FEB 89
Rudi Gebert
Truvelo

M V M E 1 0 1 B U G

D E B U G P A C K A G E

U S E R ' S M A N U A L

Third Edition, February 1986

TABLE OF CONTENTS

		Page
Chapter 1	GENERAL INFORMATION	
1.1.	INTRODUCTION	1-1
1.2.	SYSTEM ENVIRONMENT	1-1
1.3.	REFERENCE MANUALS	1-1
Chapter 2	SYSTEM INSTALLATION	
2.1.	MVME101 MONOBOARD COMPUTER	2-1
2.1.1.	Address Decoder PROM	2-1
2.1.2.	User Defined Address Maps	2-5
2.1.3.	MVME101bug Installation	2-5
2.1.4.	ROM Access Time	2-6
2.1.5.	RAM Installation	2-6
2.1.6.	VMEbus Requester Priority	2-6
2.1.7.	System Control Functions	2-7
2.1.8.	User-Vectorized Interrupts	2-7
2.1.9.	Auto-Vectorized Interrupts	2-7
2.1.10.	Serial Port 1 Configuration	2-8
2.1.11.	Serial Port 2 Configuration	2-8
2.1.12.	Serial Interface Control	2-8
2.1.13.	Programmable Timer Configuration	2-8
2.2.	OTHER VME SYSTEM COMPONENTS	2-9
2.3.	TERMINAL CONFIGURATION	2-9
2.4.	PRINTER CONNECTION	2-9
2.5.	MVME101 INITIALIZATION	2-9
Chapter 3	MVME101BUG COMMANDS	
3.1.	COMMAND INPUT	3-1
3.1.1.	Command Syntax	3-1
3.1.2.	Command Line Format	3-2
3.1.3.	Address Parameters	3-2
3.1.4.	Data Parameters	3-3
3.1.5.	Other Parameters	3-3
3.1.6.	Command Echo Back	3-4
3.1.7.	Exceptions and Errors	3-4
3.2.	COMMAND SET	3-7
3.2.1.	BD - Bootstrap Dump	3-8
3.2.2.	BF - Block of Memory Fill	3-10
3.2.3.	BH - Bootstrap Halt	3-11
3.2.4.	BI - Block of Memory Initialize	3-13
3.2.5.	BM - Block of Memory Move	3-14
3.2.6.	BO - Bootstrap Program	3-15
3.2.7.	BR / NOBR - Breakpoint Set / Remove	3-17
3.2.8.	BS - Block of Memory Search	3-19
3.2.9.	BT - Block of Memory Test	3-20
3.2.10.	DC - Data Conversion	3-21
3.2.11.	DF - Display Formatted MPU Registers	3-22
3.2.12.	Display/Set Registers	3-23
3.2.13.	DU - Dump Memory (S-Records)	3-24
3.2.14.	GD - Go Direct Execute Program	3-25
3.2.15.	GO - Go Execute Program	3-26

TABLE OF CONTENTS

		Page
3.2.16.	GT - Go Execute Program to Temporary Breakpoint	3-27
3.2.17.	HE - Help	3-28
3.2.18.	IOP - Disk I/O Physical	3-29
3.2.19.	IOT - Disk I/O Teach	3-32
3.2.20.	LO - Load Memory (S-Records)	3-39
3.2.21.	MD - Memory Display/Disassembly	3-40
3.2.22.	MM - Memory Modify/Disassembly/Assembly	3-41
3.2.23.	MS - Memory Set	3-44
3.2.24.	OF - Display All Relative Offsets	3-45
3.2.25.	PA / NOPA - Printer Attach / Detach	3-46
3.2.26.	PF - Port Format	3-47
3.2.27.	TM - Transparent Mode	3-48
3.2.28.	TR - Trace	3-49
3.2.29.	TT - Trace to Temporary Breakpoint	3-51
3.2.30.	VE - Verify Memory (S-Records)	3-53

Chapter 4	TRAP #15 USER I/O ROUTINES	
4.1.	INTRODUCTION	4-1
4.2.	CONVERT BINARY DATA TO ASCII STRING	4-2
4.3.	CONVERT ASCII STRING TO BINARY DATA	4-2
4.4.	INITIALIZE BOTH SERIAL PORTS	4-3
4.5.	CHANGE BREAK HANDLER ENTRY ADDRESS	4-3
4.6.	CHECK FOR BREAK CONDITION	4-3
4.7.	RETURN TO MVME101bug	4-4
4.8.	RECEIVE ASCII CHARACTER	4-4
4.9.	TRANSMIT ASCII CHARACTER	4-4
4.10.	RECEIVE ASCII STRING	4-5
4.11.	TRANSMIT ASCII STRING	4-5

APPENDICES

APPENDIX A	SOFTWARE ABORT	A-1
APPENDIX B	S-RECORD FORMAT	B-1
APPENDIX C	MVME101bug MESSAGES	C-1
APPENDIX D	CENTRONICS PRINTER INTERFACE	D-1

LIST OF TABLES

Table 2.1:	Standard Address Map	2-3
Table 2.2:	I/O-Register Address Map	2-4
Table 3.1:	MVME101bug Exception Routines	3-6
Table 3.2:	MVME101bug Commands	3-7
Table 4.1:	TRAP #15 User I/O Routines	4-1

LIST OF FIGURES

Figure 2.1:	MVME101 Memory and Jumper Locations	2-2
Figure D.1:	Centronics Printer Interface Schematic Diagram	D-1
Figure D.2:	Centronics Printer Interface Assembly	D-1

CHAPTER 1

GENERAL INFORMATION

1.1. INTRODUCTION

This Manual describes the MVME101bug 3.1 Debug Package. The package includes two EPROMs with the MVME101bug object code, four static RAMs, and an Address Decoder PROM for installation on the MVME101 MC68000 Monoboard Computer. MVME101bug 3.1 is also available on disk as a package of source and object modules which will run under VERSAdos in Motorola MC68000 development systems. The latter option facilitates the creation of a debug/monitor which meets the specific requirements of an application.

1.2. SYSTEM ENVIRONMENT

MVME101bug 3.1 is a firmware-resident debug package, ready for installation on the MVME101 MC68000 Monoboard Computer. The program uses the on-board I/O-devices for peripheral communications and requires only a terminal connected with Serial Port 1 to function. MVME101bug 3.1 uses this port for all communications with the operator such as command inputs and message outputs. For serial data input/output, another terminal or computer may be connected with Serial Port 2. The Peripheral Interface Adapter may be used to drive a Centronics compatible printer through the lower rear connector P2. MVME101bug 3.1 provides the necessary driver routines.

MVME101bug 3.1 provides several basic disk I/O functions in conjunction with an MVME319 Intelligent Disk/Tape Controller module and various hard disk, floppy disk, and FloppyTape drives. These disk I/O functions include disk controller initialization, physical sector read/write, and program bootstrap operations.

1.3. REFERENCE MANUALS

The following manuals may be used for further information about the system environment in which MVME101bug 3.1 is used:

* MOTOROLA MC68000 16-bit Microprocessor User's Manual
* MOTOROLA MVME101 MC68000 Monoboard Computer User's Manual
* MOTOROLA MVME319 Intelligent Disk/Tape Controller User's Manual

CHAPTER 2

SYSTEM INSTALLATION

2.1. MVME101 MONOBOARD COMPUTER

The following sections describe address map configuration, memory installation, and jumper configurations on the MVME101 Monoboard Computer for use with MVME101bug 3.1 and VERSAdos. For functional description and configuration details, refer to the MVME101 Monoboard Computer User's Manual. Figure 2.1 shows the device locations to be configured on the MVME101 module.

2.1.1. Address Decoder PROM

The Address Decoder PROM delivered in the MVME101bug 3.1 package must be installed at location U49 on the MVME101 module and defines the standard system address map shown in Table 2.1.

Addresses $000000-$001FFF are assigned to on-board RAM for MPU exception vectors, stack, and MVME101bug 3.1 scratchpad area.

Addresses $002000-$002FFF are assigned to optional on-board RAM for user programs.

The MVME101bug 3.1 program resides in on-board ROM and is located at addresses $F00000-$F07FFF.

The on-board I/O-devices on the MVME101 module occupy the address segment $FE0000-$FE0FFF. An I/O-register address list is given in Table 2.2.

The upper 64K bytes in the address map are dedicated to I/O-devices on the VMEbus which are accessed with Short I/O Address encoding in the address modifiers, such as the MVME319 Intelligent Disk/Tape Controller.

All remaining addresses in the map are decoded as VMEbus Standard Addresses for access to off-board memory and memory-mapped devices.

Figure 2.1: MVME101 Memory and Jumper Locations

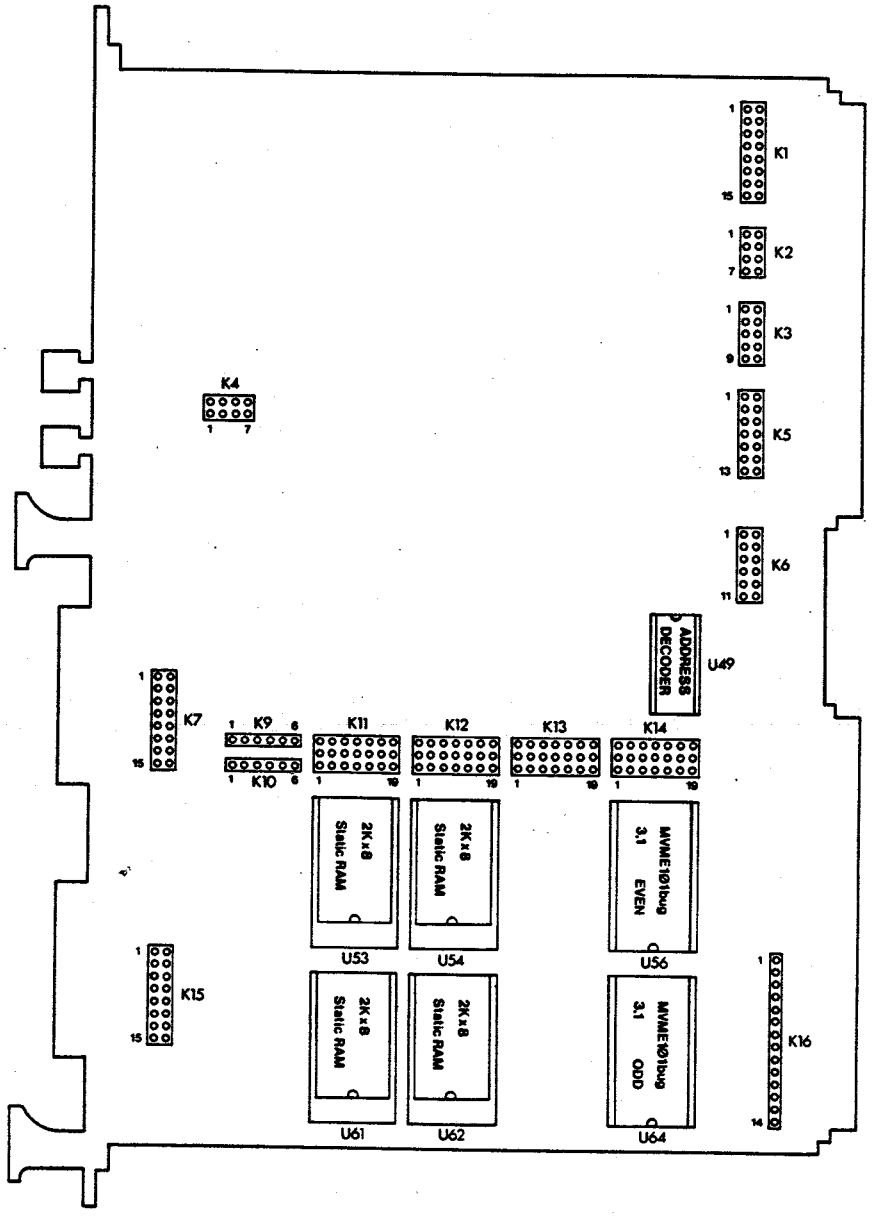


Table 2.1: Standard Address Map

ADDRESS	CONTENTS	SELECTED DEVICES
$FFFFFF : $FF0000	VMEbus Short I/O Addresses	Global I/O-devices
$FEFFFF : $FE1000	VMEbus Standard Addresses	Global Memory or Memory-mapped Devices
$FE0FFF : $FE0000	On-board I/O Registers (Only odd addresses used)	Local I/O-devices
$FDFFFF : $F08000	VMEbus Standard Addresses	Global Memory or Memory-mapped Devices
$F07FFF : $F00000	MVME101bug 3.1 Debug Package	2 x 16K bytes Local ROM in Memory Socket Pair 4
$EFFFFF : : : : : : : : : : : : : : $003000	VMEbus Standard Addresses	Global Memory or Memory-mapped Devices
$002FFF : $002000	User Program/Data	2 x 2K bytes Local RAM in Memory Socket Pair 3
$001FFF : $001000	MVME101bug 3.1 Data/Stack	2 x 2K bytes Local RAM in Memory Socket Pair 2
$000FFF : $000400	MVME101bug 3.1 Data/Stack	2 x 2K bytes Local RAM in Memory Socket Pair 1
$0003FF : $000000	MPU Exception Vectors	

Table 2.2: I/O-Register Address Map

DEVICE	ADDRESS	MODE	REGISTER
MCR	$FE00F1	r/w	Module Control Register
MSR	$FE00E1	r/w	Module Status Register
PTM	$FE00DF	read	LSB buffer register
	$FE00DF	write	Timer #3 latches
	$FE00DD	read	Timer #3 counter
	$FE00DD	write	MSB buffer register
	$FE00DB	read	LSB buffer register
	$FE00DB	write	Timer #2 latches
	$FE00D9	read	Timer #2 counter
	$FE00D9	write	MSB buffer register
	$FE00D7	read	LSB buffer register
	$FE00D7	write	Timer #1 latches
	$FE00D5	read	Timer #1 counter
	$FE00D5	write	MSB buffer register
	$FE00D3	read	status register
	$FE00D3	write	control register #2
	$FE00D1	read	no operation
	$FE00D1	write	CR20 = 1: control register #1
	$FE00D1	write	CR20 = 0: control register #3
PIA	$FE00C7	r/w	Section B control register
	$FE00C5	r/w	CRB-2 = 1: Section B peripheral register
	$FE00C5	r/w	CRB-2 = 0: Section B data direction register
	$FE00C3	r/w	Section A control register
	$FE00C1	r/w	CRA-2 = 1: Section A peripheral register
	$FE00C1	r/w	CRA-2 = 0: Section A data direction register
PCI2	$FE00B7	r/w	command register
	$FE00B5	r/w	mode register #1 / mode register #2
	$FE00B3	read	status register
	$FE00B3	write	SYN1 register / SYN2 register / DLE register
	$FE00B1	read	receive holding register
	$FE00B1	write	transmit holding register
PCI1	$FE00A7	r/w	command register
	$FE00A5	r/w	mode register #1 / mode register #2
	$FE00A3	read	status register
	$FE00A3	write	SYN1 register / SYN2 register / DLE register
	$FE00A1	read	receive holding register
	$FE00A1	write	transmit holding register

2.1.2. User Defined Address Maps

If the standard address map described above does not meet your actual requirements, you may specify any other configuration, and program the address decoder PROM accordingly. A detailed step-by-step description of this procedure is given in the MVME101 MC68000 Monoboard Computer User's Manual.

Keep the following considerations in mind when you define a new address map with MVME101bug 3.1:

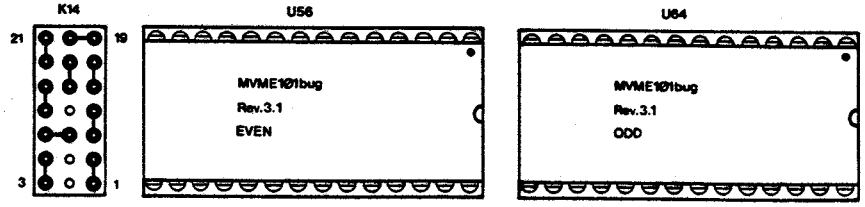
The MVME101bug EPROMs are intended for use in memory socket pair 4 on the MVME101 module. Therefore, the first two long word addresses of the EPROMs contain the initial stack pointer and program counter, and MVME101bug is entered immediately after a board reset. However, the object code of MVME101bug is position-independent and will execute anywhere in memory. If the debug package is moved, it must be moved as a unit, i.e. no portion must be moved unless it all is moved. The entry address of the moved MVME101bug code can be calculated by adding the displacement offset to the original value of the initial program counter.

Stack and scratchpad RAM is addressed absolutely, i.e. these addresses remain unchanged, even if MVME101bug is moved or linked upon another address in the map. MVME101bug requires the lower 8K bytes in the address map for MPU exception vectors, stack, and temporary data storage. You may assign these addresses either to on-board or off-board RAM; however, for optimum system performance, it is recommended to supply on-board RAM at these locations.

All on-board and off-board I/O-devices are also addressed absolutely. The address segment $FE0000-$FE0FFF must be assigned to the local I/O-devices. If an MVME319 Intelligent Disk/Tape Controller is installed in the system, the uppermost 64K addresses $FF0000-$FFFFFF must be assigned to VMEbus Short I/O.

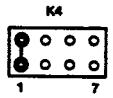
2.1.3. MVME101bug Installation

MVME101bug 3.1 is contained in a pair of 27128-type EPROMs. Insert the EPROM device with the even address locations in memory socket U56, the EPROM device with the odd address locations in memory socket U64. Configure the corresponding jumper area K14 for 27128-type EPROMs. The figure below shows the installation of EPROMs and jumpers.



2.1.4. ROM Access Time

Jumper area K4 determines the local ROM access time and must be configured according to the MVME101bug EPROM access time. The figure below shows the K4 configuration for an access time of 450 ns.

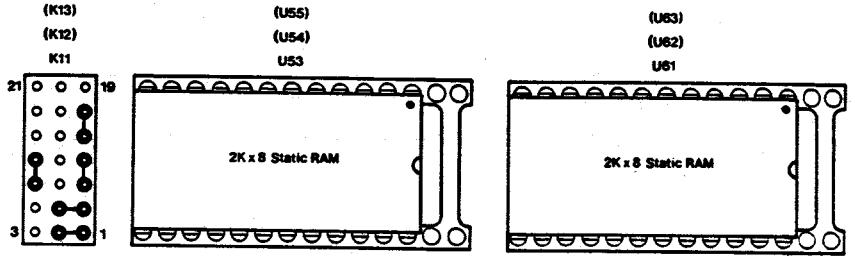


2.1.5. RAM Installation

MVME101bug requires 8K bytes of RAM at the bottom of the address map for MPU exception vectors, stack, and MVME101bug scratchpad area. The MVME101bug package contains four static RAMs, organized in 2K x 8 bits, for this purpose. Insert the RAM devices in memory sockets U53, U54, U61 and U62, and configure the corresponding jumper areas K11 and K12 for 2K x 8 RAMs.

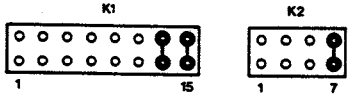
As user option, the Address Decoder PROM provides for another two 2K x 8 bits static RAM devices installed in memory sockets U55 and U63. MVME101bug does not access these devices, i. e., if installed, they are available for user programs.

The figure below shows the installation of RAMs and jumpers.



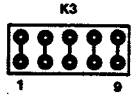
2.1.6. VMEbus Requester Priority

The VMEbus Requester on the MVME101 module has to operate on bus arbitration level 3. The figure below shows the configuration of the corresponding jumper areas K1 and K2.



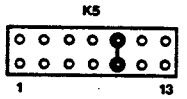
2.1.7. System Control Functions

When used as system controller module, the MVME101 Monoboard Computer supports the system clock, system failure, system reset, and VMEbus arbiter functions. The figure below shows the configuration of the corresponding jumper area K3.



2.1.8. User-Vectorized Interrupts

Jumper area K5 determines which of the seven interrupt request lines on the VMEbus may interrupt the on-board MPU. Conventionally, the interrupter on the MVME319 Intelligent Disk/Tape Controller operates on interrupt level 3. Therefore, when an MVME319 module is installed, at least the level 3 VMEbus interrupt request line IRQ3* must be jumpered to the MVME101 Interrupt Handler, as shown in the figure below. The configuration of the remaining interrupt levels depends on the distribution of additional interrupters and interrupt handlers in the VME system and must be evaluated according to the actual application.

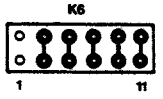


2.1.9. Auto-Vectorized Interrupts

Jumper area K6 determines the priority levels of interrupters which request auto-vectorized interrupts. MVME101bug does not respond to interrupts on levels 1-6. VERSAdos however requires the following configuration:

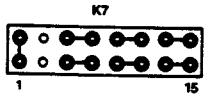
SYSFAIL*:	interrupt level 1
PTM:	interrupt level 2
PIA:	interrupt level 3
PCI1:	interrupt level 4
PCI2:	interrupt level 5

The figure below shows the corresponding jumper configuration.



2.1.10. Serial Port 1 Configuration

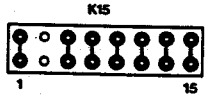
Jumper area K7 determines the pin assignment of Serial Port 1 connector SP1. Configure SP1 as asynchronous data set for terminal connection, as shown in the figure below.



2.1.11. Serial Port 2 Configuration

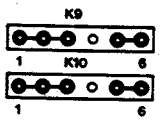
Jumper area K15 determines the pin assignment of Serial Port 2 connector SP2. K15 must be configured according to the device connected with SP2, which may be either a data terminal or a data set. As an example, the figure below shows the K15 configuration for connection with a computer being an asynchronous data set. For details on Serial Port 2 configuration, refer to the MVME101 Monoboard Computer User's Manual.

Changed to
DCE (Same as K7)



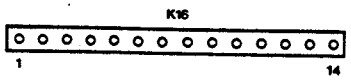
2.1.12. Serial Interface Control

VERSAdos requires that the interrupt outputs RXRDY and TXRDY of both Programmable Communication Interfaces are connected with the interrupt handler. Also, the CTS input of both devices must be controlled from the peripheral. The figure below shows the configuration of the corresponding jumper areas K9 and K10.



2.1.13. Programmable Timer Configuration

MVME101bug does not use the Programmable Timer Module and leaves the device free for user applications. VERSAdos however uses the PTM for periodic interrupt generation, and requires that no jumpers are placed on the corresponding jumper area K16.



2.2. OTHER VME SYSTEM COMPONENTS

The MVME101 Monoboard Computer and MVME101bug 3.1 may be used in conjunction with a wide variety of additional VME hardware and software products, such as the MVME200 series memory modules, the MVME319 Intelligent Disk/Tape Controller, the RMS68K Real-Time Multitasking Software, the VERSAdos Operating System, and many others.

For configuration and installation instructions of such additional VME system components refer to the user's manuals of your specific modules.

2.3. TERMINAL CONFIGURATION

The Serial Ports on the MVME101 module are initialized by MVME101bug for asynchronous operation, 8 bit character length, no parity, and one stop bit. The Serial Port signals RTS and DTR are constantly asserted. The data receivers do not rely on any control signal inputs.

After power-up or board reset, MVME101bug enters an automatic Baud rate detection routine, indicated by a "B" on the MVME101 STATUS display. MVME101bug now requires a carriage return ($0D) character being input at Serial Port 1 for measuring the terminal's Baud rate. The following Baud rates are valid:

110, 150, 300, 600, 1200, 1800, 2400, 3600, 4800, 7200, 9600, 19200 Baud.

After determining the Baud rate at Serial Port 1, MVME101bug initializes both serial ports identically. To support different port characteristics, you may reinitialize the serial ports with the PF command. Note that automatic Baud rate detection cannot be implemented at Serial Port 2.

2.4. PRINTER CONNECTION

MVME101bug supports a Centronics printer interface through the Peripheral Interface Adapter on the MVME101 module. Note that the PIA input/output lines at connector P2 are not buffered. You have to provide interfacing hardware to meet the electrical specifications of a Centronics interface. Appendix D gives an example for a connector/buffer/cable assembly.

2.5. MVME101 INITIALIZATION

After power-up or board reset, a 'B' is shown on the STATUS display of the MVME101 module. This indicates that MVME101bug is expecting a carriage return character from the terminal at Serial Port 1. After that, the program enters the initialization routines.

Both serial ports are initialized for asynchronous operation, 8 bit character length, no parity, one stop bit, and the detected Baud rate. The

Programmable Timer Module and the Peripheral Interface Adapter are reset. The Bus Request Time-Out Counter is disabled and the Data Transfer Time-Out Counter is enabled. The MPU exception vector table is initialized with the default addresses used by the debug package.

Then a "1" is shown on the STATUS display and the prompt message is transmitted to the terminal, indicating that MVME101bug is ready for command inputs.

When MVME101bug regains control after program execution, it will overwrite any number on the STATUS display with "1". Note that bits 5, 6, and 7 (BBTR, EBRTO, EDTTO) of the Module Control Register are not affected by that. Also, the I/O-devices on the MVME101 module are not re-initialized when returning to the MVME101bug prompt.

CHAPTER 3

MVME101BUG COMMANDS

3.1. COMMAND INPUT

Command entry is similar to that used in many buffer-organized systems. Typed characters, which may be either upper-case or lower-case, are accumulated in the command line buffer by the input routine. A command is processed for execution following entry of a carriage return character.

MVME101bug provides limited editing capabilities for command entry. The DELETE key, BACKSPACE key, or CTRL H can be used to delete previously entered characters. CTRL X cancels an entire line and waits for new input. CTRL D redisplays a previously entered line.

During output to the console, CTRL W suspends the output until another key is pressed. The BREAK key aborts the execution of console input/output commands.

Variations in the basic function offered by many primitive commands can be obtained by entering appropriate characters in the option field of a command line. Some command functions are switched off by prefixing the command with NO. For example, a breakpoint is set with the command BR and removed with the command NOBR.

3.1.1. Command Syntax

Commands and other I/O functions are presented in this manual in a modified Backus-Naur Form (BNF) syntax. Certain symbols in the syntax, where noted, are used in the real I/O format. However, others are meta-symbols whose usage is restricted to the syntactic structure. These meta-symbols and their meanings are as follows:

< > Angular brackets enclose a symbol, known as a syntactic variable, that is replaced in a command line by one of a class of symbols it represents.

[ ] Square brackets enclose a symbol that is optional. The enclosed symbol may occur zero or one time.

[ ]... Square brackets followed by periods enclose a symbol that is optional/repetitive. The enclosed symbol may appear zero or more times.

In the examples given in the following sections, the operator entries are shown underscored for clarity only, i.e., the underscore is not to be typed. Operator entries are followed by a carriage return unless otherwise specified.

3.1.2. Command Line Format

The format of the command line is:

MVME101bug 3.1 > [NO]<command>[<parameter>][;<options>]

MVME101bug 3.1 > is the MVME101bug prompt.

<command> is the primitive command.

<parameter> is a field containing of one or more parameters. Such parameters may be addresses, numbers, text, or other variables. Multiple parameters are separated by spaces. Although several commands do not limit the length of a parameter field, the total length of one command line must not exceed 100 characters.

<options> is a field opened with a semicolon, and containing one or more options. Multiple options need not be separated by spaces.

3.1.3. Address Parameters

An address, when used as a parameter, must follow the syntax accepted by the MC68000 assembler, except for the memory indirect mode. Numeric values may be entered in the same formats as data parameters (see section 3.1.4). The following address formats are accepted:

FORMAT	EXAMPLE	DESCRIPTION
address	140	absolute address
address+offset	130+R5	absolute address plus offset register value
(A@)	(A5)	address register indirect
(A@,D@)	(A6,D4)	address register indirect with index
address(A@)	120(A3)	address register indirect with displacement
address(A@,D@)	110(A2,D1)	address register indirect with index plus displacement
[address]	[100]	memory indirect (note that this addressing mode is not accepted by the assembler.)

3.1.4. Data Parameters

A data parameter is a numeric value which may be entered in binary, octal, decimal, or hexadecimal format. The format is specified by preceding the number with one of the following characters:

%	percent	specifies a binary number
@	commercial at	specifies an octal number
&	ampersand	specifies a decimal number
$	dollar	specifies a hexadecimal number
	none	specifies a hexadecimal number (default)

EXAMPLE: %10101010 = @252 = &170 = $AA = AA

Several commands allow arithmetic expressions consisting of multiple numbers in any format, separated by the operators plus (+) or minus (-).

EXAMPLE: %10101010+@3651-&4096+$3FFE

The data formats and maximum values which are valid for a specific command can be found in the command descriptions.

3.1.5. Other Parameters

There are several other types of parameters whose usage will be explained in the command descriptions. As an overview, some of these are summarized below:

PARAMETER	DESCRIPTION
<character>	a single ASCII character.
<controller#>	a single digit which specifies a disk controller module for disk I/O operations.
<count>	a single binary, octal, decimal or hexadecimal number following the same syntax rules as data parameters.
<drive#>	a single digit which specifies a disk drive for disk I/O operations.
<filename>	an ASCII string which specifies a VERSAdos file on a disk.
<mask>	a single binary, octal, decimal or hexadecimal number following the same syntax rules as data parameters.
<port>	a single digit which specifies an on-board I/O-device.
<text>	a string of visible ASCII characters.

3.1.6. Command Echo Back

As an aid to the user, MVME101bug displays for most commands its interpretation of the values entered as data and address parameters in a physical format. For example, if A000 had been typed in the parameter field of a command requiring that the contents of offset register R0 (1000, say) be added, then the resulting display would be 0000B000 for that address parameter:

```
MVME101bug 3.1 > .R0 1000

MVME101bug 3.1 > G A000
PHYSICAL ADDRESS=0000B000
```

3.1.7. Exceptions and Errors

MVME101bug shares resources with the target program under test, i.e., each affected resource can only be used by either the debug program or the target program at any given time.

The exception vectors in memory locations $000000 through $0003FF are initialized by MVME101bug in the reset routine after power-up or board reset. If the target program uses any of these vectors, they must be overwritten following each reset. The associated MVME101bug function for the overwritten vectors will be lost.

Table 3.1 lists the MC68000 exception vectors and gives a short description of the associated MVME101bug routines. Note that a Data Transfer Time-Out and a Bus Request Time-Out exception can occur only when the corresponding control bits in the Module Control Register are set.

When a MVME101bug exception routine is entered, first the complete MPU contents is saved on the stack. In case of Bus Errors and Address Errors, the exception status on the stack is also transmitted to the terminal:

```
XXXX  AAAAAAAA  IIII
  |       |       |
  |       |       +-- Instruction Register
  |       |           bits 0-15 : first word of aborted instruction
  |       |
  |       +---------- Access Address
  |                   bits 0-23 : address accessed during the aborted cycle
  |                   bits 24-31: not defined
  |
  +------------------ Miscellaneous Status
                      bits 0-2  : MPU function code
                      bit 3     : type of aborted process :
                                  0 = instruction or exception group 2
                                  1 = exception group 0 or group 1
                      bit 4     : data transfer direction :
                                  0 = write cycle
                                  1 = read cycle
                      bits 5-15 : not defined
```


Then, for all exception routines, a message is transmitted indicating the event which caused exception processing :

MESSAGE	VECTOR	EVENT
ACFAIL LOW	31	VMEbus ACFAIL* asserted
ADER EXCEPTION	3	Address error detected
AT BREAKPOINT	4	Breakpoint encountered
AV#1 EXCEPTION	25	Auto-vectorized interrupt level 1
: :	:	: :
AV#6 EXCEPTION	30	Auto-vectorized interrupt level 6
BERR EXCEPTION	2	VMEbus BERR* asserted
BRTO EXCEPTION	2	Bus Request Time Out occured
CHCK EXCEPTION	6	CHK instruction encountered
DIVO EXCEPTION	5	Division by zero
DTTO EXCEPTION	2	Data Transfer Time Out occured
ILLEGAL INSTRUCTION	4	Illegal operation code encountered
PRIV EXCEPTION	8	Privileged instruction in user state
SOFTWARE ABORT	31	ABORT pushbutton pressed
SPUR EXCEPTION	24	BERR during interrupt acknowledge
TP V EXCEPTION	7	TRAPV instruction encountered
TRAC EXCEPTION	9	Trace bit in status register set
UTO EXCEPTION	32	TRAP #$0 instruction encountered
: :	:	: : :
UTE EXCEPTION	46	TRAP #$E instruction encountered
1010 EXCEPTION	10	Unimplemented instruction encountered
1111 EXCEPTION	11	Unimplemented instruction encountered
???? EXCEPTION	12-23,48-255	Unitialized or unassigned exceptions

Below the exception message, the stacked contents of all MPU registers is displayed. The program counter points to the next instruction to be executed. This instruction is then disassembled and displayed below the register table in the standard assembler language format. At last, MVME101bug transmits the prompt message and is ready for command inputs from the console.

Note: The TRAP #15 instruction exception is reserved for user access to the MVME101bug I/O routines. See Chapter 4 for a detailed description.

Table 3.1: MVME101bug Exception Routines

VECTOR	ADDRESS	ASSIGNMENT	MVME101bug FUNCTION
0	$000-$007	Initial SSP and PC	Uninitialized
2	$008-$00B	Bus Error	Display bus error source (BERR, BRTO, DTTO), stack status, and MPU registers
3	$00C-$00F	Address Error	Display stack status and MPU registers.
4	$010-$013	Illegal Instruction	Breakpoint: Restore original op-code and display MPU registers.
5	$014-$017	Zero Divide	Display MPU registers.
6	$018-$01B	CHK Instruction	Display MPU registers.
7	$01C-$01F	TRAPV Instruction	Display MPU registers.
8	$020-$023	Privilege Violation	Display MPU registers.
9	$024-$028	Trace	Display MPU registers.
10	$028-$02B	1010 Code Emulator	Display MPU registers.
11	$02C-$02F	1111 Code Emulator	Display MPU registers.
12-14	$030-$03B	Unassigned, reserved	Display MPU registers.
15	$03C-$03F	Uninitialized IRQ	Display MPU registers.
16-23	$040-$05F	Unassigned, reserved	Display MPU registers.
24	$060-$063	Spurious IRQ	Display MPU registers.
25-30	$064-$07B	AV Interrupt Level 1-6	Display MPU registers.
31	$07C-$07F	AV Interrupt Level 7	Display interrupt source (ACFAIL, SOFTWARE ABORT) and MPU registers.
32-46	$080-$0BB	TRAP #0 - #14 Instr.	Display MPU registers.
47	$0BC-$0BF	TRAP #15 Instruction	Branch to I/O routine specified by the following word. (See Chapter 4)
48-63	$0C0-$0FF	Unassigned, reserved	Display MPU registers.
64-255	$100-$3FF	UV Interrupts	Display MPU registers.

3.2. COMMAND SET

This section describes the MVME101bug 3.1 commands by syntax, functions, and examples. Table 3.2 gives a summary of all MVME101bug 3.n commands. A complete list of all messages can be found in Appendix C.

Table 3.2: MVME101bug Commands

COMMAND SYNTAX	DESCRIPTION
.A .A0 - .A7 [<addr>/<data>] .D .D0 - .D7 [<addr>/<data>] .PC [<addr>/<data>] .SR [<addr>/<data>] .SS [<addr>/<data>] .US [<addr>/<data>] .R0 - .R6 [<addr>/<data>] DF OF	Display all address registers Display/set address register Display all data registers Display/set data register Display/set program counter Display/set status register Display/set superv. stack pointer Display/set user stack pointer Display/set relative offset Display formatted MPU registers Display all relative offsets
BF <addr1> <addr2> <data> BI <addr1> <addr2> BM <addr1> <addr2> <addr3> BS <addr1> <addr2> <data>/<text> BT <addr1> <addr2> MD[<port>] <addr>[<count>][;<opt>] MM <addr>[<;opts>] MS <addr> <data>/<text> ...	Block of memory fill Block of memory initialize Block of memory move Block of memory search Block of memory test Memory display/disassembly Memory modify/disassembly/assembly Memory set
BR [<addr1>[<;count1>]] ... NOBR [<addr1>] ... GO [<addr>] GT <addr> GD [<addr>] TR [<count>] TT <addr>	Breakpoint set Breakpoint remove Go execute program Go execute program to breakpoint Go direct execute program Trace one instruction Trace to temporary breakpoint
BD [<drv#>][,<ctr#>] BH [<drv#>][,<ctr#>] BO [<drv#>][,<ctr#>][,<fil>][;<opt>] IOP IOT	Bootstrap dump Bootstrap halt Bootstrap program Disk I/O physical Disk I/O teach
DU[<port>] <addr1> <addr2> [<text>] LO[;[<opts>][=<text>]] VE[;=<text>]	Dump memory (S-records) Load memory (S-records) Verify memory (S-records)
DC <addr>/<data> HE PA NOPA PF[<port>] TM [<char>]	Data conversion Help Printer attach Printer detach Port format Transparent mode

3.2.1. BD - Bootstrap Dump

BD [<drive#>][,<controller#>]

<drive#> is the logical number of the disk drive to be accessed (default = 0).

<controller#> is the logical number of the MVME319 Intelligent Disk/Tape Controller to be accessed (default = 0).

The BD command dumps the data from all local RAM on the MVME101 Monoboard Computer and from all global VME system RAM through the MVME319 Intelligent Disk/Tape Controller onto a VERSAdos disk. This feature is useful for saving the actual system status after a system crash. The dumped data may then be analyzed with the VERSAdos dump analysis utility DUMPANAL (see VERSAdos System Facilities Reference Manual for details).

The MVME319 module to be accessed is specified by its corresponding logical controller number. Up to eight MVME319 modules may be placed in the system with their command channels located adjacently in the address map, starting at address $FF0000. The controller number defines the command channel address used by the BD command as the number of 512 byte increments relative to the base address $FF0000. If no <controller#> parameter is specified in the command line, the BD command assumes a default value of 0.

The disk to be accessed is specified by a logical drive number corresponding with the device nomenclature used in VERSAdos. Anyone of up to four hard disk drives (<drive#> = 0-3) or of up to four floppy disk drives (<drive#> = 4-7) may be selected. If no <drive#> parameter is specified in the command line, the BD command assumes a default value of 0.

The BD command cannot be used with FloppyTape drives.

The memory data will be dumped on the disk into the dump area DUMP.SY which is generated during VERSAdos disk initialization (INIT utility). If there is no dump area on the disk, or if it is too small to store the complete system memory contents, an error message is displayed on the console, and control is returned to MVME101bug without dumping any data.

The BD command executes the following sequence:

1. The first 256 bytes of off-board RAM are made free by copying the contents to on-board RAM, starting beyond the scratchpad area at address $001E00. Then the volume ID (sector 0) of the specified disk is transferred into the free VME system RAM locations.

2. The volume ID locations $F8-$FF are read to ensure that they contain the string EXORMACS, which indicates a VERSAdos disk. If the string is not found, the message ERROR is displayed on the console and control is returned to MVME101bug.

3. The logical sector address (normally sector 1) of the disk configuration table is obtained from the volume ID locations $90-$93. The length of the disk configuration table (normally one sector) is obtained from the volume ID location $94.

4. If there is no configuration table on the disk, the default parameters (as initialized by system reset or by the IOT command) are used to write to the disk. If there is a configuration table on the disk, it is read into VME system RAM, and either an "Initialize Hard Disk Parameters" or an "Initialize Floppy Disk Parameters" command with the parameters given in the configuration table is transmitted to the MVME319 module.

5. The starting sector address of DUMP.SY is obtained from the volume ID locations $84-$87. If there is no dump area specified (i.e. $84-$87 = 0) the message DUMP FILE NOT FOUND is displayed on the console and control is returned to MVME101bug.

6. The length of DUMP.SY is obtained from the volume ID locations $88-$89. The dump area size is compared with the sum of local and global system RAM. If DUMP.SY is too small to store the complete system memory contents, the message DUMP.SY TOO SMALL is displayed on the console and control is returned to MVME101bug.

7. The saved copy of the first 256 bytes of off-board RAM is restored in the original locations.

8. The complete system memory contents is written into DUMP.SY, starting with local memory from address $000000 up to the end of on-board RAM, and then continuing with global memory from the first up to the last address of off-board RAM. DUMP.SY is an exact image of the system memory, i.e. the memory addresses are equivalent with the data locations within the dump file. If there is no off-board memory in the VME system, the message NO OFF-BOARD MEMORY is displayed on the console and control is returned to MVME101bug.

9. The message BD COMPLETE is displayed on the console and control is returned to MVME101bug.

EXAMPLES

```
MVME101bug 3.1 > BD                (Dump memory to hard disk #0 / IDTC #0)
BD COMPLETE

MVME101bug 3.1 > BD 6,1            (Dump memory to floppy disk #6 / IDTC #1)
BD COMPLETE
```

3.2.2. BF - Block of Memory Fill

BF <address1> <address2> <data>

<address1>	is the beginning address of the memory block to be filled.
<address2>	is the ending address of the memory block to be filled.
<data>	is the data word to be stored in the memory block.

The BF command fills all memory locations from <address1> through <address2> with <data>. Both addresses must be given as word boundaries (even addresses). <data> must not exceed word (2 bytes) size and may be expressed in binary, octal, decimal, or hexadecimal format. If data of less than word size is entered, the data is right-justified and leading zeros are inserted by MVME101bug.

EXAMPLE

```
MVME101bug 3.1 > MD 900
000900   FF FF 00 00 FF FF 00 00  FF FF 00 00 FF FF 00 00  ................

MVME101bug 3.1 > BF 900 90E 4E75
PHYSICAL ADDRESS=00000900 0000090E

MVME101bug 3.1 > MD 900
000900   4E 75 4E 75 4E 75 4E 75  4E 75 4E 75 4E 75 4E 75  NUNUNUNUNUNUNU
```


3.2.3. BH - Bootstrap Halt

BH [<drive#>][,<controller#>]

<drive#>	is the logical number of the disk drive to be accessed (default = 0).
<controller#>	is the logical number of the MVME319 Intelligent Disk/Tape Controller to be accessed (default = 0).

The BH command loads the Initial Program Load file IPL.SY from a VERSAdos disk through the MVME319 Intelligent Disk/Tape Controller into VME system memory (see the VERSAdos System Facilities Reference Manual for details on IPL.SY).

The MVME319 module to be accessed is specified by its corresponding logical controller number. Up to eight MVME319 modules may be placed in the system with their command channels located adjacently in the address map, starting at address $FF0000. The controller number defines the command channel address used by the BH command as the number of 512 byte increments relative to the base address $FF0000. If no <controller#> parameter is specified in the command line, the BH command assumes a default value of 0.

The disk to be accessed is specified by a logical drive number corresponding with the device nomenclature used in VERSAdos. Anyone of up to four hard disk drives (<drive#> = 0-3) or of up to four floppy disk drives (<drive#> = 4-7) may be selected. If no <drive#> parameter is specified in the command line, the BH command assumes a default value of 0.

The BH command cannot be used with FloppyTape drives.

The Initial Program Load file will be loaded into VME system memory starting at the address specified in the volume ID (sector 0) of the disk during system generation.

After loading IPL.SY, the BH command returns control to MVME101bug.

The BH command executes the following sequence:

1. The volume ID (sector 0) of the specified disk is transferred into RAM starting at the first VME system memory location.

2. The volume ID locations $F8-$FF are read to ensure that they contain the string EXORMACS, which indicates a VERSAdos disk. If the string is not found, the message ERROR is displayed on the console and control is returned to MVME101bug.

3. The logical sector address (normally sector 1) of the disk configuration table is obtained from the volume ID locations $90-$93. The length of the disk configuration table (normally one sector) is obtained from the volume ID location $94.

4. If there is no configuration table on the disk, the default parameters (as initialized by system reset or by the IOT command) are used to read the disk. If there is a configuration table on the disk, it is read into VME system RAM, and either an "Initialize Hard Disk Parameters" or

an "Initialize Floppy Disk Parameters" command with the parameters given in the configuration table is transmitted to the MVME319 module.

5. The file IPL.SY is read into VME system RAM starting at the load address specified in the volume ID locations $1E-$21. The starting sector address of IPL.SY is obtained from the volume ID locations $14-$17. The length of IPL.SY is obtained from the volume ID locations $18-$19.

6. After loading IPL.SY, control is returned to MVME101bug.

EXAMPLES

MVME101bug 3.1 > BH (Boot IPL.SY from hard disk #0 / IDTC #0)

MVME101bug 3.1 > BH 6,1 (Boot IPL.SY from floppy disk #6 / IDTC #1)

3.2.4. BI - Block of Memory Initialize

BI <address1> <address2>

<address1> is the beginning address of the memory block to be initialized.

<address2> is the ending address of the memory block to be initialized.

The BI command initializes the parity bits in off-board dynamic memory from <address1> through <address2>. Both addresses must be given as word boundaries (even addresses). The parity initialization is non-destructive, i.e., if a memory location contains correct parity, the data will not be changed. However, if a memory location contains incorrect parity, the data word $6D3F = "m?" will be stored in that location, forcing correct parity.

If the parity bit cannot be set, the message BERR EXCEPTION and the MPU registers are displayed on the console. The BT command may be used to isolate the failure (see section 3.2.9).

If no memory is resident at the specified addresses, the message DTTO EXCEPTION and the MPU registers are displayed on the console.

The BI command may take several seconds to initialize a large block of memory.

EXAMPLE

```
MVME101bug 3.1 > BI 14000 1FFFE
PHYSICAL ADDRESS=00014000 0001FFFE
```

3.2.5. BM - Block of Memory Move

BM <address1> <address2> <address3>

<address1>	is the beginning address of the source memory block.
<address2>	is the ending address of the source memory block.
<address3>	is the beginning address of the destination memory block.

The BM command duplicates the contents of all memory locations from <address1> through <address2> in another block of memory beginning at <address3>. The data may be moved either upward or downward in the memory map, and the source and destination memory blocks may overlap.

If no memory is resident at the specified addresses, the message DTTO EXCEPTION and the MPU registers are displayed on the console.

EXAMPLE

```
MVME101bug 3.1 > MD 900 A;DI
000900    1018              MOVE.B (A0)+,D0
000902    0C000000          CMP.B  #0,D0
000906    67F8              BEQ.S  $000900
000908    4E75              RTS

MVME101bug 3.1 > MD A00 A;DI
000A00    FFFF              DC.W   $FFFF
000A02    0000FFFF          OR.B   #-1,D0
000A06    0020FFFF          OR.B   #-1,-(A0)

MVME101bug 3.1 > BM 900 909 A00
PHYSICAL ADDRESS=00000900 00000909
PHYSICAL ADDRESS=00000A00

MVME101bug 3.1 > MD A00 A;DI
000A00    1018              MOVE.B (A0)+,D0
000A02    0C000000          CMP.B  #0,D0
000A06    67F8              BEQ.S  $000900
000A08    4E75              RTS
```


3.2.6. BO - Bootstrap Program

BO [<drive#>][,<controller#>][,<filename>][;<options>]

<drive#>	is the logical number of the disk drive to be accessed (default = 0).
<controller#>	is the logical number of the MVME319 Intelligent Disk/Tape Controller to be accessed (default = 0).
<filename>	specifies the file being loaded by the initial program loader (default = 0..VERSADOS.SY).
<options>	is one or both of the following options:
H	returns control to MVME101bug after the program has been loaded.
L=$xxxxxx	specifies the starting load address of the program in system memory.

The BO command loads a program from a VERSAdos disk through the MVME319 Intelligent Disk/Tape Controller into VME system memory. In addition to the program being loaded, the VERSAdos disk must contain the Initial Program Load file IPL.SY (see the VERSAdos System Facilities Reference Manual for details on IPL.SY).

The MVME319 module to be accessed is specified by its corresponding logical controller number. Up to eight MVME319 modules may be placed in the system with their command channels located adjacently in the address map, starting at address $FF0000. The controller number defines the command channel address used by the BO command as the number of 512 byte increments relative to the base address $FF0000. If no <controller#> parameter is specified in the command line, the BO command assumes a default value of 0.

The disk to be accessed is specified by its corresponding logical drive number. Anyone of up to four hard disk drives (<drive#> = 0-3) or of up to four floppy disk drives (<drive#> = 4-7) may be selected. If no <drive#> parameter is specified in the command line, the BO command assumes a default value of 0.

The BO command cannot be used with FloppyTape drives.

The <filename> parameter defines the program to be loaded by user number, catalog, filename and extension in the standard VERSAdos syntax. The default user number is 0, the default catalog is 8 blanks, the default filename is VERSADOS, the default extension is SY. If no <filename> is specified in the command line, the BO command will boot the VERSAdos operating system (file 0..VERSADOS.SY).

The program will be loaded into VME system memory starting at the address specified in the <option> field of the BO command. If no L option has been entered, the starting load address, as defined during system generation, is obtained from the information contained in the file IPL.SY.

After loading the specified program, the Initial Program Loader either gives control to that program, or, if the H option has been entered, returns to MVME101bug.

The BO command executes the following sequence:

1. The volume ID (sector 0) of the specified disk is transferred into RAM starting at the first VME system memory location.

2. The volume ID locations $F8-$FF are read to ensure that they contain the string EXORMACS, which indicates a VERSAdos disk. If the string is not found, the message ERROR is displayed on the console and control is returned to MVME101bug.

3. The logical sector address (normally sector 1) of the disk configuration table is obtained from the volume ID locations $90-$93. The length of the disk configuration table (normally one sector) is obtained from the volume ID location $94.

4. If there is no configuration table on the disk, the default parameters (as initialized by system reset or by the IOT command) are used to read the disk. If there is a configuration table on the disk, it is read into VME system RAM, and either an "Initialize Hard Disk Parameters" or an "Initialize Floppy Disk Parameters" command with the parameters given in the configuration table is transmitted to the MVME319 module.

5. The file IPL.SY is read into VME system RAM starting at the load address specified in the volume ID locations $1E-$21. The starting sector address of IPL.SY is obtained from the volume ID locations $14-$17. The length of IPL.SY is obtained from the volume ID locations $18-$19.

6. The MPU registers are initialized for IPL execution. If a <filename> parameter was specified in the BO command line, registers A5 and A6 point to the first and the last plus one characters of the string in the input buffer. If no filename was specified, register A5 is equal to A6, and the default filename 0..VERSADOS.SY will be taken. The status register is set to supervisor mode and interrupt mask level seven. The supervisor stack pointer is evaluated from the IPL.SY locations $0-$3. The program counter is evaluated from the IPL.SY locations $4-$7.

7. IPL.SY reads the program into VME system memory starting at the address specified in the <option> field of the BO command. If no L option has been entered, the starting load address is obtained from the IPL.SY locations $38-$3B.

8. After loading the specified program, IPL.SY gives control either to that program, or, if the H option has been entered, back to MVME101bug.

EXAMPLES

```
MVME101bug 3.1 > BO            (Boot VERSAdos from hard disk #0 / IDTC #0)

MVME101bug 3.1 > BO 6,1        (Boot VERSAdos from floppy disk #6 / IDTC #1)

MVME101bug 3.1 > BO ,,TEST.LO  (Boot 0..TEST.LO from hard disk #0 / IDTC #0)
```


3.2.7. BR / NOBR - Breakpoint Set / Remove

```
BR [<address>[;<count>]] [<address>[;<count>]] ...
NOBR [<address>] [<address>] ...
```

<address> is the memory address where program execution will stop.

<count> specifies the number of breakpoint encounters until program execution will stop.

The BR command, when used without parameters, displays all currently effective breakpoint addresses.

If one or more <address> parameters are entered, the BR command sets breakpoints at these addresses. Breakpoints can only be set on instruction addresses in RAM. A maximum of eight breakpoints may be set.

When program execution is started with the GO or the GT command, the original contents of the breakpoint address is saved in the breakpoint table and replaced by the illegal instruction op-code $4AFB. When the program encounters a breakpoint, the original instruction is restored, and the message AT BREAKPOINT and the MPU registers are displayed. The original op-code is also restored whenever MVME101bug regains control.

If program control is lost during the execution, and a board reset is used to regain control, breakpoints may be left in the user target program containing the illegal instruction op-code $4AFB.

While tracing through a program, the original contents of breakpoint addresses remains unchanged.

If a <count> parameter is specified, it will be decremented by one each time the program encounters the breakpoint. When <count> is zero, the program execution will stop at the breakpoint, and the breakpoint will be deleted from the breakpoint table. <count> may be expressed in binary, octal, decimal or hexadecimal format.

The NOBR command, when used without parameters, removes all breakpoints from the breakpoint table. If one or more <address> parameters are entered, the NOBR command removes the breakpoints from these addresses.

COMMAND FORMAT	DESCRIPTION
BR	Display all breakpoints.
BR <address>	Set a breakpoint on <address>.
BR <address>;<count>	Set a breakpoint on <address> with a count. Program execution will stop at the breakpoint after it has been encountered <count> times.
NOBR	Remove all breakpoints.
NOBR <address>	Remove a breakpoint from <address>.

EXAMPLE

```
MVME101bug 3.1 > BR 900 A00 14000 1000

BREAKPOINTS
000900    000900
000A00    000A00
014000    014000
001000    001000

MVME101bug 3.1 > NOBR A00 1000

BREAKPOINTS
000900    000900
014000    014000

MVME101bug 3.1 > NOBR

BREAKPOINTS

MVME101bug 3.1 > .R1 14000

MVME101bug 3.1 > BR 900+R1 A00+R1 14C00

BREAKPOINTS
000900+R1 014900
000A00+R1 014A00
000C00+R1 014C00

MVME101bug 3.1 > NOBR A00+R1

BREAKPOINTS
000900+R1 014900
000C00+R1 014C00
```


3.2.8. BS - Block of Memory Search

```
BS <address1> <address2> '<text>'
BS <address1> <address2> <data> [<mask>][;<option>]
```

<address1>	is the beginning address of the memory block.
<address2>	is the ending address of the memory block.
<text>	is the string to be found.
<data>	is the data to be found.
<mask>	specifies the bits to be compared.
<option>	is one of the following options:
	B search matching bytes.
	W search matching words (two bytes).
	L search matching long words (four bytes).
	If no option is entered, byte-sized data is used.

The BS command searches memory from <address1> through <address2> for the specified <text> or <data> parameter.

<text> parameters are entered as ASCII strings enclosed by quotation marks up to the input buffer length (100 characters).

<data> and <mask> parameters may be entered in binary, octal, decimal, or hexadecimal format and are right-justified and extended to the data size specified in the <option> field. The optional <mask> parameter defines the bits which are significant for the search. If a mask is given, it is logically ANDed with the memory data before the comparison is made.

All memory addresses where the specified pattern is found and the data contained in the addresses are displayed on the console.

EXAMPLE

```
MVME101bug 3.1 > MD 1000
001000   41 42 43 44 00 00 25 35  00 00 00 00 00 00 00 00  ABCD..%5........

MVME101bug 3.1 > BS 1000 1010 'ABCD'
PHYSICAL ADDRESS=00001000 00001010
001000 'ABCD'

MVME101bug 3.1 > BS 1000 1010 05 0F
PHYSICAL ADDRESS=00001000 00001010
001006 25
001007 35
```

3.2.9. BT - Block of Memory Test

BT <address1> <address2>

<address1> is the beginning address of the memory block to be tested.

<address2> is the ending address of the memory block to be tested.

The BT command tests all memory locations from <address1> through <address2>. Both addresses must be given as word boundaries (even addresses). The test is destructive. If the test is completed without detecting an error, all tested memory locations are cleared.

When a malfunction of a memory location is detected, the address, the data stored, and the data read of the failing memory location are displayed, and control is returned to MVME101bug.

The BT command may take several seconds to test a large block of memory.

If no memory is resident at the specified addresses, the message DTTO EXCEPTION and the MPU registers are displayed on the console.

EXAMPLE

```
MVME101bug 3.1 > BT 14000 17FFE
PHYSICAL ADDRESS=00014000 00017FFE

MVME101bug 3.1 > BT 18000 1FFFE
PHYSICAL ADDRESS=00018000 0001FFFE
FAILED AT 0180FE  WROTE=FFFF    READ=0000
```


3.2.10. DC - Data Conversion

DC <data>
DC <address>

<data> is the numerical value to be converted.

<address> is the address to be converted.

The DC command converts a <data> or <address> parameter into its hexadecimal and decimal equivalent.

<data> parameters may be entered in binary, octal, decimal, or hexadecimal format. A <data> parameter string of multiple values, separated by the arithmetic operators plus (+) or minus (-), is allowed.

<address> parameters may be entered as specified in section 3.1.3.

The result is displayed in hexadecimal and decimal format. Negative values are additionally shown as two's complement integer.

The offset registers R1-R6 are not used with the DC command. When R0 is set, its contents is added to the result of the DC translation.

The DC command is useful for calculating the displacement values of branch instructions or relative addressing modes.

EXAMPLE

```
MVME101bug 3.1 > DC &120
$78=&120

MVME101bug 3.1 > DC -1000
$FFFFF000=-$1000=-&4096

MVME101bug 3.1 > DC &15-$9+@14-%1100
$6=&6

MVME101bug 3.1 > .A0 1000

MVME101bug 3.1 > DC (A0)
$1000=&4096
```

3.2.11. DF - Display Formatted MPU Registers

DF

The DF command displays the current contents of all MPU registers on the console in hexadecimal data format. The program counter points to the next instruction to be executed. This instruction is disassembled and displayed in the standard assembler syntax.

The register display is also provided whenever MVME101bug gains control of the program execution, i.e., after exceptions, at breakpoints, and when tracing.

Any single MPU register can be displayed or changed with the Display/Set Registers Command described in section 3.2.12.

EXAMPLE

```
MVME101bug 3.1 > DF
PC=000000 SR=2704=.S7..Z.. US=FFFFFFFF SS=000007BC
D0=00010434 D1=230A0444 D2=04060444 D3=00000000
D4=00010031 D5=0000072C D6=00000004 D7=00000000
A0=00FE8001 A1=FFFFFFFF A2=00000454 A3=0000054E
A4=000131E0 A5=00002704 A6=00010158 A7=000007BC
--------------------000000    0F0F0F0F              MOVEP.W $0F0F(A7),D7

MVME101bug 3.1 > .PC 10000

MVME101bug 3.1 > .SS C00

MVME101bug 3.1 > DF
PC=010000 SR=2704=.S7..Z.. US=FFFFFFFF SS=00000C00
D0=00010434 D1=230A0444 D2=04060444 D3=00000000
D4=00010031 D5=0000072C D6=00000004 D7=00000000
A0=00FE8001 A1=FFFFFFFF A2=00000454 A3=0000054E
A4=000131E0 A5=00002704 A6=00010158 A7=00000C00
--------------------010000    4E75                  RTS
```


3.2.12. Display/Set Registers

.<register> [<data>]
.<register> [<address>]

<register>	specifies the register to be displayed or set.
<data>	is the numerical value to be stored in the register.
<address>	is the address to be stored in the register.

The .<register> command, when used without a parameter, displays the current contents of the specified register(s) on the console in hexadecimal data format.

The .<register> <data> command stores the <data> parameter value in the specified register. <data> may be entered in binary, octal, decimal, or hexadecimal format. A <data> parameter string of multiple values, separated by the arithmetic operators plus (+) or minus (-), is allowed.

The .<register> <address> command stores the <address> parameter value in the specified register. <address> may be entered in the formats specified in section 3.1.3.

COMMAND FORMAT	DESCRIPTION
.A	Display all address registers
.A0-.A7 [<data>/<address>]	Display/set address register
.D	Display all data registers
.D0-.D7 [<data>/<address>]	Display/set data register
.R0-.R6 [<data>/<address>]	Display/set relative offset register (see OF command)
.PC [<data>/<address>]	Display/set program counter
.SR [<data>/<address>]	Display/set MPU status register
.SS [<data>/<address>]	Display/set supervisor stack pointer
.US [<data>/<address>]	Display/set user stack pointer

EXAMPLE

```
MVME101bug 3.1 > .SS 1E00

MVME101bug 3.1 > .US 100(A7)

MVME101bug 3.1 > .US
.US=00001F00
```

3.2.13. DU - Dump Memory (S-Records)

DU[<port>] <address1> <address2> [<text>]

<port> specifies the output port number.

<address1> is the beginning address of the memory block to be dumped.

<address2> is the ending address of the memory block to be dumped.

<text> is an optional header string in the S0 record.

The DU command formats memory data from <address1> through <address2> in S-records and transmits it through the specified port. The first record output is a S0-record which contains the <text> parameter, if specified in the command line. The last record output is a S9-record. (See Appendix B for information on S-records.)

The DU command does not transmit control characters to start or stop peripheral devices. Prior to transmission, make sure that the specified port is configured properly for the connected device.

COMMAND FORMAT	DESCRIPTION
DU <addr1> <addr2>	Transmit S-records through Serial Port 1.
DU1 <addr1> <addr2>	Transmit S-records through Serial Port 1.
DU2 <addr1> <addr2>	Transmit S-records through Serial Port 2.
DU3 <addr1> <addr2>	Transmit S-records through Parallel Port.

Any <port> number other than 1, 2, or 3 is interpreted as Serial Port 1.

EXAMPLE

```
MVME101bug 3.1 > DU 900 90F DUMP TEST
PHYSICAL ADDRESS=00000900 0000090F
S00C000044554D5020544553545D
S2140009001018OC00000067F84E750000000000000BC
S9030000FC
```


3.2.14. GD - Go Direct Execute Program

GD [<address>]

<address> is the starting address of the program.

The GD command performs the following:

1. The number "0" is shown on the MVME101 STATUS display. Note that bits 5, 6, and 7 (BBTR, EBRTO, EDTTO) of the Module Control Register are unaffected.
2. The MPU registers are initialized as displayed by the DF command.
3. If an <address> parameter is given in the command line, the program counter is set to the specified value.
4. The target program execution starts immediately at the program counter address in real time.

The GD command does not change any of the exception vectors. Also, no breakpoints are set into the target program. However, all previously set breakpoints remain in the breakpoint table for future use.

EXAMPLE

```
MVME101bug 3.1 > MD A00 A;DI
000A00    1018              MOVE.B (A0)+,D0
000A02    0C000000          CMP.B  #0,D0
000A06    67F8              BEQ.S  $000A00
000A08    60FE              BRA    $000A08

MVME101bug 3.1 > BR A06 A08
BREAKPOINTS
000A06    000A06
000A08    000A08

MVME101bug 3.1 > GO A00
PHYSICAL ADDRESS=00000A00

AT BREAKPOINT
PC=000A06 SR=2700=.S7..... US=FFFFFFFF SS=00000C00
D0=00000034 D1=230A0444 D2=04060444 D3=00000000
D4=00010031 D5=0000072C D6=00000004 D7=00000000
A0=00FE8001 A1=FFFFFFFF A2=00000454 A3=0000054E
A4=000131E0 A5=00002704 A6=00010158 A7=00000C00
----------------000A06    67F8              BEQ.S  $000A00

MVME101bug 3.1 > GD A00
PHYSICAL ADDRESS=00000A00

(Program is executing ignoring any breakpoints)
```

3.2.15. GO - Go Execute Program

G[O] [<address>]

<address> is the starting address of the program.

The GO or G command performs the following:

1. The original contents of all breakpoint addresses is saved in the breakpoint table and replaced by the illegal instruction op-code $4AFB.
2. The original illegal instruction exception vector (address $10) is saved and replaced by the breakpoint routine pointer.
3. The number "0" is shown on the MVME101 STATUS display. Note that bits 5, 6, and 7 (BBTR, EBRTO, EDTTO) of the Module Control Register are unaffected.
4. The MPU registers are initialized as displayed by the DF command.
5. If an <address> parameter is given in the command line, the program counter is set to the specified value.
6. The target program execution starts at the program counter address by tracing the first instruction and then proceeding in real time.

When breakpoints with a count are reached, real time processing is not achieved. The program execution will not stop until the count is zero, but processing overhead is required for decrementing the count.

As the first instruction of the target program is traced, this instruction must not cause exception processing.

EXAMPLE

```
MVME101bug 3.1 > MD A00 A;DI
000A00     1018                 MOVE.B (A0)+,D0
000A02     0C000000             CMP.B  #0,D0
000A06     67F8                 BEQ.S  $000A00

MVME101bug 3.1 > BR A06

BREAKPOINTS
000A06     000A06

MVME101bug 3.1 > G A00
PHYSICAL ADDRESS=00000A00

AT BREAKPOINT
PC=000A06 SR=2700=.S7..... US=FFFFFFFF SS=00000C00
D0=00000034 D1=230A0444 D2=04060444 D3=00000000
D4=00010031 D5=0000072C D6=00000004 D7=00000000
A0=00FE8001 A1=FFFFFFFF A2=00000454 A3=0000054E
A4=000131E0 A5=00002704 A6=00010158 A7=00000C00
--------------------000A06     67F8                 BEQ.S  $000A00
```


3.2.16. GT - Go Execute Program to Temporary Breakpoint

GT <address>

<address> is the temporary breakpoint address.

The GT command performs the following:

1. The original contents of all breakpoint addresses is saved in the breakpoint table and replaced by the illegal instruction op-code $4AFB.
2. A temporary breakpoint is set on the specified address. Its original contents is saved and replaced by the illegal instruction op-code $4AFB.
3. The original illegal instruction exception vector (address $10) is saved and replaced by the breakpoint routine pointer.
4. The number "0" is shown on the MVME101 STATUS display. Note that bits 5, 6, and 7 (BBTR, EBRTO, EDTTO) of the Module Control Register are unaffected.
5. The MPU registers are initialized as displayed by the DF command.
6. The target program execution starts at the program counter address by tracing the first instruction and then proceeding in real time.

A temporary breakpoint must not be set at an address already existing in the breakpoint table.

When MVME101bug regains control before the specified address has been reached, the temporary breakpoint is reset.

As the first instruction of the target program is traced, this instruction must not cause exception processing.

EXAMPLE

```
MVME101bug 3.1 > MD A00 A;DI
000A00     1018                 MOVE.B (A0)+,D0
000A02     0C000000             CMP.B  #0,D0
000A06     67F8                 BEQ.S  $000A00


MVME101bug 3.1 > GT A06
PHYSICAL ADDRESS=00000A06
PHYSICAL ADDRESS=00000A00

AT BREAKPOINT
PC=000A06 SR=2700=.S7..... US=FFFFFFFF SS=00000C00
D0=00000034 D1=230A0444 D2=04060444 D3=00000000
D4=00010031 D5=0000072C D6=00000004 D7=00000000
A0=00FE8001 A1=FFFFFFFF A2=00000454 A3=0000054E
A4=000131E0 A5=00002704 A6=00010158 A7=00000C00
--------------------000A06     67F8                 BEQ.S  $000A00
```

3.2.17. HE - Help

HE

The HE command displays all commands available in MVME101bug on the console.

EXAMPLE

```
MVME101bug 3.1 > HE
.PC .SR .US .SS
.D0 .D1 .D2 .D3 .D4 .D5 .D6 .D7
.A0 .A1 .A2 .A3 .A4 .A5 .A6 .A7
.R0 .R1 .R2 .R3 .R4 .R5 .R6

BD   BF   BH   BI   BM   BO   BR   NOBR
BS   BT   DC   DF   DU   G    GD   GO
GT   HE   IOP  IOT  LO   M    MD   MM
MS   OF   PA   NOPA PF   T    TM   TR
TT   VE
```


3.2.18. IOP - Disk I/O Physical

IOP

The IOP command requests the MVME319 Intelligent Disk/Tape Controller to transfer data between a hard disk, floppy disk or FloppyTape and VME system memory. MVME101bug provides this command for verifying that disks and tapes can be addressed, read, and written, and for examining the physical data on the media. The IOP command asks whether data will be read or written, and which drive, media sectors and system memory addresses will be used, and then transmits either the "Read Sectors" or "Write Sectors" command to the MVME319 module.

As the disk or tape access is specified by block numbers rather than file names, any file security system will be bypassed. So be careful not to destroy structures on your media accidentally.

Never use your master disks with the IOP command !

Before accessing a disk or tape with the IOP command, the drive and media parameters must be initialized correctly with the Disk I/O Teach (IOT) command (see section 3.2.19).

When used with a FloppyTape, the first IOP operation must be reading sector #3 to load the bad segment list from the tape into the MVME319 module.

For detailed descriptions of the MVME319 module operations and logical disk and tape organizations refer to the MVME319 Intelligent Disk/Tape Controller User's Manual.

After being invoked, the IOP command enters an interactive subcommand mode where each currently effective operation parameter is displayed separately on the console. The operator then may either enter a new value, or leave the parameter unchanged by typing only carriage return. Note that all numerical parameters are interpreted as hexadecimal values by default. However, decimal values are also accepted when preceded by an ampersand sign (&).

In the following, each parameter is discussed in detail.

```
R/W=.....R  ?
```

R/W specifies whether data will be read from or written to the media. Type R or W, respectively.

```
MEM-ADD=.$00000000  ?
```

MEM-ADD corresponds with the starting memory address in the MVME319 commands and specifies the first address in system memory for the data transfer. The starting address may be any even address within the 16 Mbyte system address range. Note that the memory address must be accessible from the VMEbus, i.e. local memory on the MVME101 module cannot be used.

The address modifier code used by the IOP command is $3E (standard supervisory program access).

```
CTR-DRV=.......$00  ?
```

CTR-DRV is a two-digit parameter which corresponds with the controller and drive number in the MVME319 commands and specifies which drive will be accessed.

For selecting a hard disk, the first digit specifies the controller ($0x-$7x) according to the address jumper on the SASI/SCSI hard disk controller, the second digit specifies the hard disk drive ($x0-$x1) connected with that controller. For controller 0, the first digit may be omitted. Hard disks connected with controller 1 ($10 and $11) may alternatively be selected with CTR-DRV numbers $3 and $4.

For selecting a floppy disk, the first digit must be 0 (or may be omitted), the second digit specifies the floppy disk drive ($4-$7).

For selecting a FloppyTape, the first digit must be 0 (or may be omitted), the second digit must be $4.

```
IPC-NUM=.......$00  ?
```

IPC-NUM is the logical controller number ($0-$7) of the MVME319 module to be accessed. Up to eight MVME319 modules may be placed in the system with their command channels located adjacently in the address map, starting at address $FF0000. The IPC number defines the command channel address used by the IOP command as the number of 512 byte increments relative to the base address $FF0000.

```
1STSECT=.$00000000  ?
```

1STSECT corresponds with the continuous block number in the MVME319 commands and specifies the first sector to be read or written.

```
HOWMANY=.....$0000  ?
```

HOWMANY corresponds with the number of blocks in the MVME319 commands and specifies how many continuous blocks will be read or written.

```
SIZE=.....$0100  ?
```

SIZE corresponds with the block size in the MVME319 commands and specifies the number of bytes contained in one block. The block size may be 256 ($100), 512 ($200), or 1024 ($400) bytes, but must at least be equal to the physical sector size of the selected media. Note that VERSAdos uses 256 byte blocks on hard disks, floppy disks and FloppyTapes.

```
ARE YOU SURE, (Y/N)  ?
```

Before execution, the IOP command allows a verification of all specified parameters. If N is entered, the IOP command will be cancelled without accessing the media. If Y is entered, the parameter table will be closed and the disk or tape I/O performed.

The successful completion of the data transfer will be indicated by one of the following messages on the console:

```
"R" ZAPPED   ( for read from disk/tape )
"W" ZAPPED   ( for write to disk/tape )
```


In case of a malfunction, the IOP command will display an error message and the formatted MPU registers on the console. Register D7 will contain three bytes which represent the status type and status code received from the MVME319 module. This status information can be interpreted with the IDTC Status Messages descriptions in the MVME319 User's Manual.

EXAMPLE

```
MVME101bug 3.1 > MS 10000 'THIS IS A DISK WRITE/READ TEST'

MVME101bug 3.1 > IOP   (write string to floppy disk sector $100)
    R/W=.........R  ? W
MEM-ADD=.$00000000  ? 10000
CTR-DRV=.......$00  ? 4
IPC-NUM=.......$00  ? (CR)
1STSECT=.$00000000  ? 100
HOWMANY=.....$0000  ? 1
   SIZE=.....$0100  ? (CR)

ARE YOU SURE, (Y/N) ? Y

"W" ZAPPED

MVME101bug 3.1 > IOP   (read string from floppy disk to memory)
    R/W=.........W  ? R
MEM-ADD=.$00010000  ? 20000
CTR-DRV=.......$04  ? (CR)
IPC-NUM=.......$00  ? (CR)
1STSECT=.$00000100  ? (CR)
HOWMANY=.....$0001  ? (CR)
   SIZE=.....$0100  ? (CR)

ARE YOU SURE, (Y/N) ? Y

"R" ZAPPED

MVME101bug 3.1 > MD 20000 20   (verify string in memory)
020000   54 48 49 53 20 49 53 20  41 20 44 49 53 4B 20 57  THIS IS A DISK W
020010   52 49 54 45 5C 52 45 41  44 20 54 45 53 54 00 00  RITE/READ TEST..
```

3.2.19. IOT - Disk I/O Teach

IOT

The IOT command initializes the hard disk, floppy disk and FloppyTape parameters of the MVME319 Intelligent Disk/Tape Controller. The MVME319 module provides a unique and independent parameter table for each connected drive. After system reset the MVME319 firmware loads these tables with default values which must be re-initialized if the actual device parameters are different. For that purpose, MVME101bug provides the IOT command which asks the operator for the desired parameter values and then transmits either the "Initialize Hard Disk Parameters" or "Initialize Floppy Disk Parameters" or "Initialize FloppyTape Parameters" command to the MVME319 module.

For detailed descriptions of the MVME319 module operations and supported devices refer to the MVME319 Intelligent Disk/Tape Controller User's Manual.

Before accessing a disk or tape with the IOP command, or before booting from a disk without a configuration table, drive and media parameters must be initialized correctly with the IOT command.

The IOT command provides independent parameter sets for hard disks, 8 inch floppy disks, 5.25 inch floppy disks, and FloppyTapes. The default hard disk parameters support the ADAPTEC ACB4000 disk controller and the TANDON TM755 disk drive. The default floppy disk parameters have been chosen according to the characteristics of the VERSAdos media, as shipped from Motorola. The default FloppyTape parameters support the CIPHER CT525 FloppyTape drive.

After being invoked, the IOT command enters an interactive subcommand mode where each currently effective drive and media parameter is displayed separately on the console. The operator then may either enter a new value, or leave the parameter unchanged by typing only carriage return. Note that all numerical parameters are interpreted as decimal values by default. However, hexadecimal numbers are also accepted when preceded by a dollar sign ($).

In the following, the hard disk, floppy disk and FloppyTape parameters are discussed in separate sections.

Hard Disk Parameters

For hard disks, the following parameters are displayed and may be altered with the IOT command:

IDC number to be specified (0 - 7) .... { 0}

The IDC number is the logical controller number (0-7) of the MVME319 module to be initialized. Up to eight MVME319 modules may be placed in the system with their command channels located adjacently in the address map, starting at address $FF0000. The IDC number defines the command channel address used by the IOT command as the number of 512 byte increments relative to the base address $FF0000.

Drive number to be specified (0,1,4-7). { 0}

Drive numbers 0 and 1 select the respective hard disks connected with the controller specified below. Note that drive numbers 4 to 7 are reserved for floppy disk and FloppyTape drives.

SASI controller number (0 - 7) ..... { 0}

The SASI controller number (0-7) must be specified according to the address jumper configuration on the SASI/SCSI hard disk controller.

SASI controller type (0 - 2) ..... { 2}

This is a code number which specifies the SASI/SCSI hard disk controller type. 0 denotes a XEBEC S1410/S1410A controller, 2 denotes an ADAPTEC ACB4000 controller. Other numbers are invalid and reserved for future use.

Number of sectors per track ..... { 32}

The number of sectors per track depends on the physical sector size specified below. Enter 32 for 256 bytes, 17 for 512 bytes, 9 for 1024 bytes per sector.

Physical sector size in bytes ..... { 256}

The physical sector size may be 256 or 512 bytes with the XEBEC S1410/S1410A controller, and 256, 512, or 1024 bytes with the ADAPTEC ACB4000 controller. Note that VERSAdos uses 256 byte sectors on hard disks.

Number of cylinders per disk ..... { 981}

The number of cylinders per disk depends on the type of disk drive used and must be specified accordingly. The default value of 981 is valid for the TANDON TM755 disk drive.

First cylinder with compensation ..... { 490}

The number of the first cylinder with compensation depends on the type of disk drive used and must be specified accordingly. The default value of 490 is valid for the TANDON TM755 disk drive.

First cylinder with reduced current .. { 982}

The number of the first cylinder with reduced current depends on the type of disk drive used and must be specified accordingly. If reduced write current is not required, enter a value greater than the number of cylinders per disk.

Interleave factor ..... { 2}

The optimum value for the interleave factor depends on the type of hard disk controller used. Recommended values are 5 for the XEBEC S1410/S1410A controller, and 2 for the ADAPTEC ACB4000 controller. A value of 1 results in no interleaving.

Stepping rate code ..... { 2}

This is a code number which specifies the stepping rate as shown below.

0 = 3 ms not buffered (S1410/S1410A and ACB4000)
1 = 28 us buffered (ACB4000)
2 = 12 us buffered (ACB4000)
4 = 200 us buffered (S1410/S1410A)
5 = 70 us buffered (S1410/S1410A)
6 = 30 us buffered (S1410/S1410A)
7 = 15 us buffered (S1410/S1410A)

The optimum stepping rate depends on the type of disk drive used. The default value of 12 us is valid for the TANDON TM755 disk drive.

Number of heads ..... { 5}

The number of heads depends on the type of disk drive used and must be specified accordingly. The default value of 5 is valid for the TANDON TM755 disk drive.

ECC data burst length ..... { 0}

The ECC data burst length depends on the type of SASI/SCSI hard disk controller used and must be specified accordingly. The default value 0 is valid for the ADAPTEC ACB4000 controller.

Floppy Disk Parameters

For floppy disks, the following parameters are displayed and may be altered with the IOT command:

IDC number to be specified (0 - 7) .... { 0}

The IDC number is the logical controller number (0-7) of the MVME319 module to be initialized. Up to eight MVME319 modules may be placed in the system with their command channels located adjacently in the address map, starting at address $FF0000. The IDC number defines the command channel address used by the IOT command as the number of 512 byte increments relative to the base address $FF0000.

Drive number to be specified (0,1,4-7). { 0}4

If no FloppyTape drive is connected, drive numbers 4 to 7 select the respective floppy disks. If a FloppyTape drive is connected, drive number 4 selects the FloppyTape, drive number 5 is not available and must not be used with any command, and drive numbers 6 and 7 select the respective floppy disks. Drive numbers 0 and 1 are reserved for hard disk drives.

Floppy disk or FloppyTape drive (F/T) ... {F} for drive #4

This parameter is only displayed if drive number 4 is selected, and specifies whether drive 4 is a floppy disk or FloppyTape drive. Enter F to select a floppy disk.

8" or 5 1/4" disk size (8/5) ..... {8}

The disk size may be either 8 or 5.25 inch. Enter 8 or 5, respectively. Depending on the given size, the IOT command will select different default values for some of the following parameters.

Single or double density (S/D) ..... {S} for 8"
Single or double density (S/D) ..... {D} for 5.25"

For either disk size, single (FM) or double (MFM) data density may be specified. Enter S or D, respectively.

Single or double sided (S/D) ..... {D} for 8" and 5.25"

For either disk size, single or double sided media may be specified. Enter S or D, respectively.

MOTOROLA or IBM format (M/I) ..... {M} for 8"
MOTOROLA or IBM format (M/I) ..... {I} for 5.25"

For either disk size, Motorola or IBM format may be specified. Enter M or I, respectively. Note that VERSAdos is shipped in Motorola format on 8 inch diskettes, and in IBM format on 5.25 inch diskettes.

Pre-, no, or post- compensation (+/0/-) . {+} for 8"

For 8 inch disk size, write pre-compensation, no compensation, or read post-compensation may be selected. Enter +, 0, or -, respectively. 5.25 inch disks do not use compensation.

Spiral offset (Y/N) ..... {N} for 8" and 5.25"

For either disk size, you may specify a spiral offset or none. Enter Y or N, respectively.

Number of sectors per track ..... { 26} for 8"
Number of sectors per track ..... { 16} for 5.25"

The number of sectors per track depends on disk size, data density, and physical sector size as shown below.

disk size	5.25 inch						8 inch					
data density	single (FM)			double (MFM)			single (FM)			double (MFM)		
# bytes per sector	128	256	512	256	512	1024	128	256	512	256	512	1024
# sectors per track	16	9	5	16	9	5	26	15	8	26	15	8

Physical sector size in bytes ..... { 128} for 8"
Physical sector size in bytes ..... { 256} for 5.25"

The physical sector size may be 128, 256, 512, or 1024 bytes. Note that VERSAdos uses 128 byte sectors on FM and 256 byte sectors on MFM disks.

Number of cylinders per disk ..... { 77} for 8"
Number of cylinders per disk ..... { 80} for 5.25"

The number of cylinders per disk depends on media size and track density. Maximum values are 77 for 8 inch disks, 40 for 5.25 inch disks with 48 TPI, and 80 for 5.25 inch disks with 96 TPI.

First cylinder with compensation ..... { 44} for 8"

The number of the first cylinder with compensation depends on the type of disk drive used and must be specified accordingly. The default value of 44 is valid for Motorola's EXORdisk IV Disk Drive Unit. 5.25 inch disks do not use compensation.

Interleave factor ..... { 1} for 8" and 5.25"

For either disk size, an interleave factor may be specified. The maximum value is the number of sectors per track minus one. A value of 1 results in no interleaving.

Stepping rate in ms (3/6/10/15) ..... { 6} for 8"
Stepping rate in ms (6/12/20/30) ..... { 6} for 5.25"

The stepping rate depends on the type of disk drive used and must be specified accordingly. Available stepping rates are 3, 6, 10, and 15 milliseconds for 8 inch drives, and 6, 12, 20, and 30 milliseconds for 5.25 inch drives.

FloppyTape Parameters

For a FloppyTape, the following parameters are displayed and may be altered with the IOT command:

IDC number to be specified (0 - 7) .... { 0}

The IDC number is the logical controller number (0-7) of the MVME319 module to be initialized. Up to eight MVME319 modules may be placed in the system with their command channels located adjacently in the address map, starting at address $FF0000. The IDC number defines the command channel address used by the IOT command as the number of 512 byte increments relative to the base address $FF0000.

Drive number to be specified (0,1,4-7). { 0}4

Drive number 4 selects the FloppyTape. If a FloppyTape drive is connected, drive number 5 is not available and must not be used with any command. Drive numbers 0 and 1 are reserved for hard disk drives, drive numbers 6 and 7 for floppy disk drives.

Floppy disk or FloppyTape drive (F/T) ... {F}T for drive #4

This parameter is only displayed if drive number 4 is selected, and specifies whether drive 4 is a floppy disk or FloppyTape drive. Enter T to select a FloppyTape.

Number of sectors per segment ..... { 44}

The number of sectors per segment depends on the physical sector size specified below. Enter 44 for 256 bytes, 32 for 512 bytes, 17 for 1024 bytes per sector.

Physical sector size in bytes ..... {256}

The physical sector size may be 256, 512 or 1024. Note that VERSAdos uses 256 byte sectors on FloppyTapes.

Skip factor ..... { 2}

Skip factors from 1 to 4 may be specified. A value of 1 results in no skipping. Note that VERSAdos uses a skip factor of 2.

EXAMPLE

The examples show the default IOT parameter tables for hard disks, 8 inch floppy disks, 5.25 inch floppy disks, and FloppyTapes.

MVME101bug 3.1 > IOT (verify hard disk parameters)

IDC number to be specified (0 - 7) .... { 0}(CR)

Drive number to be specified (0,1,4-7). { 0}(CR)

SASI controller number (0 - 7) ..... { 0}(CR)

SASI controller type (0 - 2) ..... { 2}(CR)
Number of sectors per track ..... { 32}(CR)
Physical sector size in bytes ..... { 256}(CR)
Number of cylinders per disk ..... { 981}(CR)
First cylinder with compensation ..... { 490}(CR)
First cylinder with reduced current .. { 982}(CR)
Interleave factor ..... { 2}(CR)
Stepping rate code ..... { 2}(CR)
Number of heads ..... { 5}(CR)
ECC data burst length ..... { 0}(CR)

MVME101bug 3.1 > IOT (verify 8 inch floppy disk parameters)

IDC number to be specified (0 - 7) .... { 0}(CR)

Drive number to be specified (0,1,4-7). { 0}4(CR)

Floppy disk or FloppyTape drive (F/T) ... {F}(CR)

8" or 5 1/4" disk size (8/5) ..... {8}(CR)

Single or double density (S/D) ..... {S}(CR)
Single or double sided (S/D) ..... {D}(CR)
MOTOROLA or IBM format (M/I) ..... {M}(CR)
Pre-, no, or post- compensation (+/0/-) . {+}(CR)
Spiral offset (Y/N) ..... {N}(CR)
Number of sectors per track ..... { 26}(CR)
Physical sector size in bytes ..... { 128}(CR)
Number of cylinders per disk ..... { 77}(CR)
First cylinder with compensation ..... { 44}(CR)
Interleave factor ..... { 1}(CR)
Stepping rate in ms (3/6/10/15) ..... { 6}(CR)

```
MVME101bug 3.1 > IOT   (verify 5.25 inch floppy disk parameters)

IDC number to be specified (0 - 7) .... {  0}(CR)

Drive number to be specified (0,1,4-7). {  4}6(CR)

8" or 5 1/4" disk size (8/5) ..... {8}5(CR)

Single or double density (S/D) ..... {D}(CR)
Single or double sided (S/D) ..... {D}(CR)
MOTOROLA or IBM format (M/I) ..... {I}(CR)
Spiral offset (Y/N) ..... {N}(CR)
Number of sectors per track ..... { 16}(CR)
Physical sector size in bytes ..... { 256}(CR)
Number of cylinders per disk ..... {  80}(CR)
Interleave factor ..... {  1}(CR)
Stepping rate in ms (6/12/20/30) ..... {  6}(CR)

MVME101bug 3.1 > IOT   (verify FloppyTape parameters)

IDC number to be specified (0 - 7) .... {  0}(CR)

Drive number to be specified (0,1,4-7). {  6}4(CR)

Floppy disk or FloppyTape drive (F/T) ... {F}T(CR)

Number of sectors per segment ..... { 44}(CR)

Physical sector size in bytes ..... {256}(CR)

Skip factor ..... {  2}(CR)
```


3.2.20. LO - Load Memory (S-Records)

LO[;[<options>][=<text>]]

<options>	is one or both of the following options:
-C	ignores checksum while loading.
X	echoes received data to console.
<text>	is an optional string to be transmitted through Serial Port 2 before loading.

The LO command formats S-records received at Serial Port 2 in object data and stores it in the memory locations specified in the S-records. If the offset register R0 is set, it will be added to the specified address.

Each S-record data is compared with its checksum. Any mismatch results in an error message display. The checksum comparison may be suppressed with the -C option.

The S-records being loaded may be echoed on the console with the X option. Note that a printer must not be attached when this option is used, to avoid slowing down the character input from the host.

If a <text> string is specified in the command line, it will be transmitted through Serial Port 2 prior to loading S-records. It may happen that the host's character input routine is too slow to cope with the rate at which the string is output. In this case it is recommended to use the PF2 command for inserting some null characters. Also, no attempt is made to control the host transmission. Prior to loading, make sure that Serial Port 2 is configured properly for the connected device.

Any lines not beginning with an "S" character are ignored. If an error occurs, all records transmitted by the host during the error message output time will be lost.

Control is returned to MVME101bug when a S9-record is received.

After having loaded a file with an offset, R0 must be cleared using the command ".R0 0+R7" (see section 3.2.24).

EXAMPLE

A program is linked upon address $4000, but the target system memory starts at $10000. The program is position-independent and can therefore be loaded at $10000 by setting offset register R0 to $C000 (= $10000 minus $4000). The S-records transmission from the connected computer is initiated with the command "COPY FILE.MX,#".

```
MVME101bug 3.1 > .R0 C000

MVME101bug 3.1 > LO;=COPY FILE.MX,#

MVME101bug 3.1 > .R0 0+R7
```

3.2.21. MD - Memory Display/Disassembly

MD[<port>] <address>[<count>][;DI]

<port> specifies the output port number.

<address> is the beginning address of the memory section to be displayed.

<count> is the number of bytes to be displayed.

The MD command displays <count> bytes of memory beginning at <address> on the device connected with the specified port. The output device may be either one of the serial ports or the parallel port. The address must be given as word boundary (even address).

The <count> parameter may be entered in binary, octal, decimal or hexadecimal format. If no <count> is specified, it defaults to 16 bytes. For the display, the <count> value will be extended to the following multiple of 16 bytes. Once the MD command is entered, it will continue displaying the next 256 bytes each time a carriage return (CR) is entered on the console. Any other command exits MD and executes the new command.

The memory data is displayed both in hexadecimal format and its ASCII character equivalent.

COMMAND FORMAT	DESCRIPTION
MD <addr>[<count>]	Display memory data at Serial Port 1.
MD1 <addr>[<count>]	Display memory data at Serial Port 1.
MD2 <addr>[<count>]	Display memory data at Serial Port 2.
MD3 <addr>[<count>]	Display memory data at Parallel Port.

Any <port> number other than 1, 2, or 3 will be interpreted as <address>.

The MD command supports the DI option. This option will cause one-line disassembly to occur on the number of bytes specified. If no <count> is specified, the MD <address>;DI command disassembles one instruction. The one-line disassembler is described in the MM command (section 3.2.22).

EXAMPLE

```
MVME101bug 3.1 > MD 900 1F
000900   20 3C 00 00 43 43 24 19  D6 82 0C 82 7F FF FF FF   <..CC$.V.......
000910   67 02 00 00 FF FF 00 00  FF FF 00 04 FF FF 00 00  G...............

MVME101bug 3.1 > MD 900 A;DI
000900   203C00004343            MOVE.L  #17219,D0
000906   2419                    MOVE.L  (A1)+,D2
000908   D682                    ADD.L   D2,D3
```


3.2.22. MM - Memory Modify/Disassembly/Assembly

M[M] <address>[;<options>]

<address> is the memory address to be modified.

<options> is one or two of the following options:

- O Use byte-sized data, access only odd addresses.
- V Use byte-sized data, access only even addresses.
- W Use word-sized data (two bytes).
- L Use long-word-sized data (four bytes).
- N Do not read data. This option is useful for initializing write-only registers.
- DI Enter disassembler/assembler mode.

If no option is entered, byte-sized data and both even and odd addresses are used.

The O, V, W, and L options are mutually exclusive, but may be used with the N option, delimited by a semicolon. The DI option must not be used with any other option.

The M or MM command displays and modifies data in memory and memory-mapped devices. The command displays the examined address and its current contents in the specified data size, and then enters a subcommand mode. The subcommand prompt is the question mark (?). The following subcommands are valid:

[<data>](CR)	Update location and go to following address.
[<data>]^(CR)	Update location and go to previous address.
[<data>]=(CR)	Update location and redisplay same address.
[<data>].(CR)	Update location and return to MVME101bug.
<data>	is the new data to be stored in the location. If no <data> is entered, the current contents of the examined location remains unchanged.

<data> parameters may be entered in binary, octal, decimal, or hexadecimal format. A <data> parameter string of multiple values, separated by the arithmetic operators plus (+) or minus (-), is allowed. Before being stored, the <data> entered is right-justified and extended to the size specified in the MM command option field. Unless the N option is used, the stored data is verified. In case of a mismatch, the message DATA DID NOT STORE is displayed, and control is returned to MVME101bug.

The MM <address>;DI command enters the one-line disassembly/assembly mode. The command displays the examined address and the instruction it contains as hexadecimal value and in the assembler syntax. Then a subcommand mode is entered. The subcommand prompt is the question mark (?). The following subcommands are valid:

<instruction>(CR)	Assemble <instruction>, store it in current location, display instruction in hex and assembler syntax, and proceed to the address following the instruction op-code.
(CR)	Proceed to the following instruction.
.(CR)	Return to MVME101bug.

The one-line assembler accepts the assembler syntax as described in the MC68000 16-bit Microprocessor User's Manual. Labels cannot be used, i.e., branch destinations must be given as absolute addresses. The instruction mnemonic must be preceded by a space character. Invalid instructions or addressing modes are not accepted by the assembler. In such cases the address pointer is not advanced, and the assembler returns an "X" character to the console.

In addition to the MC68000 instruction set, the one-line assembler accepts the instruction:

DC.W <data>	Define Constant. <data> is 16-bit extended, right justified, and stored in the current address.

If an old instruction is overwritten by a new one that assembles into a longer op-code, the following instruction will be destroyed.

The disassembler evaluates data in several ways, depending on the context in which it is found. Addresses are displayed as hex numbers, operands are displayed as decimal numbers. For example, the source line

```
MOVE.L #$1234,5678
```

assembles into the following eight hex digits:

```
21FC00001234162E
```

The same line disassembled will show:

```
MOVE.L #4660,$0000162E
```

Note that the disassembled version differs from the original source line.

Many assemblers, including the M68000 Resident Structured Assembler, perform range checking to optimize the object code. Optimization occurs when an assembler, given a source instruction for which two forms exist, chooses the shorter form. For example, if the immediate data value given in an ADDI instruction is 7 or less, the assembler will substitute an ADDQ instruction. In such a case, the disassembled version will differ from the original source instruction. Note that the MVME101bug one-line assembler does not perform range checking and, therefore, will not cause optimization differences between original and disassembled instructions.

In some cases, instructions having different mnemonic forms assemble into identical machine code. In such cases, the disassembler always chooses the same one of two mnemonics. For example, the one-line assembler will produce the same code from the branch instructions BT (branch if condition true) and BRA (branch always). For both source mnemonics, the disassembler presents BRA when it encounters the instruction.

When the disassembler encounters an address containing a data word which is not defined as an MC68000 op-code, it will be interpreted as DC.W <data> instruction.

EXAMPLE

```
MVME101bug 3.1 > M 1000;L
00001000  00000000 ?200=
00001000  00000200 ?.

MVME101bug 3.1 > M 2004;V;N
00002002       ?55^
00002000       ?34.

MVME101bug 3.1 > M 1000;DI
001000   02000000
001000   2A7C00000C00
001006   0000
001006   2C4D
001008   0000
001008   4E75
00100A   0000

MVME101bug 3.1 > M 1000;DI
001000   2A7C00000C00
001006   2C4D
001008   4E75
```

NOTE SPACE!

```
AND.B    #0,D0 ? MOVE.L #3072,A5
MOVE.L   #3072,A5
DC.W     $0000 ? MOVE.L A5,A6
MOVE.L   A5,A6
DC.W     $0000 ? RTS
RTS
DC.W     $0000 ?.

MOVE.L   #3072,A5 ?(CR)
MOVE.L   A5,A6 ?(CR)
RTS      ?.
```

3.2.23. MS - Memory Set

MS <address> [<data>] ['<text>'] [<data>] ['<text>'] ...

<address>	is the beginning address of the memory section to be set.
<data>	is a hexadecimal number to be stored in memory
<text>	is a text string to be stored in memory.

The MS command stores the specified parameter string in memory, starting at the given <address>.

<data> parameters must be entered as hexadecimal numbers without leading dollar ($) signs. The maximum value for numbers is 8 hexadecimal digits. Numbers are 8-bit extended and right-justified before being stored.

<text> parameters are entered as ASCII strings enclosed by quotation marks.

A string of multiple parameters in any combination of <data> and <text> is allowed up to the input buffer length (100 characters).

The MS command verifies the data stored. In case of a mismatch, the message DATA DID NOT STORE is displayed, and control is returned to MVME101bug.

EXAMPLE

```
MVME101bug 3.1 > MS 2000 123 456

MVME101bug 3.1 > MD 2000
002000   01 23 04 56 00 00 00 00  00 00 00 00 00 00 00 00  .#.V............

MVME101bug 3.1 > MS 2100 'ABCDEFGH'

MVME101bug 3.1 > MD 2100
002100   41 42 43 44 45 46 47 48  00 00 00 00 00 00 00 00  ABCDEFGH........

MVME101bug 3.1 > MS 2200 A D 'THIS IS A MESSAGE' 0A0D 04

MVME101bug 3.1 > MD 2200 20
002200   0A 0D 54 48 49 53 20 49  53 20 41 20 4D 45 53 53  ..THIS IS A MESS
002210   41 47 45 0A 0D 04 00 00  00 00 00 00 00 00 00 00  AGE.............
```


3.2.24. OF - Display All Relative Offsets

OF

The OF command displays the current contents of all relative offset registers. These registers assist in debugging relocatible and position independent code.

The offset register R0 is used with the LO command for relocating object data (see section 3.2.20). After the download is finished, R0 must be reset to zero for suppressing its effect on further parameter calculations. R0 can be cleared with the command ".R0 0+R7".

Linked segments of code will each have different load addresses. For translating the physical addresses of the linked program into the logical addresses of the source listing, MVME101bug provides six general purpose offset registers (R1 through R6). These registers may be individually displayed or set with the Display/Set Register commands ".R1" - ".R6" (see section 3.2.12). The offset register R7 is not accessable and its value is always zero. This provides a convenient technique of entering an address without an offset. Also, R7 is used for clearing offset register R0 with the command ".R0 0+R7".

If one of the general purpose offset registers is set, MVME101bug displays addresses as the sum of the nearest offset and the logical address relative to that offset. For example, if a module is linked upon $4000, and R1 is set to $4000, the physical address $4008 is displayed as "000008+R1".

Once set, the general purpose offsets may be added to any address parameter in the MVME101bug commands. The offset register to be added must be entered behind the logical address, delimited by a plus sign (+). For example, if R1 is set to $4000, the command ".A0 1000+R1" will set the address register A0 to $00005000.

EXAMPLE

```
MVME101bug 3.1 > .R0 1000

MVME101bug 3.1 > LO;=COPY TEST.MX,#

MVME101bug 3.1 > .R0 0+R7

MVME101bug 3.1 > .R1 1000

MVME101bug 3.1 > .R2 1006

MVME101bug 3.1 > OF
R0=00000000 R1=00001000 R2=00001006 R3=00000000
R4=00000000 R5=00000000 R6=00000000 R7=00000000

MVME101bug 3.1 > MD 1000 A;DI
000000+R1 1018               MOVE.B (A0)+,D0
000002+R1 0C000000           CMP.B  #0,D0
000000+R2 67F8               BEQ.S  $001000
000002+R2 60FE               BRA.S  $001008
```

3.2.25. PA / NOPA - Printer Attach / Detach

PA
NOPA

The PA command attaches a printer such that all information transmitted through Serial Port 1 is copied to the Parallel Port.

The NOPA command terminates the data output at the Parallel Port.

When used as printer interface, the Peripheral Interface Adapter is initialized by MVME101bug to deliver a Centronics compatible signal at the lower rear connector P2.

Note that the PIA input/output lines are not buffered. Additional interfacing hardware must be provided to meet the electrical specifications of a Centronics printer interface. Appendix D gives an example for a connector/buffer/cable assembly.

3.2.26. PF - Port Format

PF[<port>]

<port> specifies the port to be formatted.

The PF command displays and changes the characteristics of the serial ports.

COMMAND FORMAT	DESCRIPTION
PF	Display characteristics of both serial ports. (first column SP1, second column SP2)
PF1	Display and change Serial Port 1.
PF2	Display and change Serial Port 2.

Any other <port> parameters are interpreted as not existant.

The following characteristics are displayed and may be changed by entering a new value behind the PF subcommand prompt (?):

BAUD RATE = hexadecimal contents of EPCI mode register 2
STOP BITS = hexadecimal contents of EPCI mode register 1
CHAR NULL = hexadecimal number of nulls between characters
C/R NULL = hexadecimal number of nulls after (CR) and (LF)

For initializing the EPCI mode registers refer to the MC68661 Enhanced Programmable Communications Interface data sheet in the MVME101 Monoboard Computer User's Manual.

EXAMPLE

```
MVME101bug 3.1 > PF

BAUD RATE=3E 3E
STOP BITS=4E 4E
CHAR NULL=00 00
C/R  NULL=00 00

MVME101bug 3.1 > PF2

BAUD RATE=3E?35    (300 Baud)
STOP BITS=4E?CE    (two stop bits)
CHAR NULL=00?2     (two character nulls)
C/R  NULL=00?4     (four (CR)/(LF) nulls)

MVME101bug 3.1 > PF

BAUD RATE=3E 35
STOP BITS=4E CE
CHAR NULL=00 02
C/R  NULL=00 04
```

3.2.27. TM - Transparent Mode

TM [<character>]

<character> specifies the exit character code.

The TM command provides a data link between the serial ports through the MVME101 module. Serial data received at SP1 is transmitted through SP2, and serial data received at SP2 is transmitted through SP1. In this mode, the MVME101 module appears as being transparent for the connected devices.

The transparent mode is switched off and control is returned to MVME101bug when the module receives the specified exit character. By default, the exit character is CTRL A ($01). However, any other character may be defined in the command line. For verification and as a reminder, the exit character is displayed on the console before MVME101bug enters the transparent mode.

For proper operation, the character transmission frequency of either port must not fall below the character input frequency at the other port. Before using the TM command, make sure that the baud rates and the null characters of both ports are formatted such that no data overrun will occur.

EXAMPLE

```
MVME101bug 3.1 > TM

*TRANSPARENT* EXIT=$01 = CTL A

(User corresponds directly with a computer connected with SP2. For example,
a file may be edited, assembled, and linked.)

CTRL A

MVME101bug 3.1 >
```


3.2.28. TR - Trace

T[R] [<count>]

<count> is the number of instructions to be traced.

The TR or T command executes <count> number of instructions, beginning at the program counter address. <count> may be entered in binary, octal, decimal, or hexadecimal format. If no count is specified, one instruction will be traced. After execution of each instruction, the MPU registers are displayed on the console.

Breakpoints and breakpoint counts are in effect during tracing. The instructions on breakpoint addresses are not overwritten. When any breakpoint address is encountered during tracing, the trace count is reset, and control is returned to MVME101bug. The trace mode is not left.

Once the trace mode is entered, the prompt includes a colon before the greater-than sign (:>). While in this mode, entering carriage return (CR) on the console will cause the following instruction to be traced. Entering any command (except trace commands) terminates the trace mode.

Trace cannot be used to step through interrupts or any other exceptions.

In detail, the TR command performs the following:

1. The number "0" is shown on the MVME101 STATUS display. Note that bits 5, 6, and 7 (BBTR, EBRTO, EDTTO) of the Module Control Register are unaffected.

2. The MPU registers are initialized as displayed by the DF command.

3. One instruction is executed, the MPU registers are displayed on the console, the <count> parameter is decremented by one. This is done until either a breakpoint address is encountered, or the trace count is zero. Then:

4. Control is returned to MVME101bug in the trace mode.

Note that if the program counter contains an address that falls into MVME101bug, the warning message ".PC within debugger" will be returned. Processing will continue, but with unexpected results if stack pointers and registers are not handled properly.

EXAMPLE

```
MVME101bug 3.1 > .PC 2000

MVME101bug 3.1 :> T 2
PHYSICAL ADDRESS=00002000
PC=002002 SR=2709=.S7.N..C US=FFFFFFFF SS=00000900
D0=00010434 D1=230A0444 D2=04060444 D3=00000000
D4=00010031 D5=0000072C D6=00000004 D7=00000000
A0=00FE8001 A1=FFFFFFFF A2=00000454 A3=0000054E
A4=000131E0 A5=00002704 A6=00010158 A7=00000900
--------------------002002    4E71                NOP
PC=002004 SR=2709=.S7.N..C US=FFFFFFFF SS=00000900
D0=00010434 D1=230A0444 D2=04060444 D3=00000000
D4=00010031 D5=0000072C D6=00000004 D7=00000000
A0=00FE8001 A1=FFFFFFFF A2=00000454 A3=0000054E
A4=000131E0 A5=00002704 A6=00010158 A7=00000900
--------------------002004    4E71                NOP

MVME101bug 3.1 :> (CR)
PHYSICAL ADDRESS=00002004
PC=002006 SR=2709=.S7.N..C US=FFFFFFFF SS=00000900
D0=00010434 D1=230A0444 D2=04060444 D3=00000000
D4=00010031 D5=0000072C D6=00000004 D7=00000000
A0=00FE8001 A1=FFFFFFFF A2=00000454 A3=0000054E
A4=000131E0 A5=00002704 A6=00010158 A7=00000900
--------------------002006    4E71                NOP
```


3.2.29. TT - Trace to Temporary Breakpoint

TT <address>

<address> is the temporary breakpoint address.

The TT command sets a temporary breakpoint on the specified address, and then starts tracing at the current program counter address. After execution of each instruction, the MPU registers are displayed on the console.

Previously set breakpoints and breakpoint counts are in effect during tracing. The instructions on breakpoint addresses are not overwritten. When such a breakpoint is encountered, the temporary breakpoint is reset, and control is returned to MVME101bug. The trace mode is not left.

Once the trace mode is entered, the prompt includes a colon before the greater-than sign (:>). While in this mode, entering carriage return (CR) on the console will cause the following instruction to be traced. Entering any command (except trace commands) terminates the trace mode.

Trace cannot be used to step through interrupts or any other exceptions.

In detail, the TT command performs the following:

1. The specified address of the temporary breakpoint is put into the breakpoint table.

2. The number "0" is shown on the MVME101 STATUS display. Note that bits 5, 6, and 7 (BBTR, EBRTO, EDTTO) of the Module Control Register are unaffected.

3. The MPU registers are initialized as displayed by the DF command.

4. One instruction is executed, the MPU registers are displayed on the console. This is done until any breakpoint is encountered. Then:

5. Control is returned to MVME101bug in the trace mode.

Note that if the program counter contains an address that falls into MVME101bug, the warning message ".PC within debugger" will be returned. Processing will continue, but with unexpected results if stack pointers and registers are not handled properly.

EXAMPLE

```
MVME101bug 3.1 > .PC 2000

MVME101bug 3.1 > TT 2006
PHYSICAL ADDRESS=00002006
PHYSICAL ADDRESS=00002000
PC=002002 SR=2709=.S7.N..C US=FFFFFFFF SS=00000900
D0=00010434 D1=230A0444 D2=04060444 D3=00000000
D4=00010031 D5=0000072C D6=00000004 D7=00000000
A0=00FE8001 A1=FFFFFFFF A2=00000454 A3=0000054E
A4=000131E0 A5=00002704 A6=00010158 A7=00000900
--------------------002002     4E71                NOP
PC=002004 SR=2709=.S7.N..C US=FFFFFFFF SS=00000900
D0=00010434 D1=230A0444 D2=04060444 D3=00000000
D4=00010031 D5=0000072C D6=00000004 D7=00000000
A0=00FE8001 A1=FFFFFFFF A2=00000454 A3=0000054E
A4=000131E0 A5=00002704 A6=00010158 A7=00000900
--------------------002004     4E71                NOP

AT BREAKPOINT
PC=002006 SR=2709=.S7.N..C US=FFFFFFFF SS=00000900
D0=00010434 D1=230A0444 D2=04060444 D3=00000000
D4=00010031 D5=0000072C D6=00000004 D7=00000000
A0=00FE8001 A1=FFFFFFFF A2=00000454 A3=0000054E
A4=000131E0 A5=00002704 A6=00010158 A7=00000900
--------------------002006     4E71                NOP
```


3.2.30. VE - Verify Memory (S-records)

VE[;=<text>]

<text> is an optional string to be transmitted through Serial Port 2 before verifying.

The VE command formats S-records received at Serial Port 2 in object data and compares it with the memory contents in the locations specified in the S-records.

Each S-record data is compared with its checksum. Any mismatch results in an error message display.

If a <text> string is specified in the command line, it will be transmitted through Serial Port 2 prior to verifying S-records. It may happen that the host's character input routine is too slow to cope with the rate at which the string is output. In this case it is recommended to use the PF2 command for inserting some null characters. Also, no attempt is made to control the host transmission. Prior to verifying, make sure that Serial Port 2 is configured properly for the connected device.

When any mismatch between S-record data and memory data is found, the S-record containing the differing data will be displayed.

Any lines not beginning with an "S" character are ignored.

If any error occurs, all records transmitted by the host during the error message output time will be lost.

Control is returned to MVME101bug when a S9-record is received.

EXAMPLE

Object data in memory which was downloaded from the file TEST.MX is to be verified. The S-record transmission from the connected computer is initiated with the command "COPY TEST.MX,#". A data mismatch is found at the addresses 1003 and 1005. (S1-record, 23 bytes, starting address = $1000)

```
MVME101bug 3.1 > VE;=COPY TEST.MX,#
S1231000-.-.49-.4E-.-.-.-.-.-.-.-.-.-.-.-.-.-.
```

CHAPTER 4

TRAP #15 USER I/O ROUTINES

4.1 INTRODUCTION

As an aid in program development, MVME101bug provides entry into some of its general purpose I/O routines. These routines include data conversion, serial port initialization, string input/output routines, BREAK handlers, and similar tasks.

The available MVME101bug routines are entered using the TRAP #15 instruction and defining the specific routine to be entered as a parameter in the following data word using the DC.W <data> instruction. Although the TRAP #15 instruction causes exception processing, it can be regarded in this context as a jump to a subroutine. Program execution will continue at the address following the DC.W <data> instruction, and the MPU registers (except the registers affected by the exit conditions) will not be changed. Note that during execution of a TRAP #15 routine, the MPU interrupt mask is set to 7.

Table 4.1 lists the available routines and which constant must be defined to enter them. The following paragraphs explain each routine in detail.

Table 4.1: TRAP #15 User I/O Routines

PARAMETER	SUBROUTINE DESCRIPTION
DC.W $00	Convert binary data to ASCII string
DC.W $01	Convert ASCII string to binary data
DC.W $02	Initialize both serial ports
DC.W $03	Change BREAK handler entry address
DC.W $10	Display MPU registers and return to MVME101bug
DC.W $11	Receive ASCII character from Serial Port 1
DC.W $12	Transmit ASCII character through Serial Port 1
DC.W $13	Receive ASCII string from Serial Port 1
DC.W $14	Transmit ASCII string through Serial Port 1
DC.W $15	Check Serial Port 1 for BREAK condition
DC.W $21	Receive ASCII character from Serial Port 2
DC.W $22	Transmit ASCII character through Serial Port 2
DC.W $23	Receive ASCII string from Serial Port 2
DC.W $24	Transmit ASCII string through Serial Port 2
DC.W $34	Transmit ASCII string through Parallel Port

4.2. CONVERT BINARY DATA TO ASCII STRING

Instruction: TRAP #15
DC.W $00

Description: A data byte is converted to a string of two hexadecimal ASCII characters and stored in RAM.

Entry Conditions: D0.B contains the data to be converted.
A6.L points to the address where the first character of the string has to be stored.

Exit Conditions: A6.L points to the address following the 2nd character.

4.3. CONVERT ASCII STRING TO BINARY DATA

Instruction: TRAP #15
DC.W $01

Description: An ASCII character string in memory, representing a maximum 32-bit number, is converted to binary data. The ASCII number may be specified in binary, octal, decimal, or hexadecimal format by a preceding character:

%<number> specifies a binary number,
@<number> specifies an octal number,
&<number> specifies a decimal number,
$<number> or <number> specify hexadecimal numbers.

Illegal strings are flagged.

Entry Conditions: A5.L points to the starting address of the string.
A6.L points to the address following the last character of the string.

Exit conditions: D0.L contains the converted binary data.
D1.B is the error flag register:
D1.B = 00 : No error has occured.
D1.B = 01 : ASCII string is illegal.

4.4. INITIALIZE BOTH SERIAL PORTS

Instruction: TRAP #15
DC.W $02

Description: Both serial ports are initialized using the current parameters in the initialization table. This table is set up by MVME101bug after the automatic baud rate detection, and may be changed at any time with the PF command.

Entry Conditions: None.

Exit Conditions: None.

4.5. CHANGE BREAK HANDLER ENTRY ADDRESS

Instruction: TRAP #15
DC.W $03

Description: This instruction is used to change the entry address of the BREAK handler routine. After having detected a framing error at Serial Port 1, the program transmits the "BREAK" message to the console, and then jumps to the specified address.

Entry Conditions: D0.L contains the address of the user BREAK handler. If D0.L = 0, the MVME101bug BREAK handler routine will be restored.

Exit conditions: None.

4.6. CHECK FOR BREAK CONDITION

Instruction: TRAP #15
DC.W $15

Description: Serial Port 1 is checked for the occurence of a BREAK input. If a framing error is flagged, the "BREAK" message is displayed on the console, and program control is given to the BREAK handler routine.

Entry Conditions: None.

Exit conditions: None.

4.7. RETURN TO MVME101bug

Instruction: TRAP #15
DC.W $10

Description: The formatted MPU registers are displayed on the console and program control is returned to MVME101bug. This instruction can be interpreted as being a non-removable breakpoint. The program counter points to the address following the DC.W $10 instruction.

Entry Conditions: None.

Exit conditions: None.

4.8. RECEIVE ASCII CHARACTER

Instructions:

TRAP #15 DC.W $11	Receive character from Serial Port 1
TRAP #15 DC.W $21	Receive character from Serial Port 2

Description: This instruction polls the specified serial port until an ASCII character is received, and then returns the character in the MPU data register D0.

Entry Conditions: None.

Exit conditions: D0.B contains the received character.

4.9. TRANSMIT ASCII CHARACTER

Instructions:

TRAP #15 DC.W $12	Transmit character through Serial Port 1
TRAP #15 DC.W $22	Transmit character through Serial Port 2

Description: The ASCII character contained in the MPU data register D0 is transmitted through the specified serial port.

Entry Conditions: D0.B contains the character to be transmitted.

Exit conditions: None.

4.10. RECEIVE ASCII STRING

Instructions:

TRAP #15 DC.W $13	Receive string from Serial Port 1
TRAP #15 DC.W $23	Receive string from Serial Port 2

Description: The ASCII string received from the specified serial port is stored in RAM. When a carriage return character ($D) is received, the routine returns to the calling program. Data fetch is done by polling the serial port. The received characters are echoed to the input device. The maximum string length is 128 ASCII characters.

Entry Conditions: A5.L and A6.L point to the address where the first character of the string has to be stored.

Exit conditions: A5.L has not changed.
A6.L points to the address following the last character of the string.

4.11. TRANSMIT ASCII STRING

Instructions:

TRAP #15 DC.W $14	Transmit string through Serial Port 1
TRAP #15 DC.W $24	Transmit string through Serial Port 2
TRAP #15 DC.W $34	Transmit string through Parallel Port

Description: An ASCII string contained in memory is transmitted through the specified port. The output device may be either one of the serial ports or the parallel port. If the parallel port is specified, the Peripheral Interface Adapter is initialized as Centronics compatible printer output prior to data transmission.

Entry Conditions: A5.L points to the starting address of the string.
A6.L points to the address following the last character of the string.

Exit conditions: None.

APPENDIX A

SOFTWARE ABORT

Software abort provides a convenient way to interrupt the current program and return control to MVME101bug. Pressing the ABORT pushbutton on the MVME101 front panel causes a non-maskable interrupt at the MPU (auto-vectorized interrupt level 7). The interrupt routine first saves all MPU registers, then displays the message "SOFTWARE ABORT" and the formatted MPU registers on the console, and at last enters the MVME101bug command input routine. All data in memory and I/O registers will be preserved.

Note that the software abort function will be lost if the target program changes the level 7 interrupt auto-vector.

Software abort cannot be used to regain control when the MPU hangs up within a cycle. This may happen when the Data Transfer Time-Out counter is disabled and non-existant or defect locations are addressed. In such cases the RESET pushbutton must be used to return to MVME101bug. By that the MVME101 module will be reinitialized as described in section 2.5, and all user data in the vector table and in the I/O registers will be lost.

APPENDIX B

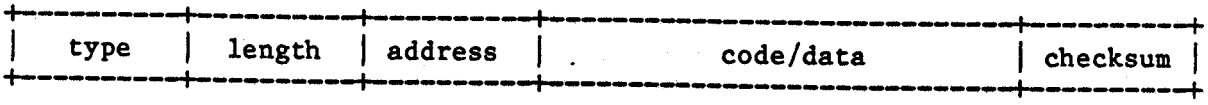
S-RECORD FORMAT

The S-record format for output modules was devised for the purpose of encoding programs or data files in a printable format for transportation between computer systems. The transportation process can thus be visually monitored and the S-records can be more easily edited.

1. S-RECORD CONTENT

When viewed by the user, S-records are essentially character strings made of several fields which identify record type, record length, memory address, code/data, and checksum. Each byte of binary data is encoded as a 2-character hexadecimal number: the first character represents the high-order 4 bits and the second character the low-order 4 bits of the byte.

The five fields which comprise an S-record are shown below:



FIELD	CHARACTERS	CONTENTS
type	2	S-record type -- S0, S1, etc.
length	2	The count of character pairs in the record excluding the type and record length.
address	4,6, or 8	The 2, 3, or 4 byte address at which the data field is to be loaded into memory.
code/data	0-2n	From 0 to n bytes of executable code, memory loadable data, or descriptive information. For compatibility with teletypewriters, some programs may limit the number of bytes to as few as 28 (56 printable characters in the S-records).
checksum	2	The least significant byte of the one's complement of the sum of the values represented by the pairs of characters making up record length, address, and code/data fields.

Each record may be terminated with a CR/LF/NULL. Additionally, an S-record may have an initial field to accommodate other data such as line numbers generated by some time-sharing systems.

Accuracy of transmission is ensured by the record length (byte count) and checksum fields.

2. S-RECORD TYPES

Eight types of S-records have been defined to accommodate the several needs of encoding, transportation, and decoding functions. The various Motorola upload, download, and other record transportation control programs, as well as cross assemblers, linkers, and other file creating or debugging programs, utilize only those S-records which serve the purpose of the program. For specific information on which S-records are supported by a particular program, the user's manual for that program must be consulted.

An S-record-format module may contain S-records of the following types:

S0 The header record for each block of S-records. The code/data field may contain any descriptive information identifying the following block of S-records. Under VERSAdos, the resident linker's IDENT command can be used to designate module name, version number, revision number, and description information which will make up the header record. The address field is normally zeroes.

S1 A record containing code/data and the 2-byte address at which the code/data is to reside.

S2 A record containing code/data and the 3-byte address at which the code/data is to reside.

S3 A record containing code/data and the 4-byte address at which the code/data is to reside.

S5 A record containing the number of S1-, S2-, and S3-records transmitted in a particular block. This count appears in the address field. There is no code/data field.

S7 A termination record for a block of S3-records. The address field may optionally contain the 4-byte address of the instruction to which control is to be passed. There is no code/data field.

S8 A termination record for a block of S2-records. The address field may optionally contain the 3-byte address of the instruction to which control is to be passed. There is no code/data field.

S9 A termination record for a block of S1-records. The address field may optionally contain the 2-byte address of the instruction to which control is to be passed. Under VERSAdos, the resident linker's ENTRY command can be used to specify this address. If not specified, the first entry point specification encountered in the object module input will be used. There is no code/data field.

Only one termination record is used for each block of S-records. S7- and S8-records are usually used only when control is to be passed to a 3- or 4-byte address. Normally, only one header record is used, although it is possible for multiple header records to occur.

3. CREATION OF S-RECORDS

S-record-format programs may be produced by several dump utilities, debuggers, VERSAdos' resident linkage editor, or several cross assemblers or cross linkers. In VERSAdos, the Build Load Module (MBLM) utility allows an executable load module to be build from S-records, and has a counterpart utility in Build S-Record (BUILDS), which allows an S-record file to be created from a load module.

Several programs are available for downloading a file in S-record format from a host system to an 8-bit or 16-bit microprocessor-based system. Programs are also available for uploading an S-record file to or from a VERSAdos system.

4. EXAMPLE

Shown below is a typical S-record-format module, as printed or displayed:

```
S006000048445218
S1130000285F245F2212226A000424290008237C2A
S113001000020008000826290018538123410018 13
S11300204 1E900084E42234300182342000824A952
S107003000144ED492
S9030000FC
```

The module consists of one S0-record, four S1-records and an S9-record.

The S0-record is comprised of the following character pairs:

S0 S-record type S0, indicating that it is a header record.

06 Hexadecimal 06 (decimal 6), indicating that six character pairs (or ASCII bytes) will follow.

0000 Four-character 2-byte address field, zeroes in this example.

48 44 52 ASCII H, D, and R - "HDR".

1B The checksum.

The first S1-record is explained as follows:

S1 S-record type S1, indicating that it is a code/data record to be loaded/verified at a 2-byte address.

13 Hexadecimal 13 (decimal 19), indicating that 19 character pairs, representing 19 bytes of binary data, follow.

0000 Four-character 2-byte address field; hexadecimal address 0000, where the data which follows is to be loaded.

The next 16 character pairs of the S1-record are the ASCII bytes of the actual program code/data. In this assembly language example, the hexadecimal opcodes of the program are written in sequence in the code/data fields of the S1-records:

OPCODE	INSTRUCTION
285F	MOVE.L (A7)+,A4
245F	MOVE.L (A7)+,A2
2212	MOVE.L (A2)+,D1
226A0004	MOVE.L 4(A2),A1
24290008	MOVE.L FUNCTION(A1),D2
237C	MOVE.L #FORCEFUNC,FUNCTION(A1)

(The balance of this code is continued in the code/data fields of the remaining S1-records, and stored in the memory location 0010, etc.)

2A The checksum of the first S1-record.

The second and third S1-records each also contain $13 (19) character pairs and are ended with checksums 13 and 52, respectively. The fourth S1-record contains 07 character pairs and has a checksum of 92.

The S9-record is explained as follows:

S9 S-record type S9, indicating that it is a termination record.

03 Hexadecimal 03, indicating that three character pairs (3bytes) will follow.

0000 The address field, zeroes.

FC The checksum of the S9 record.

Each printable character in a S-record is encoded in hexadecimal (ASCII in this example) representation of the binary bits which are actually transmitted. For example, the first S1 record above is sent as:

type	length	address	code/data	chksm
S 1	1 3	0 0 0 0	2 8 5 F 2 4	2 A
$53 $31	$31 $33	$30 $30 $30 $30	$32 $38 $35 $46 $32 $34	$32 $41

APPENDIX C

MVME101bug MESSAGES

MESSAGE	MEANING
ACFAIL LOW	VMEbus signal ACFAIL* asserted
ADER EXCEPTION	Address error detected
AT BREAKPOINT	Breakpoint encountered
AV#1 EXCEPTION : AV#6 EXCEPTION	Auto-vectorized interrupt level 1 : Auto-vectorized interrupt level 6
BD COMPLETE	Bootstrap dump successfully completed
BERR EXCEPTION	VMEbus signal BERR* asserted
BREAK	Framing error at SP1 detected
BRTO EXCEPTION	Bus request time-out occured
CHCK EXCEPTION	CHK instruction encountered
CHKSUM=XX	XX is S-record checksum
DATA DID NOT STORE	Memory data verification failed
DIV0 EXCEPTION	Division by zero
DTTO EXCEPTION	Data transfer time-out occured
DUMP FILE NOT FOUND	Specified disk does not contain dump file
DUMP.SY TOO SMALL	Dump file is smaller than system memory
ERROR	Error (Prefix)
FAILED AT.. WROTE=.. READ=..	Data verification during BT command failed
ILLEGAL INSTRUCTION	Instruction used an illegal op-code
INVALID ADDRESS	Data cannot be stored on specified address
INVALID TRAP #15 XXXX ERROR	XXXX is invalid TRAP #15 parameter
IS NOT HEX DIGIT	Invalid character in command line
MVME101bug 3.1 >	MVME101bug prompt
NO OFF-BOARD MEMORY	No memory found off-board
NOT HEX	Invalid character in command line

MESSAGE	MEANING
PHYSICAL ADDRESS	Physical address used by command
PRINTER NOT READY	Printer ready signal negated
PRIV EXCEPTION	Privileged instruction in user state
"R" ZAPPED	Disk read operation successfully completed
SOFTWARE ABORT	ABORT pushbutton pressed
SPUR EXCEPTION	Bus error during interrupt acknowledge
SYNTAX ERROR	Error in command line
TP V EXCEPTION	TRAPV instruction encountered
TRAC EXCEPTION	Trace bit in status register set
TRANSPARENT EXIT=$01=CTL A	Transparent mode, exit character
UT0 EXCEPTION	TRAP #$0 instruction encountered
: :	: : :
UTE EXCEPTION	TRAP #$E instruction encountered
WHAT	User's entry is not recognized
"W" ZAPPED	Disk write operation successfully completed
1010 EXCEPTION	Unimplemented instruction encountered
1111 EXCEPTION	Unimplemented instruction encountered
???? EXCEPTION	Unitialized or unassigned exception

APPENDIX D

CENTRONICS PRINTER INTERFACE

Figure D.1: Centronics Printer Interface Schematic Diagram

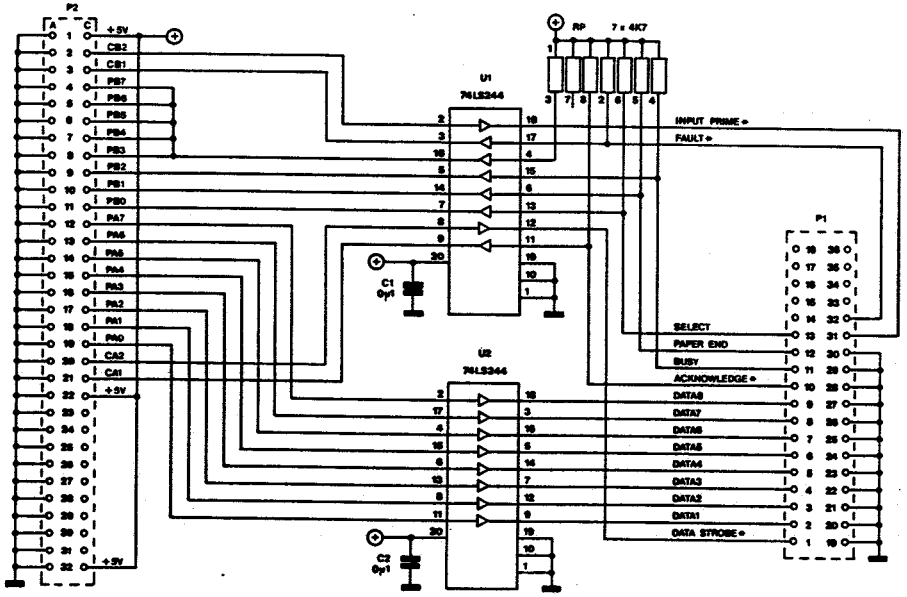
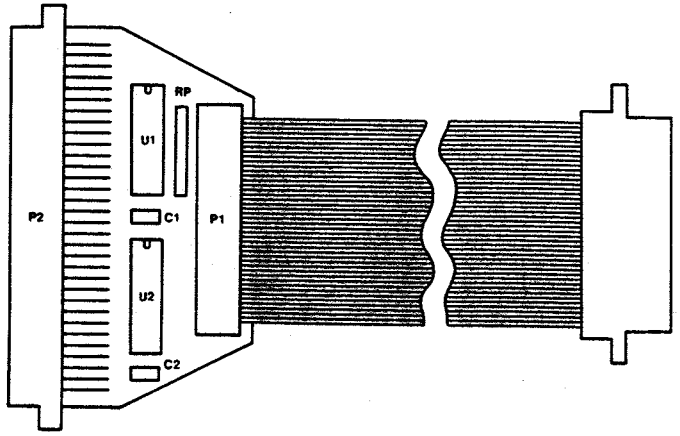


Figure D.2: Centronics Printer Interface Assembly



2.5. MODULE STATUS REGISTER

Through the Module Status Register (MSR) the current status of several on-board signals and VMEbus lines can be monitored. By that the MPU can detect certain system conditions and branch to the appropriate servicing routines.

The MSR appears as an 8-bit register in the on-board I/O-devices address segment. Paragraph 2.7 gives more detailed addressing information.

Figure 2.3 shows how the MSR is interconnected with VMEbus signals and with other functional blocks on the MVME101. During a read operation, the outputs of the MSR are enabled and put on the lower order data lines D00-D07. The outputs MSR0-MSR5 represent the current states of the signals ACFAIL*, SYSFAIL*, ABORT*, BCLR*, BAV* and PCI1RXD*. MSR6 and MSR7 are Flip-Flop outputs which are set to 0 when a bus request time-out (MSR6) or a data transfer time-out (MSR7) occured. Any write operation to the MSR clears MSR6 and MSR7 to 1, regardless of the data transferred.

All signals represented in the MSR are active low. A bit value of 0 indicates that the corresponding signal is asserted, a value of 1 means that it is negated.

Figure 2.3: Module Status Register

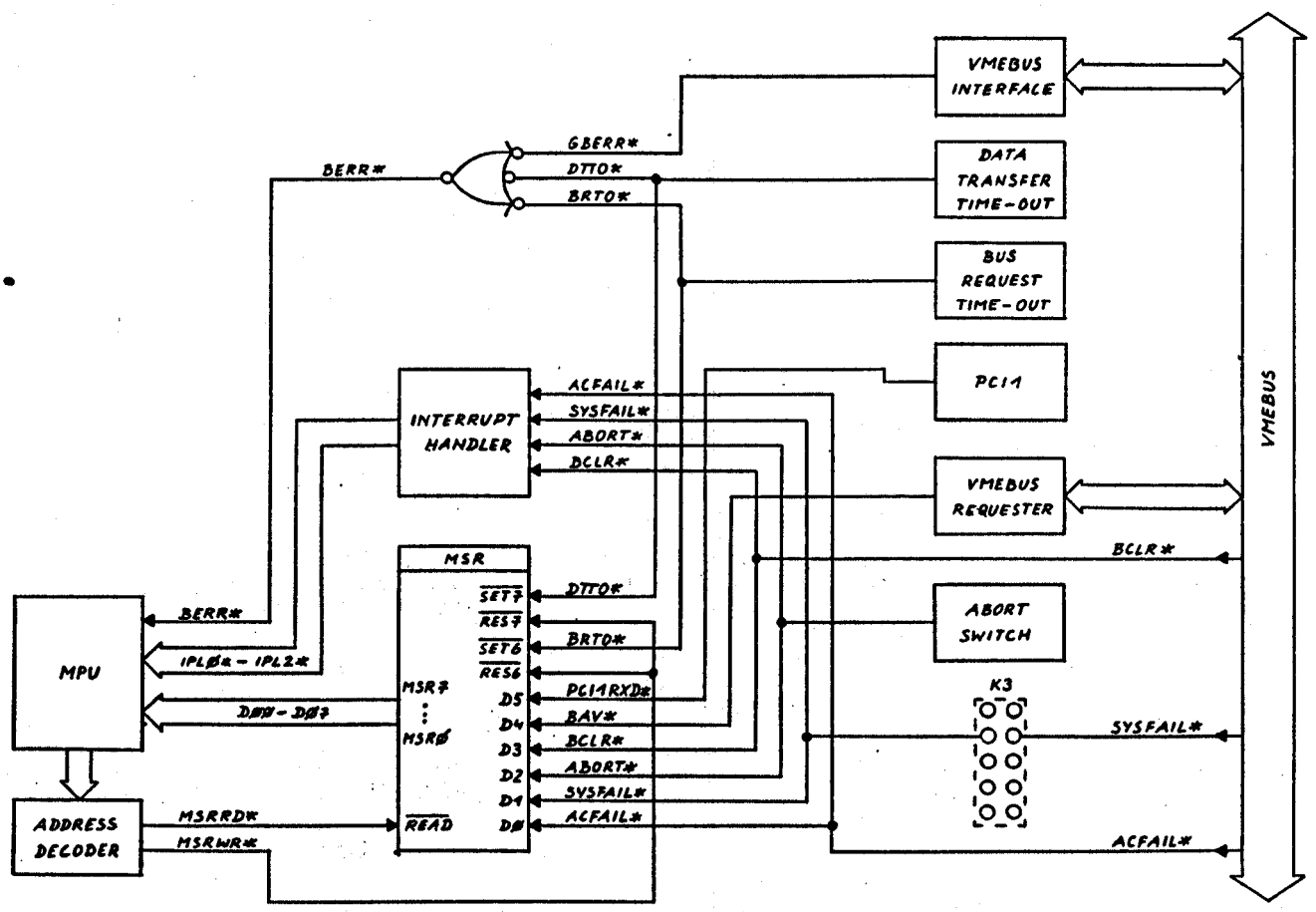


Table 2.3 shows the allocation of signals in the MSR and explaines the information contained in each bit.

Table 2.3: Module Status Register

BIT	SIGNAL	DESCRIPTION
MSR7	DTTO*	MSR7 = 0: A Data Transfer Time-Out occured. MSR7 = 1: A Data Transfer Time-Out did not occur. Note: Paragraph 2.9.4 describes the Data Transfer Time Out counter in detail.
MSR6	BRTO*	MSR6 = 0: A Bus Request Time-Out occured. MSR6 = 1: A Bus Request Time-Out did not occur. Note: Paragraph 2.9.4 describes the Bus Request Time Out counter in detail.
MSR5	PCI1RXD*	MSR5 reflects the current state of the data input of Serial Port 1. Note: Paragraph 2.4.2.5 describes how MSR5 can be used for automatic baud rate detection.
MSR4	BAV*	MSR4 = 0: The VMEbus is available. MSR4 = 1: The VMEbus is not available. Note: Paragraph 2.8.2 describes how the BAV* signal is used for bus arbitration.
MSR3	BCLR*	MSR3 = 0: The VMEbus signal BCLR* is asserted. MSR3 = 1: The VMEbus signal BCLR* is negated. Note: Paragraph 2.8.2 describes how the BCLR* signal is used for bus arbitration.
MSR2	ABORT*	MSR2 = 0: The ABORT switch is pressed. MSR2 = 1: The ABORT switch is released. Note: Paragraph 2.11.1 describes the ABORT function.
MSR1	SYSFAIL*	MSR1 = 0: The VMEbus signal SYSFAIL* is asserted. MSR1 = 1: The VMEbus signal SYSFAIL* is negated. Note: Paragraph 2.11.2 describes the SYSFAIL function.
MSR0	ACFAIL*	MSR0 = 0: The VMEbus signal ACFAIL* is asserted. MSR0 = 1: The VMEbus signal ACFAIL* is negated. Note: Paragraph 2.11.1 describes the ACFAIL function.

2.6. MODULE CONTROL REGISTER

The Module Control Register (MCR) contains eight bits for controlling various module functions and the hexadecimal STATUS display. To support single bit manipulations, the data byte in the MCR can be both written and read.

The MCR appears as an 8-bit register in the on-board I/O-devices address segment. Paragraph 2.7 gives more detailed addressing information.

Figure 2.4 shows how the MCR is interconnected with other functional blocks on the MVME101. During a write operation, the bit pattern on the lower order data lines D00-D07 is stored in the MCR. The four bits MCR0-MCR3 represent the hex number to be shown on the STATUS display in binary data format. In addition, when MCR0-MCR3 all are set to 1, i.e. when the hex number F is displayed, the VMEbus signal SYSFAIL* is asserted. MCR4 is used to switch the display on and off. MCR5 controls the bus block transfer mode of the VMEbus Requester. The bits MCR6 and MCR7 are used to enable or disable the time-out counters.

After a system reset all bits in the MCR are cleared to 0. Also, when the MPU has halted due to a double bus error, the MCR is cleared, and both decimal points on the STATUS display are lit.

All signals controlled by the MCR are active high. A bit value of 1 causes the assertion of the corresponding signal, a value of 0 causes its negation.

Figure 2.4: Module Control Register

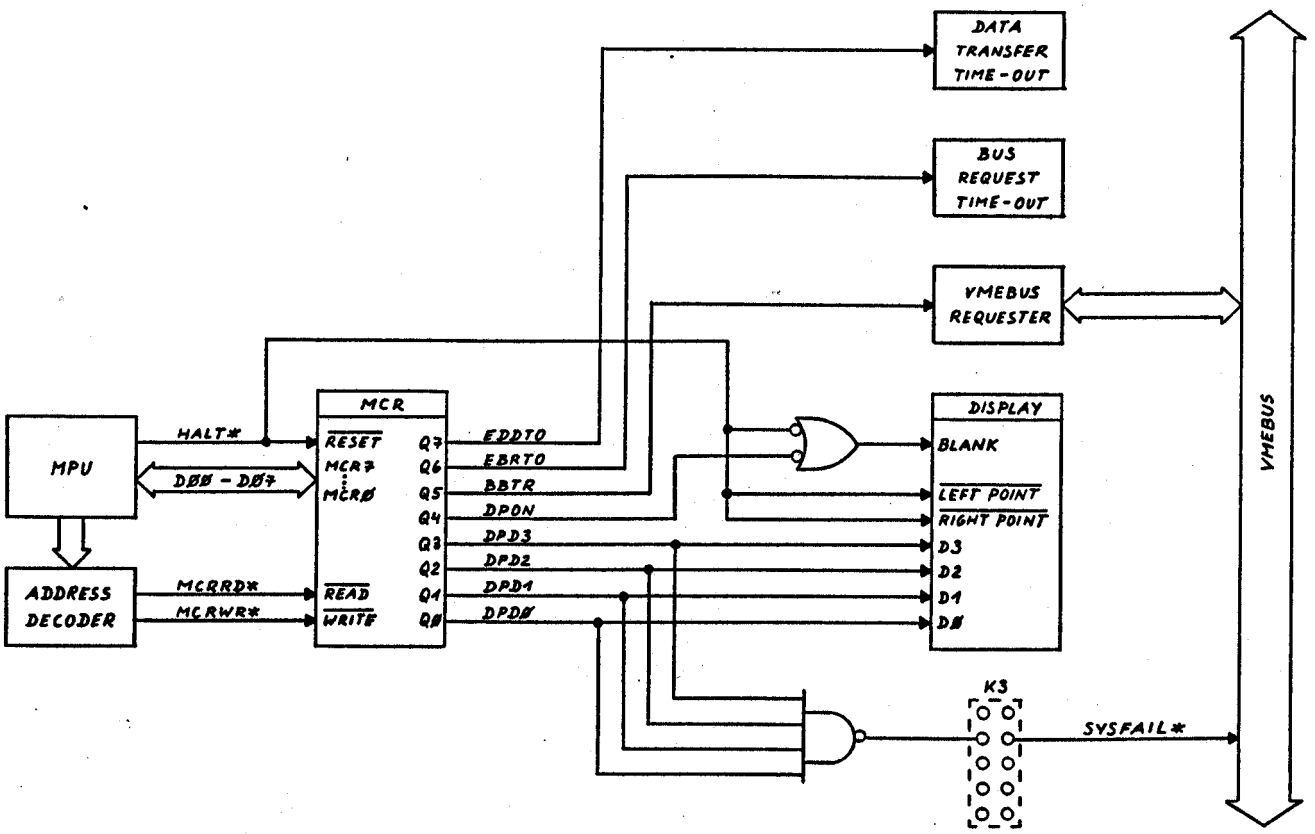


Table 2.4 shows the allocation of signals in the MCR and explaines the function of each bit.

Table 2.4: Module Control Register

BIT	SIGNAL	DESCRIPTION
MCR7	EDDTO	MCR7 = 0: Disable Data Transfer Time-Out counter. MCR7 = 1: Enable Data Transfer Time-Out counter. Note: Paragraph 2.9.4 describes the Data Transfer Time-Out counter in detail.
MCR6	EBRTO	MCR6 = 0: Disable Bus Request Time-Out counter. MCR6 = 1: Enable Bus Request Time-Out counter. Note: Paragraph 2.9.4 describes the Bus Request Time-Out counter in detail.
MCR5	BBTR	MCR5 = 0: Negate Bus Block Transfer Request. MCR5 = 1: Assert Bus Block Transfer Request. Note: Paragraph 2.8.2 describes the function of the BBTR signal.
MCR4	SDON	MCR4 = 0: Blank STATUS Display. MCR4 = 1: Lit STATUS Display. Note: The STATUS Display is also blanked after system reset and when the MPU has halted.
MCR3 MCR2 MCR1 MCR0	SDD3 SDD2 SDD1 SDD0	SDD3,SDD2,SDD1,SDD0 = 0,0,0,0: Display "0" SDD3,SDD2,SDD1,SDD0 = 0,0,0,1: Display "1" : : : : SDD3,SDD2,SDD1,SDD0 = 1,1,1,0: Display "E" SDD3,SDD2,SDD1,SDD0 = 1,1,1,1: Display "F" and assert SYSFAIL* The bits SDD0-SDD3 are the binary equivalent of the hexadecimal number on the STATUS display. Also, these bits are used to assert the SYSFAIL* signal on the VMEbus by setting them all to 1, i.e. by writing "F" into the STATUS display. Note: Paragraph 2.11.2 describes the SYSFAIL function.

Table 2.5: Original Address Map

ADDRESS	CONTENTS	SELECTED DEVICES
FFFFFF : FF0000	VMEbus Short I/O Addresses	Global I/O-devices
FEFFFF : FE1000	VMEbus Standard Addresses	Global Memory or Memory-mapped Devices
FE0FFF : FE0000	On-board I/O Registers (Only odd addresses used)	Local I/O-devices
FDFFFF : F04000	VMEbus Standard Addresses	Global Memory or Memory-mapped Devices
F03FFF : F00000	MVME101bug Debug Package or User-provided Program	2 x 8K bytes Local ROM in Memory Socket Pair 4
EFFFFF : : : : : : : : : : : 003000	VMEbus Standard Addresses	Global Memory or Memory-mapped Devices
002FFF : 002000	User Program/Data	2 x 2K bytes Local RAM in Memory Socket Pair 3
001FFF : 001000	User Program/Data	2 x 2K bytes Local RAM in Memory Socket Pair 2
000FFF : 000800	User Program/Data	2 x 2K bytes Local RAM in Memory Socket Pair 1
0007FF : 000400	MVME101bug Data/Stack	
0003FF : 000000	MPU Exception Vectors	

Table 2.6: Original I/O-Register Address Map

DEVICE	ADDRESS	MODE	REGISTER
MCR	FE00F1	r/w	Module Control Register
MSR	FE00E1	r/w	Module Status Register
PTM	FE00DF	read	LSB buffer register
	FE00DF	write	Timer #3 latches
	FE00DD	read	Timer #3 counter
	FE00DD	write	MSB buffer register
	FE00DB	read	LSB buffer register
	FE00DB	write	Timer #2 latches
	FE00D9	read	Timer #2 counter
	FE00D9	write	MSB buffer register
	FE00D7	read	LSB buffer register
	FE00D7	write	Timer #1 latches
	FE00D5	read	Timer #1 counter
	FE00D5	write	MSB buffer register
	FE00D3	read	status register
	FE00D3	write	control register #2
	FE00D1	read	no operation
	FE00D1	write	CR20 = 1: control register #1
	FE00D1	write	CR20 = 0: control register #3
PIA	FE00C7	r/w	Section B control register
	FE00C5	r/w	CRB-2 = 1: Section B peripheral register
	FE00C5	r/w	CRB-2 = 0: Section B data direction register
	FE00C3	r/w	Section A control register
	FE00C1	r/w	CRA-2 = 1: Section A peripheral register
	FE00C1	r/w	CRA-2 = 0: Section A data direction register
PCI2	FE00B7	r/w	command register
	FE00B5	r/w	mode register #1 / mode register #2
	FE00B3	read	status register
	FE00B3	write	SYN1 register / SYN2 register / DLE register
	FE00B1	read	receive holding register
	FE00B1	write	transmit holding register
PCI1	FE00A7	r/w	command register
	FE00A5	r/w	mode register #1 / mode register #2
	FE00A3	read	status register
	FE00A3	write	SYN1 register / SYN2 register / DLE register
	FE00A1	read	receive holding register
	FE00A1	write	transmit holding register

ON MACHINE I/O CARD 8

PTM @ FF00Lx
PIA @ FF00Cx
PCI2 @ FF00By
PCI1 @ FF00Ax

SO MONITOR CHANGED TO FF I/O Base

POSEC (ALL CARDS) @ FFxxxx

Figure 4.2: Schematic Diagram Sheet 1/11

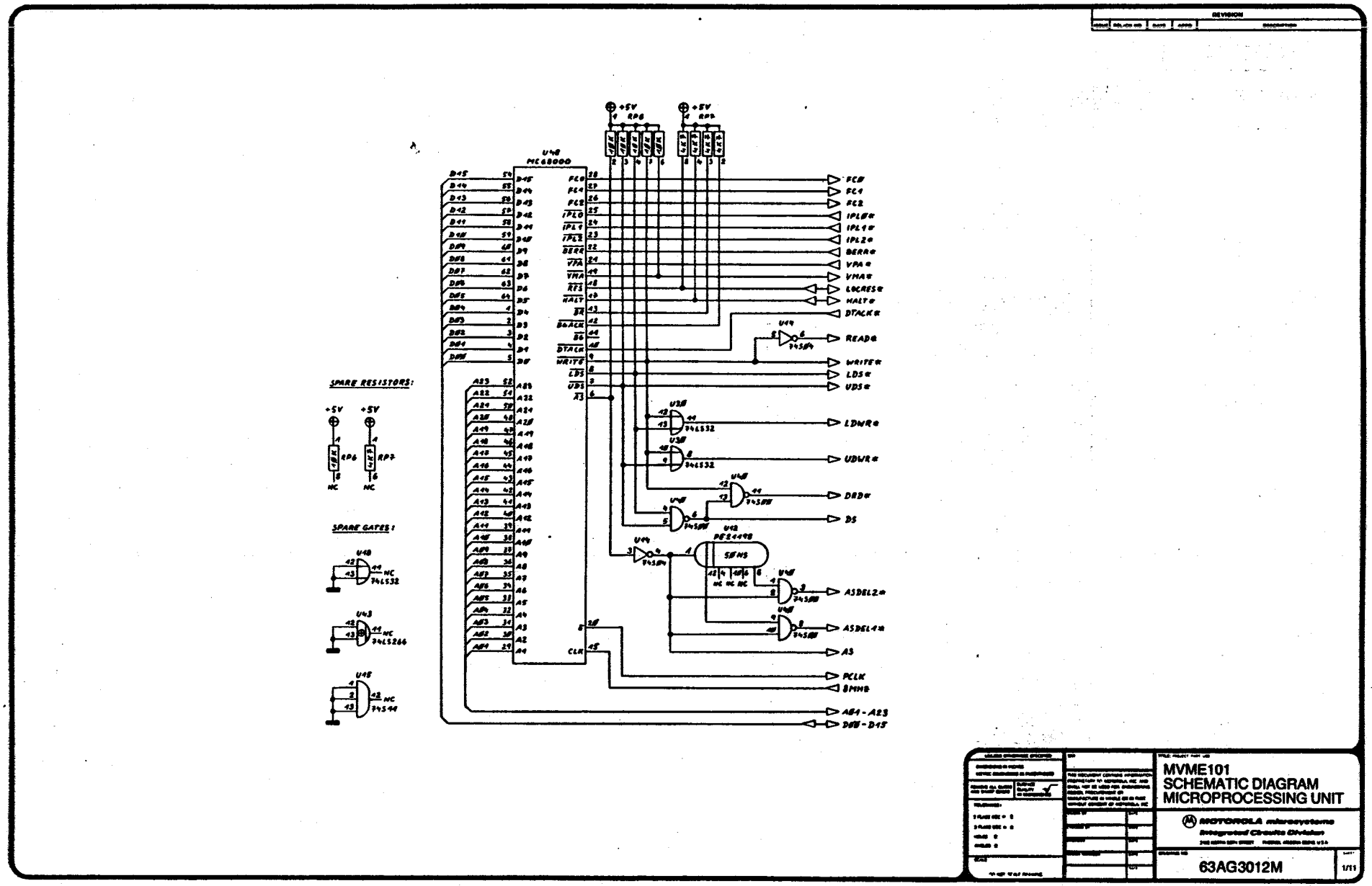


Figure 4.3: Schematic Diagram Sheet 2/11

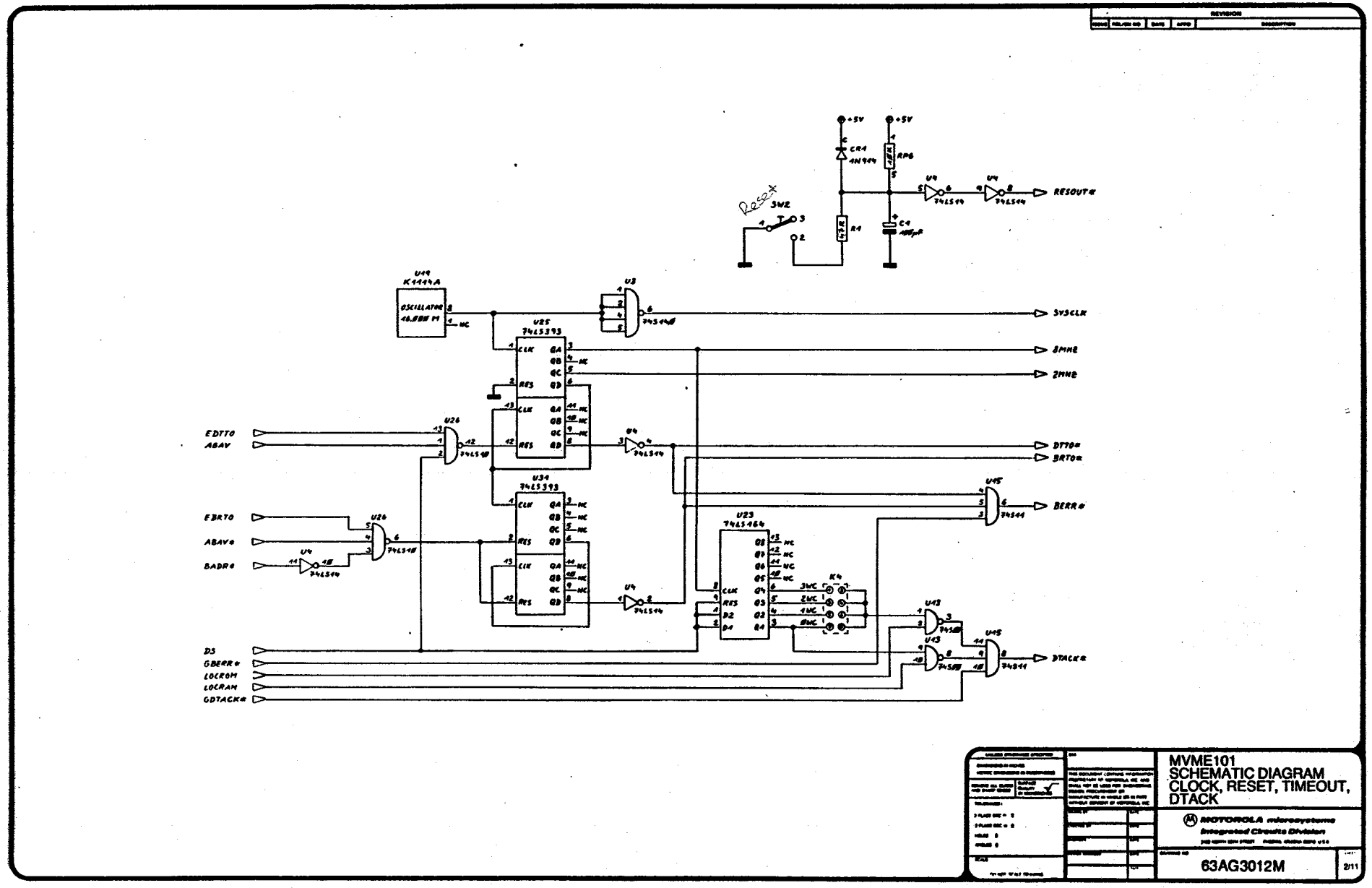


Figure 4-4. Schematic Diagram Sheet 3/11

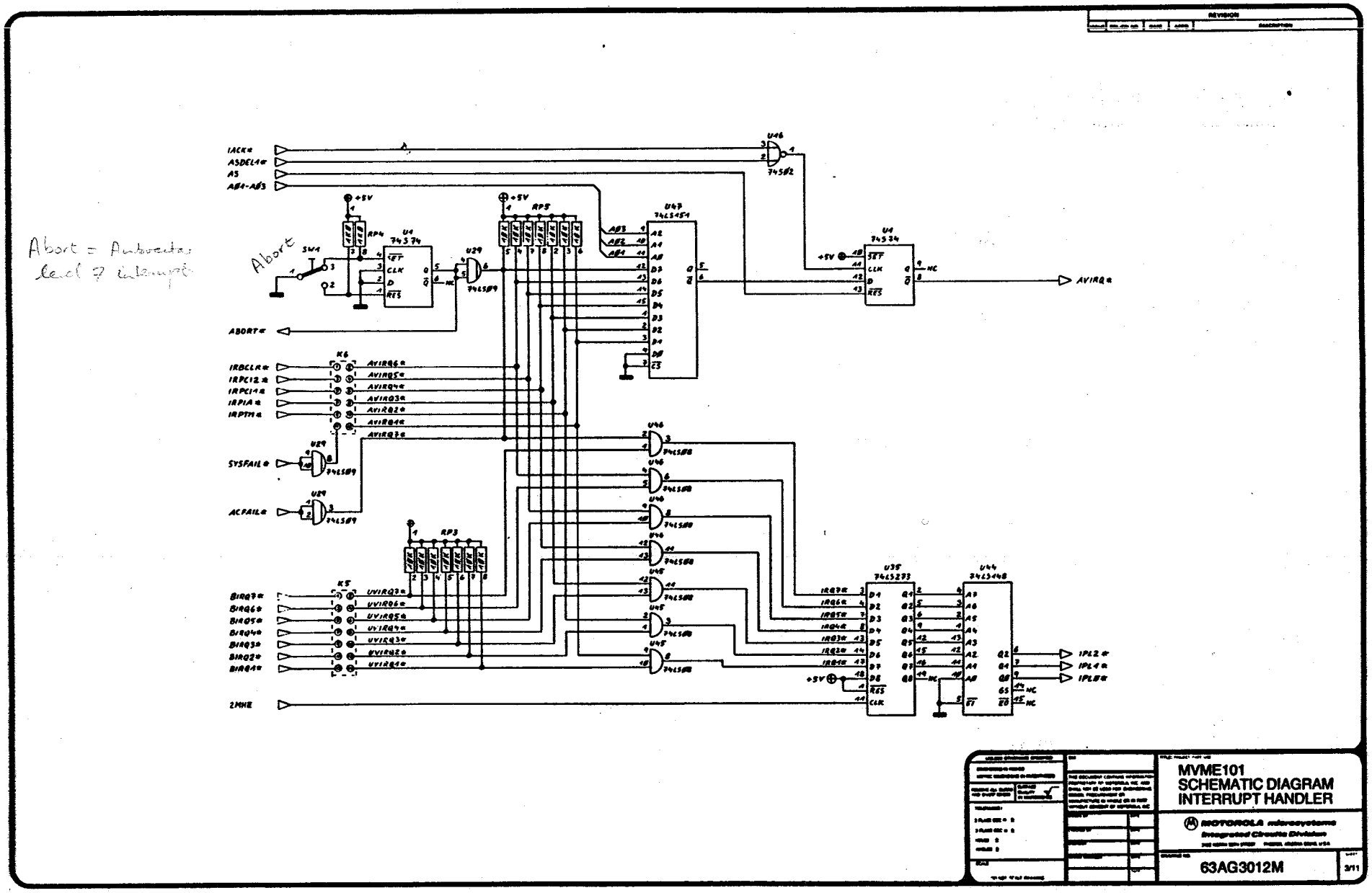


Figure 4.5: Schematic Diagram Sheet 4/11

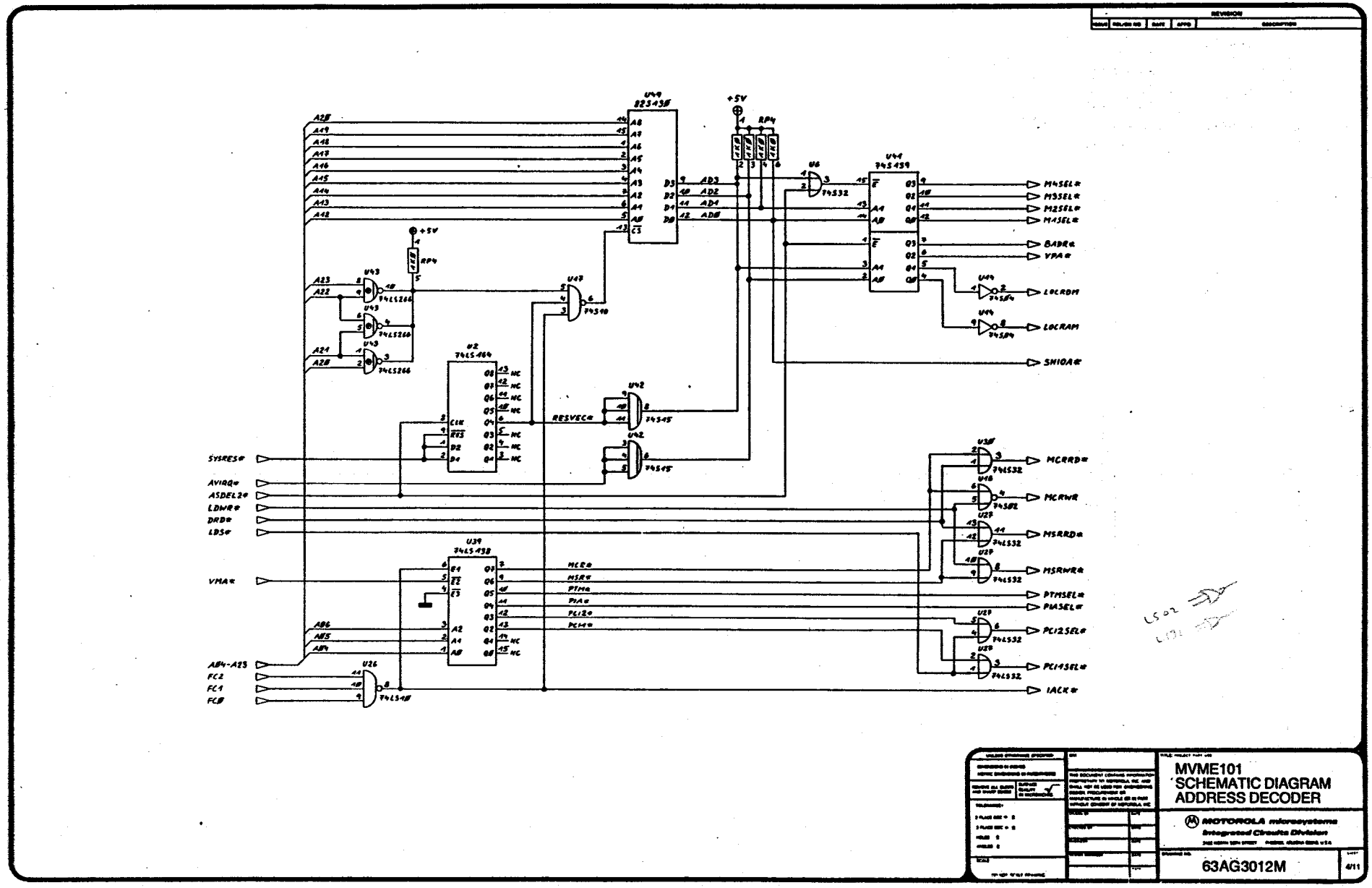


Figure 4.6: Schematic Diagram Sheet 5/11

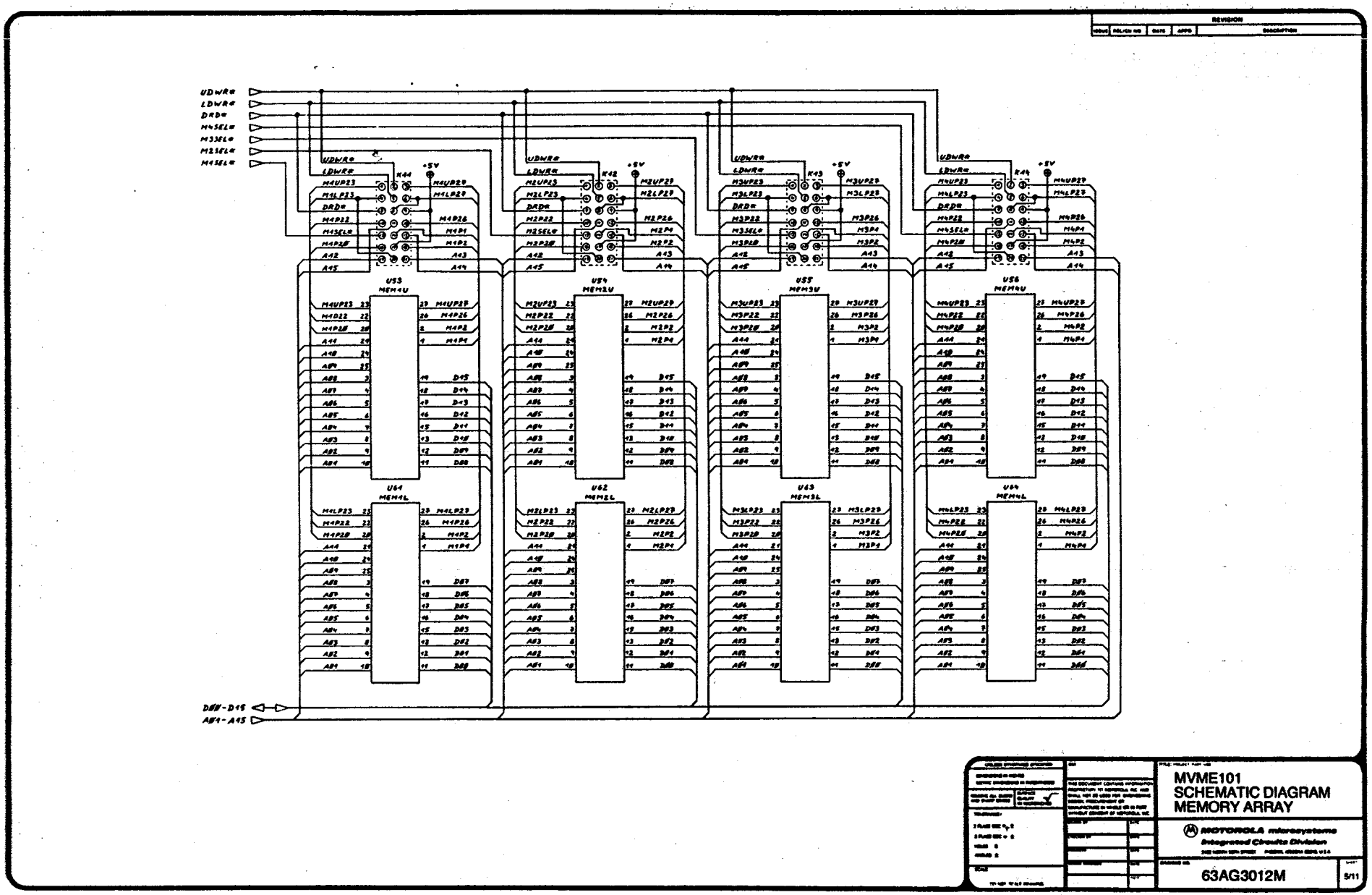


Figure 4.7: Schematic Diagram Sheet 6/11

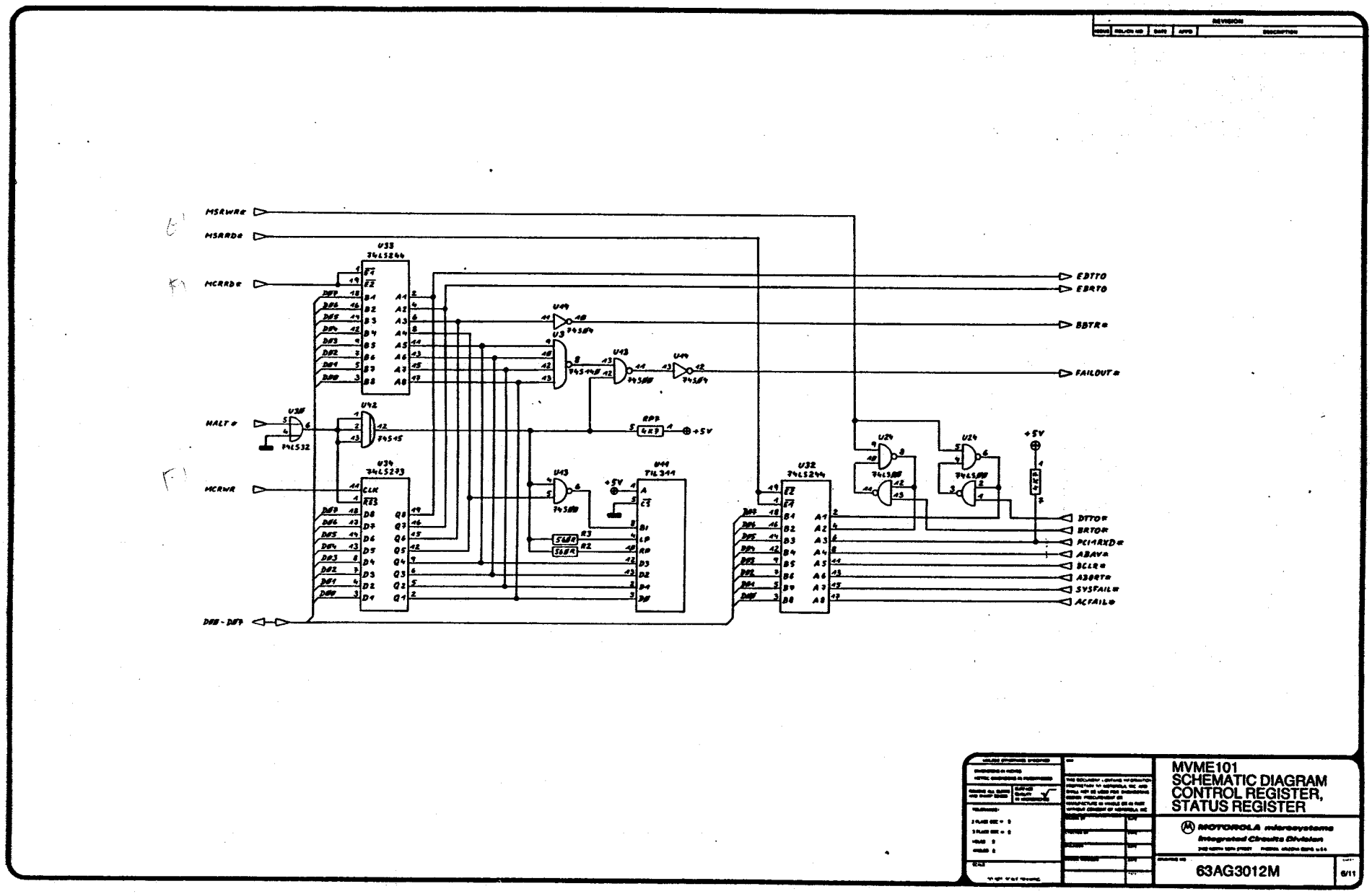


Figure 4.8: Schematic Diagram Sheet 7/11

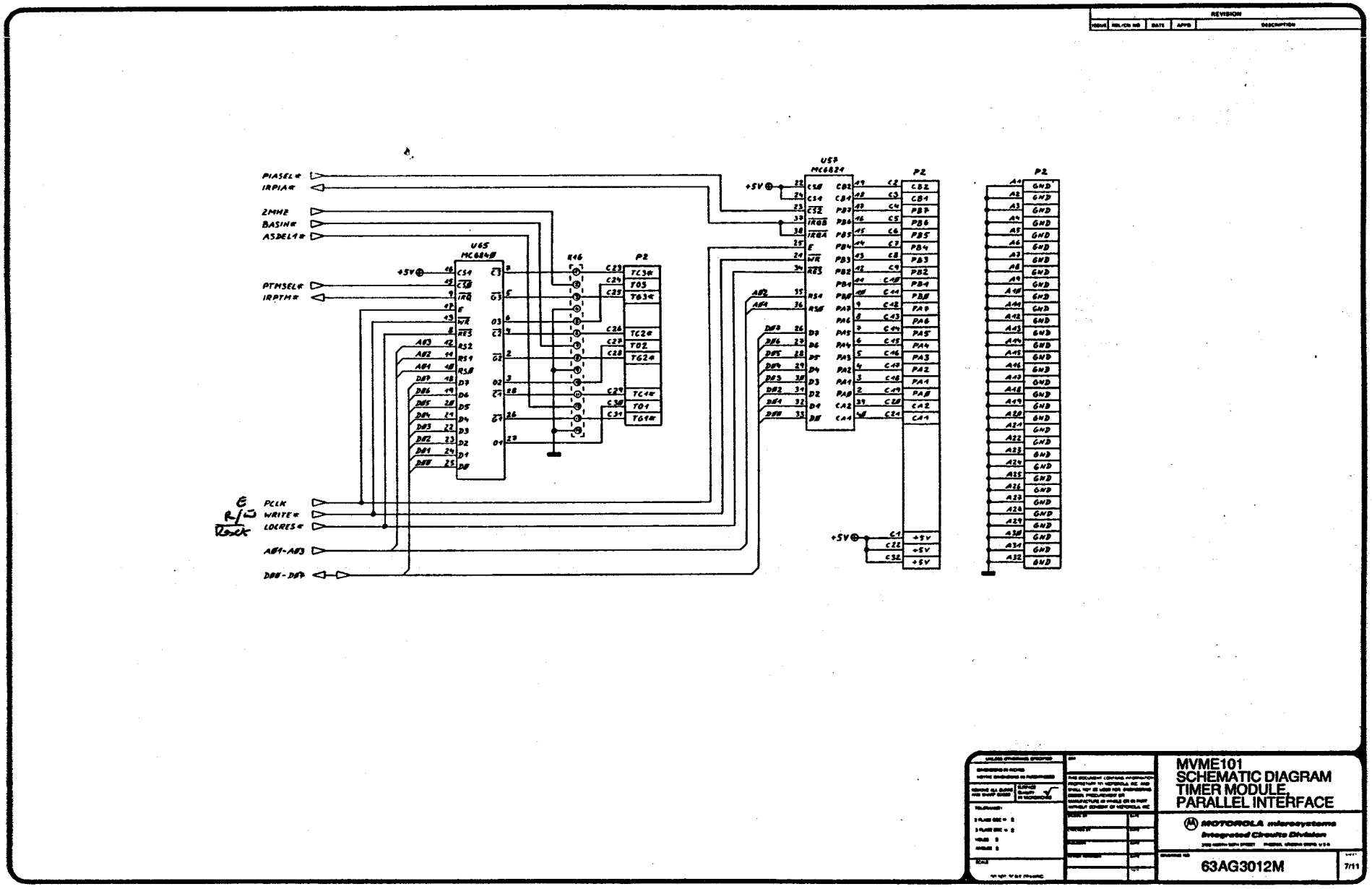


Figure 4.9: Schematic Diagram Sheet 8/11

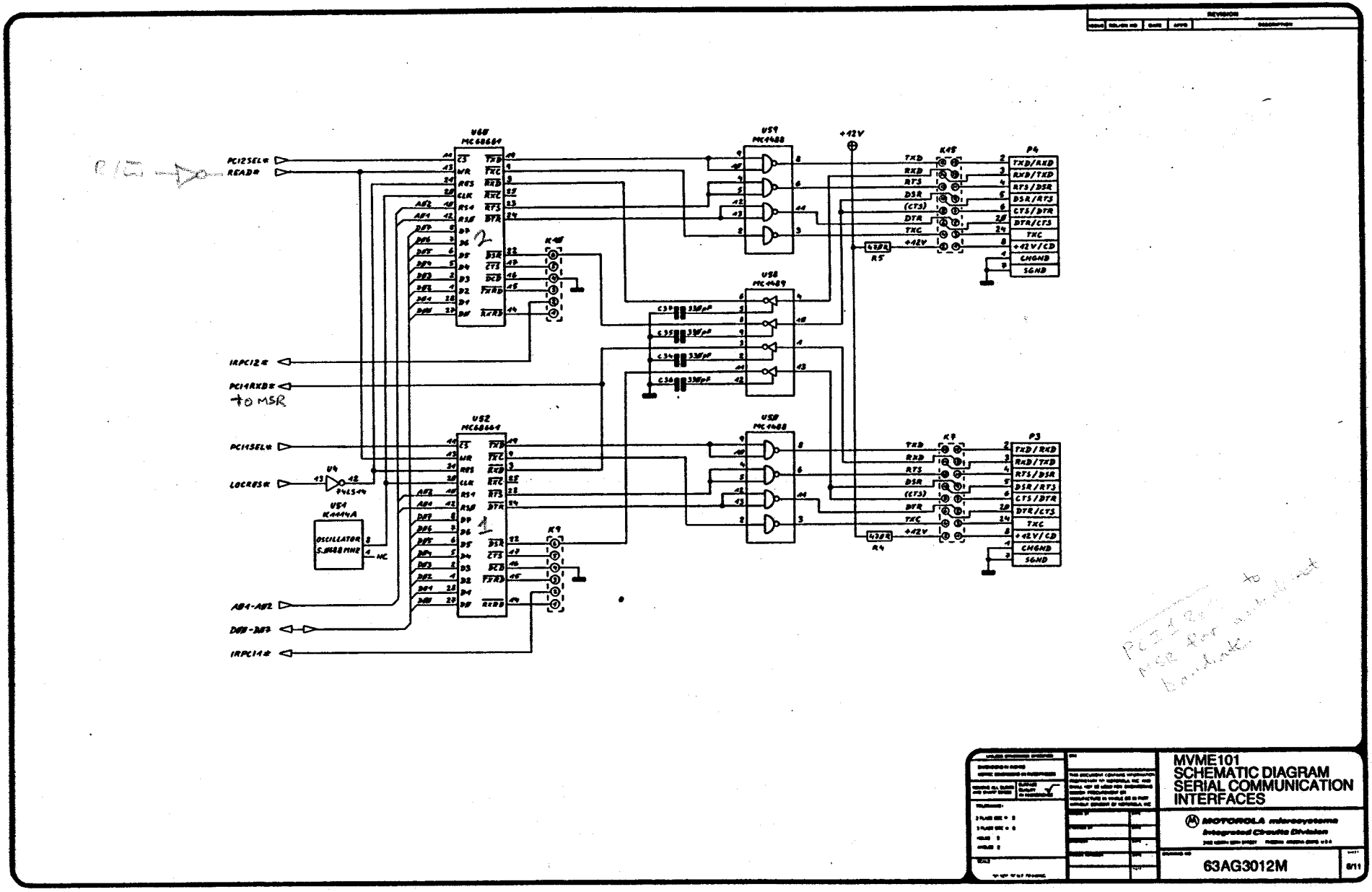


Figure 4.10: Schematic Diagram Sheet 9/11

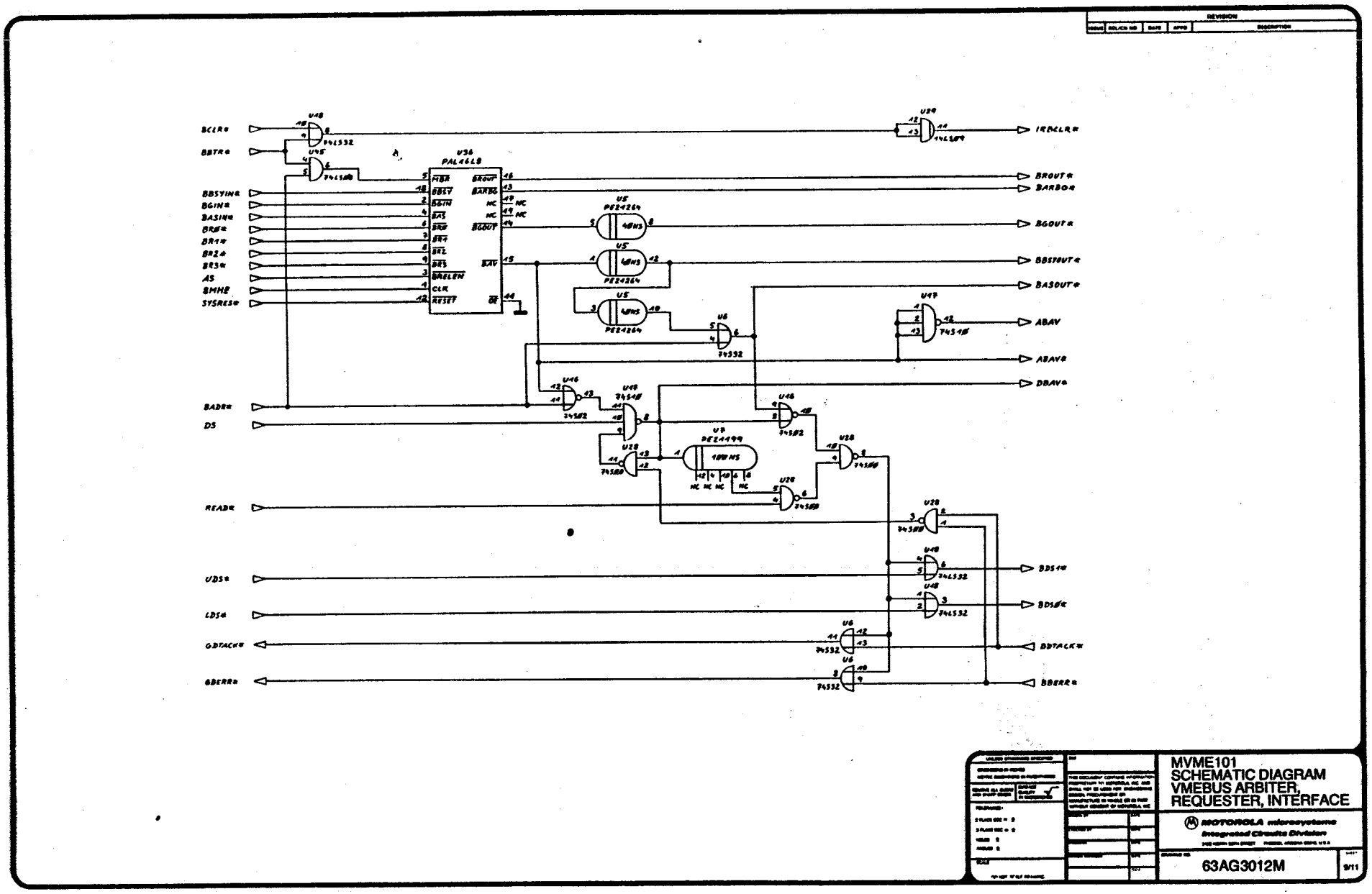
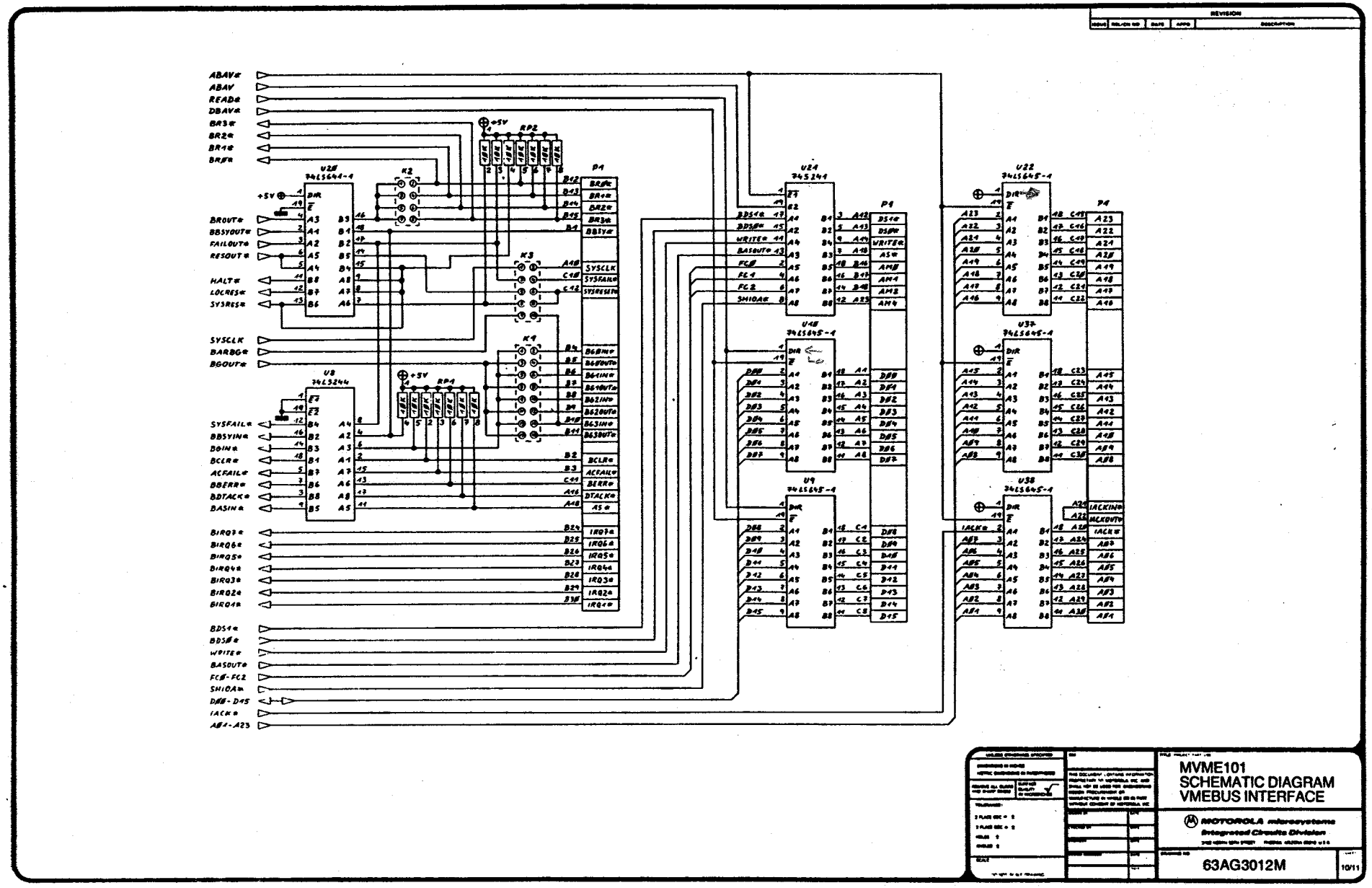


Figure 4.11: Schematic Diagram Sheet 10/11



MOTOROLA

SEMICONDUCTORS

3501 ED BLUESTEIN BLVD., AUSTIN, TEXAS 78721

MC6840 (1.0 MHz)
MC68A40 (1.5 MHz)
MC68B40 (2.0 MHz)

PROGRAMMABLE TIMER MODULE (PTM)

The MC6840 is a programmable subsystem component of the M6800 family designed to provide variable system time intervals.

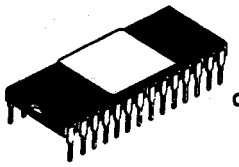
The MC6840 has three 16-bit binary counters, three corresponding control registers, and a status register. These counters are under software control and may be used to cause system interrupts and/or generate output signals. The MC6840 may be utilized for such tasks as frequency measurements, event counting, interval measuring, and similar tasks. The device may be used for square wave generation, gated delay signals, single pulses of controlled duration, and pulse width modulation as well as system interrupts.

- Operates from a Single 5 Volt Power Supply
- Fully TTL Compatible
- Single System Clock Required (Enable)
- Selectable Prescaler on Timer 3 Capable of 4 MHz for the MC6840, 6 MHz for the MC68A40 and 8 MHz for the MC68B40
- Programmable Interrupts ($\overline{IRQ}$) Output to MPU
- Readable Down Counter Indicates Counts to Go Until Time-Out
- Selectable Gating for Frequency or Pulse-Width Comparison
- $\overline{RESET}$ Input
- Three Asynchronous External Clock and Gate/Trigger Inputs Internally Synchronized
- Three Maskable Outputs

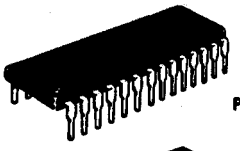
MOS

(N-CHANNEL, SILICON-GATE DEPLETION LOAD)

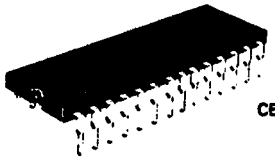
PROGRAMMABLE TIMER



L SUFFIX
CERAMIC PACKAGE
CASE 719



P SUFFIX
PLASTIC PACKAGE
CASE 710



S SUFFIX
CERDIP PACKAGE
CASE 733

MAXIMUM RATINGS

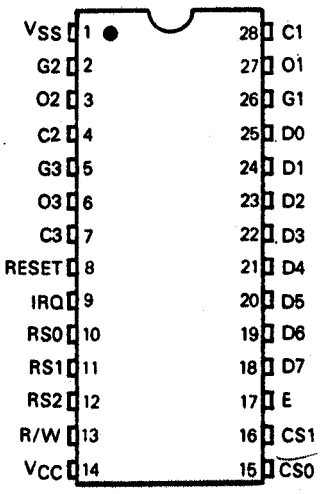
Rating	Symbol	Value	Unit
Supply Voltage	V_{CC}	−0.3 to +7.0	V
Input Voltage	V_{in}	−0.3 to +7.0	V
Operating Temperature Range — T_L to T_h MC6840, MC68A40, MC68B40 MC6840C, MC68A40C	T_A	0 to +70 −40 to +85	°C
Storage Temperature Range	T_{stg}	−55 to +150	°C

THERMAL CHARACTERISTICS

Characteristic	Symbol	Value	Unit
Thermal Resistance Cerdip Plastic Ceramic	θ_{JA}	 65 115 60	°C/W

This device contains circuitry to protect the inputs against damage due to high static voltages or electric fields; however, it is advised that normal precautions be taken to avoid application of any voltage higher than maximum rated voltages to this high-impedance circuit. Reliability of operation is enhanced if unused inputs are tied to an appropriate logic voltage level (e.g., either V_{SS} or V_{CC}).

FIGURE 1 — PIN ASSIGNMENT



DS-9802 R1

MOTOROLA

SEMICONDUCTORS

3501 ED BLUESTEIN BLVD., AUSTIN, TEXAS 78721

MC6821
(1.0 MHz)
MC68A21
(1.5 MHz)
MC68B21
(2.0 MHz)

PERIPHERAL INTERFACE ADAPTER (PIA)

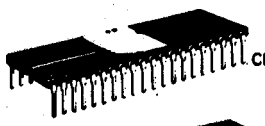
The MC6821 Peripheral Interface Adapter provides the universal means of interfacing peripheral equipment to the M6800 family of microprocessors. This device is capable of interfacing the MPU to peripherals through two 8-bit bidirectional peripheral data buses and four control lines. No external logic is required for interfacing to most peripheral devices.

The functional configuration of the PIA is programmed by the MPU during system initialization. Each of the peripheral data lines can be programmed to act as an input or output, and each of the four control/interrupt lines may be programmed for one of several control modes. This allows a high degree of flexibility in the overall operation of the interface.

- 8-Bit Bidirectional Data Bus for Communication with the MPU
- Two Bidirectional 8-Bit Buses for Interface to Peripherals
- Two Programmable Control Registers
- Two Programmable Data Direction Registers
- Four Individually-Controlled Interrupt Input Lines; Two Usable as Peripheral Control Outputs
- Handshake Control Logic for Input and Output Peripheral Operation
- High-Impedance Three-State and Direct Transistor Drive Peripheral Lines
- Program Controlled Interrupt and Interrupt Disable Capability
- CMOS Drive Capability on Side A Peripheral Lines
- Two TTL Drive Capability on All A and B Side Buffers
- TTL-Compatible
- Static Operation

MOS
(N-CHANNEL, SILICON-GATE, DEPLETION LOAD)

PERIPHERAL INTERFACE ADAPTER



L SUFFIX
CERAMIC PACKAGE
CASE 715

S SUFFIX
CERDIP PACKAGE
CASE 734

P SUFFIX
PLASTIC PACKAGE
CASE 711

MAXIMUM RATINGS

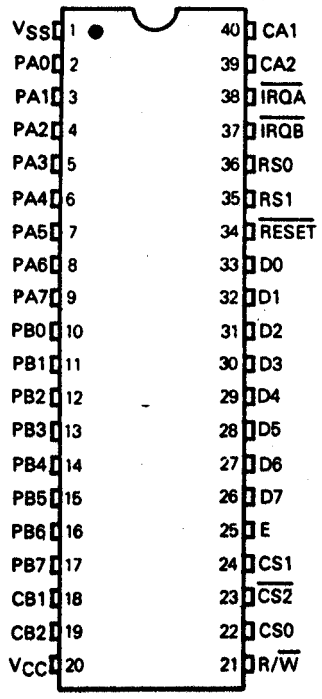
Characteristics	Symbol	Value	Unit
Supply Voltage	V_{CC}	−0.3 to +7.0	V
Input Voltage	V_{in}	−0.3 to +7.0	V
Operating Temperature Range MC6821, MC68A21, MC68B21 MC6821C, MC68A21C, MC68B21C	T_A	T_L to T_H 0 to 70 −40 to +85	°C
Storage Temperature Range	T_{stg}	−55 to +150	°C

THERMAL CHARACTERISTICS

Characteristic	Symbol	Value	Unit
Thermal Resistance Ceramic Plastic Cerdip	θ_{JA}	 50 100 60	°C/W

This device contains circuitry to protect the inputs against damage due to high static voltages or electric fields; however, it is advised that normal precautions be taken to avoid application of any voltage higher than maximum-rated voltages to this high-impedance circuit. Reliability of operation is enhanced if unused inputs are tied to an appropriate logic voltage (i.e., either V_{SS} or V_{CC}).

PIN ASSIGNMENT



DS9435-R2

MOTOROLA

SEMICONDUCTORS

3501 ED BLUESTEIN BLVD., AUSTIN, TEXAS 78721

MC2661A/MC68661A
(Baud Rate Set A)
MC2661B/MC68661B
(Baud Rate Set B)
MC2661C/MC68661C
(Baud Rate Set C)

Advance Information

MOS
(N-CHANNEL, SILICON-GATE)

ENHANCED PROGRAMMABLE COMMUNICATIONS INTERFACE (EPCI)

ENHANCED PROGRAMMABLE COMMUNICATIONS INTERFACE (EPCI)

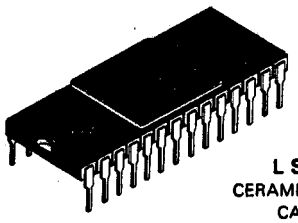
The MC2661/MC68661, Enhanced Programmable Communications Interface (EPCI), is a universal synchronous/asychronous data communications controller chip that is an enhanced version of the Signetics 2651. The EPCI directly interfaces to most 8-bit MPUs and easily to the MC68000 MPU and other 16-bit MPUs. It may be used in either a polled or interrupt driven system. Programmed instructions can be accepted from the host MPU while supporting many synchronous or asynchronous serial-data communication protocols in a full or half-duplex mode. Special support for BISYNC is provided.

The EPCI converts parallel data characters, accepted from the microprocessor data bus, into transmit-serial data. Simultaneously, the EPCI can convert receive-serial data to parallel data characters for input to the microprocessor.

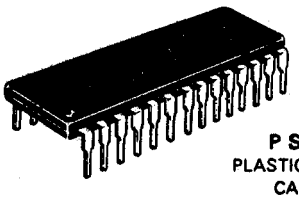
A baud rate generator in the EPCI can be programmed to either accept an external clock, or to generate internal transmit or receive clocks. Sixteen different baud rates can be selected under program control when operating in the internal clock mode. Each version of the EPCI (A, B, C) has a different set of baud rates.

FEATURES

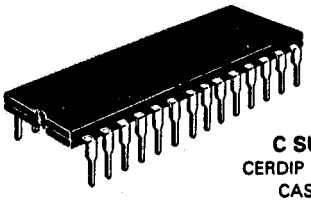
- **Synchronous Operation**
  - Single or Double SYN Operation
  - Internal or External Character Synchronization
  - Transparent or Non-transparent Mode
  - Transparent Mode DLE Stuffing (Tx) and Detection (Rx)
  - Automatic SYN or DLE-SYN Insertion
  - SYN, DLE, and DLE-SYN Stripping
  - Baud Rate: dc to 1M bps (1X Clock)
- **Asynchronous Operation**
  - 1, 1½, or 2 Stop Bits Transmitted
  - Parity, Overrun, and Framing Error Detection
  - Line Break Detection and Generation
  - False Start Bit Detection
  - Automatic Serial Echo Mode (Echoplex)
  - Baud Rate: dc 1M bps (1X Clock)
    dc to 62.5k bps (16X Clock)
    dc to 15.625k bps (64X Clock)
- **Common Features**
  - Internal or External Baud Rate Clock; No System Clock Required
  - 3 Baud Rate Sets (A, B, C); 16 Internal Rates for Each Set
  - 5- to 8-Bit Characters plus parity; Odd, Even, or No Parity
  - Double Buffered Transmitter and Receiver
  - Dynamic Character Length Switching
  - Full- or Half-Duplex Operation
  - Local or Remote Maintenance Loop-Back Mode
  - TTL-Compatible Inputs and Outputs
  - RxC and TxC Pins and Short Circuit Protected
  - 3 Open-Drain MOS Outputs can be Wire ORed
  - Single 5 V Power Supply
- **Applications**
  - Intelligent Terminals
  - Network Processors
  - Front End Processors
  - Remote Data Concentrators
  - Computer-to-Computer Links
  - Serial Peripherals
  - BISYNC Adaptors



L SUFFIX
CERAMIC PACKAGE
CASE 719

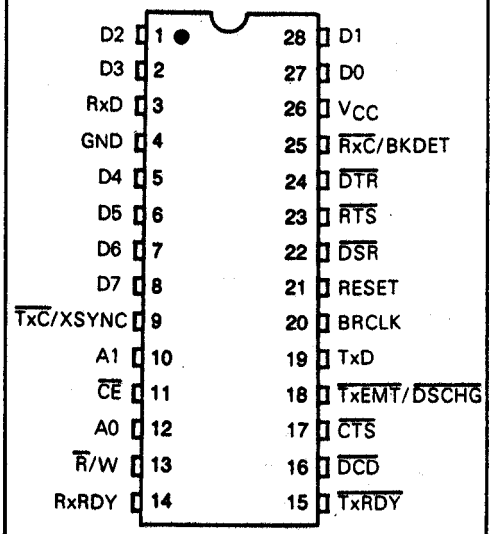


P SUFFIX
PLASTIC PACKAGE
CASE 710



C SUFFIX
CERDIP PACKAGE
CASE 733

PIN ASSIGNMENT



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

```
                0080  A TCSR   EQU    $8       TIMER C/S REGISTER
                000D  A CAPREG EQU    $D       CAPTURE REGISTER
                0011  A TRCSR  EQU    $11      TX/RX C/S REGISTER
                0012  A RXBUF  EQU    $12      RECEIVE BUFFER
                0080  A IRQTST EQU    $80      RAM BYTE AT $0080
2000                    ORG    $2000
                      *IRQ1 SERVICE ROUTINE
2000  73  0080  A IRQSRV COM    IRQTST   CHANGE IRQTST!
2003  30              TSX              X=SP+1
2004  A6  00    A     LDAA   0,X       THE CCR BYTE ON STACK
2006  8A  10    A     ORAA   #$10      SET I-BIT FOR RETURN
2008  A7  00    A     STAA   0,X
                      *BEFORE RTI, SEE IF OTHER INTERRUPTS ARE PENDING
200A  96  08    A     LDAA   TCSR      CHECK INPUT CAPTURE
200C  2B  09 2017     BMI    TIMIC1    TIMER INPUT CAPTURE PENDING
200E  DC  11    A     LDD    TRCSR     CHECK SCI IRQ2 REQUESTS
2010  85  E0    A     BITA   #$E0      CHK RDRF,ORFE,TDRE FLAGS
2012  26  08 201C     BNE    SCIIN2    SERVICE SCI INTERRUPT
2014  3B              RTI              EXIT:NO INTERRUPTS PENDING
                      *I-BIT IS SET TO PREVENT IRQ1 RESERVICE

2015  96  08    A TIMIC  LDAA   TCSR      ARM ICF FOR CLEARING
2017  DC  0D    A TIMIC1 LDD    CAPREG    CLR ICF, GET CAPTURE DATA
                *        .
2019  3B                RTI

201A  DC  11    A SCIINT LDD    TRCS      ACCA=TRCS, ACCB=RXBUF
201C  48                SCIIN2 ASLA      SORT OUT SCI FLAGS
                *        .
201D  3B                RTI
```

FIGURE A-2
Routine IRQSRV can also poll other interrupt requests when using $\overline{IRQ1}$ as an input.

INTERFACING M6800 PERIPHERAL DEVICES TO THE MC68000 ASYNCHRONOUSLY

Prepared by:
Arnold J. Morales
Microprocessor Applications Engineer

This application note describes a technique for interfacing M6800 peripheral devices to a MC68000 microprocessor using a four-chip TTL circuit. Any M6800 peripheral is easily interfaced to the MC68000 using the M6800 peripheral control interface (E, $\overline{VMA}$, $\overline{VPA}$) that is designed into the MC68000. However, when using this interface, the peripheral must be driven by the MC68000 enable (E) signal. The frequency of this clock is one-tenth of the MC68000 clock frequency with a 60/40 (6 clocks high, 4 clocks low) duty cycle. Certain applications may require a clock frequency other than the one-tenth sample that is readily available. An application using a MC68B54 Advanced Data Link Controller (ADLC) at a high data transfer rate could require an E clock frequency of up to two megahertz because the data transfer rate of the ADLC depends on the transmit and receive clocks, which are limited by the E clock frequency.

TIMING CONSIDERATIONS

Typical read and write timing for the MC68000 is shown in Figure 1. The relationship between the MC68000 timing and the access timing for the interface circuit given in this application is shown in Figure 2. The best case timing has data strobe occurring with the minimum setup time to allow peripheral selection on the next falling edge of the E clock. In the worst case timing, the data strobe did not occur in time to allow peripheral selection on the next falling edge of E. Therefore, a full E cycle has to occur and then the peripheral selection is done on the falling edge of that full cycle. The resulting cycle times for these best and worst cases and a comparison between asynchronous and synchronous interfacing is summarized in Table 1.

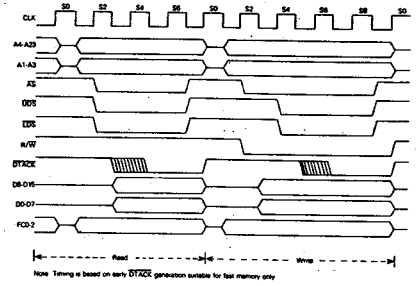


Figure 1. MC68000 Read and Write Cycle Timing Diagram

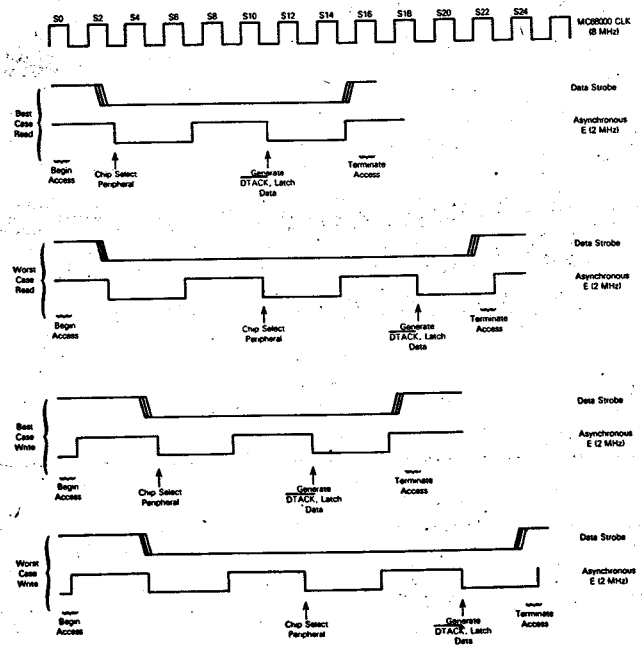


Figure 2. Asynchronous Interface Access Timing Diagrams

BLOCK DIAGRAM

Figure 3 is a block diagram of a circuit that allows M6800 peripherals, operating at any frequency within their operating range, to be asynchronously interfaced to a MC68000 processor. The data bus is driven by a pair of octal transparent latches. The latch control circuitry uses M6800 peripheral chip select and the R/$\overline{W}$ line of the MC68000 for output enable and data direction information. The $\overline{DTACK}$ signal from the $\overline{DTACK}$ generation circuit latches data into the enabled octal latch when the peripheral is deselected.

The peripheral select and $\overline{DTACK}$ generation circuit uses a data strobe (either upper or lower) from the MC68000, peripheral E, and a M6800 peripheral chip select signal to select the peripheral and generate $\overline{DTACK}$.

CIRCUIT OPERATION

Figure 4 is a schematic diagram of the interface circuitry. Refer to this diagram during the following discussion. Initially flip flops U1A and U1B are cleared causing a high $\overline{DTACK}$ output setting U2 and U3 to a transparent mode. Latch U2 is in the high-impedance state due to a high on the output enable ($\overline{OE}$) input. Latch U3 is enabled due to a low on the $\overline{OE}$ input.

At the start of a M6800 peripheral access, latch U3 remains enabled if the access is a MC68000 write. If the access is a read, the high R/$\overline{W}$ and CS inputs to U4A cause U3 to go to the high-impedance state and U2 to become enabled. The peripheral is selected by a low chip select prime ($\overline{CS}$'). Flip flop U1A is clocked high on the first falling edge of E with the system chip select (CS) and data strobe ($\overline{DS}$) high. The Q output of U1A is applied to U4D, asserting $\overline{CS}$'. Selecting the peripheral at this time ensures that the peripheral has adequate address setup time.

On the next falling edge of E, the $\overline{Q}$ output of U1B is clocked low asserting $\overline{DTACK}$ and latching data into the enabled latch. The asserted $\overline{DTACK}$ signal, inverted by U4D, deselects the peripheral by causing $\overline{CS}$' to go high. Flip flop U1 is cleared by DS going low when the access terminates. Clearing U1 also initializes the interface circuitry for the next access.

Table 1. Synchronous and Asynchronous Interface Access Time

	Read Access Times (MC68000 Cycles)		Write Access Times (MC68000 Cycles)	
	Best Case	Worst Case	Best Case	Worst Case
Synchronous	9	18	9	18
Asynchronous	8	11	9	12

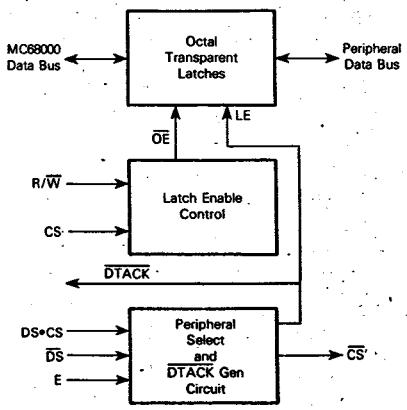


Figure 3. M6800 Peripheral to MC68000 Interface — Block Diagram

SAMPLE CIRCUIT

An example of this interface circuitry is given in the following paragraphs. This example illustrates how the MC68000 can be interfaced to both a MC6854 Advanced Data Link Controller (ADLC) and a MC6840 Programmable Timer Module (PTM) at the same time. The circuit shown in Figure 5 uses the two megahertz "B" version parts connected to a MC68000 driven at eight megahertz.

The base addresses for the peripherals in this example are $18001 for the ADLC and $18801 for the PTM. When the MC68000 transfers bytes it asserts the upper data strobe for even addresses and the lower data strobe for odd addresses. The circuit in this example uses the lower data strobe; therefore only odd MC68000 addresses are used. A memory map of the example system is given in Figure 6.

Device Selection — A SN74LS138 1-of-8 decoder (U5) used as an address decoder is used in conjunction with a chip select signal (CS') developed by U6E to select either the ADLC or the PTM. The PTM requires two chip select inputs, one high and one low, to be selected. The low input is provided by the $\overline{O5}$ output of U5 while the high input is provided by an inverted sample of the $\overline{CS}$', developed by U6E.

To select the ADLC, the $\overline{O5}$ output of U5 must be high and the $\overline{O1}$ output must be low. The ANDing of $\overline{O5}$ with the high CS' developed by U6E generates the low chip select input required by the ADLC.

Test Program — A flow chart of the test program is given in Figure 7 and a listing is provided in Figure 8. Refer to these figures during the following discussions. The first five lines of code initiate operation of timer 3 in the PTM in the continuous mode, resulting in a square wave at the output of timer 3, pin 6. The remaining lines of code are for testing the ADLC. The test program is based on the loopback test program given in Motorola publication MC6854UM (AD).

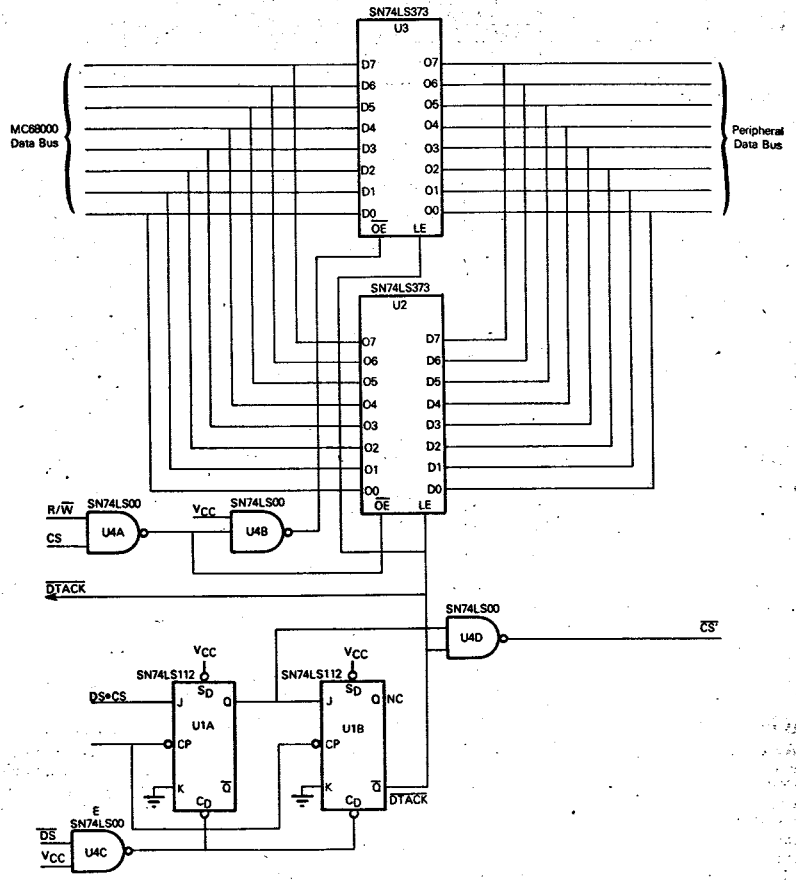


Figure 4. M6800 Peripheral to MC68000 Interface — Schematic Diagram

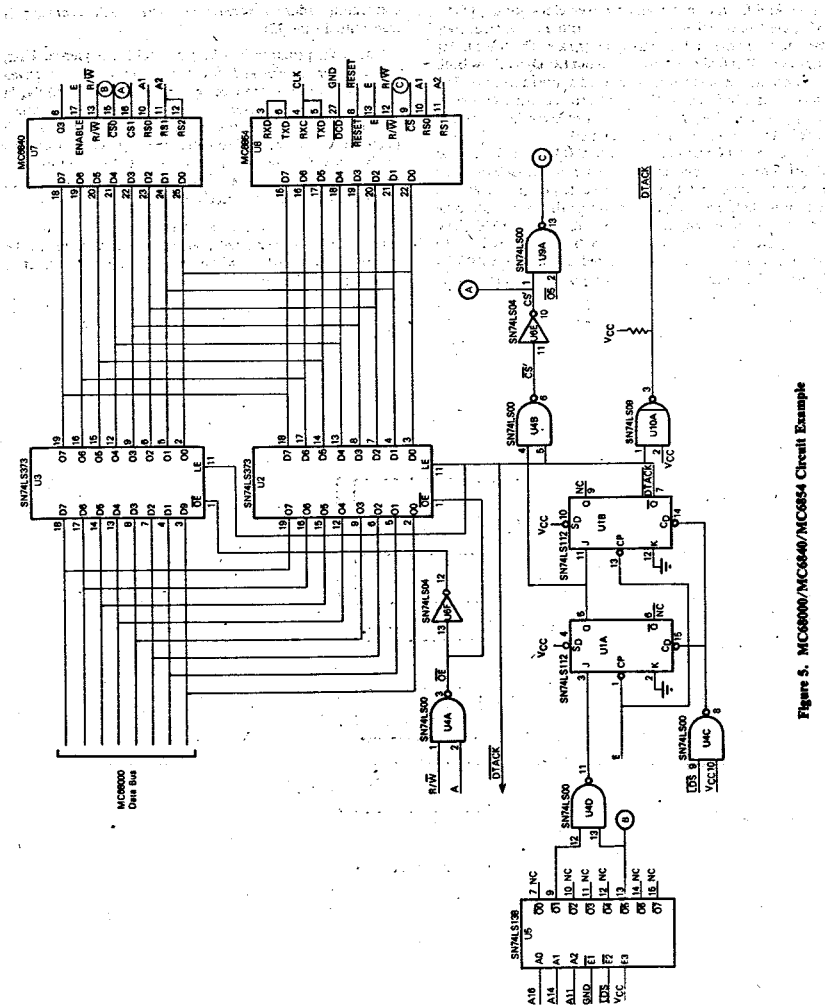


Figure 5. MC68000/MC6840/MC6854 Circuit Example

The ADLC transmitter and receiver clock inputs (TxC, RxC), are tied together and provided with a clock frequency determined by the desired data transfer rate. The transmitter output (TxD) is tied to the receiver input (RxD) to allow both the transmitter and receiver to be tested at the same time. The test consists of initializing the ADLC, transmitting a series of data bytes, and then storing the data received in a memory buffer based at address labled RECBUF.

The byte to be transmitted, labeled DATA, is located at address $3000. This address is entered into MC68000 address register A1, which will be used as the data pointer for data to be transmitted. The program transmits the same data byte 128 times, a count established by the initial value in MC68000 data register D0. The program can be easily modified to transmit a block of characters based at address $3000 by changing the initial value in data register D0, and postincrementing address register A1 after each character is transmitted (line 72).

The main program is a looping, polling sequence. First, the receiver is checked for the presence of a received character by testing the receiver data available (RDA) flag in the ADLC. If a character is present, it is stored in the received data buffer. The transmitter data register available (TDRA) is then checked to determine whether the transmitter is ready for another byte of data. If the transmitter is ready, another data byte is transmitted. The program then loops back to check the receiver again.

Before each character is transmitted, MC68000 data register D0 is decremented and tested. Termination of the program is initiated when the correct number of characters have been transmitted.

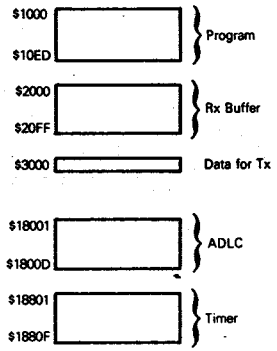


Figure 6. Memory Map

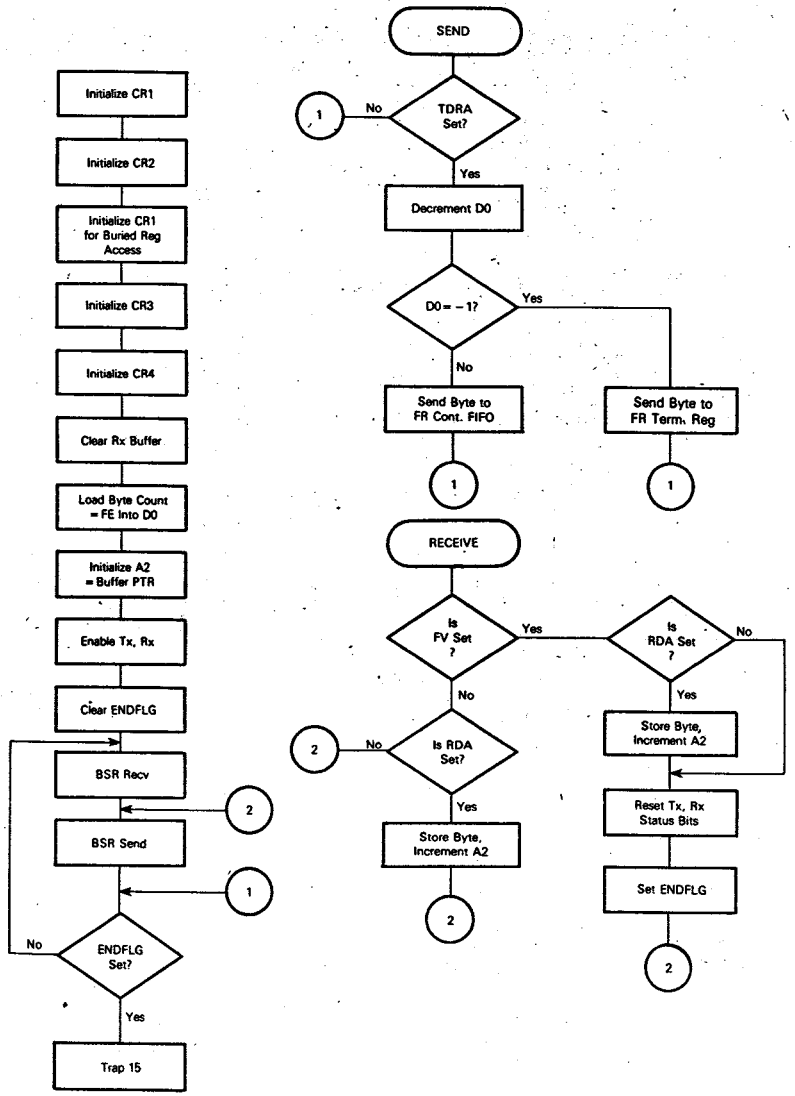


Figure 7. Program Flowchart

```
SUPERT2.  MC68000 ASM REV= 1.1 - COPYRIGHT BY MOTOROLA 1978          PAGE  1

                        *       PROGRAM TO INITIALIZE MC6840
                        *       ENABLE TIMER 3
                        *
00001000                ORG $1000
                        *       EQUATES
00018801        WCR13   EQU $18801              W.C.R. 1,3
00018803        WCR2    EQU $18803              W.C.R. 2
0001880D        MSBT3   EQU $1880D              M.S.B. BUFFER
0001880F        LSBT3   EQU $1880F              L.S.B. TIMER 3

001000 13FC0082
       00018801 START   MOVE.B #$82,WCR13       INITIALIZE TIMER 3
001008 13FC0000
       0001880D         MOVE.B #$00,MSBT3       M.S.B TIMER 3=00
001010 13FC000F
       0001880F         MOVE.B #$0F,LSBT3       L.S.B.TIMER 3=OF
001018 13FC0001
       00018803         MOVE.B #$01,WCR2        ACCESS W.C.R.1
001020 13FC0000
       00018801         MOVE.B #$00,WCR13       ENABLE TIMERS

                        NOPAGE

                * ADLC PROGRAM TO INITIALIZE ADLC AND BACKEND TRANSMIT

                * EQUATES
00018000        ADLC    EQU   $18000            ADLC BASE ADDRESS
00002000        RECVBF  EQU   $2000             RECEIVER BUFFER
00003000        DATA    EQU   $3000             LOCAT OF DATA TO BE XM
00001500        ENDFLG  EQU   $1500             ALL FINISHED FLAG

                * DEFINE RAM WORK AREA

                * START OF PROGRAM CODE

                * INITIALIZE ADLC
001028 13FC00C0
       00018001 FIREUP  MOVE.B  #$C0,ADLC+1     C.R. 1
001030 13FC0064
       00018003         MOVE.B  #$64,ADLC+3     C.R. 2
001038 13FC00C1
       00018001         MOVE.B  #$C1,ADLC+1     C.R. 1
001040 423900018003     CLR.B ADLC+3            C.R. 3
001046 13FC001F
       00018007         MOVE.B  #$1F,ADLC+7     C.R. 4

                * CLEAR MEMORY RECEIVE BLOCK
00104E 43F82000         LEA     RECVBF,A1       LOAD BUFFER PTR
001052 343C003F         MOVE.W #($100/4)-1,D2   SIZE IS 256 BYTES
001056 4299     CLEAR   CLR.L   (A1)+            CLEAR NEXT WORD, INCR
001058 51CAFFFC         DBRA    D2,CLEAR        LOOP UNTIL DONE

                * SETUP REGISTERS
00105C 203C000000FE     MOVE.L #$FE,D0          SETUP COUNT-2 , XMIT
001062 43F83000         LEA     DATA,A1         SETUP DATA ADDRESS
001066 45F82000         LEA     RECVBF,A2       LOAD BUFFER PTR
00106A 423900018001     CLR.B   ADLC+1          ENABLE RX, TX
001070 42381500         CLR.B   ENDFLG          SET FLAG FOR NOT FINI
```

Figure 8. Program Listing


```
                * PROCESS TRANSMIT AND RECEIVE TASK TILL DONE
001074 61000038 PROCES  BSR     RECV            ATTEMPT RECEIVE
001078 6100000C         BSR     SEND            ATTEMPT TO SEND
00107C 4A381500         TST.B   ENDFLG          FINISHED?
001080 67F2             BEQ     PROCES          LOOP IF NOT FINISHED
001082 4E4F             TRAP    15              BREAKPOINT WHEN FINIS
001084 0000             DC.W    #0              BREAKPOINT CODE

                * ATTEMPT TO TRANSMIT A CHARACTER
001086 08390006
       00018001 SEND    BTST.B #06,ADLC+1       TDRA SET? XMIT READY?
00108E 67000014         BEQ RETURN
001092 0C40FFFF         CMP  #-1,D0             ? LAST BYTE SENT?
001096 6700000C         BEQ     RETURN          YES IGNORE SENDING MO
00109A 51C8000A         DBRA    D0,MORE         COUNT DOWN, BRANCH NO
                * PROCESS LAST BYTE BY TERMINATING TRANSMISSION
0010A0 13D100018007     MOVE.B (A1),ADLC+7      LAST BYTE INTO FRAME
0010A4 4E75     RETURN  RTS                     RETURN TO MAINLINE
                * PROCESS NEXT BYTE TO TRANSMIT (NOT THE LAST)
0010A6 13D100018005 MORE    MOVE.B (A1),ADLC+5  SEND NEXT BYTE TO FRA
0010AC 4E75             RTS                     RETURN TO CALLER

                * ATTEMPT TO RECEIVE A CHARACTER
0010AE 08390001
       00018003 RECV    BTST.B  #1,ADLC+3       ? FRAME RECEIVED
0010B6 66000008         BNE     GOTFRM          BRANCH IF SO
0010BA 61000016         BSR     TRYINP          ATTEMPT INPUT OF NEXT
0010BE 4E75             RTS                     RETURN TO MAINLINE
                * END OF FRAME PROCESSING
0010C0 61000010 GOTFRM  BSR     TRYINP          INSURE LAST BYTE PROC
0010C4 13FC0064
       00018003         MOVE.B #$64,ADLC+3      CLEAR TX,RX STATUS
0010CC 52381500         ADD.B   #1,ENDFLG       FLAG RECEIVER DONE
0010D0 4E75             RTS                     RETURN TO MAINLINE

                * PROCESS INPUT BYTE IF ANY
0010D2 08390000
       00018001 TRYINP  BTST.B #0,ADLC+1        ? INPUT BYTE READY, T
0010DA 67C8             BEQ     RETURN          RETURN IF NO BYTE
0010DC 14F900018005     MOVE.B ADLC+5,(A2)+     STORE BYTE
0010E2 4E75             RTS                     RETURN

                        END
                *©

****** TOTAL ERRORS   0--   0

SYMBOL TABLE

ADLC    018000 CLEAR    001056 DATA     003000 ENDFLG   001500
FIREUP  001028 GOTFRM   0010C0 LSBT3    01880F MORE     0010A6
MSBT3   01880D PROCES   001074 RECV     0010AE RECVBF   002000
RETURN  0010A4 SEND     001086 START    001000 TRYINP   0010D2
WCR13   018801 WCR2     018803
```

Figure 8. Program Listing (Concluded)

AN-815

COLOR GRAPHICS FOR THE MC68000 USING THE MC6847

Prepared by:
Rex Davis
Microprocessor Applications Engineer

Color graphics can be a valuable addition to a variety of MC68000 applications. Typically, color graphics circuits are expensive and complicated. These same color graphics circuits generally have limited capabilities and are very hardware intensive. The MC6847 Video Display Generator (VDG) offers a low cost, versatile, easy to use alternative.

The VDG creates a composite video signal according to information read from the display memory in the mode defined by the VDG control pins. The composite video signal can be used to modulate the input of any commercially available color or black and white television receiver. The VDG has 12 distinct display modes available and allows certain variations within certain modes. All display modes and their variations are controlled by the state of eight different VDG pins. The eight display mode pins give the user the ability to combine display modes within a single display frame, e.g., alphanumerics and a compatible graphics mode.

The 16-bit data bus of the MC68000 can be used to give the user full control of all VDG display modes on-the-fly. This additional control over the VDG display modes, combined with the ability of the MC68000 to manipulate data through its extended arithmetic capabilities and its ability to move blocks of data quickly and efficiently, gives the user the capability to monitor and display changing data quickly and efficiently as might be necessary in commercial, industrial, or scientific applications. The MC68000/MC6847 interface described in this application note, or a variation of it, could be used in a typical application.

BASIC IMPLEMENTATION

Since the display information for the VDG is stored in memory, the MC68000 can alter the display by changing the contents of that memory. The MC68000/MC6847 interface then becomes a shared-memory interface. The primary problem then becomes a matter of guaranteeing that memory is accessed by only one device at a time. Figure 1 is a block diagram of the MC68000/MC6847 interface.

The MC68000 is buffered from the display memory by three-state buffers and transceivers which are active only during a processor access of display memory. During a processor access, the VDG address bus is forced to a high-impedance state. Display memory is only 12 bits wide, and a 4-bit latch is used to capture data for some of the VDG control pins. Not all of the VDG display modes may be used in the same frame without some additional attention by the user; however, the latch could be replaced with an additional 6K by 4 bits of memory. A data transfer acknowledge ($\overline{DTACK}$) signal is generated so that either fast or slow memory can be efficiently used.

CIRCUIT DESCRIPTION

A schematic diagram of the MC68000/MC6847 interface is given in Figure 2. The select circuitry is formed around two SN74LS138 1-of-8 decoders. If the MC68000 accesses any of the 6K of memory, pin 8 of the SN74LS21 outputs a signal which switches the display memory bus from the VDG to the MC68000. First, the VDG address bus is put into a high-impedance state. Next, SN74LS138-2 is deselected so that only the memory selected by the MC68000 is accessed. Finally, the data and address buffers connect the MC68000 address and data bus to the display memory. If the MC68000 is not accessing memory, then the MC68000 is isolated from the display memory by placing the data and address buffers in the high-impedance state. Decoder SN74LS138-2 is selected and accesses the display memory required by the VDG. The VDG is then released to scan the memories (MCM2114) for display information.

In order to guarantee that no incompatible display modes are used within the same scan frame, only the last data write to any even memory location can alter the latch; therefore, the VDG will always read the same data from the SN74LS75 latch between MC68000 writes to the display memory. However, the user could still adversely affect the display if the MC68000 writes occurred within the VDG display scan. The frame sync pin of the VDG is low when the VDG is not in the active display window and could be used as an interrupt or polled to determine when the MC68000 can safely write to the display memory.

The $\overline{DTACK}$ signal is generated by the SN74LS95 shift register and is necessary for the completion of any MC68000 display memory access. The shift register will add two additional wait states for each successive output; i.e., Q1 gives two wait states, Q2 gives four wait states. Output Q0 will give no wait states. Table 1 lists the two critical memory specifications: data setup time and access time and the shift register output necessary to guarantee operation.

Typically, access time is the more critical specification. Output Q1 was chosen to be used with the MCM2114-30 (300 ns access time) memories used in this application although faster or slower memories could have been used. The speed of the display memory determines the rate at which data can be moved into the display memory. Other factors determining the speed of the data transfer are: the method used to determine when the VDG is out of the active display area; and the software routine used to move the data.

CONCLUSION

The MC68000/MC6847 interface described in this application can be expanded or limited, according to the user's application, with relative hardware ease. Any approach of shared memory could be used as long as no bus conflicts result. Even the simple approach taken in this application results in a powerful, low cost color graphics system for the MC68000.

Table 1. Required Shift Register Output

	Q0	Q1	Q2	Q3
Data Setup Time (ns)	230	355	480	605
Access Time (ns)	250	375	500	625

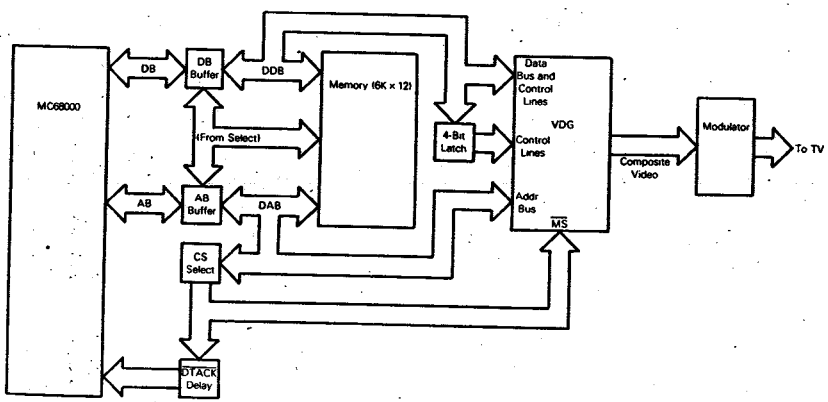


Figure 1. MC68000/MC6847 Interface — Block Diagram

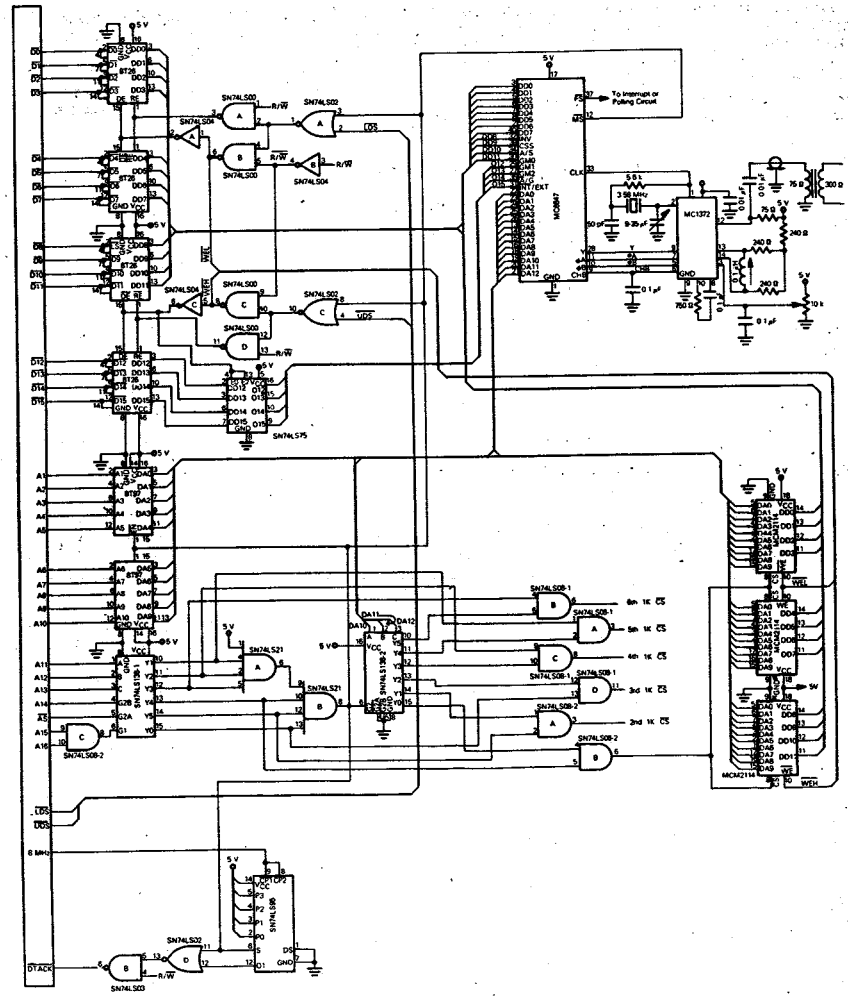


Figure 2. MC68000/MC6847 — Schematic Diagram

SOFTWARE REFRESHED MEMORY CARD FOR THE MC68000

Prepared by:
Duane Graden
Microprocessor Applications Engineer

This application describes the hardware and software to implement a software-refreshed, dynamic memory card for use in an eight megahertz MC68000 system. This refresh approach consumes less than five percent of processor time. The MCM4116 16K RAM was chosen for this design, but the techniques discussed are applicable to the MCM6664 64K RAM as well.

Refresh techniques fall into two categories, hardware and software. Hardware refresh is more component intensive with little or no overhead in program time, while software refresh has less hardware and more program overhead.

Hardware refresh means that the required circuitry must refresh the dynamic RAM cell with little or no impact on execution of instructions by the processor. Normally, this means accessing the address bus during a dead part of the cycle. Another drawback is the complex circuitry, usually requiring the use of expensive delay lines.

Software refresh means that the processor must execute a software routine to refresh dynamic memory. To accomplish this, an interrupt service routine, such as the level seven interrupt service routine on the MC68000, can be dedicated to refresh the memory. Every time the interrupt is recognized, a hardware enable allows the refresh routine to refresh the dynamic RAM.

TIMING SIGNALS

Timing requirements of MCM4116 RAMs and the MC68000 are easy to match because of the asynchronous nature of the MC68000 bus structure. The MC68000 can wait for the slowest RAM through the use of the data transfer acknowledge ($\overline{DTACK}$) signal. As long as $\overline{DTACK}$ is asserted a setup time before the falling edge of any clock state (S4 or later), it will be recognized during that state. Termination of the access is 1½ clock periods later. Figure 1 is a timing diagram for a read, write, and refresh operation.

The $\overline{RAS}$ and $\overline{CAS}$ signals are the row address and column address multiplex control inputs, respectively, for the seven memory address lines A1 through A7. Since no chip select inputs are present with this dynamic memory, $\overline{RAS}$ is the active low signal that starts a memory access cycle. When $\overline{RAS}$ falls, the row address of the location to be accessed is latched into memory. Similarly, the falling edge of $\overline{CAS}$ latches the column address into memory.

The refresh cycle shown in Figure 1, is known as $\overline{RAS}$-only refresh. Row address select is low, $\overline{CAS}$ is high, R/$\overline{W}$ does not matter, and the row address of the row to be refreshed is present on the seven address lines. Each row of memory requires a refresh cycle to be performed every two milliseconds for data to be retained. For the MCM4116 memory, there are 128 rows and, therefore 128 refresh cycles required every two milliseconds.

HARDWARE DESCRIPTION

Figure 2 is the schematic diagram for a dynamic memory card using MCM4116 memories. This card, when used with a MC68000 system, provides 64K bytes of memory from 32K to 96K of the physical address map.

Memory decoding is done with the upper and lower data strobes and address lines A15 and A16. The data strobes divide the memory into even and odd blocks, respectively. The upper data strobe chip selects even bytes from 32K to 96K by activating a row address select upper (RASU) signal. Address lines A15 and A16, through decoder U2 and gate U4, decode whichever of the two banks of even memory (RAS1U or RAS2U) is selected. Similarly, the lower data strobe activates a row address select lower (RASL) signal.

Column address select ($\overline{CAS}$) is activated on the second falling edge of the eight megahertz clock after $\overline{RAS}$ is asserted by flip flop U9. Both $\overline{RAS}$ and $\overline{CAS}$ are turned off when the data strobes are inactive.

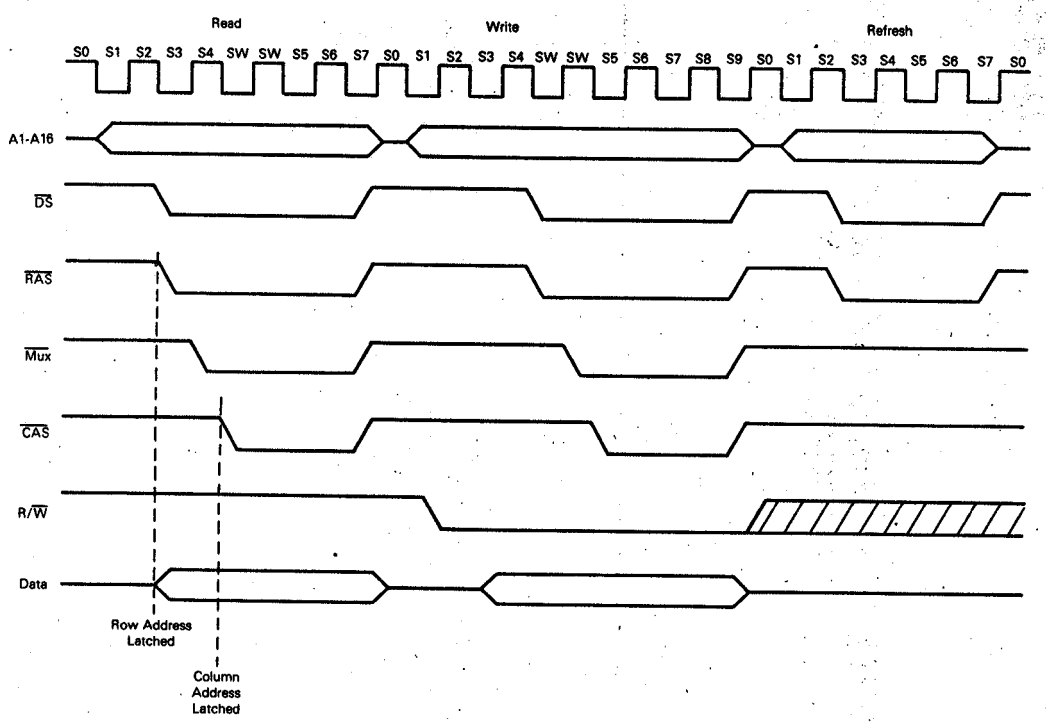


Figure 1. Read, Write, and Refresh Timing Diagrams

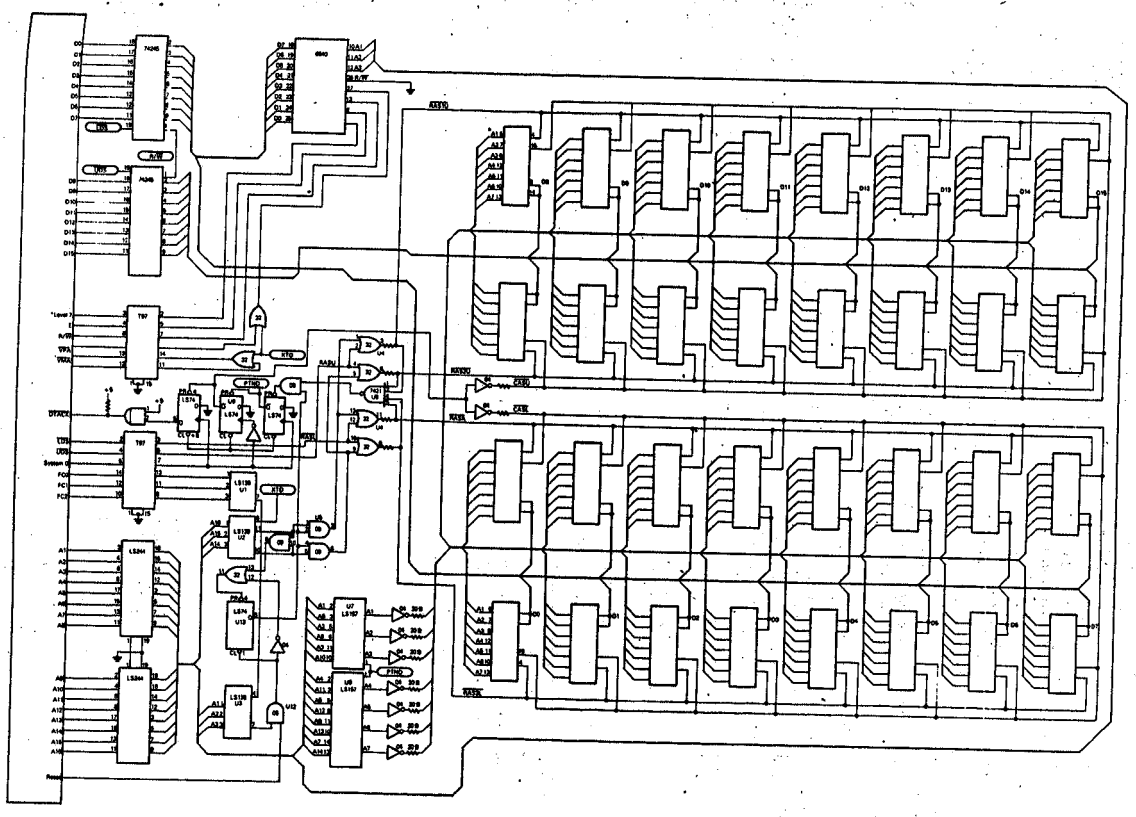


Figure 2. Dynamic Memory Card — Schematic Diagram

Multiplexed addresses for the dynamic memory are supplied by multiplexers U7 and U8. The row address on address lines A1 through A7 is present on the memory address lines until $\overline{RAS}$ is asserted. On the next rising edge of the eight megahertz clock, the column address on address lines A8 through A14 is on the memory address lines. The multiplexed address is valid only when $\overline{RAS}$ or $\overline{CAS}$ is present, making an enable for the multiplexers unnecessary.

Memory refresh is controlled by U11, a MC6840 programmable timer module (PTM). Once programmed, the PTM timer used (the PTM has three timers) causes a level seven interrupt every 1.9 milliseconds (2 milliseconds — routine execution time). This interrupt enables all four banks of memory for simultaneous refresh.

Interrupts with M6800 type peripherals are handled with a reference to the internal vector table. Figure 3 is a schematic of the hardware used with the MC68000 to create a vectored interrupt (level one to level seven). The level present on the $\overline{IPL0}$, $\overline{IPL1}$, and $\overline{IPL2}$ lines is checked against the interrupt level of the processor. If it is higher than the internal level, an interrupt sequence is started. The function code outputs will be high and address lines A1, A2, and A3 will be the vector number of the interrupt being serviced (in this case, all high). Now decoders U1 and U3 (Figure 1) decode the level seven interrupt and generate valid peripheral address ($\overline{VPA}$) to the MC68000 through U13 and U9. The assertion of valid peripheral address causes the internal vector table entry for level seven to be fetched and used as the starting location of the service routine. At the same time, U12 and U13 enable all $\overline{RAS}$ signals and disable $\overline{CAS}$ for refresh of the memories.

OPTIONAL HARDWARE

One situation may occur with the memory card where data might be lost. If the reset button is held closed too long, data could be lost. To prevent this, the circuitry shown in Figure 4 can be added. This provides for a single E cycle reset which will retain the integrity of the stored data.

When power is initially applied to the MC68000, a reset must occur for at least 100 milliseconds after the supply voltage has reached 4.75 volts for proper power-up reset. This means that a one shot or a resistor-capacitor combination should be used to hold the clear pin of the flip flops at or below the logic low level (0.8 volts) for the required time. The E signal will clock the 2-bit counter twice. This presets flip flop U3, removing the system reset. On a non-powerup reset, the reset switch is closed, clocking a low into flip flop U3. Gate U4 provides debounce of the reset switch, allowing only one clock pulse into flip flop U3. Again, E will clock the counter removing reset.

SOFTWARE

Row address select-only refresh is the refresh method used in this application. It is accomplished by a hardware enable (level seven interrupt) and 128 NOPs for the service routine. The level seven interrupt being low enables all four $\overline{RAS}$ signals and disables $\overline{CAS}$. Each NOP increments the address bus to provide the 128 row addresses (0 to 127) needed for refreshing all four banks of memory. Incrementing the address bus accesses and refreshes that row.

However, this has one problem — reset. If a reset occurs just prior to an interrupt for software refresh, data could be lost due to a late or missing refresh cycle. This problem is solved by locating the software refresh routine at the beginning of the reset code. A hardware enable for reset refresh enables $\overline{RAS}$-only refresh in the same way that the level seven interrupt signal did for a normal refresh. In addition, the refresh at reset must load the stack with a valid return address, to return to when the return from interrupt (RTI) instruction is executed at the end of the refresh routine. Figure 5 is a listing of this software with comments to document the reset refresh.

Refresh is enabled at restart by U10 and U13. All $\overline{RAS}$ signals are on and all $\overline{CAS}$ signals are off. Like a normal refresh operation, $\overline{CAS}$ is enabled by the first access to memory after the refresh routine. Software refresh with the MC68000 is an efficient option to implement dynamic RAM without costly delay lines. The application presented here has only a five percent program time overhead.

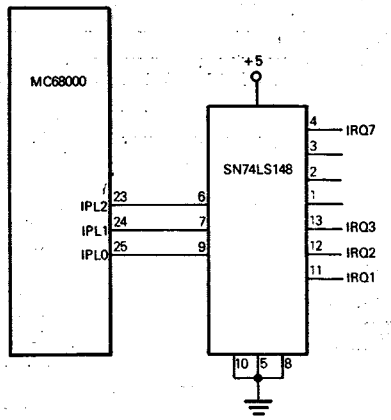
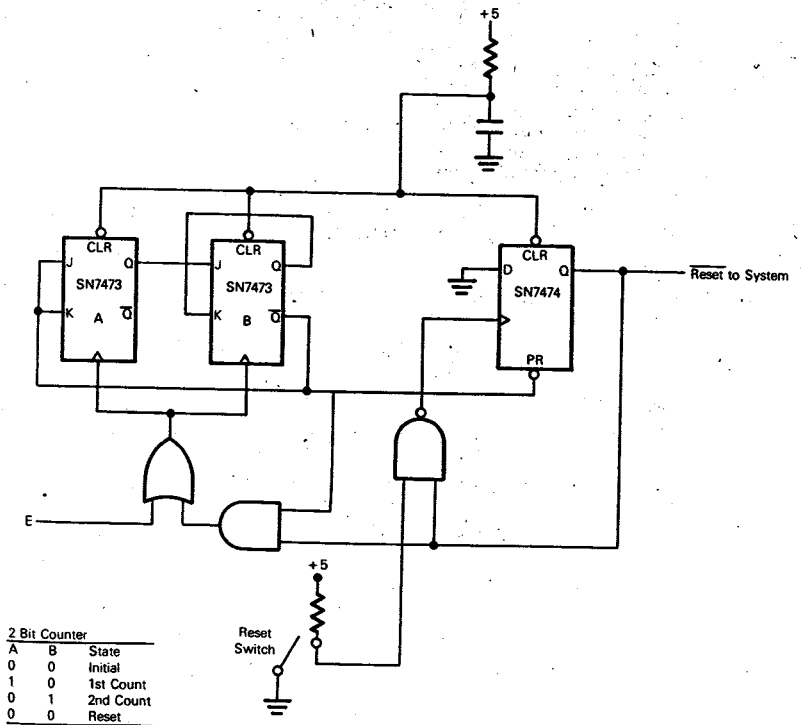


Figure 3. Single-Cycle Reset Circuit — Schematic Diagram



2 Bit Counter

A	B	State
0	0	Initial
1	0	1st Count
0	1	2nd Count
0	0	Reset

Figure 4. Vectored Interrupt Circuit — Schematic Diagram

```
PAGE 001  S68K    .SA:0

DATAR1 EQU $18005
DATAR  EQU $18007
CTR1   EQU $18001
CTR2   EQU $18003
 PEA FIREUP          ****
*                      * LOAD STACK WITH
*                      * USER INITIALIZATION
*                      * AND STATUS REGISTER
 MOVE SR,D           ****
 MOVE D1,-(SP)      *****
 MOVE,B #$FE,CTR2       *
 MOVE,B #$09,DATAR1     * INITIALIZE PIM TIMER
 MOVE,B #$47,DATAR      *
  MOVE,B #$7D,CTR1 *****
*
*                      * LEVEL 7 INTERRUPT
*                      * ENTRY POINT
*
 NOP                 ******
 NOP                    *
 NOP                    *
                        *
                        * 128 NOP'S FOR REFRESH
                        *
 NOP                    *
 NOP                    *
 NOP                 ******
 RTE
FIREUP ******* BEGINING OF USER RESET INITIALIZATION
```

Figure 5. Program Listing

ASYNCHRONOUS COMMUNICATIONS FOR THE MC68000 USING THE MC6850

Prepared by:
Charles Melear
Microprocessor Applications Engineer

Interfacing the MC6850 Asynchronous Communications Interface Adapter (ACIA) to the MC68000 is easy due to the fact that the MC68000 has a special cycle to handle M6800 peripherals. The ACIA data bus can be placed on either the upper or lower eight bits of the MC68000 data bus with equivalent results. Using the upper byte implies an even address and use of the upper data strobe ($\overline{UDS}$), and the lower byte implies an odd address and the use of the lower data strobe ($\overline{LDS}$). In this application, the ACIA is placed on the lower byte of the data bus.

INTERCONNECTIONS

Enable (E) and R/$\overline{W}$ are connected to the corresponding pins of the MC68000. Several signals are generated to form chip selects as shown in Figure 1. Valid memory address ($\overline{VMA}$) from the MC68000 is an active low signal (as opposed to active high for the MC6800) as well as $\overline{LDS}$. The NOR of the two signals is used to develop CS1. The address $F3FFXX is generated by address lines A8 through A23 to enable a SN74LS154 four-to-sixteen line selector. Address lines A4 through A7 are used to generate a low output at 02 of the SN74LS154 to be used for CS2 of the ACIA. Address line A1 is used for the register select (RS) pin of the ACIA. This puts the ACIA status register at address $F3FF21 and the control register at address $F3FF23. If the ACIA has been placed on the upper byte, the addresses would be $F3FF20 and $F3FF22, respectively. To complete the circuit, a signal called valid peripheral address ($\overline{VPA}$) must be generated and returned to the MC68000 to indicate that a M6800 cycle is being executed. The SN74LS154 has two active low chip enable lines which are driven by the gates that form address $F3FFXX from address lines A8 through A23. Since the SN74LS154 always picks M6800 peripherals, the two chip enable lines can be ORed to develop $\overline{VPA}$. Since more than 16 peripherals could exist, it is best to make the device actually driving the $\overline{VPA}$ line an open collector output so that several gates can be wire ORed.

OPERATION

Operating the ACIA is relatively easy as shown in the flow chart given in Figure 2. Once the control register is set up, the status register is monitored for receive data register full (RDRF) and transmit data register empty (TDRE) indications, as well as error signals and handshake lines. The handshake lines such as request to send ($\overline{RTS}$), clear to send (CTS), and data carrier detect ($\overline{DCD}$) indicate which conditions are present so that the MPU can ascertain when transmission can occur. Once all conditions are ready, transmission or reception or both can begin.

A sample program is given in Figure 3 that shows the MC68000 receiving a character from a terminal through the ACIA and then echoing that character back to the terminal. Essentially, the MC68000 checks to see that transmission and reception can occur. The status register is polled until a character is received. The character is read and then written back to the ACIA for transmission to the terminal as soon as the transmit data register is empty. Of course, any number of subroutines or additional code could be executed before looking for the next character from the ACIA.

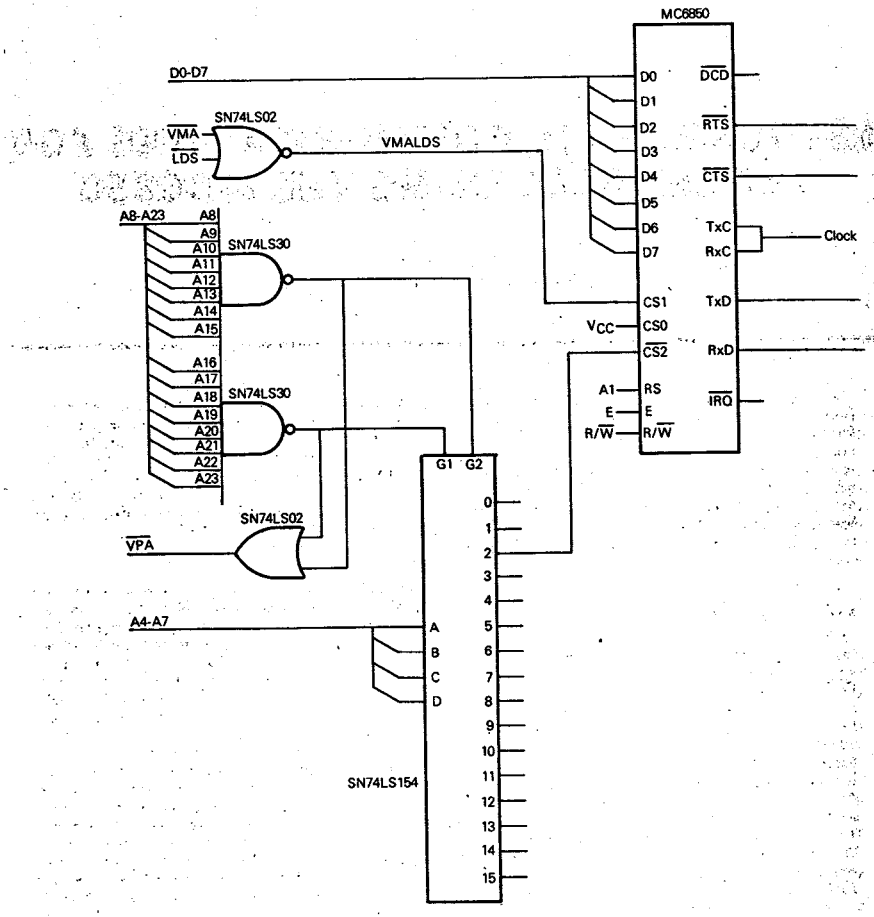


Figure 1. MC6800 to MC6850 Interconnections

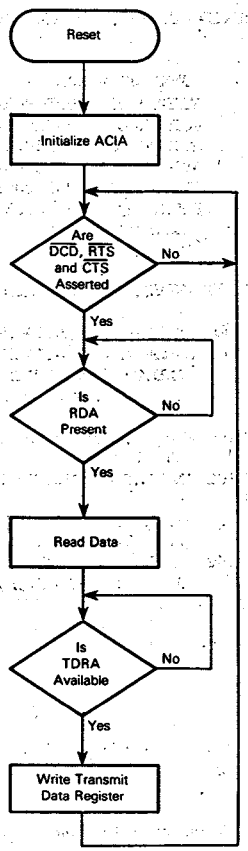


Figure 2. ACIA Operation — Flow Chart

```
ACIABUG:  MC68000 ASM REV= 1.1 - COPYRIGHT BY MOTOROLA 1978          PAGE  1

   1          00000000          ORG $00000000
   2          00F3FF00          ACIASR EQU $00F3FF00
   3          00F3FF00          ACIACR EQU $00F3FF00
   4          00F3FF02          ACIADR EQU $00F3FF02
   5          00F3FF02          ACIATR EQU $00F3FF02
   6          00020000          SYSTACK EQU $00020000
   7          00000008          RESET EQU $00000008
   8 000000 00020000            DC.L SYSTACK
   9 000004 00000008            DC.L RESET
  10 000008 13FC0003
                00F3FF00        MOVE.B #$03,ACIACR RESET ACIA
  11 000010 13FC0051
                00F3FF00        MOVE.B #$51,ACIACR INITIALIZE ACIA
  12 000018 103900F3FF00 ERROR MOVE.B ACIASR,D0 GET STATUS
  13 00001E 0200007C            AND.B #$7C,D0 MASK IRQ,TDRA,RDA
  14 000022 66F4                BNE ERROR ANY ERRORS?
  15 000024 08390001
                00F3FF00 READS1 BTST #01,ACIASR
  16 00002C 66F6                BNE READS1
  17 00002E 103900F3FF02        MOVE.B ACIADR,D0 READ CHARACTER
  18 000034 08390002
                00F3FF00 READS2 BTST #02;ACIASR IS TDRA SET?
  19 00003C 66F6                BNE READS2 LOOP IF NO
  20 00003E 13C000F3FF02        MOVE.B D0,ACIATR TRANSMIT CHARACTER
  21 000044 60D2                BRA ERROR START OVER
  22                            END

****** TOTAL ERRORS   0--   0


SYMBOL TABLE

ACIACR   F3FF00 ACIADR   F3FF02 ACIASR   F3FF00 ACIATR   F3FF02
ERROR    000018 READS1   000024 READS2   000034 RESET    000008
SYSTACK  020000
```

Figure 3. ACIA Operation — Sample Program

SYNCHRONOUS I/O FOR THE MC68000 USING THE MC6852

Prepared by:
James McKenzie
Microprocessor Applications Engineering

The MC6852 Synchronous Serial Data Adapter (SSDA) provides both a synchronous serial transmitter and synchronous serial receiver in a single, 24-pin device. Synchronous data communications is inherently more efficient than asynchronous data communications because each character need not be framed for error detection. Hence, synchronous data communications lends itself to higher data rates and applications which are synchronous in nature, such as serial communications between synchronous processors.

The SSDA is particularly well-suited for data communications applications involving byte-oriented protocols such as Bisync. Both the SSDA transmitter and receiver are interfaced to a single 8-bit bidirectional data bus. Data to be transmitted is loaded from the MPU data bus into a 3-byte FIFO on the SSDA. An 8-bit shift register is used to serially transmit data from the last FIFO location; parity may also be appended. Received data enters another 8-bit shift register where parity may be checked. Data from the shift register enters a 3-byte receiver FIFO which presents the data in parallel form to the MPU bus.

The SSDA has five write-only registers which allow software selection of variables such as transmit/receive word format, mode of synchronization, separate interrupt control configuration for transmitter and receiver, individual software reset for transmitter and receiver, and access to the transmitter data FIFO. Two read-only registers allow access to receiver data as well as a status register which has flags for the transmitter/receiver interrupts, transmitter/receiver error conditions and external sync control line status.

Any series product of the MC6852 may be interfaced to the MC68000 as shown in Figure 1. Typical timing diagrams for this interface (B part only) appear in Figure 2.

ASYNCHRONOUS OPERATION

Address Buffering and Decoding — Buffers U1, U2 and U3 buffer address lines A1 through A16, as well as $\overline{AS}$ and R/$\overline{W}$. This scheme is compatible with the MEX68KDM Design Module. All pin numbers shown on the left side of Figure 1 are for MEX68KDM/EXORciser bus pin allocations. Other forms of buffering/termination may be used to suit the user's configuration. Gates U9, U11A, U10E, and U11B decode the address lines for address block $18000 to $1801F. The SSDA is located at $18009 (mirrored at $1800D) and $1800B (mirrored at $1800F).

E Synchronization and RS Control — The continuous E signal which M6800 family peripherals require for operation will, in general, be asynchronous with MC68000 bus operation and, therefore, asynchronous with respect to chip select (pin 6, U11B). Flip-flop U13 serves to synchronize E with the chip select, supply chip select ($\overline{CS}$) to synchronize the peripheral part, and return $\overline{DTACK}$ to the asynchronous MC68000. Chip select (pin 6, U12B) is passed to the peripheral on the first falling edge of E past the assertion of CS (pin 6, U11B). This guarantees that there will always be sufficient setup time for the synchronous peripheral. Data transfer acknowledge (pin 8, U13B) is returned and $\overline{CS}$ removed from the peripheral on the next falling edge of E. Chip select for the peripheral is inverted by U10C and NANDed with address line A3 (pin 8, U12C) to complete the decoding and drive $\overline{CS}$ on the SSDA. Register select (RS) on the SSDA is driven by address line A1.

```
PAGE  030  CBUG05  .SA:1

01487                          *
01488A 1FC4 80                 TIRQWV RTI
01489                          *
01490A 1FC5 80                 IRQV   RTI
01491A 1FC6 80                        RTI
01492                          *
01493          1FC7     A TIRQV  EQU    *
01494A 1FC7 A6 40       A       LDA    #$40
01495A 1FC9 B7 09       A       STA    TIMEC
01496A 1FCB CD 1916     A       JSR    LOCSTK
01497A 1FCE E6 04       A       LDA    4,X
01498A 1FD0 BA 57       A       ORA    WORK6
01499A 1FD2 E7 04       A       STA    4,X
01500A 1FD4 CC 1928     A       JMP    PCOUNT
01501                          *
01502A 1FD7 CD 1E07     A PWRDWN JSR   CLRDIS
01503A 1FDA 8E                 STOP
01504                          *
01505A 1FF6                    ORG    $1FF6
01506                          *
01507A 1FF6     0046    A      FDB    TIRQW
01508A 1FF8     0043    A      FDB    TIRQ
01509A 1FFA     0040    A      FDB    IRQ
01510A 1FFC     1856    A      FDB    SWI
01511A 1FFE     1800    A      FDB    RESET
01512                          *
01513                          END
TOTAL ERRORS 00000--00000
```


AN-824

MC68000 DMA USING THE MC6844 DMA CONTROLLER

Prepared by
Arnold Morales
Microprocessor Applications Engineering

The MC6844 DMA Controller (DMAC) can be interfaced to the MC68000 microprocessor to provide flexible, low-cost, relatively high performance DMA control in an MC68000-based system. In designing such a system, three interface requirements must be considered:

1. The DMAC should operate at maximum frequency for efficient data transfer. High performance systems may require the use of the two megahertz device (MC68B44), so the system must allow the MC68000 to access the DMAC asynchronously.
2. Handshake logic must be implemented to arbitrate control of the system bus between the MC68000, the DMA control system, and other possible bus masters.
3. The MC6844 is an 8-bit device intended for use in MC68000 systems, capable of direct memory access through only a 64K memory space, and also lacks certain bus strobes necessary for simple implementation in an MC68000-based system. A bus interface must be designed to allow direct memory access throughout the entire 16 megabyte MC68000 memory map and to provide the required bus strobes needed for successful use in an MC68000-based system.

This application note describes designs to meet each of these requirements. These designs are then combined to form a direct memory access control system for the MC68000. An implementation of the complete system is presented in block diagram form using an MC6854 Advanced Data Link Controller (ADLC) and a static memory buffer.

MC6844 ASYNCHRONOUS INTERFACE OPERATION

The MC6844 can be interfaced asynchronously to the MC68000 using the circuitry presented in Figure 1. This circuit allows the MC68000 to access a DMAC driven by an E clock that is either synchronous or asynchronous to the MC68000 clock. It generates DMAC chip select at the proper time to satisfy DMAC timing requirements, latches data to satisfy data hold time requirements, and asserts data transfer acknowledge at the proper time to ensure valid data transfer between devices. This circuit can be used to interface other MC6800 peripherals, and is used to interface to the ADLC as well as the DMAC in the system implementation presented at the end of this application note.

CIRCUIT OPERATION — When the MC68000 performs a read or write bus cycle (access), the processor asserts one or both of the two data strobes ($\overline{DS}$), an address strobe ($\overline{AS}$), the read/write (R/$\overline{W}$) signal, and an address. The processor also outputs data during write cycles.

The MC68000 remains in this state until the bus cycle is terminated. Data transfer acknowledge ($\overline{DTACK}$) is asserted by the peripheral or memory device being accessed to initiate termination of the bus cycle by the MC68000.

The circuit in Figure 1 synchronizes MC68000 accesses to the DMAC with the E clock. Initially, flip-flops U1A and U1B are cleared causing a high $\overline{DTACK}$ output setting U2 and U3 into a transparent mode. Latch U2 is in the high-impedance state due to a high on the output enable ($\overline{OE}$) input. Latch U3 is enabled due to a low on the $\overline{OE}$ input.

At the start of a DMAC access, latch U3 remains enabled if the access is a write. If the access is a read, the high R/$\overline{W}$ and DMAC Select inputs to U4A cause U3 to go to the high-impedance state and U2 to become enabled. The DMAC Select signal is asserted when the DMAC is addressed. However, the DMAC is actually selected by the assertion of $\overline{CS}$ (DMAC). Flip-flop U1A is clocked high on the first falling edge of E after DMAC Select and data strobe (DS) are asserted. The Q output of U1A is applied to U4D, asserting $\overline{CS}$ (DMAC). Selecting the DMAC at this time ensures that the DMAC has adequate address setup time.

On the next falling edge of E, the $\overline{Q}$ output of U1B is clocked low asserting $\overline{DTACK}$ and latching data into the enabled latch. The asserted $\overline{DTACK}$ signal, inverted by U4D, deselects the DMAC by causing $\overline{CS}$ (DMAC) to go high. When the access terminates, flip-flop U1 is cleared by the negation of $\overline{DS}$, and the interface circuitry is initialized for the next access. The $\overline{DTACK}$ signal is buffered by an open-collector buffer (U5) to allow assertion of $\overline{DTACK}$ by other devices when the DMAC is not being accessed.

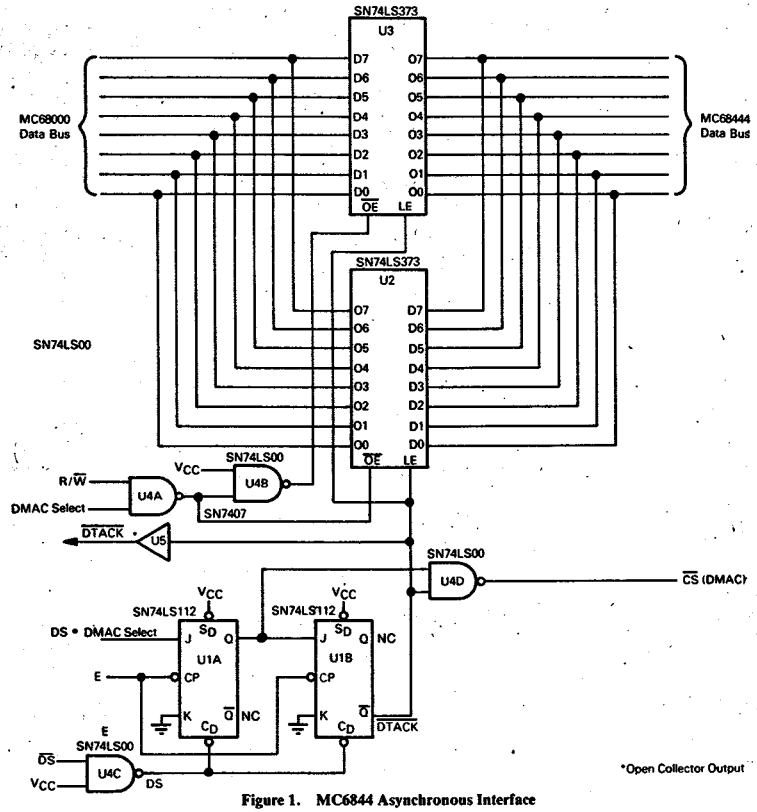


Figure 1. MC6844 Asynchronous Interface

BUS ARBITRATION INTERFACE

The MC6844 is an 8-bit, 4-channel DMA Controller capable of performing direct memory transfers of a user defined number of data bytes (data block) within a 64K byte memory space. Associated with each channel of the controller are:

- A transfer request (TxRQ) input which is asserted by a peripheral controller or a processor to request DMA service.
- A 16-bit address register which is initialized with the beginning address of the data block to be transferred.
- A 16-bit byte count register which is initialized with the desired number of data bytes (size of the data block) to be transferred.

Each channel can perform DMA transfers in one of three modes: TSC Steal, Halt Steal, and Halt Burst. Two of these modes, TSC Steal and Halt Steal, are single-byte transfer modes in which the DMAC returns control of the system bus to the processor after each transfer, while the Halt Burst mode is a block transfer mode in which the DMAC retains control of the system bus until the last byte of the data block has been transferred.

The bus arbitration circuit presented in Figure 2 is designed for the Halt Steal and Halt Burst modes of operation. The TSC Steal mode is intended for use with the MC6800 and offers no advantage over the Halt Steal mode in MC68000 applications.

In the Halt Steal mode the DMAC responds to a transfer request by asserting DMA request halt steal ($\overline{DRQH}$). The DMAC then waits until DMA grant (DGRNT), a DMAC input, is asserted. At this time, one transfer of data is initialized and transfer strobe ($\overline{TxSTB}$) is asserted, followed by the negation of $\overline{DRQH}$. This sequence is repeated until all data has been transferred.

The same sequence is followed in the Halt Burst mode with the exception that $\overline{DQRH}$ is negated only after the last byte of the data block has been transferred. In this mode, bus mastership is arbitrated once, then data transfers occur in succession until all data has been transferred.

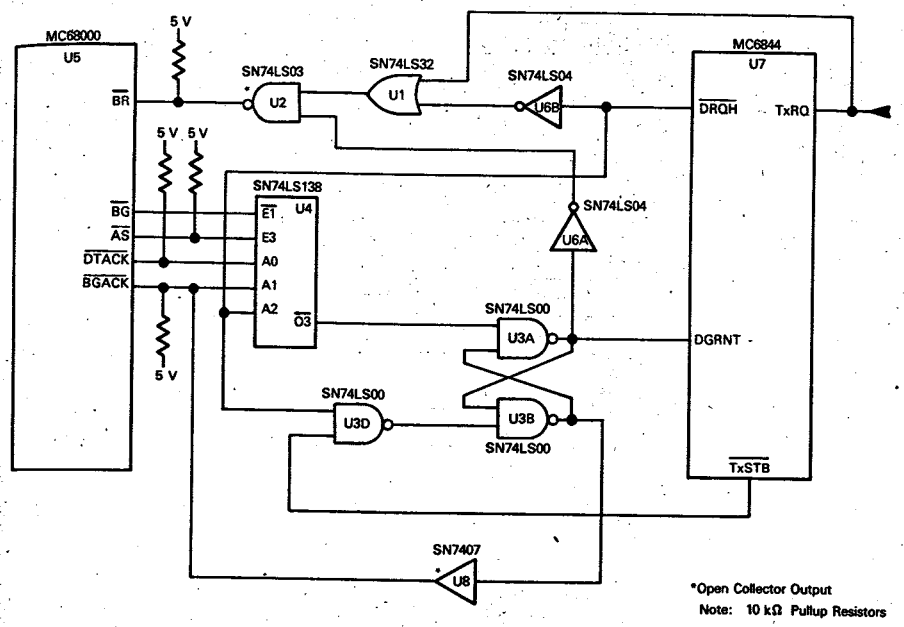


Figure 2. Bus Arbitration Interface

CIRCUIT OPERATION — For either a Halt Steal or Halt Burst DMA transfer by the control systems presented in this application note, three conditions must be met:

1. Transfer request (TxRQ) must be asserted.
2. $\overline{DRQH}$ must be asserted.
3. All bus masters must have relinquished the bus to ensure that DMA grant (DGRNT) is asserted.

Initially DGRNT is low, bus grant acknowledge ($\overline{BGACK}$) is not asserted by the interface, and $\overline{TxSTB}$ is high. The DMAC responds to a transfer request by asserting $\overline{DRQH}$. Once $\overline{DRQH}$ is asserted, it remains asserted until the DMAC performs a byte transfer in the Halt Steal mode or until the last byte of a designated memory block is transferred in the Halt Burst mode.

Transfer request (TxRQ) is coupled through U1 and U2 so that MC68000 bus request ($\overline{BR}$) is asserted when TxRQ is asserted. By requesting a DMA transfer and bus arbitration simultaneously (disregarding gate propagation delay), DMA latency time is minimized. The MC68000 responds to a bus request by asserting bus grant ($\overline{BG}$) and relinquishing the bus.

When DRQH is asserted and all bus masters are off the system bus, indicated by the negation of $\overline{AS}$, $\overline{DTACK}$, and $\overline{BGACK}$, flip-flop U3A-U3B is set by the assertion of the $\overline{O3}$ output of U4. The setting of flip-flop U3A-U3B asserts $\overline{DGRNT}$ to initiate DMA transfer(s), and also asserts $\overline{BGACK}$ to keep other bus masters off the bus. Bus grant ($\overline{BG}$) is negated by the MC68000 soon after $\overline{BGACK}$ is asserted.

Flip-flop U3A-U3B is cleared on the rising edge of $\overline{TxSTB}$ after it is asserted during each DMA cycle in the Halt Steal mode, and during the last cycle of a block transfer in the Halt Burst mode. Clearing flip-flop U3A-U3B negates $\overline{BGACK}$ to release the system bus, and negates DGRNT to stop DMAC transfer activity.

The MC68000 $\overline{BR}$ and $\overline{BGACK}$ signals are driven by open collector gates to allow other devices to also request the system bus. A pullup resistor is used to hold $\overline{AS}$ in the negated state during transitions in bus ownership.

BUS INTERFACE REQUIREMENTS

A general direct memory access controller for an MC68000-based system must allow direct memory access throughout the entire 16 megabyte memory map of the MC68000. In addition, it must assert the appropriate data strobe(s) and an address strobe. The MC6844 does not satisfy these requirements; therefore, TTL devices must be used to meet these needs.

The MC68000 can perform three types of data transfers: word transfers (D0-D15), byte transfers to/from lower data bytes (D0-D7), and byte transfers to/from upper data bytes (D8-D15). When transferring a byte, the MC68000 asserts either the upper data strobe ($\overline{UDS}$) or the lower data strobe ($\overline{LDS}$), depending on whether an upper or a lower data byte is being transferred; and when transferring a word, it asserts

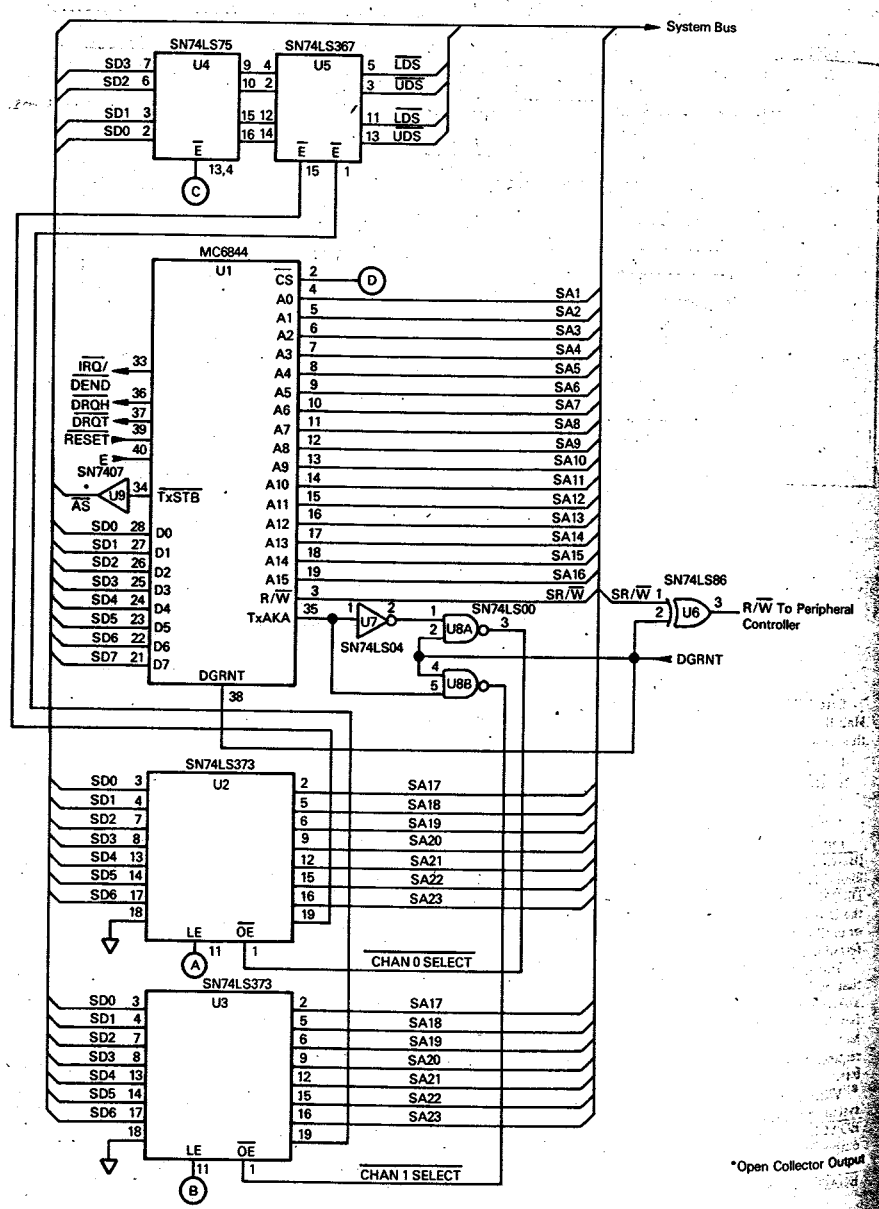


Figure 3. MC68000 Word and Non-Sequential Byte Transfer Interface System

both $\overline{UDS}$ and $\overline{LDS}$. The MC68000 asserts $\overline{AS}$ during each type of transfer.

The following are general designs which can be modified to meet individual system requirements. The two designs presented differ in the types of transfer they perform.

Only two of the four DMAC channels are used in each design. However, these interfaces can be easily modified for four-channel operation.

WORD AND NON-SEQUENTIAL BYTE TRANSFER INTERFACE SYSTEM

An MC68000 DMA control system capable of word transfers and byte transfers to/from upper byte or lower byte memory locations is presented in Figure 3.

In this system, address lines A0-A15 from the DMAC are connected to MC68000 system address lines SA1-SA23 and as the DMAC address lines increment or decrement (according to user option), the system address is incremented/decremented by words, rather than bytes; that is, the system address changes in increments of two bytes.

The system upper address lines SA17-SA23, are latched into transparent latches U2 and U3 during initialization, which are enabled during a DMA transfer. Latch U2 is the channel 0 upper address latch, with its chip select labeled A; latch U3 is the channel 1 upper address latch, with its chip select labeled B. During a direct memory access, transfer acknowledge A (TxAKA) from the DMAC is asserted during channel 1 transfers, and negated during channel 0 transfers. This DMAC output is used to enable the proper address latch during a direct memory access.

The type of direct memory access transfer (word or byte) is determined by the state of latch U4 during the access. Latch U4 with its chip select labeled C, is connected to system data bus lines SD0-SD3 and, through three-state buffer U5, to system data strobes $\overline{LDS}$ and $\overline{UDS}$. When writing to latch U4 during initialization, the states of SD2 and SD3 determine the states of the data strobes during a channel 1 direct memory access, and the states of SD0 and SD1 determine the states of the data strobes during a channel 0 direct memory access. For word transfer both of the data strobes must be asserted, while for byte transfers either the $\overline{LDS}$ or $\overline{UDS}$ is asserted, depending on whether a lower data byte (D0-D7) or an upper data byte (D8-D15) is being transferred.

Note that in memory organized in 16-bit words, byte transfers are to/from either the upper byte or the lower byte of memory during each DMA block transfer.

During a direct memory access the appropriate U4 latch states are gated onto the system bus by U5. The appropriate U5 buffers are enabled by latch U2 during channel 0 access, and by latch U3 during channel 1 access.

When $\overline{DGRNT}$ is asserted, the R/$\overline{W}$ signal to the peripheral controller is inverted by exclusive OR gate U6.

Transfer strobe ($\overline{TxSTB}$) is fed through an open collector buffer to the system $\overline{AS}$ line. During a direct memory access transfer the $\overline{AS}$ output of the MC68000 is in the high-impedance state and $\overline{TxSTB}$ is used as the system address strobe. Transfer strobe is asserted by a DMAC operating at 2 megahertz for at least 370 nanoseconds to indicate a valid address during a direct memory access, and may require conditioning for use as an address strobe during direct memory access in some systems.

SEQUENTIAL MEMORY BYTE TRANSFER INTERFACE SYSTEM

An MC68000 DMA control system capable of byte transfers to/from sequential memory locations in a memory organized in 16-bit words is presented in Figure 4. In this system, address lines A1-A15 from the DMAC are connected to MC68000 system address lines SA1-SA15. Address line A0 of the DMAC is connected to inverting buffer U4. Buffer U4 is enabled by DGRNT to generate the data strobes. Only one data strobe is asserted at a time. Each time the DMAC increments/decrements, the state of $\overline{UDS}$ and $\overline{LDS}$ alternate. System address line SA1 changes state only after each data strobe is asserted for one DMA cycle and negated for one DMA cycle. By doing this, data is transferred to/from consecutive byte locations in the word-dimensioned memory map.

Latches U2 and U3 latch upper system address lines SA16-SA23 during initialization and their operation is identical to the circuit presented in Figure 3.

COMPLETE SYSTEM IMPLEMENTATION

A block diagram of a complete MC68000 DMA system using the MC6844 DMAC for controlling DMA between an MC6854 ADLC and a block of memory is presented in Figure 5. Data transfer in this system is between the ADLC and lower memory byte locations (D0-D7).

The ADLC asserts receiver data service request (RDSR) each time the receiver FIFO register requires servicing, and transmitter data service request (TDSR) each time the transmitter is ready for data. These outputs are tied to transfer request channel 0 (TxRQ0) and transfer request channel 1 (TxRQ1) of the DMAC so that DMAC channel 0 services the ADLC receiver, and DMAC channel 1 services the ADLC transmitter.

The block labeled "MC6854 Register Select, R/W Control" is used to address the ADLC transmit or receive register and to invert the read/write signal during a direct memory access. This circuit puts the address bus from the ADLC in the high-impedance state during the direct memory access and forces ADLC register select zero (RS0) to a low state and register select one (RS1) to a high state, so that during a direct memory access either the transmit FIFO register or the receiver FIFO register is selected according to the state of the R/$\overline{W}$ signal. The circuit uses DMAC $\overline{DEND}$ to select the frame terminate register of the ADLC during the last byte of a DMA block transfer when servicing the transmitter FIFO. During a direct memory access, the ADLC is selected by assertion of $\overline{TxSTB}$ to ensure that the ADLC is selected only during valid direct memory access cycles.

System memory is connected directly to the system bus. The "Memory $\overline{DTACK}$ Gen." consists of a counter driven by the MC68000 clock, and enabled by an asserted address strobe when memory is accessed by the processor. Data transfer acknowledge ($\overline{DTACK}$) is "picked off" one of the counter pins so that it is asserted at some preset time interval after memory is accessed.

Memory address decoding is the same for both direct memory access and processor data transfers. However, during a direct memory access, memory is deselected by the NOR or $\overline{DGRNT}$ and E. This ensures that, during a direct memory access, the memory will latch written data at the fall of E, when ADLC data is valid.

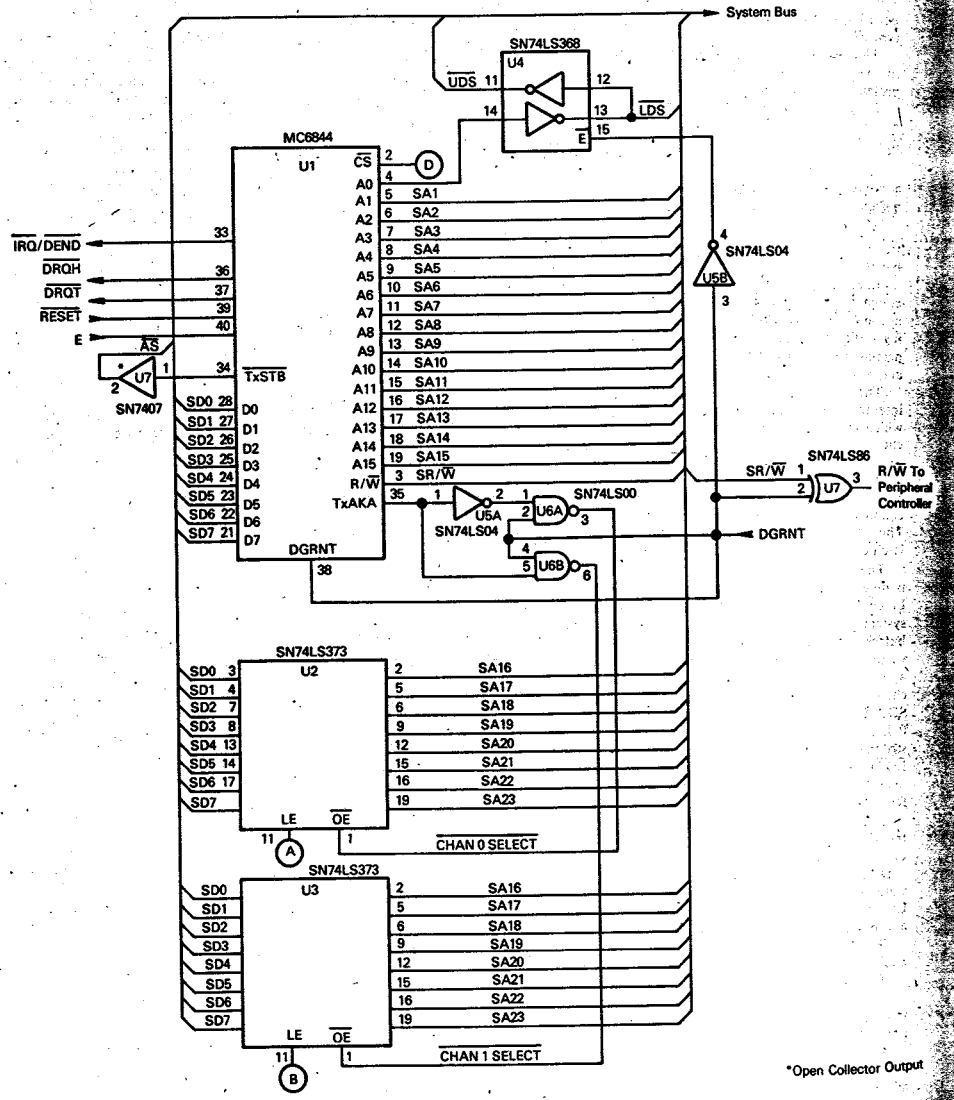


Figure 4. MC68000 Sequential Memory Byte Transfer Interface System

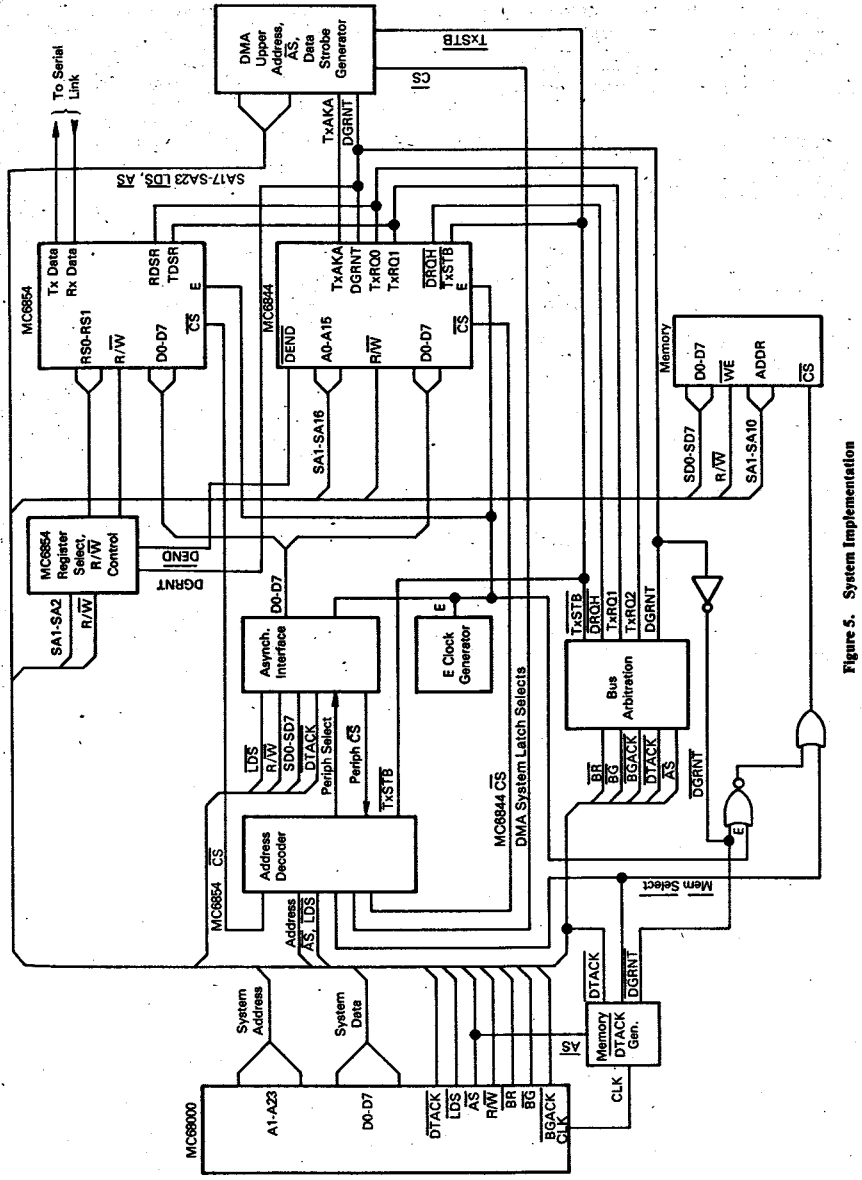


Figure 5. System Implementation

ADDITIONAL SYSTEM ENHANCEMENTS

Two enhancements to the direct memory access control systems presented in this application note should be considered. One improvement increases ADLC throughout, and the other allows memory to memory DMA data transfers.

THROUGHPUT ENHANCEMENT — Worst-case DMA latency of the systems described in this application note are 1.18 microseconds for MC68000 systems that do not implement the Read-Modify-Write instruction, and 1.68 microseconds for systems that do implement the instruction. This is the worst-case delay between assertion of TxRQ and the beginning of the direct memory access cycle to service the channel, and allows for propagation delay through the gates in the bus arbitration handshake logic. These times assume 8 megahertz processor operation, and 2 megahertz controller operation.

The ADLC service latency can be reduced by designing a FIFO buffer to handle data transfers between the ADLC and the rest of the system. In this technique, the FIFO buffer services the ADLC, and direct memory access transfer is between the FIFO buffer and system memory.

MEMORY TO MEMORY DMA — The direct memory access designs presented in this application note can be easily modified to perform memory to memory data transfer.

The DMAC will perform a direct memory access transfer for each cycle in the Halt Burst mode while TxRQ is asserted, until the block transfer is complete. In this way, an MC68B44 clocked at 2 megahertz can perform a direct access at a 2 megahertz rate. For memory to memory transfer, all that is needed is to allow one memory block to be addressed directly by the DMAC during direct memory access, and transpose the address to access the other memory block. During a direct memory access, the DMA R/$\overline{W}$ signal to one of the memory blocks must be inverted so that during each direct memory access cycle data is read from one memory location in one memory block, and is written into another location in the other memory block.

AN INTELLIGENT TERMINAL WITH DATA LINK CAPABILITY

By
Charles Melear
Microprocessor Applications Engineer

INTRODUCTION

A small but powerful terminal complete with high speed data link can be constructed with a minimum number of NMOS LSI circuits. Operating systems can be developed to make this terminal act as a word processor, point-of-sale terminal, data input source, etc. The data link capability allows the operator to call in the resources of remote computers at synchronous serial data rates of up to 1.5 megahertz.

Five devices form the core of the terminal as shown in Figure 1. An MC6809 Microprocessor (MPU) was chosen because of its many hardware and software features.

The MC6845 CRT Controller (CRTC) permits the use of a video display monitor. This controller was chosen because it allows complete software control of the video display monitor. Vertical sync delay, horizontal sync width and delay, blanking, number of characters-per-row, and rows-per-screen are all programmable.

Serial keyboard input capability is provided by an MC6850 Asynchronous Communications Interface Adapter (ACIA) which performs the serial/parallel conversions. This application polls the ACIA to check for a data present indication instead of using an interrupt. This polling method provides the highest priority and shortest response time to the high speed data link.

High speed data link capabilities are provided by an MC6854 Advanced Data Link Controller (ADLC). The ADLC detects the start of a message, receives the message, calculates and appends a CRC character, and provides a closing flag. Serial data rates of 1.5 megahertz are possible with this system. To operate at these speeds, direct memory access capability is needed and is provided, in this application, by an MC6844 Direct Memory Access Controller (DMAC). A data transfer can be processed every four bus cycles when an MC6854 ADLC and an MC6844 DMAC are used together.

MC6845 CRT CONTROLLER (CRTC)

The CRTC provides horizontal sync, vertical sync, and blanking to a video display monitor along with the memory address of the data to be displayed. A cursor output is also provided. Once the CRTC is initialized, it performs the function of controlling the video display monitor without intervention by the processor. Initialization is accomplished by writing the appropriate values into the 16 programmable registers. Figure 2 Sheet 1 is a worksheet which can be used to collect the information required to calculate the values needed for the CRTC register worksheet given in Figure 2 Sheet 2. It is assumed that the video display monitor uses a 60 hertz power source and a 15,750 hertz horizontal oscillator frequency. After initialization, the CRTC starts with the address located in the start address register. The ASCII character represented by the hexadecimal value at that location will appear in the upper left-hand corner of the video display monitor. The CRTC advances the memory address lines by one with each character clock. The first row will contain the number of characters specified in the horizontal display register.

Due to synchronization problems between the CRT clock and several other signals, it is possible that the first character could be only partially displayed. Figure 3 shows how this can happen because the time between the CRT clock and display enable (Tx) is an internal function of the CRTC. The first character will be partially displayed because display enable goes high approximately in the middle of the first character. This problem can be resolved by writing an ASCII blank (20) at the first character location and using the second character location to display the first character.

The screen memory must be accessible to the processor for updating. Since the CRTC memory address lines normally drive the screen memory, multiplexers are used to select either the CRTC memory address lines or the processor address lines. A decoding network selects the processor address lines any time an address between $0000 and $1FFF is detected. The data bus for the screen memory is isolated from the processor data bus by SN74LS243 transceivers. These devices are normally in the high-impedance state in both directions except during a processor read or write of the

AN-834

USING THE MC68000 AND THE MC6845 FOR A COLOR GRAPHICS SYSTEM

By
David L. Ruhberg
Microcomputer Systems Engineer
Motorola Semiconductor

Probably the slowest link in most computerized control systems is the display of information for human interpretation. The commonly used black and white monitor can display an adequate amount of information in most cases.

In applications where a large amount of information must be displayed in the same screen area, a color graphics system can easily provide this information by using a wide range of contrasting colors. Until recently the high cost of sophisticated components and color monitors required to generate and display color information has probably been the main prohibitive factor in development of these systems.

Recently the cost of components and color monitors has moderated to the point that using a color graphics system offers a viable solution to information display, ranging from the video games market to complex control systems.

A state-of-the-art color graphics system using the MC68000 16-bit microprocessor (MPU) with an economical MC6845 CRT controller (CRTC) is described in this application note. Hardware improvement is evident in data movement occurring in 16-bit words and multiply and divide commands while software compatibilities are greatly enhanced through the use of a processor that executes instructions which can operate on 8-, 16-, or 32-bit operands.

The general approach to a color graphics system is straightforward and almost identical to a black and white graphics system. A typical black and white graphics system is shown in Figure 1. The MPU has two responsibilities to the graphics system: first, to initially program the CRTC, and second, to transfer data to the display RAM.

Once the clock circuitry is running, the CRTC is initialized and the address lines to the dislay RAM begin incrementing sequentially. As this occurs, the appropriate data from the display RAM is loaded into the shift register and then gated out serially by the dot clock input to the shift register. The display monitor then interprets the data as either turning a particular pixel on or off.

A color graphics system (Figure 2) uses the same principle as the black and white system except that it has to control three color guns (red, green, and blue) instead of just one. Therefore, there is an increase in the amount of hardware involved, but not in complexity. The software becomes more involved due to the fact that more information is being handled and displayed. The basic display system works on the principle that three bits (one for each color) controls each pixel instead of just one as in a black and white system. If two guns are on, the resulting color is a combination of the two. If all guns are on, white is the result. With this configuration a total of eight colors, including black and white, are available. Since the three bits needed to control a pixel do not fit into an eight-bit byte evenly, the unused bits could be used to obtain more colors or some other function. In addition, color systems usually require a separate sync input.

The versatility of the internal architecture of the MC68000 (Figure 3) enhances the effectiveness of the color graphics system. Besides containing a 32-bit program counter yielding 16 megabytes of direct addressing range, the MC68000 also contains eight 32-bit data registers (D0-D7) and seven 32-bit address registers (A0-A6). The eight data registers are used for byte (8-bit), word (16-bit), and long word (32-bit) data operations. The seven address registers and the stack pointer may be used for word and long word address operations. In addition, all address and data registers may be used as index registers.

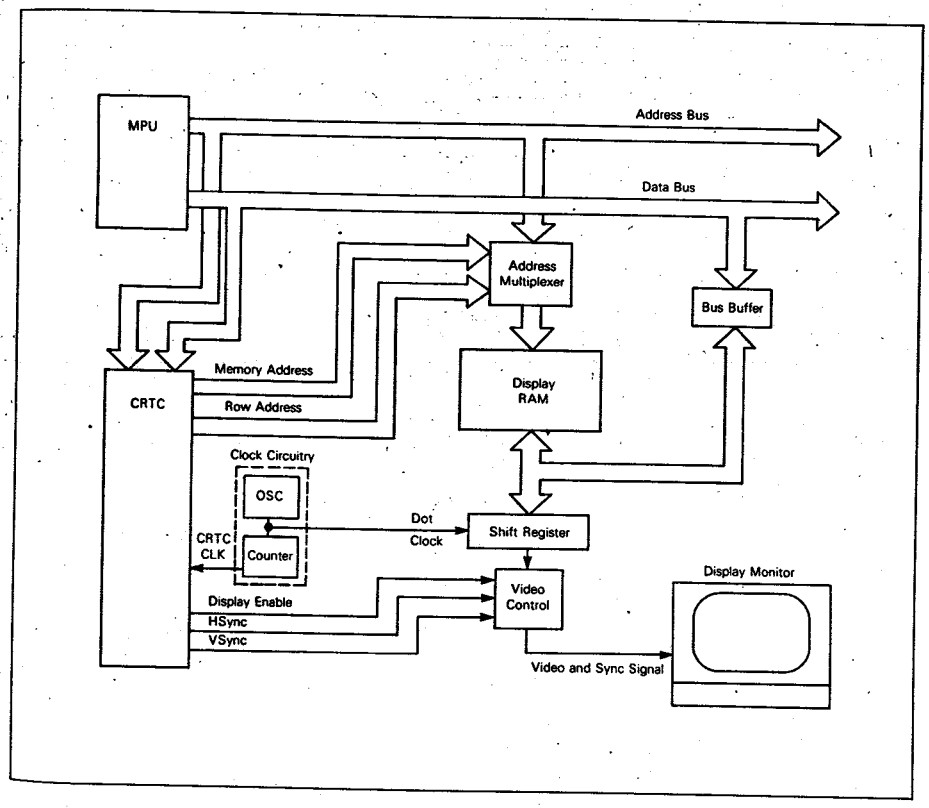


Figure 1. Black and White Graphics System — Block Diagram

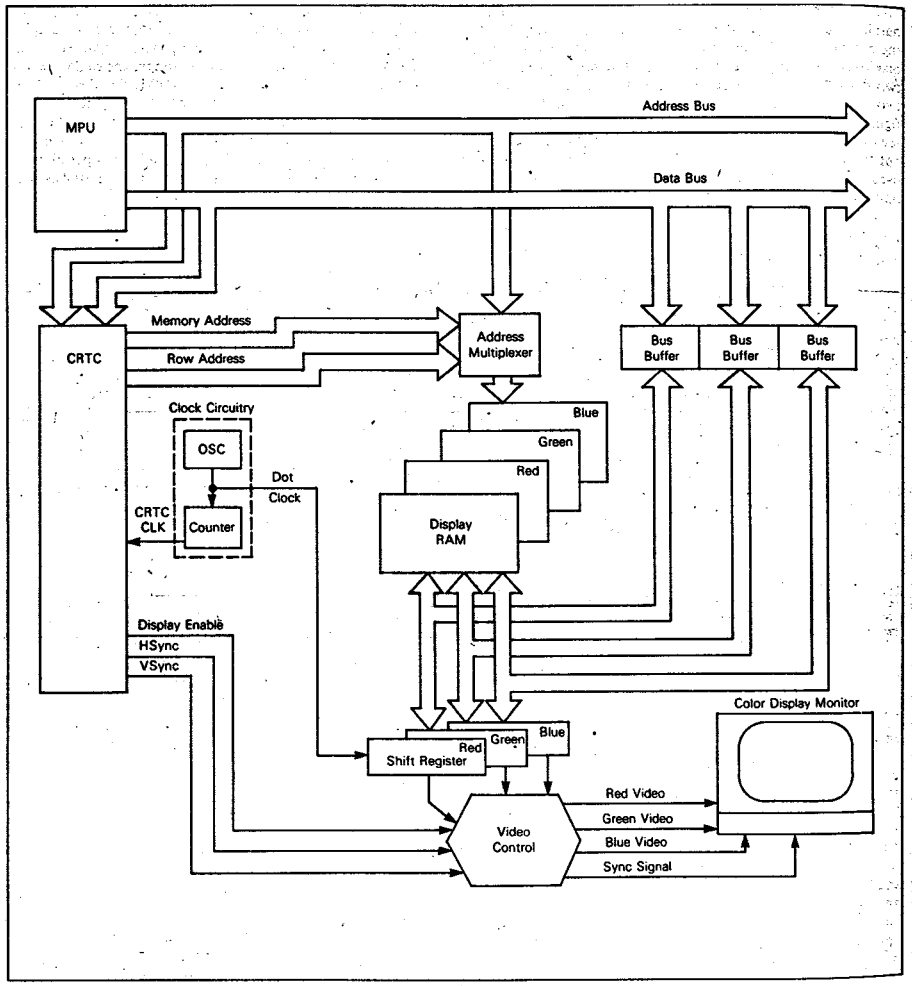


Figure 2. Color Graphics System — Block Diagram

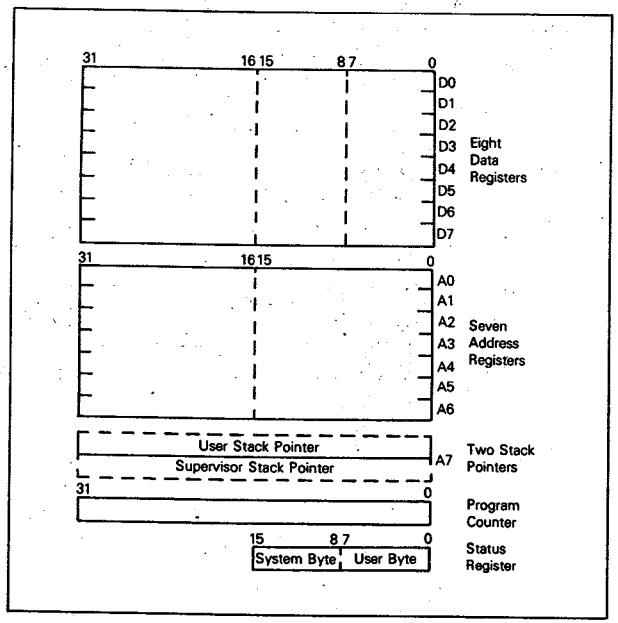


Figure 3. MC68000 Programming Model

SYSTEM HARDWARE DESCRIPTION AND FEATURES

This graphics system consists of two boards: a CPU board and a video board. The CPU board contains the processor, scratch-pad RAM, stack RAM, the program EPROM, and a terminal interface. The video board contains the CRTC, display RAM, multiplexers and buffers, parallel-to-serial shift registers, and the D/A drivers for the color display monitor.

An MC68000 Design Module (MEX68000KDM) is used as the CPU board. The resources available on the MC68000 Design Module allow more design time to be spent on the unique features of the system. The major portions of the system provided by the Design Module are the MPU (MC68000), the address decoding for the EPROM, a terminal interface, and all the software functions provided by the resident monitor (MACSbug). Included in the MACSbug is a transparent down-load feature which allows the system to communicate through the terminal to another system. The other system can provide the access to the floppy disks needed by this color graphics system for saving a full screen of data at a time.

The video board (Figure 4) contains more of the unique hardware features of the color graphics system. The video board can be separated into seven areas: the clock circuit, CRT controller, the DTACK circuit, the bus multiplexers and buffers, the display RAM, the shift registers, and the D/A converter drivers.

The clock circuit generates the five timing signals used throughout the video board; they are: a dot clock, a CRTC clock, a 2X dot clock, a shift register load, and a $\phi 2$ signal. The dot clock is used to drive the serial shift registers. The CRTC clock is used to drive the CRTC. The 2X dot clock and the shift register load are gated together to generate the parallel load (PLOAD) and chip select (PCS) signals for the shift registers and display RAM, respectively. The $\phi 2$ signal is also used to control accesses to the display RAM. A timing diagram of these signals is shown in Figure 5.

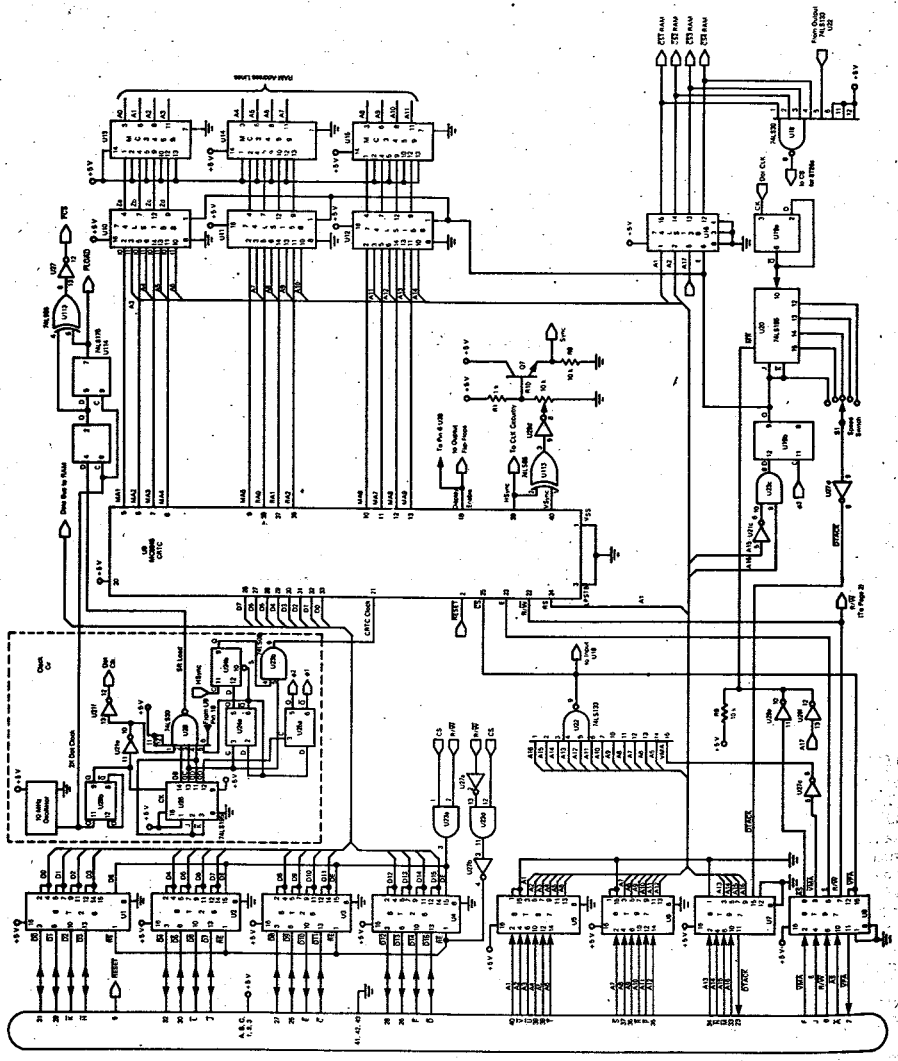


Figure 4. Color Graphics System Schematic (Sheet 1 of 3)

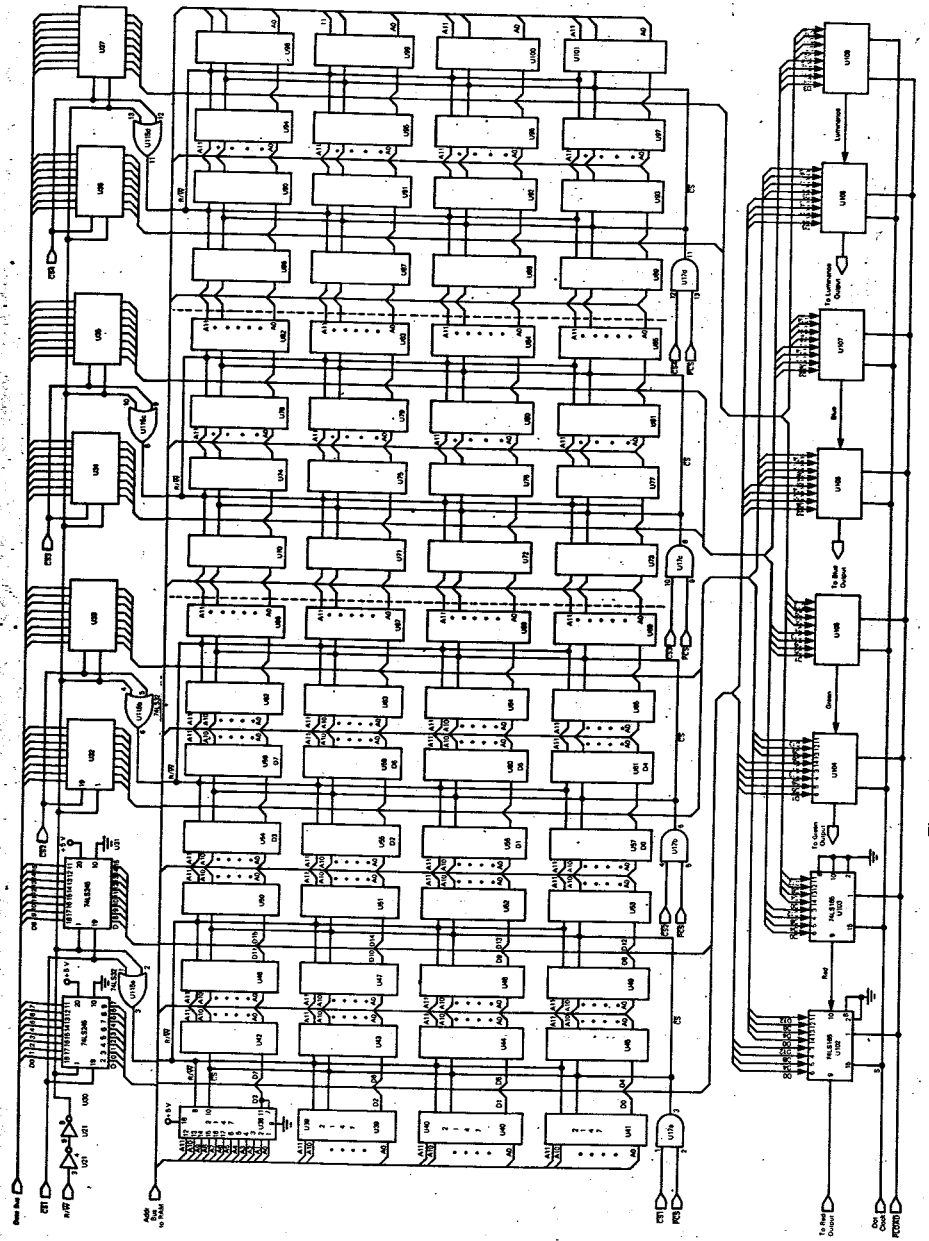


Figure 4. Color Graphics System Schematic (Sheet 2 of 3)

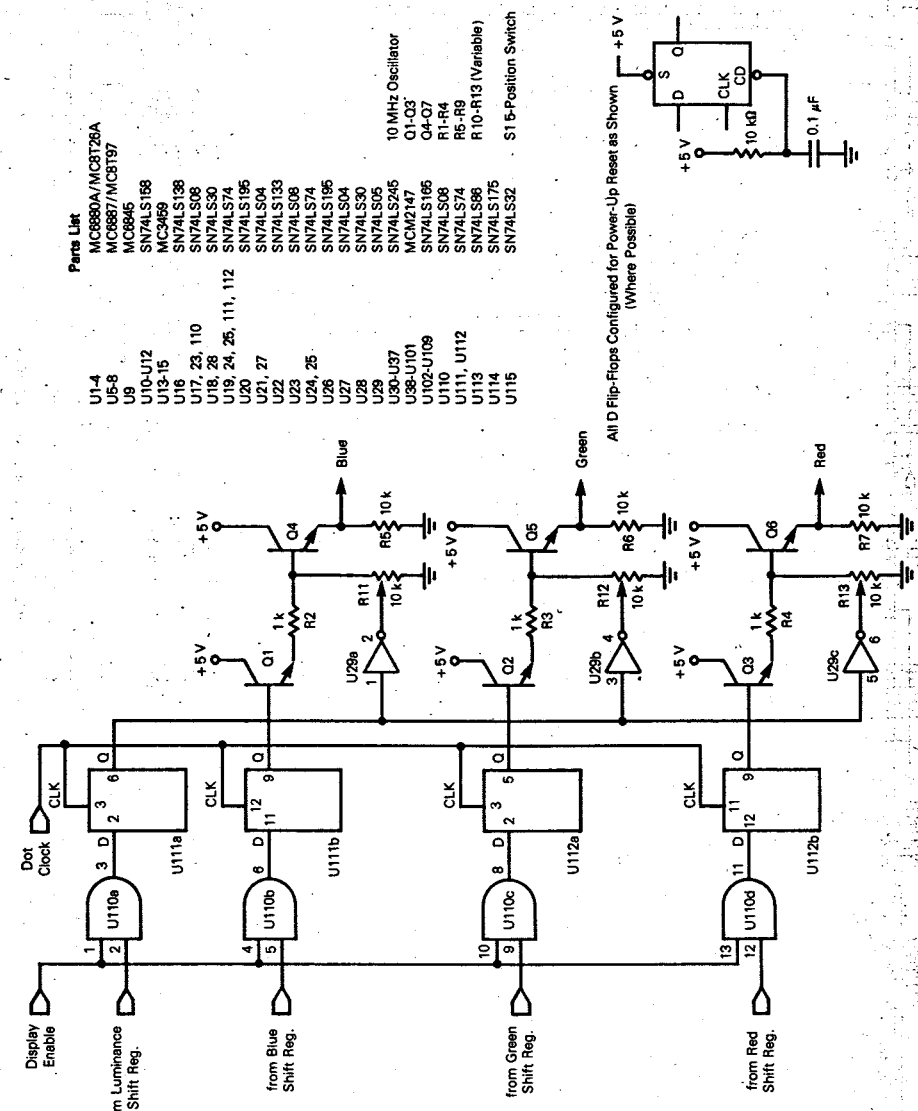


Figure 4. Color Graphics System Schematic (Sheet 3 of 3)

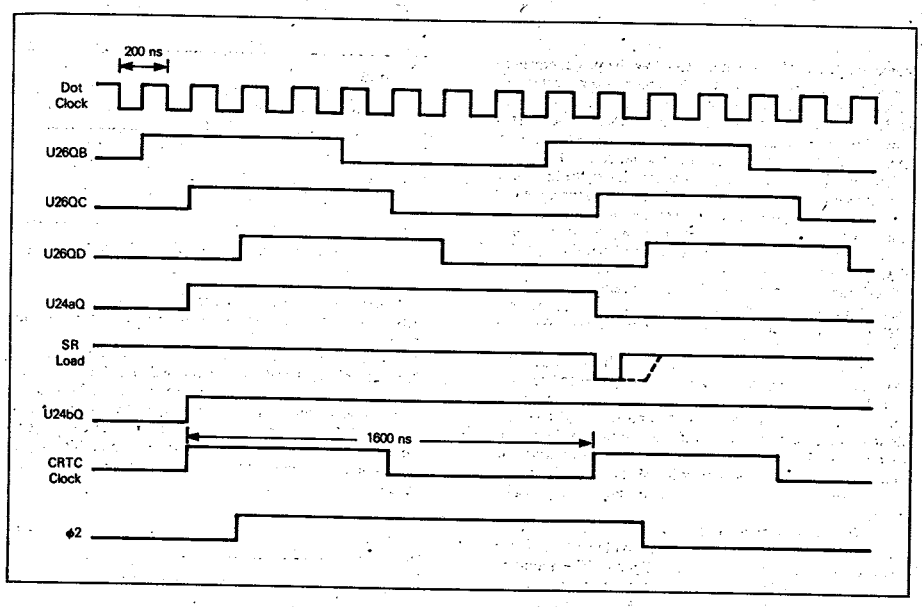


Figure 5. Clock Circuitry Timing Signals

The MC6845 CRT controller (CRTC) is a programmable controller used to prepare the information in the display RAM for use by a video display monitor. The CRTC generates the signals required to provide data at the appropriate times. Since the length and period between these signals varies from system to system, the CRTC is designed to be programmed by an MPU. In this system the internal registers are accessible synchronously through hex ($) address locations $1FFFD and $1FFFF. After programming, the CRTC provides the addresses, horizontal and vertical sync signals, and the display enable signal to the display system. The addresses, output by the CRTC in conjunction with the parallel chip select ($\overline{PCS}$) signal, are responsible for the correct data getting to the serial shift registers at the correct time. The horizontal and vertical sync signals, after being "exclusively ORed," generate the sync signal required by the color display monitor. The display enable (DE) signal is gated (U28) into either the clock circuitry to inhibit the parallel load and $\overline{PCS}$ signals or is gated (ANDed at U110, if a low represents black on the screen) with the data stream to keep the guns in the CRT off during vertical and horizontal retrace. In some cases, DE must be delayed due to specific requirements of the CRT being used. A one-shot on the output of the DE pin is usually more than adequate for providing the delay.

The $\overline{DTACK}$ circuitry is used to return an asynchronous data transfer acknowledge ($\overline{DTACK}$) signal to the MC68000 from a synchronous device (the display RAM). The $\phi 2$ signal from the clock circuitry in conjunction with address lines A15 and A16 develop the $\overline{DTACK}$ response required by the MC68000. When the display RAM address is between $10000-$17FFF, the $\overline{DTACK}$ signal is returned in 400 nanosecond increments from zero up to 1600 nanoseconds after the enabling signal goes out to the multiplexers. This time is selected by the RAM speed switch, S1. Returning $\overline{DTACK}$ to the processor is the asynchronous access method by which the MC68000 can access external devices (RAM, ROM, and peripherals). This access method was chosen over the synchronous access method used to address the CRTC because it is faster and, since this is a highly repetitive operation, any time saved here will be significant in the overall speed of the system. The synchronous access method is used to access the CRTC since the CRTC is only initialized once and this method uses fewer components.

The multiplexers and buffers are used to feed the various control signals to the rest of the system. Multiplexers U10, U11, and U12 determine which address bus will access the display RAM. When the control signal is high, the MC68000 has access to the RAM and when low, the CRTC has access. Buffers U13, U14, and U15 are used to drive the large number of devices on the address bus. Data buffers U30-U37 are used to isolate the four banks of RAM from each other. Buffers are also used for almost all the signals coming onto the video board. These board buffers interface with the modified EXORciser bus which the Design Module uses. This bus has only sixteen address lines coming from the Design Module, so address line A17 must be run separately to keep the display RAM from being accessed at the same time MACSbug or the controller program is accessed (addresses $20000 and $22000).

The display RAM is organized into four banks (red, green, blue, and luminance). However, the address lines are configured so that consecutive words are located in consecutive

banks of RAM. This was done to allow the programmer to visualize accessing one 16-bit wide bank at a time instead of accessing red, green, blue, and luminance banks all at the same time. The memories used are 4K×1 static RAMs (MCM2147) which simplify some of the chip select circuitry. Dynamic RAMs could be used and should definitely be considered in a production system since they lower the hardware cost as well as power consumption. They were omitted in this application to simplify the system configuration. It should be noted that the CRTC keeps incrementing its address lines during horizontal and vertical retrace to keep the dynamic RAM refreshed. The speed of the static memories is not critical due to the presence of the speed selection switch explained earlier. As far as the CRTC and the serial shift registers are concerned, the memory looks like one 4K×64-bit bank of RAM.

Shift registers U102-U109 consist of eight 8-bit, parallel-load, serial shift registers. They are configured to look like four 16-bit shift registers, one for each of the color guns and one for luminance. With the RAM and shift registers configured in this fashion, the RAM is accessed only 25 percent of the time. This means that the RAM has four times the amount of setup time and slower RAM can be used. The dot clock then clocks the data out to be gated with display enable.

Conversion from digital to analog voltages in this system is needed because a luminance bit is used to obtain more colors than are possible with the three guns digitally. The luminance bit is used to indicate half luminance when set and full luminance when clear. When all guns are off, the screen is black and the state of the luminance bit has no effect. Since the color display monitor uses an analog input on each gun, any number of colors may be obtained if the supporting hardware is provided. The D/A conversion used in this system was done to save space. A cleaner method would be to use special D/A converters and special line drivers for this function.

SOFTWARE DESCRIPTION AND CONSIDERATIONS

The software included to exercise this system consists of five basic commands:

CM — Clear Memory
BX — Box Draw
Q8 — Random Line
ED — Edit
BA, SH — Provides the capability of saving (BA) a screen on floppy disk and calling (SH) it back.

The clear memory (CM) command clears the screen. The box drawing (BX) command draws continuously concentric boxes which close in on each other. This gives the effect of running up a hallway. The random line (Q8) drawing command picks random points and connects them together until they form a multisided polygon and then it continues to repeat that shape, all the while collapsing in on itself and changing colors. A scaling function has been implemented to keep the figure occupying a major portion of the screen. The edit (ED) command allows the user to draw figures on the screen using the cursor controls on the terminal and allows a choice of colors. The BA command is used to store a screen full of data on floppy disk while the SH command is used to call it from the floppy disk and display it on the screen.

Each of the routines which write to the display RAM use the basic data layout for every pixel on the screen. Each pixel is controlled by four bits. Each bit corresponds to either luminance, blue, green, or red, as shown in Figure 6.

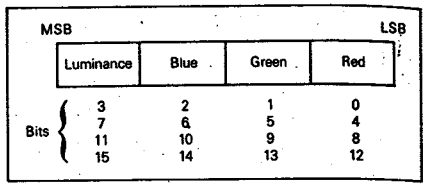


Figure 6. Pixel Control Bit — Layout

A memory map for this application is given in Figure 7. A listing of the software is given at the end of this application note.

The resolution of the display in this application is 256×256 pixels. The density could be doubled in both directions to 512×512 by quadrupling the memory. This can be easily done if dynamic RAM is used since 4K×1 and 16K×1 dynamic RAM can be arranged in the same basic configurations. As space was one of the design criteria in this application, some of the more straightforward approaches were not taken.

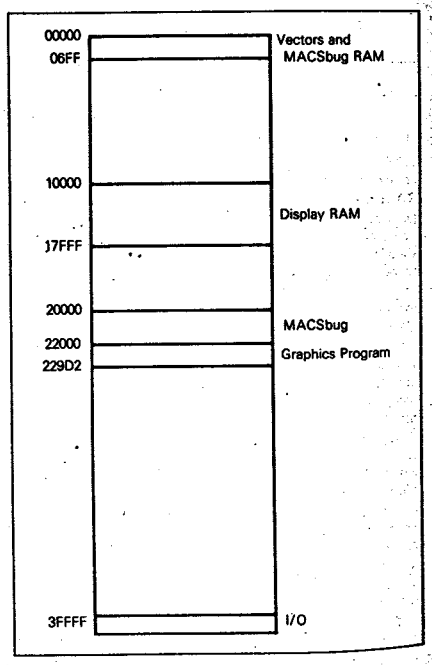


Figure 7. Memory Map

Thanks to Don Voss of Motorola Microsystems for his suggestions on the hardware and his splendid job on the software.


```
  10          00000000        ORG $0000
  20                          *
  30                          *
  40                          *
  50          000200F6        MACSBUG EQU $200F6
  60          00021BC2        OUTPUT2 EQU $21BC2
  70          00021F18        FIXBUF EQU $21F18
  80          000200EE        MSG EQU $200EE
  90          00001000        X1 EQU $1000
 100          00001001        Y1 EQU $1001
 110          00001002        X2 EQU $1002
 120          00001003        Y2 EQU $1003
 130          00001010        COLOR EQU $1010
 140          00001011        NCOLOR EQU $1011
 150          00001012        OCOLOR EQU $1012
 160          00001014        NUMPT EQU $1014
 170          00001016        SCALE EQU $1016
 180          00001018        RANADD EQU $1018
 190          00001080        ARRAY EQU $1080
 200          00001100        TABLECH EQU $1100
 210          00001800        CMDTAB EQU $1800
 220                          *
 230                          *
 240 000000 20780578          SETUP MOVE.L $578,A0
 250 000004 227C00001800       MOVE.L #CMDTAB,A1
 260 00000A 21C90578           MOVE.L A1,$578
 270 00000E 3018              SETUP1 MOVE (A0)+,D0
 280 000010 0C40FFFF           CMP #$FFFF,D0
 290 000014 6706               BEQ.S SETUP2
 300 000016 32C0               MOVE D0,(A1)+
 310 000018 22D8               MOVE.L (A0)+,(A1)+
 320 00001A 60F2               BRA SETUP1
 330 00001C B1FC00022000      SETUP2 CMP.L #$22000,A0
 340 000022 6A08               BPL.S INIT
 350 000024 207C00022082       MOVE.L #$22082,A0
 360 00002A 60E2               BRA SETUP1
 370 00002C 3280              INIT MOVE D0,(A1)
 380 00002E 207C000220D2       MOVE.L #$220D2,A0
 390 000034 303C0000           MOVE #$0000,D0
 400 000038 13C00001FFFD      INIT1 MOVE.B D0,$1FFFD
 410 00003E 1218               MOVE.B (A0)+,D1
 420 000040 4E71               NOP
 430 000042 13C10001FFFF       MOVE.B D1,$1FFFF
 440 000048 5240               ADD #1,D0
 450 00004A 0C400010           CMP #$0010,D0
 460 00004E 66E8               BNE INIT1
 470 000050 227C000229F6       MOVE.L #$229F6,A1
 480 000056 247C00001100       MOVE.L #TABLECH,A2
 490 00005C 303C0302           MOVE #770,D0
 500 000060 14D9              SETUP21 MOVE.B (A1)+,(A2)+
   0 000062 5340               SUB #1,D0
   0 000064 66FA               BNE  SETUP21
 520 000066 6014               BRA.S RETURN
 530 000068 207C00010000      CM MOVE.L #$10000,A0
 540 00006E 323C2000           MOVE #$2000,D1
```

```
 550 000072 4280           CLR.L D0
 560 000074 20C0          ,CLRM MOVE.L D0,(A0)+
   0 000076 5341            SUB #1,D1
   0 000078 66FA            BNE  CLRM
   0 00007A 4E71            NOP
 580 00007C 4EF9000200F6 RETURN JMP MACSBUG
 590 000082 43           NTABLE DC.W 'CM'
 600 000084 00022068      DC.L $22068
 610 000088 53            DC.W 'SH'
 620 00008A 000220E2      DC.L $220E2
 630 00008E 42            DC.W 'BX'
 640 000090 0002218A      DC.L $2218A
 650 000094 45            DC.W 'ED'
 660 000096 000221E8      DC.L $221E8
 670 00009A 42            DC.W 'BA'
 680 00009C 00022454      DC.L $22454
 690 0000A0 51            DC.W 'Q1'
 700 0000A2 00022498      DC.L $22498
 710 0000A6 51            DC.W 'Q2'
 720 0000A8 000224A4      DC.L $224A4
 730 0000AC 51            DC.W 'Q3'
 740 0000AE 000224B0      DC.L $224B0
 750 0000B2 51            DC.W 'Q4'
 760 0000B4 000224BC      DC.L $224BC
 770 0000B8 51            DC.W 'Q5'
 780 0000BA 000224C8      DC.L $224C8
 790 0000BE 51            DC.W 'Q9'
 800 0000C0 00022606      DC.L $22606
 810 0000C4 48            DC.W 'HP'
 820 0000C6 000226AC      DC.L $226AC
 830 0000CA 51            DC.W 'Q8'
 840 0000CC 00022818      DC.L $22818
 850 0000D0 FFFF          DC.W $FFFF
 860                      *
 870                      *
 880                      *
 890                      *
 900 0000D2 27           CRTC DC.B $27
 910 0000D3 20            DC.B $20
 920 0000D4 22            DC.B $22
 930 0000D5 A3            DC.B $A3
 940 0000D6 20            DC.B $20
 950 0000D7 06            DC.B $06
 960 0000D8 1F            DC.B $1F
 970 0000D9 1F            DC.B $1F
 980 0000DA 10            DC.B $10
 990 0000DB 07            DC.B 7
1000 0000DC 00000000      DC.L 0
1010 0000E0 0000          DC.W 0
1020                      *
1030                      *
1040 0000E2 61000004     SH BSR SHQ
1050 0000E6 6094          BRA RETURN
1060 0000E8 4EB900021BC2 SHQ JSR OUTPUT2
1070 0000EE 227C0003FF21  MOVE.L #$3FF21,A1
```



```
1080 0000F4 61000078     SH1 BSR INPUT
1090 0000F8 0C00000D      CMP.B #$0D,D0
1100 0000FC 6708          BEQ.S SH2
1110 0000FE 0C0000FF      CMP.B #$FF,D0
1120 000102 66F0          BNE SH1
1130 000104 6040          BRA.S SH3
1140 000106 61000066     SH2 BSR INPUT
1150 00010A 0C00000A      CMP.B #$0A,D0
1160 00010E 67F6          BEQ SH2
1170 000110 0C000000      CMP.B #0,D0
1180 000114 67F0          BEQ SH2
1190 000116 0C0000FF      CMP.B #$FF,D0
1200 00011A 672A          BEQ.S SH3
1210 00011C 4EB900021F18  JSR FIXBUF
1220 000122 2CFC4552524F  MOVE.L #'ERRO',(A6)+
1230 000128 2CFC52203B43  MOVE.L #'R ;C',(A6)+
1240 00012E 2CFC4845434B  MOVE.L #'HECK',(A6)+
1250 000134 2CFC2046494C  MOVE.L #' FIL',(A6)+
1260 00013A 2CFC45202020  MOVE.L #'E   ',(A6)+
1270 000140 4EF9000200EE  JMP MSG
1280 000146 207C00010000 SH3 MOVE.L #$10000,A0
1290 00014C 103C0055      MOVE.B #$55,D0
1300 000150 6100002A      BSR OUTPUT
1310 000154 61000018     SH4 BSR INPUT
1320 000158 1200          MOVE.B D0,D1
1330 00015A 61000012      BSR INPUT
1340 00015E E140          ASL 8,D0
1350 000160 1001          MOVE.B D1,D0
1360 000162 30C0          MOVE.W D0,(A0)+
1370 000164 B1FC00017F80  CMP.L #$17F80,A0
1380 00016A 66E8          BNE SH4
1390 00016C 4E75          RTS
1400 00016E 1011         INPUT MOVE.B (A1),D0
1410 000170 02000001      AND.B #1,D0
1420 000174 67F8          BEQ INPUT
1430 000176 10290002      MOVE.B 2(A1),D0
1440 00017A 4E75          RTS
1450 00017C 1E11         OUTPUT MOVE.B (A1),D7
1460 00017E 02070002      AND.B #2,D7
1470 000182 67F8          BEQ OUTPUT
1480 000184 13400002      MOVE.B D0,2(A1)
1490 000188 4E75          RTS
1500 00018A 4240         BX CLR D0
1510 00018C 3200          MOVE D0,D1
1520 00018E 3400          MOVE D0,D2
1530 000190 363C003F     BX3 MOVE #$3F,D3
1540 000194 207C00010000  MOVE.L #$10000,A0
1550 00019A 61000016     BX1 BSR SHOW
1560 00019E 5543          SUB #2,D3
1570 0001A0 6A02          BPL.S BX2
1580 0001A2 60EC          BRA BX3
1590 0001A4 5240         BX2 ADD #1,D0
1600 0001A6 5241          ADD #1,D1
1610 0001A8 5242          ADD #1,D2
1620 0001AA D1FC00000202  ADD.L #514,A0
```

```
1630 0001B0 60E8           BRA BX1
1640 0001B2 3803          SHOW MOVE D3,D4
1650 0001B4 30C0          BX11 MOVE D0,(A0)+
   0 0001B6 5344            SUB #1,D4
   0 0001B8 66FA            BNE  BX11
1670 0001BA 3080            MOVE D0,(A0)
1680 0001BC 3803            MOVE D3,D4
1690 0001BE E544            ASL 2,D4
1700 0001C0 D1FC00000080 BX22 ADD.L #128,A0
1710 0001C6 3081            MOVE D1,(A0)
   0 0001C8 5344            SUB #1,D4
   0 0001CA 66F4            BNE  BX22
1730 0001CC 3803            MOVE D3,D4
1740 0001CE 3080            MOVE D0,(A0)
1750 0001D0 3100          BX33 MOVE D0,-(A0)
   0 0001D2 5344            SUB  #1,D4
   0 0001D4 66FA            BNE  BX33
1770 0001D6 3803            MOVE D3,D4
1780 0001D8 E544            ASL 2,D4
1790 0001DA 91FC00000080 BX44 SUB.L #128,A0
1800 0001E0 3082            MOVE D2,(A0)
   0 0001E2 5344            SUB  #1,D4
   0 0001E4 66F4            BNE  BX44
1820 0001E6 4E75            RTS
1830                       *
1840                       *
1850                       *
1860 0001E8 11FC00801000 ED MOVE.B #$80,X1
1870 0001EE 11FC00801001  MOVE.B #$80,Y1
1880 0001F4 11FC00001011  MOVE.B #0,NCOLOR
1890 0001FA 6100014E     ED1 BSR BLINK
1900 0001FE 61000004       BSR CMD
1910 000202 60F6           BRA ED1
1920 000204 61000230     CMD BSR READK
1930 000208 0C010020       CMP.B #$20,D1
1940 00020C 6A48           BPL.S RTS
1950 00020E 0C01000B       CMP.B #$B,D1
1960 000212 673C           BEQ.S UPARROW
1970 000214 0C01000A       CMP.B #$A,D1
1980 000218 673E           BEQ.S DWARROW
1990 00021A 0C01000C       CMP.B #$C,D1
2000 00021E 673E           BEQ.S RTARROW
2010 000220 0C010008       CMP.B #$8,D1
2020 000224 673E           BEQ.S LTARROW
2030 000226 0C010001       CMP.B #$1,D1
2040 00022A 673E           BEQ.S CMD1  CHARMODE
2050 00022C 0C010003       CMP.B #$3,D1
2060 000230 6756           BEQ.S CMD2  NCOLOR
2070 000232 0C010004       CMP.B #$4,D1
2080 000236 6738           BEQ.S CMD3
2090 000238 0C01000D       CMP.B #$0D,D1
2100 00023C 673E           BEQ.S CR
2110 00023E 0C010005       CMP.B #$5,D1
2120 000242 6732           BEQ.S CMD4
2130 000244 0C010011       CMP.B #$11,D1
```



```
2140 000248 660A           BNE.S RTS1
2150 00024A 588F           ADD.L #4,A7
2160 00024C 6000FE2E       BRA RETURN
2170 000250 53381001     UPARROW SUB.B #1,Y1
2180 000254 4241         RTS1 CLR D1
2190 000256 4E75         RTS RTS
2200 000258 52381001     DWARROW ADD.B #1,Y1
2210 00025C 60F6           BRA RTS1
2220 00025E 52381000     RTARROW ADD.B #1,X1
2230 000262 60F0           BRA RTS1
2240 000264 53381000     LTARROW SUB.B #1,X1
2250 000268 60EA           BRA RTS1
2260 00026A 588F         CMD1 ADD.L #4,A7
2270 00026C 60000132       BRA CHARED
2280 000270 588F         CMD3 ADD.L #4,A7
2290 000272 600001A8       BRA DOT
2300 000276 588F         CMD4 ADD.L #4,A7
2310 000278 6000FF80       BRA ED1
2320 00027C 5E381001     CR ADD.B #7,Y1
2330 000280 11FC00001000  MOVE.B #0,X1
2340 000286 60CC           BRA RTS1
2350 000288 610001AC     CMD2 BSR READK
2360 00028C 267C00001011  MOVE.L #NCOLOR,A3
2370 000292 0C010052       CMP.B #'R',D1
2380 000296 6758           BEQ.S RED
2390 000298 0C010047       CMP.B #'G',D1
2400 00029C 6758           BEQ.S GREEN
2410 00029E 0C010042       CMP.B #'B',D1
2420 0002A2 6758           BEQ.S BLUE
2430 0002A4 0C010057       CMP.B #'W',D1
2440 0002A8 6758           BEQ.S WHITE
2450 0002AA 0C01005A       CMP.B #'Z',D1
2460 0002AE 6758           BEQ.S BLACK
2470 0002B0 0C010059       CMP.B #'Y',D1
2480 0002B4 6758           BEQ.S YELLOW
2490 0002B6 0C01004D       CMP.B #'M',D1
2500 0002BA 6758           BEQ.S MAG
2510 0002BC 0C010043       CMP.B #'C',D1
2520 0002C0 6758           BEQ.S CYAN
2530 0002C2 0C010054       CMP.B #'T',D1
2540 0002C6 6758           BEQ.S DRED
2550 0002C8 0C010048       CMP.B #'H',D1
2560 0002CC 6758           BEQ.S DGR
2570 0002CE 0C01004E       CMP.B #'N',D1
2580 0002D2 6758           BEQ.S DBLUE
2590 0002D4 0C010045       CMP.B #'E',D1
2600 0002D8 6758           BEQ.S DWH
2610 0002DA 0C010055       CMP.B #'U',D1
2620 0002DE 6758           BEQ.S DYEL
2630 0002E0 0C01002C       CMP.B #',',D1
2640 0002E4 6758           BEQ.S DMAG
2650 0002E6 0C010056       CMP.B #'V',D1
2660 0002EA 6758           BEQ.S DCYAN
2670 0002EC 4241         RTS2 CLR D1
2680 0002EE 4E75           RTS
```

```
2690 0002F0 16BC0009      RED MOVE.B #$9,(A3)
2700 0002F4 60F6           BRA RTS2
2710 0002F6 16BC000A      GREEN MOVE.B #$A,(A3)
2720 0002FA 60F0           BRA RTS2
2730 0002FC 16BC000C      BLUE MOVE.B #$C,(A3)
2740 000300 60EA           BRA RTS2
2750 000302 16BC000F      WHITE MOVE.B #$F,(A3)
2760 000306 60E4           BRA RTS2
2770 000308 16BC0000      BLACK MOVE.B #0,(A3)
2780 00030C 60DE           BRA RTS2
2790 00030E 16BC000B      YELLOW MOVE.B #$B,(A3)
2800 000312 60D8           BRA RTS2
2810 000314 16BC000D      MAG MOVE.B #$D,(A3)
2820 000318 60D2           BRA RTS2
2830 00031A 16BC000E      CYAN MOVE.B #$E,(A3)
2840 00031E 60CC           BRA RTS2
2850 000320 16BC0001      DRED MOVE.B #1,(A3)
2860 000324 60C6           BRA RTS2
2870 000326 16BC0002      DGR MOVE.B #2,(A3)
2880 00032A 60C0           BRA RTS2
2890 00032C 16BC0004      DBLUE MOVE.B #4,(A3)
2900 000330 60BA           BRA RTS2
2910 000332 16BC0007      DWH MOVE.B #7,(A3)
2920 000336 60B4           BRA RTS2
2930 000338 16BC0003      DYEL MOVE.B #3,(A3)
2940 00033C 60AE           BRA RTS2
2950 00033E 16BC0005      DMAG MOVE.B #5,(A3)
2960 000342 60A8           BRA RTS2
2970 000344 16BC0006      DCYAN MOVE.B #6,(A3)
2980 000348 60A2           BRA RTS2
2990                       *
3000 00034A 12381000      BLINK MOVE.B X1,D1
3010 00034E 14381001       MOVE.B Y1,D2
3020 000352 61000226       BSR GETADD
3030 000356 4643           NOT D3
3040 000358 0C03000F      BL2 CMP.B #$F,D3
3050 00035C 6706           BEQ.S BL1
3060 00035E E84B           LSR 4,D3
3070 000360 E849           LSR 4,D1
3080 000362 60F4           BRA BL2
3090 000364 11C11012      BL1 MOVE.B D1,OCOLOR
3100 000368 103C000F      BL3 MOVE.B #$F,D0
3110 00036C 12381000       MOVE.B X1,D1
3120 000370 14381001       MOVE.B Y1,D2
3130 000374 610001DE       BSR DSP
3140 000378 610000D0       BSR DLY
3150 00037C 4200           CLR.B D0
3160 00037E 610001D4       BSR DSP
3170 000382 610000C6       BSR DLY
3180 000386 10381012       MOVE.B OCOLOR,D0
3190 00038A 610001C8       BSR DSP
3200 00038E 610000BA       BSR DLY
3210 000392 10390003FF01  MOVE.B $3FF01,D0
3220 000398 02000001       AND.B #1,D0
3230 00039C 67CA           BEQ BL3
```



```
3240 00039E 4E75           RTS
3250                       *
3260 0003A0 31F810001002 CHARED MOVE X1,X2
3270 0003A6 61A2           BSR BLINK
3280 0003A8 6100FE5A       BSR CMD
3290 0003AC 4A01           TST.B D1
3300 0003AE 67F0           BEQ CHARED
3310 0003B0 61000004       BSR CHAR
3320 0003B4 60EA           BRA CHARED
3330 0003B6 04010020      CHAR SUB.B #$20,D1
3340 0003BA E741           ASL 3,D1
3350 0003BC 267C00001100   MOVE.L #TABLECH,A3
3360 0003C2 0281000003FF   AND.L #$3FF,D1
3370 0003C8 D7C1           ADD.L D1,A3
3380 0003CA 3C3C0004       MOVE #4,D6
3390 0003CE 4245          CHARED1 CLR D5
3400 0003D0 0B13          CHARED2 BTST D5,(A3)
3410 0003D2 6636           BNE.S SET
3420 0003D4 52381002      CHARED3 ADD.B #1,X2
3430 0003D8 5245           ADD #1,D5
3440 0003DA 0C450010       CMP #16,D5
3450 0003DE 6618           BNE:S CHARED4
3460 0003E0 52381003       ADD.B #1,Y2
3470 0003E4 11F810001002   MOVE.B X1,X2
3480 0003EA D7F80002       ADD.L $2,A3
   0 0003EE 5346           SUB  #1,D6
   0 0003F0 66DC           BNE  CHARED1
3500 0003F2 50381000       ADD.B #8,X1
3510 0003F6 4E75           RTS
3520 0003F8 0C450008      CHARED4 CMP #8,D5
3530 0003FC 66D2           BNE CHARED2
3540 0003FE 52381003       ADD.B #1,Y2
3550 000402 11F810001002   MOVE.B X1,X2
3560 000408 60C6           BRA CHARED2
3570 00040A 10381011      SET MOVE.B NCOLOR,D0
3580 00040E 12381002       MOVE.B X2,D1
3590 000412 14381003       MOVE.B Y2,D2
3600 000416 6100013C       BSR DSP
3610 00041A 60B8           BRA CHARED3
3620                       *
3630 00041C 10381011      DOT MOVE.B NCOLOR,D0
3640 000420 12381000       MOVE.B X1,D1
3650 000424 14381001       MOVE.B Y1,D2
3660 000428 6100012A       BSR DSP
3670 00042C 6100FF1C       BSR BLINK
3680 000430 6100FDD2       BSR CMD
3690 000434 60E6           BRA DOT
3700                       *
3710 000436 12390003FF01 READK MOVE.B $3FF01,D1
3720 00043C 02010001       AND.B #1,D1
3730 000440 67F4           BEQ READK
3740 000442 12390003FF03   MOVE.B $3FF03,D1
3750 000448 4E75           RTS
3760 00044A 3C3C00FF      DLY MOVE #$00FF,D6
3770 00044E 5346          DLY1 SUB #1,D6
```

```
3780 000450 66FC          BNE DLY1
3790 000452 4E75          RTS
3800                      *
3810                      *
3820                      *
3830 000454 207C00010000 BA MOVE.L #$10000,A0
3840 00045A 227C0003FF23  MOVE.L #$3FF23,A1
3850 000460 247C0003FF21  MOVE.L #$3FF21,A2
3860 000466 1212          L1 MOVE.B (A2),D1
3870 000468 02010002      AND.B #$2,D1
3880 00046C 67F8          BEQ L1
3890 00046E 103C0065      MOVE.B #$65,D0
3900 000472 1280          MOVE.B D0,(A1)
3910 000474 1212          LOOP MOVE.B (A2),D1
3920 000476 02010002      AND.B #$2,D1
3930 00047A 67F8          BEQ LOOP
3940 00047C 3018          MOVE (A0)+,D0
3950 00047E 1280          MOVE.B D0,(A1)
3960 000480 E048          LSR 8,D0
3970 000482 1212          L2 MOVE.B (A2),D1
3980 000484 02010002      AND.B #$2,D1
3990 000488 67F8          BEQ L2
4000 00048A 1280          MOVE.B D0,(A1)
4010 00048C B1FC00018000  CMP.L #$18000,A0
4020 000492 66E0          BNE LOOP
4030 000494 6000FBE6      BRA RETURN
4040                      *
4050                      *
4060                      *
4070 000498 2C7C000225AC Q1 MOVE.L #$225AC,A6
4080 00049E 3E3C0010      MOVE #$10,D7
4090 0004A2 602E          BRA.S RUN
4100 0004A4 2C7C000225BE Q2 MOVE.L #$225BE,A6
4110 0004AA 3E3C0010      MOVE #$10,D7
4120 0004AE 6022          BRA.S RUN
4130 0004B0 2C7C000225D0 Q3 MOVE.L #$225D0,A6
4140 0004B6 3E3C0010      MOVE #$10,D7
4150 0004BA 6016          BRA.S RUN
4160 0004BC 2C7C000225E2 Q4 MOVE.L #$225E2,A6
4170 0004C2 3E3C0010      MOVE #$10,D7
4180 0004C6 600A          BRA.S RUN
4190 0004C8 2C7C000225F4 Q5 MOVE.L #$225F4,A6
4200 0004CE 3E3C0010      MOVE #$10,D7
4210 0004D2 61000006      RUN BSR RUN1
4220 0004D6 6000FBA4      BRA RETURN
4230                      *
4240                      *
4250                      *
4260 0004DA 3C3C0080      RUN1 MOVE #128,D6
4270 0004DE 61000034      BSR RAND
4280 0004E2 4E96          RUN2 JSR (A6)
4290 0004E4 48E76000      MOVEM.L D1/D2,-(A7)
4300 0004E8 0241007F      AND #$7F,D1
4310 0004EC 0242007F      AND #$7F,D2
4320 0004F0 61000068      BSR DSPLY
```



```
4330 0004F4 4401          NEG.B D1
4340 0004F6 61000062      BSR DSPLY
4350 0004FA 4402          NEG.B D2
4360 0004FC 6100005C      BSR DSPLY
4370 000500 4401          NEG.B D1
4380 000502 61000056      BSR DSPLY
4390 000506 4CDF0006      MOVEM.L (A7)+,D1/D2
   0 00050A 5346          SUB  #1,D6
   0 00050C 66D4          BNE  RUN2
   0 00050E 5347          SUB  #1,D7
   0 000510 66C8          BNE  RUN1
4420 000512 4E75          RTS
4430                      *
4440                      *
4450                      *
4460 000514 6100001C      RAND BSR RAND1
4470 000518 3200          MOVE D0,D1
4480 00051A 61000016      BSR RAND1
4490 00051E 3400          MOVE D0,D2
4500 000520 61000010      RAND2 BSR RAND1
4510 000524 0200000F      AND.B #$F,D0
4520 000528 67F6          BEQ RAND2
4530 00052A 0C000008      CMP.B #$08,D0
4540 00052E 67F0          BEQ RAND2
4550 000530 4E75          RTS
4560 000532 10381019      RAND1 MOVE.B RANADD+1,D0
4570 000536 E500          ASL.B 2,D0
4580 000538 D0381018      ADD.B RANADD,D0
4590 00053C E140          ASL 8,D0
4600 00053E 10381019      MOVE.B RANADD+1,D0
4610 000542 E540          ASL 2,D0
4620 000544 D0781018      ADD RANADD,D0
4630 000548 06403619      ADD #$3619,D0
4640 00054C 31C01018      MOVE D0,RANADD
4650 000550 E048          LSR 8,D0
4660 000552 4E75          RTS
4670                      *
4680                      *
4690                      *
4700                      *DSPLY(C,X,Y)
4710                      *  D0=COLOR
4720                      *  D1=X 8-BITS
4730                      *  D2=Y 8-BITS
4740                      *
4750 000554 48E7F080      DSP MOVEM.L D0-D3/A0,-(A7)
4760 000558 600C          BRA.S DSP1
4770                      *
4780 00055A 48E7F080      DSPLY MOVEM.L D0-D3/A0,-(A7)
4790 00055E 06010080      ADD.B #128,D1
4800 000562 06020080      ADD.B #128,D2
4810 000566 0240000F      DSP1 AND #$F,D0
4820 00056A 6100000E      BSR GETADD
4830 00056E C243          AND D3,D1
4840 000570 8041          OR D1,D0
4850 000572 3080          MOVE D0,(A0)
```

```
4860 000574 4CDF010F       MOVEM.L (A7)+,D0-D3/A0
4870 000578 4E75           RTS
4880 00057A 024100FF      GETADD AND #$FF,D1
4890 00057E 363CFFF0       MOVE #$FFF0,D3
4900 000582 E142           ASL 8,D2
4910 000584 D242           ADD D2,D1
4920 000586 02810000FFFF   AND.L #$FFFF,D1
4930 00058C 3401           MOVE D1,D2
4940 00058E E449           LSR 2,D1
4950 000590 E341           ASL 1,D1
4960 000592 207C00010000   MOVE.L #$10000,A0
4970 000598 D1C1           ADD.L D1,A0
4980 00059A 02420003       AND #3,D2
4990 00059E 6708           BEQ.S DSPLY1
5000 0005A0 E940          DSPLY2 ASL 4,D0
5010 0005A2 E95B           ROL 4,D3
   0 0005A4 5342           SUB  #1,D2
   0 0005A6 66F8           BNE  DSPLY2
5030 0005A8 3210          DSPLY1 MOVE (A0),D1
5040 0005AA 4E75           RTS
5050                      *
5060                      *
5070 0005AC 3601          EQU1 MOVE D1,D3
5080 0005AE 3802           MOVE D2,D4
5090 0005B0 4883           EXT D3
5100 0005B2 4884           EXT D4
5110 0005B4 E64B           LSR 3,D3
5120 0005B6 E64C           LSR 3,D4
5130 0005B8 9403           SUB.B D3,D2
5140 0005BA 9204           SUB.B D4,D1
5150 0005BC 4E75           RTS
5160                      *
5170 0005BE 3602          EQU2 MOVE D2,D3
5180 0005C0 4883           EXT D3
5190 0005C2 E64B           LSR 3,D3
5200 0005C4 9203           SUB.B D3,D1
5210 0005C6 3801           MOVE D1,D4
5220 0005C8 4884           EXT D4
5230 0005CA E64C           LSR 3,D4
5240 0005CC D404           ADD.B D4,D2
5250 0005CE 4E75           RTS
5260                      *
5270                      *
5280 0005D0 3602          EQU3 MOVE D2,D3
5290 0005D2 4883           EXT D3
5300 0005D4 E24B           LSR 1,D3
5310 0005D6 D203           ADD.B D3,D1
5320 0005D8 3801           MOVE D1,D4
5330 0005DA 4884           EXT D4
5340 0005DC E24C           LSR 1,D4
5350 0005DE 9404           SUB.B D4,D2
5360 0005E0 4E75           RTS
5370                      *
5380 0005E2 3602          EQU4 MOVE D2,D3
5390 0005E4 4883           EXT D3
```



```
5400 0005E6 E64B           LSR 3,D3
5410 0005E8 9203           SUB.B D3,D1
5420 0005EA 3801           MOVE D1,D4
5430 0005EC 4884           EXT D4
5440 0005EE E64C           LSR 3,D4
5450 0005F0 9404           SUB.B D4,D2
5460 0005F2 4E75           RTS
5470                      *
5480 0005F4 3602          EQU5 MOVE D2,D3
5490 0005F6 4883           EXT D3
5500 0005F8 E24B           LSR 1,D3
5510 0005FA 9203           SUB.B D3,D1
5520 0005FC 3801           MOVE D1,D4
5530 0005FE 4884           EXT D4
5540 000600 E44C           LSR 2,D4
5550 000602 D404           ADD.B D4,D2
5560 000604 4E75           RTS
5570 000606 2C7C000225AC  Q9 MOVE.L #$225AC,A6
5580 00060C 3A3C0002      Q91 MOVE #2,D5
5590 000610 61000044      Q92 BSR CMQ
5600 000614 3E3C0020        MOVE #$20,D7
5610 000618 6100FEC0        BSR RUN1
5620 00061C 6100002C        BSR DLYQ
5630 000620 48E70402        MOVEM.L D5/A6,-(A7)
5640 000624 6100008E        BSR HP1
5650 000628 4CDF4020        MOVEM.L (A7)+,D5/A6
5660 00062C 6100001C        BSR DLYQ
   0 000630 5345            SUB  #1,D5
   0 000632 66DC            BNE  Q92
5680 000634 61000034        BSR LOGO
5690 000638 DDFC00000012    ADD.L #$12,A6
5700 00063E BDFC00022606    CMP.L #$22606,A6
5710 000644 670001D2        BEQ Q8
5720 000648 60C2            BRA Q91
5730 00064A 283C000AFFFF  DLYQ MOVE.L #$000AFFFF,D4
5740 000650 5384          DLYQ1 SUB.L #1,D4
5750 000652 66FC            BNE DLYQ1
5760 000654 4E75            RTS
5770 000656 4280          CMQ CLR.L D0
5780 000658 323C2000        MOVE #$2000,D1
5790 00065C 207C00010000    MOVE.L #$10000,A0
5800 000662 20C0          CMQ1 MOVE.L D0,(A0)+
   0 000664 5341            SUB  #1,D1
   0 000666 66FA            BNE  CMQ1
5820 000668 4E75            RTS
5830 00066A 48E7FFFE      LOGO MOVEM.L D0-D7/A0-A6,-(A7)
5840 00066E 4EB900021F18    JSR FIXBUF
5850 000674 2CFC53482053    MOVE.L #'SH S',(A6)+
5860 00067A 2CFC4C494445    MOVE.L #'LIDE',(A6)+
5870 000680 1CBC0020        MOVE.B #' ',(A6)
5880 000684 6100FA62        BSR SHQ
5890 000688 61C0            BSR DLYQ
5900 00068A 4EB900021F18    JSR FIXBUF
5910 000690 2CFC5348204D    MOVE.L #'SH M',(A6)+
5920 000696 2CFC41534B20    MOVE.L #'ASK ',(A6)+
```

```
5930 00069C 6100FA4A       BSR SHQ
5940 0006A0 4CDF7FFF       MOVEM.L (A7)+,D0-D7/A0-A6
5950 0006A4 283C0010FFFF   MOVE.L #$0010FFFF,D4
5960 0006AA 60A4           BRA DLYQ1
5970                       *
5980 0006AC 61000006       HP BSR HP1
5990 0006B0 6000F9CA       BRA RETURN
6000 0006B4 267C00001080 HP1 MOVE.L #ARRAY,A3
6010 0006BA 619A           BSR CMQ
6020 0006BC 4241           CLR D1
6030 0006BE 4242           CLR D2
6040 0006C0 363C00FF       MOVE #$FF,D3
6050 0006C4 3803           MOVE D3,D4
6060 0006C6 6100FE6A       BSR RAND1
6070 0006CA 02000007       AND.B #7,D0
6080 0006CE 5A00           ADD.B #5,D0
6090 0006D0 E340           ASL 1,D0
6100 0006D2 11C01014       MOVE.B D0,NUMPT
6110 0006D6 6100FE5A       BSR RAND1
6120 0006DA 0200001F       AND.B #$1F,D0
6130 0006DE 00000005       OR.B #$5,D0
6140 0006E2 11C01016       MOVE.B D0,SCALE
6150 0006E6 4245           CLR D5
6160 0006E8 6100FE48       H6 BSR RAND1
6170 0006EC 024000FF       AND #$FF,D0
6180 0006F0 17805000       MOVE.B D0,0(A3,D5)
6190 0006F4 B240           CMP D0,D1
6200 0006F6 6A02           BPL.S H1
6210 0006F8 1200           MOVE.B D0,D1
6220 0006FA B640           H1 CMP D0,D3
6230 0006FC 6B02           BMI.S H2
6240 0006FE 1600           MOVE.B D0,D3
6250 000700 6100FE30       H2 BSR RAND1
6260 000704 024000FF       AND #$FF,D0
6270 000708 17805001       MOVE.B D0,1(A3,D5)
6280 00070C B440           CMP D0,D2
6290 00070E 6A02           BPL.S H3
6300 000710 1400           MOVE.B D0,D2
6310 000712 B840           H3 CMP D0,D4
6320 000714 6B02           BMI.S H4
6330 000716 1800           MOVE.B D0,D4
6340 000718 BA381014       H4 CMP.B NUMPT,D5
6350 00071C 6704           BEQ.S H5
6360 00071E 5405           ADD.B #2,D5
6370 000720 60C6           BRA H6
6380         00000722      H5 EQU *
6390 000722 9203           H8 SUB.B D3,D1
6400 000724 9404           SUB.B D4,D2
6410 000726 4245           CLR D5
6420 000728 97335000       H61 SUB.B D3,0(A3,D5)
6430 00072C 99335001       SUB.B D4,1(A3,D5)
6440 000730 BA381014       CMP.B NUMPT,D5
6450 000734 6704           BEQ.S H9
6460 000736 5405           ADD.B #2,D5
6470 000738 60EE           BRA H61
```



```
6480 00073A 4243           H9 CLR D3
6490 00073C 203C0000FF00   MOVE.L #$FF00,D0
6500 000742 024100FF       AND #$FF,D1
6510 000746 80C1           DIVU D1,D0
6520 000748 4245           CLR D5
6530 00074A 16335000       H12 MOVE.B 0(A3,D5),D3
6540 00074E C6C0           MULU D0,D3
6550 000750 E04B           LSR 8,D3
6560 000752 17835000       MOVE.B D3,0(A3,D5)
6570 000756 BA381014       CMP.B NUMPT,D5
6580 00075A 6704           BEQ.S H11
6590 00075C 5405           ADD.B #2,D5
6600 00075E 60EA           BRA H12
6610 000760 203C0000FF00 H11 MOVE.L #$FF00,D0
6620 000766 024200FF       AND #$FF,D2
6630 00076A 80C2           DIVU D2,D0
6640 00076C 4245           CLR D5
6650 00076E 16335001       H14 MOVE.B 1(A3,D5),D3
6660 000772 C6C0           MULU D0,D3
6670 000774 E04B           LSR 8,D3
6680 000776 17835001       MOVE.B D3,1(A3,D5)
6690 00077A BA381014       CMP.B NUMPT,D5
6700 00077E 6704           BEQ.S H13
6710 000780 5405           ADD.B #2,D5
6720 000782 60EA           BRA H14
6730 000784 31D31000       H13 MOVE (A3),X1
6740 000788 3E3C001C       H131 MOVE #$1C,D7
6750 00078C 54381014       H132 ADD.B #2,NUMPT
6760 000790 1A381014       MOVE.B NUMPT,D5
6770 000794 37935000       MOVE (A3),0(A3,D5)
6780 000798 3C3C0004       H15 MOVE #4,D6
6790 00079C 6100FD94       BSR RAND1
6800 0007A0 0240000F       AND #$F,D0
6810 0007A4 67F2           BEQ H15
6820 0007A6 0C000008       CMP.B #$8,D0
6830 0007AA 67EC           BEQ H15
6840 0007AC 0C00000F       CMP.B #$F,D0
6850 0007B0 67E6           BEQ H15
6860 0007B2 4245           HP6 CLR D5
6870 0007B4 12335000       H17 MOVE.B 0(A3,D5),D1
6880 0007B8 14335001       MOVE.B 1(A3,D5),D2
6890 0007BC 6100008A       HP17 BSR LINE
6900 0007C0 BA381014       CMP.B NUMPT,D5
6910 0007C4 6748           BEQ.S H16
6920 0007C6 12335002       MOVE.B 2(A3,D5),D1
6930 0007CA 14335000       MOVE.B 0(A3,D5),D2
6940 0007CE 024100FF       AND #$FF,D1
6950 0007D2 024200FF       AND #$FF,D2
6960 0007D6 9242           SUB D2,D1
6970 0007D8 16381016       MOVE.B SCALE,D3
6980 0007DC 024300FF       AND #$FF,D3
6990 0007E0 C3C3           MULS D3,D1
7000 0007E2 E049           LSR 8,D1
7010 0007E4 D3335000       ADD.B D1,0(A3,D5)
7020 0007E8 12335003       MOVE.B 3(A3,D5),D1
```

```
7030 0007EC 024100FF       AND #$FF,D1
7040 0007F0 14335001       MOVE.B 1(A3,D5),D2
7050 0007F4 024200FF       AND #$FF,D2
7060 0007F8 9242           SUB D2,D1
7070 0007FA 16381016       MOVE.B SCALE,D3
7080 0007FE 024300FF       AND #$FF,D3
7090 000802 C3C3           MULS D3,D1
7100 000804 E049           LSR 8,D1
7110 000806 D3335001       ADD.B D1,1(A3,D5)
7120 00080A 5445           ADD #2,D5
7130 00080C 60A6           BRA H17
7140 00080E 5346          H16 SUB #1,D6
7150 000810 66A0           BNE HP6
7160 000812 5347           SUB #1,D7
7170 000814 6682           BNE H15
7180 000816 4E75           RTS
7190 000818 6100FE9A      Q8 BSR HP1
7200 00081C 283C000AFFFF   MOVE.L #$AFFFF,D4
7210 000822 6100FE2C       BSR DLYQ1
7220 000826 60F0           BRA Q8
7230                       *
7240                       *
7250                       *
7260 000828 12290002      DXDY MOVE.B 2(A1),D1
7270 00082C 9211           SUB.B (A1),D1
7280 00082E 650A           BCS.S XNEG
7290 000830 13410004       MOVE.B D1,4(A1)
7300 000834 42290006       CLR.B 6(A1)
7310 000838 4E75           RTS
7320 00083A 137C00010006  XNEG MOVE.B #1,6(A1)
7330 000840 4401           NEG.B D1
7340 000842 13410004       MOVE.B D1,4(A1)
7350 000846 4E75           RTS
7360                       *
7370                       *
7380         00000848      LINE EQU *
7390 000848 48E7FFFE       DRAW MOVEM.L D0-D7/A0-A6,-(A7)
7400 00084C 227C00001000    MOVE.L #X1,A1
7410 000852 13410002        MOVE.B D1,2(A1)
7420 000856 13420003        MOVE.B D2,3(A1)
7430 00085A 1211            MOVE.B (A1),D1
7440 00085C 14290001        MOVE.B 1(A1),D2
7450 000860 6100FCF2        BSR DSP
7460 000864 61C2           DRAW1 BSR DXDY
7470 000866 5289            ADD.L #1,A1
7480 000868 61BE            BSR DXDY
7490 00086A 5389            SUB.L #1,A1
7500 00086C 1211            MOVE.B (A1),D1
7510 00086E 14290001        MOVE.B 1(A1),D2
7520 000872 4A290004        TST.B 4(A1)
7530 000876 6766            BEQ.S DXZ
7540 000878 4A290005        TST.B 5(A1)
7550 00087C 67000088        BEQ DYZ
7560 000880 16290004        MOVE.B 4(A1),D3
7570 000884 B6290005        CMP.B 5(A1),D3
```



```
7580 000888 660000B0       BNE FULMOV
7590 00088C 4A290006       TST.B 6(A1)
7600 000890 6626           BNE.S SXN
7610 000892 4A290007       TST.B 7(A1)
7620 000896 6636           BNE.S SYN
7630 000898 6100FCBA      XPYP1 BSR DSP
7640 00089C 5201           ADD.B #1,D1
7650 00089E 5202           ADD.B #1,D2
7660 0008A0 B2290002       CMP.B 2(A1),D1
7670 0008A4 66F2           BNE XPYP1
7680 0008A6 607E           BRA.S XYDONE
7690 0008A8 6100FCAA      SXNSYN BSR DSP
7700 0008AC 5301           SUB.B #1,D1
7710 0008AE 5302           SUB.B #1,D2
7720 0008B0 B2290002       CMP.B 2(A1),D1
7730 0008B4 66F2           BNE SXNSYN
7740 0008B6 606E           BRA.S XYDONE
7750 0008B8 4A290007      SXN TST.B 7(A1)
7760 0008BC 66EA           BNE.S SXNSYN
7770 0008BE 6100FC94      SNP BSR DSP
7780 0008C2 5301           SUB.B #1,D1
7790 0008C4 5202           ADD.B #1,D2
7800 0008C6 B2290002       CMP.B 2(A1),D1
7810 0008CA 66F2           BNE SNP
7820 0008CC 6058           BRA.S XYDONE
7830 0008CE 6100FC84      SYN BSR DSP
7840 0008D2 5201           ADD.B #1,D1
7850 0008D4 5302           SUB.B #1,D2
7860 0008D6 B2290002       CMP.B 2(A1),D1
7870 0008DA 66F2           BNE SYN
7880 0008DC 6048           BRA.S XYDONE
7890 0008DE 4A290005      DXZ TST.B 5(A1)
7900 0008E2 6742           BEQ.S XYDONE
7910 0008E4 4A290007       TST.B 7(A1)
7920 0008E8 660E           BNE.S DXZYN
7930 0008EA 6100FC68      DXZ1 BSR DSP
7940 0008EE 5202           ADD.B #1,D2
7950 0008F0 B4290003       CMP.B 3(A1),D2
7960 0008F4 66F4           BNE DXZ1
7970 0008F6 602E           BRA.S XYDONE
7980 0008F8 6100FC5A      DXZYN BSR DSP
7990 0008FC 5302           SUB.B #1,D2
8000 0008FE B4290003       CMP.B 3(A1),D2
8010 000902 66F4           BNE DXZYN
8020 000904 6020           BRA.S XYDONE
8030 000906 4A290006      DYZ TST.B 6(A1)
8040 00090A 660E           BNE.S DYZN
8050 00090C 6100FC46      DYZ1 BSR DSP
8060 000910 5201           ADD.B #1,D1
8070 000912 B2290002       CMP.B 2(A1),D1
8080 000916 66F4           BNE DYZ1
8090 000918 600C           BRA.S XYDONE
8100 00091A 6100FC38      DYZN BSR DSP
8110 00091E 5301           SUB.B #1,D1
8120 000920 B2290002       CMP.B 2(A1),D1
```

```
8130 000924 66F4           BNE DYZN
8140 000926 32A90002      XYDONE MOVE 2(A1),(A1)
8150 00092A 1211           MOVE.B (A1),D1
8160 00092C 14290001       MOVE.B 1(A1),D2
8170 000930 6100FC22       BSR DSP
8180 000934 4CDF7FFF       MOVEM.L (A7)+,D0-D7/A0-A6
8190 000938 4E75           RTS
8200 00093A 33510008      FULMOV MOVE (A1),8(A1)
8210 00093E 16290004       MOVE.B 4(A1),D3
8220 000942 96290005       SUB.B 5(A1),D3
8230 000946 6208           BHI.S FUL1
8240 000948 337C0001000A   MOVE #$1,10(A1)
8250 00094E 6046           BRA.S FUL4
8260 000950 337C0100000A  FUL1 MOVE #$100,10(A1)
8270 000956 603E           BRA.S FUL4
8280 000958 16290008      FUL2 MOVE.B 8(A1),D3
8290 00095C 9611           SUB.B (A1),D3
8300 00095E 6402           BCC.S FUL21
8310 000960 4403           NEG.B D3
8320 000962 024300FF      FUL21 AND #$FF,D3
8330 000966 18290005       MOVE.B 5(A1),D4
8340 00096A 024400FF       AND #$FF,D4
8350 00096E C6C4           MULU D4,D3
8360 000970 18290009       MOVE.B 9(A1),D4
8370 000974 98290001       SUB.B 1(A1),D4
8380 000978 6402           BCC.S FUL22
8390 00097A 4404           NEG.B D4
8400 00097C 1A290004      FUL22 MOVE.B 4(A1),D5
8410 000980 024400FF       AND #$FF,D4
8420 000984 024500FF       AND #$FF,D5
8430 000988 C8C5           MULU D5,D4
8440 00098A 4A29000A       TST.B 10(A1)
8450 00098E 660E           BNE.S FULY
8460 000990 B883           CMP.L D3,D4
8470 000992 6710           BEQ.S GREAT
8480 000994 620E           BHI.S GREAT
8490 000996 3369000A000E  FUL4 MOVE 10(A1),14(A1)
8500 00099C 600C           BRA.S SAME
8510 00099E B883          FULY CMP.L D3,D4
8520 0009A0 6702           BEQ.S GREAT
8530 0009A2 62F2           BHI.S FUL4
8540 0009A4 337C0101000E  GREAT MOVE #$0101,14(A1)
8550 0009AA 12290008      SAME MOVE.B 8(A1),D1
8560 0009AE 14290009       MOVE.B 9(A1),D2
8570 0009B2 4A290007       TST.B 7(A1)
8580 0009B6 6606           BNE.S NEGY
8590 0009B8 D429000F       ADD.B 15(A1),D2
8600 0009BC 6004           BRA.S S2
8610 0009BE 9429000F      NEGY SUB.B 15(A1),D2
8620 0009C2 13420009      S2 MOVE.B D2,9(A1)
8630 0009C6 4A290006       TST.B 6(A1)
8640 0009CA 6606           BNE.S NEGX
8650 0009CC D229000E       ADD.B 14(A1),D1
8660 0009D0 6004           BRA.S S3
8670 0009D2 9229000E      NEGX SUB.B 14(A1),D1
```



```
8680 0009D6 13410008      S3 MOVE.B D1,8(A1)
8690 0009DA 6100FB78      FUL3 BSR DSP
8700 0009DE B2290002       CMP.B 2(A1),D1
8710 0009E2 670A           BEQ.S DRAW2
8720 0009E4 B4290003       CMP.B 3(A1),D2
8730 0009E8 6704           BEQ.S DRAW2
8740 0009EA 6000FF6C       BRA FUL2
8750 0009EE 32A90008     DRAW2 MOVE 8(A1),(A1)
8760 0009F2 6000FE70       BRA DRAW1
8770 0009F6 0000           END
```

****** TOTAL ERRORS 0-- 0

SYMBOL TABLE

ARRAY	001080	BA	000454	BL1	000364	BL2	000358
BL3	000368	BLACK	000308	BLINK	00034A	BLUE	0002FC
BX	00018A	BX1	00019A	BX11	0001B4	BX2	0001A4
BX22	0001C0	BX3	000190	BX33	0001D0	BX44	0001DA
CHAR	0003B6	CHARED	0003A0	CHARED1	0003CE	CHARED2	0003D0
CHARED3	0003D4	CHARED4	0003F8	CHTAB	0009F6	CLRM	000074
CM	000068	CMD	000204	CMD1	00026A	CMD2	000288
CMD3	000270	CMD4	000276	CMDTAB	001800	CMQ	000656
CMQ1	000662	COLOR	001010	CR	00027C	CRTC	0000D2
CYAN	00031A	DBLUE	00032C	DCYAN	000344	DGR	000326
DLY	00044A	DLY1	00044E	DLYQ	00064A	DLYQ1	000650
DMAG	00033E	DOT	00041C	DRAW	000848	DRAW1	000864
DRAW2	0009EE	DRED	000320	DSP	000554	DSP1	000566
DSPLY	00055A	DSPLY1	0005A8	DSPLY2	0005A0	DWARROW	000258
DWH	000332	DXDY	000828	DXZ	0008DE	DXZ1	0008EA
DXZYN	0008F8	DYEL	000338	DYZ	000906	DYZ1	00090C
DYZN	00091A	ED	0001E8	ED1	0001FA	EQU1	0005AC
EQU2	0005BE	EQU3	0005D0	EQU4	0005E2	EQU5	0005F4
FIXBUF	021F18	FUL1	000950	FUL2	000958	FUL21	000962
FUL22	00097C	FUL3	0009DA	FUL4	000996	FULMOV	00093A
FULY	00099E	GETADD	00057A	GREAT	0009A4	GREEN	0002F6
H1	0006FA	H11	000760	H12	00074A	H13	000784
H131	000788	H132	00078C	H14	00076E	H15	000798
H16	00080E	H17	0007B4	H2	000700	H3	000712
H4	000718	H5	000722	H6	0006E8	H61	000728
H8	000722	H9	00073A	HP	0006AC	HP1	0006B4
HP17	0007BC	HP6	0007B2	INIT	00002C	INIT1	000038
INPUT	00016E	L1	000466	L2	000482	LINE	000848
LOGO	00066A	LOOP	000474	LTARROW	000264	MACSBUG	0200F6
MAG	000314	MSG	0200EE	NCOLOR	001011	NEGX	0009D2
NEGY	0009BE	NTABLE	000082	NUMPT	001014	OCOLOR	001012
OUTPUT	00017C	OUTPUT2	021BC2	Q1	000498	Q2	0004A4
Q3	0004B0	Q4	0004BC	Q5	0004C8	Q8	000818
Q9	000606	Q91	00060C	Q92	000610	RANADD	001018
RAND	000514	RAND1	000532	RAND2	000520	READK	000436
RED	0002F0	RETURN	00007C	RTARROW	00025E	RTS	000256
RTS1	000254	RTS2	0002EC	RUN	0004D2	RUN1	0004DA
RUN2	0004E2	S2	0009C2	S3	0009D6	SAME	0009AA
SCALE	001016	SET	00040A	SETUP	000000	SETUP1	00000E

```
SETUP4.S  MC68000 ASM REV= 1.0F- COPYRIGHT BY MOTOROLA 1978      PAGE 18

SETUP2      00001C SETUP21     000060 SH          0000E2 SH1         0000F4
SH2         000106 SH3         000146 SH4         000154 SHOW        0001B2
SHQ         0000E8 SNP         0008BE SXN         0008B8 SXNSYN      0008A8
SYN         0008CE TABLECH     001100 UPARROW     000250 WHITE       000302
X1          001000 X2          001002 XNEG        00083A XPYP1       000898
XYDONE      000926 Y1          001001 Y2          001003 YELLOW      00030E
```


AN-836

USING LOW-COST 1 MHz PERIPHERALS IN A 2 MHz SYSTEM WITH THE MC68B09 AND THE MC68B09E

by
Duane Graden and Hunter Scales
NMOS Microcomputer Applications
Motorola Inc.

INTRODUCTION

With the increasing use of HMOS design techniques in VLSI circuits, the maximum speed of these devices is also on the rise. There are 2 MHz, "B," versions of the popular MC6809 and the new MC6809E with external clock. Both the MC68B09 and the MC68B09E feature a 500 nanosecond cycle time. With a 2 MHz E clock, an add immediate instruction takes just 1 microsecond! These fast, efficient processors offer designers the opportunity to use a microprocessor in applications which have been, until now, too slow.

It would appear that the speed increase necessarily carries with it a cost penalty. That is, by increasing the speed of the bus, faster and therefore more expensive memories and peripherals must be used. However, there are ways to manipulate the 2 MHz MPU access time to accommodate slower peripherals and memories.

MPU ACCESS TIME MANIPULATION

The system clocks on the MC6809 can be delayed (stretched) to allow longer access time for slow memories using the MRDY input pin. Figure 1 shows the timing for this input. The system E and Q clocks are stretched, while E is high and Q is low, in one-quarter bus cycle increments. One quarter cycle of the MC68B09 2 MHz clock is equal to 125 nanoseconds. Since the MC6809E requires an external clock generator, the MRDY signal can be implemented externally for that processor.

A problem arises when stretching the access time for slow memories in that the throughput of the 2 MHz system is reduced markedly because the majority of processor cycles are, in fact, memory accesses. One solution to this problem is a compromise: absorb the cost of fast memories to allow the processor to run all memory cycles at full speed but reduce the speed of the bus for peripheral access. Since many peripherals are accessed only infrequently, this approach incurs minor impact on total throughput.

Unfortunately, slowing the bus cycle to accommodate slow peripherals is not as simple as using slow memories. To begin with, all MC6809 family peripherals require a continuous system clock to function. If the peripherals are specified at 1 MHz, this clock cannot exceed 1 MHz. This requires a separate, 1 MHz peripheral clock. This clock may not be synchronous with the main 2 MHz processor clock. Therefore, the chip enable signals to the peripherals must be delayed until the peripheral clock is low and then meet the chip select ($\overline{CS}$) setup time. In 1 MHz chips, chip select time is 160 nanoseconds before the rising edge of the clock. Some circuits, designed to allow the use of an MC6809 peripheral chip operating at one-half the frequency of the 2 MHz system clock, are described in the following paragraphs.

USING THE MC68B09 WITH 1 MHz PERIPHERALS

The circuit shown in Figure 2 allows 1 MHz peripherals to run with a 2 MHz MC68B09 system by generating an asynchronous peripheral clock (PCLK). When an access of any 1 MHz peripheral takes place, the 2 MHz system clocks, E and Q, are stretched using the MRDY pin. A state machine then waits until PCLK is low and then chip selects the peripheral 250 nanoseconds before the rising edge of PCLK. This provides proper address setup time at the peripherals before chip selecting them. Clocks E and Q are then released and the data is latched.

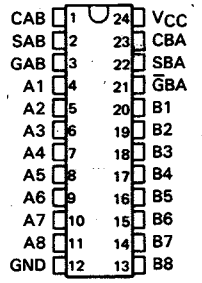
Refer to the timing diagram in Figure 3 and note the signal relationships during write and read cycles. Initiation of a

TYPES SN54HCT651, SN54HCT652, SN74HCT651, SN74HCT652 OCTAL BUS TRANSCEIVERS AND REGISTERS WITH 3-STATE OUTPUTS

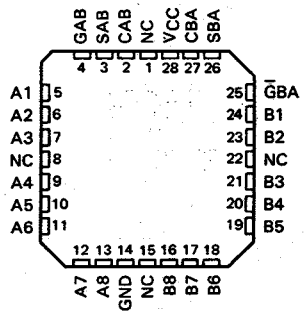
D2804, MARCH 1984

- Inputs are TTL-Voltage Compatible
- Bus Transceivers and Registers
- Independent Registers and Enables for A and B Buses
- High-Current 3-State Outputs Can Drive up to 15 LSTTL Loads
- Multiplexed Real-Time and Stored Data
- Choice of True and Inverting Data Paths
- Package Options Include Small Outline (SO) and Ceramic Chip Carriers in Addition to Plastic and Ceramic DIPs
- Dependable Texas Instruments Quality and Reliability

SN54HCT651, SN54HCT652 . . . JT PACKAGE
SN74HCT651, SN74HCT652 . . . JT OR NT OR D (= SO) PACKAGE
(TOP VIEW)



SN54HCT651, SN54HCT652 . . . FH OR FK PACKAGE
(TOP VIEW)



NC—No internal connection

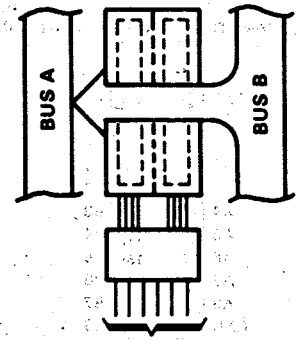
description

These devices consist of bus transceiver circuits, D-type flip-flops, and control circuitry arranged for multiplexed transmission of data directly from the data bus or from the internal storage registers. Enable GAB and $\overline{\text{G}}$BA are provided to control the transceiver functions. SAB and SBA control pins are provided to select whether real-time or stored data is transferred. A low input level selects real-time data, and a high selects stored data. The examples on the following page demonstrate the four fundamental bus-management functions that can be performed with the 'HCT651 and 'HCT652.

Data on the A or B data bus, or both, can be stored in the internal D flip-flops by low-to-high transitions at the appropriate clock pins (CAB or CBA) regardless of the select or enable control pins. When SAB and SBA are in the real-time transfer mode, it is also possible to store data without using the internal D-type flip-flops by simultaneously enabling GAB and $\overline{\text{G}}$BA. In this configuration each output reinforces its input. Thus, when all other data sources to the two sets of bus lines are at high impedance, each set of bus lines will remain at its last state.

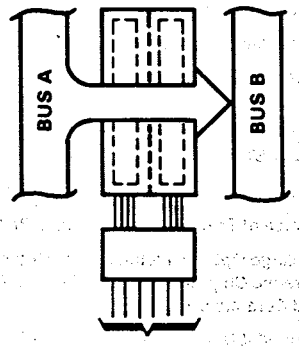
The SN54HCT651 and SN54HCT652 are characterized for operation over the full military temperature range of −55°C to 125°C. The SN74HCT651 and SN74HCT652 are characterized for operation from −40°C to 85°C.

TEXAS INSTRUMENTS



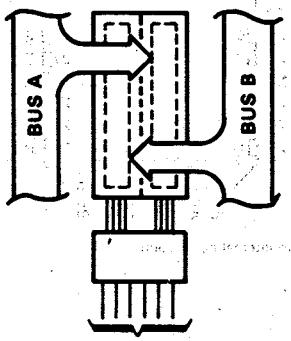
(3) GAB	(21) $\overline{G}$BA	(1) CAB	(23) CBA	(2) SAB	(22) SBA
L	L	X	X	X	L

REAL-TIME TRANSFER
BUS B TO BUS A



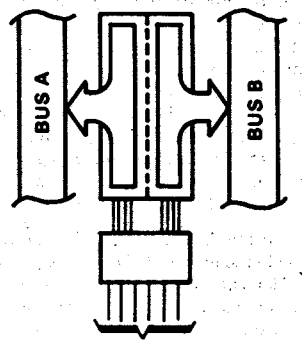
(3) GAB	(21) $\overline{G}$BA	(1) CAB	(23) CBA	(2) SAB	(22) SBA
H	H	X	X	L	X

REAL-TIME TRANSFER
BUS A TO BUS B



(3) GAB	(21) $\overline{G}$BA	(1) CAB	(23) CBA	(2) SAB	(22) SBA
X	H	↑	X	X	X
L	X	X	↑	X	X
L	H	↑	↑	X	X

STORAGE FROM
A AND/OR B



(3) GAB	(21) $\overline{G}$BA	(1) CAB	(23) CBA	(2) SAB	(22) SBA
H	L	H or L	H or L	H	H

TRANSFER
STORED DATA
TO A AND/OR B

Pin numbers shown are for JT and NT packages.

TYPES SN54HCT651, SN54HCT652, SN74HCT651, SN74HCT652
OCTAL BUS TRANSCEIVERS AND REGISTERS
WITH 3-STATE OUTPUTS

3 HCMOS DEVICES

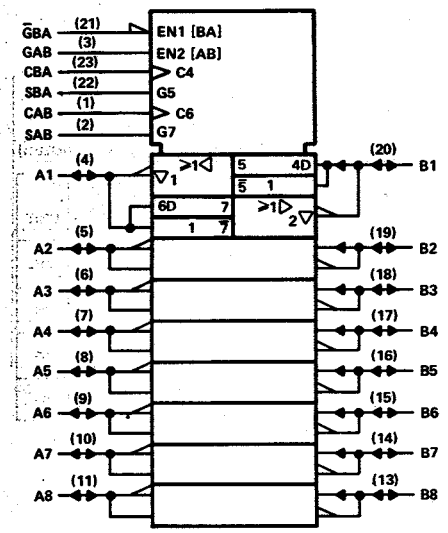
FUNCTION TABLE

INPUTS						DATA I/O†		OPERATION OR FUNCTION	
GAB	$\overline{G}$BA	CAB	CBA	SAB	SBA	A1 THRU A8	B1 THRU B8	'HCT651	'HCT652
L	H	H or L	H or L	X	X	Input	Input	Isolation	Isolation
L	H	↑	↑	X	X			Store A and B Data	Store A and B Data
X	H	↑	H or L	X	X	Input	Not specified	Store A, Hold B	Store A, Hold B
H	H	↑	↑	X	X	Input	Output	Store A in both registers	Store A in both registers
L	X	H or L	↑	X	X	Not specified	Input	Hold A, Store B	Hold A, Store B
L	L	↑	↑	X	X	Output	Input	Store B in both registers	Store B in both registers
L	L	X	X	X	L	Output	Input	Real-Time $\overline{B}$ Data to A Bus	Real-Time B Data to A Bus
L	L	X	H or L	X	H			Stored $\overline{B}$ Data to A Bus	Stored B Data to A Bus
H	H	X	X	L	X	Input	Output	Real-Time $\overline{A}$ Data to B Bus	Real-Time A Data to B Bus
H	H	H or L	X	H	X			Stored $\overline{A}$ Data to B Bus	Stored A Data to B Bus
H	L	H or L	H or L	H	H	Output	Output	Stored $\overline{A}$ Data to B Bus and	Stored A Data to B Bus and
								Stored $\overline{B}$ Data to A Bus	Stored B Data to A Bus

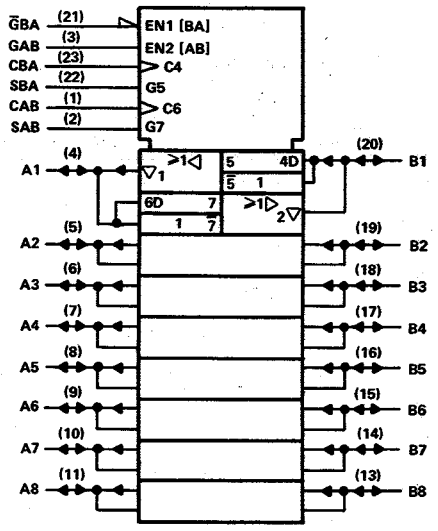
† The data output functions may be enabled or disabled by various signals at the GAB and $\overline{G}$BA inputs. Data input functions are always enabled, i.e., data at the bus pins will be stored on every low-to-high transition on the clock inputs.

logic symbols

'HCT651

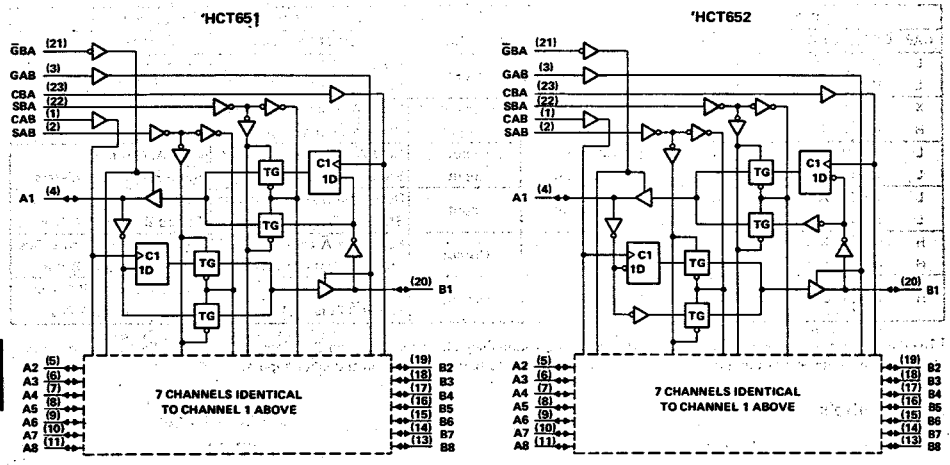


'HCT652



Pin numbers shown are for JT and NT packages.

logic diagram (positive logic)



Pin numbers shown are for JT and NT packages.

absolute maximum ratings, recommended operating conditions, and electrical characteristics

See Table VII, page 2-14.

timing requirements over recommended operating free-air temperature range (unless otherwise noted)

		V_{CC}	T_A = 25°C		SN54HCT651 SN54HCT652		SN74HCT651 SN74HCT652		UNIT
			MIN	MAX	MIN	MAX	MIN	MAX	
f_{clock}	Clock frequency	4.5 V	0	25	0	17	0	20	MHz
		5.5 V	0	28	0	19	0	22	
t_w	Pulse duration, CBA or CAB high or low	4.5 V	20		30		25		ns
		5.5 V	18		27		23		
t_{su}	Setup time, A before CAB↑ or B before CBA↑	4.5 V	15		23		19		ns
		5.5 V	14		21		17		
t_h	Hold time, A after CAB↑ or B after CBA↑	4.5 V	5		5		5		ns
		5.5 V	5		5		5		

switching characteristics over recommended operating free-air temperature range (unless otherwise noted), C_L = 50 pF (see Note 1)

PARAMETER	FROM (INPUT)	TO (OUTPUT)	V_{CC}	T_A = 25°C			SN54HCT651 SN54HCT652		SN74HCT652 SN74HCT652		UNIT
				MIN	TYP	MAX	MIN	MAX	MIN	MAX	
f_{max}			4.5 V	25	35		17		20		MHz
			5.5 V	28	40		19		22		
t_{pd}	CBA or CAB	A or B	4.5 V		18	36		54		45	ns
			5.5 V		16	32		49		41	
t_{pd}	A or B	B or A	4.5 V		14	27		41		34	ns
			5.5 V		12	24		37		31	
t_{pd}	SBA or SAB↑	A or B	4.5 V		20	38		57		48	ns
			5.5 V		17	34		51		43	
t_{en}	$\overline{G}$BA or GAB	A or B	4.5 V		25	49		74		61	ns
			5.5 V		22	44		67		55	
t_{dis}	$\overline{G}$BA or GAB	A or B	4.5 V		25	49		74		61	ns
			5.5 V		22	44		67		55	
t_t		Any	4.5 V		9	12		18		15	ns
			5.5 V		7	11		16		14	

C_{pd}	Power dissipation capacitance	No load, T_A = 25°C	50 pF typ

switching characteristics over recommended operating free-air temperature range (unless otherwise noted), C_L = 150 pF (see Note 1)

PARAMETER	FROM (INPUT)	TO (OUTPUT)	V_{CC}	T_A = 25°C			SN54HCT651 SN54HCT652		SN74HCT651 SN74HCT652		UNIT
				MIN	TYP	MAX	MIN	MAX	MIN	MAX	
t_{pd}	CBA or CAB	A or B	4.5 V		24	53		80		66	ns
			5.5 V		22	47		72		60	
t_{pd}	A or B	B or A	4.5 V		22	44		70		55	ns
			5.5 V		20	39		60		50	
t_{pd}	SBA or SAB↑	A or B	4.5 V		26	55		83		69	ns
			5.5 V		24	49		74		62	
t_{en}	$\overline{G}$BA or GAB	A or B	4.5 V		33	66		100		82	ns
			5.5 V		30	59		90		74	
t_t		Any	4.5 V		17	42		63		53	ns
			5.5 V		14	38		57		48	

NOTE 1: For load circuit and voltage waveforms, see page 1-14.
↑These parameters are measured with the internal output state of the storage register opposite to that of the bus input.