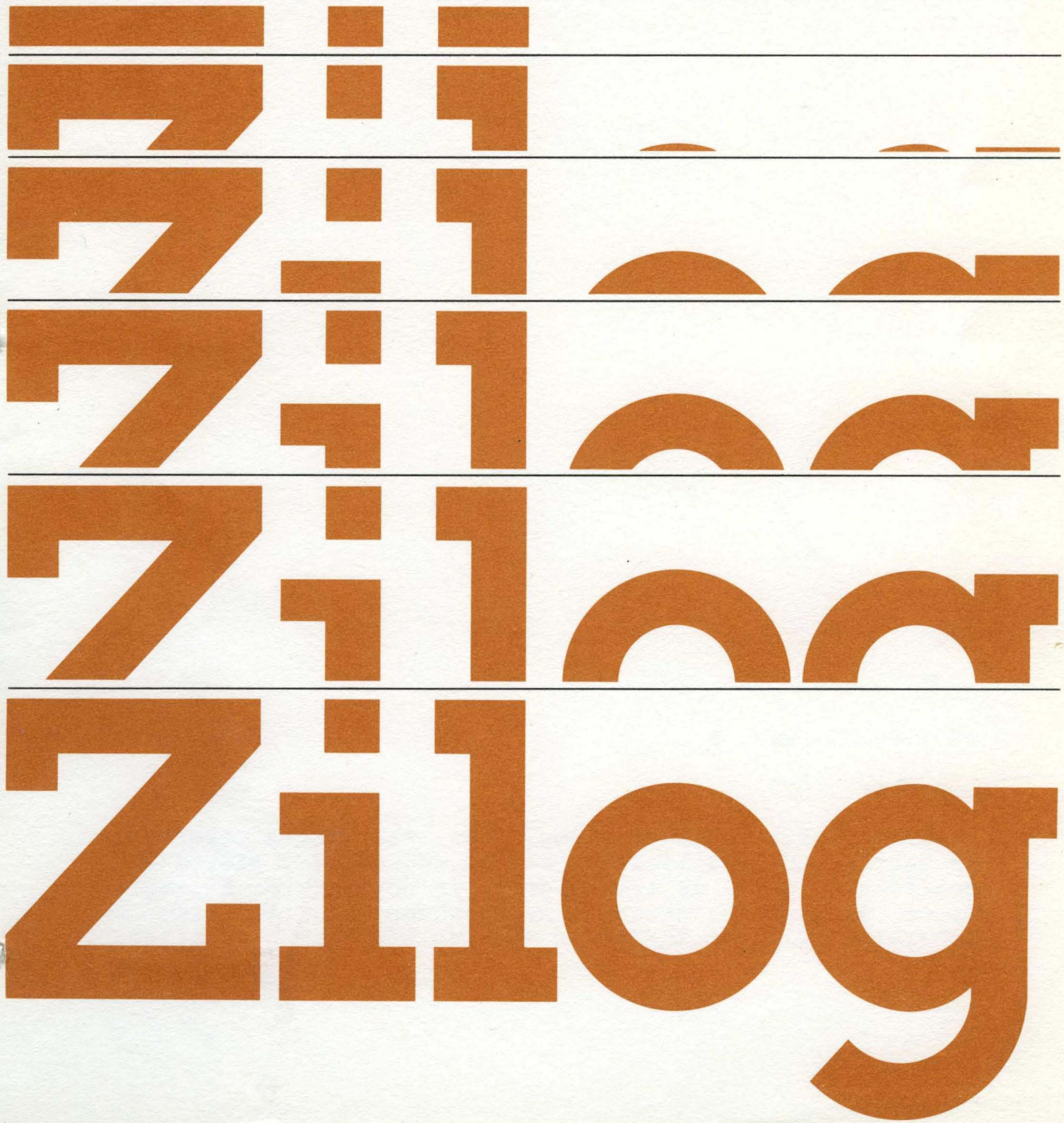
MCZ-1/20A and MCZ-1/25A MICROCOMPUTERS FLOPPY PROM

Zilog

User Guide

December 1979



03-3106-01, Rev. A

December, 1979

MCZ-1/20A and MCZ-1/25A MICROCOMPUTERS FLOPPY PROM

User Guide

December 1979

CONTENTS

SECTION 1 INTRODUCTION 1

1.1 Software Components 1
1.2 Initialization 1
1.3 Command Interpreting 1

SECTION 2 FLOPPY DISK DRIVER 3

2.1 FLOPPY Requests 3
2.2 Completion Codes 4

SECTION 3 TTY DRIVER 5

3.1 TTY Requests 5
3.2 Line-Edit Commands 6
3.3 Terminal Requirements 7

SECTION 4 DEBUG ENVIRONMENT 9

4.1 Breakpoints 9
4.2 Program Restrictions 9
4.3 Debug Commands 10
4.4 Debug Interface 15

SECTION 5 SYSTEM PARAMETERS 17

5.1 User-Accessible System Parameters 17
5.2 Port Addresses 18
5.3 PROM Monitor Routine Requests and Error Codes 19
5.4 Device Command Codes 19

INDEX .....Index 1

SECTION 1

INTRODUCTION

1.1 Software Components

The MCZ-1/20A, MCZ-1/25A Floppy PROM set provides the basic debugging commands, Input/Output, and bootstrap portions of a floppy disk operating and development system. The system consists of a bootstrap loader, a floppy disk driver, a terminal handler, and a command interpreter. The system resides in 3K bytes of PROM memory, starting at address 0000, and uses 1K bytes of RAM, including 256 bytes allocated for the system stack, starting at address 1000H.

1.2 Initialization

At system RESET, the PROM Monitor reads the switch bank to determine the baud rate required at the serial interface. All variable parameters are set in RAM to their initial values. The USART (Universal Synchronous Asynchronous Receiver Transmitter) is programmed for asynchronous operation with two transmit stop bits, no parity, 8 data bits, and divide-by-sixteen operation. The USART is further programmed to generate active (low) signals on REQUEST TO SEND and DATA TERMINAL READY lines, and both the receiver and transmitter are enabled. Finally, the PIO (Parallel I/O chip) in the disk interface is set to the bit control mode on both its ports, and appropriate directions are assigned to the bits.

1.3 Command Interpreting

After USART initialization, the monitor fetches its first command string from the terminal. If that command string is comprised of a carriage return or the characters "OS" (meaning Operating System) followed by a carriage return, control transfers to the bootstrap logic. For any other command string, control is passed to the the Debug command scanner.

The bootstrap subroutine reads in 128 bytes of a supplementary bootstrap routine from disk. If the read is successful, control is transferred to the routine. If the read is unsuccessful and the drive is attached, control returns to the Debug command scanner. If no drive is attached, the bootstrap routine reads 128 bytes of data in binary form from the terminal. These bytes of data are put into RAM, starting at address 1000H. The central processing unit (CPU) continues program execution from that address.

SECTION 2

FLOPPY DISK DRIVER

2.1 FLOPPY Requests

The Floppy driver (FLOPPY) is used to read and write from the disk. The FLOPPY routine can only access information on a diskette formatted to Zilog software standards. The entry point for the Floppy driver is at location 0BFDH.

The driver accepts a standard parameter vector as follows:

IY+0	Ignored
IY+1	Request - type of action needed
IY+2,IY+3	Data transfer address
IY+4,IY+5	Data length in bytes
IY+6,IY+7	Completion return address
IY+8,IY+9	Error return address
IY+10	Completion code
IY+11,IY+12	Disk address (sector, track, respectively)

The starting disk address for transfer has two parts: a track address from 0 to 77 decimal, and a sector address from 0 to 31 decimal. The most significant three bits of the sector address indicate which of the eight possible units is used for the transfer. The track address is considered the most significant part of the disk address, and goes in IY+12; the sector address is considered the least significant part and goes in IY+11.

If the data transfer length is not divisible by the sector size (128 bytes), the data transfer length is increased to the next integral multiple of the sector size. Data is written/read from contiguous sectors. The length should be such that the entire transfer will take place between the sector where transfer starts and the end of the track. If this limitation is not observed, a sector error results on a read operation and a permanent sector error is created on a write operation, resulting in certain sectors being permanently inaccessible.

There are two valid requests for the floppy driver: RDBIN (0AH), which reads binary data from the disk to the data transfer area, and WRTBIN (0EH), which writes binary data from the data transfer area to the disk.

If the user's request code is increased by a value of one, the driver executes a return as soon as the request has been initiated. The data transfer can then continue under interrupt

control. When the operation is complete, the routine specified by the completion return address is called. This routine should act as an interrupt routine, except that it should not execute a Return from Interrupt (RETI) instruction; a RET instruction should be used instead.

If an error occurs during processing of the user's request and the error return address bytes are non-zero, the routine at that address is called. The routine takes whatever action is appropriate and returns. If the address is zero, the return is normal, as though there were no error.

2.2 Completion Codes

Bit 7 of the completion code byte is set when the operation is complete. Bit 6 is set to indicate that an error occurred. The least significant six bits indicate which error occurred. Possible return codes are:

Code	Meaning
80H	Normal return (Bit 6 not set)
C1H	Invalid operation request
C2H	Not ready - disk is signalling a "Not Ready" condition
C3H	Disk is write protected
C4H	Sector error - the sector address in the sector header did not agree with the sector position
C5H	Track error - the track address in the sector header did not agree with the head position
C6H	CRC error - Cyclic Redundancy Checking; indicates one or more data bits in error

SECTION 3

TTY DRIVER

3.1 TTY Requests

The TTY driver, which is used to read and write from the terminal, is interrupt driven for input characters. Characters received from the terminal are stored in a 20H-byte circular buffer. This approach provides the user with a type-ahead capability. The entry point for the TTY driver is at 0BE8H. The driver accepts a standard parameter vector as follows:

IY+0	Ignored
IY+1	Request - type of action needed
IY+2, IY+3	Data transfer address
IY+4, IY+5	Data length in bytes
IY+6, IY+7	Completion return address
IY+8, IY+9	Error return address
IY+10	Completion code
IY+11, IY+12	Ignored

The four valid requests handled by the terminal driver are: RDBIN (0AH), WRTBIN (0EH), RDLIN (0CH), and WRTLIN (10H). The binary operations read or write the specified length of data between the serial interface and the block of memory beginning at the data transfer address. No special characters are honored in binary operations. No echoing on the terminal is performed for RDBIN. See the Z80 RIO Operating System User's Manual for more information on RDBIN and WRTBIN.

The RDLIN operation begins by issuing a prompt character to the terminal, then proceeds to input characters from the terminal to the data transfer area. All input characters are echoed on the terminal. Special line delete, which is initialized by the PROM Monitor to DEL (ASCII 7FH) and character delete, which is initialized by the PROM Monitor to BACKSPACE or CONTROL H (ASCII 08H) are honored by this routine. A RDLIN operation terminates either on receipt of the number of characters specified as the transfer length or on receipt of a carriage return. The carriage return is echoed, and the number of line feeds and null characters specified in the corresponding parameter storage locations is generated.

The WRTLIN operation also proceeds either for the number of characters specified as the transfer length, or until a carriage return is encountered. The carriage return is sent to the terminal. The NULLCT parameter determines how many nulls should be sent to the terminal. The LFCNT parameter determines how many linefeeds should be written to the terminal.

Both ASCII line operations return a count of the number of characters actually transferred in the transfer length field.

If the request code is incremented by one, control transfers to the completion return address after the operation. The routine at this address should act as an interrupt routine, except that it should not execute a RETI instruction; a RET should be used instead. If the request code is not incremented by one, TTY returns normally after the operation has been completed. In this case, the completion return address is ignored. In either case, the data is transferred without the use of interrupts.

If an error occurs and the error return address bytes are non-zero, the routine stored at that address is called. The routine takes the appropriate action and returns. If the address is zero, the return is normal.

Bit 7 of the completion code is set when the operation completes normally. Bit 6 is set to indicate that an error occurred. The only error condition returned is Invalid Operation (C1H). This action occurs for request codes greater (arithmetically) than those honored by TTY. Request codes lower than those honored by TTY return complete with no error, but also with no effect.

3.2 Line-Edit Commands

The two line-edit commands interpreted by the TTY driver, Delete Last Character and Delete Current Line, can be changed from the Debug environment as follows:

1. To change the Delete Last Character command:

   Set memory location 13CCH to the ASCII code for the character desired. For example, to make the "@" the character delete, set the location to 40H.

2. To change the Delete Current Line command:

   Set memory location 13CBH to the ASCII code for the character desired. For example, to make "!" the line delete, set the location to 21H.

These characters are also changed to defaults when the RIO operating system is bootstrapped, or when the RIO SET command is issued. This means that when the Debug environment is entered from the RIO operating system, the characters may not function in the same way.

3.3 Terminal Requirements

The Z80 MCB-MT board in any Zilog system interfaces to any terminal using a standard 8-bit ASCII asynchronous transmission mode, with or without parity, on RS-232 or 20 mA current-loop. The interface uses four switches on the MCB-MT to set a binary value equivalent to a specific baud rate. The speeds and their corresponding switch positions are as follows:

SPEED	SETTING	SW1	SW2	SW3	SW4
50	0	ON	ON	ON	ON
75	1	OFF	ON	ON	ON
110	2	ON	OFF	ON	ON
134.5	3	OFF	OFF	ON	ON
150	4	ON	ON	OFF	ON
200	5	OFF	ON	OFF	ON
300	6	ON	OFF	OFF	ON
600	7	OFF	OFF	OFF	ON
1200	8	ON	ON	ON	OFF
2400	9	OFF	ON	ON	OFF
4800	10	ON	OFF	ON	OFF
9600	11	OFF	OFF	ON	OFF
19200	12	ON	ON	OFF	OFF
38400	13	OFF	ON	OFF	OFF

SECTION 4

DEBUG ENVIRONMENT

4.1 Breakpoints

The Z80 MCB-MT board allows the user to set software breakpoints for program debugging. A breakpoint is a command to suspend program execution whenever a specified instruction is executed. The address specified in the command is the address of the instruction. When encountered in the course of program execution, the breakpoint suspends execution of the user's program, saves all registers in a reserved memory area, and prints a message informing of the break and the address at which it occurred.

Any number of breakpoints can be set manually by setting the desired breakpoint location to 0FFH, which causes a trap to the breakpoint routine. The breakpoint location must be the first byte of an instruction, and when the breakpoint is no longer desired, the original instruction must be restored manually. The fact that 0FFH causes a break means that anytime a program erroneously attempts to execute from non-existent memory, a break occurs immediately.

The BREAK command saves the address at which the breakpoint is being set and the instruction the 0FFH is replacing. The BREAK command has an optional parameter of a repetition counter, which is also saved when specified. If a repetition counter with value N is specified, suspension of execution does not occur until the Nth time this breakpoint is encountered, unless some other breakpoint is encountered first. If the breakpoint is cleared (BREAK <CR>), the instruction modified with 0FFH is restored.

4.2 Program Restrictions

If interrupts are disabled when a breakpoint occurs, program execution will begin with interrupts enabled. Because some timing distortion occurs each time the breakpoint is encountered, the program should not be timing-dependent.

The program must not use Channel 3 of the CTC (Counter Timer Circuit), as this channel is used to implement the multiple execution feature. The breakpoint cannot be within an interrupt routine entered by an interrupt from Channels 0-2 of the CTC.

Both the breakpoint programs and the NEXT command use instruction modification and the interrupt system. Thus, the instructions being debugged cannot be in ROM, and they cannot involve modifications of either the interrupt status or the Interrupt (I) register. Also, the byte preceding the instruction in question is temporarily modified, and so must be in writable memory.

Whenever a JUMP or GO command is executed, the user stack is utilized. This implies that the stack must be set up in writable memory. Also, if the JUMP or GO address has a system breakpoint set, the execution of the instruction immediately following the GO or JUMP will not cause suspension of execution. Subsequent executions of the breakpoint locations suspends execution. This permits breaking and continuing execution without resetting the breakpoint as, for example, inside a loop.

When debugging system programs, verify that the I-register is set to 13H and that interrupts are enabled. Before starting debugging, set the stack pointer to where the debugger software will not interfere with it. Because the debugger uses 10-20 locations down from 1100H, location 10C0H should be used under normal circumstances.

4.3 Debug Commands

In the following command descriptions, the angle brackets (<>) enclosing descriptive names for the quantities to be entered should not be typed. Square brackets ([]) enclose optional quantities. Parentheses are used for grouping repetitive items.

The commands may be abbreviated to their first letter, with the exception of SAve and GEt, which must have at least two letters. If a command is not understood, the system issues a "?", followed by a command prompt, ">". All numbers may be entered in free-form hex, with leading zeros omitted. If more than four hex digits are entered, the last four will form the number used. All fields are blank-delimited.

An example of a valid Display command is:

```
D 1000 10
```

The following commands are implemented in the PROM Monitor:

Syntax	Description
B <addr> [<nth time>]	BREAK - Sets a breakpoint at the specified address, first clearing any breakpoint previously set. <nth time> denotes the number of times the breakpoint instruction is encountered before a breakpoint occurs. The default is 1. See description of breakpoints, Section 4.1.
C <addr1> <addr2> <no. of bytes>	COMPARE - Used to compare the contents (<no. of bytes>) of two blocks of memory. <addr1> and <addr2> specify the starting addresses of the two blocks. If any locations of the two blocks differ, the address and contents of those locations are displayed on the user's terminal.
D <addr> [<no. of bytes>]	DISPLAY - Displays contents of memory locations starting at the specified address, for the specified number of bytes. If <no. of bytes> is not specified, memory locations are displayed one at a time, with an opportunity to change each byte. For each byte, the address is displayed, followed by the contents of the location, followed by a space. To change the contents of that location, type in the new contents. A carriage return, either alone or after the new contents, causes the next sequential location to be displayed. A "Q" (Quit) followed by a carriage return terminates the command. If <no. of bytes> is specified, the contents of the designated memory locations are displayed both in hex notation and as ASCII character. The character

Syntax	Description
	representation appears to the right of the screen between asterisks with all non-printing characters appearing as periods (".") .
F <addr1> <addr2> <data>	
	FILL - Stores the specified data byte into all memory locations from <addr1> to <addr2>, inclusive. This range must not include any areas of ROM or non-existent memory.
GE [<drive>/] filename	GET - This command assumes that "filename" is a procedure file, and reads the file into core using the segment information in the file description record. If <drive> is not specified, the filename is searched for in the usual drive sequence order (1-7, 0). I/O errors are reported with the message: FILE ERROR hh where "hh" consists of two hex digits representing the RIO OS error code. Refer to the Error Code listing in the Z80 RIO Operating System User's Manual.
G	GO - Branches to the last stored program counter (PC), thus continuing program execution where it was last interrupted. All registers are restored.
I	INTERRUPT - Displays the status of the interrupt enable flip-flop as either 00 (disabled) or 01 (enabled) To change the status, enter the number corresponding to the desired status after the display; otherwise, enter a carriage return.
J <addr>	JUMP - Branches unconditionally to the specified address, thus executing the user's program. All registers are restored before branching.

Syntax	Description
M <dest> <source> <n>	MOVE - Used to move the contents of a block of memory from the source address specified by <source> to the destination address specified by <dest>. <n> is the length in bytes of the block to be moved. There are no restrictions on <source>, <dest>, or <n>.
N [<n>]	NEXT - Causes execution of the next machine instruction, starting at the current PC, and displays all registers after each instruction execution. If <n>, the number of instructions, is not given, 1 is assumed. Typing a carriage return, after a NEXT command, causes execution and tracing of the next instruction.
Q	QUIT - If Debug was properly entered from an external program, such as the RIO operating system, QUIT returns to that program; otherwise, it is an invalid command.
R [<register name>]	REGISTER - Allows the contents of the indicated register to be displayed and modified. If no register name is specified, all registers are displayed on one line. If a register name is specified, individual registers are displayed, starting with the one specified. For each register, the register name is displayed, followed by its contents, followed by a space. To change the contents of that register, type in the new contents. A carriage return, either alone or after new contents, causes the next register to be displayed. To terminate the command, enter "Q" (Quit), either alone or after new contents, then enter a carriage return.

Syntax	Description
R [<register name>] (cont.)	The sequence in which the registers are displayed when they are stepped from one to the next is: A, B, C, D, E, F, H, L, I, A', B', C', D', E', F', H', L', IX, IY, PC, SP. The registers are saved at the start of the program and at each breakpoint. They are restored at each JUMP or GO command.

SA [<drive>/] filename <start1><end1>...[<startn><endn>]
[E=entry] [RL=record length]

SAVE - Builds a procedure file on the specified drive. If no drive is specified, the file is built on the first available drive in the sequence 1,2,...,7,0. The SAVED procedure file contains segments specified by the address pairs <start> and <end>. The entry point of the SAVED procedure file is set to 0, and the stack size is set to 100H. Segment sizes are rounded up to multiples of 512. The file is overwritten if it is an old file. I/O errors are repeated with the message:

FILE ERROR hh

where "hh" consists of two hex codes representing the RIO OS error code. Refer to the Error Code listing in the Z80 RIO Operating System User's Manual.

S <addr> <data> [<data> [<data> ...]]

SET - Stores the specified data words into sequential memory locations starting at the specified address. A carriage return terminates the list.

4.4 Debug Interface

An external routine can interface with the PROM Debug package in one of two ways. Either way, a flag is set and a return address is stored in an associated location.

The first way is to make a call into the Debug environment, thus passing control to the Debug command interpreter until a QUIT command is issued. This is done by setting Bit 5 of location 13CDH (BRKFLG) in system RAM, storing the return address at locations 13BEH, 13BFH (EXTRET) and jumping to the Debug command interpreter at 0BFAH.

The second method is used when an external routine is to handle breakpoints. If Bit 7 of 13CDH (BRKFLG) is set, the occurrence of a breakpoint causes a jump to the location stored at 13CEH, 13CFH (BRKRTN).

SECTION 5

SYSTEM PARAMETERS

5.1 User-Accessible System Parameters

The following system parameters are user-accessible:

Parameters	Description
NULLCT	Null Count (13C8H) - The number of null characters and line feeds inserted after a carriage return is stored in this location. Modifying the null count adapts the TTY driver to the mechanical carriage return delays of various terminals.
LFCNT	Line Feed Count (13C9H) - The number of line feeds inserted after a carriage return is stored in this location. Modifying the line feed count permits automatic multiple spacing.
PROMPT	Prompt Character (13CAH) - The character output by the GET and TTY routines before reading a line from the terminal is stored in this location. Modifying the prompt character permits various levels of interactive software to identify themselves in each command query. Prompting can be effectively eliminated by setting this location to a null character (ASCII 0).
LINDEL	Line Delete (13CBH) - The character interpreted by the GET and TTY routines as a line delete is stored in this location. When a line delete is encountered in the ASCII input stream from the terminal, GET and TTY purge their buffers and continue reading the input stream.
CHRDEL	Character Delete (13CCH) - The character interpreted by the GET and TTY routines as a character delete is stored in this location. When character delete is encountered in the ASCII input stream, the last character entered is purged from the input buffer. Multiple character deletes delete multiple characters.

Parameters	Description
BRKFLG	Breakpoint Flag (13CDH) - Bit 5 of this location determines the return address for the QUIT command. Bit 7 is used to signal the existence of an external breakpoint (see Section 4.3).
BRKRTN	Breakpoint Return (13CEH, 13CFH) - These locations are used with BRKFLG to make use of an external breakpoint handler (see Section 4.3).
EXTRET	External Return (13BEH, 13BFH) - These locations are used with BRKFLG when calling the Debug command interpreter (see Section 4.3).

5.2 Port Addresses

The following equates define the system I/O devices.

SYSTEM HARDWARE I/O PORT ADDRESSES

PORT	ADDRESS
USART DATA (TTYDAT)	DE
USART CONTROL/STATUS (TTYSTT)	DF
CTC CHAN 0 (CLK0)	D4
CTC CHAN 1 (CLK1)	D5
CTC CHAN 2 (USART TRANSMITTER READY)	D6
CTC CHAN 3 (USART RECEIVER READY)	D7
PIO DATA A	D8
PIO CONTROL A	DA
PIO DATA B	D9
PIO CONTROL B	DB
DISK SHIFT REGISTER (DSKDAT)	CF
DISK PIO DATA A (DSSTAT)	D0
DISK PIO CONTROL A	D2
DISK PIO DATA B (DSKSEL)	D1
DISK PIO CONTROL B	D3
SWITCH BANK (TTYSPD)	DD

5.3 PROM Monitor Routine Request and Error Codes

The following request codes are satisfied by the PROM routines.

PROM MONITOR I/O ROUTINE CODES

FLOPPY REQUEST CODE	VALUE
RDBIN	0A
WRTBIN	0E

TTY REQUEST CODE	VALUE
RDBIN	0A
RDLIN	0C
WRTBIN	0E
WRTLIN	10

Error codes that may occur are as follows:

ERROR CODES

CODES	VALUE
NORMAL RETURN	80
INVALID OPERATION REQUEST	C1
DISK NOT READY	C2
DISK WRITE PROTECTED	C3
SECTOR ERROR	C4
TRACK ERROR	C5
CRC ERROR	C6

5.4 Device Command Codes

The commands sent to the I/O devices are defined as follows:

I/O CONTROL PROM MONITOR RESET VALUES

USART CONTROL WORDS	VALUE
MODE INSTRUCTION	CE
COMMAND INSTRUCTION	27

CTC CHANNEL 0 CONTROL WORDS	VALUE
VECTOR REGISTER	E0
CONTROL REGISTER	D3
TIME CONSTANT	CF

DISK PIO CONTROL WORDS	VALUE
PORT A CONTROL	CF
PORT A I/O SELECT	E0
PORT B CONTROL	CF
PORT B I/O SELECT	E0

CPU INITIAL STATE	VALUE
INTERRUPT MODE	2
INTERRUPT VECTOR	13
INTERRUPT FLIP/FLOP	ENABLED
STACK POINTER	1100

INDEX

B

BACKSPACE, 5
baud rate, 7
binary operations, 5
bootstrap, 1, 7
BREAK, 9, 11
breakpoint, 9, 10, 15
BRKFLG, 18
BRKRTN, 18
buffer, 5

C

character delete, 5
CHRDEL, 17
command codes, 19
command scanner, 1
command string, 1
COMPARE, 11
completion code, 3, 4
completion return address, 3
CONTROL H, 5
CRC error, 4
CTC, 18, 20
Cyclic Redundancy Checking, 4

D

data length, 3
data transfer address, 3
Debug, 15
Debug commands, 10
Debug environment, 7
debugger, 10
defaults, 7
Delete Current Line, 6
Delete Last Character, 6
disk address, 3
DISPLAY, 11

E

echoing, 5
entry point, FLOPPY, 3
entry point, TTY, 5
environment, 15
error, 6
error codes, 19
error return address, 3
external routine, 15
EXTRET, 18

F

FILE ERROR, 11
FILL, 11

G

GEt, 10, 11
GO, 10, 11, 14

I

I/O devices, 18
I/O error, 11
initialization, 1
INTERRUPT, 11
interrupt routine, 4, 6
interrupt status, 9
interrupt system, 9
I-register, 10

J

JUMP, 10, 11, 14

L

LFCNT, 5, 17
LINDEL, 17
line delete, 5
line-edit, 6

M

MCB-MT board, 7, 9
memory, 1, 6
MOVE, 13
Multiple character deletes, 17

N

NEXT, 9, 13
NULLCT, 5, 17

P

parameter vector, 3
PIO, 1, 18, 20
port addresses, 18
procedure file, 14
program counter, 11
program restrictions, 9
PROMPT, 17

Q

QUIT, 13, 15

R

RDBIN, 3, 5, 19
RDLIN, 5, 19
REGISTER, 13
repetition counter, 9
reset values, 19
RET, 4, 6
RETI, 4, 6
routine codes, 19

S

SAve, 10, 14
sector address, 3
sector error, 3, 4
serial interface, 5
SET, 14
stack pointer, 10, 20
SWITCH, 18

switches, 7
system parameters, 17

T

timing distortion, 9
track address, 3
track error, 4
transfer length, 3, 5

U

USART, 1, 18

W

write protected, 4
WRTBIN, 3, 5, 19
WRTLIN, 5, 19

Zilog, Inc. 10460 Bubb Road, Cupertino, California 95014 Telephone (408)446-4666 TWX 910-338-7621

Printed in USA