Price: $2.00
03-0075-01
Revision A
July 1978

TABLE OF CONTENTS

1.0 INTRODUCTION

The 3K MCB Monitor is intended to provide the basic debugging commands, basic Input/Output, and bootstrap portions of a floppy disk-based 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 0, and uses lK bytes of RAM, including 256 bytes allocated for the system stack. The monitor for the MCZ l/l0 is a 4K monitor.

At the time of system RESET, the 3K Monitor reads the switch bank to determine the baud rate required at the serial interface. Then all variable parameters are set in RAM to their initial values. Next, the USART is programmed for asynchronous operation with two transmit stop bits, no parity, 8 data bits, and divide by sixteen operation. After this the USART is further programmed to generate active (low) signals on REQUEST TO SEND and DATA TERMINAL READY and both the receiver and transmitter are enabled. Finally, the PIO in the disk interface is set to the bit control mode on both of its ports and appropriate directions are assigned to the bits.

At this point, the Monitor fetches its first command string from the terminal. If that command is a simple carriage return, the bootstrap logic is entered. If not, the Debug command scanner is entered.

The bootstrap procedure involves reading in l28 bytes of data, then transferring control to it. If a disk is not attached, this data will be read in binary form without echo from the terminal input. This permits a paper tape bootstrap for early testing of routines which are to drive the disk. In general, though, it will not be necessary to resort to a paper tape bootstrap.

2.0 FLOPPY DISK DRIVER

The floppy driver (FLOPPY) is used to read and write from the disk. The FLOPPY routine is only capable of accessing pre-formatted Zilog diskettes. 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

There are two valid requests for the floppy driver. They are RDBIN (0AH), which will read data from the disk to the data transfer area, and WRTBIN (0EH), which will write data from the data transfer area to the disk.

If the request code is increased by one, then the driver will return as soon as the request has been initiated, permitting it to continue under interrupt control and jumping to the completion return address after the operation. The routine at completion return address should act as an interrupt routine, except that it should not execute a RETI. (A simple RET should be used). If the request code is used as given, FLOPPY will return after the operation is completed. In this case the completion return address is ignored.

If the data transfer length is not divisible by the sector size, i.e., 128 bytes, it will be increased to the next integral multiple of the sector size. Data will be 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 will result on a read operation and a permanent sector error will be written on a write operation, resulting in certain sectors being permanently inaccessible.

If an error occurs and the error return address bytes are non-zero, the routine at that address will be called. It may take whatever action is appropriate and return. If the address is zero, the return would be as though there were no error.

COMPLETION CODE - Bit 7 of the condition code byte is set when the operation is completed. Bit 6 is set to indicate that an error occurred. The bottom six bits indicate which error it was. Possible return codes are:

*Normal Return (80H)
*Invalid Operation Request (ClH)
*Not Ready - disk is signalling a "Not Ready" condition (C2H)
*Disk is Write Protected (C3H)
*Sector Error - the sector address in the sector header did not agree with the sector position (C4H)
*Track Error - the track address in the sector header did not agree with the head position (C5H)
*CRC Error - the Cyclic Redundancy Checker - CRC indicates one or more data bits in error (C6H)

The disk address at which the transfer is to begin is represented as a track address from 0 to 77 decimal and a sector address from 0 to 3l decimal. Additionally, three bits indicating which of the eight possible units is to be used for the transfer are carried as the top three bits of the sector address. The track address is considered to be the most significant part of the disk address (and therefore goes in IY+l2) while the sector address is considered to be the least significant (and therefore goes in IY+ll).

The entry point for the floppy driver is at location 0BFDH.

3.0 TTY DRIVER

The TTY driver is used to read and write from the terminal. On entry, IY should be pointing to a standard parameter vector identical in format to that used by FLOPPY with the exception, of course, of the last field (disk address), which is ignored. Because of the differences of the hardware devices in use, the operation of these routines is necessarily different. The significance of parameters to TTY is as follows:

There are four requests handled by terminal driver. These are RDBIN (0AH), WRTBIN (0EH), RDLIN (0CH), and WRTLIN (l0H). 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 is performed on RDBIN.

The RDLIN operation begins with the issuing of a prompt character to the terminal and proceeds to input characters to the data transfer area from the special interface. All input characters are echoed. Special line delete (initialized by the 3K Monitor to "!" or ASCII 2l) and character delete (initialized by the 3K Monitor to "@" or ASCII 40) are honored by this routine. A RDLIN operation will terminate either on receipt of the number of characters specified as the transfer length or on receipt of a carriage return. The carriage return will be echoed and the number of line feeds and null characters specified in the corresponding parameter storage locations will be generated.

The WRTLIN operation also proceeds either for the number of characters specified as the transfer length or until a carriage return is encountered. After a carriage return is encountered, the end of line sequence described above (insertion of LF and NULs) is performed.

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

If one is added to the request code, control will be transferred to this address after the operation. The routine at this address should act as an interrupt routine, except that it should not execute a RETI (a simple RET should be used). If one is not added to the request, TTY will return normally after the operation is 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 at that address will be called. It may take whatever action is appropriate and return. If the address is zero, the return is normal.

Bit 7 of the completion code is set when the operation is completed. Bit 6 is set to indicate that an error occurred. The only error condition returned is Invalid Operation. This will occur for request codes greater (arithmetically) than those honored by TTY – lower request codes return complete with no error, but with no effect, either.

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:

To change the Delete last character command:

> Set memory location l3CCH to the ASCII code for the character desired. For example, to make the '@' the character delete, one would set the location to 40H.

To change the Delete current line command:

> Set memory location l3CBH to the ASCII code for the character desired. For example, to make '!' the line delete, one would set the location to 2lH.

These characters are also changed when OS is bootstrapped to its defaults or by the Set command in RIO OS. This means that if Debug is entered from OS, the characters may not function in the same way.

The entry point for the TTY driver is at 0BE8H.

Terminal Requirements:

The Z80 MCB 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 will operate at l4 different speeds which are selected using 4 switches as a binary number. The speeds and their corresponding switch positions are as follows:

SPEED	SETTING	SW. 1	SW. 2	SW. 3	SW. 4
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

4.0 DEBUG ENVIRONMENT

4.1 BREAKPOINTS

The Z80 MCB provides the capability 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 will suspend execution of the user's program, save all registers in the memory area provided, and print 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 will cause a trap to the breakpoint routine. The breakpointed 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 FF causes a break means that anytime a program erroneously attempts to execute from non-existent memory, a break will occur immediately). The BREAK command, when entered from the keyboard, saves the address at which it is being set and the instruction which it is replacing. When the breakpoint is cleared, the instruction is automatically restored. It also stores a repetition counter, N. Suspension of execution will not occur until the Nth time this breakpoint is encountered, unless, of course, some other breakpoint is encountered first.

There are some restrictions on the program in order to use this feature. The program will begin execution with interrupts enabled after encountering the breakpoint, and the program should not be timing-dependent, as there will be some timing distortion each time the breakpoint is encountered. The program must not use channel 3 of the CTC, as this is used to implement the multiple execution feature, and the breakpoint cannot be within an interrupt routine entered by an interrupt from channel 0-2 of the CTC.

Both the breakpoint programs and the Next command make use of 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 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 it must be set to some address which has writable memory implemented. 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 will do so, however. This is to permit breaking and continuing execution without resetting the breakpoint (as, for example, inside a loop).

If system programs are to be debugged, it is important to verify that the I-register is set to 13H and that interrupts are enabled. It will also be helpful, before starting debugging, to set the stack pointer someplace where the debugger software will not interfere with it. Since the debugger uses 10-20 locations down from 1100H, 10C0H is a good choice under normal circumstances.

4.2 DEBUG COMMANDS

In the following command descriptions, angle brackets ('<>') are used to enclose descriptive names for the quantities to be entered, and are not actually to be entered. Square brackets ('[]') enclose optional quantities. Parentheses are used for grouping repetitive items. For example, a valid Display command (see below) would be

```
D  1000  10
```

The following are the commands implemented in the 3K Monitor:

BREAK <address> [<n>] - Sets a breakpoint at the given address, first clearing any breakpoint previously set. If n is given, the program execution will not be interrupted until the nth time the instruction is encountered. See description of breakpoints.

COMPARE <n1> <n2> <n3> - The Compare command is used to compare the contents of two blocks of memory. n1 and n2 specify the starting addresses of the two blocks, while n3 specifies the number of bytes to be compared. If any locations of the two blocks differ, the addresses and contents of those locations are printed on the user's terminal.

DISPLAY <address> [<number of bytes>] - Displays on the terminal the contents of memory locations starting at the given address, for the given number of bytes. If no number of bytes is given, memory locations will be displayed one at a time, with an opportunity to change each byte. For each byte, the address will be typed, followed by the contents of the location, followed by a space. If it is desired to change the contents of that location, the new contents can be typed. A carriage return, either alone or after the new contents, will cause the next sequential location to be displayed. A 'Q' (for Quit) followed by a carriage return will terminate the command. If a number of bytes is given, the contents of the designated memory locations are displayed both in hex notation and as ASCII characters. The character representation appears to the right of the screen between asterisks with all non-printing characters appearing as periods ('.').

FILL <addrl> <addr2> <data> - Stores the data byte given into all memory locations from addrl to addr2. This range must not include any areas of ROM or non-existent memory.

GO - Branches to the last stored PC, thus continuing program execution where it was last interrupted. All registers are restored.

INTERRUPT - Displays the status of the interrupt enable flip-flop as either 00 (disabled) or 0l (enabled). If it is desired to change the status, enter the number corresponding to the desired status after the display; otherwise, simply hit carriage return.

JUMP <address> - Branches unconditionally to the given address, thus executing the user's program. All registers are restored before branching.

MOVE <dest> <source> <n> - The Move command is 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 of the block to be moved. There are no restrictions on <source>, <dest>, or n, except that they be positive integers, and that the blocks be bounded within the 64K bytes addressed by the Z80 CPU.

NEXT [<n>] - Causes the execution of the next machine instruction, starting at the current PC, and displays all registers after each instruction execution. If n is not given, l is assumed. After a Next command, typing a carriage return will cause execution and tracing of the next instruction.

QUIT - If Debug was properly entered from an external program (such as OS), Quit will return to that program; otherwise, it is an invalid command.

REGISTER [<register name>] - Permits the contents of the indicated register to be examined and modified. If no register name is given, all registers will be displayed on one line. If a register name is given, individual registers will be displayed starting with the one given. The register name will be typed followed by its contents, followed by a space. If it is desired to change the contents of that register, the new contents can be typed. A carriage return, either alone or after new contents, will cause the next register to be displayed. If no more registers are desired, a "Q" (for Quit) should be entered, either alone or after new contents, followed by a carriage return. This will terminate the command.

The registers being examined and modified are memory locations in which the indicated registers are saved at the start of the program and at each breakpoint, and from which they are restored at each Jump or Go command.

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.

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

All of the above commands may be abbreviated to their first letter, or may have the command spelled out to any desired length, with the exception of Save and Get, which must have at least two letters. The first character typed on a new line will be taken as the key to which command is being invoked. If a command is not understood, a "?" will be typed and a new command requested. 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.

4.3 DEBUG INTERFACE

There are two ways in which an external routine can interface with the PROM Debug package. With either, 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 l3CDH (BRKFLG) in system RAM, storing the return address at locations l3BEH, l3BFH (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 l3CDH (BRKFLG) is set, the occurrence of a breakpoint will cause a jump to the location stored at l3CEH, l3CFH (BRKRTN).

5.0 SYSTEM PARAMETERS

There are several system parameters which are accessible to the user. They are:

NULLCT -- Null Count (l3C8H)
: In this location, the number of null characters which will be inserted after a carriage return (and whatever number of line feeds which are also inserted) is stored. Modifying the null count is the means of adapting the MCS to the carriage return delays of various terminals.

LFCNT -- Line Feed Count (l3C9H)
: In this location, the number of line feeds which will be inserted after a carriage return is stored. Modifying the line feed count permits automatic multiple spacing.

PROMPT -- Prompt Character (l3CAH)
: In this location, the character output by the GET and TTY routines before reading a line from the terminal is stored. 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 (l3CBH)
In this location, the character interpreted by the GET and TTY routines as a line delete is stored. When it is encountered in the ASCII input stream from the terminal, these routines purge their buffers and continue reading the input stream.

CHRDEL -- Character Delete (l3CCH)
In this location, the character interpreted by the GET and TTY routines as a character delete is stored. When it is encountered in the ASCII input stream, the last character entered is purged from the input buffer. Multiple character deletes may be used to delete the last "n" character entered.

BRKFLG -- Breakpoint Flag (l3CDH)
Bit 5 of this location is used to determine 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 (l3CEH, l3CFH)
This location is used with BRKFLG to make use of an external breakpoint handler (see Section 4.3).

EXTRET -- External Return (l3BEH, l3BFH)
This location is used with BRKFLG when calling the Debug command interpreter (see Section 4.3).

There are several equates used by the system. They are:

SYSTEM HARDWARE I/O PORT ADDRESS

PORT	ADDRESS
USART DATA (TTYDAT)	DE
USART CONTROL/STATUS (TTYSTT)	DF
CTC CHAN 0 (CLK0)	D4
CTC CHAN 1 (CLK1)	D5
CTC CHAN 2	D6
CTC CHAN 3	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

3K 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	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

I/O CONTROL 3K 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