Customer Order Number 424610289-002
NSC Publication Number 424610289-002A
August 1986

PRELIMINARY

ICM-3216

ROM Monitor User's Guide

2900 Semiconductor Drive
P.O. Box 58090
Santa Clara, California 95052-8090

REVISION	RELEASE DATE	SUMMARY OF CHANGES
-001A	01/86	First Release. ICM-3216 ROM Monitor User's Guide NSC Publication Number 424610289-001A.
-002A	08/86	Second Release. ICM-3216 ROM Monitor User's Guide NSC Publication Number 424610289-002A.

This document describes the operation of the stand-alone ICM-3216 ROM Monitor.

The purpose of this document is to provide users of the ICM-3216 system with a basis to begin using the features of the ROM Monitor.

Reference Documents. The following documents support use of the ICM-3216 system and the Series 32000 family of microprocessors and software development tools.

National Semiconductor 1985 Series 32000 Databook

The information contained in this manual is for reference only and is subject to change without notice.

ICM-3216 Related Documentation

MANUAL	PUBLICATION NUMBER
UNIX System V Release 2.0, National Semiconductor Corporation	
Programming Guide	420010361-610
Support Tools Guide	420010361-620
User Guide	420010361-630
Administrator Reference Manual	420010361-710
Product Overview	420010361-720
Programmer Reference Manual	420010361-730
User Reference Manual	420010361-750
ICM-3216	
Error Message Reference Manual	420610287-530
ICM-3216 Replacement Pages	420610287-700
Administrator Guide	424610287-001
ROM Monitor User's Guide	424610289-002
ICM-3216 System Hardware Reference Manual	420610289-001
Series 32000	
Instruction Set Reference Manual	420010099-001
The C Programming Language by Kernighan and Ritchie	419308225-001

CONTENTS

Chapter 1 INTRODUCTION 1-1

1.1 INTRODUCTION 1-1
1.2 PRODUCT OVERVIEW 1-1
1.2.1 The ICM-3216 System Hardware 1-1
1.3 MANUAL ORGANIZATION 1-2
1.4 DOCUMENTATION CONVENTIONS 1-3
1.4.1 General Conventions 1-3
1.4.2 Conventions in Syntax Descriptions 1-3
1.4.3 Example Conventions 1-4

Chapter 2 GETTING STARTED 2-1

2.1 INTRODUCTION 2-1
2.2 CONNECTING THE ICM-3216 TO A CONSOLE TERMINAL 2-1
2.3 CONNECTING DISK AND TAPE DRIVES 2-2

Chapter 3 ROM MONITOR COMMANDS 3-1

3.1 INTRODUCTION 3-1
3.2 GENERAL OPERATION 3-1
3.3 MONITOR MEMORY ORGANIZATION 3-2
3.4 MEMORY AND REGISTER MANIPULATION COMMANDS 3-3
3.4.1 Examine Physical Memory 3-3
3.4.2 Modify 3-5
3.4.3 Print Register 3-7
3.4.4 Change Register 3-9
3.4.5 Write Pattern 3-10
3.4.6 Set Breakpoint 3-11
3.4.7 Test Memory 3-12
3.4.8 Display Memory Size 3-13
3.5 PROGRAM EXECUTION COMMANDS 3-14
3.5.1 Step 3-14
3.5.2 Go 3-15
3.6 DISK COMMANDS 3-16
3.6.1 Format Disk 3-16
3.7 DISK PARTITIONING AND LAYOUT RULES 3-18
3.8 SMALL COMPUTER SYSTEM INTERFACE (SCSI)DEVICE SPECIFICATION 3-20
3.9 SMALL COMPUTER SYSTEM INTERFACE COMMANDS 3-21
3.9.1 Initialize SCSI 3-21
3.9.2 Load Block 3-22
3.9.3 Write Block 3-23
3.9.4 Perform SCSI I/O 3-24
3.9.5 Copy 3-25
3.9.6 Tape 3-26
3.9.7 Execute 3-27
3.9.8 Execute With Load Parameters 3-28

3.9.9 Boot System V/Series32000 3-29
3.9.10 SCSI Error Messages 3-30
3.10 SPECIAL SYSTEM COMMANDS 3-32
3.10.1 Download Data 3-32
3.10.2 Toggle Memory Management 3-32

Chapter 4 EXECUTE PROGRAMS 4-1

4.1 EXECUTE PROGRAMS 4-1
4.2 SUMMARY OF OPERATION 4-1
4.3 PROGRAM CONTENT 4-3
4.4 INFORMATION GIVEN TO THE STANDALONE PROGRAM 4-4
4.5 SAMPLE STANDALONE PROGRAM 4-5
4.6 CHECKSUMMING 4-6
4.7 POSITIONING ON THE SOURCE DEVICE 4-6
4.8 FILE FORMAT ON THE SOURCE DEVICE 4-7
4.9 LOADING PROGRAMS INTO MEMORY WITH A STANDALONE PROGRAM 4-7
4.10 SECONDARY STACK 4-8
4.11 MEMORY MAP 4-8

Chapter 5 DOWNLOADING AND DEBUGGING FEATURES 5-1

5.1 DOWNLOADING PROGRAMS 5-1
5.2 HARDWARE CONNECTION 5-2
5.3 MIRROR DEBUGGING 5-3

Chapter 6 SYSTEM INITIALIZATION AND POWER-ON CONFIDENCE CHECKS 6-1

6.1 INTRODUCTION 6-1
6.2 LED DISPLAYS 6-1
6.3 COLD STARTS 6-3
6.4 WARM RESTARTS 6-4

Appendix A ICM-3216 COMMANDS QUICK REFERENCE A-1

A.1 MEMORY & REGISTER COMMANDS A-1
A.2 I/O & SCSI COMMANDS A-2
A.3 SPECIAL SYSTEM COMMANDS A-3

Figure 1-1. The ICM-3216 System Hardware Configuration 1-2
Figure 5-1. Host tty to ICM-3216 Connection 5-3
Figure 6-1. Power-up Status Register 6-2
Figure 6-2. LED Display - Power-up 6-3

TABLE 3-1. DISK PARTITIONIONING RULES 3-18
TABLE 3-2 PARTITIONING EXAMPLE 3-19
TABLE 5-1. XMODEM PROTOCOL 5-1

1. Chapter 1 - INTRODUCTION

1.1 INTRODUCTION

This manual serves as a user's guide to the stand-alone ICM-3216 ROM Monitor software. It includes a brief description of the ICM-3216, a discussion of the functions of the monitor, a complete description of commands, and a section on the special debugging features available with the ROM monitor. A command quick-reference guide is added to assist the experienced user.

1.2 PRODUCT OVERVIEW

The ICM-3216 ROM Monitor is a stand-alone debugger with features available to execute programs written for the Series 32000(r) processor. It is designed as a development tool to aid hardware and software designers test and debug the ICM-3216. The monitor may be used to conduct cyclic tests of the system memory, print or change the contents of system registers or memory locations, and fill contiguous locations of memory with specific data. Other commands allow the user to dump portions of system memory, download program instructions or data from a host or target system, execute the instructions of a program in a single step manner, or simply run a program that was previously downloaded. Further, there are commands that allow the user to access Small Computer System Interface (SCSI) hardware.

1.2.1 The ICM-3216 System Hardware The ICM-3216 system hardware organization is illustrated in Figure 1-1. The system can be configured from 1 Mbyte to a maximum of 8 Mbytes.

For architectural descriptions of the various National Semiconductor parts, refer to the 1985 Series 32000 Databook. For a more detailed description of the ICM-3216 system hardware, refer to the ICM-3216 Hardware Reference Manual, Customer Order Number 420610289-001.

1.3 MANUAL ORGANIZATION

Chapter 2 provides the information needed in order to configure the ICM-3216 system with a console and a disk or tape drive. The disk and tape are not necessary to the operation of the monitor, however, most systems will include them.

Chapter 3 lists all available monitor commands, explaining in detail what each command accomplishes, along with examples.

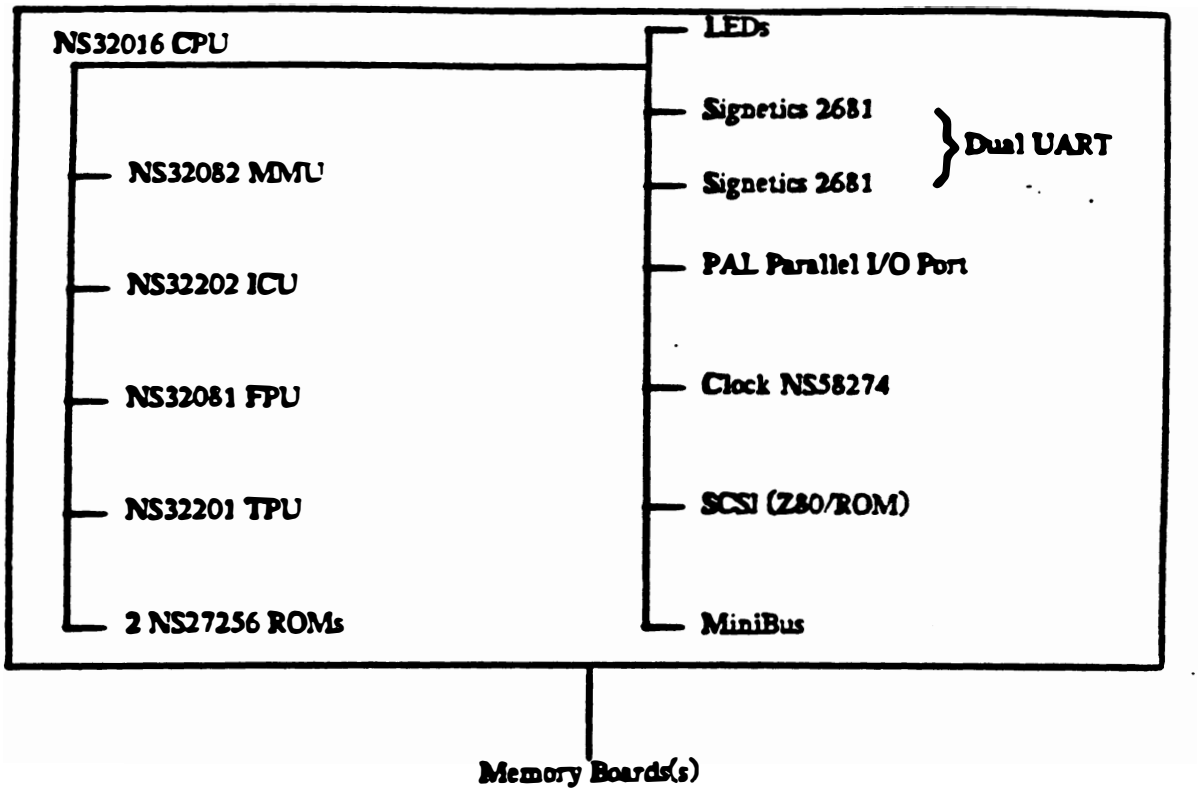


Figure 1-1. The ICM-3216 System Hardware Configuration

Chapter 4 describes how to construct standalone programs. Special features available in the monitor are also explained.

Chapter 5 describes the capability of downloading, remote debugging and mirrored debugging.

The power-on confidence checks and initialization sequence performed by the system is described in Chapter 6.

Appendix A presents a quick-reference guide for the commands.

1.4 DOCUMENTATION CONVENTIONS

The following documentation conventions are used in text, syntax descriptions, and examples in describing commands and parameters.

1.4.1 General Conventions Nonprinting characters are indicated by enclosing a name for the character in angle brackets <>. For example, <CR> indicates the RETURN key, <ctrl/[> indicates the character input by simultaneously pressing the control key and the [key.

1.4.2 Conventions in Syntax Descriptions Except where otherwise indicated, any combination of upper- and lower-case letters may actually be used when entering commands.

Italics are used for items supplied by the user. The italicized word is a generic term for the actual operand that the user enters.

Spaces or blanks, when present, are significant; they must be entered as shown. Multiple blanks or horizontal tabs may be used in place of a single blank.

{} Large braces enclose two or more items of which one, and only one, must be used. The items are separated from each other by a logical OR sign ''|.''

[] Large brackets enclose optional item(s).

| Logical OR sign separates items of which one, and only one, may be used.

... Three consecutive periods indicate optional repetition of the preceding item(s). If a group of items can be repeated, the group is enclosed in large parentheses ''().''

,,, Three consecutive commas indicate optional repetition of the preceding item. Items must be separated by commas. If a group of items can be repeated, the group is enclosed in large parentheses ''().''

() Large parentheses enclose items which need to be grouped together for optional repetition. If three consecutive commas or periods follow an item, only that item may be repeated. The parentheses indicate that

the group may be repeated.

␣ Indicates a space. ␣ is only used to indicate a specific number of required spaces.

All other characters or symbols appearing in the syntax must be entered as shown. Brackets, parentheses, or braces which must be entered, are smaller than the symbols used to describe the syntax. (Compare user-entered [], with []which show optional items.)

1.4.3 Example Conventions In interactive examples where both user input and system responses are shown, the machine output is in regular type. User-entered input is in boldface type. Output from the machine which may vary (e.g., the date) is indicated with italic type.

2. Chapter 2 - GETTING STARTED

2.1 INTRODUCTION

This chapter presents the information needed to begin using the stand-alone ICM-3216 ROM Monitor. It includes instructions for connecting the ICM-3216 system to a console terminal and for connecting and formatting a disk. For detailed information regarding pin assignments and other hardware specifications, refer to the ICM-3216 Hardware Reference Manual.

2.2 CONNECTING THE ICM-3216 SYSTEM TO A CONSOLE TERMINAL

This section describes the procedures necessary to connect a DTE device to the ICM-3216 system. The ICM-3216 ROM Monitor assumes that the console is connected to port P2.

The ICM-3216 requires a set of RS232 hoods and cables for connecting a terminal as console. The console connection (P2) responds to I-may-send and asserts with U-may-send. Soft XON/XOFF flow control is supported.

1. Connect the telephone cable to port P2.
2. Connect the other end of the telephone cable to the appropriate hood.
3. Plug the hood into the terminal.
4. Power up the terminal.
5. Power up the ICM-3216 system.

The various powerup tests and initialization will run transparently (see Chapter 6). The length of time required is dependent upon the size of the memory. About 20 secs will be required for 8 Mbyte memories. When they are done, a bell will ring. After a pause the autoboot procedure will begin. The following message will appear:

"loading ..."

If you wish to remain in the monitor, depress any key within 2 seconds after the bell and the autoboot procedure will be aborted. Two lines similar to the following should display on the terminal:

```
ICM-3216 ROM Monitor Vn.nn
%
```

The last item (n.nn) on the first line will vary according to the version of software installed on the ICM-3216 system. If nothing displays check that the RS232 connections are solidly in place, and that both ICM-3216 and the terminal have power. The percent sign (%) is the monitor prompt indicating that the ICM-3216 ROM Monitor is now waiting for commands.

Ports P3 through P5 are available for connecting additional DCE or DTE devices. Port P2 is reserved for use as a console port.

2.3 CONNECTING DISK AND TAPE DRIVES

Disk and tape drives connect to the ICM system through the SCSI interface port (connector J5). Up to seven other SCSI devices can be connected to the bus simultaneously. Disk and tape drives can be connected directly to the SCSI bus if they have a SCSI interface, or they can be connected to a disk or tape controller which in turn is connected to the SCSI bus. Each device (controller or peripheral) connected to the SCSI bus has an address on the bus between 0 and 7. Many SCSI disk controller boards have the capability of controlling 2 or 4 disk drives, although the controller board only uses one address on the SCSI bus.

A 50-pin cable should be connected to the ICM on one end and to each of the other SCSI devices. The last device at the other end of the SCSI bus should be terminated. You should consult the manual for the last device on the bus for termination details, but termination is commonly accomplished by placing resistor packs into sockets on the device.

Each device on the SCSI bus must have a unique SCSI address. As shipped, the ICM board is configured to be SCSI address 0. It is recommended that the first disk drive or controller be assigned SCSI address 1. Additional disk drives should be assigned SCSI addresses 2, 3, etc. The first tape drive or controller should be assigned SCSI address 7. Additional tape drives should be assigned SCSI addresses 6, 5, etc.

Controller boards connected to the SCSI bus and to disk and tape drives commonly have switches or jumpers on the board to configure certain characteristics about the drive(s) that are attached. Refer to the documentation for the controllers to make sure you are configured properly before

beginning to use these devices with the ICM.

Before using a disk drive, it must be formatted. This can be done with the monitor 'F' command (see Chapter 3 for details). Even if the drive has been formatted at the factory, it will probably need to be reformatted.

3. Chapter 3 - ROM MONITOR COMMANDS

3.1 INTRODUCTION

This chapter describes the general operation of the stand-alone ICM-3216 ROM Monitor. The commands are explained in detail.

3.2 GENERAL OPERATION

The monitor operates in either of two modes: user mode (default), or system mode. To toggle between the two modes, use the toggle mode command:

a <switch> where switch is either 0 (user) or 1 (system).

For example, to toggle from user mode to system mode type:

%a 1<CR>

To toggle from system mode to user mode type:

%a 0<Line Feed>

When in user mode, the monitor provides a prompt (%) and readable output where appropriate. System mode results in less readable output terminated by a <CR>, and a nonprintable prompt (<ctrl/P>). In user mode, the monitor reads each command input at a terminal until a carriage return (<CR>) is encountered. A <CR> prompts the monitor to execute the command entered. The backspace key (<BS>) may be used to delete previously typed input.

In system mode, the command input may come from a terminal keyboard, or from a program generating command strings. System mode is intended for use by programs sending commands to the monitor without user intervention.

At initialization or reset, the monitor is in user mode. Typing <DEL> at any time during the operation of the monitor causes a software reset. The current command is aborted and software reinitialization occurs.

Numeric quantities must be entered in hexadecimal formats, except where specifically noted in the command descriptions. Leading blanks are not allowed on the command line. Most commands must be entered in lower case. The exception occurs with commands related to input/output. The appropriate format for specific commands is shown in the command description.

The monitor's output responses may be suspended by entering a <ctrl/S>. They may be resumed by typing any character.

3.3 MONITOR MEMORY ORGANIZATION

Specific high and low memory addresses must not be changed by the user, these addresses must be available for the monitor. In particular, the monitor uses the low 32 Kbytes of the first Mbyte of memory for stack and data space. The stack starts one word (16 bits) below 32 Kbyte; the data starts at 0x00.

Locations 0x00 through 0x88 are reserved for interrupt vectors.

The system protects low memory by effecting a warm restart (refer Chapter 6) if a user program modifies the ROM's memory space. The top of physical memory is used to pass information to and from programs executed using the X command.

3.4 MEMORY AND REGISTER MANIPULATION COMMANDS

The following sections provide the syntax, description and examples for memory and register manipulations.

3.4.1 Examine Physical Memory The examine physical memory command allows the user to view the contents of memory locations. In user mode, output is given in both hexadecimal and ASCII format. Only the hexadecimal values are printed in system mode. The byte ordering of the hexadecimal display is dependent on the form of the command as shown in the following command syntaxes. The ASCII display is always in byte order. The three forms of this command are:

e <addr1> [addr2]	Examine bytes starting at addr1, displayed in byte order.
ew <addr1> [addr2]	Examine words starting at addr1, displayed in word order.
el <addr1> [addr2]	Examine long words starting at addr1, displayed in word order. Use <CR> for user mode, and <Line Feed> for system mode.

addr1 is the starting address, and addr2 is either 1.) the ending address, if addr2 is greater than or equal to addr1; or 2.) a count, if addr2 is less than addr1. In the latter case, addr2 bytes, words, or long words will be displayed. This usage for addr2 is a general convention.

If addr2 is not specified, a 16 (0x10) byte count is assumed.

The first command examines 16 bytes beginning at address 0x00. The dots printed at the end of the output line represent nonprintable ASCII characters. The second command examines 21 words of memory starting with address 0x24. The third example illustrates that odd addresses are rounded down to the nearest addressable byte, word or long word, as appropriate. Notice that the output is aligned according to the value of the beginning address.

```
%e 0<CR>

00000000:   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
%ew 24 15<CR>
00000024:               0000  0000  0000  0000  0000  0000
00000030:   0000  0000  0000  0000  0000  0000  0000  0000
00000040:   0000  0000  0000  0000  0000  0000  0000    .........
%el 5 7<CR>
00000004:               0000  0000    ....
%
```

The same examples are displayed for system mode.

```
e 0<LF>
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

ew 24 15<LF>
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000

el 5 7<LF>
00000000
```

For further examples of the examine command, see the modify command (refer to Section 3.4.2).

3.4.2 Modify The modify command changes the contents of specific memory locations. There are three forms for the command to accommodate byte, word and long word nomenclature.

m <addr1> [b1 ... bn] Modify memory starting at address addr1 with data bytes b1, b2, etc.

mw <addr1> [w1 ... wn] Modify memory starting at address addr1 with data words w1, w2, etc.

ml <addr1> [lw1 ... lwn] Modify memory starting at address addr1 with long data words lw1, lw2, etc.

The data arguments are separated by blanks. If the optional data arguments are not present, modify behaves interactively, prompting for data by printing the current contents of the location in question. The user can either enter new data followed by a carriage return, skip that location by pressing <CR>, or quit by entering **q** <CR>. These examples illustrate the modify command in user mode only. The system mode is not exemplified because the interactive system mode modify behaves identically to the user mode modify, and the system mode non-interactive modify produces no output. The following command line causes locations 0 and 1 to contain the values 0x21 and 0x22, respectively. The results are verified by using the examine command (see Section 3.4.1).

```
%m 0 21 22<CR>
%e 0 1<CR>
00000000:  21 22                                                !"
%
```

The next set of commands further illustrate the examine command (Section 3.4.1).

```
%ew 0 0<CR>
00000000:  2221                                                 !"
%el 0 0<CR>
00000000:  0000 2221                                            !"..
```

Notice that the ASCII display always prints in byte order, regardless of the order of the hex display. The interactive modify is exemplified below:

```
%ml 0<CR>
00000000:  0000 2221 -> <CR>
00000004:  0000 0000 -> ffff024c<CR>
00000008:  0000 0000 -> q <CR>
%
```

This session results in the contents of the long word at address 4 to be modified. The values of addresses 00 and 08 remain unchanged.

The next set of examples illustrates the differences in results from the byte, word and long word forms of the examine and the modify commands.

```
%m 1000 12 34 56 78<CR>
%e 1000<CR>
00001000:  12 34 56 78 00 00 00 00 00 00 00 00 00 00 00 00  .4Vx.
%ew 1000<CR>
00001000:  3412  7856  0000  0000  0000  0000  0000  0000   .4Vx.
%el 1000<CR>
00001000:  7856  3412  0000  0000  0000  0000  0000  0000   .4Vx.

%mw 1000 1234 5678<CR>
%e 1000<CR>
00001000:  34 12 78 56 00 00 00 00 00 00 00 00 00 00 00 00  4.x
%ew 1000<CR>
00001000:  1234  5678  0000  0000  0000  0000  0000  0000   4.xV.
%el 1000<CR>
00001000:  5678  1234  0000  0000  0000  0000  0000  0000   4.xV.

%ml 1000 12345678<CR>
%e 1000<CR>
00001000:  78 56 34 12 00 00 00 00 00 00 00 00 00 00 00 00  xV4..
%ew 1000<CR>
00001000:  5678  1234  0000  0000  0000  0000  0000  0000   xV4..
%el 1000<CR>
00001000:  1234  5678  0000  0000  0000  0000  0000  0000   xV4..
%
```

3.4.3 Print Register The print register commands display the contents of the various sets of registers. The values printed reflect the value saved on the last breakpoint or step interrupt.

r	Print values of all registers.
rg	Print values of the general registers.
rd	Print values of the dedicated registers.
rm	Print values of the memory registers.
rf	Print values of the floating point registers.

```
%r<CR>

r0        r1        r2        r3        r4        r5        r6
00000000  00000000  00000000  00000000  00000000  00000000  0000
pc        sb        fp        sp1       sp0       intbase   mod
00001000  00000000  000e0000  000d0000  000e0000  00000000  0000
f0        f1        f2        f3        f4        f5        f6
00000000  00000000  00000000  00000000  00000000  00000000  0000
msr       ptb0      ptb1      eia       bpr0      bpr1      bcnt
00000000  00000000  00000000  00000000  00000000  00000000  0000
pf1       sc
00000000  00000000

%rg<CR>
r0        r1        r2        r3        r4        r5        r6
00000000  00000000  00000000  00000000  00000000  00000000  0000
%rd<CR>
pc        sb        fp        sp1       sp0       intbase   modp
00001000  00000000  000e0000  000d0000  000e0000  00000000  0000
%rf<CR>
f0        f1        f2        f3        f4        f5        f6
00000000  00000000  00000000  00000000  00000000  00000000  0000
%rm<CR>
msr       ptb0      ptb1      eia       bpr0      bpr1      bcnt
00000000  00000000  00000000  00000000  00000000  00000000  0000
pf1       sc
00000000  00000000
%
```

In system mode, the session would look like this:

```
r<LF>
00000000   00000000   00000000   00000000   00000000   00000000   0000
00001000   00000000   000e0000   000d0000   000e0000   00000000   0000
00000000   00000000   00000000   00000000   00000000   00000000   0000
00000000   00000000   00000000   00000000   00000000   00000000   0000
00000000   00000000

rg<LF>
00000000   00000000   00000000   00000000   00000000   00000000   0000

rd<LF>
00001000   00000000   000e0000   000d0000   000e0000   00000000   0000

rf<LF>
00000000   00000000   00000000   00000000   00000000   00000000   0000

rm<LF>
00000000   00000000   00000000   00000000   00000000   00000000   0000
00000000   00000000
```

3.4.4 Change Register The change register command changes the current register value to the value specified. In user mode, the monitor confirms the change by displaying the new value.

c <rname> <data>	rname is a valid register name and data is an appropriate hex value. Data is in long word format for all registers except **psr** and **mod**, which are 2 bytes each. Register names are lower case.

The following session changes the contents of registers **r4** and **psr**.

```
%c r4 1a2f223c<CR>
r4: 1a2f223c
%c psr ffff<CR>
psr: ffff
%
```

In system mode:

```
c r4 1a2f223c<LF>

c psr ffff<LF>
```

3.4.5 Write Pattern The write pattern command is used to fill a sequence of bytes with a fixed value.

p <addr1> <addr2> <pattern>	A one byte pattern is written into memory starting at byte address addr1. If addr2 is greater than or equal to addr1, then addr2 is the last location to be written. Otherwise, it is taken as a count indicating the number of bytes to be written.

The following command lines will clear locations 0x1000 to 0x2000, inclusive, and then fill locations 0x2000 to 0x2fff, inclusive, with hex aa.

```
%p 1000 2000 0 <CR>
%p 2000 1000 aa<CR>
%
```

3.4.6 Set Breakpoint

b r <addr1>[<mode> [<count>]] where r is either 0 or 1, corresponding to **ptb0** or **ptb1**; addr1 is the address to be loaded into the register; mode is a hex constant loaded into the AS, VP, BR, and BW bits; and count is a count loaded into the **bcnt** register.

Set Breakpoint enables breakpoint mode in the memory management unit (MMU). With this command, the user sets breakpoints in any of the ways allowed by the MMU hardware.

If mode is not specified, the mode defaults to execution of a physical address. If count is not specified, the count register **bcnt** is disabled. Bit **bn** in the MMU is set to r, and the appropriate breakpoint register is loaded.

Breakpoints cause the MMU to pull the processor NMI line, and cause a non-maskable interrupt. This forces execution back to the monitor and, in user mode, a message indicating the event is printed. In system mode, a prompt is printed to indicate the event. The set breakpoint command assumes that low memory (0x00 - 0x88) is available to it. To set a breakpoint at address 1000, using **ptb0** and breaking on the 16th (0x10) read of the address, type:

```
%b 0 1000 10 10<CR>
%
```

3.4.7 Test Memory The test memory command writes a cyclic byte pattern through a specified range of memory and then reads back those locations, testing for that pattern.

t <addr1> <addr2>	Test memory ranging from address addr1 to address addr2. If addr2 is less than addr1, a2 is taken as a count indicating the size of the block to be tested.

This test does long word (4 byte) writes, incrementing the pattern by one for each long word. The test loops infinitely, writing, then reading. A message indicating the completion of each pass is printed to the console. If an error is detected, the message

Memory test error: address addr is errvalue should be expected

is printed on the console and the test aborts. Typing <CR> aborts the test at any time.

3.4.8 Display Memory Size

$ Display the memory size and system diagnostics.

This command reports the current top RAM location, in both decimal and hexadecimal. The messages

```
     Memory size: mdecsize (mhexsize) bytes
         ROM uses: rdecsize (rhexsize) bytes
   powerup test results -> xx xx xx xx
```

are printed, where mdecsize and mhexsize are the decimal and hexadecimal equivalents of the current top RAM location; rdecsize and rhexsize are the decimal and hexadecimal equivalents of the space used by the ICM-3216 system software. The powerup test results are the contents of the powerup status register (R7), generated during cold starts of the system. The meaning of these hexadecimal values is explained in Chapter 6.

3.5 PROGRAM EXECUTION COMMANDS

The following sections provide the syntax, definitions and examples for program execution.

3.5.1 Step The step command causes single-step program execution.

s [addr1] — The trace bit in **psr** is set, the environment is restored, and program execution begins at the address currently stored in **pc**. If address addr1 is specified, it is loaded into **pc** before the environment is restored.

One program instruction executes and then control returns to the monitor. In user mode, the system prints one of the following messages indicating that the trap was encountered:

```
NMI: pc = x          (A non-maskable interrupt occurred.)
Break: pc = x        (A breakpoint was encountered.)
Trace: pc = x        (Execution proceeded normally.)
Trap: type = t, pc = x
```

where x is a hex number indicating the contents of the program counter, and t is the trap type. Trap types are listed in the Series 32000 Databook.

In system mode, a prompt is returned.

3.5.2 Go

g [addr1] Load **pc** with addr1, and continue execution.

The go command causes execution to continue, using the current environment. If addr1 is specified, then register **pc** is loaded with addr1 and execution begins at that address. Otherwise, execution continues at the address currently specified in **pc**. Tracing is turned off.

3.6 DISK COMMANDS

The commands described in this section are used to download and manipulate data on a disk connected to the ICM-3216 system via the SCSI bus.

3.6.1 Format Disk

F

This command is used to prepare a disk for use with the ICM. The format command asks you for information about your disk, uses this information to perform a SCSI Mode Select command to your controller to inform the controller about your desired configuration of the disk, performs the actual formatting of the disk, and writes a default partition table onto the disk.

To execute this command, enter 'F' at the command line. There are no arguments. All values supplied to questions asked by this command should be decimal values unless otherwise specified.

After entering the command, you will be presented with a menu of choices depending on which type of disk controller you have. The choices are:

A) Adaptec ACB4000
B) Emulex MD01
C) Common Command Set
D) Other

You should select choice A if you have an Adaptec controller board (model 4000, 4070, or 5500), choice B if you have an Emulex controller board (model MD01), choice C if you have a controller or disk drive that uses the Common Command Set, and choice D if you are using any other disk controller.

You will then be presented with prompts for the SCSI device address. Normally, the device address for the main system disk will be SCSI channel 1, logical unit number 0. Disks other than the main system disk will naturally have different addresses.

If you have selected choice A or B (Emulex or Adaptec) in the first menu, you will then be asked a series of questions about various disk parameters. Answers to these questions can be found in the literature supplied by your disk drive manufacturer.

If you have selected choice C (Common Command Set) in the first menu, you will be asked if you wish to perform a SCSI Mode Select command. Normally, for stand alone controller boards you should answer yes ('y') and for controllers embedded into disk drives you should answer no ('n'). Answering no will use the controller's default parameters. A yes answer will produce a series of questions about various disk parameters.

If you have selected choice D (Other) in the first menu, you have the ability to format your disk, but you will have to supply various parameters about the disk by hand. You are first asked the number of Mode Select bytes followed by that number of prompts for the byte values. You must enter the data, one hex byte at a time, required by the Mode Select command in your controller. You must consult your controller's manual section for the Mode Select command to properly determine the format of this data.

After you have answered all of the questions about the configuration of your disk, a SCSI Mode Select command will be performed (except in one case for the Common Command Set as described above). If the controller accepts this data, you will be asked to verify that you wish to format the drive.

The actual formatting of the disk is then performed. Formatting a typical disk will take between 3 and 15 minutes. If an error is detected during formatting, it will be reported on the screen and the format operation will be aborted.

If the formatting completes successfully, the size of the disk will be displayed on the screen. A default partition table will be written on the first block of the disk. The disk is now ready to use with the ICM.

3.7 Disk Partitioning and Layout Rules

Default disk partitions are set by the ROM-based "F" (format) command as part of the disk formatting operation.

The monitor command "P" (Partition) can be used to change the partition tables from the monitor without reformatting the disk. Partition 4 is used for the boot loader, with a default of 32 Kbytes of space for the loader's text and data sections.

The following table shows the default size and use of partitions for System V/Series 32000.

TABLE 2. Disk Partitioning Rules
(terms defined following table)

partition	first	last+1	size
p0	P	P+R	R
p1	P+R	P+R+U	U
p2	P+R+U	P+R+U+s(p2)	T-(P+R+U+s(p5)+B)
p3	P+R	P+R+U+s(p2)	U+s(p2)
p4	T-B	T	B
p5	T-B-s(p5)	T-B	m(S,T-(P+R+U+B))
p6	0	P	P
p7	P	T	T-P

Terms used:

Disk-dependent

T = total size of disk

Independent of disk used:

R = size of root partition (16384 blocks)
U = size of user partition (38912 blocks)
B = size of boot loader partition (64 blocks)
P = size of partition table partition (2 blocks)
S = maximum swap/dump partition size (16384 blocks)

s(partition) = size of partition
m(size1, size2) = least of size1 and size2

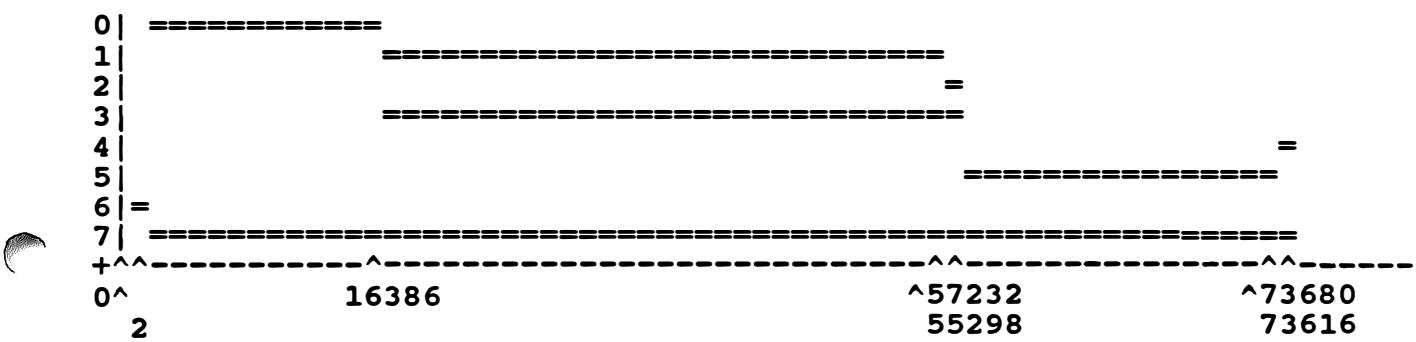
Partitioning Example

The following provides an example of these partitioning rules on a 40 Mb disk drive. Note that the block size is 512 bytes, and that the total disk space is 73680 blocks.

TABLE 4. Partitioning Example

partition	first block	last block	# of blocks	use
0	2	16385	16384 (8 Mb)	root file system
1	16386	55297	38912 (19 Mb)	usr file system
2	55298	57295	1998 (0 Mb)	undesignated
3	16386	57295	40910 (19 Mb)	undesignated
4	73616	73679	64 (0 Mb)	boot loader
5	57232	73615	16384 (8 Mb)	dump/swap area
6	0	1	2 (0 Mb)	HW parameters/ partition table
7	2	73679	73678 (35 Mb)	entire disk - partition 6

Diagram of the same thing (partition/block #):



3.8 SMALL COMPUTER SYSTEM INTERFACE (SCSI) DEVICE SPECIFICATION

The following conventions are used to designate SCSI devices within the ROM monitor:

If the device is a disk (RANDOM)

<channel>[:<logical_unit>[:<partition>]]

or

[<channel>]:[<logical_unit>]:[<partition>]

Disk Defaults

<channel> = 1
<logical_unit> = 0
<partition> = 0 (4 for 'X' and 'B' command)

<channel>[:logical_unit>]
or
[<channel>]:[<logical_unit>]

Tape Defaults:

<channel> = 7
<logical_unit> = 0

Valid ranges for the above values (tape or disk):

<channel> = [0-7]
<logical_unit> = [0-7]
<partition> = [0-7]

These designations are valid wherever a <SCSI> device address is called for.

3.9 SMALL COMPUTER SYSTEM INTERFACE (SCSI) COMMANDS

This sectiuon describes commands used to control devices attached to the SCSI bus. For further information regarding the main CPU and SCSI control hardware, refer to ICM-3216 System Hardware Reference Manual.

3.9.1 Initialize SCSI

S Initialize the SCSI hardware.

This command performs a SCSI bus reset, to reset the SCSI hardware.

3.9.2 Load Block

l <SCSI> <address> [<block count> [<block offset>]]

Load <block_count> 512-byte blocks from device <SCSI> beginning <block_offset> blocks from the beginning of the device into memory starting at address <address>.

This command allows the user to load a contiguous set of blocks from a SCSI device into physical memory.

If SCSI refers to a tape device <block_offset> is not used. The read begins at the current tape position. If not used <block_offset> defaults to 0.

The values <address> <block_count> and <block_offset> are represented as hexadecimal values. The value <block_count> defaults to 1 unless otherwise specified.

Examples: To load one block from disk device at channel 1, logical unit 0, partition 0, and block 5 into memory starting at address 3e8, enter:

```
%l 1:0:0 3e8 1 5 <cr>
```

To load a 6 block program that resides on disk device channel 1, logical unit 1, partition 4, starting at block 8 into memory starting at address 0x400, enter:

```
%l 1:1:4 400 6 8 <cr>
```

To load a 6 block program that resides on tape device channel 7, logical unit 0 into memory starting at address 0x400, enter:

```
%l 7 400 6 <cr>
```

3.9.3 Write Block

w <SCSI> <address> [<block_count> [<block_offset>]]

Write <block_count> 512 byte blocks from memory beginning at address <address> to device <SCSI> beginning <block_offset> blocks from the beginning of the device.

This command allows the user to write a contiguous set of blocks from physical memory onto a SCSI device.

If SCSI refers to a tape device <block_offset> is not used and the write begins at the current location of the tape. If not used <block_offset> defaults to 0.

The values <address>, <block_count>, and <block_offset> must be represented as hexadecimal values. The value <block_count> defaults to 1 unless otherwise specified.

Examples: To write one block to disk device at channel 1, logical unit 0, partition 0, and block 5 from memory starting at address 3e8, enter:

```
%l 1:0:0 3e8 1 5 <cr>
```

To write a 6 block program to disk device channel 1, logical unit 1, partition 4, starting at block 8, from memory address 0x400, enter:

```
%l 1:1:4 400 6 8 <cr>
```

To write a 6 block program to tape device channel 7, logical unit 0 from memory address 0x400, enter:

```
%l 7 400 6 <cr>
```

3.9.4 Perform SCSI I/O

i <SCSI> <cdb addr> <data addr> <length>

This command allows the user to perform any SCSI I/O command. A SCSI command descriptor block at address cdbaddr is sent to the device specified by <SCSI>. Data address (data addr) and length of data (length) are used in the operation.

When the command is successfully completed, the system prompt is displayed:

```
%
```

If the command procedure is not successfully completed, an error message will be displayed.

3.9.5 Copy

C <chan>[:<lun>[:<partition>]] <chan>[:<lun>[:<partition>]]

This command copies data from one SCSI device to another.

Any data checking is done by the controllers. The user should refer to the appropriate controller manual for further information.

For example, to copy the first file on a tape at SCSI address 7 to the first partition (0) on a disk at SCSI address 1:

```
%C 7:0 1:0:0<CR>
Copy xxxxx blocks
from SCSI address 7, unit 0, device type tape
to SCSI address 1, unit 0, partition 0, device type disk
Are you sure (y/n)?y<CR>
Copying ...
copy complete  (error message appears here if any)
Rewind tape (y/n)? y<CR>
rewinding...
%
```

There are several possible fatal errors that may occur prior to starting the actual copy. Any one of the following messages may print prior to aborting the command:

```
Cannot open source device
Cannot open destination device
```

If an unrecoverable error occurs during the copy, one of the following will print:

```
Copy aborted: error on source device ( )
Copy aborted: error on destination device ( )
```

Following the copy, if the device is a sequential device, the user will be asked if the device should rewind. After rewinding, the command quits.

3.9.6 Tape The tape command allows the user to perform various tape operations.

T <SCSI> r|f|s|w [<arg>]

The options have the following meanings:

r - rewind
f - forward space to file mark
s - space record
w - write file mark

Only one option may be specified per T command. Arg represents the number of records or file marks to process, and if not specified defaults to 1. Arg is ignored if the r option is given.

3.9.7 Execute

x [<arg> ...]

Load and execute a standalone program from SCSI device 1:0:4, with arguments [<arg> ...].

(CAUTION) Avoid striking the "z" key by mistake; it will require resetting the system.

The execute command is intended to load a program into memory and execute the program.

The precise syntax of this function depends on the standalone program used. For more detailed information see Ch.4.

3.9.8 Execute with load parameters

X <SCSI> [<arg> ...]

CAUTION) Avoid striking the "Z" key by mistake; it will result in requiring a system reset.

Load and execute a standalone program from device <SCSI> with arguments [<arg> ...].

This execute command is intended to load a program into memory from a user-specified device and execute that program.

The precise syntax of this function depends on the standalone program used. For more detailed information see Ch.4.

3.9.9 Boot System V/Series 32000

B Execute the System V/Series 32000 operating system.

This command effects an x unix command. The command looks for the file unix in the file system on /dev/dsk/0s0.

3.10 SCSI Error Messages

If an error occurs during the execution of a command that uses the SCSI interface, detailed status of that error is reported in parentheses following a general statement from the command.

(channel status = xx)

The two hex digit value displayed is the channel status returned by the SCSI hardware on the ICM board. Receipt of this error is an indication that the ICM and controller boards are not properly communicating on the SCSI bus. The most likely causes of this error are improper cabling on the SCSI bus, and a controller board that is not responding at all.

The ICM-3216 System Hardware Reference Manual lists the channel status values.

(SCSI status = xx)

The two hex digit value displayed is the status byte returned from the target device. The most common values returned are

```
08  =  device busy
18  =  reservation conflict
```

The monitor will not return a value of 02 (check condition). For this condition, a REQUEST SENSE command will be issued and detailed status will be reported in one of the error reports below.

The following messages display errors as reported by the SCSI REQUEST SENSE command. Different controllers may display different messages. For a list and explanation of the values displayed below, you must consult the manual for your SCSI controller.

(sense value = xx)

For devices that do not support extended sense (such as Adaptec disk controllers), this 8-bit value is the error class and error code.

(sense key = x)

For devices that support extended sense, but do not conform to SCSI standards for the extended sense value, the 4-bit sense key is displayed.

(sense key = x, sense value = xx)

For most devices that support extended sense, both the 4-bit sense key and 8-bit detailed sense status value are displayed.

Note: The Adaptec ACB3530 tape controller does not return an additional sense status, so the sense value displayed for this controller will always be zero. The error must be determined solely from the sense key.

EXAMPLES

Error in accessing tape device (sense key = 2, sense value = 09)

Sense key 2 is "not ready." Sense value 9 is "media not loaded." Message was diplayed as the result of a "T 7 R" command to an unloaded tape drive attatched to an Emulex controller.

Copy aborted: error on destination device
(sense key = 7, sense value = 0)

Sense key 7 is "data protect." An attempt was made to write onto a write protected device. This was an Adaptec tape controller, so the sense value is always zero.

3.11 SPECIAL SYSTEM COMMANDS

These commands are available to aid the user when setting the system up for System V/Series 32000.

3.11.1 Download Data

d addr1 — Download data from host to memory address addr1. This command is used in conjunction with the xm(1) command (System V/Series 32000 User Reference Manual). See Chapter 5 for details on its use.

The ICM-3216 System Hardware Reference Manual lists the channel status values.

3.11.2 Toggle Memory Management

U [0|1] — This command allows the user to turn memory management off or on, as needed. Zero as an option turns memory management off, one turns it on. If no value is given, the command reports whether memory management is turned on or off.

4. Chapter 4 LOAD AND EXECUTE PROGRAMS

4.1 EXECUTE PROGRAMS

The ROM monitor has an execute function which loads standalone programs from SCSI secondary storage devices and executes them. After it is executed, the program may return to the ROM monitor, but it is not required.

Three similar commands , "X", "x", and "B" are used to execute standalone programs. Since "X", "x", and "B" are nearly identical, the rest of the chapter will apply to all of them unless explicitly stated otherwise. For more information on the use of these monitor commands see Chapter 3.

Standalone programs are loaded from tape files or disk partitions on SCSI secondary storage devices. Constraints on the form of standalone programs and appropriate location on the device are described in sections 7 and 8.

A standalone program which is used to load another program into memory is called a secondary loader. Specialized features of the execute function can be used to set up a stack for the newly loaded program, and execute it. This simplifies the task of the secondary loader. All the loader must do is find the program on disk or tape, load it into memory, and return to the ROM monitor.

These features can also be used to implement a two stage bootstrap process. A second stage is often used to isolate file system and operating system specific tasks needed to perform the boot process. This is the mechanism used to boot System V/Series 32000 on the ICM-3216.

4.2 Summary of Operation

The execute function loads the entire contents of a tape file or disk partition into memory starting at physical address 0x8000, reads a checksum from 0x800c and byte count from 0x8008, and compares them to a checksum it calculates on the data read. If the loaded data checks out, the ROM does a "jsr" to 0x8010.

Program execution begins at physical address 0x8010 so a valid entry point must exist there. Six arguments are supplied by the ROM via the local stack. Additional command line arguments are made available to the standalone program on a variable length argument list 0x400 bytes below the top of physical memory. Up to 32 command line arguments may be

passed in this way. These are summarized in section 4, and access to them is illustrated in section 5. Since no part of memory is protected or checked, the standalone program can do just about anything. See section 3, Program Content, for more information on this. If return to the ROM is required the program should do a "ret 0" as shown in section 5. If the standalone program is not being used as a secondary loader, be sure to put a -1 in "r0" before the return. See example in section 5.

After the program has returned, the ROM checks r0. If a -1 is found in r0, the execution is complete and the program falls into the monitor loop. If a positive integer is found, the monitor assumes it to be a valid entry point of a program loaded by a standalone program (secondary loader), and prepares to execute the program.

If the standalone program is used as a secondary loader and the ROM monitor is used to execute what it loads, the following stack is set up 1K below the top of memory and used. This stack is constructed after the secondary loader is loaded into memory and before it is run. Thus, the secondary loader can use and modify the stack if necessary.

TOP is the top of physical memory found at power up.

Address Designation	Points of Interest
TOP - 0x414	ROM return address
TOP - 0x410	0x12345678
TOP - 0x40c	number of arguments
TOP - 0x408	address of argument list
TOP - 0x404	mem-size
TOP - 0x400	beginning of argument list
argument list	
TOP	

Before executing a program loaded by a secondary loader, the ROM checksums the ROM data area, saves registers, sets up registers and starts execution at the entry point specified in r0. Also, if "biased" and "biaslen" were adjusted by the secondary loader, the ROM will relocate the program in memory before running it.

If the bootstrapped program modifies the ROM data area a warm restart will be attempted when the program returns to the monitor. Otherwise, a return will continue in the ROM monitor.

4.3 Program Content

If return to the ROM is not required there are no restrictions on program content. If, on the other hand, return to the ROM is desired, a few rules need to be observed. Since the standalone program is free to write anywhere in memory, and since no checking is done to insure the integrity of the ROM data area upon return from a standalone program, take care not to let the standalone program write above or below the the bounds of available memory referred to on the stack. Also, if the program is not a secondary loader, it is important to put a -1 in r0 before returning to the ROM. If a positive integer is found in r0 the ROM assumes it is the entry point of a program bootstrapped into memory by a secondary loader and will attempt to run it. See section 5 for example of stack access and correct return procedure.

4.4 Information Given to the Standalone Program

These are the locations and meanings of values made available to the standalone program.

8(fp) beginning of available memory

This is a 32 bit pointer to the first byte of memory available to the standalone program.

12(fp)

This is a 32 bit pointer to the second argument to the "X" command. This argument is a character string terminated by a null character. If the argument is omitted, or the "x" command is used, the string is empty.

16(fp)

This is a 32 bit pointer to the second argument to the "x" command or the third argument to "X" command. This argument is a character string terminated by a null character. If the argument is omitted, the string is empty. (This is a copy of the the first argument in the variable length argument list located near the top of memory.)

20(fp) end of available memory

This is a 32 bit address pointing to the last byte available to a standalone program. This is just before the area reserved for the secondary stack.

24(fp) address of "biased"

This is the address of a 32 bit integer in the ROM data area. This value is set by the standalone program when secondary load biasing is done (see 4.9).

28(fp) address of "biaslen"

This is the address of a 32 bit integer in the ROM data area. This value is set by the standalone program when secondary load biasing is done (see 4.9).

20(fp)+0x400 beginning of the variable length argument list

4.5 Sample Standalone Program

This example successively accesses each element of the stack, then returns to the ROM.

```
_entry:
        enter   [], 0

        movd    8(fp), r0         # beginning of available memory
        movd    12(fp), r0        # argument <arg2>
        movd    16(fp), r0        # argument <arg1>
        movd    20(fp), r0        # end of available memory
        movd    0(24(fp)), r0     # "biased"
        movd    0(28(fp)), r0     # "biaslen"

        movqd   -1, r0            # return -1
        exit    []
        ret     0
```

Note: If a positive integer is moved into r0 before the exit the ROM will construct a stack 1K before the end of memory and start execution at the physical address identified by the positive integer.

4.6 Checksumming

The following is the calculation done to determine the checksum on the standalone program. The result of this calculation is compared with the checksum at address 0x800c read from the SCSI device. The standalone program will not be executed if these numbers don't match.

```
#define LOAD            0x8000  /* address of loader load point */
#define BYTECOUNT       0x8008  /* address of loader byte count */
#define CHECKSUM        0x800c  /* address of loader checksum */
#define ENTRY           0x8010  /* address of loader entry point */

char *address;  /* address used in calculation */
int sum;        /* checksum should match value at CHECKSUM */

bytecount = *((unsigned *)BYTECOUNT);   /* value of bytecount */

{
        address = (char *)(ENTRY);
        sum = 0;
        while (address < (char *)(ENTRY + bytecount)) {
                if (sum&01)
                        sum = (sum >> 1) + 0x8000;
                else
                        sum >>= 1;
                sum += *address;
                sum &= 0xffff;
                address++;
        }
}
```

4.7 Positioning on the Source Device

This section describes where and how to locate the standalone program on the SCSI source device.

The "x" and "B" versions of execute use SCSI device 1:0:4 as the source of the load. To specify some other device use the "X" command.

If the load device is a disk, position the standalone program as follows: the program must start at the beginning of a valid partition. (Use ROM monitor "P" command to list and modify disk partitioning information.).

If the load device is a tape drive, position the standalone program as follows: Locate the program at a position that can be found with the ROM monitor "T" command. Before attempting

to execute a standalone program from tape be sure to position the tape at the beginning of the file containing the program.

The program is an image of the instruction and data sections of the program as they appear in memory, preceded by size and checksum information.

4.8 File Format on the Source Device

The file should consist of the program's checksum and size in bytes (as described in section 6), followed by the instruction and data sections. A valid entry point must exist at the beginning of the instruction section.

4.9 Loading Programs into Memory with a Standalone Program

If the standalone program is used as a secondary loader, special care must be taken to ensure that the ROM data area and secondary loader itself are preserved in memory. These reside in memory from physical address 0x0 to 0x14000. There are two ways to avoid such a conflict. The simplest, but least versatile approach is to restrict standalone programs to memory above physical address 0x14000. Since this restriction is often not possible, a second alternative is to load the program into a free area of memory and move the program into position immediately before it is executed. If the ROM monitor is used to execute the bootstrapped program after it has been loaded into a temporary location by the standalone program, the values "biased" and "biaslen" referenced through the stack must be adjusted by the standalone program. The meaning of these values is as follows:

"Biased" is the beginning of the temporary region where the program was loaded by the standalone program. It must be equal to the physical address of the beginning of the bootstrapped program as read from it's header by the standalone program plus 0x14000.

"Biaslen" is the length of the bootstrapped program. This is the difference between the highest and the lowest physical address used by the program.

4.10 Secondary Stack

If the standalone program is a secondary loader and the ROM monitor is used to execute the program it loads, then the ROM monitor will use the following stack 1K below the top of memory. This stack is constructed immediately before the secondary loader is run so that the secondary loader can modify it if necessary. This stack contains the variable length argument list containing command line arguments from the "x" or "X" command.

TOP is the top of physical memory found at power up.

Address Designation	Points of Interest
TOP - 0x414	ROM return address
TOP - 0x410	0x12345678
TOP - 0x40c	number of arguments
TOP - 0x408	address of argument list
TOP - 0x404	mem_size
TOP - 0x400	beginning of argument list
	argument list
TOP	

4.11 Memory Map

This summarizes physical memory as seen by the standalone program. TOP is the top of physical memory found at power up.

Address Designation	Points of Interest
0x0	
ROM data area	
0x8000	standalone program load point
0x8008	contains the standalone program byte count
0x800c	contains the standalone program checksum
0x8010	standalone program entry point
standalone program	
0x10000	end of loaded part of standalone program
zeroed data area reserved for running standalone program	
0x14000	bottom of available memory
TOP - 0x40a	top of available memory
secondary stack	
TOP	

5. Chapter 5 - DOWNLOADING AND DEBUGGING FEATURES

5.1 DOWNLOADING PROGRAMS

The download memory command works only in conjunction with commands given on a host computer. The command (**d** a1) provides the user with a means to load data from a host computer into the ICM-3216 system memory.

d a1 — Download data from a host to memory address a1.

This command is used in conjunction with the xm(1) command on the host. The xm command is issued on the host after the download command is issued on the ICM-3216 system. A version of the xmodem protocol is used to assure that data received by the ICM-3216 system is correct. Table 5-1 illustrates this protocol.

TABLE 5-1. XMODEM PROTOCOL

RECEIVER (the ICM-3216 system)	SENDER (HOST-System V/Series 32000)
C -->	
	-- <SOH><BN>(cnt)(checksum) (128 data bytes)(2 bytes CRC)
<ACK>/<NAK> -->	
	-- Retransmit or <EOT>
<ACK> -->	

The checksum is the sum of the block number BN and the count cnt of bytes to be transferred. The two bytes of CRC are calculated according to the CCITT CRC polynomial, X^16 + X^12 + X^5 + 1.

An example of typical usage of the d and System V/Series 32000 xm commands follows. It is assumed that the configuration is such that an the ICM-3216 system is connected via a tty line to the host. For the sake of example, we will refer to that line as /dev/icm.

1. On the remote end (the ICM-3216 system), the command

   %d 2000<CR>

is given. This sets up the ICM-3216 system for receiving data from a host, starting at memory location 0x2000. When the command is issued, the ICM-3216 system puts out the xmodem initiation character (C if CRC checksums are used, or NAK if standard checksums are used.)

2. At the host end, the command

   xm -th -l /dev/icm < datafile<CR>

   is entered. The option t indicates that the program should transmit data read from standard input (here, datafile). The h option indicates that local headers including checksums will be used. The l option indicates that the comm line will be specified by the next argument, in this case /dev/icm.

3. Upon receiving the xmodem initiation character from the remote end, the host starts up by sending a packet as described in Table 5-1.

4. Upon receipt of the block, the remote machine computes the checksum. If the block is good the remote machine acknowledges with an ACK, and prints a period on the user's terminal to indicate the acknowledgement. If the block is not good, an N is printed on the user's terminal, and the block is retried. Ten retries are allowed before an error is given and the transmission is halted.

5. If an ACK is received from the remote system and there is more data to send, the host sends another packet. If there is no more data, the host sends an <EOT>. If no <ACK> is received after 30 seconds, the host assumes no response from the remote system, and aborts.

For further details regarding this protocol and other options of the xm command, see xm(1), System V User Reference Manual.

5.2 HARDWARE CONNECTION

Connect host tty to ICM-3216 tty2 (P3). See Figure 5-1.

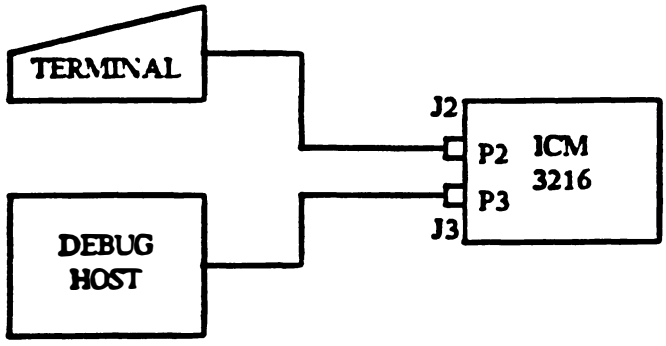


Figure 5-1 Host tty to ICM-3216 Connection

One ICM-3216 may connected to another (P3 to P3) using standard 6 wire Telco style cable.

The debug connection allows a console on P2 and a debugging host connection on P3. Control signal (IMAYSEND) must be asserted during the ICM-3216 system reset to recognize P3 as a debug connection.

5.3 MIRROR DEBUGGING

This feature is useful if two users at separate locations are collaboratively debugging a program. If a second terminal is connected to P3, the monitor will take input from either keyboard while printing output to both terminals. Both terminals run at 9600 baud. If more than 3 framing errors occur on P3 (the second terminal), the baud rate of the second terminal is dropped to 1200.

Both users may enter commands and view the output. **The users must coordinate command input.** If both users attempt to type commands at the same time, the results are unpredictable.

6. Chapter 6 - SYSTEM INITIALIZATION AND POWER-ON CONFIDENCE CHECKS

6.1 INTRODUCTION

The power-up self-test components in the ICM-3216 system boot ROMs provide a low level mechanism to verify the system hardware during the power-up sequence. The tests are primarily targeted at the board memory system, but a test of the board console UART is provided as well. The power-up code also provides a mechanism to display the results of the hardware checks using the on-board LEDs.

The system power-up sequence does not require that any RAM be functional. This allows the power-up procedure to be useful when a memory problem does not allow running the system monitor.

The firmware distinguishes between warm restarts and cold starts (reboot) by varying the set of functions performed in each case. During a cold start a set of basic system hardware evaluations are performed. The results of this evaluation are saved in a 32-bit power-up status word. Figure 6-1 illustrates the mapping of the bits in this word. Bits 16-30 of the word are unused. Bit 31 is used to indicate that a cold start has occurred. It is reset to zero on a warm restart.

After the various tests are run, several system parameters are initialized.

1. The console channel baud rate is set to 9600 baud.

2. The character structure is set to 1 stop bit and 8 bits, no parity.

3. The MiniBus gate array is initialized by unlocking the interrupt circuitry so that a power fail or Bus Conflict NMI can be recognized.

4. The Memory Management Unit, Floating Point Unit, Interrupt Control Unit, MiniBus Interface Controller registers, Centronics Parallel Port and the SCSI hardware are initialized.

6.2 LED DISPLAYS

Four of the five LEDs on the ICM-3216 system are used during the power-up sequence to indicate the results of the evaluation of several aspects of hardware performance. Figure 6-2 illustrates the meaning of the various LEDs. The LEDs are viewed from the RS232 edge of the board at the bottom.

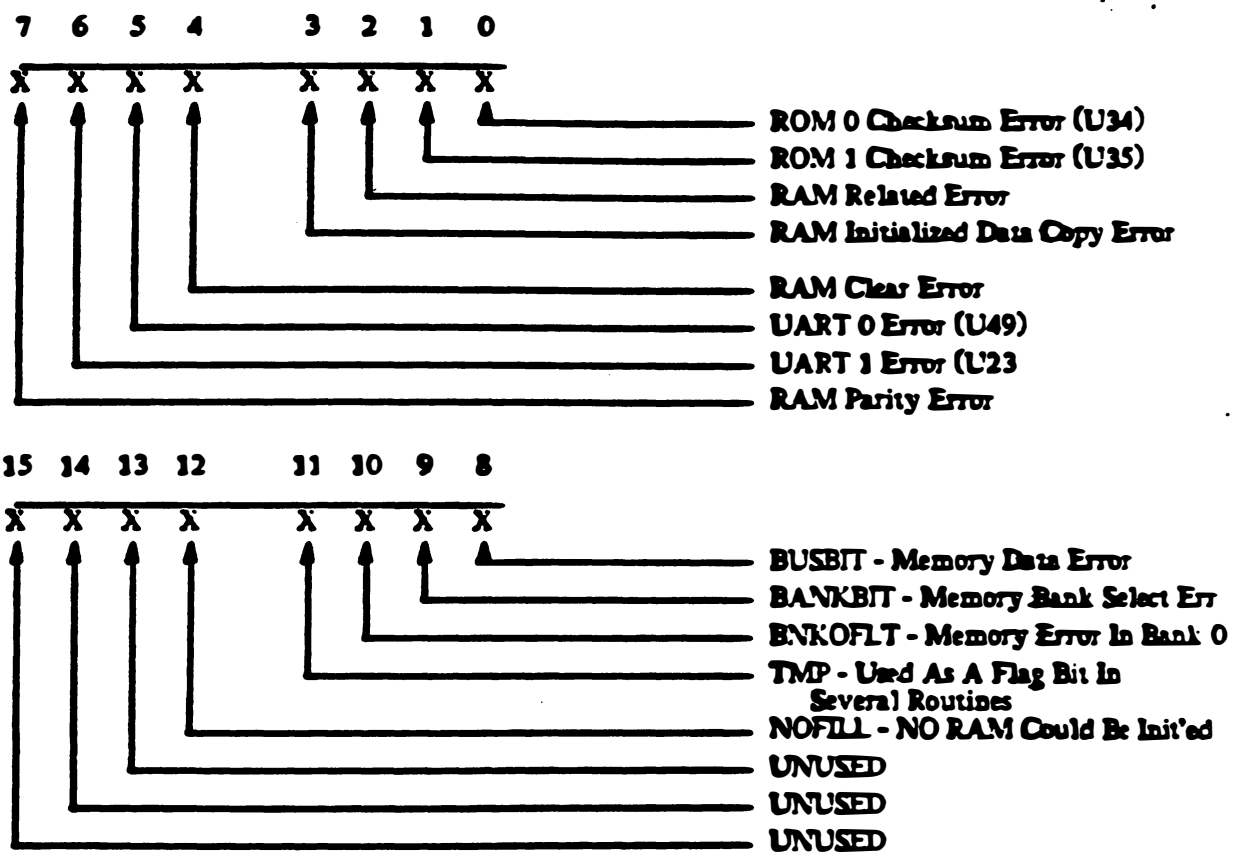


Figure 6-1. Power-up Status Register

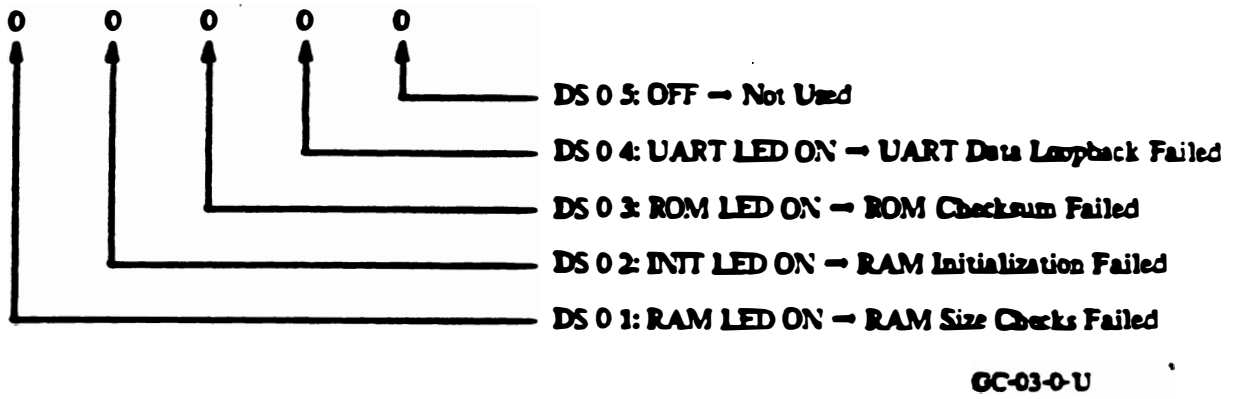


Figure 6-2. LED Display - Power-up

6.3 COLD STARTS

Various power-on confidence checks are performed to insure that certain basic functions of the ICM-3216 system are working correctly. Memory and the parity circuits are tested as described below.

1. Checksums. Checksums are computed for each of the two ROMs on the board. If the tests succeed, the ROM LED is extinguished. If either of the tests fail, an inspection of bits 0-1 of the power-up status word should determine which ROM failed to check out. Refer to Figure 6-1 for details.

2. Memory Size Checks. Three separate memory tests are run. If a data failure occurs below the 8 Mbyte maximum memory limit, the BUSBIT (bit 8) is set in the power-up status word. BANKBIT (bit 9) is set in the power-up status if memory size is limited by bank selection. If low memory (bank 0) gives an error, bit 10 BNK0FLT is set. The PARERR (bit 7) is set if a parity error is enountered.

   If the actual memory available is determined to be 1 Mbyte, 2 Mbyte, 4 Mbyte or 8 Mbyte, the RAM LED is extinguished. If not, an inspection of bits 8, 9, and 10 along with bits 2, 3, 4 and 7 (refer to Figure 6-1) should yield the reason for a lesser amount of available memory.

3. Data Loopback. A local data loopback check of the two dual-channel UARTs is tried. Each UART is considered individually. UBIT0 (bit 5) in the power-up status word indicates a failure in UART 0 (U49), UBIT1 (bit 6) indicates a problem with UART1 (U23). If neither of these bits is set after completing the UART checks, the UART LED is extinguished.

4. Memory initialization. Memory is initialized to 00, and then verified. A failure causes the RAM clear error (bit 4) in the power-up status word to be set. If no memory could be initialized, the NOFILL (bit 12) is set. Finally, ROM is copied to RAM, and the copied data is verified. If no errors are found, the INIT LED is extinguished.

Following the cold start checks, the user may opt to display the results of the power-up procedure, or to immediately enter the ROM monitor. While in the monitor, the $ command can be used to display the results of the power-up procedure.

6.4 WARM RESTARTS

A warm restart occurs as a result of a system break, the monitor catching a system trap, or a user program modifying the ROM's memory space (0-32K). The memory size checks, clearing memory, and initialization are performed. The LEDs are not modified during a warm restart, nor is the power-up status word modified. The $ command will not display the contents of the power-up status word after a warm restart.

The ROM monitor is entered immediately following a warm restart.

1. ICM-3216 COMMANDS QUICK REFERENCE

1.1 MEMORY & REGISTER COMMANDS

a 0|1	Toggle between user mode (1) and system mode (0).
e <a1> [a2]	Examine bytes starting at a1, displayed in byte order.
ew <a1> [a2]	Examine words starting at a1, displayed in word order.
el <a1> [a2]	Examine long words starting at a1, displayed in word order.
m <a1> [b1 ... bn]	Modify memory starting at address a1 with data bytes b1, b2, etc.
mw <a1> [w1 ... wn]	Modify memory starting at address a1 with data words w1, w2, etc.
ml <a>1 [lw1 ... lwn]	Modify memory starting at address a1 with long data words lw1, lw2, etc.
r	Print values of all registers.
rg	Print values of the general registers.
rd	Print values of the dedicated registers.
rm	Print values of the memory registers.
rf	Print values of the floating point registers.
c <rname> <data>	Change contents of register rname to data.
p <a1> <a2> <pattern>	A one byte pattern is written into memory starting at byte address a1. If a2 is greater than or equal to a1, then it is the last location to be written. Otherwise, it is taken as a count indicating the number of bytes to be written.
b 0|1 <a1> [<mode> [<count>]]	Set breakpoint register bptr. Address a1 is loaded into the address portion of the register; mode is a hex constant loaded into the AS, VP, BR, and BW bits; and count is a count loaded into the **bcnt** register.

t <a1> <a2> Test memory ranging from address a1 to address a2. If a2 is less than a1, a2 is taken as a count indicating the size of the block to be tested.

s [a1] Step through program instructions. The trace bit in **psr** is set, the environment is restored, and program execution begins at the address currently stored in **pc**. If address a1 is specified, it is loaded into **pc** before the environment is restored.

g [a1] Load **pc** with a1, and continue execution.

$ Print current RAM memory size and diagnostic results.

1.2 I/O & SCSI COMMANDS

F Format disk. The user is prompted for disk parameters.

P <chan>[:<lun>] Change and/or examine the partition tables without reformatting the disk. Where <chan> is the channel and <lun> is the logical unit.

l <chan>[:<lun>[:<partition>]] <addr>[<count>[<sector>]]
Load data from disk into memory.

w <chan>[:<lun>[:<partition>]] <addr>[<count>[<sector>]]
Write data from memory onto disk.

S Reset the SCSI hardware.

i <chan>[:<lun>] <cdb addr> <data addr> <length>
Initiate a SCSI command. SCSI command descriptor block at address cdb addr is sent down channel chan on logical unit lun. Data address data addr and length of data length are used in the operation.

C <chan>[:<lun>[:<partition>]] <chan>[:<lun>[<partition>]]
Copy data from device (SCSI address) to device (SCSI address).

T <chan>[:<lun>] {r|f|s|w} [<arg>]
Perform various tape operations on tape device at <SCSI>. for <arg> number of records or tape marks. Options are r(ewind), f(orward space to file mark), s(pace record), w(rite file mark).

x [<arg> ...]
Load contents of SCSI 1:0:4 into memory and execute the loaded program. CAUTION Do not strike the "z" key by mistake: it will require a system reset.

X <chan>[:<lun>[:<partition>]] [<arg> ...]
Load contents of SCSI <chan>[:<lun>[:<partition>] and execute as a standalone program. CAUTION Do not strike a "Z" key by mistake; it will require a system reset.

B Boot System V/Series32000

1.3 SPECIAL SYSTEM COMMANDS

d <addr> Download data from host to memory address <addr>.

U [0|1] Turn memory management off or on.