reg
000B64 00                STATS    DC.B      0                  board status register stg
000B66 0000              COMMAND  DC.W      0                  attn command word storage
000B68 1E                ACS      DC.B      $1E                attn interrupt control+level
                         *-------------------------------------------------------------------*
                         * Slave mode command table
                         *
000B6A 000003D6          SCMDS    DC.L      NULLCMD            0
000B6E 000003EA                   DC.L      ACKNCMD            1
000B72 000003E0                   DC.L      LINKCMD            2
000B76 00000590                   DC.L      SETBYTE            3
000B7A 00000566                   DC.L      SETWORD            4
000B7E 0000041E                   DC.L      SET_TX             5
000B82 0000043E                   DC.L      SET_RX             6
000B86 000004BE                   DC.L      SETBRST            7
000B8A 000004E0                   DC.L      SETCYCL            8
000B8E 0000045E                   DC.L      SET_GO             9
000B92 000003FC                   DC.L      EXITCMD            A
000B96 000003F4                   DC.L      ABRTCMD            B
000B9A 0000000B          VALID    DC.W      $0000,$000B        ^
                         *-------------------------------------------------------------------*
                         * registers:  csr|dcr|ocr|amr| mtc.w | mac.h | mac.l |ccr|dicr
                         *              0 | 1 | 2 | 3 | 4   5 | 6   7 | 8   9 | A | B
000B9E FF3812391000FB+   TX.W  DC.B    $FF,$38,$12,$39,$10,$00,$FB,$00,$00,$00,$88,$39
000BAA FF3002391000FB+   TX.B  DC.B    $FF,$30,$02,$39,$10,$00,$FB,$00,$00,$00,$88,$39
000BB6 FF3892391000FB+   RX.W  DC.B    $FF,$38,$92,$39,$10,$00,$FB,$00,$00,$00,$88,$39
000BC2 FF3082391000FB+   RX.B  DC.B    $FF,$30,$82,$39,$10,$00,$FB,$00,$00,$00,$88,$39
                         *-------------------------------------------------------------------*
                         * Transfer mode pointers
                         *
000BCE 00000B9E          TXS      DC.L      TX.W               default is word tx table
000BD2 00000BB6          RXS      DC.L      RX.W               default is word rx table
000BD6 00000618          FBS      DC.L      FILL_W             def is word fill buff subr
000BDA 0000065A          TBS      DC.L      BUFF_W             def is word test buff subr
                         *-------------------------------------------------------------------*
                         * DRB32E Test Date Table (TDATA) note: end of table is null $0
                         *
000BDE AAAAAAAA+         DTBL     DC.L      $AAAAAAAA,$55555555,$CCCCCCCC,$33333333
000BEE F0F0F0F0+                  DC.L      $F0F0F0F0,$FF00FF00,$FFFF0000,$FFFFFFFF
000BFE 00000000                   DC.L      $00000000
000C02 00000BDE          DPTR     DC.L      DTBL               test data table pointer
000C06 00000000          TDATA    DC.L      0                  current test data variable
000C0A 00                DSIZE    DC.B      0                  current test data size
                         ***********************************************************************
                                  END       DR11W4
 0xC0C bytes code/data,  0x0 uninitialized
```

DOCUMENTATION EVALUATION FORM

VMIC welcomes your comments and suggestions.

Please return this form to:

VME MICROSYSTEMS INTERNATIONAL CORPORATION
12090 South Memorial Parkway
Huntsville, Alabama 35803-3308
(205) 880-0444
1-800-322-3616

Evaluation: Please rate the following areas on a scale of 1 to 5 (1 = Poor; 5 = Excellent).

DOCUMENT NO.: _____
REVISION DATE: _____

READABILITY	_____	ILLUSTRATIONS	_____
ORGANIZATION	_____	PROGRAMMING INFORMATION	_____
ACCURACY	_____	SPECIFICATIONS	_____
COMPLETENESS	_____	MAINTENANCE DIAGRAMS	_____

SPECIFIC PROBLEMS: **PAGE(s)**

( ) CLARIFICATION REQUIRED _____

( ) NOT ENOUGH INFORMATION GIVEN _____

( ) TYPOGRAPHICAL ERRORS _____

( ) TECHNICAL ERRORS (EXPLAIN): _____

DOCUMENT USE: (check all that apply)

( ) HARDWARE	( ) SOFTWARE	( ) PRODUCT EVALUATION
( ) OPERATION	( ) MAINTENANCE	( ) TRAINING

ADDITIONAL COMMENTS: _____

YOUR NAME: _____
TITLE: _____
COMPANY: _____
MAIL STOP: _____
STREET: _____
CITY, STATE, ZIP: _____
PHONE: _____

Artisan Technology Group is an independent supplier of quality pre-owned equipment

Gold-standard solutions

Extend the life of your critical industrial, commercial, and military systems with our superior service and support.

We buy equipment

Planning to upgrade your current equipment? Have surplus equipment taking up shelf space? We'll give it a new home.

Learn more!

Visit us at **artisantg.com** for more info on price quotes, drivers, technical specifications, manuals, and documentation.

Artisan Scientific Corporation dba Artisan Technology Group is not an affiliate, representative, or authorized distributor for any manufacturer listed herein.

We're here to make your life easier. How can we help you today?

(217) 352-9330 | sales@artisantg.com | artisantg.com

ARTISAN® TECHNOLOGY GROUP