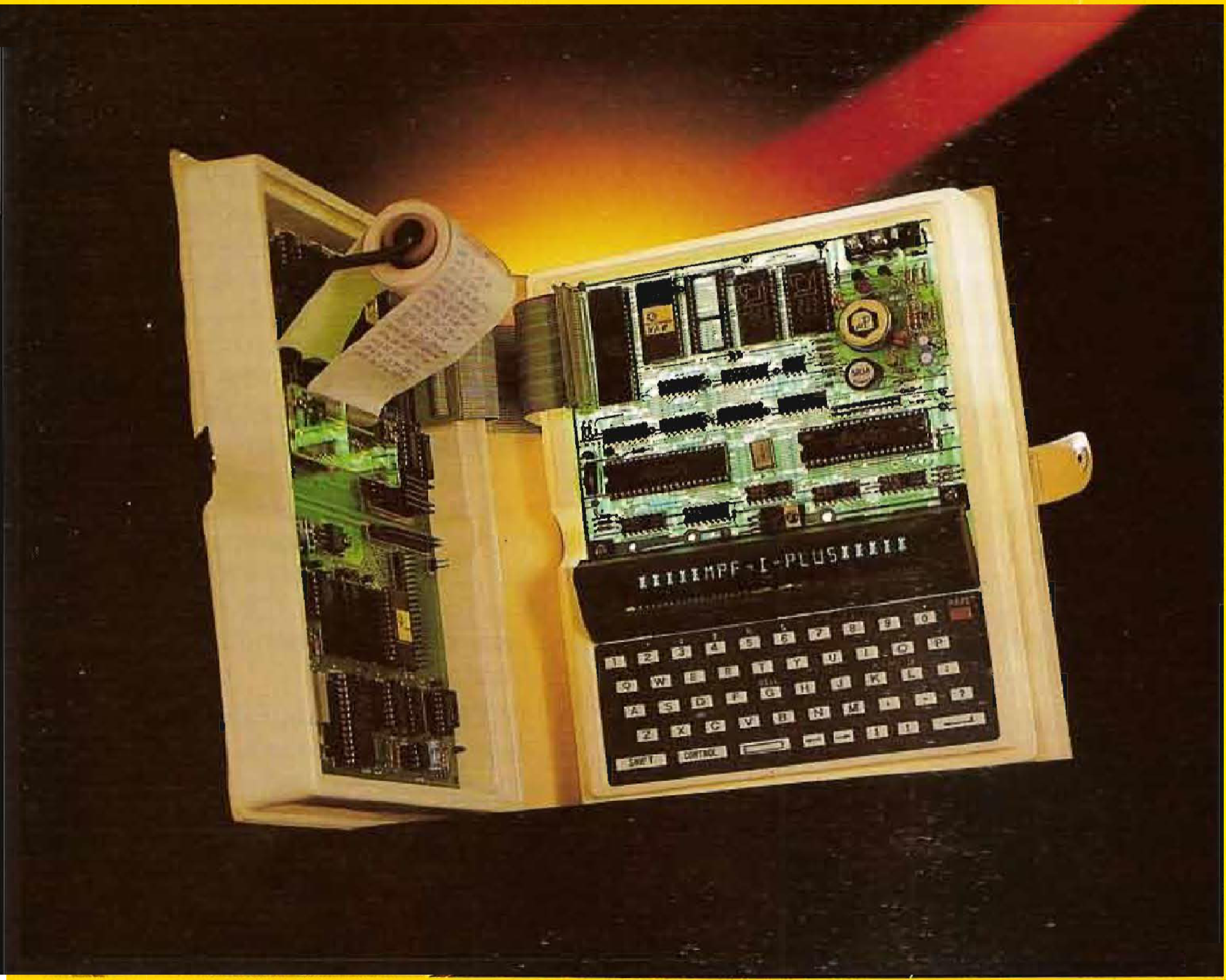
MPF-IP

USER'S MANUAL



CHR. M.T.S. MARINUS A. POTTER
Postbus 40413
2504 LK 's-GRAVENHAGE

MPF-IP
USER'S MANUAL

COPYRIGHT

Copyright © 1983 by MULTITECH INDUSTRIAL CORP. All rights reserved. No part of this publication may be reproduced, transmitted, transcribed, stored in a retrieval system, or translated into any language or computer language, in any form or by any means, electronic, mechanical, magnetic, optical, chemical, manual or otherwise, without the prior written permission of MULTITECH INDUSTRIAL CORP.

DISCLAIMER

MULTITECH INDUSTRIAL CORP. makes no representations or warranties, either express or implied, with respect to the contents hereof and specifically disclaims any warranties or merchantability or fitness for any particular purpose. MULTITECH INDUSTRIAL CORP. software described in this manual is sold or licensed "as is". Should the programs prove defective following their purchase, the buyer (and not MULTITECH INDUSTRIAL CORP., its distributor, or its dealer) assumes the entire cost of all necessary servicing, repair, and any incidental or consequential damages resulting from any defect in the software. Further, MULTITECH INDUSTRIAL CORP. reserves the right to revise this publication and to make changes from time to time in the content hereof without obligation of MULTITECH INDUSTRIAL CORP. to notify any person of such revision or changes.

Multitech
INDUSTRIAL CORP.

OFFICE/
9FL. 266 SUNG CHIANG ROAD. TAIPEI 104.
TAIWAN. R.O.C
TEL: (02)551-1101
TELEX 19162 MULTIIC FAX. (02)542-2805
FACTORY/
1 INDUSTRYE. ROAD. III.
HSINCHU SCIENCE-BASED INDUSTRIAL PARK.
HSINCHU. TAIWAN 300. R.O.C.

TABLE of CONTENTS

Appendices

Chapter 1 Overview and Installation

1.1 Introduction

The Micro-Professor I Plus (MPF-IP) is a low-cost, versatile microcomputer system featuring sophisticated software and hardware capabilities. It is not only ideal for those who intend to familiarize themselves with micro-processing and advanced microcomputer hardware and software, but also can be used for many dedicated purposes and OEM applications such as industrial control and instrumentation.

Good design techniques and the use of a Z-8Ø central processing unit (CPU) results in a high performance unit.

The Z8Ø microprocessor features a powerful instruction set, which has 158 instructions. The Z8Ø operates at 2.5 MHz and processes 8 bits of data at a time. Thus, Z8Ø is one of the most commonly used microprocessors with wide-ranging applications.

The MPF-IP uses a display panel that can display 2Ø characters using 16-segment font. All the 64 standard ASCII characters can be displayed. The display length corresponds with the 2Ø-column printer.

Printing at 48 lines per minute, the printer provides the means to permanently record the commands, data, programs, status, and other messages. Each character printed by the printer is in a 5 by 7 dot matrix.

The keyboard has 49 keys.

The operation of MPF-IP is controlled by an 8K monitor program which resids in the read only memory (ROM). The monitor, aided by 4K random access memory (RAM), enables the user to enter a comprehensive set of single keystroke commands, which make it easier for the user to use the CPU, memory, and I/O devices. Thus, the user can concentrate on microprocessor software development and application design.

1.2 An Overview of MPF-IP Specifications

1) CPU: Z80
2) ROM: 8K
3) RAM: 4K
   (Refer to the system memory map illustrated in section 8.1)
4) Contains a text editor
5) The MPF-IP can execute programs written in assembly language, because its 8K ROM contains a two-pass assembler, line assembler, and a disassembler.
6) Battery backup.
7) A 20-character display that can display a full 64 ASCII character set.
8) A 49-key standard typewriter keyboard.
9) 8K BASIC Interpreter provided as an option.

Options for the MPF-IP also include:

* PRT-MPF-IP: a thermal printer.
* EPB-MPF-IP: an EPROM programmer board.
* SSB-MPF-IP: a speech synthesizer board.
* SGB-MPF-IP: a sound generation board.
* IOM-MPF-IP: an Input/Output and Memory Board.
* EPB-MPF-IBP: an EPROM programmer board which can used on MPF-IP or MPF-1 at random.
* FORTH-MPF-IP: an 8K EPROM which enables a user to program in FORTH language.
* BASIC-MPF-IP: an 8K EPROM which enables a user to program in BASIC language.

Note:

In order to make it easier for the user to learn the operation of MPF-IP, a comprehensive, yet easy-to-read, student work book is available which provides effective, explanation-exercise-answer formats on all key operations, applications and functions.

1.3 Installation Procedure

1) If the MPF-IP is to be used with the PRT-MPF-IP, connect PRT-MPF-IP to the MPF-IP first with a flat cable connector. (For details, please refer to PRT-MPF-IP Printer Operation Manual.)

2) Insert the thermal paper into the printer as illustrated in PRT-MPF Printer Operation Manual, II Installation Procedure. Note that the finer surface of the thermal paper should face up, because that side of the paper is specially treated so that dot matrix characters can be formed by the heat produced by the thermal head of the printer.

3) Connect AC power adaptor (9V/1A) to the PRT-MPF-IP.

4) Connect AC power adaptor (9V/6ØØmA) to the MPF-IP.

5) When the display shows

   ****MPF-I-PLUS****

   and the printer prints out the same, the MPF-IP is ready to run.

Chapter 2
MPF-IP Specifications

2.1 MPF-IP Hardware Specification

2.1.1 Central Processing Unit

The Zilog Z-8Ø CPU has a powerful instruction set, comprising of 158 instructions. It can operate at a maximum speed of 2.5 MHz. However, MPF-IP operates at 1.79 MHz.

2.1.2 ROM

The ROM of the MPF-IP is a single +5V EPROM 2764 that can store up to 8K bytes of data. The monitor EPROM address is from ØØØØ to 1FFF.

2.1.3 RAM

The MPF-IP has two static RAMs, thus the total capacity of RAM is 4K bytes. The user can jump at board location U4 so that an I2732 or TMS2532 can be used at location U4.

The address of the RAMs ranges from FØØØ to FFFF (The locations from FØØØ to F7FF is assigned for the chip at board location U4.).

2.1.4 Memory Expansion Area

The board location U3 is reserved for a single +5V EPROM 2764 x 1 or 2732 x 1. The addresses reserved for this location are from 2ØØØ to 3FFF.

2.1.5 Input/Output Port

The I/O port of the MPF-IP consists of two programmable 8255 chips, which have 48 parallel I/O lines.

I/O addresses: 8Ø ~ 83 (at board location U14)
9Ø ~ 93 (at board location U13)

2.1.6 Display

The display of the MPF-IP is a fluorescent indicator panel that can display 2Ø 16-segment font characters.

2.1.7 Keyboard

The MPF-IP has 49 keys, including alphanumeric keys (from A to Z, and Ø to 9) and function keys.

2.1.8 Speaker

A 2.25 inch speaker is built on the MPF-IP main board.

2.1.9 Audio Tape Interface

The MPF-IP can be connected to any cassette tape recorders. The speed of data transfer is 165 bits per second (bps).

2.1.10 System Clock Rate

The crystal oscillator of the MPF-IP oscillates at the frequency of 3.5795 MHz. Between the crystal circuit and the CPU is an IC, namely, 74LS14, which divides the clock frequency by 2. Thus, the system clock rate is 1.79 MHz. The cycle time is Ø.56 microseconds.

2.1.11 System Power Consumption

Single +5V power supply, current consumption is 45ØmA.

2.1.12 Main Power Input

The main power input to the MPF-IP is DC 9V/6ØØmA.

2.1.13 Physical Characteristics

6-1/6" X 8-2/3" X 3/8"

2.2 MPF-IP Software Specifications

(The major functions of the monitor program)

Immediately after power-up of the MPF-IP, the monitor program is executed immediately. The monitor program resides in the 8K ROM. It performs the following tasks:

2.2.1

Initializes a reset cycle:
Initializes the MPF-IP so that it is ready to execute user programs.

2.2.2

Keyboard scanning:
Scans the keyboard for any key press and responds accordingly.

2.2.3

Scans the display buffer and can display any character in the MPF-IP ASCII character set, which contains 64 characters.

2.2.4

Stores and retrieves data through audio tape interface at the speed of 165 bits per second (bps). Each time the monitor reads from or write to tape, a checksum will be produced by the monitor and will be matched with the checksum on tape. Filenames can be given to data stored on tape for easy access.

2.2.5

Displays and alters the data stored in memory or registers. Commands used for performing these tasks include DISPLAY, CHANGE, FILL, MOVE, INSERT, DELETE, NEXT, and LAST.

2.2.6

Sets or clears the breakpoint in a program.

2.2.7

Program debugging can be achieved by setting breakpoint or executing a program in STEP mode. One breakpoint is allowed in a program. A programmer can examine the contents of registers or memory locations if a breakpoint is set in a program. A programmer can also look into the contents of certain memory locations or registers each time an instruction is executed, if the program is executed in STEP mode.

2.2.8

Calculates the relative addresses to be used by the JR or DJNZ instructions.

2.2.9 Editor

Provides a text editor. It enables a user to input, change, or list source programs, data, or general text conveniently.

2.2.10 Line Assembler (One Pass Assembler)

Provides a line assembler (one pass assembler), which only converts one line of assembly language program into machine code at a time and does not process pseudo instructions such as ORG, EQU, LABEL, DEFB, DEFW, DEFS, DEFM, and comments. It uses less memory than two pass assembler does, but it can only process absolute values.

2.2.11 Two Pass Assembler

Provides a two pass assembler, which can convert source programs into machine codes and process pseudo instructions. It has the functions of a linker, and can print program listings when using together with a printer.

2.2.12 Disassembler

The disassembler can convert machine codes back into the form of assembly program.

Note: In addition to that specified otherwise, all the addresses used in this book are expressed in hexadecimal.

Chapter 3
System Description

3.1 The Functions of the Monitor

1) Stores the program into the RAM. Change or examine the data in the RAM.

2) Executes the program stored in RAM.

3) Executes the program in STEP mode or sets breakpoint in a program. Executing the program in STEP mode is very helpful for learning and debugging purposes.

4) Other functions include audio tape interface, relative address calculation, and text editing.

5) The user can develop a dedicated computer system based on the MPF-IP. The MPF-IP is very flexible for both software and hardware development.

3.2 Battery Backup

The MPF-IP features a battery backup so that data will not disappear even after power is turn off.

On the left of the MPF-IP main board, there is a switch. When the switch is on, the power of the battery backup is not supplied to the MPF-IP. When the system power supply from the adaptor is cut out suddenly (because of a power failure or the adaptor is disconnected), power will be supplied automatically from the battery to the RAM of MPF-IP and CD4556BE. Thus, data stored in the RAM will be preserved.

To test the battery backup, the user can disconnect the adaptor and then re-connect it to see if data in RAM is lost.

If you don't intend to use the battery backup, turn off the switch. The batteries are to be installed on the back of the PC board.

3.2.1 RAMs

If the RAM of the MPF-IP consists of two CMOS HM6116 (4K bytes), the battery backup--which includes four UM3 batteries--can preserve the data in RAM for about a year.

If NMOS chips such as TMM2Ø16P-2 or M58725P-15 are used as the RAM of the MPF-IP, the battery backup can preserve the data in RAM for only five hours.

If the TC5516APL is used as the RAM of the MPF-IP, data in RAM can be preserved for several years. However, to use TC5516APL; refer to Chapter 8 for the correct wire cutting and jumping at J2.

3.2.2 Address Decoder

The [illegible] 74LS139 [illegible] CD4556BE) is used as the address decoder.

3.2.3

If 2732 or 2532 is installed at board location U4 as RAM, the power from battery backup will be consumed much quicker.

3.3 Keyboard Familiarization

The MPF-IP can generates 64 ASCII characters. They include alphanumerical letters (from A to Z, and 0 to 9), space, special signs, etc. To enter any of these characters, press the key marked accordingly.

The SHIFT key, which is located at the lower left corner of the keyboard, is used to generate the characters which are marked above the keyboard keys.

3.3.1 The Monitor Commands

The CONTROL key is used to enter major monitor commands. The monitor commands are entered by typing the control characters while holding down the CONTROL key. They are listed as follows:

CONTROL A (Assembler)
CONTROL B (BASIC)
CONTROL C (Re-enter BASIC)
CONTROL D (Disassembler)
CONTROL E (Editor)
CONTROL L (Line Assembler)
CONTROL R (Re-enter Editor)
CONTROL P (Printer Control)
CONTROL Q (Software Escape)
CONTROL G (Beep Control)

The monitor commands and their functions are explained in detail in Chapter 4. Only CONTROL P and Q will be discussed here. Because the printer of the MPF-IP PRT-MPF-IP) only prints on thermal paper, CONTROL P is used as an on/off (toggle) switch. When a user thinks there is no need for printing paper copies, he can turn off the printer with the CONTROL P command. For example, when the assembler is converting a source program into machine code, the user can turn off the printer to save thermal paper. If there are errors in the source program, the user can use CONTROL R to re-edit the source program. After the source program has been modified and the assembly completed, the user can turn on the printer to print hard copies.

CONTROL Q stands for software escape. After pressing CONTROL Q, the monitor regain control without affecting any parameters in the RAM.

3.3.2 The TAB Key

The TAB key can be used efficiently by a programmer to type in assembly programs. The $\boxed{\rightarrow}$ key on the MPF-IP keyboard is used as the tab key. Pressing this key once causes the cursor to move six spaces to the right on the display. The key code of this key can be found on the table of MPF-IP ASCII Code. The use of this key enables a programmer to save RAM space when entering a program.

3.3.3 Input Line Buffer

The input line buffer accepts input line of up to 4Ø characters. Therefore, each time a programmer type in an input line, the length of the input line should not exceed this limit. Because the length of the display is 2Ø characters, the display will shift right to display the characters typed after the 2Øth character of an input line.

3.4 PRT-MPF-IP

The printer of the MPF-IP is discussed in detail in PRT-MPF-IP manual. Please refer to that manual for detailed operation of the printer. The use of disassembler and memory dump usually synchronizes with the operation of the printer. If no printer is connected to the MPF-IP, the functions of disassembler and memory dump can not be performed.

3.5 Addresses Related with System Expansion

To determine whether peripherals are interfaced to the MPF-IP, the MPF-IP examines the values of certain memory locations -- 6ØØØ, 2ØØØ, and AØØØ. If the values of these memory locations are the same with their preset values, then the MPF-IP is connected with peripherals. The memory range form 6ØØØ to 6FFF is used by the PRT-MPF-IP, and that for TVB (TV Interface, Board) is from AØØØ to A7FF, and 8K BASIC Interface, from 2ØØØ to 3FFF. The MPF-IP checks the values of memory locations 2ØØØ, 6ØØØ, and AØØØ to see whether these external devices are interfaced to the MPF-IP. If the values of these locations are FF, then the MPF-IP is not connected with external devices. If the monitor program returns the values that are identical with the preset values of location 2ØØØ, 6ØØØ, and AØØØ, then the MPF-IP is connected with external devices.

It should be noted that when a programmer accesses these locations, the programmer should take into consideration the use of these addresses.

3.6 LED Lamp

There are two LED (light emitting diode) on the upper right part of the MPF-IP main board. The functions of them are as follows:

Green LED: When the monitor program scans the keyboard and detects a key press, the green LED lamp will illuminate. The speaker will generate a "beep" sound at the same time.

Red LED: Once the CPU executes the HALT instruction, the red LED lamp will illuminate.

3.7 When the Monitor doesn't Respond

When the monitor doesn't respond to a command line after the carriage return $\boxed{\hookleftarrow}$ key is pressed (This is usually resulted from incorrect format of the command line), use the back space key ← to revise the format of the command line or re-type a correct command line after typing $\boxed{\text{CONTROL}}$ $\boxed{\text{Q}}$. For example, if you intend to look at the contents of memory locations from F8ØØ to F8Ø3. You should press

```
     <M>=F8ØØ
```
$\boxed{\hookleftarrow}$

Instead of typing in F8ØØ, you typed incorrectly F8ØP. Because the letter "P" is not a hexadecimal character, MPF-IP does not respond to this command line. You can use the back space key ← to backspace to P and type in a Ø and then a $\boxed{\hookleftarrow}$ to re-enter the command line.

3.8 Software Break—The Instruction: RST 30H

The instruction RST 30H causes a software break to the program being executed. A programmer can place this instruction at locations wherever he intends to examine the results after certain instructions have been executed, e.g., the contents of certain memory locations and registers. Multiple software breakpoints can be set within a program with the use of the RST 30H instruction for program debugging. After the CPU executed an RST 30H instruction, control will be returned to the monitor. Pressing the G key and ↵ causes the CPU of the MPF-IP to continue to execute the instructions following the software break.

The instruction RST 38H also causes a software break. For details of the use of the RST 38H for software break, refer to the MPF-IP Monitor Program Listing.

The RST 30H has the same effect as the hardware break achieved by pressing the B key. After a program is entered into the RAM, pressing B will cause the display to show

<B>

which prompts a programmer to enter a breakpoint in a program. When the CPU proceeds to the breakpoint as the program is being executed, the monitor will gain control. Only one breakpoint can be set using the B key.

3.9 Number Systems

3.9.1

Hexadecimal numbers are frequently used with microcomputers. The following table (Table 3-1) shows the hexadecimal, binary, and decimal numbers.

Hexadecimal	Decimal	Binary
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111

3.9.2

Whenever a programmer is prompted to set default values in the following functions--Editor, Line Assembler, Two Pass Assembler, and Disassembler, the values to be input should be in hexadecimal. Leading zero and trailing H (which stands for hexadecimal) may be omitted.

3.9.3

When the MPF-IP is in Editor and Line Assembler modes, hexadecimal values are identified by a trailing H. For example, "1Ø" represents a decimal 1Ø, while "1ØH" stands for a hexadecimal 1Ø--which equals to 16 in decimal. For hexadecimal values preceded by the letters from A through F, leading zero should be placed so that they will not be mistaken as symbols or labels. An example is listed as follows:

```
DISPBF     EQU  0FF2CH
           ORG  0F000H
           DEFW 0FF65H
           DEFB 0BDH
           LD   HL,0F560H
           LD   A,0F8H
```

3.9.4

When the MPF-IP is in other modes than the Editor and Line Assembler modes, all values to be input should be in hexadecimal. But the trailing H is not required.

3.10 Audio Tape Interface

If operating in the Editor mode, a programmer can store the program or data on tape with the "W" (write) command; or read back program or data from tape with the "L" command and return to the monitor. If the user intends to re-enter the editor mode after loading a tape to the MPF-IP, the monitor control command "CONTROL R" is used. If a user uses the "R" (read) command in the Editor to read data or program from tape, the MPF-IP will remain in the Editor after reading.

3.11 CONTROL Q or Q

The Q command is used to re-enter the monitor. Pressing the Q key will re-enter the monitor when the display shows the following:

* ERRORS
* SYS-SP
* ERR-SP
* The contents of registers or memory locations

3.12 CONTROL P and CONTROL G

CONTROL P and CONTROL G (they are usually shortened to ↑ P and ↑ G.) only function under the following conditions:

* When the display shows ****MPF-I-PLUS****
* When the MPF-IP is under condition 3.12
* When the display shows /_/\

Otherwise, pressing the CONTROL P or CONTROL G will cause the display to show meaningless characters.

Chapter 4
Operating MPF-IP

This chapter will discuss the basic operations of the MPF-IP. At the end of this chapter, the reader will have a basic understanding 1) how to operate the MPF-IP, 2) program debugging, 3) support functions. Readers are suggested to learn how to operate the MPF-IP by following this chapter closely.

4.1 The Major Monitor Commands

The major monitor commands of the MPF-IP are listed in the following table (Table 4-1):

Category	Command	Function
* Major Function Entry	RESET	Enter and initialize the monitor
	Q	Re-enter the monitor
	E	Enter and initialize the text editor
	R	Re-enter the text editor
	A	Enter two pass assembler
	L	Enter one pass assembler
	D	Enter disassembler
	B	Enter the BASIC language
	C	Re-enter BASIC
Fill in Data	F	Store data in the RAM buffer
Jump Relative	J	Calculate the relative address
Insert Data	I	Insert the contents of a memory block into the RAM
Delete Data	D	Dalete one byte of data from the memory
Execution	G	Execute a program which starts from a specified address
Step	S	Single-step a program (Execute a program instruction by instruction.)
	↑ ↓	Used to set the register pairs whose contents are to be examined

Display/Alter Registers	R $\boxed{\downarrow}$ $\boxed{\uparrow}$:	Display the contents of registers Display the contents of the next pairs of registers Display the contents of the register pairs that precedes the registers currently displayed Change the contents of registers
Display/Alter Memory	M $\boxed{\downarrow}$ $\boxed{\uparrow}$: /	Display the contents of specified memory locations Display the contents of the next four bytes Display the contents of the four bytes that precede the current displayed location Alter the contents of specified memory Move the contents of a memory block to another location
Manipulate Breakpoint	B	Set or clear breakpoint
Load/Dump Memory	L W	Load data from tape to memory Write data from memory to tape

* Note: Any of the major functions are entered by typing the related control character while holding down the CONTROL key.

4.2 Major Function Entry and Exit

Seven commands are provided to enter major functions. Four of these commands allow initial entry and re-entry into the editor and BASIC. All the seven major functions are entered by pressing the related control characters while holding down the CONTROL key.

4.2.1 E Command—Enter and Initialize the Editor

The editor is initialized by pressing the E key while holding down the CONTROL key. For details, refer to Chapter 6 The Text Editor.

4.2.2 B Command—Enter and Initialize the BASIC

Pressing the B key while holding down the CONTROL key will enter and initialize BASIC, if the board location U3 is installed with a BASIC Interpreter. However, if the board location U3 is not installed with a BASIC Interpreter, pressing the B key while holding down the CONTROL key will return to the monitor.

4.2.3 R Command—Re-enter the Text Editor

Pressing the R key while holding down the CONTROL key will re-enter the text editor.

4.2.4 C Command—Re-enter BASIC

Pressing the C key while holding down the CONTROL key will re-enter BASIC without changing the data in memory.

4.2.5 L Command—Enter the One Pass Line Assembler

Pressing the L key while holding down the CONTROL key will enter the one pass line assembler, which will convert the mnemonic opcode (entered from the keyboard) to object code and store the resultant object code in memory. When a line assembler is in use, the user can specify the RAM area for storing the mnemonic opcode and resultant object code, respectively.

4.2.6 A Command—Enter and Initialize the Two Pase Assembler

Pressing the A key while holding down the CONTROL key will enter the two pass line assembler, which can convert the source program into executable machine code. It can also process pseudo instructions.

4.2.7 D Command—Enter and Initialize the Disassembler

Pressing the D key while holding down the CONTROL key will enter the disassembler, which convert machine code into Z-8Ø mnemonic code. It disassembles the contents of memory from a specified memory location until an opcode is found. Then it will disassemble the contents of the bytes that follow the locations where the disassembled opcode is stored. In case an incorrect opcode appears, the display will show a question mark.

For details of the L, A, and D commands, please refer to Chapter 7 the Assembler.

4.3 Basic Operations

4.3.1 System Initialization-The RESET Key

When the RESET key is pressed, a RESET signal will be generated and the MPF-IP will start a reset cycle. Normally the reset signal is automatically generated after power-up. The MPF-IP is to be initialized to its reset state by the reset signal. The monitor control variables are set and the CPU is ready to accept monitor commands. Finally, the display will show

*****MPF-I-PLUS*****

Note when a "cold reset" (the reset initialized by power-up) is initialized, the 20 characters which are to be displayed--*****MPF-I-PLUS*****--appear one by one on the display. In the case of a "warm reset" (the reset initialized by the pressing of the RESET key), the 20 characters are displayed simultaneously.

When the RESET key is pressed, the operation of the CPU is interrupted and control is returned to the monitor which will initialize the CPU. The monitor will examine whether a cold reset or a warm reset is to be performed. The cold reset or power-on initialization will be performed if the monitor determines that power has been interrupted. A cold reset causes the monitor control parameters and user alterable parameters to be initialized. A warm reset only initializes the monitor control parameters and leaves the user alterable parameters unchanged.

The warm reset should be performed any time the CPU has performed unknown operations or the CPU appears lost while executing a command or an instruction. The monitor control parameters can be easily changed when an unvalidated user program is executed. This will cause the MPF-IP function improperly. Performing a warm reset allows the control to be returned to the monitor.

4.3.2 Printer Control—CONTROL P

The printer control command is entered by pressing the P key while holding down the CONTROL key. The default state of the printer is on, e.g., after the power to the MPF-IP is turned on, the printer is automatically turned on. Pressing CONTROL P will turn off the printer. Sometimes a user may not want all the displayed data to be printed. Instead, a user may only wish to print the necessary data. CONTROL P allows the user to use the printer at will. CONTROL P is an off/on (toggle) switch which selects only two states--either on or off.

4.3.3 Software Escape—CONTROL Q

This command is entered by pressing the Q key while holding down the CONTROL key. Whenever the monitor loses control of a program or the MPF-IP, pressing CONTROL Q will return control to the monitor without changing the preset parameters. Though pressing the RESET key will also return control to the monitor, some variables preset by the program might be damaged.

4.3.4 Bell Control—Control G

This command is entered by pressing the G key while holding down the CONTROL key. CONTROL G is also a toggle switch. The default state of this switch is on, e.g., after power is applied to the MPF-IP, CONTROL G is on. When this switch is on, the MPF-IP will sound a beep each time a key is pressed. When this switch is off, the beep sound will be suppressed each time a key is pressed.

4.4 Support Functions

4.4.1 Display/Alter the Contents of Memory

The M command displays the hexadecimal contents of four consecutive memory locations. The use of the M command is listed as follows:

1. Display the contents of four memory locations starting from the specified address. The format of the command is

   M <starting address> ⏎

   The command line is entered following the following steps:

   a. Type M, the MPF-IP will respond with

   ```
   <M>= ∧
   ```

   b. Enter the starting address of a memory range whose contents are to be displayed.

   c. Press the carriage return key ⏎. The MPF-IP will display the contents of four consecutive memory locations immediately after the ⏎ key is pressed.

   ```
   <M>=0010 3E FF D3 92
   ```

2. The ↑ and ↓ Key:

   These two keys are used in conjunction with the M command. While the MPF-IP is displaying the contents of four consecutive memory locations after a user typed in M, <starting address>, and ⏎, pressing the ↓ key causes the MPF-IP display the contents of the next four memory locations.

   ```
   <M>=0014 D3 80 D3 81
   ```

   Pressing the ↑ key causes the MPF-IP display the contents of the four consecutive memory locations that precede the currently displayed locations.

   If no starting address is given after the M command, e.g., the ⏎ is pressed immediately after the M command, the MPF-IP will display contents from location ØØØØ.

3. The ⊡ Key
--The Key Used to Perform a Memory Dump

The ⊡ key when used in conjunction with the M command allows a memory dump to be performed. The format of the memory dump command line is:

M <starting address> ⊡ <ending address> ↵

A printer must be used to perform a memory dump. If the MPF-IP is not connected with a printer or the printer is off, the MPF-IP will return control to the monitor after the command line is entered.

```
<M>=0.10

0000 01 00 03 ED
0004 A9 EA 03 00
0008 3E 88 D3 83
000C 3E 81 D3 93
0010 3E
```

A user may use the memory dump function to set linking address. The format of command is:

M <starting address> ⊡ <ending address> space bar ↵

```
<M>=0.10 6000

6000 01 00 03 ED
6004 A9 EA 03 00
6008 3E 88 D3 83
600C 3E 81 D3 93
6010 3E
```

Note:

If the system does not respond after a memory command has been entered, please check if your printer is " on ".

4. The ⁝ Key
--The Key to Alter the Contents of Memory

The command format is:

M <starting address> ⁝ <data1 data2 dataN> ↵

```
<M>=F800:0 1 22 33
```

After executing the above command, use the M command again to examine the contents the four bytes starting from location F8Ø0

```
<M>=F800 00 01 22 33
```

Note: The MPF-IP accepts ınput lıne of 4Ø characters or less. Any input line should conform to this rule.

5. The ⧄ Key
   --The Key to Move the Contents of a Memory Range

   The command format is:

M <starting address> ⧄ <ending address> space bar <destination address> ⏎

```
<M>=0.10

0000 01 00 03 ED
0004 A9 EA 03 00
0008 3E 88 D3 83
000C 3E 81 D3 93
0010 3E


<F>=F800 F810 FF
<M>=F800.F810

F800 FF FF FF FF
F804 FF FF FF FF
F808 FF FF FF FF
F80C FF FF FF FF
F810 FF

<M>=0/10 F800
<M>=F800.F810

F800 01 00 03 ED
F804 A9 EA 03 00
F808 3E 88 D3 83
F80C 3E 81 D3 93
F810 3E
```

Note: When moving the contents of a memory range to another location; be careful not to damage the data stored in system RAM. If the data in system RAM is damaged, the monitor will function improperly.

If the input parameters are incorrect, the error message "ERRORS" will appear on the display. Pressing Q will return control to the monitor.

4.4.2 The F Command

The command is applied to fill data into a memory range. The format of the command is:

F <starting address> space bar <ending address>
space bar <data> ↵

```
<M>=0/10 F800
<M>=F800.F810

F800 01 00 03 ED
F804 A9 EA 03 00
F808 3E 88 D3 83
F80C 3E 81 D3 93
F810 3E


<F>=F800 F810 34
<M>=F800.F810

F800 34 34 34 34
F804 34 34 34 34
F808 34 34 34 34
F80C 34 34 34 34
F810 34
```

The above example shows that the contents in the bytes from F8Ø0 to F81Ø are scrambled. After the F command that fills 34 into the same memory range, all the locations from F8ØØ to F81Ø are stored with 34. Note if the starting address of the command line is not in RAM, the display will show "ERRORS", which will disappear after pressing Q or CONTROL Q.

4.4.3 Display/Alter the Contents of Registers

The R command is provided to display or alter the contents of registers. The format of the command is:

R <register> ↵

If a 16-bit register is entered in an R command line, the MPF-IP will display the contents of that register only.

If a 8-bit register is entered in an R command line, the MPF-IP will display the contents of that register and the contents of the register which is normally paired with the specified register.

```
<R>= AF FF13
<R>= HL FF35
<R>= AF A0FD
<R>= IX FF00
<R>= SP FEA0
```

1. To display the contents of register A, type R A ↵.
2. To display the contents of register pairs such as BC, DE, HL, B'C', D'E', H'L', just type in either one of the register that is contained in a regiser pair. For example, to display the contents of the register pair HL, a user can type either R H ↵ or R L ↵.
3. To display the contents of all registers, press

   R ↵

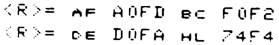
   The display will show the contents of two register pairs--AF and BC. To examine the contents of the successive register pairs, press the ↓ key. Using the procedure described here, the contents of the registers are to be displayed following the order--

   AF, BC, DE, HL, A'F', B'C', D'E', H'L', IX, IY, SP, PC, IF

4. To display the contents of the A', press

   R A ' ↵

   The registers A', F', B', C', D', E', H', L' are printed by the printer as follows:



```
<R>= AF A0FD BC F0F2
<R>= DE D0FA HL 74F4
```

5. The Use of the ↑ Key:
   The use of the ↑ key is quite the contrary to that of the ↓ key when used in conjunction with the R key.

6. The : Key
   --The Key to Alter the Contents of Registers
   The format of the command is:

   R <register> : <data> ↵

   To alter the contents of 16-bit registers or register pairs, such as IX, IY, SP, PC or BC, DE, four hexadecimal letters should be entered after : . If more than four hexadecimal letters are input, the MPF-IP only accepts the four hexadecimal letters last entered. To alter the contents of 8-bit registers, only two hexadecimal letters should be entered. If more than two hexadecimal letters are entered, the MPF-IP only accepts the two hexadecimal letters last entered.

To alter the contents of A', F', B', C', D', E', H', L', type either

```
R B C ' : 1 2 3 4 or

R B ' : 1 2 ↵

R C ' : 3 4 ↵
```

After the R command has been entered, if a key, which is not related with registers, is pressed, the display will show the contents beginning from the first register pair--AF.

4.4.4 The W Command—The Command Used for Storing Data on Tape

With its audio tape interface, the MPF-IP can write data from its RAM to tape. The W command is provided for achieving that purpose. The command format is:

W <starting address> space bar <ending address> space bar <filename> ↵

The above command stores the data of a memory range specified by starting and ending addresses under a given filename. The filename consists of four alphanumeric characters or less. If more than four alphanumeric characters are entered as a filename, only the first four are accepted as legal filename.

Because more than one files can be stored on a tape, various program or data files are identified by different filenames.

Before pressing the ↵ key, make sure

1) Both ends of the recorder line are plugged into the MIC jacks of the MPF-IP and the recorder.

2) Rewind the tape properly--
Rewind a new tape to the beginning of the tape. For a tape on which files already exist, rewind the tape so that the newly created file will not overlap with files created previously.

3) The PLAY and REC buttons of the tape recorder are already depressed.

During the data transmission from the MPF-IP to tape, the TONE-OUT lamp lights up and the speaker sounds a noise. But the display shows nothing during the data transmission process.

4.4.5 The L Command—The Command to Read Data from Tape back to Memory

The format of the command is:

```
L <filename> ↵
```

To read the file whose filename is PACE from tape to the MPF-IP, press the following:

```
<L>=PACE ↵
```

Before pressing the ↵ key on the MPF-IP, make sure

1) Both ends of the recorder line are plugged into the EAR jacks of the MPF-IP and the recorder.

2) The voice volume of the recorder is set higher than middle level.

After the ↵ key was pressed, press the PLAY button on the recorder.

Because the filename, starting and ending addresses are recorded on tape, a user only has to type in the filename--in this case, PACE. The MPF-IP will search the filename on tape. When the MPF-IP found the specified file on tape, it will read data contained in the memory range, which is specified by the starting and ending addresses, to the SAME memory location in RAM.

When the MPF-IP writes from memory to tape, a checksum will be produced and written at the end of a file. When the MPF-IP reads from tape to memory, it will produce a checksum according to the values (data) being read. At the end of the reading operation, the MPF-IP will compare the checksum so generated with the checksum that is written on tape when data is first recorded from memory to tape. If thre two checksums are identical, the reading operation is performed successfully. Otherwise, the error message "ERRORS" will appear on the display.

During the reading process, four dots will illuminate on the display. If a tape contains several files, the filenames will be displayed one after another until the specified file is located. When the MPF-IP locates the specified file, four -'s will be displayed. After the reading operation is performed successfully, the display will appear in basic form:

<L>=<filename>

When reading data from tape to memory, make sure that data cannot be read to the area used as system RAM. If data from tape is read into system RAM area, program will not execute properly.

Because the L command generates noises while reading data from tape to memory, entering a filename that does not exist on the tape enables a programmer to locate the blank area on a tape. When a user intends to write data from memory to tape, this skill is very helpful for a programmer to locate usable space on a tape.

4.4.6 The J Command—The Command Used to Calculate Relative Address

Relative address is used in such instructions as JR and DJNZ. The J command enables a programmer to calculate relative address easily. The format of the command is:

J <starting address> space bar <destination address> ↵

The starting address is where the opcode of a JR or DJNZ instruction is located, or from where a JR or DJNZ is to jump. The destination is where a JR or DJNZ instruction will jump to. Because a JR or DJNZ instruction can jump +127 or -128 locations, if the result of a J command is greater than +127 or less than -128, the display will show "ERRORS".

The following example demonstrates how to use the J command.

The JR instruction at address F860 is to jump to location F8C4. The relative address should be put into location F861. First use the M command to put 18--the opcode for JR--into F860. Then type

```
<J>=F860 space bar F8C4 ↵
```

to calculate the relative address and then put the resulant relative address to the location F861--the location for storing the oprand. Because the display does not echo what the MPF-IP has achieved, a user can examine the locations F860 and F861 by using the M command. If a printer is connected to the MPF-IP, the printer will print the above as follows:

```
<M>=F860:18

<J>=F860 F8C4

<M>=F860 F867

F860 18 62 00 00
F864 00 00 00 00
```

The J command is very useful when the one pass line assembler is in use. When a user intends to use the JR or DJNZ instruction but cannot make certain where will be the destination address, the programmer can first type in JR xxxx (which stands for a decimal number between -128 and +127). The programmer can use the J command to calculate the relative address for the JR or DJNZ instruction when the line assembler proceeds to the destination of the JR or DJNZ instrucion.

4.4.7 The I Command—The Command for Inserting Data into Memory

The use of the I command is demonstrated in the following example.

Here is a memory range started from F800 to F813. The contents of this memory range are listed as follows:

```
<M>=F800:10 11 12 13
 14 15 16 17 18 19
<M>=F80A:20 21 22 23
 24 25 26 27 28 29
<M>=F800.F813

F800 10 11 12 13
F804 14 15 16 17
F808 18 19 20 21
F80C 22 23 24 25
F810 26 27 28 29


<I>=FE00/
<I>=F804 1 2 3 4 5
```

Now the contents of five bytes--1, 2, 3, 4, 5--are to be inserted into this memory range. The contents of the first byte "1" is to be placed into F8Ø5; that of the second byte "2" is to be placed into F8Ø6; "3" into F8Ø7; "4" into F8Ø8; and "5" into F8Ø9. Type J F8Ø4 space bar 1 space 2 space 3 space 4 space 5 ⏎ . The printer should print

<J>=F8Ø4 1 2 3 4 5

Use the M command to examine if the data is inserted properly. Type M F8ØØ ⊡ F813 ⏎. The display should show:

```
F800 10 11 12 13
F804 14 01 02 03
F808 04 05 15 16
F80C 22 23 24 25
F810 26 27 28 29
```

Note when inserting data into memory, the insertion is made beginning from the address following the address specified in the command line.

Because five bytes of data were inserted to the memory range, the five bytes of data--15, 16, 17, 18, 19--which previously occupied the locations from F8Ø5 through F8Ø9 were shifted five locations.

Since the insertion causes shifting of data, a limit address (or default value) is set as soon as the I command is entered, so that the shifting of data will be limited by the default value--data will never be shifted beyond the limit address (default value) FEØØ. Note that after pressing I, the display always shows

<I>=FEØØ/

The default value is set to prevent data contained in system and user RAM from being destroyed by the shifting of data. Before using the I command, a user may examine the default value by typing

I ↵

After typing I and ↵, the display (or the prrinter--if the printer is on) will print

```
<I>=FEØØ/
```

The default value can be changed by the user. Type

I F B Ø Ø ↵

The display or the printer will print

```
<I>=FEØØ/FBØØ
```

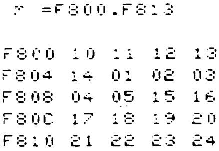
After changing the default value, a programmer can reset the default value by typing

I C ↵

The display or the printer will print

```
<I>=FBØØ/C
```

Each time a byte of data is inserted, the byte that precedes the limit address (default value) before the insertion is shifted out. Thus, if five bytes of data are to be inserted, the five bytes that precede the limit address will be shifted out. The following example shows how data of some bytes is lost after an insertion of data.



```
>  =F800.F813

F800 10 11 12 13
F804 14 01 02 03
F808 04 05 15 16
F80C 17 18 19 20
F810 21 22 23 24
```

The above example shows that after a data insertion, the four bytes that precede the limit address before the data insertion was shifted out. It should be noted that the I and D commands are very useful when using the line assembler. However, after using the I and D commands, the relative address following the JR or DJNZ commands should be verified using the J command.

Note data can not be inserted into ROM area. Because the MPF-IP only accepts input lines of 4Ø or less characters, the command line for an insertion should not exceed 4Ø characters. For more details of the I command, refer to the MPF-IP Monitor Program Source Listing.

4.4.8 The D Command—The Command for Deleting Data from Memory

The functions of the D command is contrary to that of the I command. The D command also causes the shifting of data in memory. Therefore, a default value is set as soon as the D command is entered to prevent data in system and user stack from being changed. The default value is also FEØØ.

A. Before using the D command, a user may want examine the default value by typing

I ←┘

After typing D and ←┘, the display (or the printer--if the printer is on) will print

```
<D>=FEØØ/
```

B. The default value can be changed by the user. Type

D F B Ø Ø ←┘

The display or the printer will print

```
<D>=FEØØ/FBØØ
```

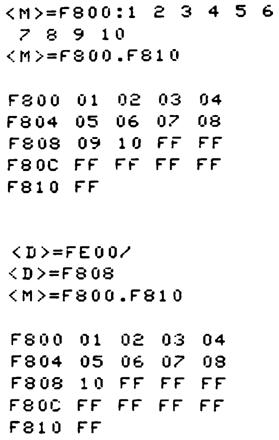
C. After changing the default value, a programmer can reset the default value by typing

D C ←┘

The display or the printer will print

```
<D>=FBØØ/C
```

After entering the D command following step A, or B, or C, the display will prompt the user to enter a starting address by printing <D>= . The user may enter the starting address to perform a data deletion. The example below shows a data deletion process.



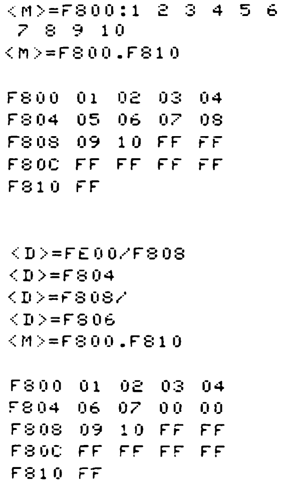
```
<M>=F800:1 2 3 4 5 6
 7 8 9 10
<M>=F800.F810

F800 01 02 03 04
F804 05 06 07 08
F808 09 10 FF FF
F80C FF FF FF FF
F810 FF


 <D>=FE00/
<D>=F808
<M>=F800.F810

 F800 01 02 03 04
 F804 05 06 07 08
 F808 10 FF FF FF
 F80C FF FF FF FF
 F810 FF
```

The following example deletes the data in two bytes-- F8Ø4 and F8Ø6. The limit address is set to F8Ø8 in the beginning.



```
<M>=F800:1 2 3 4 5 6
 7 8 9 10
<M>=F800.F810

F800 01 02 03 04
F804 05 06 07 08
F808 09 10 FF FF
F80C FF FF FF FF
F810 FF


 <D>=FE00/F808
 <D>=F804
 <D>=F808/
 <D>=F806
 <M>=F800.F810

 F800 01 02 03 04
 F804 06 07 00 00
 F808 09 10 FF FF
 F80C FF FF FF FF
 F810 FF
```

Each time a byte is deleted from the memory, the byte which precedes the limit address is filled with a zero and the contents in the bytes that follow the deleted byte are shifted.

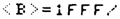
4.5 Program Debugging

4.5.1 The B Command—The Command to Set and Clear Breakpoint

The default value of breakpoint is 1FFF after power is applied to the MPF-IP or a warm reset.

A. To examine the breakpoint, type $\boxed{B}$ $\boxed{\hookleftarrow}$. The display or printer should print

```
<B>=1FFF
```



```
<B>=1FFF/
```

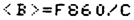
B. To change the breakpoint, type $\boxed{B}$ F860 $\boxed{\hookleftarrow}$. The printer or display should print

```
<B>=1FFF/F860
```

```
<B>=1FFF/F860
```

C. To reset the breakpoint after changing the breakpoint, type $\boxed{B}$ $\boxed{C}$ $\boxed{\hookleftarrow}$. The display or printer should print

```
<B>=F860/C
```



```
<B>=F860/C
```

Note only one breakpoint can be set with the B command. If an instruction has more than one byte, the breakpoint should be set at the first byte of the instruction. Otherwise, it will cause error when executing the program.

When processing breakpoint, the MPF-IP will use user's stack. When the execution of a program is interrupted, the state of the CPU remains unchanged, including the interrupt mode and the state of the interrupt flip-flop.

4.5.2 The S Command—The Command to Single-step a Program

The format of the command is:

$\boxed{S}$ <starting address> $\boxed{\hookleftarrow}$

The command allows a program to be executed instruction by instruction. This command allows a programmer to examine the state of registers and memory after an instruction is executed.

After the ↵ key is pressed, the CPU will execute a instruction specified by the starting address then stop. When the CPU stops, the display of the MPF-IP will display the address of the next instruction to be executed, e.g., the contents of the program counter.

To execute the next instruction, press the S key. After an instuction is executed, control will be returned to the monitor.

↓ and ↑ Key:

The up-down arrow keys can be used to set the register (register pairs) whose contents are to be examined each time you press the S key. Once the up-down arrow key is used to set the register (register pairs), the contents of the selected register (register pairs) are displayed each time an instruction is single-stepped.

To execute the next instruction, press the S key. and the display of MPF-IP will display, besides the address of the next instruction, a pair of register which you had set to be displayed every time you press the S key. If you intend to change the register on the display, just press the ↓ or ↑ key. At the moment you press one of these two keys, the CPU will not execute any program instruction, i.e., the CPU is broken off for a moment until you press the S key again. After an instruction is executed, control will be returned to the monitor.

If the starting address is not entered in the command line, the CPU will execute from address ØØØØ.

The following example program starts from F8ØØ.

```
F8ØØ  LD   A,1
F8Ø2  LD   D,2
F8Ø4  ADD  A,2
F8Ø6  LD   A,D
F8Ø7  LD   A,5
F8Ø9  LD   A,6
```

Single-step the program from F8Ø0, and keep single-stepping the program. The printer should print:

```
<S>F8ØØ (S)
F8Ø2 DE Ø2FØ HL ØFØF ( ▲ )
F8Ø2 AF Ø1ØØ BC Ø216 ( S )
F8Ø4 AF Ø1ØØ BC Ø216 ( S )
F8Ø6 AF Ø4ØØ BC Ø216 ( ▼ )
F8Ø6 DE Ø2FØ HL ØFØF ( ▲ )
F8Ø6 AF Ø4ØØ BC Ø216 ( S )
F8Ø7 AF Ø2ØØ BC Ø216 ( S )
F8Ø9 AF Ø5ØØ BC Ø216 ( S )
F8ØB AF Ø6ØØ BC Ø216 ( ▼ )
F8ØB DE Ø2FØ HL ØFØF ( CTRL_Q )
<
```

The monitor uses the user's stack when a program is single-stepped. Thus, the stack pointer should point to the user's stack in RAM--location FEAØ. Otherwise, the MPF-IP will detect immediately and display ERR-SP. If the stack pointer points to the system stack used by the monitor, SYS-SP will be displayed, because stack overlapping causes mistake when the instruction of RET is executed. When stack overlapping occurs, the stack pointer should be reset to its default value. or RESET be pressed.

Once the MPF-IP is reset, the monitor will set the user's stack pointer to its default value--FEAØ. If a user's program does not affect the SP register, then the stack overlapping will not occur.

The purpose of single-stepping a program is to enable a programmer to trace the running process of a program. However, if a program is too long, single-stepping a program is too time consuming. In this case, the tracing of a running program can be achieved by setting breakpoint in the program.

4.5.3 The G Command—The Command for Executing a program

The format of the command is:

<G>=<starting address> ↵

If no starting address is specified in the command line, the CPU will execute according to the value of the program counter.

The following example calls for a programmer to

1) Type in a short assembly program;
2) Set a breakpoint in the program;
3) Use the G command to execute the program;
4) Use the R command together with the v key to examine the registers;
5) Use the S command to single-step the remaining instructions of the program after execution of the program was interrupted by a breakpoint.

The program to be entered is listed below:

```
F8ØØ LD   A,1
F8Ø2 LD   A,2
F8Ø3 LD   A,3
F8Ø4 LD   A,4
F8Ø5 LD   A,5
```

The program may be entered by using the M command:

```
<M>=F800:3E 1 3E 2 3
E 3 3E 4 3E 5
```

Use the disassembler by typing CONTROL D to examine if the program is entered correctly:

```
<D>=F800 F809

F800 3E LD A,01
F802 3E LD A,02
F804 3E LD A,03
F806 3E LD A,04
F808 3E LD A,05
```

Set the breakpoint:

```
<B>=1FFF/F804
```

Use the G command to execute the program:

<G>=F8ØØ ↵

Use the R command and ↓ key to examine the contents of registers.

```
F806 AF 0300 BC FF00
F806 DE FF00 HL FF00
F806 AF 96FF BC 90FF
F806 DE 41FB HL 08FF
F806 IX FF00 IY FF00
F806 SP FEA0 PC F806
```

Use the S command to single-step the remaining instructions of the program.

Note: After the execution of a program was interrupted by a breakpoint, the display will show the current value of the program counter--the next instruction to be executed--and the contents of register pairs AF and BC. The user may press the v key to examine the contents of other registers. After the user has examined the registers, he can press the G or the S key to execute the remaining instructions.

EXERCISES

4.1 Print the contents of the memory range from ØØØØ to ØØ1Ø.

4.2 Move the contents of the memory range from ØØØØ to ØØ1Ø to the memory range starting from F9ØØ to F91Ø.

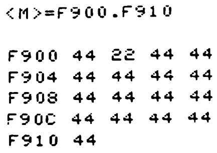
4.3 Print the contents of the memory range from F9ØØ to F91Ø.

4.4 Change the contents of memory location F9ØØ to 44 and that of F9Ø1 to 22.

4.5 Examine the contents of the memory locations F9ØØ and F9Ø1 to see whether their contents have been altered to 44 and 22.

4.6 Fill 44 to the memory range from F9Ø2 to F91Ø.

4.7 Dump the contents of the memory range F9ØØ through F91Ø to see if the contents have been changed.



```
<M>=F900.F910

F900 44 22 44 44
F904 44 44 44 44
F908 44 44 44 44
F90C 44 44 44 44
F910 44
```

Chapter 5 Useful Subroutines

5.1 MPF-IP System Parameters

ADDRESS	LABEL	BYTES	FUNCTION
0FED0H	STEPBF	4	Tape File Name
0FED4H	STEPBF+4	2	Tape Starting Address
0FED6H	STEPBF+6	2	Tape Ending Address
0FED7H	STEPBF+8	1	Tape Check Sum
0FED9H	EDIT-START-ADDR	2	Editor Bottom Assembler Text Buffer From
0FEDBH	END-DATA-ADDR	2	Editor Top Assembler Text Buffer To
0FEDDH	END-LN-NO	2	Editor Last Line Number
0FEDFH	RAM-START-ADDR	2	Editor Low Limit
0FEE1H	EDIT-END-ADDR	2	Editor High Limit
0FEE3H	ST-F	2	Assembler Symbol Table From
0FFE5H	ST-T	2	Assembler Symbol Table To
0FEE7H	OBJ-F	2	Assembler Object Code From
0FEE9H	OBJ-T	2	Assembler Object Code To
0FFEBH	END-ADDR	2	Limit of Insert and Delete
0FEEDH	BRAD	2	Break Point Address
0FEFFH	BRDA	1	Data Of Break Point Address
0FFF0H	POWERUP	1	Power Up Initialization

ADDRESS	LABEL	BYTES	FUNCTION
0FEF1H	TEST	1	Test Flag
0FEF2H	STEPFG	1	Step Test Flag
0FEF3H	PRTFLG	1	STEP mode test flag
0FEF4H	BEEPSET	1	BEEP toggle switch
0FEF5H	FBEEP	1	Beep Frequency
0FEF6H	TBEEP	2	Time Duration Of Beep
0FFF8H	MADDR	2	Temporary Storage
0FFFAH	TEMP1	4	Temporary Storage
0FEFFH	ATEMP	1	Temporary Storage
0FEFFH	HLTEMP	2	Temporary Storage
0FF01H	IMIAD	2	Contains the address of Opcode FF' Service Routine (RST38H)
0FF03H	RCOUNT	1	Register Counter
0FF04H	INPBF	40	Input Buffer
0FF2CH	DISPBF	82	Display Buffer
0FF7EH	GETPT	2	Check Hex pointer
0FF80H	TYPEFG	1	Memory and Register Test Flag
0FF81H	CRSET	1	Display delay time
0FF82H	OUTPTR	2	Input buffer pointer
0FF84H	DISP	2	Display buffer pointer
0FF86H	INPTR	2	Limit of input buffer pointer
0FF88H	REGBF	26	Register Buffer
0FFA2H	EDITOR	14	RAM Buffer For Editor
0FFB0H	ASSEMBLER	79	RAM Buffer For Assembler

5.2 Input/Output Parameters and Summary of Subroutines

1. Input buffer: INPBF - INPBF+39
   The input buffer consists of 40 bytes starting from INPBF to INPBF+39. Data is stored in the input buffer in ASCII format. Thus, up to 40 ASCII characters can be stored in the input buffer.

   When a user intends to print the contents in the input buffer, set IX = INPBF and then call MTPPRT, then the data in the input buffer will be printed out.

2. Input buffer pointer: (OUTPTR)
   The input buffer pointer is expressed by (OUTPTR).

3. Input buffer lower limit: (INPTR)

4. Display buffer: DISPBF - DISPBF + 81
   Display buffer pointer: (DISP)
   The address of (DISP) is the address in the display buffer from where the display pattern for the next character to be displayed is stored.

5.2.1 BEEP

```
[Address ]: Ø8Ø3H
[Function]: Call TONE to generate sound.
[Input   ]: None
[Output  ]: None
[Register]: AF, BC, DE, HL
[Call    ]: None
```

5.2.2 CHK 40

```
[Address ]: Ø912H
[Function]: Check the number of contents in the display
            buffer.  If the number is greater than 4Ø,
            change the IX pointer.
[Input   ]: (DISP)
[Output  ]: IX ←— IX  (If the number of  contents  is
            less than 4Ø.)
            IX ←— (DISP)-38
            Carry flag = 1 if (DISP) < (DISPBF+38)
[Register]: AF, DE, HL, IX
[Call    ]: None
```

5.2.3 CHRWR

```
[Address ]: Ø924H
[Function]: Convert  a byte (ASCII code) in A  register
            to  display  patterns and store  them  into
            display    buffer    and    input    buffer
            respectively.  Then call CURSOR.
[Input   ]: A, (DISP), (OUTPTR)
[Output  ]: Store   the  ASCII  code  contained  in   A
            register in (OUTPTR).  The display pattern
            is made up of two bytes.  The first byte is
            stored  in (DISP),  and the second byte  is
            stored in (DISP)+1.
            (OUTPTR) ←— (OUTPTR)+1
            (DISP) ←— (DISP)+2
[Register]: AF
[Call    ]: CONVER, CURSOR
```

5.2.4 CLEAR

```
[Address ]: Ø9B9H
[function]: Clear  the  display  buffer,  and  set  the
            contents of DISP and OUTPTR to the starting
            address  of display buffer and input buffer
            respectively.
[Input   ]: None
[Output  ]: (OUTPTR) ←— INPBF
            (DISP)   ←— DISPBF
```

[Registe]: None
[Call]: CLRDSP

5.2.5 CLRBF

[Address]: Ø7F6H
[Function]: Call CLEAR, set IX to be the starting address of the display buffer, and call CHRWR to generate
[Input]: None
[Output]: (OUTPTR) ← INPBF+1
(DISP) ← DISPBF+2
IX ← DISPBF
[Register]: AF, IX
[Call]: CLEAR, CHRWR

5.2.6 CLRDSP

[Address]: Ø84ØH
[Function]: Clear the display buffer.
[Input]: None
[Output]: None
[Register]: None
[Call]: None

5.2.7 CONVER

[Address]: Ø821H
[Function]: Convert a byte (ASCII code) in A register to display pattern and store them in display buffer.
[Input]: A, (DISP)
[Output]: The display pattern consists of two bytes. The first byte is stored in (DISP), and the second byte in (DISP)+1.
(DISP) ← (DISP)+2
[Register]: AF
[Call]: None

5.2.8 CR

[Address]: Ø93BH
[Function]: Print out all the contents in input buffer. Check the TV interface. If TV interface board exists, then jump to TV interface service routine.
[Input]: (OUTPTR)
[Output]: (OUTPTR) ← INPBF
(DISP) ← DISPBF
[Register]: AF
[Call]: CRØ, PTEST, PRINTT, CLEAR, CURSOR

5.2.9 CR 1

[Address]: Ø97AH
[Function]: The same as CR but the display timing is about 1 sec.
[Input]: (OUTPTR)
[Output]: (OUTPTR) ← INPBF
(DISP) ← DISPBF
[Register]: AF, B, A'F', B'C', D'E', H'L', HL.
[Call]: CRØ, PTEST, SCAN1, PRINTT, CLEAR, CURSOR

5.2.10 CR 2

[Address]: Ø981H
[Function]: The same as CR but CR2 do not call CLEAR and CURSOR.The display time is about 32Ø msec.
[Input]: (OUTPTR)
[Output]: None
[Register]: AF,B,A'F',B'C',D'E',H'L'.
[Call]: CRØ, PTEST, PRINTT

5.2.11 CR 3

[Address]: Ø985H
[Function]: The same as CR but CR3 call routine CLRBF instead of CLEAR. The display time is about 48Ø msec.
[Input]: (OUTPTR)
[Output]: (OUTPTR) ← INPBF+1
(DISP) ← DISPBF+2
[Register]: AF, IX
[Call]: CRØ, PTEST, CLRBF

5.2.12 CURSOR

[Address]: ØA79H
[Function]: Get cursor message
[Input]: (DISP)
[Output]: The first byte of cursor in (DISP) and the second byte of cursor in (DISP)+1.
(DISP) ← (DISP)
The contents of (DISP) remains the same.
[Register]: AF
[Call]: CONVER

5.2.13 DECBIN

[Address]: ØB28H
[Function]: Convert decimal numbers (in ASCII codes) to hexadecimal numbers until a non-decimal number (the numbers not in the range from Ø to 9) is encountered.

[Input]: DE. The value of DE is a pointer that points to the first ASCII code to be converted.
[Output]: HL. The hex values returned by the subroutine are stored in HL.
[Register]: AF, BC, DE, HL
[Call]: None

5.2.14 DECIMAL

[Address]: ØAB8H
[Function]: Convert hexadecimal values in HL to corresponding decimal values (in ASCII code format). Store decimal value into input buffer and its corresponding display pattern to display buffer.
[Input]: HL is used to store the *hex values to be converted.
(OUTPTR) points to the starting address of the input buffer.
(DISP) points to the starting address of the display buffer.
[Output]: (OUTPTR) ⟵ (OUTPTR)+?
(DISP) ⟵ (DISP)+2*?
? is the number of characters to be printed.
[Register]: AF, BC, DE, HL, IY
[Call]: CHRWR
[Example]: Given the value of HL is Ø2ØØH, its decimal equivalent--512--will be returned after calling DECIMAL. 512 will be stored in ASCII form (35 31 32) in the input buffer and its display pattern is stored in display buffer.

5.2.15 DEC-SP

[Address]: Ø399H
[Function]: Put FF in (DISP) and (DISP)+1
[Input]: (DISP)
[Output]: (DISP) remains unchanged.
[Register]: AF, HL
[Call]: None

5.2.16 ERROR

[Address]: Ø6C4H
[Function]: Print ERROR message and call PRTMES
[Input]: None
[Output]: (OUTPTR) ⟵ INPBF+8
(DISP) ⟵ DISPBF+16
[Register]: AF, HL
[Call]: PRTMES

5.2.17 GETCHR

[Address]: Ø8AEH
[Function]: Use (GETPT) as a pointer. Load (GETPT) to HL and increment HL until (HL-1) is one of the following delimiters: SPACE, TAB, :, ., =, / and (HL+1) is not SPACE or TAB.
[Input]: (GETPT)
[Output]: HL ← HL+?
(GETPT) ← (GETPT)+?
[Register]: AF, HL
[Call]: None

5.2.18 GETHL

[Address]: Ø8E5H
[Function]: Call GETCHR. Using HL as a pointer, convert ASCII codes to hex values and store them into HL.
[Input]: (GETPT)
[Output]: (GETPT) ⟵ (GETPT)+?
HL is stored with hex value. If there is only one hex digit, H = Ø and the digit is stored in L. If the data is not hexadecimal digits, carry flag = 1. If the last ASCII code is <CR>, zero flag = 1.
[Register]: AF, DE, HL
[Call]: GETCHR, ONE

5.2.19 HEXBIN

[Address]: ØAF4H
[Function]: Convert ASCII codes to hex values until non-hex digit is encountered. DE is used as a pointer.
[Input]: The value of DE is the pointer, which points to the first location of ASCII code to be converted.
[Output]: The value of HL is the hex numbers after being converted. (HEXFLAG) is set if there exists a digit within (A, B, C, ...F) or the last none hexadecimal character is 'H'.
[Register]: AF, BC, DE, HL
[Call]: ONE
[Note]: 1) The execution of this subroutine stops when the value of (DE) is not within the range from 3Ø to 39 and the range from 41 to 46. Refer to the MPF-IP ASCII code table.
2) If the data to be converted is stored from the location F8ØØ, then the value of DE should be set to F8ØØ. After

calling HEXBIN, the value of HL will be 1234 and (HEXFLAG) = Ø.

5.2.20 HEX 1

```
[Address ]: ØAADH
[Function]: Convert  the least significant four bits in
            register A (binary data) to ASCII code  and
            display pattern, and call CHRWR.
[Input   ]: A, (DISP), (OUTPTR)
[Output  ]: The ASCII code is stored in (OUTPTR).   The
            display  pattern consists of two bytes--the
            first  byte is stored in  (DISP),  and  the
            second is stored in (DISP)+1.
            (OUTPTR) ← (OUTPTR)+1
            (DISP) ← (DISP)+2
[Register]: AF
[Call    ]: CHRWR
```

5.2.21 HEX 2

```
[Address ]: ØA9AH
[Function]: Convert the contents in A register (two hex
            numbers  - one byte) to two ASCII codes and
            display patterns. Call HEX1 twice.
[Input   ]: A, (DISP), (OUTPTR)
[Output  ]: The  ASCII  code converted  from  the  most
            significant four bits is stored in (OUTPTR)
            while the first byte of its display pattern
            is  placed into (DISP) and the second  byte
            of its display pattern into (DISP)+1.

            The  ASCII  code converted from  the  least
            significant   four   bits   is   stored   in
            (OUTPTR)+1  while  the first  byte  of  its
            display pattern is placed into (DISP)+2 and
            the second byte of its display pattern into
            (DISP)+3.
            (OUTPTR) ← (OUTPTR)+2
            (DISP) ← (DISP)+4
[Register]: AF
[Call    ]: HEX1
```

5.2.22 HEX 4

```
[Address ]: ØA92H
[Function]: Call HEXX and SPACE1.
[Input   ]: HL, (DISP), (OUTPTR)
[Output  ]: In  addition  to the  output  generated  by
            HEXX,  the  ASCII code of 'SPACE' is stored
            in  (OUTPTR)+5 while the first byte of  its
            display  pattern is placed in (DISP)+8  and
```

the second byte in (DISP)+9.
(OUTPTR) ← (OUTPTR)+5
(DISP) ← (DISP)+10

[Register]: A

[Call]: HEXX, SPACE1

5.2.23 HEXX

[Address]: 0A89H

[Function]: Convert the two bytes of hex values in HL to ASCII codes and display patterns. Call HEX2 twice.

[Input]: HL, (DISP), (OUTPTR)

[Output]: The ASCII code converted from the most significant four bits in H is stored in (OUTPTR) while the first byte of its display pattern is placed in (DISP) and the second byte is placed in (DISP)+1.

The ASCII code converted from the least significant four bits in H is stored in (OUTPTR)+1 while the first byte of its display pattern is placed in (DISP)+2 and the second byte is placed in (DISP)+3.

The ASCII code converted from the most significant four bits in L is stored in (OUTPTR)+2 while the first byte of its display pattern is placed in (DISP)+4 and the second byte is placed in (DISP)+5.

The ASCII code converted from the least significant four bits in L is stored in (OUTPTR)+3 while the first byte of its display pattern is placed in (DISP)+6 and the second byte is placed in (DISP)+7.
(OUTPTR) ← (OUTPTR)+4
(DISP) ← (DISP)+8

[Register]: AF

[Call]: HEX2

5.2.24 LDA

[Address]: 08B1H

[Function]: The same as that of GETCHR. But LDA sets HL directly.

[Input]: HL

[Output]: The same as that of GETCHR.

[Register]: AF, HL

[Call]: None

5.2.25 MSG

[Address]: Ø9CAH

[Function]: Convert ASÇII code stored in input buffer to display patterns and put the resultant display patterns to display buffer until a <CR> is encountered. HL is used as the pointer for the input buffer.

[Input]: HL, (DISP), (OUTPTR)

[Output]: HL ← HL+?
(OUTPTR) ← (OUTPTR)+?
(DISP) ← (DISP)+2*?

[Register]: AF, HL

[Call]: CHRWR

5.2.26 MTPPRT

[Address]: 6A4ØH

[Function]: Print the contents of the memory range pointed by IX until a <CR> is encountered.

[Input]: IX

[Output]: None

[Register]: A'F', B'C'

[Note]: The use of MTPPRT is listed below:

1) Set the value of IX, which points to the starting address of a memory range to be printed;
2) MTPPRT regards ØA as a line feed signal, Ø9 as a TAB, and ØD as the end of the memory range;
3) The data to be printed is stored in memory in the form of ASCII codes and should be ended with ØDH;
4) When the data to be printed exceed 2Ø characters, MTPPRT will generate a line feed signal automatically.

5.2.27 ONE

[Address]: ØB14H

[Function]: Convert a byte (ASCII code) in A register to hex digit.

[Input]: A (ASCII code)

[Output]: A (hex number)
If the data is not a hex number,
carry flag = 1.
If the value of A falls within A to F,
* (HEXFLAG) ≠ Ø.

[Register]: AF

[Call]: None

5.2.28 PLINE

[Address]: 6A3ØH
[Function]: Call PLINEFD twice and perform line feed twice.
[Input]: None
[Output]: None
[Register]: AF, B

5.2.29 PLINEFD

[Address]: 6A1ØH
[Function]: Perform a line feed action.
[Input]: None
[Output]: None
[Register]: AF, B

5.2.30 PRINTT

[Address]: Ø893H
[Function]: Call PTEST. If MPF-IP is connected with PRT-MPF and the printer is on, print out all contents in the display buffer.
[Input]: None
[Output]: None
[Register]: AF
[Call]: PTEST, MTPPRT

5.2.31 PRTMES

[Address]: Ø886H
[Function]: Call MSG. Display the contents of a memory range on display and print the same with PRT-MPF.
[Input]: HL: The starting address of the memory range.
[Output]: (OUTPTR)+?
(DISP)+2*?
[Register]: AF, HL
[Call]: CLEAR, MSG, DECDSP, CR2

5.2.32 PTEST

[Address]: Ø8A3H
[Function]: Check the condition of the toggle switch of the printer. If it is on, call PTESTT.
[Input]: None
[Output]: 1) Zero flag = 1 if a printer exists and the toggle switch is on.
2) Zero flag = Ø when the printer does not exist the toggle switch is on.
3) Zero flag = Ø when the printer is off.

[Register]: AF

[Call]: PTESTT

5.2.33 PTESTT

[Address]: 08A8H

[Function]: Check if the MPF-IP is connected with the PRT-MPF.

[Input]: None

[Output]: Zero flag = 1 if the MPF-IP is connected with the PRT-MPF.

[Register]: AF

[Call]: None

5.2.34 RAMCHK

[Address]: 0819H

[Function]: Check if a memory address is in RAM.

[Input]: HL is stored with the address to be checked.

[Output]: Zero flag = 1 if the address is in RAM.
Zero flag = 0 if the address is not in RAM.

[Register]: None

[Call]: None

5.2.35 READLN

[Address]: 09D4H

[Function]: Read a string of characters ended with <CR>

[Input]: (DISP), (OUTPTR)

[Output]: (INPTR) ← (OUTPTR)
(OUTPTR) ← (OUTPTR)+?
(DISP) ← (DISP)+2*?

[Register]: AF, BC, DE, HL, A'F', B'C', D'E', H'L'

[Call]: CHK40, CURSOR, CR0, SCAN, CHRWR

5.2.36 SCAN

[Address]: 0246H

[Function]: Call SCAN2 and BEEP.

[Input]: IX points to the buffer containing display patterns.

[Output]: Internal code for the key pressed.

[Register]: AF, BC, DE, HL, A'F', B'C', D'E', H'L'

[Call]: SCAN2, BEEP

5.2.37 SCAN 1

[Address]: 029BH

[Function]: Scan the keyboard and display one cycle. Total execution time is about 16 ms (exactly 15.7 ms, 28040 clock states @ 1.79

MHz).

[Input]: The same as SCAN.

[Output]: 1) If no key is pressed, then carry flag = 1.
2) If a key press is detected during one scan, then carry flag = Ø and the position code of the key pressed is stored in A. (The position code is determined by its position in the 2Ø by 3 keyboard matrix. Refer to Chapter 8)

[Register]: AF, A'F', B'C', D'E', H'L'

[Call]: None

5.2.38 SCAN 2

[Address]: Ø24DH

[Function]: Similar to that of SCAN1, but differ with SCAN1 in two respects:
1) SCAN1 only scans once, while SCAN2 keeps scanning until a key is pressed.
2) SCAN1 gets a position code, while SCAN2 returns an ASCII code.

[Input]: The same as SCAN1.

[Output]: Internal code (ASCII code) of the key pressed.

[Register]: AF, BC, HL, A'F', B'C', D'E', H'L'

[Call]: SCAN1

5.2.39 SHIFT

[Address]: 6AØDH

[Function]: For controlling the PRT-MPF. Move the thermal head to the right. The greater the value in B, the farther the thermal head will be shifted to the right.

[Input]: B

[Output]: None

[Register]: AF, B

5.2.40 SKIP

[Address]: ØB4ØH

[Function]: Skip TABs and BLANKs. Use HL as a pointer, increment HL until (HL) is not SPACE or TAB.

[Input]: HL

[Output]: HL ← HL+?
? is the number of TABs or BLANKs and (HL) is not TAB or BLANK.
A ← (HL)
Carry flag = 1 if (HL) is not within the range from A to Z.
Carry flag = Ø if (HL) is within A to Z.

[Register]: AF, HL

[Call   ]: None

5.2.41 SPACE 1

[Address]: 0A95H

[Function]: Load 20H (SPACE) to A and then call CHRWR.

[Input   ]: (DISP), (OUTPTR)

[Output   ]: The same as that of CHRWR.

[Register]: AF

[Call   ]: CHRWR

5.2.42 TONE

[Address]: 0874H

[Function]: Generate a square wave to the MIC and speaker on MPF-IP.

[Input   ]:

1) The register C is used to control the frequency of the tone to be generated. Its cycle is 2*(44+13*C)*0.56 micro-sec, which equals to 200/(10+3*C)KHz.
2) HL is used to store the number of periods, which should be less than or equal to 32768.

[Output   ]: None

[Register]: AF, B(C), DE, HL

[Call   ]: None

5.2.43 TONE 1K

[Address]: 086FH

[Function]: Generate a sound of 1KHz.

[Input   ]: HL is used to store the number of periods, which should be less than or equal to 32768.

[Output   ]: None

[Register]: AF, BC, DE, HL

[Call   ]: None

5.2.44 TONE 2K

[Address]: 0872H

[Function]: Generate a tone of 2KHz.

[Input   ]: The same as that of TONE1K.

[Output   ]: None

[Register]: AF, BC, DE, HL

[Call   ]: None

Chapter 6
The Text Editor

The text editor of the MPF-IP is used to create text file--which normally consists of assembly language source programs. Source program or data is first entered from input devices such as the keyboard to the text buffer, which is an area in the RAM. Then, source program or data will be output to memory devices or executed.

The MPF-IP keyboard is normally used as the input device, and its 2Ø-character display and printer are used as output devices. Cassette tape is used as permanent storage for data and programs.

6.1 Text Buffer

On MPF-IP, text is stored in the text buffer, which may be specified by the user. When the text editor is initialized, a user may specify the starting address and the ending address of the text buffer. If the user does not specify the starting and ending addresses, the MPF-IP will set them automatically. In this case, two default values are set automatically by the MPF-IP.

When a 4K RAM is installed on board location U4, the default values are FØØØ and FAFF. That means the text buffer is the RAM area starting from FØØØ through FAFF. When the board location U4 is installed with a 2K RAM, the default values are F8ØØ through FCFF.

Text is stored in the text buffer in ASCII form. Each ASCII character is stored in a byte. A text line may consist of different number of characters and is always ended with a carriage return character "ØD" (Refer to the MPF-IP ASCII Code table). The ASCII code for carriage return "ØD" also requires one byte to store.

Thus, a user can easily calculate the RAM space necessary for the text buffer which can meet his specific programming need. When a user allocates a memory space in the RAM of the MPF-IP to be used as the text buffer, it is desirable that the text buffer be set larger than what is actually needed for the text buffer, enabling easier editing and modification of the source file in the future.

6.1.1 Line Pointer

A logic current line pointer is used to point to the location at which data (such as a character) will be stored. By logic, it means that the line pointer is actually used internally by the computer but does not has a physical form. Because the MPF-IP's editor is a line-oriented editor, a line pointer is neseccary to point to the location upon which an editor operation is to take place.

The current line pointer is always positioned in front of the first character of the current opened (accessed) line. The current opened line is also known as an active line. All editing operations begin from an active line. After an editor operation is completed, the line pointer either points to the beginning of the line last accessed or a newly opened line (the line that is one line down from the line last accessed.)

6.1.2 Length of a Line

In order to keep data stored in the memory accurately, the maximum number of characters you can key into each line is limited to 39 characters. If you enter more than 39 characters, the editor will not accept the characters following the 39th character and respond with a " Beep " sound.

6.2 Enter and Re-enter the Editor

There are two ways for you to enter from the monitor to the editor:

6.2.1 The "E" Command-Using the Editor in Input Mode

The E command is entered by typing the E key while holding down the CONTROL key.

After the E command is entered, the display will prompt a user to specify the starting address (lower limit) of the text buffer by displaying

```
F:
```

When being prompted by the editor, you can

1) Enter the starting address followed by a carriage return ↵. After you typed in the starting address and ↵ the MPF-IP will prompt you again to enter the ending address (upper limit) for the text buffer by displaying a "T". After typing in the ending address for the text buffer and the carriage return key ↵ the MPF-IP will display INPUT then the cursor of the editor. At this time, the user can enter the instructions of a mnemonic source program.

2) Press the ↵ key to enter the input mode of the editor and type in your program. Note that after the ↵ key is pressed, the MPF-IP will display the default values for the text buffer for a few seconds and then prompt you to enter your program.

3) The address range for text buffer:

Default	User Define
F:FBØØ T:FAFF	F:xxxxT:yyyy

When the MPF-IP is under the control of the monitor, the E command allows a user to enter the editor in input mode. Once the E command is entered, all editor parameters (default values) will be reset.

When the MPF-IP is in input mode, the display will print the editor prompt character /\ . After an instruction line, a carriage return is entered to seperate it from the next instruction line. Pressing the carriage return key twice allows a user to re-enter the editor in edit mode.

6.2.2 The "R" Command-Using the Editor in Edit Mode

The R command is entered by typing the R key while holding down the CONTROL key.

When the MPF-IP is under the control of the monitor, pressing "R" allows a user to re-enter the editor in edit mode without changing the parameters and the data that is already stored in the text buffer.

Note that the difference of the editor's E and R commands is that the E command resets the parameters (default values), while the R command enters the editor without changing the default values and the text entered with the text editor. After typing in the R command to re-enter the editor, the line pointer is always positioned in the beginning of the top line of the text buffer.

6.2.3 The-(TAB) Key

When the editor is in input mode, the ⇥ key is used the same way as the TAB key. The ⇥ key can be used efficiently to save memory space. For example, if the following instructions are to be entered, memory space can be used most efficiently by typing the keys in accordance with the following sequence:

⇥

I   FOR INC HL

N

C

␣ (SPACE)

H

L

↵ (CARRIAGE RETURN)

L   FOR LOOP CALL SCAN1

O

O

P

⇥

C

A

L

L

␣

S

C

A

N

1

↵ (CARRIAGE RETURN)

6.3 Summary of the Editor Commands

Category	Commands	Function
Editor Entry and Exit	Enter (CONTROL) Re-enter (CONTROL) Quit	Enter the editor from monitor Enter the editor from monitor Quit the editor and enter the monitor
Text Manipulating Commands	Delete Insert Print n Read/filename/ Write/filename/ Z	Delete a line Insert a line Print n lines Read data from tape Write data to tape Print all the data in text buffer
Line Pointer Manipulating Commands	Bottom G n Line number Next n Top Up n	Move the line pointer to the bottom of the file Move the line pointer to the nth line in the text buffer Print the line number of the line pointed to by the line pointer Move the line pointer to the next n line Move the line pointer to the top of the file Move the line pointer up n lines
String Handling Commands	Change/old string/ new string Find/string/	Change a string in the current line Find the line with the specified string
Other Commands	Space X Carriage Return	Print text buffer default values and the memory space used to store the current text file Control the prnter (a toggle switch) Display the next line

6.4 Editor Entry and Exit Commands

6.4.1 The E Command-Enter and initialize the editor

The E command has been discussed in detail in 6.2.1

6.4.2 The R Command-Re-enter the editor

The R command has been discussed in 6.2.2. Note that after entering the editor while the MPF-IP is under the control of the monitor, the edit mode prompt character "$" will prompt you to enter your program.

6.5 Text Manipulating Commands-The commands for data input/output/ update

6.5.1 The I Command-Insert Lines

The I command is used to insert program lines beginning from the active line. The following procedure examplifies the use of the I commad:

1) Find the current line with the following commands T, B, U, N, G or F.
2) Press I, and the MPF-IP will respond with

```
$I
INPUT
/\
```

3) When the editor prompt character "/\" appears on the display, input your instruction lines. A carriage return should follow each instruction line to identify the end of a program line. After all the program lines have been entered, type the carriage return key twice to return to the edit mode.

Example :

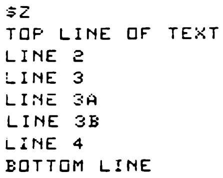
Use the T and Z commands to print the text file currently in the text buffer.

```
EDIT
$Z
TOP LINE OF TEXT
LINE 2
LINE 3
LINE 4
BOTTOM LINE
```

If two lines are to be input after the third line, use the T, G, and I commands.

```
$G 3
LINE 3
$I
INPUT
LINE 3A
LINE 3B
```

After the two lines have been inserted, print the file currently in the text buffer with the T and Z commands.



```
$Z
TOP LINE OF TEXT
LINE 2
LINE 3
LINE 3A
LINE 3B
LINE 4
BOTTOM LINE
```

6.5.2 The D Command-Delete a line

The D command allows a user to delete a line from the text file. The use of the command is examplified as follows:

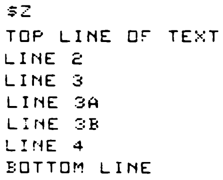
1) Locate the line to be deleted using the T, B, U, N, G and F commands.
2) Enter the D command, the MPF-IP will respond with

$D

3) Press the carriage return key, and the editor will delete the current line and move the line pointer up one line.

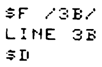
Example :

Print the data in the text buffer with the T and Z commands:



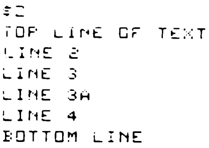
```
$Z
TOP LINE OF TEXT
LINE 2
LINE 3
LINE 3A
LINE 3B
LINE 4
BOTTOM LINE
```

Locate the line to be deleted with the T and F commands, and delete the line with the D command



```
$F /3B/
LINE 3B
$D
```

Print the data now in the text buffer using the Z command.



```
$Z
TOP LINE OF TEXT
LINE 2
LINE 3
LINE 3A
LINE 4
BOTTOM LINE
```

6.5.3 The P Command-Print a specified number of lines.

The P command allows a user to print n lines beginning from the current line. If the P command is not followed by a number, the editor will only print one line. The use of the command is examplified as follows:

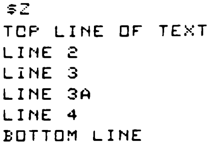
1) Locate the line to be printed using the T, B, U, N, G and F command.
2) Enter the P command which may or may not be followed by a number to specify the number of lines to be printed. The MPF-IP will respond as follows:

```
$P n
```

3) Press the carriage return key. The PRT-MPF-IP will print the data as specified.
4) After the MPF-IP executed the P command, the line points to the last line printed.
5) If the command line does not include the number of lines to be printed, the MPF-IP will print only one line.

Example :

Given the data in the text buffer is as follows:



```
$Z
TOP LINE OF TEXT
LINE 2
LINE 3
LINE 3A
LINE 4
BOTTOM LINE
```

If line 3, 4, and 5 are to be printed, you can use the G command to locate line 3.

```
$G 3
LINE 3
```

Then enter the command line "P 3" to print the three desired line.

```
$P 3
LINE 3
LINE 3A
LINE 4
```

On MPF-IP version 1.1, after a source program instruction is displayed, the Editor waits for the user to press the carriage return key. If the currently accessed instruction line has more than 20 characters, pressing the backspace [<--] key continuously will allow you to shift left the displayed line.

After you have finished examining the currently accessed line, pressing the carriage return key enables you to enter into the Edit Mode, and the Edit Mode prompt character " $ " will appear and wait for you to enter the next Editor command. If you want to examine the next line, you can simply press the carriage return key.

You can follow the above mentioned steps to watch several consecutive lines. However, it is inconvenient to press the carriage return key repeatedly to proceed to the next line. Thus, the new version of the Editor also provides a simpler way to examine several consecutive line.

When you want to examine several lines in a row, you can first enter the " P n " command. (You can refer to section 6.3 Summary of the Editor Commands in the MPF-IP User's Manual on the " P n " command.) After the command " P n " is entered, you only have to press the carriage return key once to proceed to the next line. For example, if you intend to examine 10 lines in a row from the current line, you can enter " P 10 " and then press the carriage return key to proceed to the next line.

6.5.4 The Z Command-Print all The lines in the text buffer

The use of the Z command is as follows:

1) Use the R command to enter the edit mode. (Skip this step, if the MPF-IP is already in the edit mode.)
2) Enter the Z command. The MPF-IP should respond with

```
$Z
```

3) Press the carriage return key. The MPF-IP will print all the data currently in the text buffer.

6.6 Line pointer Manipulating Commands

Five of the line pointer manipulating commands allow a user to move the line pointer to a desired position and one enables a user to display the line number currently pointed to by the line pointer.

6.6.1 The B command-Move the cursor to the bottom of a file

The use of the command is as folows:

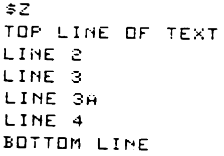
1) Type in B. The MPF-IP responds with

$B

2) Press the carriage return key. The MPF-IP will print the last line of the file currently in the text buffer and move the line pointer to that line.

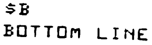
Example :

The data now in the text buffer is as follows:



```
$Z
TOP LINE OF TEXT
LINE 2
LINE 3
LINE 3A
LINE 4
BOTTOM LINE
```

Type the B command, the MPF-IP will print



```
$B
BOTTOM LINE
```

6.6.2 The G n command-Move the line pointer to the nth line of the file currently in the text buffer

The use of the "G n" command is depicted as follows:

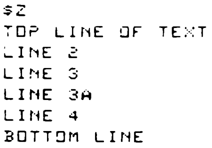
1) Enter the G n command. The MPF-IP will responed with

$G n

2) Press the carriage return key. The PRT-MPF will print the nth line of the file currently in the text buffer and move the line pointer to that line.

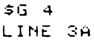
Example :

The following data is stored in the text buffer.



```
$Z
TOP LINE OF TEXT
LINE 2
LINE 3
LINE 3A
LINE 4
BOTTOM LINE
```

If a user intends to move the line pointer to the 4th line of the file, he can use the G 4 commad.



```
$G 4
LINE 3A
```

After the command has been executed by the MPF-IP, the line pointer points to the start of the 4th line.

6.6.3 The U command-The command to move the line pointer of line up.

the use of the command is as follows:

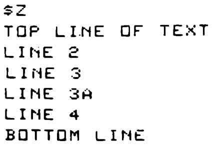
1) Enter the U n command. The MPF-IP will respond with

$U n

2) Press the carriage return key. The MPF-IP will print the line that is n lines up from the current line pointer to that line.
3) If the command line of the U command does not include the number of lines, the line pointer will only be moved up one line.

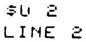
Example :

The data now in the text buffer is as follows.



```
$Z
TOP LINE OF TEXT
LINE 2
LINE 3
LINE 3A
LINE 4
BOTTOM LINE
```

In the example in 6.6.2, the line pointer has been positioned in the 4th line. If a user intends to move the line pointer to the second line, he can use the "U 2" command.



```
$U 2
LINE 2
```

6.6.4 The N n Command-The command that moves the line pointer n line down

The use of the command is listed as follows:

1) Enter the N n command. The MPF-IP will respond with

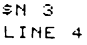
$N n

2) Press the carriage return key. The display and printer of the MPF-IP will print the line that is n lines down and move the line pointer to that line.

3) If the command line does not specify a number, then the command line will have a default value of Ø, e.g., the command line assumes that the numer "1" is specified.

Example :

The data in the text buffer is the same as that in the example in 6.6.3. In the above example, the line pointer was moved to the second line. If a user intends to move the line pointer to the fifth line, the command "N 3" should be used.



```
$N 3
LINE 4
```

6.6.5 The T Command-The command that moves the line pointer to the top of the file.

The use of the command is as follows:

1) Enter the T command. The MPF-IP will respond with

$T

2) Press the carriage return key. The MPF-IP will print the top of the file and move the line pointer there.

Example :

The data in the text buffer is the same as that in the previous example. In the above example, the line pointer has been moved to the fifth line. To move the line pointer to the top of the file, enter the T command by pressing T. The MPF-IP will respond.

```
$T
```

6.6.6 The L command-The command that prints the line number whish is now pointed to by the line pointer.

The use of the command is as follows:

1) Type in the L command. The MPF-II will respond

```
$L
```

2) Press the [←┘] key. The printer and display of the MPF-IP will print the value of the line pointer.

Example :

The data in the text buffer is as follows:

```
$Z
.TOP LINE OF TEXT
 LINE 2
 LINE 2A
 LINE 2B
 LINE 3
 LINE 3A
 LINE 4
 LINE 5
 BOTTOM LINE
```

Suppose that the user has applied the F command to move the line pointer to the line "LINE 3A"

```
 $F /3A/
LINE 3A
```

To find out the line number of the line "LINE 3A", enter the L command.

```
$L
  6
```

6.7 String Handling Commands

Two editor commands are used for string handling. The Find command allows a user to locate a specific string in the text buffer. The Change command enables a user to change the contents (characters) of a string.

6.7.1 The F Command-To locate a string

The use of the command is as follows:

1) Enter the F command. The MPF-IP will respond with

$F

2) Specify the string to be located by typing /string/, then type in the carriage return key. After the carriage return key is pressed, the MPF-IP will start searching for the specified string.

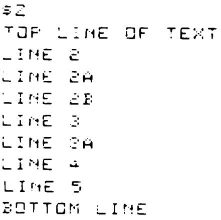
   If the string is located by the MPF-IP, the MPF-IP will print the line containing the string and move the line pointer to that line. If the MPF-IP can not find the string, it will print

   ?$

3) The specified string should be enclosed in certain delimiters such as /, ., *, -, =.

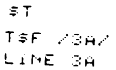
Example :

The data in the text buffer is as follows:



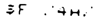
```
$Z
TOP LINE OF TEXT
LINE 2
LINE 2A
LINE 2B
LINE 3
LINE 3A
LINE 4
LINE 5
BOTTOM LINE
```

If a user intends to locate the line containing the string "3A", enter the F command as follows:



```
$T
T$F /3A/
LINE 3A
```

If a user intends to locate the line containing the string "4A", type as follows:

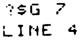


```
$F .'4A.'
```

Because these is no such a string containing "4A", the MPF-IP couldn't locate the string 4A. It will print

?$

If you first move the line pointer to the 7th line in the buffer, and then type in the F command to locate the string containing "3A", the MPF-IP still can not find the string containing 3A. That is because there is no such a string containing 3A after the 7th line of the file.



```
?$G 7
LINE 4
```

6.7.2 Thc C Command-To change a string

The C command is used to change a string in the active line. The use of the command is as follows:

1) Use the F, G, N, and U commands to move the line pointer to the line where a string is to be changed.
2) Enter the C command. The MPF-IP will respond with

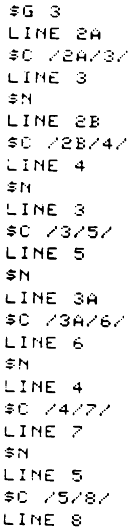
$C

3) Enter the string (which should be enclosed in delimiters), and then press the ⏎ key. If the user intends to change "INC A" to "DEC A", he should type /INC/DEC or *INC*DEC.
4) The PRT-MPF will print the corrected line.

Example :

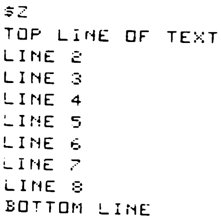
The data in the text buffer is as follows:

If the user intends to change the third line to "LINE 3", the fourth line to "LINE 4", and the fifth line to "LINE 5", and the sixth line to "LINE 6", and the seventh line to "LINE 7", and the eighth line to "LINE 8", use the G, N, and C commands as follows:



```
$G 3
LINE 2A
$C /2A/3/
LINE 3
$N
LINE 2B
$C /2B/4/
LINE 4
$N
LINE 3
$C /3/5/
LINE 5
$N
LINE 3A
$C /3A/6/
LINE 6
$N
LINE 4
$C /4/7/
LINE 7
$N
LINE 5
$C /5/8/
LINE 8
```

After all the corrections have been made, the data in the text buffer will be as follows:

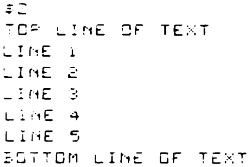


```
$Z
TOP LINE OF TEXT
LINE 2
LINE 3
LINE 4
LINE 5
LINE 6
LINE 7
LINE 8
BOTTOM LINE
```

6.8 Other Commands

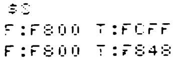
6.8.1 The S Command-Display the Default Values and the Current Text File

The data in the text buffer is as follows:



```
$Z
TOP LINE OF TEXT
LINE 1
LINE 2
LINE 3
LINE 4
LINE 5
BOTTOM LINE OF TEXT
```

Enter the S command causes the MPF-IP to print the default values of the text buffer and the upper and lower limits (starting and ending addresses) of the current file.



```
$S
F:F800 T:FCFF
F:F800 T:F848
```

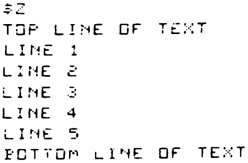
The above print-out shows that the lower limit of the text buffer is F8ØØ and the upper limit is FCFF, and the memory now being used to store the current file begins from F8ØØ to F848.

6.8.2 The X Command-Printer Control Command

When the MPF-IP is in edit mode, the X command functions as a toggle switch. It toggles on or off the printer.

6.8.3 The W Command-Write data from memory to tape

The file (whose filename is POIU) in the text buffer is as follows:



```
$Z
TOP LINE OF TEXT
LINE 1
LINE 2
LINE 3
LINE 4
LINE 5
BOTTOM LINE OF TEXT
```

To write data from memory to tape, first plug one end of the recorder line to the the MIC jack of the MPF-IP and the other end to the MIC jack of the cassette tape recorder. Put the tape recorder in record mode, and set the voice volume control switch properly. Then enter the W command following the command format below:

```
W  POIU  ↵
```

After the carriage return is pressed, the MPF-IP will write data from its RAM to cassette tape. The file is stored on the tape with the filename POIU. Note that while typing in the command line, a space should be entered between the W command and the filename.

6.8.4 The R Command-Read data from tape to memory

The R command is applied to read data from cassette tape to the RAM of the MPF-IP.

Plug the recorder line to the EAR jacks of the MPF-IP and the tape recorder properly before reading data from tape to the RAM of the MPF-IP. Rewind the cassette tape to the beginning. Enter the R command following the command format below.

```
R  POIU  ↵
```

Before pressing the carriage return key, put the recorder in play mode. After you have entered the R command and the filename and set the tape recorder to play mode, you can type the carriage return key. After the carriage return key is pressed, the MPF-IP begins reading data from tape to its RAM.

6.8.5 Error messages

1) When the MPF-IP is in edit mode, the ?$ represents that an incorrect command has been entered (For example, the MPF-IP will not accept such commands as Y or V.) and the MPF-IP is ready to accept a correct command.

2) When the MPF-IP is in input mode, if the input data has overflown the memory space specified, the MPF-IP will print *$, exit from input mode and enter edit mode.

Example :

When the text buffer is allocated the memory space starting from F800 through F803, entering the instruction "INC HL" will cause the MPF-IP display the following error mesage because that instruction line requires seven bytes to store -- The "INC HL" line requires one byte to store the code for TAB, another one byte to store the code for a SPACE, and five bytes to store the characters.

6.9 The [←] Key ---- Shift data on the display of MPF-IP

When you use Editor commands such as " P ", " N ", or " U "... to examine the source program, the number of characters the display of MPF-IP can display at a time is 20 characters. However, there may be a number of lines in a file which contain more than 20 characters, you can see those characters following the 20th character by using the [←] key.

Chapter 7
The Assembler and Disassembler

The resident assembler and disassembler of the MPF-IP, together with the editor, makes the MPF-IP a very powerful and unique microcomputer.

The major application of the assembler is to convert source program written in mnemonic form to binary code which can be understood by the computer. For example, the instruction "LD A,3" will be converted to "3E03" in hexadecimal or "0011 1110 0000 0011" in binary. The binary code generated after the convertion process is also known as machine code or object code. The conversion process carried out by the assember program is called assembly.

The disassembler is a program that converts binary machine code into mnemonic form assembly source program that is more readable than machine code. Strictly speaking, a disassembler disassembles machine code. Another useful application of disassembler is that it can be used to read the contents of an EPROM -- the program contained in an EPROM together with the PRT-MPF.

The MPF-IP assembler resides in an 8K EPROM that houses the monitor and editor programs, while the disassembler shares a 4K EPROM with the printer control program that controls the operations of the PRT-MPF.

The functions of both a two-pass assembler and a one-pass assembler (line assembler) are provided by the MPF-IP. Both the two-pass and one-pass assembler use a routine whose function is to convert mnemonic source program instructions to machine code.

When an assembly language source program is assembled by the MPF-IP two-pass assembler. Duromg pass one, the two-pass assembler will first fetch the labels in a source program to create a symbol table which contains the labels and their corresponding values. During pass two, the assembler will use the values provided by symbol table to generate the actual object code.

The one-pass assembler, however, does not accept symbols and labels. When a user applies the one-pass assembler to assembler source code to object code, he can only give absolute values as addresses and displacement. The greatest advantage of the line assembler is that it saves memory space. When a one-pass assembler is in use, source program is directly assembled to object code and stored in the memory. No memory space is required to store mnemonic source program.

The MPF-IP assembly language conventions are similar to Z8Ø assembly language conventions. Note the differences of MPF-IP assembly language conventions and that of Z8Ø's:

1) A comma "," should be inserted to separate operands.
2) A semicolon ";" should precede each comment.
3) The MPF-IP only accepts values in base 1Ø and base 16 number systems.

7.1 Two-Pass Assembler

7.1.1 The use of MPF-IP Two-Pass Assembler

a. To enter the two-pass assembler --
When the MPF-IP is in monitor mode, pressing A while holding down the CONTROL key allows a user to enter and initialize the two-pass assembler. The MPF-IP will respond with:

```
ORG : ^
```

b. To enter the starting address of the source program-
What the MPF-IP prints on the display prompts the user to enter the starting address of the source program. The starting address should be specified with a hexadecimal number. After the starting address is entered, you have to press the carriage return key. If the starting address of the source program is FAØØ, the MPF-IP will print

```
ORG : FA00
```

If no starting address is entered before you press the carriage return key, the MPF-IP will select a default value as the starting address of the source program. The default value for model with 2K RAM is FDØØ, and that for the model with 4K RAM is FBØØ. If the MPF-IP (with 4K RAM) assigns a default value to the source program, it will print:

```
ORG : FB00
```

c. Enter the starting address for the symbol table:
After the starting address of the source program has been decided, the MPF-IP will request the user to input the starting address for the symbol table by printing the following:

```
SYM >F:^
```

A hexadecimal address is to be entered, followed by a carriage return. If the user enters F8ØØ, the MPF-IP will respond with

```
SYM >F:F800 T:
```

d. Enter the ending address of the symbol table:
Type in the ending address of the symbl table and the carriage return key, the MPF-IP will respond with

```
SYM >F:F800 T:F9FF
```

If no starting address is assigned to the symbol table in step c., then default values will be assigned automatically as the starting and ending address by the MPF-IP. In model with 4K RAM, the default values are FDØØ and FEAØ. In model with 2K RAM, the default values are FEØØ and FEAØ. The PRT-MPF will print

```
SYM >F:FD00 T:FEA0
```

e. Enter the starting address for the object code: After the PRT-MPF has printed the starting and ending addresses for the symbol table, the MPF-IP will prompt a user to enter the starting address for the object code:

```
OBJ >F: ^
```

When being prompted, the user may type in the starting address of the memory which is to be assigned to store the object code. Usually, the address is the same as the starting address of the source program. Then type in the carriage return key. If the address FAØØ is entered, the MPF-IP will respond with

```
OBJ >F:FA00 T:
```

f. Enter the ending address for the object code, then press the <--' key. If FBFF is entered by the user as the ending address, the PRT-MPF will respond with

```
OBJ >F:FA00 T:FBFF
```

If no starting address is entered in step e. before entering the carriage return, the MPF-IP will assign two default values as the starting and ending addresses for the memory where the object code generated from the assembly process will be stored. The default values for MPF-IP model with 2K RAM are FDØØ and FDFF, and that for model with 4K RAM are FBØØ and FCFF. When operating on 4K RAM model, the PRT-MPF will print

```
OBJ >F:FB00 T:FCFF
```

g. After the carriage return is pressed in step f., the MPF-IP will start fetching data from the beginning of the text buffer and assembling the data into machine code. If error occurs during the assembly process, the MPF-IP will stop the assembly process and print the error messages. Refer Section 7.3 for error messages.

7.1.2 Assembly Language Pseudo-Ops

In addition to the executable instructions, assembly language uses pseudo opcodes in a source program to facilitate the generation of object code during the assembly process. The pseudo-ops are applied in a program the same way as an opcode is used in a program. The only difference between the pseudo-op and opcode is that the opcode performs a specific operation when executed, while the pseudo-op does not. The use of pseudo-ops are descriped as follows:

1) Data Defination

1. DEFB -- Define Byte

The function of this pseudo-op is to store an 8-bit operand into the memory location pointed to by the current value of the reference counter. The reference counter is used as a pointer to the location in memory and corresponds to the program counter. The format of a pseudo-op line is as follows:

```
Label   Opcode    Operand     Comment

XXX:    DEFB      expression  ;YYY
```

If a label is used in a pseudo-op instruction, then the value of the label is assigned with the value of the reference counter and is the address of the data. You can refer to the MPF-IP Monitor Program Source Listing for the use of the DEFB pseudo-op.

2. DEFW -- Define Word

The function of this pseudo-op is to store a 16-bit operand into the two consecutive memory locations pointed to by the current value of the reference counter. The reference counter is used as a pointer to the location in memory and corresponds to the program counter. The format of a DEFW pseudo-op line is as follows:

```
Label   Opcode    Operand     Comment
```

```
XXX:    DEFW      expression  ;YYY
```

The low order byte of the operand is to be stored in the memory location pointed to by the current value of the reference counter, while the high order byte of the operand is to be placed into the next higher memory location.

3. DEFM -- Define Message

The operand of this pseudo-op is a string of characters enclosed in two single quotation marks. The function of the DEFS pseudo-op is to store the ASCII code for each character

2) DEFS -- Define Storage

During the execution of a program, a certain number of bytes may be reserved to store the results of the executed instructions. The pseuds-op DEFS reserves a memory space which starts from the location pointed to by the current value of the reference counter. The length of the memory space is specified by the operand of the pseudo-op. The format of a DEFS pseudo-op line is as follows:

```
Label      Opcode     Operand      Comment

BUFFER     DEFS       128        ; Define a storage of
                                   128 bytes
```

The example above defines a memory space of 128 bytes.

3) Program Termination -- END

Any source program should be ended with the END pseudo-op. Thus, any subsequent instructions following the END pseudo-op is ignored. If an assembly language program is not ended with the END pseudo-op, errors may occur.

4) The pseudo-op to assign a value:

EQU -- Equate

The EQU pseudo-op assigns the value of the operand to a label. The vaue is a 16-bit hexadecimal number. Only one value can be assigned to a symbol (label) in a program. If two values are assigned to the same symbol, errors will occur. The EQU pseudo-op is used in the followng way:

```
Label     Opcode       Operand          Comment
PWCODE     EQU         ØA5H          ;Power up code.
P82551     EQU         83H           ;8255 I control port
```

5) Reference Counter Control -- ORG

The assembler uses a reference counter to count the memory locations which will be stored with the assembled machine code of the program being assembled. After an instruction has been assembled by the assembler, the number of bytes it takes to store the machine code of the instruction is added to the value of the reference counter. Thus, the current value of the reference counter always corresponds to the memory location into which the object code of the next instruction is to be stored. When a program is to be stored into the memory, starting from a specific address, the ORG pseudo-op is used to set the value of the reference counter to that specific address.

6) LABEL --

A label may consist of up to six alphanumeric characters. The first character of a label must be a letter of the alphabet.

7) The Summary of Pseudo-ops.

```
1 DEFB ØF8H          ; Define byte. ( low byte value )
2 DEFW ØF786H        ; Define word. (value)
3 DEFM 'AAAA'        : Define message.
4 DEFS n             ; Define storage. (CONST)
5 ORG ØF850H         ; Origin. (CONST)
6 EQU ØF850H         ; EQU. (CONST)
7 END                ; end of assembler.
8 ;                  ; Comments.
```

7.1.3 Examples of the Use of the Pseudo-op

The following examples may help the reader to further understand the use of the pseudo-ops. The following monitor subroutines will be used in the examples:

CLEAR: The function of the subroutine is to clear the contents of the display buffer, i.e., to store FF into the display buffer.

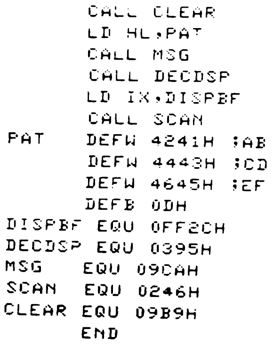
MSG : The MSG subroutine converts the ASCII code pointed to by the HL register pair to display pattern and then store the display pattern into display buffer until the "ØD" code is encountered.

SCAN : The subroutine displays a sequence of characters (The starting address is (IX), and the display pattern is stored from that address through the next 4Ø bytes) until a key is pressed.

DISPBF:The subroutine displays the starting address of the display buffer.

Example 1:

The following program, when executed, displays the six characters -- A, B, C, D, E, F, -- until a key is pressed.



```
             CALL CLEAR
             LD HL,PAT
             CALL MSG
             CALL DECDSP
             LD IX,DISPBF
             CALL SCAN
    PAT      DEFW 4241H ;AB
             DEFW 4443H ;CD
             DEFW 4645H ;EF
             DEFB 0DH
    DISPBF EQU 0FF2CH
    DECDSP EQU 0395H
    MSG    EQU 09CAH
    SCAN   EQU 0246H
    CLEAR  EQU 09B9H
           END
```

Because this is a simple program, it only takes a limited memory space to store the source program and the object code. When a two-pass assembler is in use, there is no need to change the default values. If the user only wants to see the results of the program, he/she may turns off the PRT-MPF-IP, uses the assembler to convert the source program into object code, and then press GO FBØØ (4K RAM) or GO FDØØ (2K RAM) to execute the program.

Example 2:

The following example program, when being executed, displays the characters "ABCDEFWELCOME" until a key is pressed.

```
        CALL CLEAR
        LD HL,PAT
        CALL MSG
        CALL DECDSP
        LD IX,DISPBF
        CALL SCAN
PAT     DEFW 4241H ;AB
        DEFW 4443H ;CD
        DEFW 4645H ;EF
        DEFM 'WELCOME'
        DEFB 0DH
DISPBF EQU 0FF2CH
DECDSP EQU 0395H
MSG    EQU 09CAH
SCAN   EQU 0246H
CLEAR  EQU 09B9H
       END
```

Example 3:

The example program has the same function as the program in Example 2. However, this program does not simply display the display patterns stored in the display buffer. Instead, the program moves the contents of the display buffer to a working storage area, and then displays the contents of the working storage.

```
        CALL CLEAR
        LD HL,PAT
        CALL MSG
        CALL DECDSP
        LD HL,DISPBF
        LD DE,BUFFER
        LD BC,40
        LDIR
        LD IX,BUFFER
        CALL SCAN
PAT     DEFW 4241H ;AB
        DEFW 4443H ;CD
        DEFW 4645H ;EF
        DEFM 'WELCOME'
        DEFB 0DH
BUFFER EQU 0F9E0H
DISPBF EQU 0FF2CH
DECDSP EQU 0395H
MSG    EQU 09CAH
SCAN   EQU 0246H
CLEAR  EQU 09B9H
        END
```

7.2 Line Assembler (One-Pass Assembler)

The use of the line assemlber is inconvenient to a user. However, the advantage of the line assembler is that when a line assembler is in use, no memory space is required to store source program. When a user's source program is very long, the use of line assembler saves user's RAM space.

After an instruction line is entered from the keyboard, the line assembler immediately assembles the source code into object code. Because no symbol table is created when a line assembler is in use, absolute values should be given to labels or symbols.

7.2.1 The Use of the Line Assembler

1. Enter the line assembler by pressing L while holding down the CONTROL key. The MPF-IP should show:

```
ORG : ^
```

2. Enter a value for the reference counter, and then press the ↵ key. For example, if the user types in F8ØØ, the PRT-MPF-IP would respond with

```
ORG : F800
```

while the display should show

```
OBJ >F: ^
```

3. Enter the starting address of the memory space to be used to store the object code, then press the ↵. For example, if the user types in FCØØ, the PRT-MPF-IP will print

```
OBJ >FC00
INPUT
```

while the display of the MPF-IP shows

```
F800
```

4. If the user does not enter the starting address of the object code, then default values will be set automatically by the MPF-IP. Model with 4K RAM sets the default vaue to FØØØ (the object code will then be stored beginning from FØØØ.), while model with 2K RAM sets the default value to F8ØØ.

5. As soon as the starting address (or the value of the

reference counter) is shown on the display, you can begin entering the instructions. An instruction line is separated with another instruction line by the carriage return. As soon as the carriage reutrn is pressed, the line assembler assembles one instruction line from source code to object code immediately.

6. Pressing the carriage return key twice returns the control to the monitor.

7. When error occurs, the MPF-IP prints "?" and returns the control to the monitor.

8. An absolute value should be entered as the operand for a relative jump instruction.

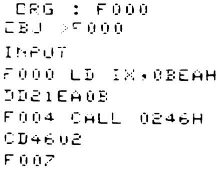
7.2.2 The Method For Calculating Displacement for Relative Jumps

1. Use the JR $±N instruction.

2. Use the J monitor command.
   When the programmer uses a relative jump instruction without knowing the exact displacement, he can use a random number or zero as the operand for the relative jump instruction and enter the exact displacement as the operand until the exact displacement is calculated correctly. The following examples shows the use of the line assembler.

Example 1:

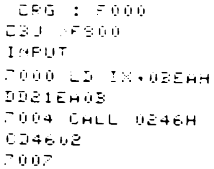
The following example program displays the alphabetical letters from A to T. FØØØ is assigned as the starting address for the program, while the object code of the program is also stored beginning from FØØØ.



```
 ORG : F000
OBJ >F000
INPUT
F000 LD IX,0BEAH
DD21EA0B
F004 CALL 0246H
CD4602
F007
```

Examples 2:

The example program also performs the same task as the program in Example 1. However, 7000 is assgned as the starting address of the program, while the object code is stored beginning from F800.



```
 ORG : 7000
OBJ :F800
INPUT
7000 LD IX,02EAH
DD21EA03
7004 CALL 0246H
CD4602
7007
```

Though the address 7000 is not in the RAM of the MPF-IP, the assigning of 7000 as th starting address for a program is significant considering the fact that the IC memory to be inserted on the board location U6 (of PRT-MPF-IP) is assigned the addresses from 7000 to 7FFF. If a programmer intends to write data to an EPROM to be inserted to U4 of the MPFIP, he should use the skill's examplified in this example to set the starting address of the program to 7000, store the assembled object code in the RAM, and then write the data (assembled object code) from RAM of the MPF-IP to EPROM.

7.3 Error Messages

7.3.1 Errors Resulted from the Use of Assembler

1. 'OBJECT OVER':
   When the object code resulted from an assembly process requires more memory space than originally set upon entering into the assembler, the MPF-IP will print 'OBJECT OVER' after pass one. Press the "Q" key returns the control to the monitor.

2. 'SYMBOL OVER':
   When the symbol table takes more memory space than originally set upon entering into the assembler, the MPF-IP will print 'SYMBOL OVER'. Pressing the "Q" key allows the monitor to regain control.

7.3.2 Errors Resulted from Mistakes in the Assembly Language Instructions

I ILLEGAL INSTRUCTION

U UNDEFINED SYMBOL

E EXPRESSION OUT OF RANGE

D DUPLICATED SYMBOL

L ILLEGAL LABEL

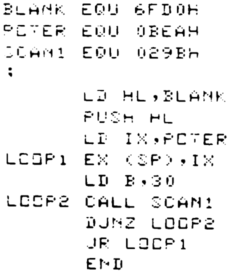
Q QUOT EXPECTED

C CONSTANT EXPECTED

(In this case, the operand of ORG or EQU sould be preceded with a leading zero.)

Example

Given the program below is to flash the 20 alphabetical letters from A to T:



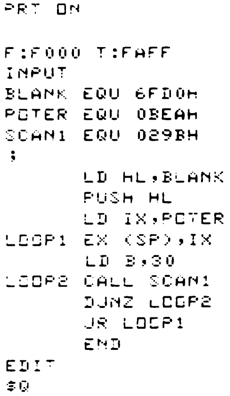
```
BLANK  EQU 6FD0H
PETER  EQU 0BEAH
SCAN1  EQU 029Bh
;
       LD HL,BLANK
       PUSH HL
       LD IX,PETER
LOOP1  EX (SP),IX
       LD B,30
LOOP2  CALL SCAN1
       DJNZ LOOP2
       JR LOOP1
       END
```

SCAN 1: The subroutine to scan the keyboard and display characters for one cycle.

POTER : The starting address of the buffer in which the display pattern is stored.

BLANK : The starting address of the buffer in which the display pattern for "blank" is stored.

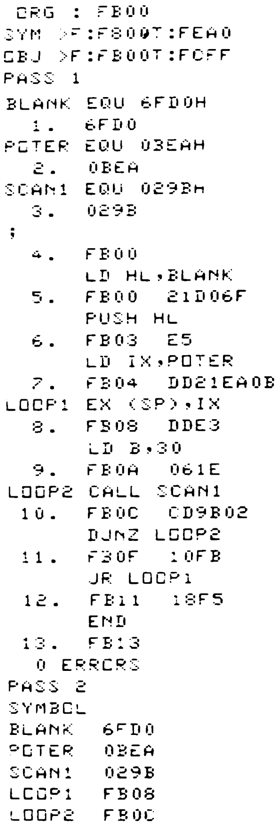
1. Enter the input mode of the text editor, and type in the program with default values unchanged:



```
PRT ON

F:F000 T:FAFF
INPUT
BLANK EQU 6FD0H
POTER EQU 0BEAH
SCAN1 EQU 029BH
;
       LD HL,BLANK
       PUSH HL
       LD IX,POTER
LOOP1  EX (SP),IX
       LD B,30
LOOP2  CALL SCAN1
       DJNZ LOOP2
       JR LOOP1
       END
EDIT
$Q
```

2. After the program has been entered, use the assembler to assemble the source program to machine code with the default values unchanged:



```
 ORG : FB00
SYM >F:F800T:FEA0
OBJ >F:FB00T:FCFF
PASS 1
BLANK EQU 6FD0H
  1.   6FD0
POTER EQU 03EAH
  2.   0BEA
SCAN1 EQU 029BH
  3.   029B
;
  4.   FB00
       LD HL,BLANK
  5.   FB00  21D06F
       PUSH HL
  6.   FB03  E5
       LD IX,POTER
  7.   FB04  DD21EA0B
LOOP1 EX (SP),IX
  8.   FB08  DDE3
       LD B,30
  9.   FB0A  061E
LOOP2 CALL SCAN1
 10.   FB0C  CD9B02
       DJNZ LOOP2
 11.   FB0F  10FB
       JR LOOP1
 12.   FB11  18F5
       END
 13.   FB13
  0 ERRORS
PASS 2
SYMBOL
BLANK  6FD0
POTER  0BEA
SCAN1  029B
LOOP1  FB08
LOOP2  FB0C
```

The assembled object code of the above program is stored in the memory starting from FBØØ. As soon as you press <G> FBØØ [⟵], the 2Ø alphabetical letters will flash on the display.

3. If the user adds the pseudo-op ORG F9ØØH in the beginning of the source program, the new program will be as follows:

```
BLANK  EQU 6FD0H
PCTER  EQU 0BEAH
SCAN1  EQU 029BH
;
       ORG 0F900H
       LD HL,BLANK
       PUSH HL
       LD IX,PCTER
LOOP1  EX (SP),IX
       LD B,30
LOOP2  CALL SCAN1
       DJNZ LOOP2
       JR LOOP1
       END
```

Example: The carriage eturn key ↵ may be used to shift the display to the left. Each time the ↵ key is pressed, the display is shifted left one character. For example, when the following error (error D) occurs, the characte "D" is not displayed on the display. Any afte the ↵ key is pessed, "D" will be displayed.

```
F:F000 T:FAFF
INPUT
      ORG F000H
LOOP  LD A,B
LOOP JP LOOP
EDIT
$0
 ORG : FB00
SYM >F:FD00 T:FEA0
OBJ >F:FB00 T:FCFF
PASS 1
```

```
      ORG F000H
      0000   *C*
LOOP  LD A,B
  2.  0000  78
LOOP JP LOOP
  3.  0001  C30000 *
D*
  2 ERRORS
PASS 2
SYMBOL
LOOP   0000
```

Example: When using line assembler, the effect of JR XXXX is the same as JR + - n.

```
 ORG : F000
OBJ >F000
INPUT
F000 JR 0F034H
1832
F002 JR 0EFF8H
18F4
```

```
 ORG : F000
OBJ  F000
INPUT
F000 JR $+3-
1832
F002 JR $-0AH
18F4
```

4. After the source program is assembled, it looks like:

```
 ORG : FB00
SYM >F:F800T:FEA0
OBJ >F:F900T:F9FF

PASS 1
BLANK EQU 6FD0H
  1.   6FD0
PCTER EQU 0BEAH
  2.   0BEA
SCAN1 EQU 029BH
  3.   029B
;
  4.   FB00
       ORG 0F900H
  5.   F900
       LD HL,BLANK
  6.   F900  21D06F
       PUSH HL
  7.   F903  E5
       LD IX,PCTER
  8.   F904  DD21EA0B
```

```
LOOP1 EX (SP),IX
  9.   F908  DDE3
       LD B,30
 10.   F90A  061E
LOOP2 CALL SCAN1
 11.   F90C  CD9B02
       DJNZ LOOP2
 12.   F90F  10FB
       JR LOOP1
 13.   F911  18F5
       END
 14.   F913
  0 ERRORS
PASS 2
SYMBOL
BLANK   6FD0
PCTER   0BEA
SCAN1   029B
LOOP1   F908
LOOP2   F90C
```

Though the starting address of the source program and the memory space to store object code have been set upon entry into the assembler, the ORG pseudo-op in the program changed the starting address to F9ØØ while the object machine code is stored into the memory space starting from FBØØ.

5. If the user sets the RAM area for object code and symbol table improperly, the following errors will happen:

   a. The RAM area for symbol table is too small:

```
 ORG : FB00
SYM >F:FD00T:FD05
OBJ >F:FB00T:FCFF
SYM OVER
```

   b. The RAM area for the object code is too small:

```
 ORG : F900
SYM >F:FD00T:FEB0
OBJ >F:F900T:F90A
OBJ OVER
```

7.4 Disassembler

The major function of the disassembler is to convert object code to mnemonic source program for debugging purpose. The disassembler of the MPF-IP resides in the same IC that holds the monitor program for the PRT-MPF-IP. Thus, if an MPF-IP is not connected with the PRT-MPF-IP, it is impossible to enter the disassembler. The use of the disassembler is as follows:

1. Use the monitor command D to enter disassembler by typing [CONTROL] [D] . The display should show <D> =

2. Enter the starting address of the object code. To use line disassembler, press the carriage return key directly. Each time the [↵] key is pressed, the line disassembler converts one line of object code to source code. If an instruction line has more than 20 characters, you can press the [◄] key to shift left the display. Each time the [◄] is pressed, the display is shifted one character left.

3. Press the space key [⎵], then enter the ending address of the object code.

4. Then follow step (1) or (2):

   (1) Press the [↵] key:
   This will directly converts the object code specified by the starting and ending addresses into mnemonic source code.

   (2) Press the space key, enter the linking address, then press the [↵] key.
   This will convert the object code between the starting address and the ending address into source code and assign the linking address as the starting address of the source code.

5. Press the [↵] key, then PRT-MPF-IP will print out the disassembled source code.

The following examples disassemble machine code contained in the first 16 bytes of ROM to source code.

Example 1:

```
<D>=0 10

0000 01 LD BC,0300
0003 ED CPD
0005 EA JR PE,0003
0008 3E LD A,88
000A D3 OUT (83),A
000C 3E LD A,81
000E D3 OUT (93),A
```

Example 2:

```
<D>=0 10 6000

6000 01 LD BC,0300
6003 ED CPD
6005 EA JP PE,0003
6008 3E LD A,88
600A D3 OUT (83),A
600C 3E LD A,81
600E D3 OUT (93),A
```

Example 3:

```
<D>=0

0000 01 LD BC,0300
0003 ED CPD
0005 EA JR PE,0003
0008 3E LD A,88
000A D3 OUT (83),A
000C 3E LD A,81
000E D3 OUT (93),A
```

The linking address allows a programmer to disassemble the object code from an EPROM correctly. For example, if a programmer intends to disassemble the machine code of an EPROM in which a program is stored with the starting address of 2000H, the programmer can insert the EPROM into the socket at board location U6 on the PRT-MPF-IP, initialize the disassembler and enter 7000 and 7FFF as the starting and ending addresses and 2000 as the linking address, then the object code in the EPROM will be disassembled correctly to the source code.

7.5 Summary of Text Editor and Assembler Parameters

The monitor program of the MPF-IP antomatically sets the default values concerning memory usage if a user does not specify the starting and ending addresses of the memory space used for storing text buffer, source code, or object. The default values are as follows:

(1) For use with line assembler

ORG : F000 (4K RAM)
      F800 (2K RAM)

OBJECT: F000 (4K RAM)
        F800 (2K RAM)

For use with two-pass assembler

ORG   FB00 (4K RAM)
           FD00 (2K RAM)
SYMBOL :   FD00-FEA0 (4K RAM)
           FE00-FEA0 (2K RAM)
OBJECT :   FB00-FCFF (4K RAM)
           FD00-FDFF (2K RAM)

For use with text editor

ORG   F000 (4K RAM)
           F800 (2K RAM)

(2) The default values for text editor, two-pass assembler, and line assembler are so assigned because the MPF-IP (the model with 4K RAM) assigns the memory space from F000 to FAFF for storing program or data when the MPF-IP is in the text editor input mode, FB00 to FCFF for storing the object code of the source program, and FD00 to FEA0 for storing the symbol table.

For the model with 2K RAM, the RAM area from F800 to FCFF is used as the text buffer for storing program or data entered in the text editor's input mode, FD00 to FDFF for storing the object code, and FE00 to FEA0 for storing the symbol table.

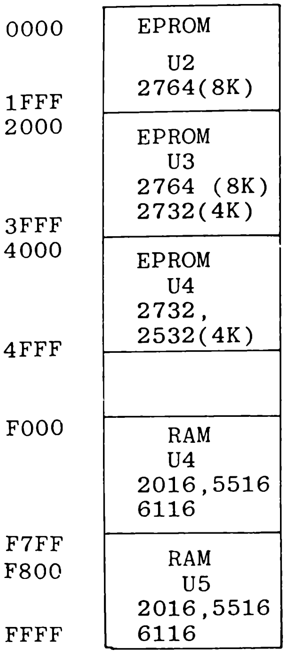
(3) Because the default values were assigned with proper usage of RAM space in mind, it may not be necessary to change the default values when a programmer has no special requirements for memory space allocation.

(4) If entering a simple, short program, the user may use the F or M monitor commands to enter the object code of a source proqram directly into the memory.

Chapter 8
System Hardware Configuration

8.1 System Memory Organization

The memory map of the MPF-IP is as follows:



1. U2:
   On board location U2, as 8K EPROM (0000 through 1FFF) is inserted -- the monitor.

2. U3:
   Either an 8K EPROM or a 4K EPROM may be inserted into U3. If an 8K EPROM (2764) is installed, the memory on the 8K EPROM ranges from 2000 to 3FFF. If a 4k EPROM (2732) is installed, the memory on the 4K EPROM ranges from 2000 to 2FFF. Because the socket at U3 accepts 28 pins, while the 2732 has 24 pins, the top four pin holes of the socket at U3 are left empty when a 2732 is to be installed.

3. U4:
   Either RAM or EPROM may be inserted into U4. If a RAM is inserted, the addresses of the RAM range from F000 to F7FF. Either 2016, 5516 or 6116 may be used as the RAM inserted in U4. When your MPF-IP has battery back-up, 6116 or 5516 is suggested because they consume less electricity than 2016. Your MPF-IP may be installed with either a 2016 or 6116. If a user intends to use 5516 on U4, then he should jump and cut several wires at board location J2. (The J2 area is located near the top edge of the U4.)

   If an EPROM is installed on U4, the addresses of the EPROM range from 4000 to 47FF. Either 2732 or 2532 may be installed on U4. However, several wires should be re-routed at J2.

4. U5:
   A RAM (F800 to FFFF) is installed here. The system RAM used by the monitor resides in this RAM. Either 2016, 5516 or 6116 may be installed on U5.

5. After J2 has been re-routed to allow a 2732 to be installed on U4, a 2716 may also be used on U4 without changing the routing at J2.

   After J2 has been re-routed to allow a 2532 to be installed on U4, either 2716 or 2516 may also be used on U4 without modification at J2.

6. When different RAMs or EPROM are to be installed on U4 or U5, several wires should be re-routed at the J2 area. The re-routing at the J2 area is shown as follows (<-X-> represents that the wire should be cut, while <--> represents that wires should be jumped.)

   (1) When 5516 is to be used on U4 and U5:

Wire Cutting	Wire Jumping
3 ↮ 5	1 ↔ 5
10 ↮ 11	3 ↔ 4
1 ↮ 4	5 ↔ 10
4 ↮ 9	9 ↔ 11

(2) When 2732 is to be used on U4:

Wire Cutting	Wire Jumping
3 ↮ 5	2 ↔ 5
6 ↮ 8	7 ↔ 8

(3) When 2532 is to be used on U4:

Wire Cutting	Wire Jumping
1 ↮ 4	5 ↔ 7
4 ↮ 9	2 ↔ 4
3 ↮ 5	8 ↔ Vcc
6 ↮ 8	1 ↔ 9

Note: When an EPROM is inserted on U4 and the RAM on U5 is to be connected to battery back-up, the user must first disconnect the VCC line between U4 and U5 and then connect the VCC line for U4 to the VCC line of any other IC on the circuit board. (Refer to sheet 6 of the schematic.)

8.2 Input/Output Addresses

The input/output adresses of the MPF-IP are as follows:

80	A	8255-1 U14
81	B	
82	C	
83	CONTROL	
90	A	8255-2 U13
91	B	
92	C	
93	CONTROL	

1. The 8255 is a programmable 4Ø-pin large scale integrated circuit with three 8-bit ports -- A, B, C. The three ports have 24 parallel input/output lines. The functions of the 8255 are programmable.

2. The functions of the I/O lines of the two 8255s on the MPF-IP are defined by the MPF-IP monitor and hardware configuration as follows:

   8255-I

   (1) Port A: PAØ through PA7 are output lines used to select digit 1 through digit 8.
   (2) Port B: PBØ through PB7, output lines, selects digit 9 through digit 16.
   (3) Port C: PCØ through PC3, output lines, selects digit 17 through digit 2Ø. PC4 is the input line for the SHIFT key, PC5 is the input line for the CONTROL key, while PC6 and PC7 are not used.

   8255-II

   (1) Port A: PAØ through PA7, output lines, select segment A through H of the 14-segment display.
   (2) Port B: PBØ through PB6, output lines, select segment I through dp, while PB7 is not used.
   (3) Port C: PCØ and PC2 are the input lines from the keyboard. PC3 is the input line from audio

tape recorder. PC4 is used by the monitor to handle single-step and break-point functions. The bit is usually one. The user must not send zero to this bit at will. PC5 is the output line to tape, and is also connected to the speaker and the TONE-OUT green LED lamp. This bit is used when the MPF-IP beeps or writes to tape. This bit is active low. PC6 and PC7 are not used.

8.3 Interrupt

Non-maskable interrupt can only be enabled by the monitor prgram, and cannot be enabled or disabled by the programmer.

PC6 is normally high. When a high (or one) is sent to the counter at U9, the counter, 74LS9Ø, is reset and will remain inactive. When the MPF-IP single-steps a program or the CPU reaches a break point, a zero or low is sent out from PC4 to U9, causing the counter start counting. During the first four machine cycles generated by the counter, the CPU saves all user's registers and status and checks the validity of usr's stack. Then during the fifth machine cycle, QA becomes high, and the program counter points to the instruction to which the break point is set. The high signal is inverted at U1Ø, and activates the $\overline{NMI}$ ($\overline{NMI}$ is active low.) This will interrupt program execution and jump back to monitor program.

The following is the logic state of U9 (74LS9Ø).

	R_9	R_0	Q_A	Q_D	Q_C	Q_B	$\overline{NMI}$	Remark
Normal State	0	1	0	0	0	0	1	U9 is reset to Ø.
BREAK becomes low	0	0	0	0	0	0	1	$R_0 = \overline{Break} = 0$ starts . counting
1st M1	0	0	0	0	0	1	1	Q_D › Q_C › Q_B is Mod 5 Counter
2nd M1	0	0	0	0	1	0	1	when Q_D from 1 → 0 & Q_A
3rd M1	0	0	0	0	1	1	1	from Ø → 1 °
4th M1	0	0	0	1	0	0	1	
5th M1	0	0	1	0	0	0	0	

8.4 Stack

Fig. 8-1 shows the stack configuration. The default value of the system stack pointer is FEDØ, while that of the user's stack pointer is FEAØ. The monitor keeps checking the value of the stack pointer. Once the monitor discovers that the user's stack pointer points to a location in the system stack, the error message SYS-SP will be displayed. If there is a stack-related instruction (e.g. RET) in the user's program, an error may occur when user's stack and system stack overlap.

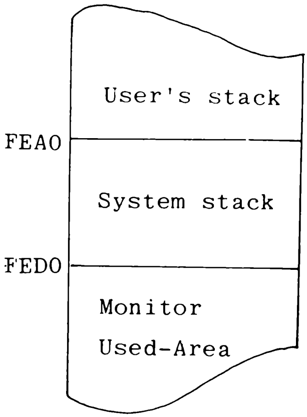


Fig. 8-1 Stack Configuration

8.5 Reset

The MPF-IP performs two types of RESET -- "cold" reset (power-on reset) and "warm" reset.

8.5.1 Power-on RESET

The reset cycle performed immediately after powering on the MPF-IP is referred to as a cold reset. The MPF-IP will perform the following in a power-on reset cycle.

(1) Disable interrupt (IFF set to Ø);
(2) I register set to Ø;
(3) Interrupt mode set to Ø;
(4) User's SP is set to FEAØ
(5) Reset 1FFF as default break-point;
(6) Reset the default values for the text editor and assembler;
(7) Reset the upper limit to FEØØ for the DELETE and INSERT editor command;
(8) Turn on the PRT-MPF-IP and reset the value of RST 38H.
(9) Display ***** MPF-I-PULS ***** character by character.

8.5.2 Warm RESET

When the RESET key is pressed, the MPF-IP performs the same first four functions as in 8.5.1 The display ***** MPF-I-PLUS ***** shows up on the display at the same time. However, the parameters, which are reset in a cold reset in steps 5 through 7, are not changed in a warm reset.

8.6 Tape Data Format

8.6.1 Bit Format

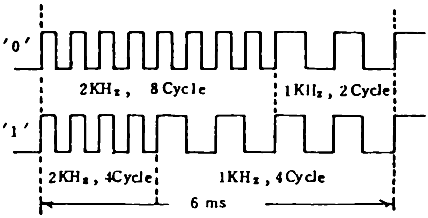


Fig. 8-2

8.6.2 Byte Format

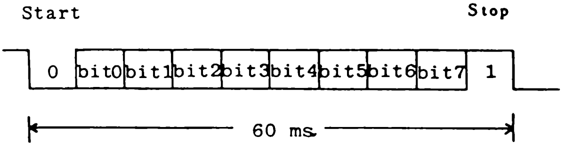


Fig. 8-3

8.6.3 File Format

Lead syne	File name	Start addr	End addr	Chk sum	Editor & Assembler	Mid sync	Data	Tail sync
1KHz 4sec	4 Byte	2 Byte	2 Byte	1 Byte	18 Byte	2KHz 2sec	Variable Length	2KHz 2sec

Fig. 8-4

8.6.4 Audio Cassette Tape

1. Labeling your cassettes: Make it a good habit to record the filenames, comments and remarks, and the starting and the ending positions of the tape counter.

2. When writing data to tape, make sure that the tape onto which data is to be stored is blank.
3. After data has been stored on tape, you should load the data or program which has just been stored on tape to the MPF-IP to examine if the program or data is stored correctly. If it is, you can turn off the power to the MPF-IP.

8.7 System Clock

A crystal oscillator, which generates square wave at 3.579 MHz, is used to generate clock pulse for controlling transfer of data in the CPU. The output of the crystal oscillator is connected to pin 3 of 74LS74, the D-type flip-flop, which divides the output of the crystal oscillator by two. The output of the 74LS74, clock pulse at 1.79MHz, is used as the system clock pulse.

8.8 Reset

When the RESET key is pressed, the flip-flop (74LS74) at U11 generates two shaping wave -- RST and RST (Refer to the schematic sheet 1 and 2). The RST is sent to the CPU and the RST is sent to the 8255 to start a warm reset cycle. Because of the functioning of RA1 and C17 which are connected to 74LS74, the MPF-IP will perform a cold (power on) reset cycle when power is supplied to the MPF-IP.

8.9 Audio Tape Inteface

The audio output is output from the PC5 of 8255-II. It is output after being filtered and attenuated through C5, R6, R7, R8. The audio output is also sent to the built-in speaker and green LED through Q2. Thus, the PC5 of 8255-II not only provides audio tape interface but also controls sound output.

Data stored on tape is read into the MPF-IP through C7, R9, R1Ø, D1, D2, U8 and U1Ø to the PC3 of 8255-II under the control of the software.

8.10 The Display and Keyboard

The display of the MPF-IP is a fluorescent indicator panel (FIP), featuring low power consumption, low voltage operation, clear, bright light output, and compatibility with MOS LSI.

8.10.1 Principle of Operation

FIPs Utilize the principle of directly heated triodes, composed of hot cathode (Filament), control Grid, and Anode. Electrons emitted by the hot cathode are accelerated through the electrical field by the application of positive signal potential to the control Grid and Anode. The electrons impact the fluorescent material on the Anode, exciting it to luminesce.

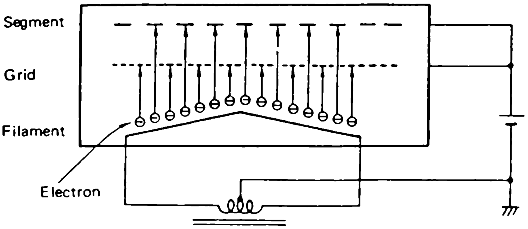


Fig. 8-5

In Fig. 8-5, the filament is the cathode, and the segment is the anode.

8.10.2 The Driving Modes

As to driving method of FIP, both dynamic and static modes are available, and they are related to the construction of FIP. Electrode connections of Anode segments are shown in Fig. 8-7 Features of both dynamic and static mode are summarized as follows.

(1) Static Driving Mode

i) Only one common Grid covers all digits and always supplied with positive voltage.

ii) Selection of display position, display pattern (Numerals, Characters, Symbols) are decided by segment signal. (Electrode terminals of each segment are independently drawn out.)

iii) Segment selection time is arbitrary.

iv) This driving method is suitable for FIPs which display comparatively few digits. (Number of electrode terminals increase repidly in accordance with the increase of display digits.)

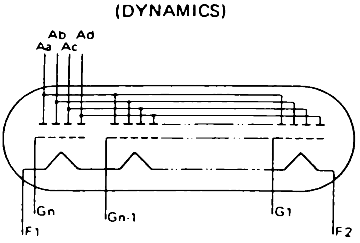


Fig. 8-6

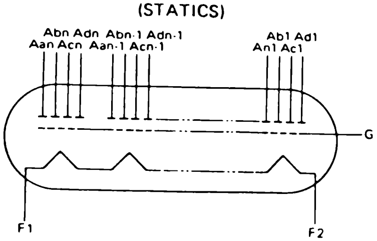


Fig. 8-7

(2) Dynamic Driving Mode

i) Grids are divided in each digit, and electrode terminals of individual Grids are drawn out.

ii) Segments of each digit (Grid) are parellel connected, so that total number of segment electrodes per one panel is equivalent to those of one digit.

iii) Segment selection signal must be supplied in timing with digit (Grid) singal to be lighted.

iv) This driving method is suitable for FIPs which must display comparatively many digits. (Total driving circuit cost is cheaper.)

8.10.3 FID Buffer Driver

The Fluorescent Indicator Panel (FIP) is an excellent display device, easy to use, low operating voltage, low power consumption and provides good matching with MOS LSIs and u-COMs. However some FIPs require high voltage and current due to increase in size of the panel and number of digits. In order to drive these FIPs, the interface circuits between FIPs and logic are indispensable. The description covers the fundamental ideas of interface circuits for FIPs.

At present, engineering studies with the object of making FIPs operate at still lower voltages and lower power consumption are being continued. The demands of increse in size of the panel and the number of digits tend to require an increase in the driving voltage or current. To drive these FIPs, the voltage and current capacity of the driving circuit become problem. Maximum operating voltage of LSIs is in most cases up to 40 V and it is impossible to drive directly the large FIPs with these LSIs. In case of the circuits are assembled with discrete components, buffer drivers are required as interface. The driving voltage of FIPs, including the cutoff voltage, ranges from 12 V to 60 V. Direct drive is possible up to about 40 V by using LSIs or microcomputers. However, above this voltage buffer drivers are always required.

Buffer-drives of the FIP interfaces are considered as follows:

The determination of necessity of buffer-driver is shown in Fig. 8-8.

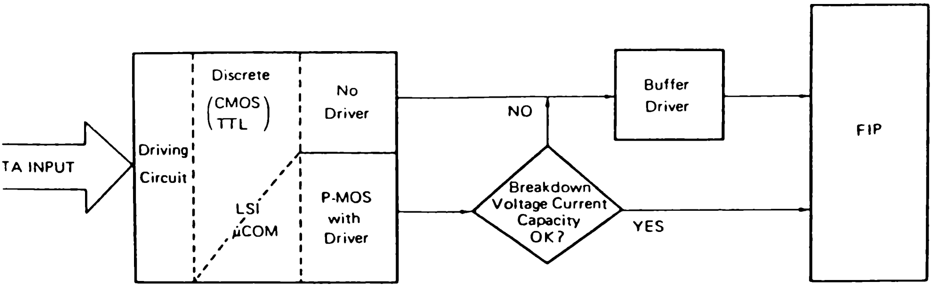


Fig. 8-8 Determination of Necessity of Buffer-Driver

As shown in Fig. 8-8, some driving circuit contains driver in it, but in case of output of the driving circuit is not enough to drive the FIP directly, a buffer-driver is necessary.

The buffer-driver must be chosen in accordance with output voltage, current from and output mode required.

On MPF-IP, NEC's UPA8ØC is used as the buffer driver for FIP. Fig. 8-9 shows connection of the buffer driver to the system and the FIP.

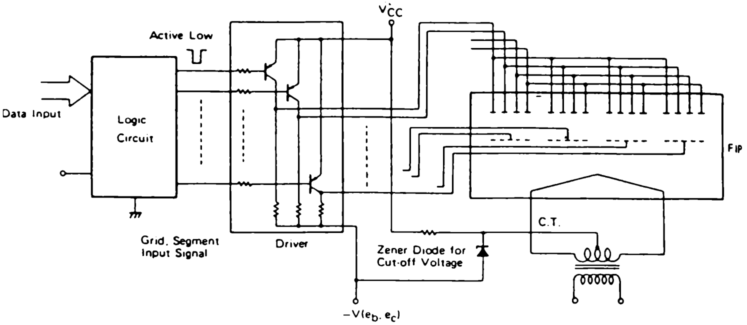


Fig. 8-9

In Fig. 8-9, the system sends an active low to the FIP and provides a voltage of 3ØV to the driver. The conception driving circuit and wave form of the filament voltage are shown in Fig. 8-1Ø.

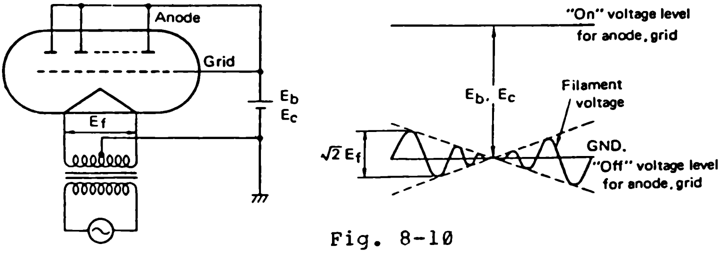

Anode
Grid
E_b
E_c
E_f
"On" voltage level for anode, grid
E_b, E_c
Filament voltage
GND.
$\sqrt{2}E_f$
"Off" voltage level for anode, grid

Fig. 8-1Ø

8.10.4 The Structure of FIP

The dynamic FIP is a display panel that can display up to 2Ø characters. Each display on the FIP consists of 16 segments, including the decimal point and single quote mark. The 16 segments are identified as a, b, c, d, e, f, g, h, i, j, k, l, m, n, dp, and COM. Each segment is wired to a control line, while each digit on the FIP is also controlled by a wire, identified as dgl, dg2, ... dg2Ø. A segment is illuminated only when the digit selection signal and segment-selection signal are supplied simultaneously to the FIP. But it requires a scanning circuit to display each digit.

(1) Scanning method of the FIP:

The principle of scanning the seven-segment display is as follows:

Each time a digit-selection signal is output, it is coupled with the segment-selection signal to display an alphanumeric character, a symbol, or a punctuation mark. For example, if the digit-selection signal selects dg1, while the segment-selection signals choose segments a, d, i, j, then the digit to which the dg1 is connected will be lighted, displaying the letter "I".

The scanning method is: Apply a signal voltage to the digit-selection lines in the sequence of dg1, dg2, dg3,...dg2Ø. When a digit-selection line is activated, voltage signals are applied to the segment-control lines a, b, c,...COM to display a desired character.

After all the digits in the FIP have been scanned once, the scanning is repeated from the beginning. Each digit must be scanned at least 2Ø times per second. Because of the persistence of vision of human eyes, all digits in the display appear to be lit simultaneously. The scanning speed can not be too fast, since the residual light of the neighboring digit may cause confusion.

(2) Scanning period and keybounce:

The keypad is usually depressed by hand. In general, the microcomputer's reaction is much faster than a human's response. To key in data or a command from the keyboard, the microcomputer must scan the keyboard repeatedly until a key is found depressed.

A key bounces for a short time when being depressed or released. Fig. 8-11 is a time response diagram of typical key-depressing or key-releasing operation.

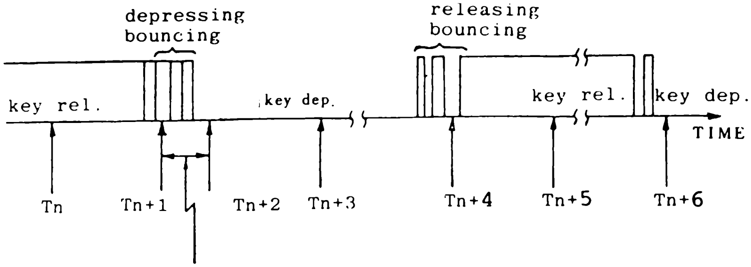


Fig. 8-11 The Time Response of Keyboard Scanning

Thus, a key-depression might be identified as two or more key-depressions if the key-board scanning rate is too fast. To avoid this problem, the period of scanning must be longer than the bouncing time (usually bouncing time is no longer than 1Øm sec). Since it takes more than 5Øm second for a human to release a key after he pressed a key, the period of scanning is between 1Øm sec and 5Øm sec.

In Fig. 8-11, an upward arrow indicates when a key is examined. At Tn+2, microcomputer program found that the key was depressed and returned the keycode. At Tn+3, the key was also found depressed. Since the key was found depressed in a previous scan, the microcomputer program would determine that this was not a new key-depression (i.e. the key had not been released during this time interval). Only if the key is found depressed at Tn+4 or Tn+5, a key-depression found at Tn+6 is really a new key-depression.

A program for getting input data from a keyboard designed in accordance with this rule will be error-free, no matter how long the duration of key-depression is and whatever is found at Tn+1 and Tn+4 (Ø or 1).

(3) Keyboard and Display Scanning Program

Usually the microcomputer scans the keyboard to fetch input of data from the keyboard. However, the keyboard scanning can not to be too fast because of key bouncing. Therefore, the CPU has sufficient time to scan the display while scanning the keyboard. Thus,the keyboard and display scanning is performed by a single subroutine -- SCAN1. The execution cycle of SCAN1 is 15.7 m second, e.g., it scans the keyboard and display 1ØØ times per second.

(4) Construction of MPF-IP display:

The display of the MPF-IP is an FIP consisting of 2Ø digits. A total of 35 control lines are used to control the display. Twenty lines are used for digit selection, and the remaining 15 control lines are used for segment selection.

The MPF-IP has two 8255s -- designated as 8255-I and 8255-II for input/output control. The Port A's eight output lines PAØ through PA7 of 8255-I control eight digit selection lines dg1 through dg8, the Port B's eight output lines PBØ through PB7 control digit selection lines dg9 through dg16, and PCØ through PC3 control dg17 through dg2Ø. The 8255-II's PAØ through PA7 control segments a through h, and its PBØ through PB6 control segments i through DP.

All the segments are controlled by logic "Ø" signals. If a segment is at logic "Ø", then it is lit. If a segment is at logic "1", then it is extinguished.

The digits of the FIP are also controlled by the logic "Ø" signals. If a digit is at logic "Ø", then it is selected, If a digit is at logic "1", then it is not selected.

(5) The structure of matrix-from keyboard:

A matrix-form keyboard is an important yet inexpensive input device for the micromputer. The structure of the keyboard is a number of wires in a matrix form. At each node of the matrix, a keypad is positioned. Please refer to the schematic of the MPF-IP.

The keyboard consists of 2Ø vertical lines and 3 horizontal lines. As a result, there are 6Ø (2Ø x 3) nodes -- contact -- points for keyboards. Of the 6Ø contact points, 45 are connected with signal lines.

Each key on the MPF-IP keyboard has a unique position code. When a key is pressed, the position code of the key pressed is fetched by the monitor program.

The three horizontal lines are connected to PCØ through PC2 of 8255-II. Refer to the schematic of MPF-IP sheet 2 and sheet 4. On sheet 4, you can see that three resistors are cnnected to the +5V power. Therefore, when no key is pressed, the input to the three pins -- PCØ through PC2 -- must be high or 1.

The 2Ø horizontal lines -- PAØ through PA7, PBØ through PB7, and PCØ through PC3 -- are wired to the keyboard and display. Refer to the schematic sheet 2, 3, and 4.

Each key on the keyboard is assigned a key position code. In the beginning of keyboard scanning, a counter is set to zero. Once a key being examined is found to be undepressed, the counter's value is increased until

(6) Keyboard scanning program

At the beginning of keyboard scanning, the Port A of 8255-I outputs "11111110" for 10 m sec, illuminating the rightmost digit on the FIP and scanning the first horizontal line to detect whether a "0" signal is entered. If a key is pressed (a "0" signal is detected), the key pressed can be identified by the port address (which is resulted from examing the state of the pin PC0 through PC2 of 8255-II.)

If no key in the first column is depressed, then the Port A of 8255-I will output "11111101", illuminating the second digit from right on the FIP and scanning the second row lines to detect whether a "0" signal is entered.

In general the keyboard scanning proceeds in the sequence, from top to bottom, from right to left of the key matrix, to examine if any key is depressed.

a key is found depressed. Thus, when a key is found depressed, the counter's value is the position code of that key.

(7) Conversion table

After the monitor program has fetched the position code, it will convert the position code to internal code. Then, it will check whether the SHIFT (8255-I PC4) and CONTROL (8255-I PC5) keys are pressed. If both keys were not depressed, then the internal code of the key pressed is the ASCII code of this key. If the monitor found that either SHIFT or CONTROL key is pressed, then the internal code should be processed further by the subroutines KCTRL and KSHIFT in order to get the ASCII code of this key.

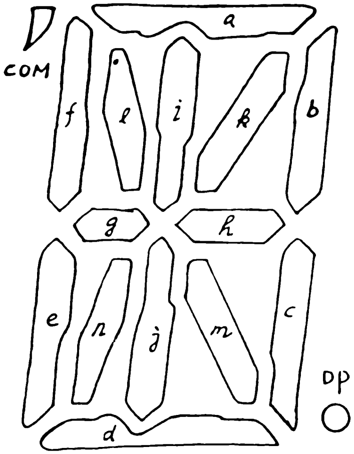
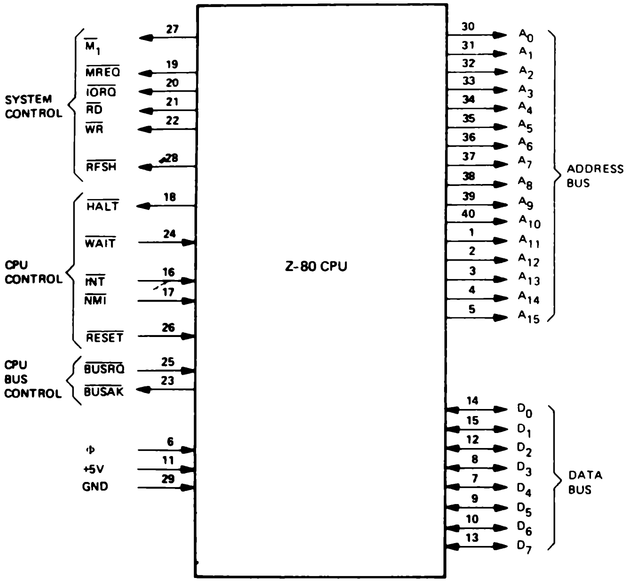


Fig. 8-12 Segment Pattern

Appendix A

Z-80 Pin Configuration

The Z-80 CPU is packaged in an industry standard 40 pin Dual In-Line Package. The I/O pins are shown in figure 3.0-1 and the function of each is described below.



Z-80 PIN CONFIGURATION
FIGURE 3.0-1

A_0-A_{15}
(Address Bus)

Tri-state output, active high. A_0-A_{15} constitute a 16-bit address bus. The address bus provides the address for memory (up to 64K bytes) data exchanges and for I/O device data exchanges. I/O addressing uses the 8 lower address bits to allow the user to directly select up to 256 input or 256 output ports. A_0 is the least significant address bit. During refresh time, the lower 7 bits contain a valid refresh address.

D_0-D_7
(Data Bus)

Tri-state input/output, active high. D_0-D_7 constitute an 8-bit bidirectional data bus. The data bus is used for data exchanges with memory and I/O devices.

$\overline{M_1}$
(Machine Cycle one)

Output, active low. $\overline{M_1}$ indicates that the current machine cycle is the OP code fetch cycle of an instruction execution. Note that during execution of 2-byte op-codes, $\overline{M1}$ is generated as each op code byte is fetched. These two byte op-codes always begin with CBH, DDH, EDH or FDH. $\overline{M1}$ also occurs with $\overline{IORQ}$ to indicate an interrupt acknowledge cycle.

$\overline{MREQ}$
(Memory Request)

Tri-state output, active low. The memory request signal indicates that the address bus holds a valid address for a memory read or memory write operation.

Appendix B

Z80-CPU Instruction Set

INTRODUCTION:

The assembly language provides a means for writing a program without having to be concerned with actual memory addresses or machine instruction formats. It allows the use of symbolic addresses to identify memory locations and mnemonic codes (opcodes and operands) to represent the instructions themselves. Labels (symbols) can be assigned to a particular instruction step in a source program to identify that step as an entry point for use in subsequent instructions. Operands following each instruction represent storage locations, registers, or constant values. The assembly language also includes assembler directives that supplement the machine instruction. A pseudo-op, for example, is a statement which is not translated into a machine instruction, but rather is interpreted as a directive that controls the assembly process.

A program written in assembly language is called a source program. It consists of symbolic commands called statements. Each statement is written on a single line and may consist of from one to four entries: A label field, an operation field, an operand field and a comment field. The source program is processed by the assembler to obtain a machine language program (object program) that can be executed directly by the Z80-CPU.

Zilog provides several different assemblers which differ in the features offered. Both absolute and relocatable assemblers are available with the Development and Microcomputer Systems. The absolute assembler is contained in base level software operating in a 16K memory space while the relocating assembler is part of the RIO environment operating in a 32K memory space.

A. THE ASSEMBLY LANGUAGE

The assembly language of the Z80 is designed to minimize the number of different opcodes corresponding to the set of basic machine operations and to provide for a consistent description of instruction operands. The nomenclature has been defined with special emphasis on mnemonic value and readability.

The movement of data is indicated primarily by a single opcode, LD for example, regardless of whether the movement is between different registers or between registers and memory locations.

The first operand of an LD instruction is the destination of the operation, and the second operand is the source of the operation. For example:

```
LD A,B
```

indicates that the contents of the second operand, register B, are to be transferred to the first operand, register A. Similarly,

```
LD C,3FH
```

indicates that the constant 3FH is to be loaded into the register C. In addition, enclosing an operand wholly in parentheses indicates a memory location addressed by the contents of the parentheses. For example,

```
LD HL,(1200)
```

indicates the contents of memory locations 1200 and 1201 are to be loaded into the 16-bit register pair HL. Similarly,

```
LD (IX+6),C
```

indicates the contents of the register C are to be stored in the memory location addressed by the current value of the 16-bit index register IX plus 6.

The regular formation of assembly instructions minimizes the number of mnemonics and format rules that the user must learn and manipulate. Additionally, the resulting programs are easier to interpret which in turn reduces programming errors and improves the maintainability of the software.

B. OPERANDS

Operands modify the opcodes and provide the information needed by the assembler to perform the designated operation.

Certain symbolic names are reserved as key words in the assembly language operand fields. They are:

1) The contents of 8-bit registers are specified by the character corresponding to the register names. The register names are A,B,C,D,E,H,L,I,R.

2) The contents of 16-bit double registers and register pairs consisting of two 8-bit registers are specified by the two characters corresponding to the register name or register pair. The names of double registers are IX,IY and SP. The names of registers pairs are AF,BC,DE and HL.

3) The contents of the auxiliary register pairs consisting of two 8-bit registers are specified by the two characters corresponding to the register pair names followed by an apostrophe. The auxiliary register pair names are AF′,BC′,DE′ and HL′. Only the pair AF′ is actually allowed as an operand, and then only in the EX AF,AF′ instruction.

4) The state of the four testable flags is specified as follows:

FLAG CONDITION	ON CONDITION	OFF
Carry	C	NC
Zero	Z	NZ
Sign	M (minus)	P (plus)
Parity	PE (even)	PO (odd)

OPERAND NOTATION

The following notation is used in the description of the assembly language:

1) r specifies any one of the following registers: A,B,C,D,E,H,L.
2) (HL) specifies the contents of memory at the location addressed by the contents of the register pair HL.
3) n specifies a one-byte expression in the range (0 to 255) nn specifies a two-byte expression in the range (0 to 65535).
4) d specifies a one-byte expression in the range (-128,127).
5) (nn) specifies the contents of memory at the location addressed by the two-byte expression nn.
6) b specifies an expression in the range (0,7).
7) e specifies a one-byte expression in the range (-126,129).
8) cc specifies the state of the Flags for conditional JR, JP, CALL and RET instructions.
9) qq specifies any one of the register pairs BC, DE, HL or AF.
10) ss specifies any one of the following register pairs: BC,DE,HL,SP.
11) pp specifies any one of the following register pairs: BC,DE,IX,SP.
12) rr specifies any one of the following register pairs: BC,DE,IY,SP.
13) s specifies any of r,n,(HL),(IX+d),(IY+d).
14) dd specifies any one of the following register pairs: BC,DE,HL,SP.
15) m specifies any of r,(HL),(IX+d),(IY+d).

C. RULES FOR WRITING ASSEMBLY STATEMENTS (SYNTAX)

An assembly language program (source program) consists of labels, opcodes, operands, comments and pseudo-ops in a sequence which defines the user's program.

There are 74 generic opcodes (such as LD), 25 operand key words (such as A), and 694 legitimate combinations of opcodes and operands in the Z80 instruction set.

ASSEMBLER STATEMENT FORMAT:

Statements are always written in a particular format. A typical Assembler statement is shown below:

```
LABEL      OPCODE     OPERANDS  COMMENT
LOOP:      LD         HL,VALUE  ;GET VALUE
```

In this example, the label, LOOP, provides a means for assigning a specific name to the instruction LOAD (LD), and is used to address the statement in other statements. The operand field contains one or two entries separated by one or more commas, tabs or spaces. The comment field is used by the programmer to quickly identify the action defined by the statement. Comments must begin with a semicolon and labels must be terminated by a colon, unless the label starts in column No. 1.

ALPHABETICAL ASSEMBLY MNEMONIC	OPERATION
ADC HL,ss	Add with Carry Reg. pair ss to HL
ADC A,s	Add with carry operand s to Acc.
ADD A,n	Add value n to Acc.
ADD A,r	Add Reg. r to Acc.
ADD A,(HL)	Add location (HL) to Acc.
ADD A,(IX+d)	Add location (IX+d) to Acc.
ADD A,(IY+d)	Add location (IY+d) to Acc.
ADD HL,ss	Add Reg. pair ss to HL
ADD IX,pp	Add Reg. pair pp to IX
ADD IY,rr	Add Reg. pair rr to IY
AND s	Logical AND′ of operand s and Acc.
BIT b,(HL)	Test BIT b of location (HL).
BIT b,(IX+d)	Test BIT b of location (IX+d)
BIT b,(IY+d)	Test BIT b of location (IY+d)
BIT b,r	Test BIT b of Reg. r
CALL cc,nn	Call subroutine at location nn if condition cc is true
CALL nn	Unconditional call subroutine at location nn
CCF	Complement carry flag
CP s	Compare operand s with Acc.
CPD	Compare location (HL) and Acc. decrement HL and BC
CPDR	Compare location (HL) and Acc. decrement HL and BC, repeat until BC=0
CPI	Compare location (HL) and Acc. increment HL and decrement BC
CPIR	Compare location (HL) and Acc. increment HL, decrement BC repeat until BC=0
CPL	Complement Acc. (1′s comp)
DAA	Decimal adjust Acc.
DEC m	Decrement operand m
DEC IX	Decrement IX
DEC IY	Decrement IY
DEC ss	Decrement Reg. pair ss
DI	Disable interrupts
DJNZ e	Decrement B and Jump relative if B≠0
EI	Enable interrupts
EX (SP),HL	Exchange the location (SP) and HL

EX (SP),IX	Exchange the location (SP) and IX
EX (SP),IY	Exchange the location (SP) and IY
EX AF,AF´	Exchange the contents of AF and AF´
EX DE,HL	Exchange the contents of DE and HL
EXX	Exchange the contents of BC,DE,HL with contents of BC´,DE´,HL´ respectively
HALT	HALT (wait for interrupt or reset)
IM 0	Set interrupt mode 0
IM 1	Set interrupt mode 1
IM 2	Set interrupt mode 2
IN A,(n)	Load the Acc. with input from device n
IN r,(C)	Load the Reg. r with input from device (C)
INC (HL)	Increment location (HL)
INC IX	Increment IX
INC (IX+d)	Increment location (IX+d)
INC IY	Increment IY
INC (IY+d)	Increment location (IY+d)
INC r	Increment Reg. r
INC ss	Increment Reg. pair ss
IND	Load location (HL) with input from port (C), decrement HL and B
INDR	Load location (HL) with input from port (C), decrement HL and decrement B, repeat until B=0
INI	Load location (HL) with input from port (C); and increment HL and decrement B
INIR	Load location (HL) with input from port (C), increment HL and decrement B, repeat until B=0
JP (HL)	Unconditional Jump to (HL)
JP (IX)	Unconditional Jump to (IX)
JP (IY)	Unconditional Jump to (IY)
JP cc,nn	Jump to location nn if condition cc is true
JP nn	Unconditional jump to location nn
JR C,e	Jump relative to PC+e if carry=1
JR e	Unconditional Jump relative to PC+e
JR NC,e	Jump relative to PC+e if carry=0

JR NZ,e	Jump relative to PC+e if non zero (Z=0)
JR Z,e	Jump relative to PC+e if zero (Z=1)
LD A,(BC)	Load Acc. with location (BC)
LD A,(DE)	Load Acc. with location (DE)
LD A,I	Load Acc. with I
LD A,(nn)	Load Acc. with location nn
LD A,R	Load Acc. with Reg. R
LD (BC),A	Load location (BC) with Acc.
LD (DE),A	Load location (DE) with Acc.
LD (HL),n	Load location (HL) with value n
LD dd,nn	Load Reg. pair dd with value nn
LD dd,(nn)	Load Reg. pair dd with location (nn)
LD HL,(nn)	Load HL with location (nn)
LD (HL),r	Load location (HL) with Reg. r
LD I,A	Load I with Acc.
LF IX,nn	Load IX with value nn
LD IX,(nn)	Load IX with location (nn)
LD (IX+d),n	Load location (IX+d) with value n
LD (IX+d),r	Load location (IX+d) with Reg. r
LD IY,nn	Load IY with value nn
LD IY,(nn)	Load IY with location (nn)
LD (IY+d),n	Load location (IY+d) with value n
LD (IY+d),r	Load location (IY+d) with Reg. r
LD (nn),A	Load location (nn) with Acc.
LD (nn),dd	Load location (nn) with Reg. pair dd
LD (nn),HL	Load location (nn) with HL
LD (nn),IX	Load location (nn) with IX
LD (nn),IY	Load location (nn) with IY
LD R,A	Load R with Acc.
LD r,(HL)	Load Reg. r with location (HL)
LD r,(IX+d)	Load Reg. r with location (IX+d)
LD r,(IY+d)	Load Reg. r with location (IY+d)
LD r,n	Load Reg. r with value n
LD r,r'	Load Reg. r with Reg. r'
LD SP,HL	Load SP with HL
LD SP,IX	Load SP with IX
LD SP,IY	Load SP with IY
LDD	Load location (DE) with location (HL), decrement DE,HL and BC
LDDR	Load location (DE) with location (HL) decrement DE,HL and BC; repeat until BC=0

LDI	Load location (DE) with location (HL), increment DE,HL, decrement BC
LDIR	Load location (DE) with location (HL), increment DE,HL, decrement BC and repeat until BC=0
NEG	Negate Acc. (2's complement)
NOP	No operation
OR s	Logical 'OR' of operand s and Acc.
OTDR	Load output port (C) with location (HL) decrement HL and B, repeat until B=0
OTIR	Load output port (C) with location (HL), increment HL, decrement B, repeat until B=0
OUT (C),r	Load output port (C) with Reg. r
OUT (n),A	Load output port (n) with Acc.
OUTD	Load output port (C) with location (HL), decrement HL and B
OUTI	Load output port (C) with location (HL), increment HL and decrement B
POP IX	Load IX with top of stack
POP IY	Load IY with top of stack
POP qq	Load Reg. pair qq with top of stack
PUSH IX	Load IX onto stack
PUSH IY	Load IY onto stack
PUSH qq	Load Reg. pair qq onto stack
RES b,m	Reset Bit b of operand m
RET	Return from subroutine
RET cc	Return from subroutine if condition cc is true
RETI	Return from interrupt
RETN	Return from non maskable interrupt
RL m	Rotate left through carry operand m
RLA	Rotate left Acc. through carry
RLC (HL)	Rotate location (HL) left circular
RLC (IX+d)	Rotate location (IX+d) left circclar
RLC (IY+d)	Rotate location (IY+d) left circular
RLC r	Rotate Reg. r left circular
RLCA	Rotate left circular Acc.....
RLD	Rotate digit left and right between Acc. and location (HL)
RR m	Rotate right through carry operand m
RRA	Rotate right Acc. through carry
RRC m	Rotate operand m right circular

RRCA	Rotate right circular Acc.
RRD	Rotate digit right and left between Acc. and location (HL)
RST p	Restart to location p
SBC A,s	Subtract operand s from Acc. with carry
SBC HL,ss	Subtract Reg. pair ss from HL with carry
SCF	Set carry flag (C=1)
SET b,(HL)	Set Bit b of location (HL)
SET b,(IX+d)	Set Bit b of location (IX+d)
SET b,(IY+d)	Set Bit b of location (IY+d)
SET b,r	Set Bit b of Reg. r
SLA m	Shift operand m left arithmetic
SRA m	Shift operand m right arithmetic
SRL m	Shift operand m right logical
SUB s	Subtract operand s from Acc.
XOR s	Exclusive 'OR' operand s and Acc.

Appendix C

Z80-CPU Programming Reference

Z80–CPU INSTRUCTIONS SORTED BY OP-CODE

OBJ CODE	SOURCE STATEMENT
00	NOP
018405	LD BC NN
02	LD (BC).A
03	INC BC
04	INC B
05	DEC B
0620	LD B.N
07	RLCA
08	EX AF.AF'
09	ADD HL.BC
0A	LD A.(BC)
0B	DEC BC
0C	INC C
0D	DEC C
0E20	LD C.N
0F	RRCA
102E	DJNZ DIS
118405	LD DE.NN
12	LD (DE).A
13	INC DE
14	INC D
15	DEC D
1620	LD D.N
17	RLA
182E	JR DIS
19	ADD HL.DE
1A	LD A.(DE)
1B	DEC DE
1C	INC E
1D	DEC E
1E20	LD E.N
1F	RRA
202E	JR NZ.DIS
218405	LD HL.NN
228405	LD (NN).HL
23	INC HL
24	INC H
25	DEC H
2620	LD H.N
27	DAA
282E	JR Z.DIS
29	ADD HL.HL
2A8405	LD HL.(NN)
2B	DEC HL
2C	INC L
2D	DEC L
2E20	LD L.N
2F	CPL
302E	JR NC.DIS
318405	LD SP.NN
328405	LD (NN).A
33	INC SP
34	INC (HL)
35	DEC (HL)
3620	LD (HL).N
37	SCF
382E	JR C.DIS
39	ADD HL.SP
3A8405	LD A.(NN)
3B	DEC SP
3C	INC A
3D	DEC A
3E20	LD A.N
3F	CCF
40	LD B.B
41	LD B.C
42	LD B.D
43	LD B.E
44	LD B.H.NN
45	LD B.L
46	LD B.(HL)
47	LD B.A
48	LD C.B
49	LD C.C
4A	LD C.D
4B	LD C.E
4C	LD C.H
4D	LD C.L
4E	LD C.(HL)
4F	LD C.A
50	LD D.B
51	LD D.C
52	LD D.D
53	LD D.E
54	LD D.H
55	LD D.L
56	LD D.(HL)
57	LD D.A
58	LD E.B
59	LD E.C
5A	LD E.D
5B	LD E.E
5C	LD E.H
5D	LD E.L
5E	LD E.(HL)
5F	LD E.A
60	LD H.B
61	LD H.C
62	LD H.D
63	LD H.E
64	LD H.H
65	LD H.L
66	LD H.(HL)
67	LD H.A
68	LD L.B
69	LD L.C
6A	LD L.D
6B	LD L.E
6C	LD L.H
6D	LD L.L
6E	LD L.(HL)
6F	LD L.A
70	LD (HL).B
71	LD (HL).C
72	LD (HL).D
73	LD (HL).E
74	LD (HL).H
75	LD (HL).L
76	HALT
77	LD (HL).A
78	LD A.B
79	LD A.C
7A	LD A.D
7B	LD A.E
7C	LD A.H
7D	LD A.L
7E	LD A.(HL)
7F	LD A.A
80	ADD A.B
81	ADD A.C
82	ADD A.D
83	ADD A.E
84	ADD A.H
85	ADD A.L
86	ADD A.(HL)
87	ADD A.A

88	ADC A,B
89	ADC A,C
8A	ADC A,D
8B	ADC A,E
8C	ADC A,H
8D	ADC A,L
8E	ADC A,(HL)
8F	ADC A,A
90	SUB B
91	SUB C
92	SUB D
93	SUB E
94	SUB H
95	SUB L
96	SUB (HL)
97	SUB A
98	SBC A,B
99	SBC A,C
9A	SBC A,D
9B	SBC A,E
9C	SBC A,H
9D	SBC A,L
9E	SBC A,(HL)
9F	SBC A,A
A0	AND B
A1	AND C
A2	AND D
A3	AND E
A4	AND H
A5	AND L
A6	AND (HL)
A7	AND A
A8	XOR B
A9	XOR C
AA	XOR D
AB	XOR E
AC	XOR H
AD	XOR L
AE	XOR (HL)
AF	XOR A
B0	OR B
B1	OR C
B2	OR D
B3	OR E
B4	OR H
B5	OR L
B6	OR (HL)
B7	OR A

B8	CP B
B9	CP C
BA	CP D
BB	CP E
BC	CP H
BD	CP L
BE	CP (HL)
BF	CP A
C0	RET NZ
C1	POP BC
C28405	JP NZ,NN
C38405	JP NN
C48405	CALL NZ,NN
C5	PUSH BC
C620	ADD A,N
C7	RST O
C8	RET Z
C9	RET
CA8405	JP Z,NN
CC8405	CALL Z,NN
CD8405	CALL NN
CE20	ADC A,N
CF	RST 8
D0	RET NC
D1	POP DE
D28405	JP NC,NN
D320	OUT (N),A
D48405	CALL NC,NN
D5	PUSH DE
D620	SUB N
D7	RST 10H
D8	RET C
D9	EXX
DA8405	JP C,NN
DB20	IN A,(N)
DC8405	CALL C,NN
DE20	SBC A,N
DF	RST 18H
E0	RET PO
E1	POP HL
E28405	JP PO,NN
E3	EX (SP),HL
E48405	CALL PO,NN
E5	PUSH HL
E620	AND N
E7	RST 20H
E8	RET PE
E9	JP (HL)

EA8405	JP PE,NN
EB	EX DE,HL
EC8405	CALL PE,NN
EE20	XOR N
EF	RST 28H
F0	RET P
F1	POP AF
F28405	JP P,NN
F3	DI
F48405	CALL P,NN
F5	PUSH AF
F620	OR N
F7	RST 30H
F8	RET M
F9	LD SP,HL
FA8405	JP M,NN
FB	EI
FC8405	CALL M,NN
FE20	CP N
FF	RST 38H
CB00	RLC B
CB01	RLC C
CB02	RLC D
CB03	RLC E
CB04	RLC H
CB05	RLC L
CB06	RLC (HL)
CB07	RLC A
CB08	RRC B
CB09	RRC C
CB0A	RRC D
CB0B	RRC E
CB0C	RRC H
CB0D	RRC L
CB0E	RRC (HL)
CB0F	RRC A
CB10	RL B
CB11	RL C
CB12	RL D
CB13	RL E
CB14	RL H
CB15	RL L
CB16	RL (HL)
CB17	RL A
CB18	RR B
CB19	RR C
CB1A	RR D
CB1B	RR E

CB1C	RR H
CB1D	RR L
CB1E	RR (HL)
CB1F	RR A
CB20	SLA B
CB21	SLA C
CB22	SLA D
CB23	SLA E
CB24	SLA H
CB25	SLA L
CB26	SLA (HL)
CB27	SLA A
CB28	SRA B
CB29	SRA C
CB2A	SRA D
CB2B	SRA E
CB2C	SRA H
CB2D	SRA L
CB2E	SRA (HL)
CB2F	SRA A
CB38	SRL B
CB39	SRL C
CB3A	SRL D
CB3B	SRL E
CB3C	SRL H
CB3D	SRL L
CB3E	SRL (HL)
CB3F	SRL A
CB40	BIT 0,B
CB41	BIT 0,C
CB42	BIT 0,D
CB43	BIT 0,E
CB44	BIT 0,H
CB45	BIT 0,L
CB46	BIT 0,(HL)
CB47	BIT 0,A
CB48	BIT 1,B
CB49	BIT 1,C
CB4A	BIT 1,D
CB4B	BIT 1,E
CB4C	BIT 1,H
CB4D	BIT 1,L
CB4E	BIT 1,(HL)
CB4F	BIT 1,A
CB50	BIT 2,B
CB51	BIT 2,C
CB52	BIT 2,D
CB53	BIT 2,E

CB54	BIT 2,H
CB55	BIT 2,L
CB56	BIT 2,(HL)
CB57	BIT 2,A
CB58	BIT 3,B
CB59	BIT 3,C
CB5A	BIT 3,D
CB5B	BIT 3,E
CB5C	BIT 3,H
CB5D	BIT 3,L
CB5E	BIT 3,(HL)
CB5F	BIT 3,A
CB60	BIT 4,B
CB61	BIT 4,C
CB62	BIT 4,D
CB63	BIT 4,E
CB64	BIT 4,H
CB65	BIT 4,L
CB66	BIT 4,(HL)
CB67	BIT 4,A
CB68	BIT 5,B
CB69	BIT 5,C
CB6A	BIT 5,D
CB6B	BIT 5,E
CB6C	BIT 5,H
CB6D	BIT 5,L
CB6E	BIT 5,(HL)
CB6F	BIT 5,A
CB70	BIT 6,B
CB71	BIT 6,C
CB72	BIT 6,D
CB73	BIT 6,E
CB74	BIT 6,H
CB75	BIT 6,L
CB76	BIT 6,(HL)
CB77	BIT 6,A
CB78	BIT 7,B
CB79	BIT 7,C
CB7A	BIT 7,D
CB7B	BIT 7,E
CB7C	BIT 7,H
CB7D	BIT 7,L
CB7E	BIT 7,(HL)
CB7F	BIT 7,A
CB80	RES 0,B
CB81	RES 0,C
CB82	RES 0,D
CB83	RES 0,E

CB84	RES 0,H
CB85	RES 0,L
CB86	RES 0,(HL)
CB87	RES 0,A
CB88	RES 1,B
CB89	RES 1,C
CB8A	RES 1,D
CB8B	RES 1,E
CB8C	RES 1,H
CB8D	RES 1,L
CB8E	RES 1,(HL)
CB8F	RES 1,A
CB90	RES 2,B
CB91	RES 2,C
CB92	RES 2,D
CB93	RES 2,E
CB94	RES 2,H
CB95	RES 2,L
CB96	RES 2,(HL)
CB97	RES 2,A
CB98	RES 3,B
CB99	RES 3,C
CB9A	RES 3,D
CB9B	RES 3,E
CB9C	RES 3,H
CB9D	RES 3,L
CB9E	RES 3,(HL)
CB9F	RES 3,A
CBA0	RES 4,B
CBA1	RES 4,C
CBA2	RES 4,D
CBA3	RES 4,E
CBA4	RES 4,H
CBA5	RES 4,L
CBA6	RES 4,(HL)
CBA7	RES 4,A
CBA8	RES 5,B
CBA9	RES 5,C
CBAA	RES 5,D
CBAB	RES 5,E
CBAC	RES 5,H
CBAD	RES 5,L
CBAE	RES 5,(HL)
CBAF	RES 5,A
CBB0	RES 6,B
CBB1	RES 6,C
CBB2	RES 6,D
CBB3	RES 6,E

CBB4	RES 6,H
CBB5	RES 6,L
CBB6	RES 6,(HL)
CBB7	RES 6,A
CBB8	RES 7,B
CBB9	RES 7,C
CBBA	RES 7,D
CBBB	RES 7,E
CBBC	RES 7,H
CBBD	RES 7,L
CBBE	RES 7,(HL)
CBBF	RES 7,A
CBC0	SET 0,B
CBC1	SET 0,C
CBC2	SET 0,D
CBC3	SET 0,E
CBC4	SET 0,H
CBC5	SET 0,L
CBC6	SET 0,(HL)
CBC7	SET 0,A
CBC8	SET 1,B
CBC9	SET 1,C
CBCA	SET 1,D
CBCB	SET 1,E
CBCC	SET 1,H
CBCD	SET 1,L
CBCE	SET 1,(HL)
CBCF	SET 1,A
CBD0	SET 2,B
CBD1	SET 2,C
CBD2	SET 2,D
CBD3	SET 2,E
CBD4	SET 2,H
CBD5	SET 2,L
CBD6	SET 2,(HL)
CBD7	SET 2,A
CBD8	SET 3,B
CBD9	SET 3,C
CBDA	SET 3,D
CBDB	SET 3,E
CBDC	SET 3,H
CBDD	SET 3,L
CBDE	SET 3,(HL)
CBDF	SET 3,A
CBE0	SET 4,B
CBE1	SET 4,C
CBE2	SET 4,D
CBE3	SET 4,E

CBE4	SET 4,H
CBE5	SET 4,L
CBE6	SET 4,(HL)
CBE7	SET 4,A
CBE8	SET 5,B
CBE9	SET 5,C
CBEA	SET 5,D
CBEB	SET 5,E
CBEC	SET 5,H
CBED	SET 5,L
CBEE	SET 5,(HL)
CBEF	SET 5,A
CBF0	SET 6,B
CBF1	SET 6,C
CBF2	SET 6,D
CBF3	SET 6,E
CBF4	SET 6,H
CBF5	SET 6,L
CBF6	SET 6,(HL)
CBF7	SET 6,A
CBF8	SET 7,B
CBF9	SET 7,C
CBFA	SET 7,D
CBFB	SET 7,E
CBFC	SET 7,H
CBFD	SET 7,L
CBFE	SET 7,(HL)
CBFF	SET 7,A
DD09	ADD IX,BC
DD19	ADD IX,DE
DD218405	LD IX,NN
DD228405	LD (NN),IX
DD23	INC IX
DD29	ADD IX IX
DD2A8405	LD IX, (NN)
DD2B	DEC IX
DD3405	INC (IX+d)
DD3505	DEC (IX+d)
DD360520	LD (IX+d),N
DD39	ADD IX,SP
DD4605	LD B,(IX+d)
DD4E05	LD C,(IX+d)
DD5605	LD D,(IX+d)
DD5E05	LD E,(IX+d)
DD6605	LD H,(IX+d)
DD6E05	LD L,(IX+d)
DD7005	LD (IX+d),B
DD7105	LD (IX+d),C

DD7205	LD (IX+d),D
DD7305	LD (IX+d),E
DD7405	LD (IX+d),H
DD7505	LD (IX+d),L
DD7705	LD (IX+d),A
DD7E05	LD A,(IX+d)
DD8605	ADD A,(IX+d)
DD8E05	ADC A,(IX+d)
DD9605	SUB (IX+d)
DD9E05	SBC A,(IX+d)
DDA605	AND (IX+d)
DDAE05	XOR (IX+d)
DDB605	OR (IX+d)
DDBE05	CP (IX+d)
DDE1	POP IX
DDE3	EX (SP),IX
DDE5	PUSH IX
DDE9	JP (IX)
DDF9	LD SP,IX
DDCB0506	RLC (IX+d)
DDCB050E	RRC (IX+d)
DDCB0516	RL (IX+d)
DDCB051E	RR (IX+d)
DDCB0526	SLA (IX+d)
DDCB052E	SRA (IX+d)
DDCB053E	SRL (IX+d)
DDCB0546	BIT 0,(IX+d)
DDCB054E	BIT 1,(IX+d)
DDCB0556	BIT 2,(IX+d)
DDCB055E	BIT 3,(IX+d)
DDCB0566	BIT 4,(IX+d)
DDCB056E	BIT 5,(IX+d)
DDCB0576	BIT 6,(IX+d)
DDCB057E	BIT 7,(IX+d)
DDCB0586	RES 0,(IX+d)
DDCB058E	RES 1,(IX+d)
DDCB0596	RES 2,(IX+d)
DDCB059E	RES 3,(IX+d)
DDCB05A6	RES 4,(IX+d)
DDCB05AE	RES 5,(IX+d)
DDCB05B6	RES 6,(IX+d)
DDCB05BE	RES 7,(IX+d)
DDCB05C6	SET 0,(IX+d)
DDCB05CE	SET 1,(IX+d)
DDCB05D6	SET 2,(IX+d)
DDCB05DE	SET 3,(IX+d)
DDCB05E6	SET 4,(IX+d)
DDCB05EE	SET 5,(IX+d)

DDCB05F6	SET 6,(IX+d)
DDCB05FE	SET 7,(IX+d)
ED40	IN B,(C)
ED41	OUT (C),B
ED42	SBC HL,BC
ED438405	LD (NN),BC
ED44	NEG
ED45	RETN
ED46	IM 0
ED47	LD I,A
ED48	IN C,(C)
ED49	OUT (C),C
ED4A	ADC HL,BC
ED4B8405	LD BC,(NN)
ED4D	RETI
ED50	IN D,(C)
ED51	OUT (C),D
ED52	SBC HL,DE
ED538405	LD (NN),DE
ED56	IM 1
ED57	LD A,I
ED58	IN E,(C)
ED59	OUT (C),E
ED5A	ADC HL,DE
ED5B8405	LD DE,(NN)
ED5E	IM 2
ED60	IN H,(C)
ED61	OUT (C),H
ED62	SBC HL,HL
ED67	RRD
ED68	IN L,(C)
ED69	OUT (C),L
ED6A	ADC HL,HL
ED6F	RLD
ED72	SBC HL,SP
ED738405	LD (NN),SP
ED78	IN A,(C)
ED79	OUT (C),A
ED7A	ADC HL,SP
ED7B8405	LD SP,(NN)
EDA0	LDI
EDA1	CPI
EDA2	INI
EDA3	OUTI
EDA8	LDD
EDA9	CPD
EDAA	IND
EDAB	OUTD

EDB0	LDIR
EDB1	CPIR
EDB2	INIR
EDB3	OTIR
EDB8	LDDR
EDB9	CPDR
EDBA	INDR
EDBB	OTDR
FD09	ADD IY,BC
FD19	ADD IY,DE
FD218405	LD IY,NN
FD228405	LD (NN),IY
FD23	INC IY
FD29	ADD IY,IY
FD2A8405	LD IY,(NN)
FD2B	DEC IY
FD3405	INC (IY+d)
FD3505	DEC (IY+d)
FD360520	LD (IY+d),N
FD39	ADD IY,SP
FD4605	LD B,(IY+d)
FD4E05	LD C,(IY+d)
FD5605	LD D,(IY+d)
FD5E05	LD E,(IY+d)
FD6605	LD H,(IY+d)
FD6E05	LD L,(IY+d)
FD7005	LD (IY+d),B
FD7105	LD (IY+d),C
FD7205	LD (IY+d),D
FD7305	LD (IY+d),E
FD7405	LD (IY+d),H
FD7505	LD (IY+d),L
FD7705	LD (IY+d),A
FD7E05	LD A,(IY+d)
FD8605	ADD A,(IY+d)
FD8E05	ADC A,(IY+d)
FD9605	SUB (IY+d)
FD9E05	SBC A,(IY+d)
FDA605	AND (IY+d)
FDAE05	XOR (IY+d)
FDB605	OR (IY+d)
FDBE05	CP (IY+d)
FDE1	POP IY
FDE3	EX (SP),IY
FDE5	PUSH IY
FDE9	JP (IY)
FDF9	LD SP,IY
FDCB0506	RLC (IY+d)

FDCB050E	RRC (IY+d)
FDCB0516	RL (IY+d)
FDCB051E	RR (IY+d)
FDCB0526	SLA (IY+d)
FDCB052E	SRA (IY+d)
FDCB053E	SRL (IY+d)
FDCB0546	BIT 0,(IY+d)
FDCB054E	BIT 1,(IY+d)
FDCB0556	BIT 2,(IY+d)
FDCB055E	BIT 3,(IY+d)
FDCB0566	BIT 4,(IY+d)
FDCB056E	BIT 5,(IY+d)
FDCB0576	BIT 6,(IY+d)
FDCB057E	BIT 7,(IY+d)
FDCB0586	RES 0,(IY+d)
FDCB058E	RES 1,(IY+d)
FDCB0596	RES 2,(IY+d)
FDCB059E	RES 3,(IY+d)
FDCB05A6	RES 4,(IY+d)
FDCB05AE	RES 5,(IY+d)
FDCB05B6	RES 6,(IY+d)
FDCB05BE	RES 7,(IY+d)
FDCB05C6	SET 0,(IY+d)
FDCB05CE	SET 1,(IY+d)
FDCB05D6	SET 2,(IY+d)
FDCB05DE	SET 3,(IY+d)
FDCB05E6	SET 4,(IY+d)
FDCB05EE	SET 5,(IY+d)
FDCB05F6	SET 6,(IY+d)
FDCB05FE	SET 7,(IY+d)

Z80—CPU INSTRUCTIONS SORTED BY MNEMONIC

OBJ CODE	SOURCE STATEMENT
8E	ADC A,(HL)
DD8E05	ADC A,(IX+d)
FD8E05	ADC A,(IY+d)
8F	ADC A,A

88	ADC A,B
89	ADC A,C
8A	ADC A,D
8B	ADC A,E
8C	ADC A,H
8D	ADC A,L
CE20	ADC A,N
ED4A	ADC HL,BC
ED5A	ADC HL,DE
ED6A	ADC HL,HL
ED7A	ADC HL,SP
86	ADD A,(HL)
DD8605	ADD A,(IX+d)
FD8605	ADD A,(IY+d)
87	ADD A,A
80	ADD A,B
81	ADD A,C
82	ADD A,D
83	ADD A,E
84	ADD A,H
85	ADD A,L
C620	ADD A,N
09	ADD HL,BC
19	ADD HL,DE
29	ADD HL,HL
39	ADD HL,SP
DD09	ADD IX,BC
DD19	ADD IX,DE
DD29	ADD IX,IX
DD39	ADD IX,SP
FD09	ADD IY,BC
FD19	ADD IY,DE
FD29	ADD IY,IY
FD39	ADD IY,SP
A6	AND (HL)
DDA605	AND (IX+d)
FDA605	AND (IY+d)
A7	AND A
A0	AND B
A1	AND C
A2	AND D
A3	AND E
A4	AND H
A5	AND L
E620	AND N
CB46	BIT 0,(HL)
DDCB0546	BIT 0,(IX+d)
FDCB0546	BIT 0,(IY+d)
CB47	BIT 0,A
CB40	BIT 0,B
CB41	BIT 0,C
CB42	BIT 0,D
CB43	BIT 0,E
CB44	BIT 0,H
CB45	BIT 0,L
CB4E	BIT 1,(HL)
DDCB054E	BIT 1,(IX+d)
FDCB054E	BIT 1,(IY+d)
CB4F	BIT 1,A
BC48	BIT 1,B
CB49	BIT 1,C
CB4A	BIT 1,D
CB4B	BIT 1,E
CB4C	BIT 1,H
CB4D	BIT 1,L
CB56	BIT 2,(HL)
DDCB0556	BIT 2,(IX+d)
FDCB0556	BIT 2,(IY+d)
CB57	BIT 2,A
CB50	BIT 2,B
CB51	BIT 2,C
CB52	BIT 2,D
CB53	BIT 2,E
CB54	BIT 2,H
CB55	BIT 2,L
CB5E	BIT 3,(HL)
DDCB055E	BIT 3,(IX+d)
FDCB055E	BIT 3,(IY+d)
CB5F	BIT 3,A
CB58	BIT 3,B
CB59	BIT 3,C
CB5A	BIT 3,D
CB5B	BIT 3,E
CB5C	BIT 3,H
CB5D	BIT 3,L
CB66	BIT 4,(HL)
DDCB0566	BIT 4,(IX+d)
FDCB0566	BIT 4,(IY+d)
CB67	BIT 4,A
CB60	BIT 4,B
CB61	BIT 4,C
CB62	BIT 4,D
CB63	BIT 4,E
CB64	BIT 4,H
CB65	BIT 4,L
CB6E	BIT 5,(HL)
DDCB056E	BIT 5,(IX+d)
FDCB056E	BIT 5,(IY+d)
CB6F	BIT 5,A
CB68	BIT 5,B
CB69	BIT 5,C
CB6A	BIT 5,D
CB6B	BIT 5,E
CB6C	BIT 5,H
CB6D	BIT 5,L
CB76	B'T 6,(HL)
DDCB0576	BIT 6,(IX+d)
FDCB0576	BIT 6,(IY+d)
CB77	BIT 6,A
CB70	BIT 6,B
CB71	BIT 6,C
CB72	BIT 6,D
CB73	BIT 6,E
CB74	BIT 6,H
CB75	BIT 6,L
CB7E	BIT 7,(HL)
DDCB057E	BIT 7,(IX+d)
FDCB057E	BIT 7,(IY+d)
CB7F	BIT 7,A
CB78	BIT 7,B
CB79	BIT 7,C
CB7A	BIT 7,D
CB7B	BIT 7,E
CB7C	BIT 7,H
CB7D	BIT 7,L
DC8405	CALL C,NN
FC8405	CALL M,NN
D48405	CALL NC,NN
CD8405	CALL NN
C48405	CALL NZ,NN
F48405	CALL P,NN
EC8405	CALL PE,NN
E48405	CALL PO,NN
CC8405	CALL Z,NN
3F	CCF
BE	CP (HL)
DDBE05	CP (IX+d)
FDBE05	CP (IY+d)
BF	CP A
B8	CP B
B9	CP C
BA	CP D
BB	CP E
BC	CP H

BD	CP L
FE20	CP N
EDA9	CPD
ED89	CPDR
EDA1	CPI
EDB1	CPIR
2F	CPL
27	DAA
35	DEC (HL)
DD3505	DEC (IX+d)
FD3505	DEC (IY+d)
3D	DEC A
05	DEC B
0B	DEC BC
0D	DEC C
15	DEC D
1B	DEC DE
1D	DEC E
25	DEC H
2B	DEC HL
DD2B	DEC IX
FD2B	DEC IY
2D	DEC L
3B	DEC SP
F3	DI
102E	DJNZ DIS
FB	EI
E3	EX (SP),HL
DDE3	EX (SP),IX
FDE3	EX (SP),IY
08	EX AF,AF'
EB	EX DE,HL
D9	EXX
76	HALT
ED46	IM 0
ED56	IM 1
ED5E	IM 2
ED78	IN A,(C)
DB20	IN A,(N)
ED40	IN B,(C)
ED48	IN C,(C)
ED50	IN D,(C)
ED58	IN E,(C)
ED60	IN H,(C)
ED68	IN L,(C)
34	INC (HL)
DD3405	INC (IX+d)
FD3405	INC (IY+d)

3C	INC A
04	INC B
03	INC BC
0C	INC C
14	INC D
13	INC DE
1C	INC E
24	INC H
23	INC HL
DD23	INC IX
FD23	INC IY
2C	INC L
33	INC SP
EDAA	IND
EDBA	INDR
EDA2	INI
EDB2	INIR
E9	JP (HL)
DDE9	JP (IX)
FDE9	JP (IY)
DA8405	JP C NN
FA8405	JP M NN
D28405	JP NC NN
C38405	JP NN
C28405	JP NZ,NN
F28405	JP P,NN
EA8405	JP PE,NN
E28405	JP PO,NN
CA8405	JP Z,NN
382E	JR C,DIS
182E	JR DIS
302E	JR NC,DIS
202E	JR NZ,DIS
282E	JR Z,DIS
02	LD (BC),A
12	LD (DE),A
77	LD (HL),A
70	LD (HL),B
71	LD (HL),C
72	LD (HL),D
73	LD (HL),E
74	LD (HL),H
75	LD (HL),L
3620	LD (HL),N
DD7705	LD (IX+d),A
DD7005	LD (IX+d),B
DD7105	LD (IX+d),C
DD7205	LD (IX+d),D

DD7305	LD (IX+d),E
DD7405	LD (IX+d),H
DD7505	LD (IX+d),L
DD360520	LD (IX+d),N
FD7705	LD (IY+d),A
FD7005	LD (IY+d),B
FD7105	LD (IY+d),C
FD7205	LD (IY+d),D
FD7305	LD (IY+d),E
FD7405	LD (IY+d),H
FD7505	LD (IY+d),L
FD360520	LD (IY+d),N
328405	LD (NN),A
ED438405	LD (NN),BC
ED538405	LD (NN),DE
228405	LD (NN),HL
DD228405	LD (NN),IX
FD228405	LD (NN),IY
ED738405	LD (NN),SP
0A	LD A,(BC)
1A	LD A,(DE)
7E	LD A,(HL)
DD7E05	LD A,(IX+d)
FD7E05	LD A,(IY+d)
3A8405	LD A,(NN)
7F	LD A,A
78	LD A,B
79	LD A,C
7A	LD A,D
7B	LD A,E
7C	LD A,H
ED57	LD A,I
7D	LD A,L
3E20	LD A;N
46	LD B,(HL)
DD4605	LD B,(IX+d)
FD4605	LD B,(IY+d)
47	LD B,A
40	LD B,B
41	LD B,C
42	LD B,D
43	LD B,E
44	LD B,H,NN
45	LD B,L
0620	LD B,N
ED4B8405	LD BC,(NN)
018405	LD BC,NN
4E	LD C (HL)

DD4E05	LD C,(IX+d)
FD4E05	LD C,(IY+d)
4F	LD C,A
48	LD C,B
49	LD C,C
4A	LD C,D
4B	LD C,E
4C	LD C,H
4D	LD C,L
0E20	LD C,N
56	LD D,(HL)
DD5605	LD D,(IX+d)
FD5605	LD D,(IY+d)
57	LD D,A
50	LD D,B
51	LD D,C
52	LD D,D
53	LD D,E
54	LD D,H
55	LD D,L
1620	LD D,N
ED5B8405	LD DE,(NN)
118405	LD DE,NN
5E	LD E,(HL)
DD5E05	LD E,(IX+d)
FD5E05	LD E,(IY+d)
5F	LD E,A
58	LD E,B
59	LD E,C
5A	LD E,D
5B	LD E,E
5C	LD E,H
5D	LD E,L
1E20	LD E,N
66	LD H,(HL)
DD6605	LD H,(IX+d)
FD6605	LD H,(IY+d)
67	LD H,A
60	LD H,B
61	LD H,C
62	LD H,D
63	LD H,E
64	LD H,H
65	LD H,L
2620	LD H,N
2A8405	LD HL,(NN)
218405	LD HL,NN
ED47	LD I,A

DD2A8405	LD IX,(NN)
DD218405	LD IX,NN
FD2A8405	LD IY,(NN)
FD218405	LD IY,NN
6E	LD L,(HL)
DD6E05	LD L,(IX+d)
FD6E05	LD L,(IY+d)
6F	LD L,A
68	LD L,B
69	LD L,C
6A	LD L,D
6B	LD L,E
6C	LD L,H
6D	LD L,L
2E20	LD L,N
ED7B8405	LD SP,(NN)
F9	LD SP HL
DDF9	LD SP,IX
FDF9	LD SP,IY
318405	LD SP,NN
EDA8	LDD
EDB8	LDDR
EDA0	LDI
EDB0	LDIR
ED44	NEG
00	NOP
B6	OR (HL)
DDB605	OR (IX+d)
FDB605	OR (IY+d)
B7	OR A
B0	OR B
B1	OR C
B2	OR D
B3	OR E
B4	OR H
B5	OR L
F620	OR N
EDBB	OTDR
EDB3	OTIR
ED79	OUT (C),A
ED41	OUT (C),B
ED49	OUT (C),C
ED51	OUT (C),D
ED59	OUT (C),E
ED61	OUT (C),H
ED69	OUT (C),L
D320	OUT (N),A
EDAB	OUTD

EDA3	OUTI
F1	POP AF
C1	POP BC
D1	POP DE
E1	POP HL
DDE1	POP IX
FDE1	POP IY
F5	PUSH AF
C5	PUSH BC
D5	PUSH DE
E5	PUSH HL
DDE5	PUSH IX
FDE5	PUSH IY
CB86	RES 0,(HL)
DDCB0586	RES 0,(IX+d)
FDCB0586	RES 0,(IY+d)
CB87	RES 0,A
CB80	RES 0,B
CB81	RES 0,C
CB82	RES 0,D
CB83	RES 0,E
CB84	RES 0,H
CB85	RES 0,L
CB8E	RES 1,(HL)
DDCB058E	RES 1,(IX+d)
FDCB058E	RES 1,(IY+d)
CB8F	RES 1,A
CB88	RES 1,B
CB89	RES 1,C
CB8A	RES 1,D
CB8B	RES 1,E
CB8C	RES 1,H
CB8D	RES 1,L
CB96	RES 2,(HL)
DDCB0596	RES 2,(IX+d)
FDCB0596	RES 2,(IY+d)
CB97	RES 2,A
CB90	RES 2,B
CB91	RES 2,C
CB92	RES 2,D
CB93	RES 2,E
CB94	RES 2,H
CB95	RES 2,L
CB9E	RES 3,(HL)
DDCB059E	RES 3,(IX+d)
FDCB059E	RES 3,(IY+d)
CB9F	RES 3,A
CB98	RES 3,B

CB99	RES 3,C
CB9A	RES 3,D
CB9B	RES 3,E
CB9C	RES 3,H
CB9D	RES 3,L
CBA6	RES 4,(HL)
DDCB05A6	RES 4,(IX+d)
FDCB05A6	RES 4,(IY+d)
CBA7	RES 4,A
CBA0	RES 4,B
CBA1	RES 4,C
CBA2	RES 4,D
CBA3	RES 4,E
CBA4	RES 4,H
CBA5	RES 4,L
CBAE	RES 5,(HL)
DDCB05AE	RES 5,(IX+d)
FDCB05AE	RES 5,(IY+d)
CBAF	RES 5,A
CBA8	RES 5,B
CBA9	RES 5,C
CBAA	RES 5,D
CBAB	RES 5,E
CBAC	RES 5,H
CBAD	RES 5,L
CBB6	RES 6,(HL)
DDCB05B6	RES 6,(IX+d)
FDCB05B6	PES 6,(IY+d)
CBB7	RES 6,A
CBB0	RES 6,B
CBB1	RES 6,C
CBB2	RES 6,D
CBB3	RES 6,E
CBB4	RES 6,H
CBB5	RES 6,L
CBBE	RES 7,(HL)
DDCB05BE	RES 7,(IX+d)
FDCB05BE	RES 7,(IY+d)
CBBF	RES 7,A
CBB8	RES 7,B
CBB9	RES 7,C
CBBA	RES 7,D
CBBB	RES 7,E
CBBC	RES 7,H
CBBD	RES 7,L
C9	RET
D8	RET C
F8	RET M

D0	RET NC
C0	RET NZ
F0	RET P
E8	RET PE
E0	RET PO
C8	RET Z
ED4D	RETI
ED45	RETN
CB16	RL (HL)
DDCB0516	RL (IX+d)
FDCB0516	RL (IY+d)
CB17	RL A
CB10	RL B
CB11	RL C
CB12	RL D
CB13	RL E
CB14	RL H
CB15	RL L
17	RLA
CB06	RLC (HL)
DDCB0506	RLC (IX+d)
FDCB0506	RLC (IY+d)
CB07	RLC A
CB00	RLC B
CB01	RLC C
CB02	RLC D
CB03	RLC E
CB04	RLC H
CB05	RLC L
07	RLCA
ED6F	RLD
CB1E	RR (HL)
DDCB051E	RR (IX+d)
FDCB051E	RR (IY+d)
CB1F	RR A
CB18	RR B
CB19	RR C
CB1A	RR D
CB1B	RR E
CB1C	RR H
CB1D	RR L
1F	RRA
CB0E	RRC (HL)
DDCB050E	RRC (IX+d)
FDCB050E	RRC (IY+d)
CB0F	RRC A
CB08	RRC B
CB09	RRC C

CB0A	RRC D
CB0B	RRC E
CB0C	RRC H
CB0D	RRC L
0F	RRCA
ED67	RRD
C7	RST 0
D7	RST 10H
DF	RST 18H
E7	RST 20H
EF	RST 28H
F7	RST 30H
FF	RST 38H
CF	RST 8
9E	SBC A,(HL)
DD9E05	SBC A,(IX+d)
FD9E05	SBC A,(IY+d)
9F	SBC A,A
98	SBC A,B
99	SBC A,C
9A	SBC A,D
9B	SBC A,E
9C	SBC A,H
9D	SBC A,L
DE20	SBC A,N
ED42	SBC HL,BC
ED52	SBC HL,DE
ED62	SBC HL,HL
ED72	SBC HL,SP
37	SCF
CBC6	SET 0,(HL)
DDCB05C6	SET 0,(IX+d)
FDCB05C6	SET 0,(IY+d)
CBC7	SET 0,A
CBC0	SET 0,B
CBC1	SET 0,C
CBC2	SET 0,D
CBC3	SET 0,E
CBC4	SET 0,H
CBC5	SET 0,L
CBCE	SET 1,(HL)
DDCB05CE	SET 1,(IX+d)
FDCB05CE	SET 1,(IY+d)
CBCF	SET 1,A
CBC8	SET 1,B
CBC9	SET 1,C
CBCA	SET 1,D
CBCB	SET 1,E

CBCC	SET 1,H
CBCD	SET 1,L
CBD6	SET 2,(HL)
DDCB05D6	SET 2,(IX+d)
FDCB05D6	SET 2,(IY+d)
CBD7	SET 2,A
CBD0	SET 2,B
CBD1	SET 2,C
CBD2	SET 2,D
CBD3	SET 2,E
CBD4	SET 2,H
CBD5	SET 2,L
CBD8	SET 3,B
CBDE	SET 3,(HL)
DDCB05DE	SET 3,(IX+d)
FDCB05DE	SET 3,(IY+d)
CBDF	SET 3,A
CBD9	SET 3,C
CBDA	SET 3,D
CBDB	SET 3,E
CBDC	SET 3,H
CBDD	SET 3,L
CBE6	SET 4,(HL)
DDCB05E6	SET 4,(IX+d)
FDCB05E6	SET 4,(IY+d)
CBE7	SET 4,A
CBE0	SET 4,B
CBE1	SET 4,C
CBE2	SET 4,D
CBE3	SET 4,E
CBE4	SET 4,H
CBE5	SET 4,L
CBEE	SET 5,(HL)
DDCB05EE	SET 5,(IX+d)
FDCB05EE	SET 5,(IY+d)
CBEF	SET 5,A
CBE8	SET 5,B
CBE9	SET 5,C
CBEA	SET 5,D
CBEB	SET 5,E
CBEC	SET 5,H
CBED	SET 5,L
CBF6	SET 6,(HL)
DDCB05F6	SET 6,(IX+d)
FDCB05F6	SET 6,(IY+d)
CBF7	SET 6,A
CBF0	SET 6,B
CBF1	SET 6,C

CBF2	SET 6,D
CBF3	SET 6,E
CBF4	SET 6,H
CBF5	SET 6,L
CBFE	SET 7,(HL)
DDCB05FE	SET 7,(IX+d)
FDCB05FE	SET 7,(IY+d)
CBFF	SET 7,A
CBF8	SET 7,B
CBF9	SET 7,C
CBFA	SET 7,D
CBFB	SET 7,E
CBFC	SET 7,H
CBFD	SET 7,L
CB26	SLA (HL)
DDCB0526	SLA (IX+d)
FDCB0526	SLA (IY+d)
CB27	SLA A
CB20	SLA B
CB21	SLA C
CB22	SLA D
CB23	SLA E
CB24	SLA H
CB25	SLA L
CB2E	SRA (HL)
DDCB052E	SRA (IX+d)
FDCB052E	SRA (IY+d)
CB2F	SRA A
CB28	SRA B
CB29	SRA C
CB2A	SRA D
CB2B	SRA E
CB2C	SRA H
CB2D	SRA L
CB3E	SRL (HL)
DDCB053E	SRL (IX+d)
FDCB053E	SRL (IY+d)
CB3F	SRL A
CB38	SRL B
CB39	SRL C
CB3A	SRL D
CB3B	SRL E
CB3C	SRL H
CB3D	SRL L
96	SUB (HL)
DD9605	SUB (IX+d)
FD9605	SUB (IY+d)
97	SUB A

90	SUB B
91	SUB C
92	SUB D
93	SUB E
94	SUB H
95	SUB L
D620	SUB N
AE	XOR (HL)
DDAE05	XOR (IX+d)
FDAE05	XOR (IY+d)
AF	XOR A
A8	XOR B
A9	XOR C
AA	XOR D
AB	XOR E
AC	XOR H
AD	XOR L
EE20	XOR N

Example Values

```
nn  EQU  584H
d   EQU  5
n   EQU  20H
e        30H
```

Z80 CPU Register Configuration

MAIN REG SET		ALTERNATE REG SET	
ACCUMULATOR A	FLAGS F	ACCUMULATOR A'	FLAGS F
B	C	B	C'
D	E	D'	E'
H	L	H'	L'

GENERAL PURPOSE REGISTERS

INTERRUPT VECTOR I	MEMORY REFRESH R
INDEX REGISTER IX	
INDEX REGISTER IY	
STACK POINTER SP	
PROGRAM COUNTER PC	

SPECIAL PURPOSE REGISTERS

ASCII Character Set (7-Bit Code)

	MSD	0	1	2	3	4	5	6	7
LSD		000	001	010	011	100	101	110	111
0	0000	NUL	DLE	SP	0	@	P	.	p
1	0001	SOH	DC1	!	1	A	Q	a	q
2	0010	STX	DC2	"	2	B	R	b	r
3	0011	ETX	DC3	=	3	C	S	c	s
4	0100	EOT	DC4	$	4	D	T	d	t
5	0101	ENG	NAK	%	5	E	U	e	u
6	0110	ACK	SYN	&	6	F	V	f	v
7	0111	BEL	ETB	'	7	G	W	g	w
8	1000	BS	CAN	(	8	H	X	h	x
9	1001	HT	EM	)	9	I	Y	i	y
A	1010	LF	SUB	•	:	J	Z	j	z
B	1011	VT	ESC	+	;	K	[	k	{
C	1100	FF	FS	'	<	L	\	l	:
D	1101	CR	GS	—	=	M	]	m	}
E	1110	SO	RS	.	>	N	!	n	–
F	1111	SI	US	/	?	O	–	o	DEL

	D_7							D_0	
Instruction	**S**	**Z**		**H**		**P/V**	**N**	**C**	**Comments**
ADD A s ADC A s	↕	↕	X	↕	X	V	0	↕	8 bit add or add with carry
SUB s SBC A. s. CP s NEG	↕	↕	X	↕	X	V	1	↕	8 bit subtract subtract with carry compare and negate accumulator
AND s	↕	↕	X	1	X	P	0	0	Logical operations
OR s XOR s	↕	↕	X	0	X	P	0	0	
INC s	↕	↕	X	↕	X	V	0	•	8 bit increment
DEC s	↕	↕	X	↕	X	V	1	•	8 bit decrement
ADD DD. ss	•	•	X	X	X	•	0	↕	16 bit add
ADC HL. ss	↕	↕	X	X	X	V	0	↕	16-bit add with carry
SBC HL. ss	↕	↕	X	X	X	V	1	↕	16 bit subtract with carry
RLA. RLCA. RRA. RRCA	•	•	X	0	X	•	0	↕	Rotate accumulator
RL m. RLC m. RR m. RRC m. SLA m. SRA m. SRL m	↕	↕	X	0	X	P	0	↕	Rotate and shift locations
RLD. RRD	↕	↕	X	0	X	P	0	•	Rotate digit left and right
DAA	↕	↕	X	↕	X	P	•	↕	Decimal adjust accumulator
CPL	•	•	X	1	X	•	1	•	Complement accumulator
SCF	•	•	X	0	X	•	0	1	Set carry
CCF	•	•	X	X	X	•	0	↕	Complement carry
IN r (C)	↕	↕	X	0	X	P	0	•	Input register indirect
INI. IND. OUTI OUTD	X	↕	X	X	X	X	1	•	Block input and output Z = 0 if B ≠ 0 otherwise Z = 0
INIR INDR. OTIR. OTDR	X	1	X	X	X	X	1	•	
LDI. LDD	X	X	X	0	X	↕	0	•	Block transfer instructions P/V = 1 if BC ≠ 0. otherwise P/V = 0
LDIR. LDDR	X	X	X	0	X	0	0	•	
CPI CPIR. CPD. CPDR	X	↕	X	X	X	↕	1	•	Block search instructions Z = 1 if A = (HL). otherwise Z = 0 P/V = 1 if BC ≠ 0. otherwise P/V = 0
LD A. I. LD A R	↕	↕	X	0	X	IFF	0	•	The content of the interrupt enable flip-flop (IFF) is copied into the P/V flag
BIT b. s	X	↕	X	1	X	X	0	•	The state of bit b of location s is copied into the Z flag

Symbol	**Operation**
S	Sign flag S = 1 if the MSB of the result is 1
Z	Zero flag Z = 1 if the result of the operation is 0
P/V	Parity or overflow flag Parity (P) and overflow (V) share the same flag Logical operations affect this flag with the parity of the result while arithmetic operations affect this flag with the overflow of the result If P/V holds parity. P/V = 1 if the result of the operation is even. P/V = 0 if result is odd If P/V holds overflow. P/V = 1 if the result of the operation produced an overflow
H	Half-carry flag H = 1 if the add or subtract operation produced a carry into or borrow from bit 4 of the accumulator
N	Add/Subtract flag N = 1 if the previous operation was a subtract
H & N	H and N flags are used in conjunction with the decimal adjust instruction (DAA) to properly correct the result into packed BCD format following addition or subtraction using operands with packed BCD format
C	Carry/Link flag C = 1 if the operation produced a carry from the MSB of the operand or result

Symbol	**Operation**
↕	The flag is affected according to the result of the operation
•	The flag is unchanged by the operation
0	The flag is reset by the operation
1	The flag is set by the operation
X	The flag is a "don't care"
V	P/V flag affected according to the overflow result of the operation
P	P/V flag affected according to the parity result of the operation
r	Any one of the CPU registers A. B. C. D. E. H. L
s	Any 8-bit location for all the addressing modes allowed for the particular instruction
ss	Any 16-bit location for all the addressing modes allowed for that instruction
ii	Any one of the two index registers IX or IY
R	Refresh counter
n	8-bit value in range < 0. 255 >
nn	16-bit value in range < 0. 65535 >

8-Bit Load Group

SOURCE

DESTINATION

		IMPLIED		REGISTER							REGISTER INDIRECT			INDEXED		EXT. ADDR	IMME.
		I	R	A	B	C	D	E	H	L	(HL)	(BC)	(DE)	(1X + d)	(1Y + d)	(nn)	n
REGISTER	A	ED 57	ED 5F	7F	78	79	7A	7B	7C	7D	7E	0A	1A	DD 7E d	FD 7E d	3A n n	3E n
	B			47	40	41	42	43	44	45	46			DD 46 d	FD 46 d		06 n
	C			4F	48	49	4A	4B	4C	4D	4E			DD 4E d	FD 4E d		0E n
	D			57	50	51	52	53	54	55	56			DD 56 d	FD 56 d		16 n
	E			5F	58	59	5A	5B	5C	5D	5E			DD 5E d	FD 5E d		1E n
	H			67	60	61	62	63	64	65	66			DD 66 d	FD 66 d		26 n
	L			6F	68	69	6A	6B	6C	6D	6E			DD 6E d	FD 6E d		2E n
REGISTER INDIRECT	(HL)			77	70	71	72	73	74	75							36 n
	(BC)			02													
	(DE)			12													
INDEXED	(1X + d)			DD 77 d	DD 70 d	DD 71 d	DD 72 d	DD 73 d	DD 74 d	DD 75 d							DD 36 d n
	(1Y + d)			FD 77 d	FD 70 d	FD 71 d	FD 72 d	FD 73 d	FD 74 d	FD 75 d							FD 36 d n
EXTERNAL ADDRESS	(nn)			32 n n													
IMPLIED	I			ED 47													
	R			ED 4F													

Mnemonic	Symbolic Operation	S	Z		H		P/V	N	C	Opcode 76 543 210	Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
LD r, r'	r – r'	•	•	X	•	X	•	•	•	01 r r'		1	1	4	r, r' Reg 000 B 001 C 010 D 011 E 100 H 101 L 111 A
LD r, n	r – n	•	•	X	•	X	•	•	•	00 r 110		2	2	7	
										– n –					
LD r, (HL)	r – (HL)	•	•	X	•	X	•	•	•	01 r 110		1	2	7	
LD r, (IX + d)	r – (IX + d)	•	•	X	•	X	•	•	•	11 011 101	DD	3	5	19	
										01 r 110					
										– d –					
LD r, (IY + d)	r – (IY + d)	•	•	X	•	X	•	•	•	11 111 101	FD	3	5	19	
										01 r 110					
										– d –					
LD (HL), r	(HL) – r	•	•	X	•	X	•	•	•	01 110 r		1	2	7	
LD (IX + d), r	(IX + d) – r	•	•	X	•	X	•	•	•	11 011 101	DD	3	5	19	
										01 110 r					
										– d –					
LD (IY + d), r	(IY + d) – r	•	•	X	•	X	•	•	•	11 111 101	FD	3	5	19	
										01 110 r					
										– d –					
LD (HL), n	(HL) – n	•	•	X	•	X	•	•	•	00 110 110	36	2	3	10	
										– n –					
LD (IX + d), n	(IX + d) – n	•	•	X	•	X	•	•	•	11 011 101	DD	4	5	19	
										00 110 110	36				
										– d –					
										– n –					
LD (IY + d), n	(IY + d) – n	•	•	X	•	X	•	•	•	11 111 101	FD	4	5	19	
										00 110 110	36				
										– d –					
										– n –					
LD A, (BC)	A – (BC)	•	•	X	•	X	•	•	•	00 001 010	0A	1	2	7	
LD A, (DE)	A – (DE)	•	•	X	•	X	•	•	•	00 011 010	1A	1	2	7	
LD A, (nn)	A – (nn)	•	•	X	•	X	•	•	•	00 111 010	3A	3	4	13	
										– n –					
										– n –					
LD (BC), A	(BC) – A	•	•	X	•	X	•	•	•	00 000 010	02	1	2	7	
LD (DE), A	(DE) – A	•	•	X	•	X	•	•	•	00 010 010	12	1	2	7	
LD (nn), A	(nn) – A	•	•	X	•	X	•	•	•	00 110 010	32	3	4	13	
										– n –					
										– n –					
LD A, I	A – I	↕	↕	X	0	X	IFF	0	•	11 101 101	ED	2	2	9	
										01 010 111	57				
LD A, R	A – R	↕	↕	X	0	X	IFF	0	•	11 101 101	ED	2	2	9	
										01 011 111	5F				
LD I, A	I – A	•	•	X	•	X	•	•	•	11 101 101	ED	2	2	9	
										01 000 111	47				
LD R, A	R – A	•	•	X	•	X	•	•	•	11 101 101	ED	2	2	9	
										01 001 111	4F				

NOTES: r, r' means any of the registers A, B, C, D, E, H, L
IFF the content of the interrupt enable flip-flop (IFF) is copied into the P/V flag

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown
↕ = flag is affected according to the result of the operation.

16-Bit Load Group

			SOURCE									
			REGISTER							IMM. EXT.	EXT. ADDR.	REG. INDIR.
			AF	BC	DE	HL	SP	IX	IY	nn	(nn)	(SP)
DESTINATION	REGISTER	AF										F1
		BC								01 n n	ED 4B n n	C1
		DE								11 n n	ED 5B n n	D1
		HL								21 n n	2A n n	E1
		SP				F9		DD F9	FD F9	31 n n	ED 7B n n	
		IX								DD 21 n n	DD 2A n n	DD E1
		IY								FD 21 n n	FD 2A n n	FD E1
	EXTERNAL ADDRESS	(nn)		ED 43 n n	ED 53 n n	22 n n	ED 73 n n	DD 22 n n	FD 22 n n			
PUSH INSTRUCTIONS	REGISTER IND.	(SP)	F5	C5	D5	E5		DD E5	FD E5			

NOTE: The Push & Pop Instructions adjust the SP after every execution.

Mnemonic	Symbolic Operation	S	Z		Flags H		P/V	N	C	Opcode 76 543 210	Hex	No.of Bytes	No.of M Cycles	No.of T States	Comments
LD dd, nn	dd – nn	•	•	X	•	X	•	•	•	00 dd0 001 – n – – n –		3	3	10	dd Pair 00 BC 01 DE 10 HL 11 SP
LD IX, nn	IX – nn	•	•	X	•	X	•	•	•	11 011 101 00 100 001 – n – – n –	DD 21	4	4	14	
LD IY, nn	IY – nn	•	•	X	•	X	•	•	•	11 111 101 00 100 001 – n – – n –	FD 21	4	4	14	
LD HL, (nn)	H – (nn + 1) L – (nn)	•	•	X	•	X	•	•	•	00 101 010 – n – – n –	2A	3	5	16	
LD dd, (nn)	dd_H – (nn + 1) dd_L – (nn)	•	•	X	•	X	•	•	•	11 101 101 01 dd1 011 – n – – n –	ED	4	6	20	
LD IX, (nn)	IX_H – (nn + 1) IX_L – (nn)	•	•	X	•	X	•	•	•	11 011 101 00 101 010 – n – – n –	DD 2A	4	6	20	
LD IY, (nn)	IY_H – (nn + 1) IY_L – (nn)	•	•	X	•	X	•	•	•	11 111 101 00 101 010 – n – – n –	FD 2A	4	6	20	
LD (nn), HL	(nn + 1) – H (nn) – L	•	•	X	•	X	•	•	•	00 100 010 – n – – n –	22	3	5	16	
LD (nn), dd	(nn + 1) – dd_H (nn) – dd_L	•	•	X	•	X	•	•	•	11 101 101 01 dd0 011 – n – – n –	ED	4	6	20	
LD (nn), IX	(nn + 1) – IX_H (nn) – IX_L	•	•	X	•	X	•	•	•	11 011 101 00 100 010 – n – – n –	DD 22	4	6	20	
LD (nn), IY	(nn + 1) – IY_H (nn) – IY_L	•	•	X	•	X	•	•	•	11 111 101 00 100 010 – n – – n –	FD 22	4	6	20	
LD SP, HL	SP – HL	•	•	X	•	X	•	•	•	11 111 001	F9	1	1	6	
LD SP, IX	SP – IX	•	•	X	•	X	•	•	•	11 011 101 11 111 001	DD F9	2	2	10	
LD SP, IY	SP – IY	•	•	X	•	X	•	•	•	11 111 101 11 111 001	FD F9	2	2	10	
PUSH qq	(SP – 2) – qq_L (SP – 1) – qq_H SP – SP – 2	•	•	X	•	X	•	•	•	11 qq0 101		1	3	11	qq Pair 00 BC 01 DE 10 HL 11 AF
PUSH IX	(SP – 2) – IX_L (SP – 1) – IX_H SP – SP – 2	•	•	X	•	X	•	•	•	11 011 101 11 100 101	DD E5	2	4	15	
PUSH IY	(SP – 2) – IY_L (SP – 1) – IY_H SP – SP – 2	•	•	X	•	X	•	•	•	11 111 101 11 100 101	FD E5	2	4	15	
POP qq	qq_H – (SP + 1) qq_L – (SP) SP – SP + 2	•	•	X	•	X	•	•	•	11 qq0 001		1	3	10	
POP IX	IX_H – (SP + 1) IX_L – (SP) SP – SP + 2	•	•	X	•	X	•	•	•	11 011 101 11 100 001	DD E1	2	4	14	
POP IY	IY_H – (SP + 1) IY_L – (SP) SP – SP + 2	•	•	X	•	X	•	•	•	11 111 101 11 100 001	FD E1	2	4	14	

NOTES dd is any of the register pairs BC DE HL SP
qq is any of the register pairs AF BC DE HL
$(PAIR)_H$ $(PAIR)_L$ refer to high order and low order eight bits of the register pair respectively
e g BC_L = C AF_H = A

Flag Notation • = flag not affected 0 = flag reset 1 = flag set X = flag is unknown
‡ = flag is affected according to the result of the operation

Exchange, Block Transfer, and Search Groups

Exchange Group

		IMPLIED ADDRESSING				
		AF'	BC', DE' & HL'	HL	IX	IY
IMPLIED	AF	08				
	BC, DE & HL		D9			
	DE			EB		
REGISTER INDIRECT	(SP)			E3	DD E3	FD E3

Block Transfer Group

			SOURCE	
			REG. INDIR.	
			(HL)	
DESTINATION	REG. INDIR	(DE)	ED A0	'LDI'—Load (DE) – (HL) Inc HL & DE, Dec BC
			ED B0	'LDIR'—Load (DE) – (HL) Inc HL & DE, Dec BC, Repeat until BC = 0
			ED A8	'LDD'—Load (DE) – (HL) Dec HL & DE, Dec BC
			ED B8	'LDDR'—Load (DE) – (HL) Dec HL & DE, Dec BC, Repeat until BC = 0

HL points to source
DE points to destination
BC is byte counter

Block Search Group

SEARCH LOCATION	
REG. INDIR.	
(HL)	
ED A1	'CPI' Inc HL, Dec BC
ED B1	'CPIR'—Inc HL, Dec BC repeat until BC = 0 or find match
ED A9	'CPD'—Dec HL & BC
ED B9	'CPDR'—Dec HL & BC Repeat until BC = 0 or find match

HL points to location in memory
to be compared with accumulator
contents
BC is byte counter

Exchange, Block Transfer, and Search Groups

Mnemonic	Symbolic Operation	S	Z		H		P/V	N	C	Opcode 76 543 210	Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
EX DE HL	DE – HL	•	•	X	•	X	•	•	•	11 101 011	EB	1	1	4	
EX AF AF	AF – AF	•	•	X	•	X	•	•	•	00 001 000	08	1	1	4	
EXX	BC – BC DE – DE HL – HL	•	•	X	•	X	•	•	•	11 011 001	D9	1	1	4	Register bank and auxiliary register bank exchange
EX (SP) HL	H – (SP + 1) L – (SP)	•	•	X	•	X	•	•	•	11 100 011	E3	1	5	19	
EX (SP) IX	IX_H – (SP + 1) IX_L – (SP)	•	•	X	•	X	•	•	•	11 011 101 11 100 011	DD E3	2	6	23	
EX (SP) IY	IY_H – (SP + 1) IY_L – (SP)	•	•	X	•	X	•	•	•	11 111 101 11 100 011	FD E3	2	6	23	
LDI	(DE) – (HL) DE – DE + 1 HL – HL + 1 BC – BC – 1	•	•	X	0	X	① ↕	0	•	11 101 101 10 100 000	ED A0	2	4	16	Load (HL) into (DE) increment the pointers and decrement the byte counter (BC)
LDIR	(DE) – (HL) DE – DE + 1 HL – HL + 1 BC – BC – 1 Repeat until BC = 0	•	•	X	0	X	0	0	•	11 101 101 10 110 000	ED B0	2 2	5 4	21 16	If BC ≠ 0 If BC = 0
LDD	(DE) – (HL) DE – DE – 1 HL – HL – 1 BC – BC – 1	•	•	X	0	X	① ↕	0	•	11 101 101 10 101 000	ED A8	2	4	16	
LDDR	(DE) – (HL) DE – DE – 1 HL – HL – 1 BC – BC – 1 Repeat until BC = 0	•	•	X	0	X	0	0	•	11 101 101 10 111 000	ED B8	2 2	5 4	21 16	If BC ≠ 0 If BC = 0
CPI	A – (HL) HL – HL + 1 BC – BC – 1	↕	② ↕	X	↕	X	① ↕	1	•	11 101 101 10 100 001	ED A1	2	4	16	
CPIR	A – (HL) HL – HL + 1 BC – BC – 1 Repeat until A = (HL) or BC = 0	↕	② ↕	X	↕	X	① ↕	1	•	11 101 101 10 110 001	ED B1	2 2	5 4	21 16	If BC ≠ 0 and A ≠ (HL) If BC = 0 or A = (HL)
CPD	A – (HL) HL – HL – 1 BC – BC – 1	↕	② ↕	X	↕	X	① ↕	1	•	11 101 101 10 101 001	ED A9	2	4	16	
CPDR	A – (HL) HL – HL – 1 BC – BC – 1 Repeat until A = (HL) or BC = 0	↕	② ↕	X	↕	X	① ↕	1	•	11 101 101 10 111 001	ED B9	2 2	5 4	21 16	If BC ≠ 0 and A ≠ (HL) If BC = 0 or A = (HL)

NOTES ① P/V flag is 0 if the result of BC – 1 = 0 otherwise P/V = 1
② Z flag is 1 if A = (HL) otherwise Z = 0

Flag Notation • = flag not affected 0 = flag reset 1 = flag set X = flag is unknown
↕ = flag is affected according to the result of the operation

8-Bit Arithmetic and Logical Group

SOURCE

	REGISTER ADDRESSING							REG. INDIR.	INDEXED		IMMED.
	A	B	C	D	E	H	L	(HL)	(IX + d)	(IY + d)	n
'ADD'	87	80	81	82	83	84	85	86	DD 86 d	FD 86 d	C6 n
ADD w CARRY 'ADC'	8F	88	89	8A	8B	8C	8D	8E	DD 8E d	FD 8E d	CE n
SUBTRACT 'SUB'	97	90	91	92	93	94	95	96	DD 96 d	FD 96 d	D6 n
SUB w CARRY 'SBC'	9F	98	99	9A	9B	9C	9D	9E	DD 9E d	FD 9E d	DE n
'AND'	A7	A0	A1	A2	A3	A4	A5	A6	DD A6 d	FD A6 d	E6 n
'XOR'	AF	A8	A9	AA	AB	AC	AD	AE	DD AE d	FD AE d	EE n
'OR'	B7	B0	B1	B2	B3	B4	B5	B6	DD B6 d	FD B6 d	F6 n
COMPARE 'CP'	BF	B8	B9	BA	BB	BC	BD	BE	DD BE d	FD BE d	FE n
INCREMENT 'INC'	3C	04	0C	14	1C	24	2C	34	DD 34 d	FD 34 d	
DECREMENT 'DEC'	3D	05	0D	15	1D	25	2D	35	DD 35 d	FD 35 d	

Mnemonic	Symbolic Operation	S	Z		Flags H		P/V	N	C	Opcode 76 543 210	Hex	No.of Bytes	No.of M Cycles	No.of T States	Comments
ADD A, r	A ← A + r	↕	↕	X	↕	X	V	0	↕	10 $\boxed{000}$ r		1	1	4	r Reg
ADD A, n	A ← A + n	↕	↕	X	↕	X	V	0	↕	11 $\boxed{000}$ 110		2	2	7	000 B
										– n –					001 C 010 D
ADD A, (HL)	A ← A + (HL)	↕	↕	X	↕	X	V	0	↕	10 $\boxed{000}$ 110		1	2	7	011 E
ADD A, (IX + d)	A ← A + (IX + d)	↕	↕	X	↕	X	V	0	↕	11 011 101	DD	3	5	19	100 H
										10 $\boxed{000}$ 110					101 L
										– d –					111 A
ADD A, (IY + d)	A ← A + (IY + d)	↕	↕	X	↕	X	V	0	↕	11 111 101	FD	3	5	19	
										10 $\boxed{000}$ 110					
										– d –					
ADC A, s	A ← A + s + CY	↕	↕	X	↕	X	V	0	↕	$\boxed{001}$					s is any of r, n, (HL), (IX + d), (IY + d) as shown for ADD instruction The indicated bits replace the $\boxed{000}$ in the ADD set above
SUB s	A ← A − s	↕	↕	X	↕	X	V	1	↕	$\boxed{010}$					
SBC A, s	A ← A − s − CY	↕	↕	X	↕	X	V	1	↕	$\boxed{011}$					
AND s	A ← A ∧ s	↕	↕	X	1	X	P	0	0	$\boxed{100}$					
OR s	A ← A ∨ s	↕	↕	X	0	X	P	0	0	$\boxed{110}$					
XOR s	A ← A ● s	↕	↕	X	0	X	P	0	0	$\boxed{101}$					
CP s	A − s	↕	↕	X	↕	X	V	1	↕	$\boxed{111}$					
INC r	r ← r + 1	↕	↕	X	↕	X	V	0	•	00 r $\boxed{100}$		1	1	4	
INC (HL)	(HL) ← (HL) + 1	↕	↕	X	↕	X	V	0	•	00 110 $\boxed{100}$		1	3	11	
INC (IX + d)	(IX + d) ← (IX + d) + 1	↕	↕	X	↕	X	V	0	•	11 011 101 00 110 $\boxed{100}$ – d –	DD	3	6	23	
INC (IY + d)	(IY + d) ← (IY + d) + 1	↕	↕	X	↕	X	V	0	•	11 111 101 00 110 $\boxed{100}$ – d –	FD	3	6	23	m is any of r, (HL), (IX + d), (IY + d) as shown for INC DEC same format and states as INC Replace $\boxed{100}$ with $\boxed{101}$ in opcode
DEC m	m ← m − 1	↕	↕	X	↕	X	V	1	•	$\boxed{101}$					

NOTES The V symbol in the P/V flag column indicates that the P/V flag contains the overflow of the result of the operation Similarly the P symbol indicates parity V = 1 means overflow V = 0 means not overflow P = 1 means parity of the result is even P = 0 means parity of the result is odd

Flag Notation • = flag not affected, 0 = flag reset 1 = flag set X = flag is unknown.
↕ = flag is affected according to the result of the operation

General Purpose Arithmetic

Decimal Adjust Acc. 'DAA'	27
Complement Acc. 'CPL'	2F
Negate Acc. 'NEG' (2's compllement)	ED 44
Complement Carry Flag. 'CCF'	3F
Set Carry Flag. 'SCF'	37

Miscellaneous CPU Control

'NOP'	00	
'HALT'	76	
DISABLE INT '(DI)'	F3	
ENABLE INT '(EI)'	FB	
SET INT MODE 0 'IM 0'	ED 46	8080A MODE
SET INT MODE 1 'IM 1'	ED 56	RESTART TO LOCATION 0038_H
SET INT MODE 2 'IM 2'	ED 5E	INDIRECT CALL USING REGISTER 1 AND 8 BITS FROM INTERRUPTING DEVICE AS A POINTER.

Mnemonic	Symbolic Operation	S	Z		Flags H		P/V	N	C	Opcode 76	543	210	Hex	No.of Bytes	No.of M Cycles	No.of T States	Comments
DAA	Converts acc content into packed BCD following add or subtract with packed BCD operands	↕	↕	X	↕	X	P	•	↕	00	100	111	27	1	1	4	Decimal adjust accumulator
CPL	A ← $\overline{A}$	•	•	X	1	X	•	1	•	00	101	111	2F	1	1	4	Complement accumulator (one's complement)
NEG	A ← 0 − A	↕	↕	X	↕	X	V	1	↕	11 01	101 000	101 100	ED 44	2	2	8	Negate acc (two's complement)
CCF	CY ← $\overline{CY}$	•	•	X	X	X	•	0	↕	00	111	111	3F	1	1	4	Complement carry flag
SCF	CY ← 1	•	•	X	0	X	•	0	1	00	110	111	37	1	1	4	Set carry flag
NOP	No operation	•	•	X	•	X	•	•	•	00	000	000	00	1	1	4	
HALT	CPU halted	•	•	X	•	X	•	•	•	01	110	110	76	1	1	4	
DI ★	IFF ← 0	•	•	X	•	X	•	•	•	11	110	011	F3	1	1	4	
EI ★	IFF ← 1	•	•	X	•	X	•	•	•	11	111	011	FB	1	1	4	
IM 0	Set interrupt mode 0	•	•	X	•	X	•	•	•	11 01	101 000	101 110	ED 46	2	2	8	
IM 1	Set interrupt mode 1	•	•	X	•	X	•	•	•	11 01	101 010	101 110	ED 56	2	2	8	
IM 2	Set interrupt mode 2	•	•	X	•	X	•	•	•	11 01	101 011	101 110	ED 5E	2	2	8	

NOTES IFF indicates the interrupt enable flip flop
CY indicates the carry flip flop
★ indicates interrupts are not sampled at the end of EI or DI

Flag Notation • = flag not affected 0 = flag reset 1 = flag set X = flag is unknown
↕ = flag is affected according to the result of the operation

16-Bit Arithmetic Group

SOURCE

			BC	DE	HL	SP	IX	IY
DESTINATION	'ADD'	HL	09	19	29	39		
		IX	DD 09	DD 19		DD 39	DD 29	
		IY	FD 09	FD 19		FD 39		FD 29
	ADD WITH CARRY AND SET FLAGS 'ADC'	HL	ED 4A	ED 5A	ED 6A	ED 7A		
	SUB WITH CARRY AND SET FLAGS 'SBC'	HL	ED 42	ED 52	ED 62	ED 72		
	INCREMENT 'INC'		03	13	23	33	DD 23	FD 23
	DECREMENT 'DEC'		08	1B	2B	3B	DD 2B	FD 2B

	Symbolic				Flags					Opcode		No.of	No.of M	No.of T	
Mnemonic	**Operation**	**S**	**Z**		**H**		**P/V**	**N**	**C**	**76 543 210**	**Hex**	**Bytes**	**Cycles**	**States**	**Comments**
ADD HL, ss	HL ← HL + ss	•	•	X	X	X	•	0	↕	00 ss1 001		1	3	11	ss Reg
															00 BC
ADC HL, ss	HL ← HL + ss + CY	↕	↕	X	X	X	V	0	↕	11 101 101	ED	2	4	15	01 DE
										01 ss1 010					10 HL
															11 SP
SBC HL, ss	HL ← HL – ss – CY	↕	↕	X	X	X	V	1	↕	11 101 101	ED	2	4	15	
										01 ss0 010					
ADD IX, pp	IX ← IX + pp	•	•	X	X	X	•	0	↕	11 011 101	DD	2	4	15	pp Reg
										01 pp1 001					00 BC
															01 DE
															10 IX
															11 SP
ADD IY, rr	IY ← IY + rr	•	•	X	X	X	•	0	↕	11 111 101	FD	2	4	15	rr Reg
										00 rr1 001					00 BC
															01 DE
															10 IY
															11 SP
INC ss	ss ← ss + 1	•	•	X	•	X	•	•	•	00 ss0 011		1	1	6	
INC IX	IX ← IX + 1	•	•	X	•	X	•	•	•	11 011 101	DD	2	2	10	
										00 100 011	23				
INC IY	IY ← IY + 1	•	•	X	•	X	•	•	•	11 111 101	FD	2	2	10	
										00 100 011	23				
DEC ss	ss ← ss – 1	•	•	X	•	X	•	•	•	00 ss1 011		1	1	6	
DEC IX	IX ← IX – 1	•	•	X	•	X	•	•	•	11 011 101	DD	2	2	10	
										00 101 011	2B				
DEC IY	IY ← IY – 1	•	•	X	•	X	•	•	•	11 111 101	FD	2	2	10	
										00 101 011	2B				

NOTES ss is any of the register pairs BC DE HL SP
pp is any of the register pairs BC, DE IX SP
rr is any of the register pairs BC DE IY SP

Flag Notation • = flag not affected 0 = flag reset 1 = flag set X = flag is unknown
↕ = flag is affected according to the result of the operation

			CONDITION									
			UN COND	CARRY	NON CARRY	ZERO	NON ZERO	PARITY EVEN	PARITY ODD	SIGN NEG	SIGN POS	REG B ≠ 0
JUMP 'JP'	IMMEDIATE EXTENSION	nn	C3 n n	DA n n	D2 n n	CA n n	C2 n n	EA n n	E2 n n	FA n n	F2 n n	
JUMP 'JP'	RELATIVE	PC + e	18 e - 2	38 e - 2	30 e - 2	28 e - 2	20 e - 2					
JUMP 'JP'	REGISTER INDIRECT	(HL)	E9									
JUMP 'JR'		(IX)	DD E9									
JUMP 'JP'		(IY)	FD E9									
DECREMENT B. JUMP IF NON ZERO 'DJNZ'	RELATIVE	PC·e										10 e - 2

Mnemonic	Symbolic Operation	S	Z	Flags	H		P/V	N	C	Opcode 76 543 210	Hex	No.of Bytes	No.of M Cycles	No.of T States	Comments
JP nn	PC – nn	•	•	X	•	X	•	•	•	11 000 011 – n – – n –	C3	3	3	10	
JP cc. nn	If condition cc is true PC – nn. otherwise continue	•	•	X	•	X	•	•	•	11 cc 010 – n – – n –		3	3	10	cc Condition 000 NZ non-zero 001 Z zero 010 NC non-carry 011 C carry 100 PO parity odd 101 PE parity even 110 P sign positive 111 M sign negative
JR e	PC – PC+e	•	•	X	•	X	•	•	•	00 011 000 – e–2 –	18	2	3	12	
JR C. e	If C = 0. continue	•	•	X	•	X	•	•	•	00 111 000 – e–2 –	38	2	2	7	If condition not met
	If C = 1. PC – PC+e											2	3	12	If condition is met
JR NC. e	If C = 1. continue	•	•	X	•	X	•	•	•	00 110 000 – e–2 –	30	2	2	7	If condition not met
	If C = 0. PC – PC+e											2	3	12	If condition is met
JP Z. e	If Z = 0 continue	•	•	X	•	X	•	•	•	00 101 000 – e–2 –	28	2	2	7	If condition not met
	If Z = 1 PC – PC+e											2	3	12	If condition is met
JR NZ. e	If Z = 1. continue	•	•	X	•	X	•	•	•	00 100 000 – e–2 –	20	2	2	7	If condition not met
	If Z = 0. PC – PC+e											2	3	12	If condition is met
JP (HL)	PC – HL	•	•	X	•	X	•	•	•	11 101 001	E9	1	1	4	
JP (IX)	PC – IX	•	•	X	•	X	•	•	•	11 011 101 11 101 001	DD E9	2	2	8	
JP (IY)	PC – IY	•	•	X	•	X	•	•	•	11 111 101 11 101 001	FD E9	2	2	8	
DJNZ. e	B – B–1 If B = 0. continue	•	•	X	•	X	•	•	•	00 010 000 – e–2 –	10	2	2	8	If B = 0
	If B ≠ 0. PC – PC+e											2	3	13	If B ≠ 0

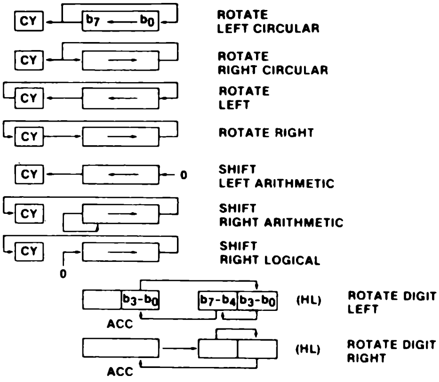
NOTES e represents the extension in the relative addressing mode
e is a signed two s complement number in the range < – 126 129 >
e – 2 in the opcode provides an effective address of pc + e as PC is incremented
by 2 prior to the addition of e

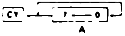
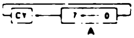
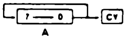
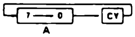
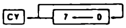
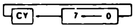
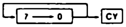
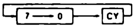
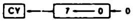
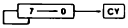
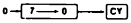
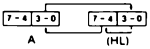
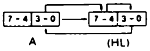
Flag Notation • = flag not affected 0 = flag reset 1 = flag set X = flag is unknown
↕ = flag is affected according to the result of the operation

Rotate and Shift Group

TYPE OF ROTATE OR SHIFT	SOURCE AND DESTINATION									
	A	B	C	D	E	H	L	(HL)	(1X + d)	(1Y + d)
'RLC'	CB 07	CB 00	CB 01	CB 02	CB 03	CB 04	CB 05	CB 06	DD CB d 06	FD CB d 06
'RRC'	CB 0F	CB 08	CB 09	CB 0A	CB 0B	CB 0C	CB 0D	CB 0E	DD CB d 0E	FD CB d 0E
'RL'	CB 17	CB 10	CB 11	CB 12	CB 13	CB 14	CB 15	CB 16	DD CB d 16	FD CB d 16
'RR'	CB 1F	CB 18	CB 19	CB 1A	CB 1B	CB 1C	CB 1D	CB 1E	DD CB d 1E	FD CB d 1E
'SLA'	CB 27	CB 20	CB 21	CB 22	CB 23	CB 24	CB 25	CB 26	DD CB d 26	FD CB d 26
'SRA'	CB 2F	CB 28	CB 29	CB 2A	CB 2B	CB 2C	CB 2D	CB 2E	DD CB d 2E	FD CB d 2E
'SRL'	CB 3F	CB 38	CB 39	CB 3A	CB 3B	CB 3C	CB 3D	CB 3E	DD CB d 3E	FD CB d 3E
'RLD'								ED 6F		
'RRD'								ED 67		

	A
'RLCA'	07
'RRCA'	0F
'RLA'	17
'RRA'	1F



Mnemonic	Symbolic Operation	S	Z		Flags H		P/V	N	C	Opcode 76 543 210	Hex	No.of Bytes	No.of M Cycles	No.of T States	Comments
RLCA		•	•	X	0	X	•	0	↕	00 000 111	07	1	1	4	Rotate left circular accumulator
RLA		•	•	X	0	X	•	0	↕	00 010 111	17	1	1	4	Rotate left accumulator
RRCA		•	•	X	0	X	•	0	↕	00 001 111	0F	1	1	4	Rotate right circular accumulator
RRA		•	•	X	0	X	•	0	↕	00 011 111	1F	1	1	4	Rotate right accumulator
RLC r		↕	↕	X	0	X	P	0	↕	11 001 011 00 $\boxed{000}$ r	CB	2	2	8	Rotate left circular register r
RLC (HL)		↕	↕	X	0	X	P	0	↕	11 001 011 00 $\boxed{000}$ 110	CB	2	4	15	r Reg 000 B 001 C 010 D 011 E 100 H 101 L 111 A
RLC (IX + d)	 r,(HL),(IX + d),(IY + d)	↕	↕	X	0	X	P	0	↕	11 011 101 11 001 011 – d – 00 $\boxed{000}$ 110	DD CB	4	6	23	
RLC (IY + d)		↕	↕	X	0	X	P	0	↕	11 111 101 11 001 011 – d – 00 $\boxed{000}$ 110	FD CB	4	6	23	
RL m	 m ≡ r,(HL),(IX + d),(IY + d)	↕	↕	X	0	X	P	0	↕	$\boxed{010}$					Instruction format and states are as shown for RLC s To form new opcode replace $\boxed{000}$ or RLC s with shown code
RRC m	 m ≡ r,(HL),(IX + d),(IY + d)	↕	↕	X	0	X	P	0	↕	$\boxed{001}$					
RR m	 m ≡ r,(HL),(IX + d),(IY + d)	↕	↕	X	0	X	P	0	↕	$\boxed{011}$					
SLA m	 m ≡ r,(HL),(IX + d),(IY + d)	↕	↕	X	0	X	P	0	↕	$\boxed{100}$					
SRA m	 m ≡ r,(HL),(IX + d),(IY + d)	↕	↕	X	0	X	P	0	↕	$\boxed{101}$					
SRL m	 m ≡ r,(HL),(IX + d),(IY + d)	↕	↕	X	0	X	P	0	↕	$\boxed{111}$					
RLD		↕	↕	X	0	X	P	0	•	11 101 101 01 101 111	ED 6F	2	5	18	Rotate digit left and right between the accumulator and location (HL)
RRD		↕	↕	X	0	X	P	0	•	11 101 101 01 100 111	ED 67			18	The content of the upper half of the accumulator is unaffected

NOTES e represents the extension in the relative addressing mode
e is a signed two s complement number in the range < -126 129>
e-2 in the opcode provides an effective address of pc + e as PC is incremented by 2 prior to the addition of e

Flag Notation • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown
↕ = flag is affected according to the result of the operation

Bit Manipulation Group

		REGISTER ADDRESSING							REG. INDIR.	INDEXED	
	BIT	A	B	C	D	E	H	L	(HL)	(IX + d)	(IY + d)
TEST 'BIT'	0	CB 47	CB 40	CB 41	CB 42	CB 43	CB 44	CB 45	CB 46	DD CB d 46	FD CB d 46
	1	CB 4F	CB 48	CB 49	CB 4A	CB 4B	CB 4C	CB 4D	CB 4E	DD CB d 4E	FD CB d 4E
	2	CB 57	CB 50	CB 51	CB 52	CB 53	CB 54	CB 55	CB 56	DD CB d 56	FD CB d 56
	3	CB 5F	CB 58	CB 59	CB 5A	CB 5B	CB 5C	CB 5D	CB 5E	DD CB d 5E	FD CB d 5E
	4	CB 67	CB 60	CB 61	CB 62	CB 63	CB 64	CB 65	CB 66	DD CB d 66	FD CB d 66
	5	CB 6F	CB 68	CB 69	CB 6A	CB 6B	CB 6C	CB 6D	CB 6E	DD CB d 6E	FD CB d 6E
	6	CB 77	CB 70	CB 71	CB 72	CB 73	CB 74	CB 75	CB 76	DD CB d 76	FD CB d 76
	7	CB 7F	CB 78	CB 79	CB 7A	CB 7B	CB 7C	CB 7D	CB 7E	DD CB d 7E	FD CB d 7E
RESET BIT 'RES'	0	CB 87	CB 80	CB 81	CB 82	CB 83	CB 84	CB 85	CB 86	DD CB d 86	FD CB d 86
	1	CB 8F	CB 88	CB 89	CB 8A	CB 8B	CB 8C	CB 8D	CB 8E	DD CB d 8E	FD CB d 8E
	2	CB 97	CB 90	CB 91	CB 92	CB 93	CB 94	CB 95	CB 96	DD CB d 96	FD CB d 96
	3	CB 9F	CB 98	CB 99	CB 9A	CB 9B	CB 9C	CB 9D	CB 9E	DD CB d 9E	FD CB d 9E
	4	CB A7	CB A0	CB A1	CB A2	CB A3	CB A4	CB A5	CB A6	DD CB d A6	FD CB d A6
	5	CB AF	CB A8	CB A9	CB AA	CB AB	CB AC	CB AD	CB AE	DD CB d AE	FD CB d AE
	6	CB B7	CB B0	CB B1	CB B2	CB B3	CB B4	CB B5	CB B6	DD CB d B6	FD CB d B6
	7	CB BF	CB B8	CB B9	CB BA	CB BB	CB BC	CB BD	CB BE	DD CB d BE	FD CB d BE
SET BIT 'SET'	0	CB C7	CB C0	CB C1	CB C2	CB C3	CB C4	CB C5	CB C6	DD CB d C6	FD CB d C6
	1	CB CF	CB C8	CB C9	CB CA	CB CB	CB CC	CB CD	CB CE	DD CB d CE	FD CB d CE
	2	CB D7	CB D0	CB D1	CB D2	CB D3	CB D4	CB D5	CB D6	DD CB d D6	FD CB d D6
	3	CB DF	CB D8	CB D9	CB DA	CB DB	CB DC	CB DD	CB DE	DD CB d DE	FD CB d DE
	4	CB E7	CB E0	CB E1	CB E2	CB E3	CB E4	CB E5	CB E6	DD CB d E6	FD CB d E6
	5	CB EF	CB E8	CB E9	CB EA	CB EB	CB EC	CB ED	CB EE	DD CB d EE	FD CB d EE
	6	CB F7	CB F0	CB F1	CB F2	CB F3	CB F4	CB F5	CB F6	DD CB d F6	FD CB d F6
	7	CB FF	CB F8	CB F9	CB FA	CB FB	CB FC	CB FD	CB FE	DD CB d FE	FD CB d FE

Bit Manipulation Group

	Symbolic				Flags					Opcode		No.of	No.of M	No.of T	
Mnemonic	Operation	S	Z		H		P/V	N	C	76 543 210	Hex	Bytes	Cycles	States	Comments
BIT b r	$Z - \overline{r}_b$	X	↕	X	1	X	X	0	•	11 001 011 01 b r	CB	2	2	6	r Reg. 000 B 001 C 010 D 011 E 100 H 101 L 111 A
BIT b (HL)	$Z - (\overline{HL})_b$	X	↕	X	1	X	X	0	•	11 001 011 01 b 110	CB	2	3	12	
BIT b (IX + d)$_b$	$Z - (\overline{IX + d})_b$	X	↕	X	1	X	X	0	•	11 011 101 11 001 011 – d – 01 b 110	DD CB	4	5	20	
BIT b (IY + d)$_b$	$Z - (\overline{IY + d})_b$	X	↕	X	1	X	X	0	•	11 111 101 11 001 011 – d – 01 b 110	FD CB	4	5	20	b Bit Tested 000 0 001 1 010 2 011 3 100 4 101 5 110 6 111 7
SET b r	$r_b - 1$	•	•	X	•	X	•	•	•	11 001 011 $\boxed{11}$ b r	CB	2	2	8	
SET b (HL)	$(HL)_b - 1$	•	•	X	•	X	•	•	•	11 001 011 $\boxed{11}$ b 110	CB	2	4	15	
SET b (IX + d)	$(IX + d)_b - 1$	•	•	X	•	X	•	•	•	11 011 101 11 001 011 – d – $\boxed{11}$ b 110	DD CB	4	6	23	
SET b (IY + d)	$(IY + d)_b - 1$	•	•	X	•	X	•	•	•	11 111 101 11 001 011 – d – $\boxed{11}$ b 110	FD CB	4	6	23	
RES b m	$m_b - 0$ m ≡ r, (HL), (IX + d), (IY + d)	•	•	X	•	X	•	•	•	$\boxed{10}$					To form new opcode replace $\boxed{11}$ of SET b, s with $\boxed{10}$ Flags and time states for SET instruction

NOTES The notation m_b indicates bit b (0 to 7) or location m

Flag Notation: • = flag not affected 0 = flag reset 1 = flag set X = flag is unknown.
↕ = flag is affected according to the result of the operation

Input and Output Groups

Input Group

				PORT ADDRESS		
				IMMED.	REG INDIR.	
				n	(C)	
INPUT DESTINATION	INPUT 'IN'	REGISTER ADDRESSING	A	DB n	ED 78	
			B		ED 40	
			C		ED 48	
			D		ED 50	
			E		ED 58	
			H		ED 60	
			L		ED 68	
	'INI'-INPUT & Inc HL, Dec B	REGISTER INDIRECT	(HL)		ED A2	BLOCK INPUT COMMANDS
	'INIR'-INP, Inc HL, Dec B, REPEAT IF B ≠ 0				ED B2	
	'IND'-INPUT & Dec HL, Dec B				ED AA	
	'INDR'-INPUT, Dec HL Dec B, REPEAT IF B ≠ 0				ED BA	

Output Group

			SOURCE								
			REGISTER							REG. IND.	
			A	B	C	D	E	H	L	(HL)	
'OUT	IMMED.	n	D3 n								
	REG. IND.	(C)	ED 79	ED 41	ED 49	ED 51	ED 59	ED 61	ED 69		
'OUTI'-OUTPUT Inc HL Dec b	REG. IND.	(C)								ED A3	
'OTIR'-OUTPUT, Inc HL, Dec B, REPEAT IF B ≠ 0	REG. IND.	(C)								ED B3	BLOCK OUTPUT COMMANDS
'OUTD'-OUTPUT Dec HL Dec B	REG. IND.	(C)								ED AB	
'OTDR'-OUTPUT, Dec HL Dec B, REPEAT IF B ≠ 0	REG. IND.	(C)								ED BB	
	PORT DESTINATION ADDRESS										

Mnemonic	Symbolic Operation	S	Z		H		P/V	N	C	Opcode 76 543 210	Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
IN A, (n)	A ← (n)	•	•	X	•	X	•	•	•	11 011 011 ← n →	DB	2	3	11	n to $A_0 \sim A_7$ Acc to $A_8 \sim A_{15}$
IN r (C)	r ← (C) if r = 110 only the flags will be affected	↕	↕	X	↕	X	P	0	•	11 101 101 01 r 000	ED	2	3	12	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
INI	(HL) ← (C) B ← B − 1 HL ← HL + 1	X	① ↕	X	X	X	X	1	•	11 101 101 10 100 010	ED A2	2	4	16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
INIR	(HL) ← (C) B ← B − 1 HL ← HL + 1 Repeat until B = 0	X	1	X	X	X	X	1	•	11 101 101 10 110 010	ED B2	2 2	5 (If B ≠ 0) 4 (If B = 0)	21 16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
IND	(HL) ← (C) B ← B − 1 HL ← HL − 1	X	① ↕	X	X	X	X	1	•	11 101 101 10 101 010	ED AA	2	4	16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
INDR	(HL) ← (C) B ← B − 1 HL ← HL − 1 Repeat until B = 0	X	1	X	X	X	X	1	•	11 101 101 10 111 010	ED BA	2 2	5 (If B ≠ 0) 4 (If 9 = 0)	21 16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
OUT (n), A	(n) ← A	•	•	X	•	X	•	•	•	11 010 011 ← n →	D3	2	3	11	n to $A_0 \sim A_7$ Acc to $A_8 \sim A_{15}$
OUT (C), r	(C) ← r	•	•	X	•	X	•	•	•	11 101 101 01 r 001	ED	2	3	12	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
OUTI	(C) ← (HL) B ← B − 1 HL ← HL + 1	X	① ↕	X	X	X	X	1	•	11 101 101 10 100 011	ED A3	2	4	16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
OTIR	(C) ← (HL) B ← B − 1 HL ← HL + 1 Repeat until B = 0	X	1	X	X	X	X	1	•	11 101 101 10 110 011	ED B3	2 2	5 (If B ≠ 0) 4 (If B = 0)	21 16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
OUTD	(C) ← (HL) B ← B − 1 HL ← HL − 1	X	① ↕	X	X	X	X	1	•	11 101 101 10 101 011	ED AB	2	4	16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
OTDR	(C) ← (HL) B ← B − 1 HL ← HL − 1 Repeat until B = 0	X	1	X	X	X	X	1	•	11 101 101 10 111 011	ED	2 2	5 (If B ≠ 0) 4 (If B = 0)	21 16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$

NOTE ① If the result of B − 1 is zero the Z flag is set, otherwise it is reset

Flag Notation • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown
↕ = flag is affected according to the result of the operation

Call and Return Groups and Restart

Call and Return Group

			CONDITION									
			UN COND.	CARRY	NON CARRY	ZERO	NON ZERO	PARITY EVEN	PARITY ODD	SIGN NEG.	SIGN POS.	REG. B≠0
'CALL'	IMMEDIATE EXTENSION	nn	CD n n	DC n n	D4 n n	CC n n	C4 n n	EC n n	E4 n n	FC n n	F4 n n	
RETURN 'RET'	REGISTER INDIRECT	(SP) (SP + 1)	C9	D8	D0	C8	C0	E8	E0	F8	F0	
RETURN FROM INT 'RETI'	REGISTER INDIRECT	(SP) (SP + 1)	ED 4D									
RETURN FROM NON MASKABLE INT 'RETN'	REGISTER INDIRECT	(SP) (SP + 1)	ED 45									

Note: Certain flags have more than one purpose. Refer to the Z80 CPU Technical Manual fo. details.

Restart Group

		OP CODE	
CALL ADDRESS	0000_H	C7	'RST 0'
	0008_H	CF	'RST 8'
	0010_H	D7	'RST 16'
	0018_H	DF	'RST 24'
	0020_H	E7	'RST 32'
	0028_H	EF	'RST 40'
	0030_H	F7	'RST 48'
	0038_H	FF	'RST 56'

Mnemonic	Symbolic Operation	S	Z		Flags H		P/V	N	C	Opcode 76 543 210	Hex	No.of Bytes	No.of M Cycles	No.of T States	Comments
CALL nn	(SP – 1) – PC_H (SP – 2) – PC_L PC – nn	•	•	X	•	X	•	•	•	11 001 101 – n – – n –	CD	3	5	17	
CALL cc, nn	If condition cc is false continue, otherwise, same as CALL nn	•	•	X	•	X	•	•	•	11 cc 100 – n – – n –		3 3	3 5	10 17	If cc is false If cc is true
RET	PC_L – (SP) PC_H – (SP + 1)	•	•	X	•	X	•	•	•	11 001 001	C9	1	3	10	
RET cc	If condition cc is false continue, otherwise same as RET	•	•	X	•	X	•	•	•	11 cc 000		1 1	1 3	5 11	If cc is false If cc is true cc Condition 000 NZ non-zero 001 Z zero 010 NC non-carry 011 C carry 100 PO parity odd 101 PE parity even 110 P sign positive 111 M sign negative
RETI	Return from interrupt	•	•	X	•	X	•	•	•	11 101 101 01 001 101	ED 4D	2	4	14	
RETN[1]	Return from non-maskable interrupt	•	•	X	•	X	•	•	•	11 101 101 01 000 101	ED 45	2	4	14	
RST p	(SP – 1) – PC_H (SP – 2) – PC_L PC_H – 0 PC_L – p	•	•	X	•	X	•	•	•	11 t 111		1	3	11	t p 000 00H 001 08H 010 10H 011 18H 100 20H 101 28H 110 30H 111 38H

NOTE [1]RETN loads IFF_2 – IFF_1

Flag Notation • = flag not affected 0 = flag reset 1 = flag set X = flag is unknown
ɪ = flag is affected according to the result of the operation

MASKABLE ($\overline{INT}$)
MODE 0

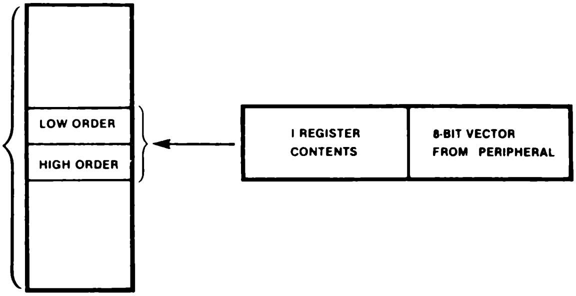
PLACE INSTRUCTION ONTO DATA BUS DURING $\overline{INTA}$ = $\overline{MI}$ • $\overline{IORQ}$ LIKE 8080A

MODE 1

RESTART TO 38_H OR 56_{10} ('RST 56')

MODE 2

USED BY Z80 PERIPHERALS


INTERRUPT SERVICE ROUTINE STARTING ADDRESS TABLE
LOW ORDER
HIGH ORDER
I REGISTER CONTENTS
8-BIT VECTOR FROM PERIPHERAL

NON MASKABLE ($\overline{NMI}$)

RESTART TO 66_H OR 102_{10}

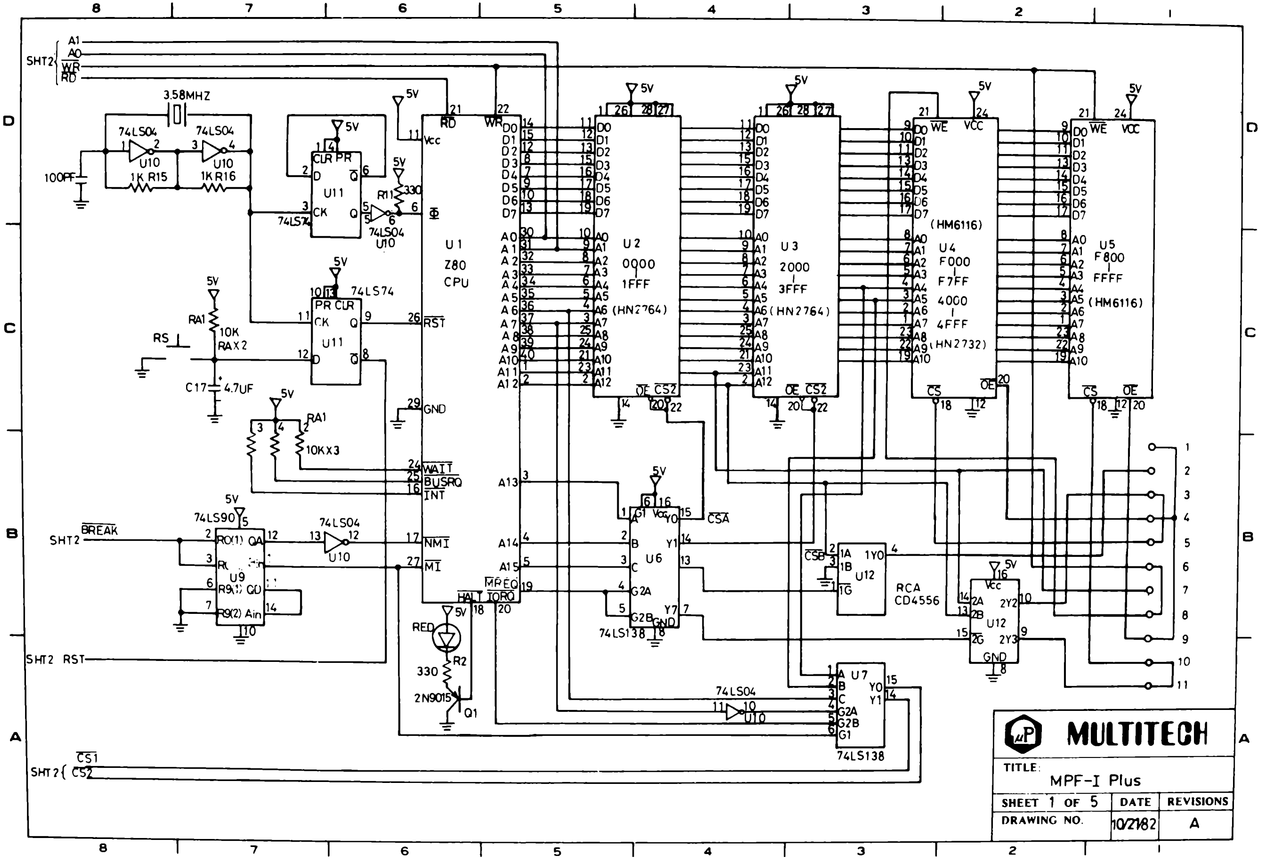
INTERRUPT ENABLE / DISABLE FLIP-FLOPS

ACTION	IFF_1	IFF_2	
CPU RESET	0	0	
DI	0	0	
EI	1	1	
LD A, I	•	•	IFF_2 - PARITY FLAG
LD A, R	•	•	IFF_2 - PARITY FLAG
ACCEPT $\overline{NMI}$	0	•	
RETN	IFF_2	•	IFF_2 - IFF_1
ACCEPT $\overline{INT}$	0	0	
RETI	•	•	

"•" INDICATES NO CHANGE

Appendix D

MPF-IP Schematic

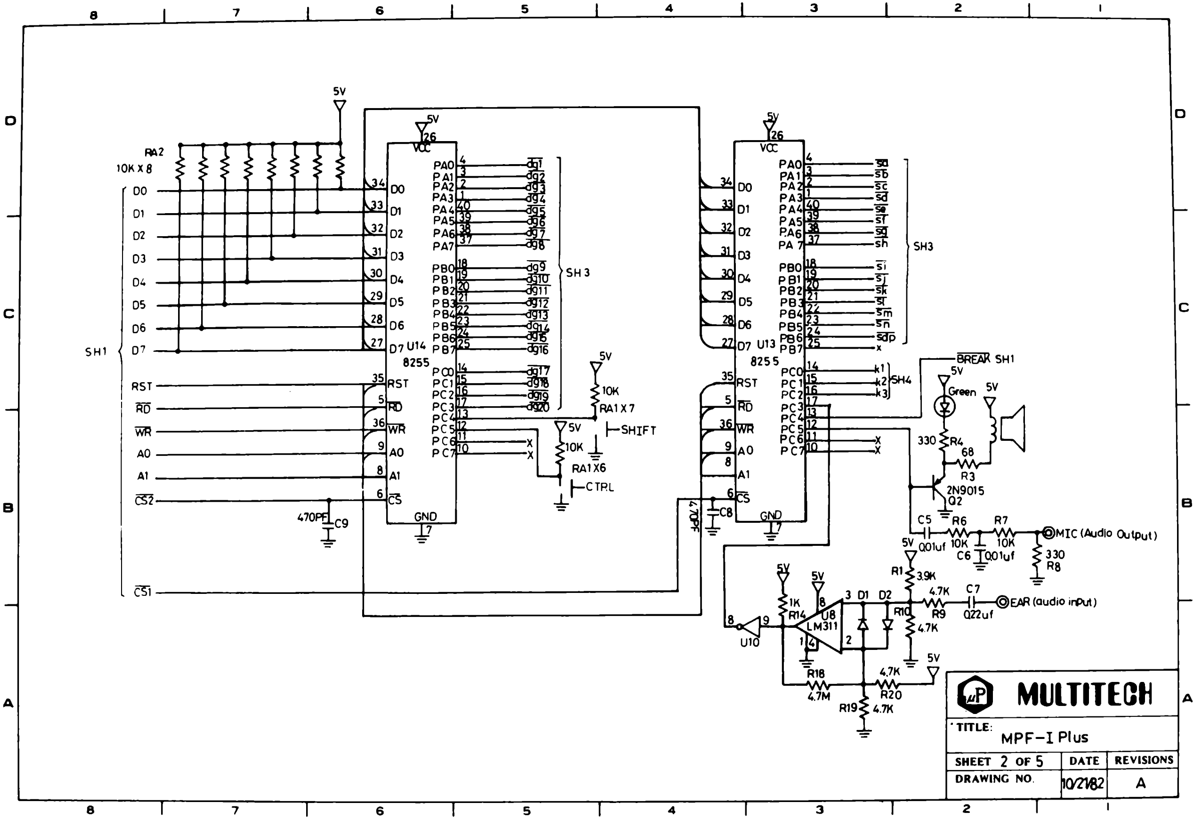


MULTITECH

TITLE: MPF-I Plus

SHEET 1 OF 5	DATE	REVISIONS
DRAWING NO.	10/21/82	A

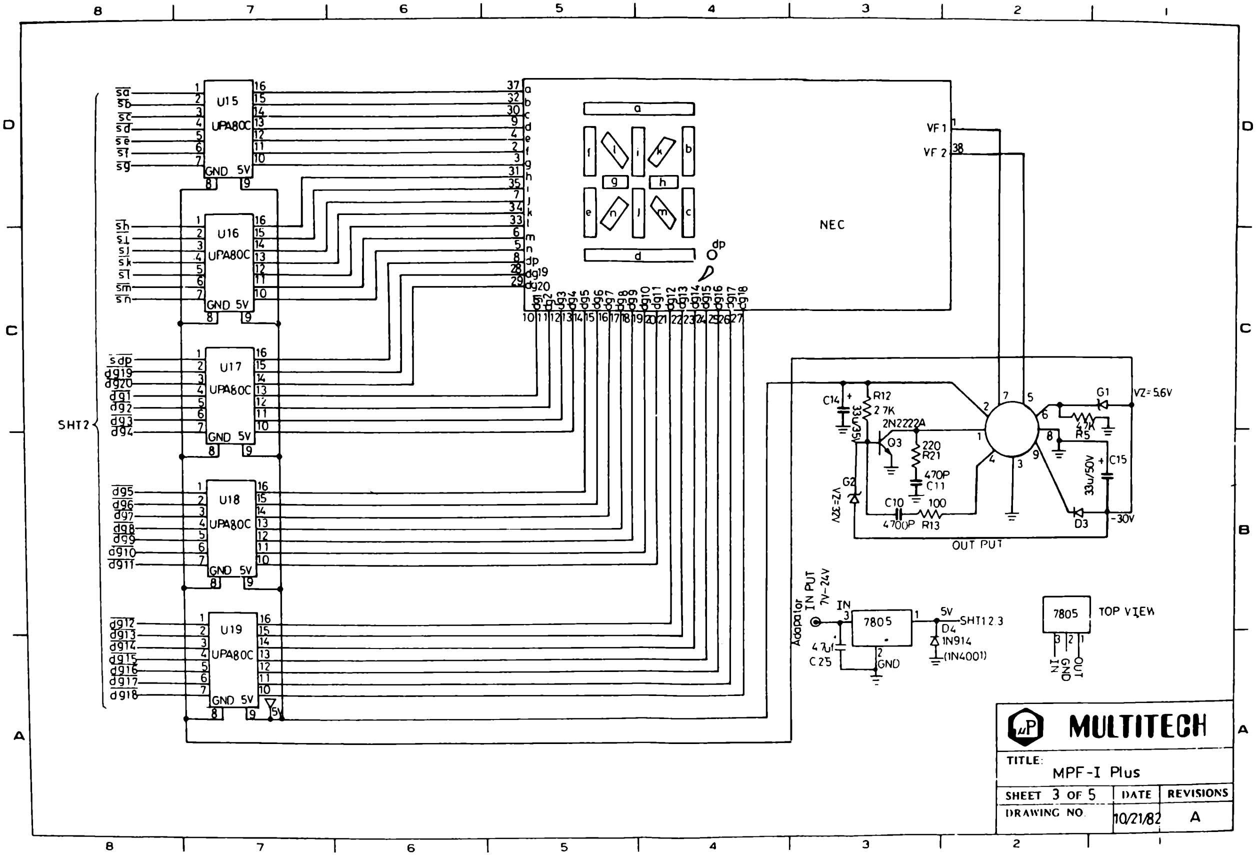
D-1



MULTITECH

TITLE: MPF-I Plus

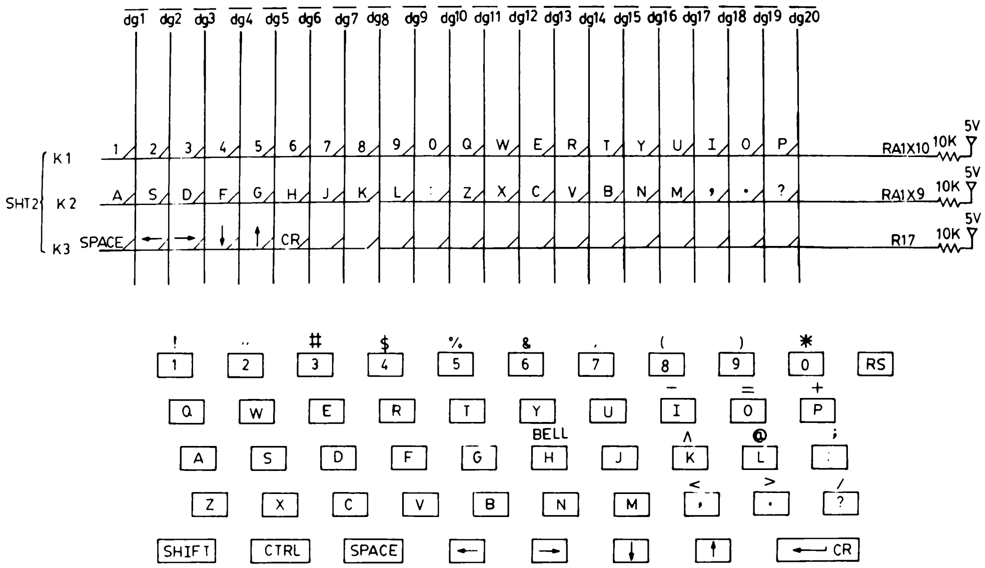
SHEET 2 OF 5	DATE	REVISIONS
DRAWING NO.	10/21/82	A



MULTITECH

TITLE: MPF-I Plus

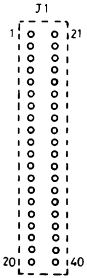
SHEET 3 OF 5	DATE	REVISIONS
DRAWING NO.	10/21/82	A



MULTITECH

TITLE: MPF-I Plus

SHEET 4 OF 5	DATE	REVISIONS
DRAWING NO	10/21/82	A

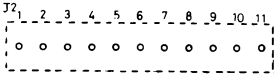


J1 PIN FUNCTION

PIN NO	SIGNAL	PIN NO	SIGNAL
1	A11	21	A10
2	A12	22	A9
3	A13	23	A8
4	A14	24	A7
5	A15	25	A6
6	Φ	26	A5
7	D4	27	A4
8	D3	28	A3
9	D5	29	A2
10	D6	30	A1
11	5V	31	A0
12	D2	32	GND
13	D7	33	$\overline{\mathrm{RFSH}}$
14	D0	34	$\overline{\mathrm{MI}}$
15	D1	35	$\overline{\mathrm{RESET}}$
16	$\overline{\mathrm{INT}}$	36	$\overline{\mathrm{BUSRQ}}$
17	$\overline{\mathrm{NMI}}$	37	$\overline{\mathrm{WAIT}}$
18	$\overline{\mathrm{HALT}}$	38	$\overline{\mathrm{BUSAK}}$
19	$\overline{\mathrm{MREQ}}$	39	$\overline{\mathrm{WR}}$
20	$\overline{\mathrm{IORQ}}$	40	$\overline{\mathrm{RD}}$

J2 PIN FUNCTION

PIN NO	SIGNAL
1	Q5-C
2	U12-4
3	U12-10
4	U4-20
5	U4-18
6	$\overline{\mathrm{WR}}$
7	A11
8	U4-21
9	U5-20
10	U5-18
11	U12-9



U4,U5 Default connection is destined for 6116

J2: PIN 1,4,9 Short

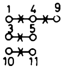
PIN 3,5 Short

PIN 6,8 Short

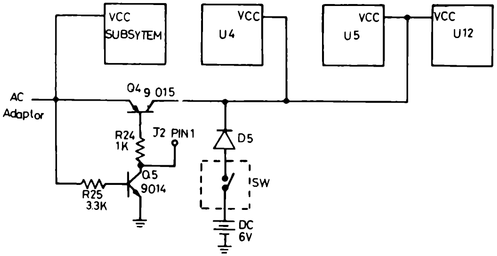
PIN10,11 Short

If user's want to change 6116 into 5516 connection for lower power battery back up

first. Cut J2 PIN 1,4,9
PIN 3,5
PIN 10,11



Second: Connect PIN 1,5,10
PIN 3,4
PIN 9,11



SW	Description
ON	Auto battery back up
OFF	No battery back up

MULTITECH

TITLE: MPF-I Plus			
SHEET 5 OF 5		DATE	REVISIONS
DRAWING NO.		10/21/82	A

Appendix E

MPF-IP Monitor Command Summary

Category	Command	Function
* Major Function Entry	RESET	Enter and initialize the monitor
	Q	Re-enter the monitor
	E	Enter and initialize the text editor
	R	Re-enter the text editor
	A	Enter two pass assembler
	L	Enter one pass assembler
	D	Enter disassembler
	B	Enter the BASIC language
	C	Re-enter BASIC
Fill in Data	F	Store data in the RAM buffer
Jump Relative	J	Calculate the relative address
Insert Data	I	Insert the contents of a memory block into the RAM
Delete Data	D	Dalete one byte of data from the memory
Execution	G	Execute a program which starts from a specified address
Step	S	Single-step a program (Execute a program instruction by instruction.)
Display/Alter Registers	R	Display the contents of registers
	↓	Display the contents of the next pairs of registers
	↑	Display the contents of the register pairs that precedes the registers currently displayed
	:	Change the contents of registers

Display/Alter Memory	M	Display the contents of specified memory locations
	↓	Display the contents of the next four bytes
	↑	Display the contents of the four bytes that precede the current displayed location
	:	Alter the contents of specified memory
	/	Move the contents of a memory block to another location
Manipulate Breakpoint	B	Set or clear breakpoint
Load/Dump Memory	L	Load data from tape to memory
	W	Write data from memory to tape

* Note: Any of the major functions are entered by typing the related control character while holding down the CONTROL key.

Appendix F

Editor Command Summary

Appendix F: Editor Command Summary

A. Editor Operation Sequence

I. Enter into the input mode of the text editor

1. CONTROL E
2. F: [nnnn] T: [nnnn]
3. INPUT (Flash for a few seconds.)
4. Type in a source program.
5. After typing in the soure program, type the carriage return key twice and the "Quit" command to exit to the monitor.

"n" represents a hexadecimal digit. The value enclosed in the square parentheses is optional. If a programmer does not want to set the starting and ending addresses for the text buffer, he may type the carriage return key when prompted by F: and T:. This will set two default values for the text buffer.

II. Enter into the edit mode of the text editor

1. CONTROL R
2. F: [nnnn] T: [nnnn]
3. Edit (Flash for a few seconds on the display.)
4. $ (Display the prompt of the text editor in edit mode. The line pointer is pointing to the top of the file in the text buffer.)
5. Use editor commands to revise the source program. After finishing editing the source code, type carriage return key twice and the "Q" command to exit to the monitor.

B. Summary of the Editor Commands

Category	Commands	Function
Editor Entry and Exit	Enter (CONTROL)	Enter the editor from monitor
	Re-enter (CONTROL)	Enter the editor from monitor
	Quit	Quit the editor and enter the monitor
Text Manipulating Commands	Delete	Delete a line
	Insert	Insert a line
	Print n	Print n lines
	Read/filename/	Read data from tape
	Write/filename/	Write data to tape
	Z	Print all the data in text buffer
Line Pointer Manipulating Commands	Bottom	Move the line pointer to the bottom of the file
	G n	Move the line pointer to the nth line in the text buffer
	Line number	Print the line number of the line pointed to by the line pointer
	Next n	Move the line pointer to the next n line
	Top	Move the line pointer to the top of the file
	Up n	Move the line pointer up n lines
String Handling Commands	Change/old string/ new string	Change a string in the current line
	Find/string/	Find the line with the specified string
Other Commands	Space	Print text buffer default values and the memory space used to store the current text file
	X	Control the prnter (a toggle switch)
	Carriage Return	Display the next line

Appendix G

Assembler Operation Sequence

Appendix G: Assembler Operation Sequence

I. Two-Pass Assembler Operation Sequence

1. CONTROL A
2. ORG:
3. ORG:[nnnn]
4. SYM>F:
5. SYM>F:[nnnn]
6. SYM>F:[nnnn] T:[nnnn]
7. OBJ>F:
8. OBJ>F:[nnnn]
9. OBJ>F:[nnnn] T:[nnnn]

"n" represents a hexadecimal digit. The value enclosed in the square parentheses is optional. If a programmer does not want to set the starting and ending addresses for the text buffer, he may type the carriage return key when prompted by F: and T:. This will set default values for the memory space for storing source code, symbol table, and object code.

II. One-Pass Assembler Operation Sequence

1. CONTROL L
2. ORG:
3. ORG:[nnnn]
4. OBJ>F:
5. OBJ>[nnnn]
6. INPUT
7. The display of the MPF-IP will show the value of the reference counter. The user may begin typing in a source program.

Appendix H
MPF-IP ASCII Code

MPF-IP ASCII CODE (CALL SCAN)

MSD	0	1	2	3	4	5	6	7
LSD	000	001	010	011	100	101	110	111
0 0000			space	0	@	P		
1 0001			!	1	A	Q		
2 0010			"	2	B	R		
3 0011			#	3	C	S		
4 0100			$	4	D	T		
5 0101			%	5	E	U		
6 0110			&	6	F	V		
7 0111			1	7	G	W		
8 1000			(	8	H	X	→	
9 1001			)	9	I	Y	↓	
A 1010			*	:	J	Z		
B 1011			+	;	K			
C 1100			,	<	L			
D 1101	CR		–	=	M			
E 1110			.	>	N	↑		
F 1111			/	?	O	←		

Appendix I

MPF-IP Keyboard Position Code

Position-code (CALL SCAN1):

00	'1' '!'	01	'A'	02	'space'
03	'2' '"'	04	'S'	05	' ← '
06	'3' '#'	07	'D'	08	' → '
09	'4' '$'	0A	'F'	0B	' ↓ '
0C	'5' '%'	0D	'G'	0E	' ↑ '
0F	'6' '&'	10	'H'	11	' CR '
12	'7' '''	13	'J'	14	
15	'8' '('	16	'K' '∧'	17	
18	'9' ')'	19	'L' '@'	1A	
1B	'0' '*'	1C	':' ';'	1D	
1E	'Q'	1F	'Z'	20	
21	'W'	22	'X'	23	
24	'E'	25	'C'	26	
27	'R'	28	'V'	29	
2A	'T'	2B	'B'	2C	
2D	'Y'	2E	'N'	2F	
30	'U'	31	'M'	32	
33	'I' '-'	34	',' '<'	35	
36	'O' '='	37	'.' '>'	38	
39	'P' '+'	3A	'?' '/'	3B	

Appendix J

The Display Patterns for Alphanumeric Letters and Special Symbols

Character	Segment name	dpnmlkji	hgfedcba	2nd byte	1st byte
A	a,b,c,e,f,g,h	11111111	00001000	FF	08
B	a,b,c,d,k,i,j,	11111100	01110000	FC	70
C	a,d,e,f	11111111	11000110	FF	C6
D	a,b,c,d,i,j	11111100	11110000	FC	F0
E	a,d,e,f,g,h	11111111	00000110	FF	06
F	a,e,f,g,h	11111111	00001110	FF	0E
G	a,c,d,e,f,h	11111111	01000010	FF	42
H	b,c,e,f,g,h	11111111	00001001	FF	09
I	a,d,i,j	11111100	11110110	FC	F6
J	b,c,d,e	11111111	11100001	FF	E1
K	e,f,g,k,m	11101011	10001111	EB	8F
L	d,e,f	11111111	11000111	FF	C7
:	h,m	11101111	01111111	EF	7F
&	a,b,d,e,g,h,e,m	11100111	00100100	E7	24
M	b,c,e,f,k,l	11110011	11001001	F3	C9

N	b,c,e,f,l,m	11100111	11001001	E7	C9
O	a,b,c,d,e,f	11111111	11000000	FF	C0
P	a,b,e,f,g,h	11111111	00001100	FF	0C
Q	a,b,c,d,e,f,m	11101111	11000000	EF	C0
R	a,b,e,f,g,h,m	11101111	00001100	EF	0C
S	a,c,d,f,g,h	11111111	00010010	FF	12
T	a,i,j	11111100	11111110	FC	FE
U	b,c,d,e,f	11111111	11000001	FF	C1
V	e,f,k,n	11011011	11001111	DB	CF
W	b,c,e,f,m,n	11001111	11001001	CF	C9
X	k,l,m,n	11000011	11111111	C3	FF
Y	j,k,l	11110001	11111111	F1	FF
∧	m,n	11001111	11111111	CF	FF
'	n	11011111	11111111	DF	FF
Z	a,d,k,n	11011011	11110110	DB	F6
.	dp	10111111	11111111	BF	FF

1	i,j	11111100	11111111	FC	FF
2	a,b,d,e,g,h	11111111	00100100	FF	24
3	a,b,c,d,g,h	11111111	00110000	FF	30
4	f,g,h,i,j	11111100	00011111	FC	1F
5	a,c,d,f,g,h	11110111	01110010	F7	72
6	a,c,d,e,f,g,h	11111111	00000010	FF	02
7	a,b,c,f	11111111	11011000	FF	D8
8	a,b,c,d,e,f,g,h	11111111	00000000	FF	00
9	a,b,c,d,f,g,h	11111111	00010000	FF	10
0	a,b,c,d,e,f,k,n	11011011	11000000	DB	C0
+	g,h,i,j	11111100	00111111	FC	3F
#	b,c,d,g,h,i,j	11111100	00110001	FC	31
@	a,b,c,d,e,g,j	11111101	10100000	FD	A0
—	g,h	11111111	00111111	FF	3F
(	k,m	11101011	11111111	EB	FF
)	e,n	11010111	11111111	D7	FF
/	k,n	11011011	11111111	DB	FF

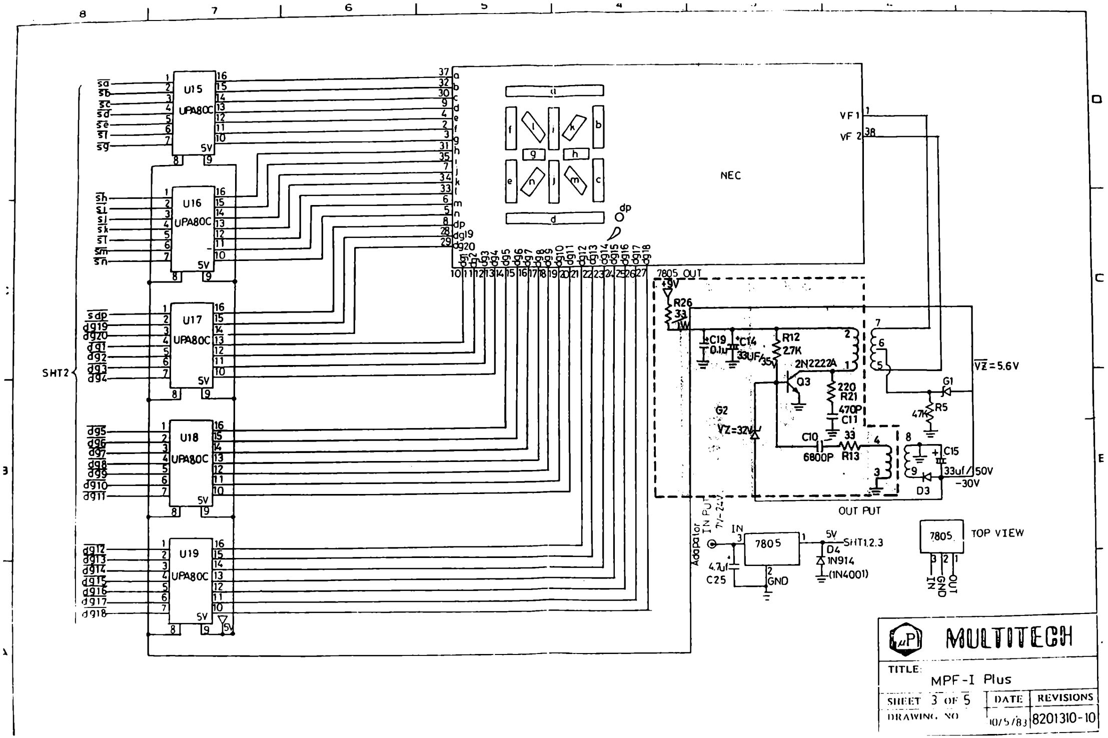
*	g,h,i,j,k,l,m,n	10000000	00111111	80	3F
″	f,l	11110111	11011111	F7	DF
′	k	11111011	11111111	FB	FF
=	d,g,h	11111111	00110111	FF	37
?	a,b,h,j	11111101	01111110	FD	7C
%	c,f,g,h,k,l,m,n	11000011	00011011	C3	1B
<	d,k,n	11011011	11110111	DB	F7
>	d,l,m	11100111	11110111	E7	F7
$	a,c,d,f,g,h,i,j	11111100	00010010	FC	12
!	a,j,k,l	11110001	11111110	F1	FE

Appendix K

Memory Mapping & I/O Ports

MPF-IP

ITEM	Memory		I/O
	RAM	ROM	
MAIN BOARD	(1) F000-FFFF	(1) 0000-1FFF (2) 2000-3FFF (BASIC) (3) 4000-4FFF (Option)	(1) 80-8F 8255 U14 (2) 90-9F 8255 U13
PRT	NONE	(1) 6000-6FFF (2) 7000-7FFF (Option)	(1) CA, CB
EPB	(1) D800-EFFF	(1) 9000-9FFF	(1) 78-7F
SSB	NONE	(1) 5000-5FFF (2) 6000-7FFF (Option)	(1) FE
SGB	NONE	(1) C000-CFFF (2) D000-DFFF (Option)	(1) C0-C3
TVB	(1) A800-AFFF	(1) A000-A7FF	(1) 40-4F
IOM	(1) D800-EFFF (2) C000-D7FF (Option)	(1) B000-BFFF	(1) 60-6F
STB		(1) 7000-7FFF	

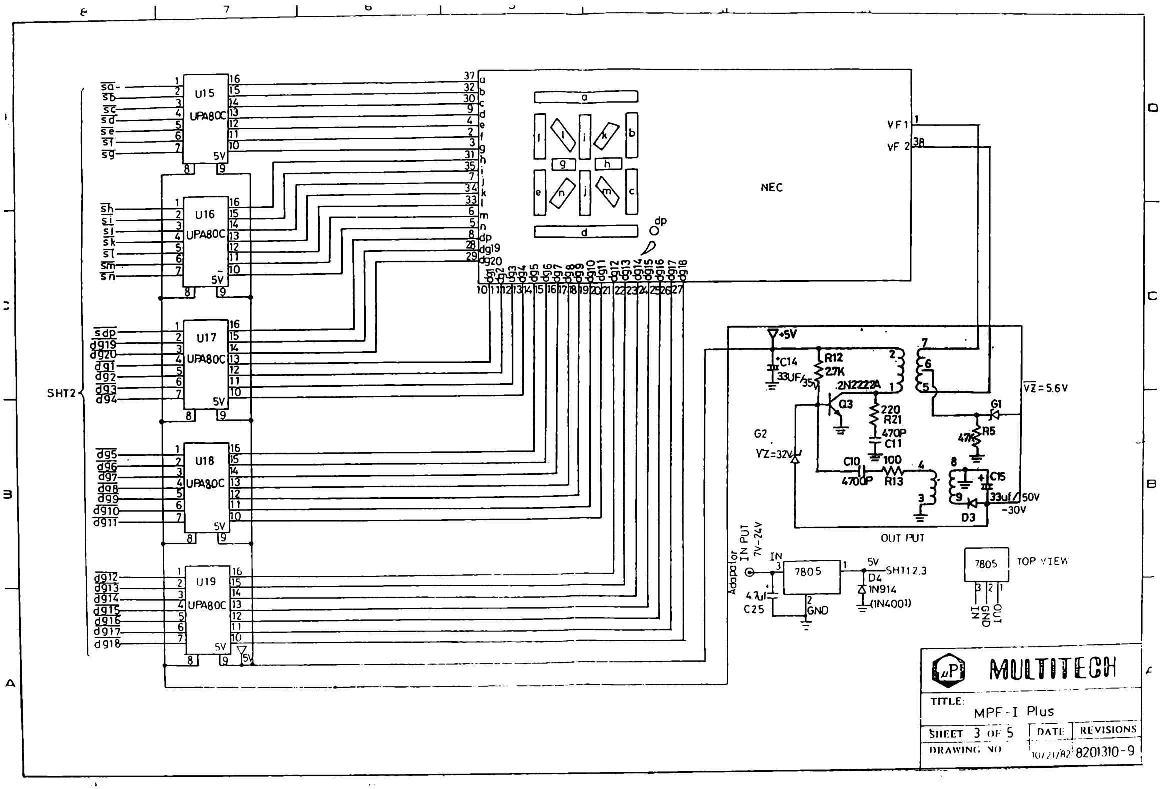


MULTITECH

TITLE: MPF-I Plus

SHEET 3 OF 5 | DATE 10/5/83 | REVISIONS

DRAWING NO 8201310-10



MULTITECH

TITLE: MPF-I Plus

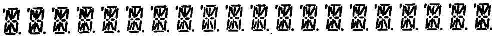
SHEET 3 OF 5 | DATE 10/21/82 | REVISIONS

DRAWING NO 8201310-9

Appendix L

Fluorescent Indlcator Panel Specifications

CHARACTER FORMAT


(actual size)

FEATURES Compatible with MOS LSI. Possible to indicate Numeric or Alphabetic character.

APPLICATIONS Electronic computer and other digital display.

OPERATION MODE The FIP20B6R is designed for a multiplexed drive mode.

MECHANICAL DATA

External Dimensions, Terminal, Shape and Size of Digit . . . . . See attached drawing
Operating Temperature Range . . . . −10 to +60 °C
Storage Temperature Range . . . . −40 to +70 °C
Weight . . . . 29 g approx.
Mounting Position . . . . Any

OPTICAL DATA

Color . . . . Green
Brightness . . . . 860 cd/m^2 (TYP.)
(250 ft.L)

ELECTRICAL DATA

		E_f	e_b	e_c	Du*	t_p**	t_b	f	E_{cco}	E_{bco}	L	
	UNIT	Vac	V_{p-p}	V_{p-p}	–	µs	µs	Hz	Vdc	Vdc	cd/m^2	(ft.L)
Maximum Ratings	MAX.	6.16	36	36	1/20	–	–	–	–	–	–	
	MIN.	5.04	–	–	–	40	10	200	–	–	–	
Typical Operation	TYP.	5.6	30	30	1/24	100	20	417	−6	−6	860	(250)

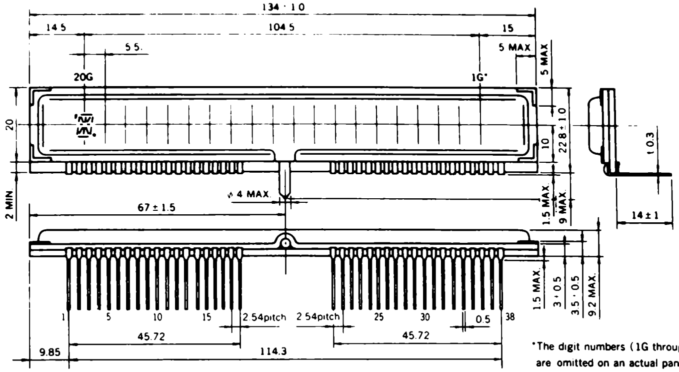
* effective value
** without branking duration

ELECTRICAL CHARACTERISTICS (Test circuit is specified by FEB−1001)

ITEM	SYMBOL	CONDITION	TOLERANCE			UNIT
			MIN.	NOM.	MAX.	
Filament Current	I_f	E_f = 5.6 Vac, e_b = e_c = 0	33	37	41	mAac
Anode Current	i_b/digit	E_f = 5.6 Vac, e_b = 30 V_{p-p}, e_c = 30 V_{p-p}, Du = 1/24, t_p = 100 µs, All segments are lit.	–	3.5	7.0	mA_{p-p}
Grid Current	i_c/digit		–	3.5	7.0	mA_{p-p}
Brightness	L		340 (100)	860 (250)	–	cd/m^2 (ft.L)
Brightness Ratio Between Digit	–		50	–	–	%
Anode Cut-off Voltage	E_{bco}	E_f = 5.6 Vac, e_c = 30 V_{p-p}, Du = 1/24, t_p = 100 µs, All segments are lit.	−4	–	–	Vdc
Grid Cut-off Voltage	E_{cco}	E_f = 5.6 Vac, E_b = 30 Vdc, All segments are lit.	−6	–	–	Vdc

Note : These values are specified when e_b and e_c are supplied from the center tap of the filament transformer, and also specified at 25 °C.

Fig. 1 OUTLINE DRAWING (Unit mm)

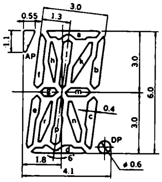


*The digit numbers (1G through 20G) are omitted on an actual panel.

TERMINAL CONNECTION

Terminal No.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Electrode	F	P(f)	P(g)	P(e)	P(r)	P(n)	P(p)	P(DP)	P(d)	20G	19G	18G	17G	16G	15G	14G	13G	12G	11G
Terminal No.	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38
Electrode	10G	9G	8G	7G	6G	5G	4G	3G	2G	1G	P(c)	P(m)	P(b)	P(h)	P(k)	P(j)	P(AP)	P(a)	F

Fig. 2 SEGMENT PATTERN (Unit : mm)



NOTE FOR USAGE

1. The panel display "FIP" is composed mainly of high quality glass, and so careful handling should be taken.
2. The connecting section between the panel and the lead wires must be free from extreme tensile and bending stresses.
3. When mounting the panel display and equipment, the surface should be utilized except for the exhaust tube and connecting section.
4. The panel display "FIP" should be utilized with green color filter to obtain a good display appearance, and also filterable to red, orange and yellow.
5. Subject to change without any notice.

Appendix M

μPA80c Specificatications

TRANSISTOR ARRAY
μPA80C

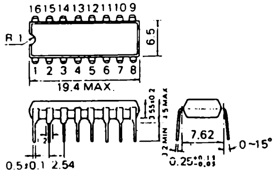
FLUORESCENT INDICATOR PANEL DRIVER
PNP-NPN SILICON EPITAXIAL TRANSISTOR ARRAY

DESCRIPTION

The μPA80C is a monolithic array of seven PNP-NPN structured transistors. This device is especially suited for driving FIP (Fluorescent Indicator Panel).

PACKAGE DIMENSIONS

in millimeters



FEATURES

- High voltage rating V_{SS}: −60 V
- Pull down resistors incorporated
- Base current limiting resistors incorporated
- Package is 16 pin plastic DIP (Dual In-Line Package).

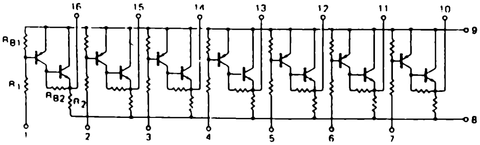
ABSOLUTE MAXIMUM RATINGS

Maximum Voltages and Currents (Ta=25 °C)			
Supply Voltage	V_{SS}	−60	V
Input Voltage	V_I	−20	V
Output Current	I_O	50	mA/unit
Maximum Power Dissipation			
Total Power Dissipation	P_d	550	mW
Maximum Temperature			
Storage Temperature	T_{stg}	−40 to +125	°C
Operating Temperature	T_{opt}	−25 to + 75	°C

ELECTRICAL CHARACTERISTICS (Ta=25 °C)

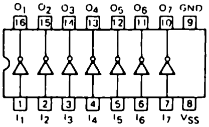
CHARACTERISTIC	SYMBOL	MIN.	TYP.	MAX.	UNIT	TEST CONDITIONS
Output Leakage Current	I_L			1.0	μA	V_{CE}=50 V
DC Current Gain	h_{FE1}	100	280			V_{CE}=2.0 V, I_O=20 mA
	h_{FE2}	250	450			V_{CE}=2.0 V, I_O=40 mA
Collector Saturation Voltage	$V_{CE(sat)}$		0.95	1.5	V	I_O=20 mA, I_I=0.3 mA
Input Current	I_I			1.0	mA	V_I=−5.0 V

EQUIVALENT CIRCUIT



R_{B1} 20 kΩ
R_1 20 kΩ
R_{B2} 2 kΩ
R_2 100 kΩ

CONNECTION DIAGRAM (Top View)

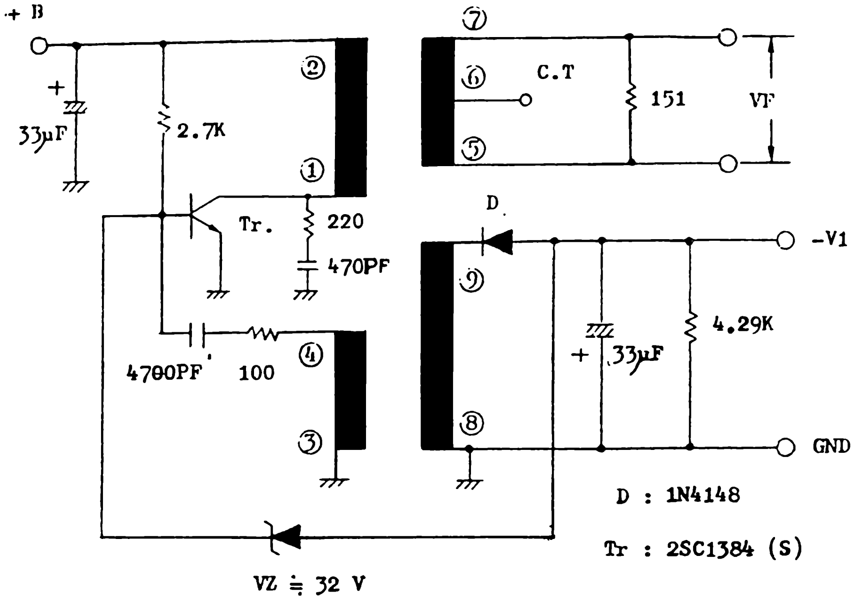

O_1 O_2 O_3 O_4 O_5 O_6 O_7 GND
16 15 14 13 12 11 10 9
1 2 3 4 5 6 7 8
I_1 I_2 I_3 I_4 I_5 I_6 I_7 V_{SS}

I=Input (Base)
O=Output

Appendix N

DC DC Converter

Test Circuit:



D : 1N4148

Tr : 2SC1384 (S)

Electrical Characteristic:

+ B (V)	Iin (mA)	Fosc (KHZ)	VF (Vrms)	-V1 (-V)	n (%)
6	1Ø5	148.3	6.3	31.48	78.2
5	113	136.3	5.7	31.24	78.Ø
4	124	122.2	5.Ø	3Ø.86	77.5

Appendix O

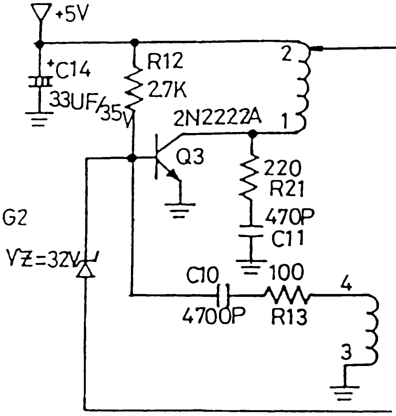
MPF-IP Model 1 & Model 2

MPF-IP Main Board

1) Model 1: PB82Ø131Ø-9

(a) DC DC Converter Oscillation Circuit has no current limit resistor, so it may fail under abnormal operation.

(b) Circuit

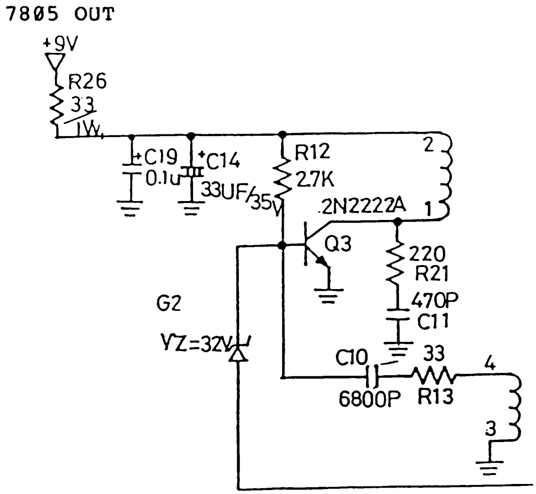


R = Ø.2

Xtor 2N2222A will fail if the oscillation circuit is not active.

(2) Model 2: PB8201310-10

(a) DC DC Converter Oscillation Circuit has current limit resistor R26. And we have enhanced the Oscillation Circuit by changing C10 4700p to 6800p, and R13 100 to 33. Therefore, the fail rate of 2N2222A is decreased to 0.1% under mass production.



Current Limit Resister

(3) Xtor 2N2222A

Vendor Approved: (1) Sieg (Siemens)
(2) KEC (Korea)
(3) TS 2N2222A (Tiff)

Multitech
INDUSTRIAL CORP.

OFFICE /
9FL 266 SUNG CHIANG ROAD TAIPEI 104
TAIWAN, R O C
TEL (02)551 1101
TELEX 19162 MULTIIC FAX (02)542 2805
FACTORY/
1 INDUSTRYE ROAD, III,
HSINCHU SCIENCE BASED INDUSTRIAL PARK
HSINCHU TAIWAN 300, R O C