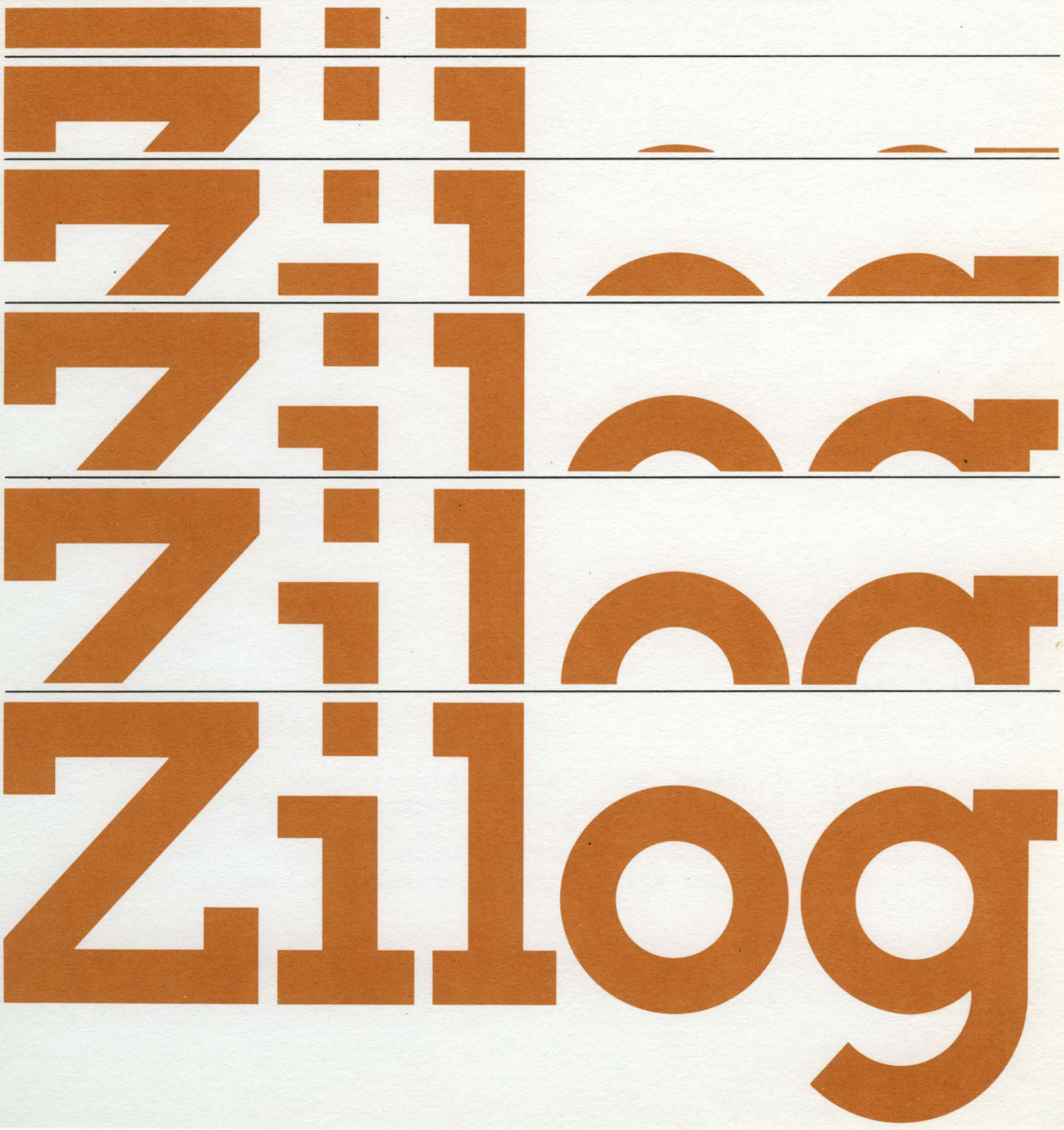
MCZ-1/20A and MCZ-1/25A MICROCOMPUTERS RIO UTILITIES

Zilog

User Guide

December 1979



03-3121-01, Rev. A

December, 1979

MCZ-1/20A and MCZ-1/25A MICROCOMPUTERS RIO UTILITIES

User Guide

December 1979

PREFACE

This manual describes how to use the RIO Utilities for the MCZ-1/20A and MCZ-1/25A microcomputers. The RIO Utilities described herein run on MCZ-1/20A,-1/25A systems with 64K memory. A complete description of the features in RIO can be found in the Z80 RIO Operating System User's Manual. These utilities provide the following services for the user:

- File printout
- File modification without program relinking
- Memory test

The standard port assignments and correct switch settings necessary to use the utilities are provided in Section 2. A method for changing port assignments to meet particular system configuration requirements is also described.

The file printout capability is supported by a choice of a standard or optional printer driver, described in Section 4 and 5, respectively. These drivers must be linked at high memory before they can be used, as described in Section 3.

The PATCH program, described in Section 6, allows the user to append a new segment of code to an existing RIO procedure file, or to modify bytes in any kind of file.

A thorough diagnostic capability is provided by the memory test program, described in Section 7.

CONTENTS

CONTENTS (cont.)

LIST OF TABLES

SECTION 1

INTRODUCTION

1.1 General Description

The utilities for the MCZ-1/20A,-1/25A consist of drivers for line printers with a "Centronics parallel" interface, a test program for system memory, and a PATCH program for appending code or modifying bytes. The utilities are contained on a single floppy disk or the system disk cartridge, which has object files (named xxx.OBJ), and command files for linking (named LINK.xxx). The source files MCZ.EQUATES.S, PRINTER.S, and LP.S are also included for user convenience.

1.2 Release Dates

Each of the utilities has a release date programmed into the code that may be examined in the source, object, or procedure files. The correct release date codes for this utilities release are:

UTILITY	RELEASE DATE
Printer Driver	790201.1
Optional Printer Driver	781015.1
Memory Test	780719.1
PATCH	790711.1

SECTION 2

PORT ASSIGNMENTS

2.1 Standard Port Assignments

The standard I/O port address values assigned to the MCZ systems are shown below. In order for individual utilities to work correctly, the switches on the appropriate interface board must be set so that they agree with the most significant digit (hex) of the standard port address.

UTILITY	PORT ADDRESSES USED
Printer Driver	0D8H - 0DCH *
Memory Test	None

* Z80 Assembly Language convention requires that a zero be inserted preceding alpha hex numbers to distinguish the addresses from labels.

2.2 Changing Port Assignments

The standard port assignments may be changed to meet particular configuration requirements, as follows:

1. Use the RIO Editor to change the port assignments in the file MCZ.EQUATES.S

2. Reassemble the edited file as follows:

   %ASM MCZ.EQUATES (A NOL)

3. Relink the driver (refer to Section 3)

SECTION 3

DRIVER LINKING

3.1 Link Addresses

The printer drivers must be linked at high memory before they can be used. The standard link address for the standard printer is 0FB00H; and 0F980H for the optional printer.

If the system is configured with a hard disk drive or has more than two floppy drives activated, the link addresses will be different. The user can determine the correct link address by using the DISPLAY command (see the Z80 RIO Operating System User's Manual) to find the highest unallocated memory, and then allowing enough room for the driver to be loaded. The standard printer driver, PRINTER, requires 200H bytes, while the optional printer driver, LP, needs 380H bytes.

Example:

The system is configured with 64K of memory and has four active floppy drives. All memory above 0FA00H is allocated to these floppy drives. The printer should be linked as follows:

%DO DRIVER.LINK F800 PRINTER

or

%DO DRIVER.LINK F680 LP

3.2 RIO DO File

The RIO DO file, DRIVER.LINK, is included to assist the user in linking the drivers. It requires a parameter list consisting of the link address followed by one to three driver names. The command file also identifies the resultant procedure file as a driver to the RIO operating system. The driver names are as follows:

DRIVER	NAME
"Centronics parallel" printer	PRINTER
"Centronics parallel" printer with optional capability	LP

SECTION 4

STANDARD PRINTER DRIVER

4.1 Introduction

Zilog provides a printer driver (PRINTER) to interface to a "Centronics parallel" interface line printer within the RIO environment.

4.2 Driver Implementation

The program PRINTER is the driver required to interface a line printer with the RIO system. The driver accepts a standard RIO parameter vector pointed to by the IY register. This driver recognizes the I/O requests listed below; all others are considered invalid:

0 - Initialize request: initializes the printer interface

2 - Assign request: initializes the printer interface

4 - Open request: null operation -- returns operation complete

6 - Close request: null operation -- returns operation complete

E - Write Binary request: outputs number of bytes specified by the length field of the parameter vector or until an end-of-file (FFH) is found

10 - Write ASCII request: outputs number of bytes specified by the length field of the parameter vector or until a carriage return (0DH) is found

The PRINTER driver is provided as an object file (output of the assembler). This file must be linked prior to using the PRINTER driver. Instructions for linking are found in Section 3 of this document.

Before output can be made to the driver, PRINTER must be loaded and initialized via the ACTIVATE command using one of the following three methods:

Method #1

Enter a keyboard command:

%ACTIVATE $PRINTER

Method #2

Execute from a program the command:

ACTIVATE $PRINTER

To do this, format a parameter vector as follows:

```
Byte 0   - Zero
     1   - Unused
   2-3   - Pointer to buffer containing command string
   4-7   - Unused
   8-9   - Error handler address
    A    - Completion code
```

For more details on this method, see Section 4.5, System Calls in the Z80 RIO Operating System User's Manual.

Method #3

Incorporate the command (ACTIVATE $PRINTER) for loading and initializing the driver into the OS.INIT "DO" command file that is executed every time the RIO system is bootstrapped. Note that if requirements of the PRINTER program conflict with those of the DO command, a MEMORY PROTECT VIOLATION error results. If so, relink the DO command. See the Z80 RIO Operating System User's Manual, Appendix L.

As delivered, the printer driver prints a line 85 characters wide. Lines over 85 characters are truncated. To change the line width, the user must:

1. Edit MCZ.EQUATES.S. Change the value of LINVAR (line variable) a global equate, to the desired line width.

2. Reassemble MCZ.EQUATES with the absolute option:

   %ASM MCZ.EQUATES.S (A)

3. Relink the printer driver (refer to Section 3).

SECTION 5

OPTIONAL PRINTER DRIVER

5.1 Introduction

This section describes the optional RIO printer driver, LP, which interfaces a Centronics model 306C printer to an MCZ system. The program is also suitable for driving any printer with a Centronics-compatible interface, although some small changes to the code may be needed for correct operation. Examples of such printers are the Tally 1202 or 1602, and the Wang 200W.

The printing format, may be altered at any time under software control or with the supplied utility program, SET.LP.

5.2 Hardware Requirements

The electrical interface to the printer is implemented as described in the Z80 MCB-MT Hardware Reference Manual.

5.3 Installing the Driver

Like any other device driver, LP is loaded into memory and made known to the RIO operating system by use of the ACTIVATE command. This command may be entered from the console keyboard in the following manner:

%ACTIVATE $LP

Alternatively, a user program can activate the driver by making a system call with the IY register pointing to a vector, the first byte of which is zero, while the third and fourth contain the start address of a string:

ACTIVATE $LP<carriage return>

Note that the user program must not reside in the same area of memory as the RIO ACTIVATE command.

The mechanism of system calls from user programs is described in more detail in the Z80 RIO Operating System User's Manual.

It is possible that RIO will respond to the ACTIVATE command with a "MEMORY PROTECT VIOLATION" message. In this case, the driver must be relinked at an address where there is free memory. The same applies if any other program cannot be loaded after the driver has been loaded. Section 5.14 contains advice on relinking.

Upon receiving the command "ACTIVATE $LP", RIO loads the driver into memory, then sends it an Initialize command. As a result of this command, the driver sends a Select command to the printer interface. When a "handshake" signal has been returned by the printer, a form feed character is output, causing the printer to advance the paper to the top of the next page. If the printer is not connected to the system or is not powered on, the driver will wait indefinitely until the printer is ready. Such a wait can usually be terminated by powering the printer on, connecting it to the system, then operating the manual Select or Line switch on the printer. If this fails, the system must be re-booted.

5.4 Using the Driver

Once the driver has been activated, it may be used by the RIO utilities and system software as well as by the user's program.

Certain RIO utilities send output to a named device, as in the following examples, where LP is the named device.

%CAT L=$LP

prints a catalog of all non-secret files kept on the master device.

%COPY OS.INIT $LP

prints the file OS.INIT.

%ASM LP (L=$LP)

assembles the driver source file, printing the listing during the second pass. No ZDOS listing file is produced because listing output is directed to the printer.

Other utilities direct certain output to the system volume output device, SYSLST, logical unit 3 by default. LP can be defined to be this device by use of the command,

%DEFINE SYSLST $LP

or, alternatively

%DEFINE 3 $LP

Once this logical unit assignment has been made, volume output from system programs is sent, by default, to the printer:

%CAT

prints a catalog of all non-secret files kept on the master device.

%MOVE D=$NULL P=&

reads all files on the master device, Drive 0, and then writes them to the null device. A listing of all files moved is sent to the printer, while errors are reported on the console output device.

%ASM LP (X P)

assembles the driver source file, printing a listing followed by a cross-reference listing. A ZDOS listing file is also generated.

User programs may use the driver either implicitly, by directing output to logical unit 3, or explicitly, by assigning a unit number to the driver, then directing output to that logical unit. The process of assignment is described in the Z80 RIO Operating System User's Manual. Refer to Section 5.11, below, for details of the software interface to the driver.

5.5 Automatic Installation of the Driver

The printer driver can be activated whenever the system is bootstrapped by incorporating the commands required for the activation into the OS.INIT file. RIO automatically executes the command "DO OS.INIT" when it is bootstrapped. Thus, any commands in the file are automatically invoked.

Example:

The user edits OS.INIT, and inserts the following commands:

```
BRIEF
ECHO RIO with installed printer driver
ACTIVATE $LP
DEFINE SYSLST $LP
SET TABSIZE=4
SET.LP LEFT_MARGIN=5 RIGHT_MARGIN=80
```

DATE 780708
APPLICATION

The system is then booted as follows:

1. Brief mode is selected and a message is displayed on the console.

2. The printer driver is activated and defined to be the system volume output device.

3. The default system tab size of eight is changed, as are some of the default options of the printer driver.

When all this has been done, the date is printed on the console, and control is transferred to the user program named "APPLICATION". The SET.LP utility named in the example above is described in Section 5.12.2.

5.6 Removing the Driver

When the driver is no longer required, the DEACTIVATE command is invoked to erase the RIO record of its use, and free the memory it occupied, as follows:

```
%DEACTIVATE $LP
```

Note that RIO automatically redefines logical unit numbers allocated to a device when it is deactivated. It is therefore not necessary to use the DEFINE command to reallocate these logical numbers following the deactivation. However, if the default device for a given logical unit number is being deactivated, that logical unit will be made unknown to RIO after the deactivation. For further details concerning the Deactivate command, see the Z80 RIO Operating System User's Manual, Section 5.12.

5.7 Demonstrating the Use of the Driver

The supplied file PRINTER_DEMONSTRATION demonstrates the use and capabilities of the driver. The demonstration can be run by entering the command,

```
%DO PRINTER_DEMONSTRATION
```

An adequate supply of paper at least 80 columns wide is necessary for the demonstration. Some of the demonstrated features are specific to the Centronics model 306C printer, and so may not function with other types of printers.

The command file uses the supplied utility, SET.LP, to alter the format control options of the printer. This utility is described in Section 5.12.2.

5.8 Software Interface

The software interface for the optional printer involves a correct calling sequence, completion of requests, and return sequence.

5.8.1 Calling Sequence

In order to make the driver action a command, control must be passed to its entry point, which has the global name "LP". The IY register must point to a RIO parameter vector having the format shown in Table 5-1.

Table 5-1. Parameter Vector Format

Byte	Contents	Notes
0	Logical unit	Not used by this driver
1	Request code	See Section 3.2
2-3	Data transfer address	
4-5	Data length in bytes	
6-7	Completion return address	Not used by this driver
8-9	Error return address	See Section 3.12
10	Completion code	See Section 3.13
11-12	Supplemental parameter address	Not used by this driver

5.8.2 Logical Unit Assignment

It is strongly recommended that a user make use of RIO's logical unit I/O structure to access the driver. This frees the user from the requirement of knowing the address of the driver's entry point. It also brings the benefit of RIO's error handling facilities.

In order for the driver to be called through RIO by a user program, the driver must first be assigned a logical unit number by sending an assign request naming the driver to RIO. Section 3.3 of the Z80 RIO Operating System User's Manual describes the format of this request.

Once the logical unit assignment has been made, the user can call the RIO entry point, SYSTEM, with IY pointing to a vector in the format shown in Table 5-1. RIO examines the logical unit field of the vector, then passes control to the device driver corresponding to that logical unit.

5.8.3 Return Sequence

When the requested operation has been completed, or when an error condition has arisen, control returns to the calling program. A completion code is returned in the request vector. In addition, if an error has arisen, and the completion code is not the normal "operation complete" (80 hex), the return code is also written into the system error flag location, ERCODE.

On return from Write Binary, ASCII and Status, and from Read Status requests, the data length field of the vector is set to the actual number of bytes transferred.

Exit from the driver is usually via a Return. If Return is due to an error condition and the error return address field of the request vector is non-zero, the driver exits by jumping to that address.

The driver may corrupt all registers, including the alternate set, except IY.

5.8.4 Possible Errors

Three possible errors can occur when the driver is called. Table 5-2, below, lists these errors.

Table 5-2. Possible Errors

Return Code (hex)	Error	Cause
47	Nonexistent command	Unsupported request code
49	Program abort	Escape character (1B Hex) input on console during Write Binary request.
C9	End of file	End-of-file mark (FF Hex) found during Write ASCII operation.

5.9 Operations Supported

Table 5-3 summarizes the operations supported by the driver and lists the corresponding request codes. Each operation is described in detail in the paragraphs that follow.

Table 5-3. Operations Supported

Operation	Request Code (hex)
Initialize	00
Assign	02
Open	04
Close	06
Write Binary	0E
Write ASCII	10
Read Status	40
Write Status	42
Deactivate	44

Note that the driver does not support immediate return following initiation of an operation.

If the user requests an operation not listed in the table, the driver returns a "nonexistent command" error (see Sections 5.8.2 and 5.8.3).

5.9.1 Initialize (Request Code 00)

The driver performs the Initialize command by first programming the electrical interface to the printer, then sending Select and Form-Feed characters to the printer. This puts the printer on-line and makes it advance the paper to the top of the next page. Control returns to the calling program as soon as the form-feed has been sent.

5.9.2 Assign (Request Code 02)

A Select code is sent to the printer when the driver receives an Assign request, insuring that the printer is on-line.

5.9.3 Open (Request Code 04)

An Open request sends the printer a Select character, putting it on-line.

5.9.4 Close (Request Code 06)

The printer is "closed" by sending it a "deselect" code, putting it off-line.

5.9.5 Write Binary (Request Code 0E Hex)

Characters from the buffer specified by the data transfer address field of the request vector are written to the printer until one of the following conditions is met:

1. The number of characters specified in the data length field of the vector has been transferred.

2. An end-of-file mark (FF hex) has been found in the buffer.

3. An escape character has been input from the console keyboard while the printer driver's Abort option is enabled (see Section 5.11.3).

In the last case, return to the caller is with a status of "program abort" instead of the normal "operation complete".

The data sent to the printer is modified by the format control options in force at the time of the request. The options are described in Section 5.11.

Note that the printer will not accept data unless the printer has been "selected" (on-line). If a Write Binary operation is attempted when the printer has not been selected, the driver will wait indefinitely for it to become ready. This "hang up" can be terminated by operating the manual Select or Line switch on the printer.

The operator may suspend a printer Write Binary operation by putting the printer off-line. The operation will be resumed when the operator reselects the printer.

5.9.6 Write ASCII (Request Code 10 Hex)

Characters from the buffer specified by the data transfer address field of the request vector are written to the printer until one of the following conditions is met:

1. The number of characters specified in the data length field of the vector has been transferred.

2. A carriage-return character has been transferred.

3. An end-of-file mark (FF Hex) has been found in the buffer.

In the last case, return to the caller is with a status of "end-of-file" instead of the normal "operation complete".

The data sent to the printer is modified by the format control options in force at the time of the request. The options are described in Section 5.11.

Note that the printer will not accept data unless it has been "selected" (on-line). If a Write ASCII operation is attempted when the printer is off-line, the driver will wait indefinitely for it it to become ready. This "hang up" can be terminated via the manual Select or Line switch on the printer.

5.9.7 Read Status (Request Code 40 Hex)

Data is copied from the driver's status buffer to the buffer specified in the data transfer address field of the request vector. Four bytes, or the number specified in the data length field, whichever is smaller, are transferred. The format of the status buffer is described in Section 5.12.1.

5.9.8 Write Status (Request Code 42 Hex)

Data from the buffer specified in the data transfer address field of the request vector is copied into the driver's status buffer. Four bytes, or the number specified in the data length field, whichever is smaller, are transferred. The format of the status buffer is described in Section 5.12.1.

5.9.9 Deactivate (Request Code 44 Hex)

The driver deactivates the printer by sending it a "deselect" code, putting it off-line.

5.10 Control Characters

Certain characters are not sent directly to the printer, but are, instead trapped by the driver. Table 5-4 lists these characters. The action taken by the driver is described in the following paragraphs.

Table 5-4. Printer Control Characters

Character Code (hex)	Character
09	Horizontal tab
0A	Line feed
0C	Form feed
0D	Carriage return
0E	Extended typeface
FF	End of file

In general, control characters are passed directly to the printer and are considered to have the same printing width as printable characters.

5.10.1 Horizontal Tab

The Horizontal Tab character is not passed to the printer. Instead, spaces are output until the current print position corresponds to that of the next tab stop. Tab stops are defined by the state of the console status buffer at the time the driver is called. Sections 3.4.3 and 5.28 of the Z80 RIO Operating System Users Manual, describe the console status buffer and the means of altering tab positions.

5.10.2 Line Feed

Line Feed characters are only passed to the printer if they occur as the first character on a new line. Thus, line feeds embedded in text are ignored. This procedure has been adopted because the printer does not flush its line buffer before performing a line feed. As a consequence, printable data may be lost or may appear in an unexpected position. It is suggested that the user send carriage returns, not line feeds, to advance the paper.

5.10.3 Form Feed

Form Feed characters are always passed to the printer. It is advised, however, that the user send them only as the first character on a new line because the printer does not flush its line buffer before executing the command. Thus, data may be printed in an unexpected position if a form feed terminates it.

5.10.4 Carriage Return

If a Carriage Return character appears as the first character on a new line, a line feed is sent to the printer. In all other situations the character is passed to the printer, causing the current contents of its line buffer to be printed. A Write ASCII request is terminated by a carriage return.

5.10.5 Extended Typeface

This character instructs the Centronics 306C printer to print the current line in extended typeface. The driver always sends it to the printer. Unless it is sent as the first character on a line, it will displace tabbing positions to the right of its position.

5.10.6 End-of-File

This character terminates a Write Binary or Write ASCII request.

5.11 Format Control Options

The optional printer driver provides a user with several tools to use in determining print format.

5.11.1 Tab Position

The driver uses the tabbing positions in force for the console driver at the time of each call to the driver. Thus, any change to the console tab positions also affects printer output. These positions may be changed in four ways:

1. By using the RIO command SET TABSIZE=n.

2. From the console keyboard, by first positioning the cursor then entering the sequence <control T>T to set a tab position, or <control T><space> to clear one.

3. By sending a Write Status request to the console driver from a user program.

4. By using the RIO command RESTORE_TABS.

Refer to the Z80 RIO Operating System User's Manual for further details on the use of the above mentioned RIO commands.

5.11.2 Print Size

The driver can be instructed to precede each line of output with an "extended typeface" character, which causes the Centronics 306C printer to print in the larger of its two print widths. When selecting this option, the right margin value must be reduced (see Section 5.11.6) because of the reduction in the number of printing positions across the paper.

This option may be altered by sending the driver a Write Status command (Section 5.9.8), or by use of the SET.LP utility (Section 5.12.2).

5.11.3 Abort Operation

When the Abort option is selected, a Write Binary operation can be aborted by entering an escape character from the console keyboard. With the option disabled, the driver does not monitor the escape key.

This option may be altered by sending the driver a Write Status command (Section 5.9.8 and 5.12.1) or by use of the SET.LP utility (Section 5.12.2).

The user is advised to disable this option when using Zilog BASIC to output strings containing embedded carriage return characters. BASIC will abort with an Error 88 (RIO interface error) if a driver returns a status of "program abort" to it.

5.11.4 Treatment of Over-Length Lines

The driver has two options for dealing with lines containing characters that extend beyond the current position of the right margin (see Section 5.11.6): 1) It can truncate the line, so the characters beyond the margin do not print or, 2) the characters can be "wrapped round" onto the next line. This process is repeated as many times as necessary to output the whole line. Printing of the next line commences on a new line.

This option may be altered by sending the driver a Write Status command (Section 5.9.8 and 5.12.1) or by use of the SET.LP utility (Section 5.12.2).

5.11.5 Left Margin Width

The left margin width can be adjusted by instructing the driver to output a specified number of spaces ahead of the first character position on each printing line.

This option may be altered by sending the driver Write Status command (Section 5.9.8 and 5.12.1) or by use of the SET.LP utility (Section 5.12.2). This option can also be used to accommodate different form types under software control, or to allow space for stapling or punching of printer output.

5.11.6 Right Margin Position

This option allows a user to select the number of printing positions across the form. Note that this number includes any positions allocated for the left margin (see Section 5.11.5). Characters beyond the right margin can be truncated or wrapped round according to option (see Section 5.11.4).

This option may be altered by sending the driver a Write Status command (Section 5.9.8 and 5.12.1) or by use of the SET.LP utility (Section 5.12.2).

5.11.7 Page Length

The driver keeps track of the number of lines output since the last Form-Feed character. When this number exceeds a selectable value, a form feed is output, which makes the printer advance the paper to the top of the next page. This automatic form feed feature, which can be disabled, allows convenient printing of data not already formatted into pages. Examples of such data are program source files and directory listings.

This option may be altered by sending the driver a Write Status command (Section 5.9.8 and 5.12.1) or by use of the SET.LP utility (Section 5.12.2).

5.12 Selecting Format Options

Establishing and changing format options is handled either by means of the Read and Write Status commands, or by means of the SET.LP utility.

5.12.1 Read/Write Status Commands

The driver contains a four-byte status buffer that can be examined or altered by way of the Read and Write Status commands, respectively. The format of this buffer is described in Table 5-5.

Table 5-5. Status Buffer Format

Byte	Contents	Default Value
0	Flags:	
	Bit 0: Set for extended typeface, clear for compressed.	Compressed
	Bit 1: Set to prevent console escape from terminating Write Binary request.	Abort enabled
	Bit 2: Set to cause over-length lines to be wrapped round.	Lines truncated
1	Left Margin Width: The number of spaces output preceding first character of any non-blank line.	0
2	Right Margin Position: The total number of print positions (including the right margin). No characters are printed beyond the right margin.	132
3	Page Length: The number of lines to be output following each form feed before an automatic form-feed is forced. A page length of zero inhibits automatic form feed.	63

If the Write Status command is used, the user is responsible for ensuring that the status written is meaningful, because the driver performs no checks. In particular, the left margin width should not be greater than the right margin position.

5.12.2 SET.LP Utility

The SET.LP utility is used to modify driver options from the console or from command files. The format of the command is as follows:

%SET.LP [option]*

That is, the command may be input by itself or may be followed by any number of options. Valid options, which may be input in any order, are described in Table 5-7.

All commands may be truncated to their shortest differentiated length, but must be spelled correctly when input at greater length. The Equal sign (=) is obligatory where shown and must be followed by a valid number.

Options not specified in the command string are not affected by SET.LP; they retain their previously set values.

If no parameters follow the command, as shown below:

%SET.LP

then all printer options are set to their default values. These are defined in Table 5-6.

Table 5-6. Default Option Values

Option	Default Values
Print size	Compressed
Abort feature	Enabled
Over-length lines	Over-length lines truncated
Left margin width	0
Right margin position	132
Page length	63 lines (11 inches)

The SET.LP utility uses logical unit 4 to communicate with the driver. An error exit will result if the driver is not activated, or if invalid options are input.

Table 5-7. SET.LP Options

Option	Effect
CONDENSED	Selects compressed typeface
EXTENDED	Selects extended typeface
ABORT	Allows input of an escape character on the console keyboard to abort Write Binary operation.
NO_ABORT	Prevents abortion of Write Binary operation.
TRUNCATE	Over-length lines are truncated at the right margin.
NO_TRUNCATE	Over-length lines are wrapped round onto the next line.
LEFT_MARGIN=nnn	Selects the (decimal) number of spaces to be output before the first character of any non-blank line. Must be less than RIGHT-MARGIN value.
RIGHT_MARGIN=nnn	Sets the rightmost (decimal) print column. The value includes the left margin width and must be greater than LEFT-MARGIN.
PAGE_LENGTH=xxx	Sets the page length in inches. Valid lengths and the corresponding numbers of print lines are as follows: Length — Lines 0 — Automatic form feed inhibited 5.5 — 30 6 — 33 7 — 39 8 — 45 8.5 — 48 11 — 63 12 — 69 14 — 81

EXAMPLES

The following examples illustrate the use of SET.LP:

%SET.LP

Sets all printer options to their default values.

%SET.LP LEFT_MARGIN=10 RIGHT_MARGIN=80 NO_TRUNCATE

or

%SET.LP L=10 R=80 NO_T

Forces output of ten spaces before each line; if a line contains more than seventy characters, it is "wrapped-round" onto the next print line.

%SET.LP NO_ABORT PAGE_LENGTH=0 EXTENDED

or

%SET.LP NO_A P=0 E

Disables the abort and automatic form feed features; prints in extended typeface.

5.13 Associated Utilities

There are two optional printer driver utilities that perform specific printer functions: the first, LIST, copies a file to the system volume output devices; the second, FF, advances forms by sending a form-feed character to the system volume output device.

5.13.1 LIST Utility

The LIST command copies a file to the system volume output device. Section 5.4 describes the way to define the printer to be this device. The format of the command is as follows:

%LIST <file name>

where <file name> is the name of an ASCII type file.

%LIST OS.INIT

lists the specified file OS.INIT.

5.13.2 FF Utility

FF (Form Feed) is a utility that sends a form-feed character to the system volume output device. Section 5.4 describes the method of defining the printer to be the system volume output device. The format of the command is:

%FF

Logical unit 3 is used for output. No errors are possible. An example of the use of the command is as follows:

FF;X

sends two form feeds to the system volume output device.

5.14 Relinking the Driver

A command file, DRIVER.LINK, is provided to facilitate relinking of the driver. The command shown below,

%DO DRIVER.LINK LP <start address>

relinks the file LP and sets its subtype to 1, indicating that it is a device driver.

SECTION 6

PATCH PROGRAM

6.1 Purpose

The PATCH program allows the user to append a new segment of code to an existing RIO procedure file, or to modify bytes in any kind of file without program relinking.

6.2 Operation

The PATCH program has two modes of operation, PATCH and MODIFY, as described below.

Example:

```
%PATCH
#FILE PATCH 1.0
#SELECT MODE:
MODIFY (M or m) OR PATCH (P or p)
P
```

6.2.1 PATCH Mode

The PATCH program stores a string of bytes at a given address in a program. The length of the string of bytes is user-defined and must be less than or equal to 255 bytes. The PATCH program stores a CALL instruction and as many NOP instructions as are needed to fill out the length supplied by the user. The bytes replaced by this string become the first bytes in the new code segment. In this way, the user may move as many instructions as are needed to the new code segment. PATCH creates a segment that starts at a user supplied address and contains the bytes taken from the original segment and any user supplied bytes. When the user terminates the byte input, PATCH writes the last segment and fixes the descriptor record to include the new segment.

The user input for PATCH mode is as follows:

1. Procedure file name - A RIO procedure-type file.

   ```
   #PROCEDURE FILE NAME
   test_program
   ```

2. CALL address (hex) - The address in the procedure where the call instruction is to be placed.

```
#ADDRESS TO STORE CALL
5A35
```

3. PATCH address (hex) - The low address of the new segment.

```
#ADDRESS FOR PATCH
6E00
```

4. Number of bytes to be transferred to the new segment (decimal).

```
#NUMBER OF BYTES TO MOVE OUT OF OLD SEGMENT
NULL INPUT (CARRIAGE RETURN) SETS DEFAULT (=3)
5
```

5. Code bytes (hex) - The bytes of data or code for the new program segment.

```
#ENTER PATCH BYTES (ONE AT A TIME),
NULL INPUT (CARRIAGE RETURN) TERMINATES INPUT
FD
21
0B
50
CD
03
14
C9
```

6.2.2 MODIFY Mode

In MODIFY mode, PATCH allows the user to replace existing bytes within a file (any type). PATCH prompts the user for the record number to be modified (records index from 1), and the index into the record for the byte being modified (bytes also index from 1). PATCH then asks the user for the new byte value, inserts the new value in the record, and updates the file. This mode of the PATCH program operates in a loop so that multiple bytes may be changed in a single session.

NOTE: The PATCH program replaces the original file with the new one. If the user is uncertain about the changes, a backup copy of the original should be made.

The user input for MODIFY mode is as follows:

1. File name to be modifed - Any RIO file

```
#MODIFY FILE NAME
test_modify
```

2. Record number to be modified (decimal) - The record number (indexed from 1) that contains the byte to be modified.

```
#RECORD NUMBER
1
```

3. Byte position to be modified (decimal) - The position (indexed from 1) in the specified record.

```
#BYTE POSITION
23
```

4. New value for the byte (hex)

```
#NEW VALUE
FF
```

6.3 I/O Unit Utilization

```
Unit 1 - User input
Unit 2 - Console messages
Unit 4 - File I/O
```

SECTION 7

MEMORY TEST

7.1 Introduction

The Memory Test program (MEM) verifies the correct operation of MCZ system memory. It requires a functioning CPU, disk device, and terminal. In addition, at least 3000 (decimal) bytes of error-free memory are required. The actual range of tested memory may be user-selected. Any test sequence may be automatically repeated, and an error log of the eight most recent errors is kept.

7.2 Memory Test Naming Conventions

The memory test may be linked and run anywhere in memory. The test name has three fields:

MEM.<MCZ>.<NN>

where <NN> specifies the first two digits of the address at which it begins. For example, the normal MCZ memory test is called MEM.MCZ.14.

7.3 Linking MEM

To make a new version of MEM, execute the DO file MEM.<MCZ>.LINK, with one parameter that specifies the high order two hex digits of the test's starting address. To make a test to run at the top of a 64K system, enter:

DO MEM.MCZ.LINK F0

7.4 Running MEM

MEM is run from the PROM monitor by entering:

```
>GET MEM.MCZ.14
>R SP
SP <some value> 10F0
>J 1400
```

To stop the program, either press the reset button and reboot RIO, or use the D command (see Section 7.6) to enter the PROM monitor and use the OS command to reboot the system.

To find out whether memory is working, after MEM.MCZ.14 prompts with a pound sign (#), enter:

```
#L ON
#T@
```

This executes all tests for the memory range from 2000 to the size of the system's memory. Successful completion of an overnite test normally indicates a completely error-free memory subsystem.

7.5 Operational Characteristics

MEM sizes the memory and enters the Command mode, which is indicated with the prompt character (#). Commands allow test parameters to be modified and examined, as well as tests to be executed. Commands are typed one per line according to the normal PROM monitor conventions. The RIO line editing characters ctrl-h (backspace) and line delete (Del) can be used to delete one character or a whole line, respectively. Upper and lower case letters may be used interchangeably.

All numbers are printed by the program in hexadecimal. User-typed numbers are assumed to be hexadecimal unless preceded by a period (.); for example, 10 is 16 decimal, but .10 is really 10 decimal.

All commands are one letter followed by parameters which may be optional depending on the specific command. The first parameter may begin immediately after the command letter as in t3, but succeeding parameters must be separated from each other by at least one delimiter (comma or space).

At any time, the user can force the memory test back into the Command mode by pressing the non-maskable interrupt or BREAK key. This is especially useful for stopping memory tests, and the only way to halt a test in the Loop mode is to cause such a break. BREAK does not work from inside the PROM monitor debugger, so a QUIT command must be issued to leave the debugger and to reenter the Command mode.

7.6 Commands

Table 7-1 lists the commands and their syntax, which are described individually in the succeeding paragraphs.

Table 7-1. Memory Test Command Summary

CMD	NAME	DESCRIPTION	SYNTAX
A	ADDR.RNG	Selects area of memory to be tested	#A <new low range> <new high range> #A <new high range>
D	DEBUG	Enter Debugger, Q (QUIT) to Return to MEM	#D
E	ERR LOG	Prints last 8 error messages	#E
I	INITIALIZE	Reset MEM, reset loop, reset error log to 0	#I
L	LOOP MODE	"ON" loops until BREAK key is pressed	#LON
S	STATUS	Prints values of test parameters	#S
T	TEST	There are 6 types of tests (see 7.7)	#T @ (all 6) #T2 3 (addressing tests only)

The following commands are entered as one-character command entries, qualified by one or more required or optional parameters.

7.6.1 Address Range Set Command (A)

This command changes the address range for subsequent tests. The address range determines which part of memory will be tested when actual tests are run (see T command below). It must be followed by one or two numbers. If two numeric parameters are typed, the first is taken as the new low address and the second as the new high range address. If only one number is typed, only the high end of the range is changed. When MEM begins, it sizes memory and assumes all memory above itself is to be tested and therefore sets the test range accordingly. The Initialize (I) command also resets the address range to extend from immediately above the test to the top of memory.

The typed range values are rounded up to the next 8-byte boundary and used as the actual range. This means that typing two range addresses that both round up to the same modulo eight boundary produces a bad range error. Also note that MEM uses 0000 as the highest possible memory address. It is correct to type FFFF as the highest address, but it will be stored and displayed as 0.

Example: #A F000 Useful for preserving the ZDOS map.

#a .9500 .9600 Useful if location of the problem is known.

7.6.2 Debug Command (D)

Upon entering the PROM monitor debugger, the debugger outputs the PROM monitor prompt character (>). The user may exit back to MEM by typing the debugger QUIT (Q) command.

This command is useful for setting and displaying parts of memory which need closer inspection.

```
Example:  #D
          >
```

7.6.3 Error Log Print Command (E)

This command prints the eight most recent error messages on the user terminal. The order of printing is from the most recent message to the eighth most recent message. The error message is identical to the one printed when the error occurred (see Section 7.9). The Initialize (I) command empties the error log.

Example: #E

7.6.4 Initialize Command (I)

This command resets MEM to its beginning state. Memory is sized again, the Loop mode is turned off, the total error count is set to zero, and the error log is emptied.

Example: #I

7.6.5 Loop Mode Change Command (L)

This command requires one parameter, either "ON" (turns Loop mode on), or "OFF" (turns Loop mode off). If Loop mode is ON, any test command is repeated until the user causes a break.

Example: #LON

7.6.6 Status Command (S)

This command prints out the current values of all the test parameters which are reset by the Initialize command. These values include: low and high values of the test range, whether Loop mode is on or off, the total error count, and the total number of completed passes. The error count is the total number of errors since MEM was started or an I command was typed.

Example: #S

7.6.7 Test Command (T)

Use this command to execute actual memory tests. The "T" is followed by one or more test numbers. The parameter list is scanned from left to right and each test is executed in turn. There are currently six tests numbered 1 through 6 (see Section 7.7 below). The "at" sign (@) character may be used as an abbreviation for all tests. If Loop mode is on, the specified tests are repeated in their given order until the user stops them with a break.

Example:	#T @	Execute all tests
	#T1 2 3 4,5,6	Same as t @
	#T2 3	Run addressing tests only
	#T 6 5 4 3 2 1	Run all tests in reverse order

7.7 Test Descriptions

There are six memory tests, each with a specific set of attributes and tasks, as described below.

7.7.1 Test 1: Fixed Pattern Test

This test writes and verifies the following 20 preset memory patterns within the current address range:

FF, 00, AA, 55, 01, 02, 04, 08, 10, 20, 40, 80
FE, FD, FB, F7, EF, DF, BF, 7F

7.7.2 Test 2: Memory Address Test

This test writes into each two-byte location its own address. The addresses are then read and verified in ascending order. The test is first performed for even addresses and then repeated for odd addresses.

7.7.3 Test 3: Reverse Memory Address Test

This test writes into each two-byte location the complement of its address. The addresses are then read and verified in descending order. The test is performed for even addresses and then repeated for odd addresses.

7.7.4 Test 4: Walking (Marching Pattern Test)

This test writes a background pattern of alternating 00 and FF throughout the test range. Then for each cell (bit location) the following operations are performed:

1. The cell is read and verified.
2. The contents of the cell are complemented.
3. The cell is read and the complement is checked.
4. The cell is put back to its original pattern.

The background pattern is reversed and the test is then repeated.

7.7.5 Test 5: 1-0 Block Pattern Test

This test writes and verifies four patterns consisting of alternating blocks of FF followed by 00. The block lengths are 4, 32, 64, and 128 bytes.

7.7.6 Test 6: Random Data Test

The low 1024 bytes of the test range are filled with random data. That area is then compared to itself. Next the random data block is moved to the next 1024-byte block. That block is then compared to itself and to the original random data block. The process is repeated for each 1024-byte block until reaching the high address of the range. The last block may be smaller than 1024 bytes. If the entire range is less than 1024 bytes, the range is filled with random data and compared to itself.

7.8 Memory Compare Error Messages

All memory error messages have the following standard format:

```
ERROR - TEST tt
                VALUE IS  bbbbbbbb   ADDRESS IS hhhh
                SHOULD BE  bbbbbbbb
```

where "tt" is the test number, "bbbbbbbb" is some 8-bit binary value, and "hhhh" is some hexadecimal address. For the address tests (2 and 3), "bbbbbbbb" is some 16-bit binary number. If any 16-bit binary number contains a zero leading byte, that byte is omitted.

MEM contains a mechanism to suppress multiple error messages that indicate the same error within a given chip. For a given test, at most two error messages are printed per 1000 hexadecimal block. If a test involves multiple passes or patterns, two error messages are printed per block per pass. The error log also only records errors whose messages are actually printed, but the total error count includes every error, whether or not the error message was actually printed.

Note that the leftmost bit is bit 7 (or 15) and the rightmost bit is bit 0, which is the reverse of the actual chip layout on Zilog boards. Therefore, the leftmost chip in a row actually corresponds to the rightmost bit in an error message.

7.9 User Error Messages

The following error messages can occur from user errors in typing commands or values:

Message	Meaning
BAD COMMAND	Command not recognized.
BAD PARAMETER	Required parameter missing or wrong parameter type. For example, a number was expected or an unknown test number was given.
BAD TEST RANGE	Invalid test range. In a valid test range, the low address must be a least 8 bytes less than the upper range after rounding. Both range addresses are first rounded up to the next 8-byte boundary, and the range check is performed.

7.10 User Notes

Certain user actions will cause MEM to die:

1. If registers or memory values used by MEM are changed from inside the debugger (entered with the D command).

2. If MEM is included in the current test range, MEM will be overwritten.

In the above cases, reload MEM and restart it.

Bad memory can cause the memory sizing routine to fail, in which case the user must set the proper range with the A command.

INDEX

Zilog, Inc. 10460 Bubb Road, Cupertino, California 95014

Telephone (408)446-4666 TWX 910-338-7621

Printed in USA