system reliability/system integrity
DUAL SYSTEMS CONTROL CORPORATION
2530 San Pablo Avenue Berkeley,CA. 94702
415/549-3854 • TELEX 172029 SPX

CPU/68000

USER'S MANUAL

68000-BASED
CENTRAL PROCESSING UNIT BOARD
FOR THE I.E.E.E. 696/S-100 BUS

Dual Systems Control Corp.
2530 San Pablo Avenue
Berkeley, CA 94702

Rev. B

Dual Systems CPU/68000 User's Manual

Table of Contents

INTRODUCTION

The **Dual Systems CPU/68000** is a high-performance CPU board combining the Motorola MC68000 chip, the logic circuitry necessary for interfacing to the S-100 bus in full compliance with the IEEE-696 specification, and ROMs containing a powerful monitor. Among its highlights are:

- o 8 MHz 68000 microprocessor
- o 4 MHz S-100 bus operation
- o 24 bit extended address bus
- o 16 bit data transfers
- o 8 bit transfers for compatibility with older peripherals
- o On chip interrupt controller
- o Operation with up to 16 DMA devices
- o Up to 8 Kilobytes of on board ROM
- o Supports I/O mapped peripherals

The **Dual Systems** CPU/68000 board is based on the Motorola 68000 processor, a high-performance microprocessor with 32-bit internal architecture and a large, uniform memory space. The 68000 features 16 32-bit registers, eight for addresses and eight for data. Data can be accessed in byte, word, and long word (32 Bit) quantities.

The board is designed to take full advantage of new IEEE-696 S-100 features. 16 bit memory accesses double the effective transfer rate of the 4 MHz S-100 bus. The processor fully complies with IEEE specifications for a permanent bus master and supports temporary bus master operation. Twenty-four address lines allow direct access to 16 Megabytes of memory.

SPECIFICATIONS

Processor:	Motorola MC68000-L8
Clock Speed:	8 Megahertz
Bus:	Meets all requirements of IEEE-696 (S-100)
Address Bus:	24 bits; conforms to S-100 extended addressing specifications (16 Megabytes)
Data Bus:	16 bit bidirectional data transfers. Also supports byte data transfers to eight-bit peripherals.
ROM:	Two sockets are provided on board for up to 8K of ROM. This ROM can be used for program storage or exception vectors or both.
Control:	Configured as bus master, provides TMA protocol per IEEE-696. Provides automatic 8/16 bit data path selection. (requires 16 bit memory for program execution). Provides 64k programmable I/O space.
Machine Cycle Time:	Standard S-100 cycle: 750nS (min) Fast Mode: 500nS (min)
Memory Speed:	Memory must have data on the bus no later than 450 nS after address is valid on bus.
Status Indicators:	RUN (Green LED) HALT (Red LED) HOLD (Yellow LED)
PC board:	High quality epoxy, solder masked both sides, screened component legend, plated through holes, gold plated edge connector fingers.
Sockets:	Provided for all IC's
Power Consumption:	950 mA nominal at 5 V.
User-Selectable Options:	Hardware relocatable boot and exception vectors. A0 line of address bus may be asserted for high byte or low byte. Phantom line asserted while in USER mode. (for example disk controller may be disabled while not in SYSTEM mode.)

Booting the CPU/68000 with Macsbug[1]

The CPU/68000 comes with the Macsbug[1] monitor installed in the on-board ROM sockets. The monitor is factory configured for use with a Godbout Interfacer·I serial I/O board. If the CPU/68000 is ordered with the Interfacer[2] and CMEM memory cards, then the system can be brought up immediately.

Set the dip-switches on the CPU/68000, Interfacer[2], and CMEM cards as shown in figures 1, 2, and 3.

After the dip-switches have been set properly, insert the CPU/68000, the Interfacer, and the CMEM boards into the S-100 card cage. Then connect the serial I/O cables between the Interfacer card and the terminal. Be sure to connect pin 1 on the ribbon cable by the index on the edge connector. **Set the terminal for 9600 BAUD and upper case only.** Now apply power. If everything was done properly, you should see the Macsbug prompt on the terminal:

MACSBUG 1.31
*****

If this does not appear, turn off the power and recheck all connections and dip-switch settings. Be sure the Interfacer and the terminal are set for identical BAUD rates. Try again. If there is still no response please call Dual Systems.

The dip-switch settings on the CPU/68000 map the monitor program to location 020000H and provide for the boot vectors to be read from the ROMs. These switches are described fully in this manual.

The Interfacer[2] switch setting define the first port to be at I/O location 0H and the second port (for printer or host computer) at I/O location 2H.

In order to configure the board for use with your terminal and printer, the port 1 baud rate must be set for the speed of the console terminal and the Port 2 baud rate must match the speed of the printer or the connection to the host computer. In the figure these rates are 9600 and 300 respectively. Parity and stop bits are set for use with an ADM 3A or ADM 5 terminal. For more information regarding baud rates, stop bits, parity etc., refer to the Godbout Interfacer I manual.

The CMEM is set to span memory locations 0H to 7FFFH. The stacks reside in the top 1 Kbyte of this memory, the exeption vectors in the low 1 Kbytes and the middle is available for user programs. The remaining switches are set to enable extended addressing, initially enable the board, and to allow writing to the board. For more details refer to the CMEM manual.

---

1 Macsbug is a trademark of Motorola.
2 Interfacer is a trademark of Godbout Electronics

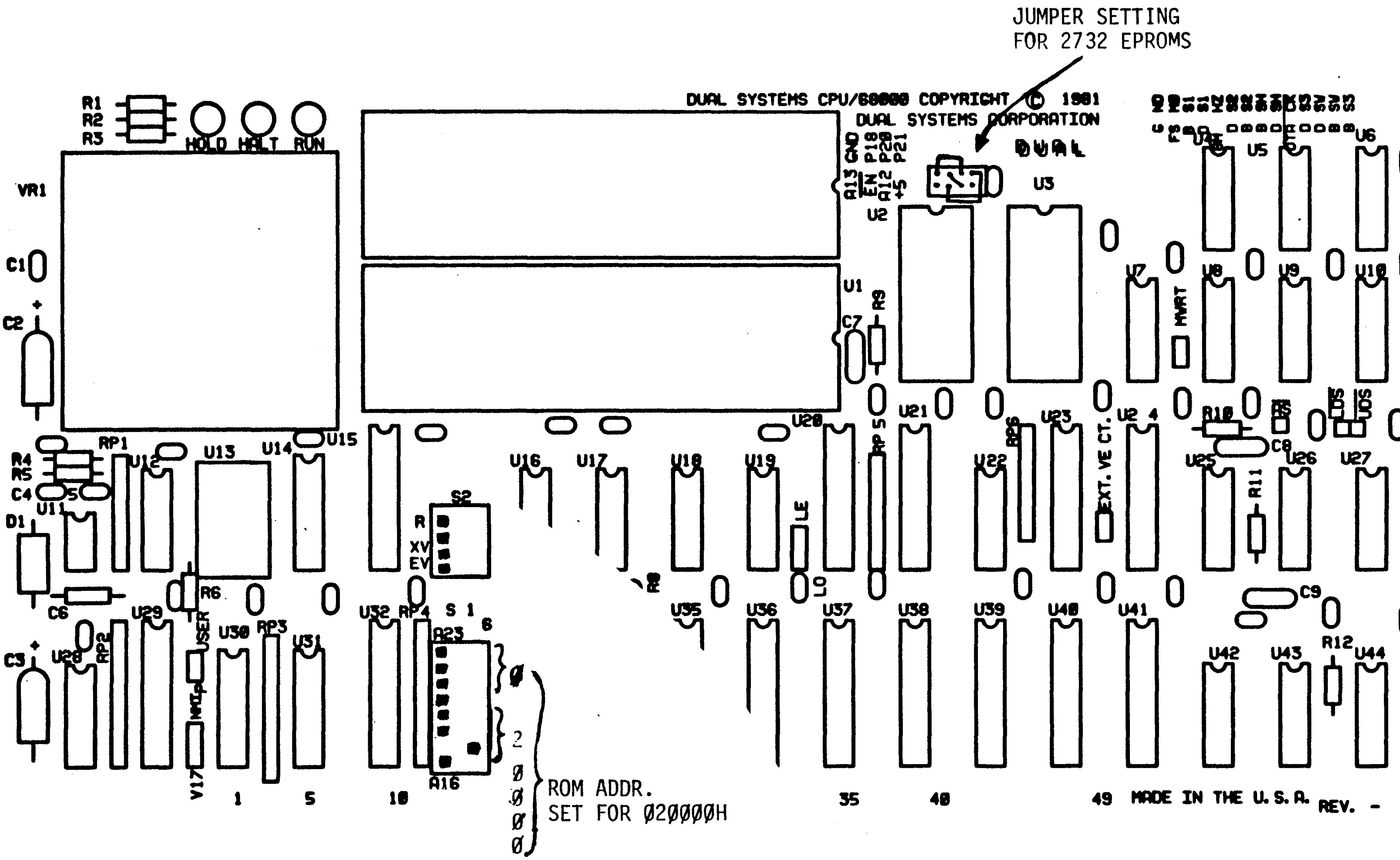

FIGURE 1

Figure 1. Factory settings for switches S1 and S2, and jumpers on CPU/68000 board.

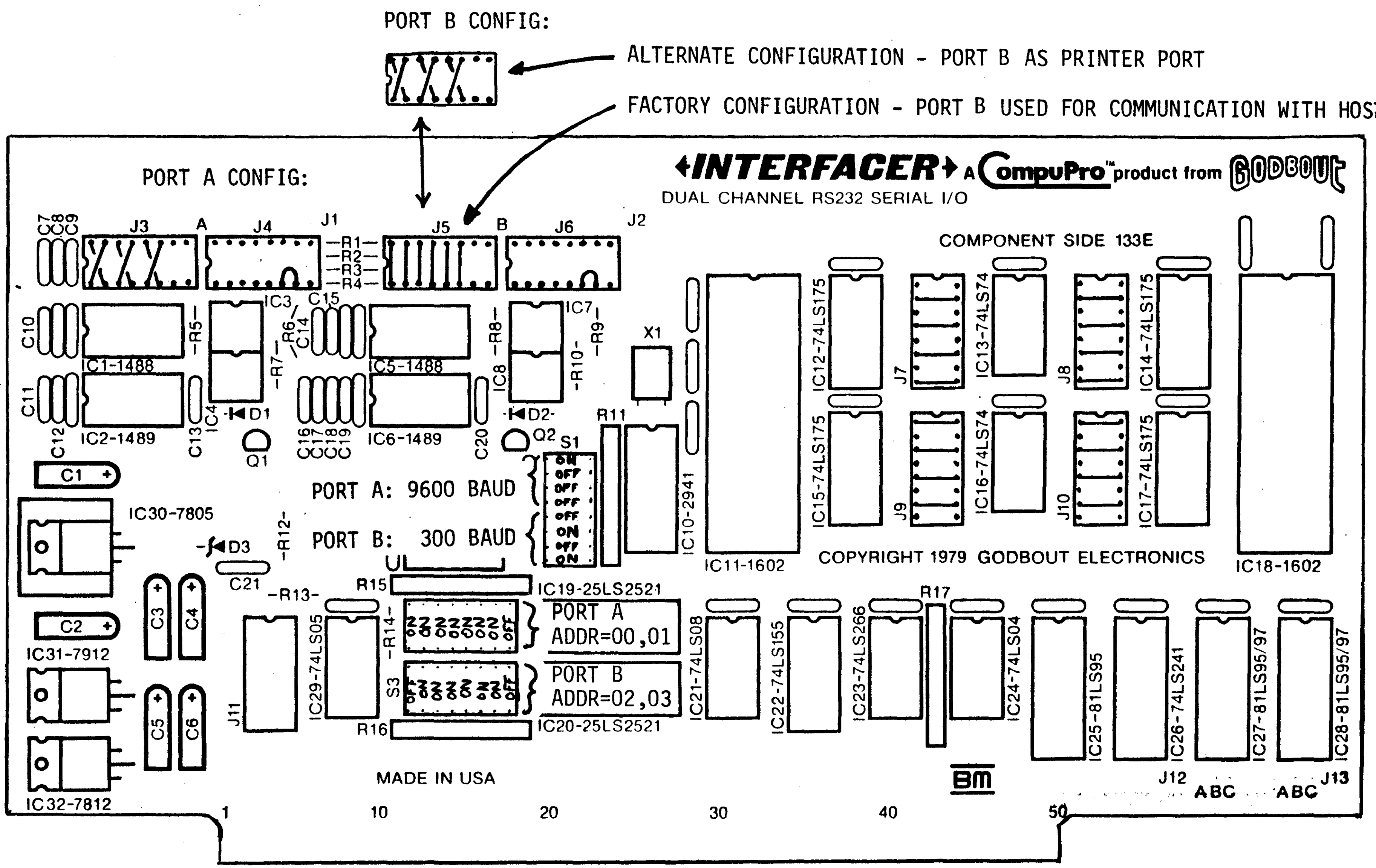

FIGURE 2

Figure 2. Factory configuration of serial I/O board for operation with CPU/68000.

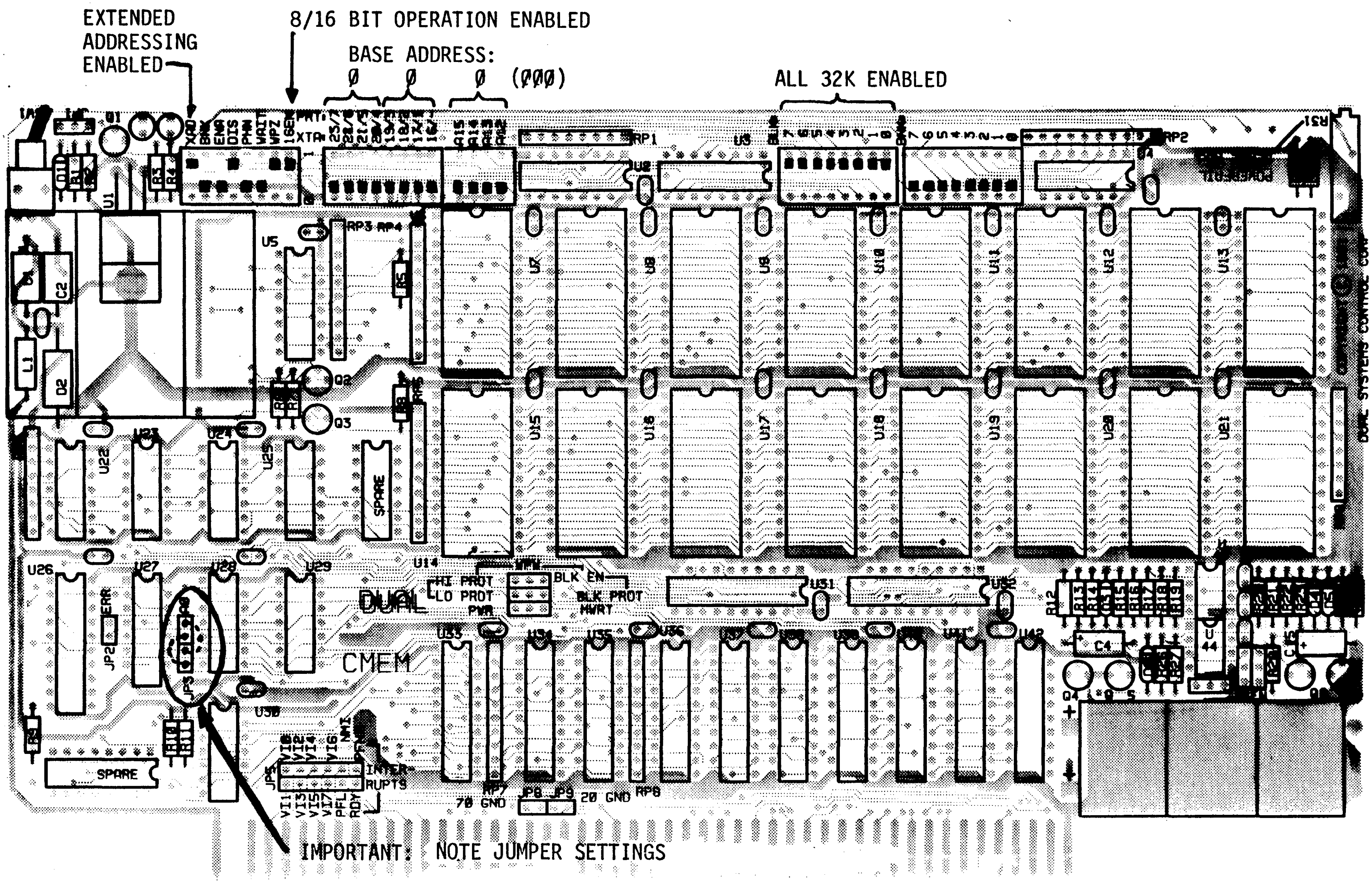

FIGURE 3

Figure 3. Switch and jumper settings for CMEM series nonvolatile memories when used with CPU/68000.

ON-BOARD ROM

Two ROM sockets are available on the CPU board to store up to 8K bytes of data. The ROMs can be used to store programs, interrupt vectors, or both. A variety of five volt only, 8 bit EPROMs or ROMs may be used. The CPU/68000 is factory configured for use with 2732 EPROMs, see Appendix A for jumper installation for alternate EPROMs.

Switches **S1** and **S2** determine the ROMs base address and their mode of operation. Either one of two conditions may enable the ROMs:

Using the ROMs to Store Programs:

The ROMs may be selected on any read from the address space starting at the address specified by **S1.** This mode is selected if **S2-3** is on. **Since only the most significant 8 address bits are decoded the ROMS occupy a full 64 kilobytes of memory.**

Hardware relocation of the Exception Vectors:

The ROMs can be enabled on memory reads to the exception vector address space. The MC68000 expects to find the exception vectors stored in the first 1024 bytes of memory. Typically it is desirable to store most vectors in RAM to allow software control of traps, interrupts etc. However, the power on sequence requires two 32 bit vectors for the initial stack pointer and program counter. The 68000 expects these 8 bytes at memory locations zero (0) through 7 and they should be stored in ROM to ensure their validity on power up.

Switch **S2-1** ("EV", Enable for Vectors) determines whether the vectors are read from on-board ROM or off-board memory (usually ROM). If switch **S2-1** is **ON** the on-board ROMs will be accessed on a read from the exception vector space. If switch **S2-1** is **OFF**, the vectors will be read from off-board memory.

Switch **S2-2** ("XV", eXception Vectors) determines which exception vectors enable the ROMs. **S2-2** in an **ON** position enables the ROMs only for the two reset vectors. **S2-2** in an **OFF** position enables the ROMs for the first 64 exception vectors. Normally **S2-2** is kept in an on position. However, for some dedicated applications it may be desirable to store the many system exception vectors (divide by zero, trap, interrupt etc.) in ROM.

If one desires to store exception vectors in an off-board ROM (i.e. **S2-1** off) **S1** determines the new starting address of the vectors.

If the address translation feature is not desired, **S1** should be set to all zeros. In this case the address appearing on the bus is identical to the processor's address lines. The user must not disable this feature unless non-volatile (and previously set) memory resides in the first 8 bytes of memory.

Switch 2

	OFF	ON
1) "EV"	Read vectors from OFF-BOARD memory at S1 address	Read vectors from ON-BOARD ROMS
2) "XV"	Enable for ALL system vectors.	Enable ONLY for reset vectors.
3) "R"	ROMs for vectors only.	Enable ROMs when reading from address space set by switch 1.
4)	Unused.	Unused.

Summary

The possible configurations are:

S2-1	S2-3	Effect
OFF	OFF	Read exception vectors from off-board memory starting at S1 address.
OFF	ON	Read exception vectors from off board memory and program starting at S1 address from ROM.
ON	OFF	Read vectors only from ROM.
ON	ON	Read vectors and programs from ROM. Program starts at S1 address.

For each of these configurations, if S2-2 is ON then "vectors" only means the first two boot vectors, otherwise all the system vectors (the first 64) are referred to.

Note that even though the program address space starts at the S1 address, you must not overlap the program with the exception vectors. If S2-2 is ON then the program can start at location 08H, if S2-2 is OFF then the program must start after location OFFH.

Exception Vector Assignment

Vector Number(s)	Address Dec	Address Hex	Address Space	Assignment
0	0	000	SP	Reset: Initial SSP
	4	004	SP	Reset: Initial PC
2	8	008	SD	Bus Error
3	12	00C	SD	Address Error
4	16	010	SD	Illegal Instruction
5	20	014	SD	Zero Divide
6	24	018	SD	CHK Instruction
7	28	01C	SD	TRAPV Instruction
8	32	020	SD	Privilege Violation
9	36	024	SD	Trace
10	40	028	SD	Line 1010 Emulator
11	44	02C	SD	Line 1111 Emulator
12*	48	030	SD	(Unassigned, reserved)
13*	52	034	SD	(Unassigned, reserved)
14*	56	038	SD	(Unassigned, reserved)
15*	60	03C	SD	(Unassigned, reserved)
16-23*	64	040	SD	(Unassigned, reserved)
	95	05F		—
24	96	060	SD	Spurious Interrupt
25	100	064	SD	Level 1 Interrupt Autovector
26	104	068	SD	Level 2 Interrupt Autovector
27	108	06C	SD	Level 3 Interrupt Autovector
28	112	070	SD	Level 4 Interrupt Autovector
29	116	074	SD	Level 5 Interrupt Autovector
30	120	078	SD	Level 6 Interrupt Autovector
31	124	07C	SD	Level 7 Interrupt Autovector
32-47	128	080	SD	TRAP Instruction Vectors
	191	0BF		—
48-63*	192	0C0	SD	(Unassigned, reserved)
	255	0FF		—
64-255	256	100	SD	User Interrupt Vectors
	1023	3FF		—

*Vector numbers 12 through 23 and 48 through 63 are reserved for future enhancements by Motorola. No user peripheral devices should be assigned these numbers.

Figure 4. Exception Vector Assignment

Level 1 interrupt autovector: VI5
Level 2 interrupt autovector: VI4
Level 3 interrupt autovector: VI3
Level 4 interrupt autovector: VI2
Level 5 interrupt autovector: VI1
Level 6 interrupt autovector: VI0
Level 7 interrupt autovector: NMI

The SYSTEM vectors are vector numbers 0 through 63, at addresses 0 through 0FF (255).

What happens on Power Up

After power up the 68000 loads the system stack pointer and program counter from the first two exception vectors. These two 32 bit vectors are stored in the least significant eight bytes of memory. Since these vectors are required when power is first applied, they should be stored in ROM. In this example, the program counter vector points to location 020008H which is the first instruction in the program in ROM, (after the boot vectors).

If you wish to modify the monitor, you could copy the contents of the ROMS into another memory board, preferably non-volatile RAM. (To read the ROMS, simply read from locations 20000 through 21FFF.) Then you can modify the copy in RAM. To execute the new version you must relocate the RAM to location 20000 and set S2-1 and S2-3 to OFF, so the monitor and the boot vectors are read from the RAM. A sample program for a block move is listed in Appendix D.

Format of Data Stored in ROMs

Since the ROMs support word transfers, sequential addresses are stored in alternate ROMs. That is, one ROM (U2) holds the low byte of each word and the other (U1) holds the high byte of each word.

NOTE:

If S2-3 is ON (so that the S1 address is mapped to the ROMS) you must make sure that no other memory lies in the address space of the 64K block of memory starting at the S1 address.

ADDRESS BUS

The processor board supports an extended 24 bit address bus. This allows the CPU to directly address up to 16 megabytes of memory. Such a vast address space eliminates the need for cumbersome bank select schemes. Older boards responding to only the 16 bit address bus may be used with this CPU but this would restrict the total system address space to 64 kilobytes.

I/O Space vs. Memory Space

The 68000 instruction set does not have an explicit Input/Output instruction. Motorola architects intended for all 68000 I/O to be memory mapped. Memory mapped I/O takes advantage of the many powerful addressing modes for fast, efficient I/O routines.

To support S-100 I/O mapped peripherals the processor board dedicates the most significant 64 kilobytes of memory to I/O. As a result, any memory access to hex address FF0000 through FFFFFF results in an I/O bus cycle. That is, such an access asserts status outputs sINP or sOUT. This configuration allows efficient memory mapped software while maintaining full compatibility with existing I/O devices.

For example, hex address FF0002 corresponds to I/O port with address 02. So the 68000 instruction:

```
MOVE.B    0FF0002H,D0
```

is similar to the 8080 instruction:

```
IN        02H
```

Note that 64 kilobytes of address space are dedicated to I/O devices. This allows over 64 thousand input and output ports. To support this many ports requires that I/O devices decode the least significant 16 address bits. The IEEE specification allows extended I/O addressing but does not require it.

The majority of current I/O boards decode only the least significant 8 address bits. This gives 256 input and output ports. The processor board can be used with such an I/O device. Since the I/O board does not decode the full 16 bit I/O address its ports address is replicated throughout the 64 kilobyte I/O address space. The processor board functions quite well with existing I/O boards and is capable of fully supporting future extended I/O address boards as well.

If you are using an I/O board which only decodes the low eight bits of the address then you can use the 16-bit word addressing mode of the 68000. Since to the I/O board address 0FF0002 is indistinguishable from 0FFFF02, and the 68000 sign extends the word long address, you can also use the address

OFF02. So the above example could also be coded:

```
    MOVE.B    OFF02.W,D0
```

A0

The 68000 address bus directly drives A1 through A23. The CPU/68000 comes factory jumpered for the updated IEEE-696 standard. That is, the most significant byte of each word is stored at an even address and the least significant byte is stored at the next odd address. Note that instructions, operands, stack data, address vectors etc. are all stored at even addresses.

The definition of A0 may be reversed by **carefully** cutting the trace marked LO (Low Odd) and installing a jumper to the pad marked LE (Low Even).

DATA BUS

The 68000 transfers data over a single 16 bit bidirectional bus. **Programs must reside in 16 bit memory**, however, data bytes may be accessed from byte wide memory. Long words must be transferred in sequential 16 bit bus cycles. Byte data is transferred over the corresponding data lines; high order (even address) bytes on D15-D8, low order (odd address) bytes on D7-D0.

The S-100 bus has two 8 bit data paths, Data Odd and Data Even). For byte transfers data is sent over the Data Even bus for write operations and over the Data Odd bus for read operations. For word transfers Data Even and Data Odd are ganged, forming a 16 bit bidirectional bus. During word bus cycles the even (A0=0) byte is transferred over the Data Even bus and the odd (A0=1) byte over the Data Odd bus. On the 68000 the even byte is most significant (D15-D8). If you have changed the A0 jumper on the CPU board then these definitions are reversed.

TEMPORARY BUS MASTER INTERFACE (TMA CONTROL)

The 68000 processor board functions as a permanent bus master as specified in the IEEE proposed S100 standard. Temporary bus masters (DMA devices) request the bus by asserting control input HOLD. They receive control of the bus when the bus master (68000 CPU) asserts control output hold acknowledge (pHLDA).

Upon receipt of HOLD the 68000 completes the current bus cycle and then asserts pHLDA. The 68000 suspends all processing until HOLD is released. A temporary master may now disable the permanent bus masters address, data, status and control buses by asserting the four disable lines ADSB, DODSB, SDSB and CDSB. The temporary master now has complete control of the bus for as long as it wishes. When the bus is no longer needed control is returned to the permanent master by releasing the bus disable signals and finally, releasing HOLD.

The method of transferring the bus from the permanent bus master to a temporary master is explicitly specified in the IEEE bus standard section 2.8. Of significance is the method used to transfer ownership of the control output bus. To ensure glitch free transfer, both the permanent and temporary master drive the control output bus during the transfer period. Except for pHLDA, all lines are driven at their non-asserted levels. After a specified time (125 nanoseconds) the temporary master asserts CDSB, disabling the permanent master,s control output bus drivers and acquiring control of the bus.

Up to 16 temporary masters may coexist in a system. A distributed arbitration scheme determines the highest priority device which then takes control of the bus upon assertion of pHLDA.

In general, the 68000 will relinquish control of the bus after the current bus cycle. However, if HOLD is received just before the start of a bus cycle, the 68000 will go ahead with the bus cycle, relinquishing control after its completion.

The 68000 instruction TAS (Test And Set) results in different CPU timing than other instructions. Motorola defines it as a read-modify write cycle. The instruction results in sequential read and write cycles on the S-100 bus. The two cycles are indivisible, that is, the write cycle must follow the read cycle. This type of instruction allows meaningful communications within a multiprocessor or multiprocessing environment. TAS is designed to prevent transfer of bus control until the entire instruction has completed execution. Note that two distinct S-100 cycles are completed, but no interrupts or bus requests will be accepted until the second cycle has completed.

INTERRUPTS

The 68000 has a powerful internal interrupt controller. There are seven levels of interrupt priority. All except the non-maskable interrupt are software maskable via the system status word.

The processor board is configured to accept seven of the S-100 interrupt signals, VI5 through VI0 and NMI, where VI5 has the lowest priority. Note that NMI will always generate an interrupt when asserted. VI6 and VI7 are not supported. The S-100 interrupt signals correspond to the MC68000 IPL interrupt levels as follows:

S-100 definition:	VI5	VI4	VI3	VI2	VI1	VI0	NMI
68000 CPU notation:	IP1	IP2	IP3	IP4	IP5	IP6	IP7

After receiving an interrupt with priority greater than that specified by the system status word, the 68000 loads the program counter from the appropriate exception vector (a 32-bit address) and begins execution of the interrupt routine. The seven autovectors are vector numbers 25 through 31 (decimal) and reside at locations 100 through 124 (hex). No interrupt acknowledge cycle is needed.

Appendix A

Selecting ROMS

The ROM type is selected by jumpers on H1. ROMs supported are the 2716, 2732, 2516, and 2532. The CPU comes configured for use with 2732 ROMs. Following is a diagram of H1:

```
G  P  P  P          ROM pins
N  1  2  2
D  8  0  1

*  *  *  *


*  *  *  *

A  A  E  +          EN is active low
1  1  N  5
3  2     V
```

Examples:

2716 2732 2516 2532

Appendix B

Details of the S-100 bus Interface for the 68000

FUNCTION OF M1

Status signal sM1 is asserted during any program (as opposed to data) fetch. Historically, sM1 indicated that the current bus cycle would require four clock periods instead of three clock periods. The extra clock period, required for instruction decode, allowed time to refresh dynamic memory. With the 68000, no assumption can be made about the length of a bus cycle based on the level of sM1.

SIXTN Line

The CPU/68000 does not support seqential byte operations to implement a sixteen-bit data transfer. Therefore it has no need for the SIXTN line on the S-100 bus and it is ignored.

Special Configurations

Faster Memory Access When Used with Dual Systems Memories

When the CPU/68000 is used with the Dual Systems line of FAST CMEM (Rev. B and later) memories, memory cycle time is decreased by 25%. This allows the CPU/68000 to run at absolutely full speed with no CPU wait states. This increased speed is possible through the use of an asynchronous bus transfer protocol. When the CPU commences a memory cycle, the CMEM memories respond to a valid address on the bus by asserting a manufacturer-definable line (#66) called FASTACK* and either gates data onto or latches the data from the data bus. Immediately after the CPU detects that FASTACK* has been asserted, the processor completes the cycle.

If the memories being accessed do not respond with FASTACK* a standard S-100 bus cycle is completed. Thus, both Dual FAST CMEM and regular 16 bit S-100 memories may be used in the same system.

The CPU/68000 must have the pins labled "FAST" and "66" jumpered together to enable fast mode.

Using the Phantom Line for System Protection

The 68000 is always in one of two modes: system mode or user mode. When in user mode, it is usually desirable to not allow the user access to anything which might impair the integrity of the operating system or file system.

The CPU/68000 is capable of supporting a simple protection scheme. Install a jumper between the pads marked "USER" and "P" (Phantom). When this jumper is installed, the Phantom line will be asserted whenever the CPU is in user mode. Then any I/O (especially disks) which should only be acessed when in system mode can be set to disable themselves when the Phantom line is asserted. In addition, memory that should only be seen read or changed by the operating system directly, can also be set to be disabled when the phantom line is asserted.

Appendix D

A Few Utility Programs

This program performs a block move, enter it with:

```
A0   Starting address of source
A1   Starting address of destination
A2   Last address to move + 1

0000       32D8           LP1: MOVW A0@+,A2@+       MOVE A WORD
0002       B1CA                CMPL A2,A0           DONE?
0004       6DFA                BLTS LP1             REPEAT
0006       4EF9 0002 00D8      JMP /200D8           RETURN TO MACSBUG
```

This fills a block with a word.

```
A0   ADDRESS of word to fill with
A1   Starting address of block
A2   Last address of block + 1

0000       32D0           LP2: MOVW A0@,A1@+        MOVE A WORD
0002       B3CA                CMPL A2,A1           DONE?
0004       6DFA                BLTS LP2             REPEAT
0006       4EF9 0002 00D8      JMP  /200D8          RETURN TO MACSBUG
```

For testing hardware with a scope, this repeatedly sends a byte to any address (could be an I/O port). Sends the byte in D0 to the address pointed to by A0.

```
           1080           LP3: MOVB D0,A0@
           60FC                BRAS LP3
```

This reads from the address in A0 and puts the result in D0.

```
           1010           LP4: MOVB A0@,D0
           60FC                BRAS LP4
```

All of these routines are relocatable. They can be entered into any free area of memory (such as 2000) with the MACSBUG OP command. The entry parameters can be directly placed in the registers, and the routine executed with the G command.

MACSBUG OPERATING INSTRUCTIONS

1. INTRODUCTION

This document describes the operation of the MACSbug monitor after it has been installed. It includes a complete description of all the commands and examples of its use.

2. OPERATIONAL PROCEDURE

After the CPU/68000 board has been installed, as per the manual, the user should perform the following:

a. Turn power ON to the system.

b. Depress the RESET (black) button.

The system should initialize and print:

```
MACSBUG 1.31
*
```

If these two lines do not print out, go back and check the CPU/68000 manual. Check especially that the terminal and I/O board have the same BAUD rates.

3. COMMAND LINE FORMAT

Commands are entered the same as in most other buffer organized computer systems. A standard input routine controls the system while the user types a line of input. The delete (RUBOUT) key or control 'H' will delete the last character entered. A control 'X' will cancel the entire line. Control 'D' will redisplay the line. Processing begins only after the carriage return has been entered.

During output to the console the control 'W' will suspend the output until another character is entered. The BREAK key will abort most commands.

The format of the command line is:

*COmmand parameters; options

where: * — is the prompt from the monitor. The user does not enter this. In the examples given, the lines beginning with this character are lines where the user entered a command.

CO	is the necessary input for the command. Each command has one or two upper case letters necessary in its syntax. In the examples, the entire command may be used, but only those letters in upper case in the syntax definition are necessary.
mmand	is the unnecessary part of the command. It is given in the syntax definition only to improve readability. If this part of the command was actually entered on the command line, it would be ignored.
parameters	depends upon the particular command. Data is usually in hex but most printable ASC11 characters may be entered if enclosed in single quotes. The system also supports a limited symbolic feature allowing symbols to be used interchangeably with data values.
;options	modifies the nature of the command. A typical option might be to disregard the checksum while reading a file.

Note: MACSbug requires all commands to be entered in upper case letters. If lower case letters are used, MACSbug will respond with

```
WHAT?
*
```

4. EXAMPLE OF COMMAND PROCEDURES

```
MACSBUG 1.0                      Power up or reset condition
*P2                              MACSbug prompts with '*' user enters P2 to enter
                                    transparent mode.  (see page 3-19)
*TRANSPARENT*                    Message printed to indicate user is now directly
                                    connected with host system
```

User may now communicate directly with the host system. Typing a control A at any time will exit to MACSbug.

```
(Control A)
*MACSBUG*                        Message put out by MACSbug to indicate user is
                                 now in MACSbug command mode
*READ ;=COPY FILE.MX,#CN         Download from EXORciser host
*DM 1000                         Display memory
001000 70 01 70 02 70 03 70 04 70 05 4E F8 10 00 FF FF  p.p.p.p.p.N.....
*PC 1000                         Set program counter to START
*TD CLEAR                        Clear the trace display
*TD PC.22 DO.1                   Specify which registers to print in display
*TD                              Print the trace display
PC=1000 DO=00
*BR 1004                         Set a breakpoint
*T TILL 0                        Trace command
PC=1002 DO=01
PC=1004 DO=02                    Stopped at breakpoint
:*GO
PC=1004 DO=02                    Stopped at breakpoint
*                                Program is ready to run
```

3.6 MACSbug COMMAND SUMMARY

COMMAND	DESCRIPTION	PAGE
reg#	Print a register	3-5
reg# hexdata	Put a hex value in the register	
reg# 'ASCII'	Put hex-equivalent characters in register	
reg#:	Print the old value and request new value	
class	Print all registers of a class (A or D)	
class:	Sequence through-print old value request new	
DM start end	Display memory, hex-ASCII memory dump	3-6
SM address data	Set memory with data	
OPen address	Open memory for read/change	3-7
SYmbol NAME value	Define and print symbols	3-8
W#	Print the effective address of the window	3-9
W#. len EA	Define window length and addressing mode	
M# data	Memory in window, same syntax as register	
Go	Start running from address in program counter	3-10
Go address	Start running from this address	
Go TILL add	Set temporary breakpoint and start running	
BReakpoint	Print all breakpoint addresses	
BR add: count	Set a new breakpoint and optional count	
BR —address	Clear a breakpoint	
BR CLEAR	Clear all breakpoints	
TD	Print the trace display	3-11
TD reg#. format	Put a register in the display	
TD Clear	Take all registers out of the display	
TD ALl	Set all registers to appear in the display	
TD A. 1 D. 1 L. c	Set register blocks or line separator	3-12
T	Trace one instruction	3-13
T count	Trace the specified number of instructions	
T TILL Address	Trace until this address	
:*(CR)	Carriage return-trace one instruction	
OFfset address	Define the global offset	3-14
CV decimal	Convert decimal number to hex	3-15
CV $hex	Convert hex to decimal	
CV value,value	Calculate offset or displacement	
REad ; =text	Expect to receive 'S' records	3-16
VErify ; =text	Check memory against 'S' records	
PUnch start end	Print 'S' records (tape image)	
FOrmat hex	Program/initialize an ACIA	3-17
NUll hex	Set character null pads	
CR hex	Set carriage return null pads	
TErminal baud	Set terminal null pads to default values	
CAll address	JSR to user utility routine	3-18
P2	Enter transparent mode	3-19
*..data..	Transmit command to host	
Break	The BREAK key will abort most commands	
CTL-A	The control A key ends transparent mode	
CTL-D	The control D key redisplays the line	
CTL-H	The control H key deletes the last character entered	
CTL-W	The control W key suspends output until another character is entered	
CTL-X	The control X key cancels the entire line	
Rubout	The RUBOUT key deletes the last character entered	
Del	The DEL key deletes the last character entered	

3.6.1 Set and Display Registers

68000 REGISTER MNEMONICS	DESCRIPTION
D0,D1,D2,D3,D4,D5,D6,D7	Data registers
A0,A1,A2,A3,A4,A5,A6,A7	Address registers
PC	Program counter
SR	Status register (condition codes)
SS	Supervisor stack pointer (A7 in supervisor mode)
US	User stack pointer (A7 in user mode)

COMMAND FORMATS	DESCRIPTION
reg# hexdata	Put a hex value into register 'reg#'
reg# 'ascii data'	Put ASCII value into register 'reg#'
reg#:	Print register value and take in new value
reg#	Print register value
class (where class=D or A)	Print values of all registers in the class
class:	Cycle through all registers in the class printing old value and requesting new value

EXAMPLES	COMMENTS
*A5 123	Set address register A5 to hex value 123
*A5	Command to print the value of register A5
A5=00000123	Computer response
*D4 FFFFFF	Set a data register
*D0:	Command to print old value and take in new value
D0=00000000=? 45FE	Computer prompts with old value; new value entered
*D:	Command to cycle through all data registers
D0=000045FE=? 9EAB3	Change value of register D0 from 45FE to 9EAB3
D1=00000000=? (CR)	Carriage return (null line) means the value remains the same.
D2=00000000=? (CR)	
D3=00000000=? (CR)	
D4=00FFFFFF=? (CR)	
D5=00000000=? 55555	Change register D5 to a new value
D6=00000000=? (CR)	
D7=00000000=? (CR)	
*D	Display all data registers

D0=0009EAB3 D1=00000000 D2=00000000 D3=00000000
D4=00FFFFFF D5=00055555 D6=00000000 D7=00000000

*PC:	Display and request input for program counter
PC=0008B3=? 2561	Set the program counter to new value
*SR 0	Set status register to zero (user mode)
*A7 4321	Set address register (same as US now)
*US	Display user stack pointer
US=00004321	
*SS 7FFF	Set supervisor stack pointer
*SR 2000	Set status register to supervisor mode
*A7	Print A7 which is now the SS register
A7=00007FFF	
*	

COMMAND FORMAT	DESCRIPTION
DM start end	Display Memory in hex and ASCII where start < end
DM start count	Where start > count
DM2 start end	Send output to PORT 2
SM address data	Set Memory to hex
SM address 'ASCII'	Set Memory to ASCII
SM address data N	The 'N' as the last character means start a new line; the system will prompt with the current address

EXAMPLES **COMMENTS**

```
*SM 2000 'ABC'                  Set memory to some ASCII data
*SM 2003 4445 46 'G'            Set some more locations
*DM 2000 2010                   Command to dump memory

002000 41 42 43 44 45 46 47 00 00 00 00 00 00 00 00 00  ABCDEFG.........
002010 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................

                In this version of the command the second number is smaller
                than the first so it is decoded as a count
*DM 2003 12
002003 44 45 46 47 00 00 00 00 00 00 00 00 00 00 00 00  DEFG............
002013 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................

*SM 1000 1 23 456 7890 ABCDE 12345678    Size can be up to 8 characters
*DM 1000
001000 01 23 04 56 78 90 0A BC DE 12 34 56 78 00 00 00  ................

*SM 1000        'TABLE    '  00005678 N       Use of the 'N' parameter to
                                                start a new line
0000100C ?      'START    '  00023456

*DM 1000 20
001000 54 41 42 4C 45 20 20 20 00 00 56 78 53 54 41 52  TABLE.....VxSTAR
001010 54 20 20 20 00 02 34 56 00 00 00 00 00 00 00 00  T.....4V........

*OFFSET 2030                    Global offset will be added to command parameters
*DM 1000
003030 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................

*SM 1005 1234 N                 Global offset added to address 1005
00003037 ? AB
*DM 1000
003030 FF FF FF FF FF 12 34 AB FF FF FF FF FF FF FF FF  ................

*SM 20000 AB CD EF              Trying to set ROM
ERROR                           Error message
*
```

3.6.3 Open Memory for Read/Change

COMMAND FORMAT	DESCRIPTION
OPen address	Open memory at specified address and enter subcommand mode

SUBCOMMAND FORMAT

ADDRESS	CONTENT	USER ENTERS	COMMENTS
*OP 1000			Open memory location 1000
001000	= FF ?	12	User enters data and system goes to next location
001001	= AB ?	(CR)	Carriage return means go to the next location
001002	= 44 ?	34↑	Up arrow means go to previous location
001001	= AB ?	↑	Can be entered without data
001000	= 12 ?	77=	Equal sign means stay at same address
001000	= 77 ?	=	Can be used without any data
001000	= 77?		Period means return to MACSbug
*			Returns to command level
*OP 1234			
021234	= FF ?	99=	Example of trying to change ROM
NO CHANGE			Warning message
021234	= FF ?		Does not abort command

COMMAND FORMAT	DESCRIPTION
SYmbol name hex value	Put a symbol in the symbol table with a hex value or assign a new value to a previously defined one. NAME can be 8 characters long, consisting of: A-Z, 0-9, (period), and $ (dollar sign). It must begin with letter (A-Z) or period.
SY −name	Remove a symbol from the symbol table
SY name	Print the current value of the symbol (absolute)
SY value	Print the first symbol with the given value
SY	Print the sorted symbol table

NOTE

Offset is not used by this command. Some commands recognize the words TILL, ALL, and CLEAR as key words and will not interpret them as symbols.

EXAMPLES	COMMENTS
*SY XYZ 5000	Puts the symbol in the table
*SY XYZ	Command prints out the symbol's current value
XYZ =5000	
*SY XYZ 123	Change a symbol's value
*SY ABC34 2500	Define another symbol
*SY Z17.RT5 XYZ	Define a symbol with value from another symbol
*SY 123	Print first symbol with value of 123
XYZ =123	
*SY B$67ABC 4300	Define some more symbols
*SY RFLAG 2300	
*SY MVP2 9990	
*SY	Print the sorted symbol table

```
ABC34    00002500    B$67ABC    00004300    MVP2       00009990
RFLAG    00002300    XYZ        00000123    Z17.RT5    00000123
```

*SY TTT	Print a value for symbol not in table, when not found, it tries to convert parameter to number
T IS NOT A HEX DIGIT	
*SY 567	Attempt to print value for symbol not in table
00000567=567	

SYNTAX EXAMPLES	COMMENTS
*BR MVP2	Set a symbolic breakpoint
*CALL RFLAG	User defined routine
*PC ABC34	Set a register
*DM MVP2 10	Display some memory

EXAMPLES OF KEY WORDS IN COMMANDS

*BR CLEAR	The word CLEAR is not considered a symbol here
*GO TILL Z17.RT5	The word TILL is part of the command
*T TILL ABC34	The word TILL is part of the command

3.6.5 Displaying and Accessing Memory through Windows

WINDOWS

A "window" is an effective address through which the user can "see" memory. The windows are labeled W0 to W7 and are defined using the syntax listed below. The windows address corresponding memory locations labeled M0 to M7 which use the same syntax as registers. These memory locations can be examined, set or defined in the display the same as a register.

COMMAND FORMAT	DESCRIPTION
W#	Print the effective address of a given window
W#. len EA	Define a window size and effective address # is the window number 0 to 7 len is the length in bytes 1=byte; 2=word; 3=3 bytes; 4=long word 0=close a window (undefine it) EA is Effective Addressing mode (see EA SYNTAX EXAMPLES in table below)
M# data or 'ASCII'	Pseudo registers have same syntax as registers

EA SYNTAX EXAMPLES	DESCRIPTION
FE84	Absolute address
(A6)	Address register indirect
100(A6)	Indirect with displacement
−10(A6,D2)	Indirect with index and displacement
−100(*)	Program counter with displacement
10(*,A4)	Program counter with index and displacement

EXAMPLES	COMMENTS
*W3. 4 (A6)	Define a window
*A6 2000	Enter a value for the address register
*W3	Print the effective address of a window
W3. 4 (A6)=2000	
*M3 87342	Set memory through the window
*M3	Command to print memory through the window
M3=00087342	
*DM 2000	Display a line of memory
002000 00 08 73 42 00 00 00 00	00 00 00 00 00 00 00 00 . . sB . . . .
*TD CLEAR	Clear all registers from the trace display
*TD PC. 2 A6. 3 M3. 1	Define some registers for the display
*TD	Command to print the trace display
PC=00A2 A6=002000 M3=42	NOTE: W3. 4 and M3. 1 only lowest byte displayed
*W3. 2 (A6)	Change width of window
*TD M3. 2	Change width of display
*TD	
PC=00A2 A6=002000 M3=0008	
W0. 1 10(,A6)	Define a new window: PC+A6+10
*W0	Print effective address of window W0
W0. 1 10(*,A6)=20B2	
*W3. 0	Close window W3, undefine it
*TD	
PC=00A2 A6=002000	Closed/undefined windows are not in the display

3.6.6 GO and Breakpoints

COMMAND FORMAT	DESCRIPTION
Go	Begin execution at address in PC register
Go address	Begin execution at this address
Go TILL address	Set a temporary breakpoint at the address and run until a breakpoint is encountered
BR	Print the address of all breakpoints (8 maximum)
BR address	Set a breakpoint at this address
BR −address	Remove the breakpoint at this address
BR address;count	Set a breakpoint at this address with a count
BR CLEAR	Remove all breakpoints

EXAMPLES	COMMENTS
(see example program on page 3-3)	
*PC 1000	Set program counter to starting address
*TD CLEAR	
*TD PC. 2 D0. 1	Set trace display format
*TD	Print trace display
PC=1000 D0=00	
*G TILL 1008	Run until address
PC=1008 D0=04	System displays when it stops
*BR 1002	Set a breakpoint
*G	Run until breakpoint
PC=1002 D0=01	Trace display
*BR 1008: 4	Set a breakpoint with a count
*BR	Print the breakpoints
BRKPTS= 1002 1008: 4	
*G	Run
PC=1000 D0=4	Decrements count, prints display, continues
PC=1002 D0=1	Stops at breakpoint with zero count
*BR	Print the breakpoints
BRKPTS= 1002 1008: 3	Count has been decremented by one
*BR −1002	Remove a breakpoint
*G	Run
PC=1000 D0=4	Count from 3 to 2. . .
PC=1008 D0=4	. . . 2 to 1 . . .
PC=1008 D0=4	. . . 1 to 0 and it stops here
*BR	Print the breakpoints
BRKPTS= 1008	No count for this breakpoint
*BR 1000	Set another breakpoint
*G 1000	Start running from 1000, bypass breakpoint at starting address and stop at next breakpoint
PC=1008 D0=4	
*SY JUMPER 100A	Define a symbol
*BR JUMPER: 5	Set a breakpoint at a symbolic address
*BR 123456: 7897 11 22 33 44 55 66	Try to overflow table (holds 8)
TABLE FULL BRKPTS= 1008 1000 100A: 5 123456: 7897 11 22 33 44	
*OFFSET 3000	
*BR CLEAR	
*BR 50	When setting breakpoints the global offset is added to the parameter but all addresses printed are absolute
*BR	
BRKPTS= 3050	

3.6.7 Set the Trace Display Format (Individual Registers)

COMMAND FORMAT	DESCRIPTION
TD	Print the trace display
TD CLear	Take everything out of the display
TD ALl	Put all registers in display (see page 3-12)
TD reg#. format	Add or delete registers in display where reg# is D0-D7, A0-A7, W0-W7, M0-M7, PC, SR, US, SS, A, D, or L (see page 3-12 for A,D,L) . format can be 0,1,2,3,4,Z,D,R, or S 0=remove the item from the display 1,2,3,4=print this number of bytes as hex characters, include all leading zeros Z=signed long word hex with zero suppress D=signed long word decimal with zero suppress R=subtract offset (see OFfset command) then print with Z format with letter 'R' at end S=search symbol table for 4 byte value, if found print symbol name as 8 characters, if not found print hex value as 8 characters

EXAMPLES	COMMENTS
*TD CLEAR	Turn off all the registers in display
*TD PC. 3 D1. 1	Define PC as 3 bytes and D1 as one
*TD	Command to display
PC=000000 D1=05	This is the trace display
*TD PC. 0 A6	Remove PC and add A6 which defaults to 4 bytes
*TD	Display
D1=05 A6=0000008F	Display with two new registers
*W3. 2 2000	Define a window
*M3 20	Set value of memory pseudo register
*TD M3. 2	Add a memory pseudo register to the display
*TD	Display
D1=05 A6=0000008F M3=0020	New display
*TD A6. 1 D1. 3 M3. Z	Change length of registers already in display
*TD	Display
D1=000005 A6=8F M3=20	New display, M3 now suppresses leading zeroes
*TD D1. R M3. D	D1 is relative and M3 is decimal
*OFFSET 12345	Set the offset (see OFfset command)
*TD	Display
D1=-12340R A6=8F M3=32	5-offset=-12340R; 20 hex = 32 decimal
*SY TABLE 8F	Define a symbol (see SYmbol command)
*TD A6. S M3. 0	Make A6 print symbol if value is in table
*TD	
D1=-12340R A6=TABLE	Prints symbolic value
*A6 123	Set A6 to a value NOT in symbol table
*TD	
D1=-12340R A6=00000123	A6 prints value with 4 byte format

3.6.8 Set the Trace Display Format (Blocks of Registers)

COMMAND FORMAT	DESCRIPTION
TD CLear	Take everything out of the display
TD D. 1	Put all data registers in display as a block
TD A. 1	Put all address registers in display as a block (for D. 1 and A. 1 the format is fixed at 4 bytes)
TD L. character	Define a line separator at the end of display (.0 will reverse A. 1, D. 1, and L char commands)
TD ALl	Same as keying in: *TD PC. 3 SR. 2 US. 4 SS. 4 D. 1 A. 1 L. – does not affect other registers and windows that have been previously defined to display

EXAMPLES	COMMENTS

```
*TD CLEAR                       Clear the display
*TD D. 1                        Define all data registers in a block
*TD                             Print the trace display
D0=00000000 D1=00000000 D2=00000000 D3=00000000
D4=00000000 D5=00000000 D6=00000000 D7=00000000
*TD CLEAR
*TD A. 1                        Define all address registers in a block
*TD
A0=00000000 A1=00000000 A2=00000000 A3=00000000
A4=00000000 A5=00000000 A6=00000000 A7=00007FFE
*TD L. @                        Define a line separator (a row of '@')
*TD
A0=00000000 A1=00000000 A2=00000000 A3=00000000
A4=00000000 A5=00000000 A6=00000000 A7=00007FFE
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
*TD L. &                        Define a line separator (a row of '&')
*TD
A0=00000000 A1=00000000 A2=00000000 A3=00000000
A4=00000000 A5=00000000 A6=00000000 A7=00007FFE
&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
*TD ALL                         Turn on commonly used registers. . .
*TD                               . . . this is also the default or reset condition
PC=000000 SR=2000 US=00007F00 SS=00007FFE
D0=00000000 D1=00000000 D2=00000000 D3=00000000
D4=00000000 D5=00000000 D6=00000000 D7=00000000
A0=00000000 A1=00000000 A2=00000000 A3=00000000
A4=00000000 A5=00000000 A6=00000000 A7=00007FFE
------------------------------------------------
*
```

3.6.9 Tracing **TRACE**

COMMAND FORMAT	DESCRIPTION
Trace	Execute one instruction and print trace display
Trace count	Trace specified number of instructions
Trace TILL address	Trace to the given address (breakpoint will stop the trace)
:* (CR)	A colon (:) before the prompt indicates a special trace mode is in effect, a carriage return will trace the next instruction

EXAMPLES	COMMENTS
(see example program on page 3-3)	
*DM 1000	Example program in memory
001000 70 01 70 02 70 03 70 04 70 05 4E F8 10 00 FF FF p.p.p.p.p.N.	
*PC 1000	Set the program counter
*TD	Print the trace display
PC=1000 D0=00	
*T	Trace one instruction
PC=1002 D0=01	
:* (CR)	Special prompt appears, carriage return will trace the next instruction
PC=1004 D0=02	
:*T 3	Trace three instructions
PC=1006 D0=03	
PC=1008 D0=04	
PC=100A D0=05	
:*T TILL 1004	Trace till instruction at address 1004
PC=1000 D0=05	
PC=1002 D0=01	
PC=1004 D0=02	
:*	

3.6.10 Offset

The 68000 instruction set lends itself to relocatability and position independence. A general purpose, global offset feature has been provided. The single offset address applies to all of the commands listed below. Registers displayed in the trace display may have the offset subtracted by using 'R' as the format. See paragraph 3.6.7 on trace display.

The offset may be overriden by entering a comma and alternate offset. All commands do not use the offset but any number can be forced to be relative (have the offset added) by entering an 'R' as the last character of the number.

WARNING: This is a very simple offset feature and may not be able to solve complex relocation problems. The user is encouraged to experiment with the global offset and the window features to determine their limitations and usefulness in a particular application.

COMMAND FORMAT	DESCRIPTION
OFfset	Display offset
OFfset hex value	Set the offset to a given value
OFfset 0	Set the offset to zero — begin absolute addressing
command data,alternate	Disregard offset, add alternate offset to data
command data,	Data is absolute, no offset added
command data,0R	Used in commands that do not normally use offset, adds offset to data

The offset affects the following commands:

TD reg.R	Trace display, subtract offset from register value
BReakpoint	Set breakpoint (display is in absolute)
Go	All addresses
SM	All addresses
DM	All addresses (display is in absolute)
PUnch	All addresses
REad	All addresses

EXAMPLE	COMMENTS
*PC 2010	Set the program counter
*TD PC.R	Set trace display .R means hex long word minus offset
*TD	Display
PC=2010R	Displayed relative to offset (zero now)
*OF 2000	Set the offset to 2000
*TD	Display
PC=10R	PC – offset = 2010–2000 = 10 Relative
*BR 6	Set a breakpoint: hex data+offset = 6+2000 = 2006
*BR	Display breakpoint
BRKPTS=2006	Breakpoints are always displayed as absolute hex
*BR 24,3000	Set a breakpoint with alternate offset 24+3000
*BR	
BRKPTS=2006 3024	

3.6.11 Number Base Conversion

COMMAND FORMAT	DESCRIPTION
CV decimal	Decimal to hex conversion
CV $hex	Hex to decimal conversion
CV symbol	Use value from symbol table
CV value,offset	Calculate offset or displacement

NOTE

This command DOES NOT automatically use the global offset. The default base for this command only is decimal. All numbers are signed 32 bit values.

EXAMPLES	COMMENTS
*CV 128	Command to convert decimal to hex
$80=&128	Computer response
*CV $20	Hex to decimal
$20=&32	
*CV −$81	Negative numbers
$FFFFFF7F=−$81=−&129	
*CV $444, 111	Adding an offset (second number's base defaults to first number's)
$555=&1365	
*CV $444, −111	Subtracting an offset (forward displacement)
$333=&819	
*CV $111, −444	Backward displacement
$FFFFFBBC=−$333=−&819	
*SY TEN &10	Defining a symbolic decimal constant
*SY THIRTY &30	
*CV TEN	Command can be used with symbols
$A=&10	
*CV −TEN	
$FFFFFFF6=−$A=−&10	
*CV THIRTY, −TEN	
$14=&20	
*OF 2000	Define a global offset
*CV $123R	'R' at the end of a number means add the global offset
$2123=&8483	
*CV TEN,0R	Symbolic relative
$200A=&8202	

3.6.12 Upload, Download and Verify **LOAD**

COMMAND FORMAT	DESCRIPTION
REad ;−CX =text	Load 'S' records-default PORT 2 option −C means ignore checksum; option X means display data being read; if equal sign is used in this command line everything after it is sent to PORT 2
VErify ;=text	Verify memory with 'S' records-print difference; verify does not use checksum
PUnch add. add.	Write 'S' records between address range
PU address count	Write specified number of bytes where count < address

NOTE

These commands use the offset. No attempt is made to control the host transmissions. For the REad and VErify, any line received not beginning with an 'S' is ignored. If an error occurs causing the system to take the time to print out an error message, one or more lines sent during the error message may have been ignored.

EXAMPLE **COMMENTS**

(See example program on page 3-3)

```
*READ ;=COPY FILE. MX,#CN           Download from an EXORciser
*DM 1000 10                         Check to see if data was loaded
001000 70 01 70 02 70 03 70 04 70 05 4E F8 10 00 FF FF p.p.p.p.p.N. . . . .
*VERIFY ;=COPY FILE. MX,#CN         Normal verify returns with prompt
*SM 1005 FF                         Deliberately change memory to show verify
*DM 1000                            Verify that 03 was changed to FF
001000 70 01 70 02 70 FF 70 04 70 05 4E F8 10 00 FF FF p.p.p.p.p.N. . . . .
*VERIFY ;=COPY FILE. MX,#CN
S1111000      03                    Displays only nonmatching data bytes
*RE ;=COPY FILE1. MX,#CN            Example of file with bad character
S11110007001700270/3700470054EF8100049 NOT HEX=/
*RE ;=COPY FILE2. MX,#CN            Example of file with bad checksum
S1111000700170027003700470054EF8100039 CHKSUM=49
*RE ;=COPY FILE. MX,#CN             Normal read returns with prompt
*PUNCH 1000 D                       Print 'S' records on console
S0010000FE                                              Header
S1111000700170027003700470054EF8100049                  Data with address of 1000
S9120000A4                                              End-of-file
*OF 1000                            Define a global offset
*PUNCH O D, 0
S0010000FE                                              Header
S1110000700170027003700470054EF8100049                  Data with address of zero
S9120000A4                                              End-of-file
*OF 5423
*RE ;=COPY FILE. MX,#CN             Download with offset
*DM 1000                            Display memory, adds offset to parameters
006423 70 01 70 02 70 03 70 04 70 05 4E F8 10 00 FF FF p.p.p.p.p.N. . . . .
```

There are two serial ports numbered 1 and 2. The following commands may program a specific port or if a port number is not used in the command, both ports will be set by the command.

For port commands shown below, '#' may be either 1 for PORT 1 (console), or 2 for PORT 2 (host). If the '#' field is left blank, the command applies to both ports.

COMMAND FORMAT		DESCRIPTION
FO#	hex	FOrmat — initialize ACIA (default=$15 = 8 bit words, no parity, 1 stop bit, and clock/16)
NU#	hex	NUll pads; nulls sent after each character
CR#	hex	Carriage return null pads sent after each CR
TE#	baud	TErminal format; set NU and CR null parameters for TI 700 series terminals

BAUD	NU	CR	
110	0	0	(default)
300	0	4	
1200	3	$17	
2400	7	$2F	

NOTE

The TE command does not change the hardware BAUD rate.
Port BAUD rates are changed by switches on the serial I/0 board.

EXAMPLE	COMMENTS
*NU1 5	Set character null padding on PORT #1 to 5 nulls
*NU	Print out current NU parameters
NU1=5 NU2=0	Zero is the default at system restart
*TE2 1200	Set PORT #2 to 3 character nulls and $17 CR nulls
*NU	Print null parameters . . . the NU and CR parameters for PORT #2 were set by the TE2 command
NU1=5 NU2=3	
*CR	
CR1=0 CR2=17	
*TE 2400	Change both ports to 2400 baud null pattern
*CR	Print the CR parameters
CR=2F	If both ports have the same parameter, one number is printed
*NU	
NU=7	
*NU 8	Change null values for both ports
*NU	When no port #specified, both ports are changed.
NU=8	
*CR2 FF	Send $FF nulls to PORT 2 (host) after every carriage return, this is the maximum value
*	

3.6.14 The CALL Command and Adding Commands to MACSbug **CALL**

There are two ways for the user to add commands. The simplest way is for the user to write the new command as a subroutine which ends with an RTS. The user can then use the CAll command.

This command does not affect the user's registers and is not to be confused with the GO command. The user may use a symbol as the command parameter instead of an absolute starting address. Registers A5 and A6 point to the start and end of the I/O BUFFER (see RAM equate file listing, paragraph 3.11) so the user may pass additional parameters on the command line.

COMMAND FORMAT	DESCRIPTION
CAll address	JSR to user subroutine, routine ends with RTS

EXAMPLE	COMMENTS
*CALL 3000 23 45 ZZ	JSR to user routine at location 3000 note that 23 45 & ZZ may be additional parameters that the user's subroutine will decode and are ignored by MACSbug
*SY FIXUP 2300	Define a symbol as absolute address 2300
*CALL FIXUP	JSR to symbolic address

The second method of adding commands involves MACSbug's command table. There is a RAM location CMDTABLE that is MACSbug's pointer to the start of the command table. The user may wish to copy this table into RAM, add his own commands or change the names of the existing ones, and change CMDTABLE to point to the new table.

The format of the table is very simple. Each command occupies six bytes in the table. The first two bytes are the command name and the next four bytes are the starting address of the code. The commands are not subroutines and all end by reentering the command decoder routine. The last entry in the table has $FFFF as the two byte name.

There are two special characters that may be used in the name field. The '@' means that the command must contain an ASCII digit from 0 to 7 in that character position. The'*' is a wild character that will match anything. For example, the use of the wild character '*' must follow after and not before a similar command, such as 'TE' then 'T*'in the table.

COMMAND FORMAT	DESCRIPTION
P2	Enter transparent mode: Transparent mode sends all characters typed at the terminal to the host computer. All transmissions from the host are typed on the local terminal. For this mode to work properly, the BAUD rate of the host connection MUST be slower than than the terminal.
(control A)	Control 'A' ends the transparent mode
*. . . data. . .	Asterisk, '*', as the first character of the console input buffer means transmit the rest of the buffer to the host (PORT2), the BAUD rates DO NOT have to be the same

EXAMPLES	COMMENTS
MACSBUG 1.0	Start up or reset condition
*P2	Command to enter transparent mode (NOTE: the BAUD rate of the host must be slower than the terminal)
TRANSPARENT	MACSbug prints this

User talks directly to the host, uses the editor, assembler, etc.

(CONTROL A)	Ends the transparent mode
MACSBUG	MACSbug prints this and system is ready for new command
**MAID	System prompts with '*' and user enters '*MAID'
**E800; G	(NOTE: the BAUD rates DO NOT have to be the same)

3.7 I/O SPECIFICATIONS

Provision has been made for the user to substitute his own I/O routines and direct the I/O for some commands to these routines. There are three pairs of locations in RAM that hold the addresses of the I/O routines. (See paragraph 3.11 on the equate file of RAM locations used by MACSbug.) They are initialized when the system is reset to contain the addresses of the default ACIA routines in ROM.

INPORT1 and OUTPORT1 are defaulted to ACIA #1 (PORT 1) which is the system console. The system prompt, command entry, all error messages, and all other unassigned I/O use these addresses to find the I/O routines. Most commands do not need a port specifier to use PORT 1. The REad and VErify commands, however, default to PORT 2.

INPORT2 and OUTPORT2 are defaulted to ACIA #2 (PORT 2) which is the host system (an EXORciser or timesharing system, etc.). Output or input is directed to this port by including a port specifier in the command field of the command line.

For example: *PU2 1000 50

The 2 in the command PU2 specifies that the addresses for the I/O routines will be found in the RAM locations INPUT2 and OUTPUT2. Error messages, however, will be printed on PORT 1 — the system console.

INPORT3 and OUTPORT3 are initialized to the same routine addresses as PORT 1 when the system is reset. The user can insert the addresses of his own I/O routines into these locations. I/O can then be directed to his configuration by using a 3 in the command field.

EXAMPLES OF COMMANDS WITH PORT SPECIFIERS:

*READ3 ; −C	Memory load from PORT 3; checksum ignored
*VERIFY1	Verify memory with 'S' records coming in from PORT 1
*PUNCH2 5000 10	Send tape image 'S' records to PORT 2
*DM2 50 80	Display memory sending output to PORT 2

3.8 USER I/O THROUGH TRAP 15

Format in user program:

TRAP 15	Call to MACSbug trap handler
DC. W #function	Valid functions listed below.
	Program resumes with next instruction.

Function #	Destination	Function	Buffer
0		Coded Breakpoint	
1	PORT1 console	Input line	A5=A6 is start of buffer.
2	PORT1 console	Output line	A5 to A6-1 is buffer.
3	PORT2 host	Read line	A5=A6 is start of buffer.
4	PORT2 host	Print line	A5 to A6-1 is buffer.

EXAMPLE PROGRAM:

```
                          *
                          *     TEST OF TRAP 15 USER I/O
                          *
        00002000                ORG $2000             PROGRAM STARTS HERE
002000  2E7C00004000      START MOVE.L #$4000,A7      INITIALIZE STACK
002006  2A7C0000201C            MOVE.L #BUFFER,A5     FIX UP A5 & A6 FOR I/O
00200C  2C4D                    MOVE.L A5,A6
                          *
00200E  4E4F                    TRAP 15               INPUT BUFFER FROM CONSOLE
00210   0001                    DC.W #1
                          *
002012  4E4F                    TRAP 15               PRINT BUFFER TO CONSOLE
002014  0002                    DC.W #2
                          *
002016  4E4F                    TRAP 15               STOP HERE LIKE BREAKPOINT
002018  0000                    DC.W #0
00201A  60E4                    BRA START             DO IT AGAIN
                          *
00201C  0200              BUFFER DS.L 128             THIS IS THE I/O BUFFER
                          *
                          *     EXAMPLE OF HOW TO PUT SYMBOLS IN SYMBOL TABLE
                          *     (SEE RAM EQUATE FILE FOR EXACT VALUE OF STRSYM)
                          *
00221C  53                SYMB  DC.L 'START        ',START
002228  42                      DC.L 'BUFFER       ',BUFFER
        00002234          SYMBE EQU *
        00000570                ORG STRSYM             MACSBUG'S POINTERS TO
000570  0000221C                DC.L SYMB,SYMBE        START/END OF TABLE
                                END
```

3.9 GENERAL INFORMATION

The trace display print routine has a CRT screen control feature. There are two four byte parameters, SCREEN1 and SCREEN2, that are listed in the RAM equate file. These parameters are normally null but the user may set them to appropriate values for his particular brand of CRT. The four bytes of SCREEN1 are printed before the trace display and the four bytes of SCREEN2 are printed after the display. Motorola EXORterms use a $CO to 'home' the cursor. If this is put in SCREEN1, it will give the effect of a stationary trace display.

TRAP ERROR is the general message given when an unexpected trap occurs. Nearly all of the low vectors including the user traps, interrupts, divide by zero, etc. are initialized during the reset to point to this simple error routine. No attempt is made to decipher which trap happened, but the user's registers are saved. The system usually retrieves the right program counter from the supervisor stack but some exception traps push additional information on to the stack and the system will get the program counter from the wrong place. It is recommended that the user's program reinitialize all unused vectors to his own error handler.

The REad command may have problems in some configurations. No attempt is made to control the equipment sending the information. When the system recognizes the end of a line it must process the buffer fast enough to be able to capture the first character of the next line. Normally the system can download from an EXORciser at 9600 BAUD. If the system is having problems, it might be worthwhile to experiment with lower BAUD rates.

The REad and PUnch used with cassette systems may also have speed problems. Typically the cassette can record faster than the console can print. The user may have to switch null padding profiles with the TErminal command when recording or reading a tape.

When sending data to the printer with the DM2 or PU2 type commands, additional nulls may be required after each carriage return. The maximum number of nulls is 255 with the CR2 $FF command. With high BAUD rates and slow printers, even this may not be enough. The BAUD rate may have to be set down in some situations. A 6800 assembly language program is provided in paragraph 3.12 for use with EXORciser host systems that want to use the printer.

The REad routine DOES NOT protect any memory locations. The routine will not protect itself from programs trying to overlay the I/O buffer. This will, of course, lead to errors during the download. Any location in memory can be loaded into, including MACSbug's RAM area. This allows the user to initialize such locations as the starting and ending address of the symbol table. An example of this is given with program listing in paragraph 3.8 on User I/O through TRAP 15. All the registers may be initialized except the program counter which takes its address from the S8 or S9 record.

The REad and PUnch commands support the normal S0, S1, and S9 record formats. Two new formats have been added to handle three byte addresses. The S2 record is the new data record, exactly the same as the S1 except for an extra address byte. The S8 is the upgraded version of the S9.

TRAP 15 is used by both the user I/O feature and breakpoints. When the program is running, the address of the breakpoint routine is normally in the TRAP 15 vector. When program execution is stopped, the I/O routine address is normally inserted into TRAP 15 vector. If I/O is not needed in the program, the user may change the vector with the SM command. If breakpoints are not needed, the program may change the vector while the program is running. It is recommended, however, that the user should use the other 15 vectors (or other programming techniques) and let MACSbug control TRAP 15.

NOTE: this is an excerpt from a MOTOROLA document, but is still applicable to our version of MACSBUG.

The LOOP feature suppresses the printing of the trace display in a given address range. This feature uses two RAM parameters, LOOPR1 and LOOPR2, whose locations are listed in the equate file (paragraph 3.11). These locations can be set with the SM command and displayed with the DM command. The trace display routine will check these locations to see if the program counter is within the range. The routine will always print the display whenever it hits a breakpoint with a count, or the program stops due to a breakpoint, or counts down to the end of a trace.

3.10 MACSbug RAM MEMORY MAP

MACSbug RAM

	MACSbug RAM	Address
*	..RESET..SSP........	0
*	..RESET..PC.........	4
	..BUS ERROR.........	8
	..ILL..ADD...........	C
	..ILL..INST..........	10
	..DIV ZERO...........	14
	CHK................	18
	TRAP V.............	1C
	PRIV INS............	20
	TRACE.............	24
	..EML 1010...........	28
	..EML 1111...........	2C
	SPURIOUS	60
	LEVEL 1	64
	LEVEL 2	68
	etc.	
	LEVEL 7	7C
	TRAP 0	80
	TRAP 1	84
	etc.	
	TRAP 15	BC
	USER INTER.	100
	etc.	
		3FF

MACSbug RAM

Address	MACSbug RAM
400	REGISTERS
	PC SR
	D0-D7
	A0-A7
	US SS
	WINDOWS
	BREAKPOINT ADDRESSES
	BREAKPOINT CONTENTS
	WORK RAM
57C	I/O BUFFER
	VVV
	ΛΛΛ
6B8	STACK
	DEFAULT SYMBOL TABLE
6BA	VVV

*NOTE: RESET SSP,PC are actually stored in ROM at addresses 20000 and 20004.

Warranty and Service

Dual Systems Control Corporation guarantees its products, under normal use and service as described in the manufacturer's product literature, free from defects in material and workmanship, for a period of one year from date of shipment. This warranty is limited to the repair or replacement of the product, or any part of the product found to be defective at the manufacturer's factory, when returned to Dual Systems Control Corporation, transportation charges prepaid by customer. This warranty does not apply to any equipment that has been repaired or altered, except by Dual Systems Control Corporation, or which has been misused or damaged by accident. In no case shall the manufacturer's liability exceed the original cost of the product.

DUAL SYSTEMS CONTROL CORPORATION

720 Channing Way • Berkeley, CA 94710

system reliability/system integrity