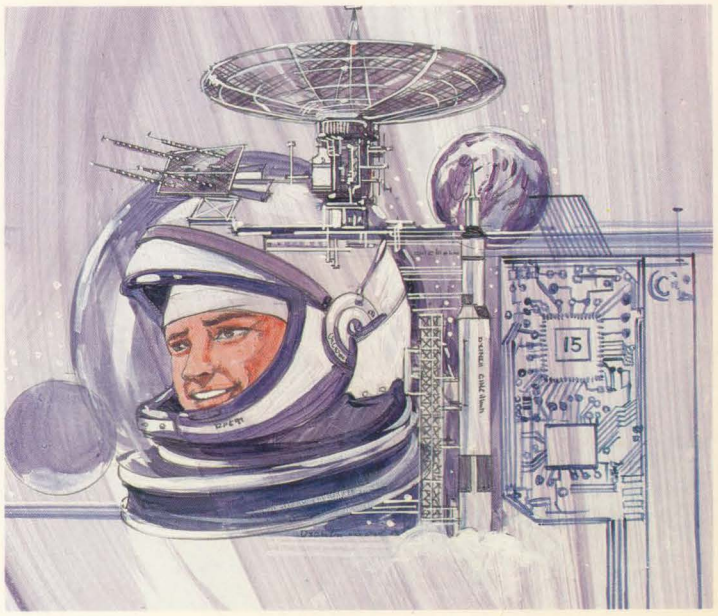
Personal Computer

MZ-80B

OWNER'S MANUAL



SHARP

The MZ-80B Personal Computer supplied in the U.K. and Republic of Ireland have 64k of RAM and optional graphic RAM-1 (MZ-80GM) fitted as standard.

Page 7 of the Owner's Manual shows various peripherals which can be connected to the MZ-80B. The Mark Card Reader, Hard Disk, Colour Intelligent Terminal and Colour Display are planned for future development.

SHARP

Personal Computer

MZ-80B

Owner's Manual

January 1981

080211-150281

Printed in Japan

IMPORTANT

For users in the United Kingdom:

The wires in the power cable of this device are colored in accordance with the following code:

BLUE : Neutral

BROWN : Live

As the colors of the wires in the power cable of this device may not correspond with the colored markings identifying the terminals in your plug, proceed as follows:

- The blue colored wire must be connected to the terminal which is marked with the letter N or colored black.
- The brown colored wire must be connected to the terminal which is marked with the letter L or colored red.

Preface

This manual describes the Sharp MZ-80B personal computer. Read this manual thoroughly to become familiar with the operating procedures and precautions before operating your MZ-80B. This manual is one of a series of publications describing the MZ-80B and associated software.

- Owner's Manual . . . This publication
- BASIC Language Manual
- MONITOR SB-1510 Reference Manual

Chapters 1 and 2 describe the features of the MZ-80B and general operating procedures; read these chapters first. Chapter 3 and 4 describe the hardware. This information will be helpful to you if you intend to expand system.

All software is supplied in the form of files. A cassette tape which contains the SB-5510 BASIC interpreter and MONITOR SB-1510 (which support the standard BASIC programming language) is included with the MZ-80B.

Refer to the BASIC Language Manual for details on the BASIC language.

For details on MONITOR SB-1510, refer to the MONITOR SB-1510 Reference Manual.

Keep the warranty card and list of service centers as well as this manual and the other two manuals.

Precautions

The MZ-80B is one of the finest personal computers in the world; its design incorporates all the technical knowledge accumulated by Sharp in its many years of experience in the electronics field. All units are thoroughly inspected prior to shipment so that each will operate normally when it is unpacked. However, be sure to check visually for any damage caused during transportation. If any damage is found or any parts are missing, contact your dealer immediately.

Observe the following guidelines to keep your set in optimum operating condition:

- Do not place the MZ-80B in locations where the temperature is extremely high or low or where it varies to a great extent. Avoid exposing the unit to direct sunlight, vibration or dust.
- Handle the power cable carefully to prevent it from being damaged. When removing it from the AC outlet, turn the power off first, then pull the plug (do not pull on the cable).
- If the power switch is turned off then immediately turned on again, initialization may not be performed correctly. Allow a few moments after turning the power off before turning it on.

For more detailed information, see Appendix 4.

Contents

Chapter 1

The World of the MZ-80B Personal Computer

What can computers do? You will see that computers are used for many different purposes in many places. Computers carry out complicated scientific calculations, various business procedures, simulations and statistical processing with the aid of high level languages such as BASIC, PASCAL, FORTRAN and COBOL. Computers operate measuring systems and automatic control systems in a variety of plants and networks. In laboratories engaged in software development, the computer is even used to study itself.

What can your MZ-80B do? There is no definite answer to this question, since the MZ-80B can be used in such a wide range of applications. You may apply it to any purpose you wish.

Chapter 1 of this manual describes the features of the MZ-80B, hardware expansion and the scope of the software.

1.1 Features

The MZ-80B is a compact personal computer with superb operability which features a variety of software and freely expandable hardware.

The CPU (Central Processing Unit) and the main memory form the nucleus of the computer. The MZ-80B uses the Z80A microprocessor (equivalent to the LH0080A produced by Sharp), one of the best microprocessors currently available for central processing units. The main memory which can be directly accessed by the CPU is constituted entirely of random access memory. It is expandable to 64K bytes. Consequently, no fixed programs or data reside in the main memory and any type of system software can be loaded into it from an external file. This makes it possible to make the best possible use of the main memory area.

The I/O devices, timer, initial program loader, etc., support the CPU and main memory. The initial program loader is automatically started when the power switch of the MZ-80B is turned on. It loads programs from a cassette tape or diskette file, then transfers control to the program loaded.

A typewriter keyboard, numeric pad, special function keys, cursor control keys and cassette tape deck control keys are included on the control panel. A variety of control commands and data can be entered with these keys.

Both character display and graphic display are possible, allowing various forms of data representation.

FIGURE 1.1 Personal Computer MZ-80B

1.1.1 Memory configuration

Random access memory (RAM) is the type of memory which is most naturally suited to computers. When this type of memory is employed, the user can select the programming language and the program to be executed at will. The MZ-80B employs this method to allow you to select the programming language which best suits your purpose. Further, if you want the computer to execute a machine language program, you can code and execute it.

In the MZ-80B, the IPL (Initial Program Loader) automatically loads programs which are stored on cassette tape or (if a disk drive is connected) diskette into the main memory when the power is turned on, then transfers control to the program loaded. Initial program loading from cassette tape is completed in a few minutes; loading from a diskette is accomplished in seconds.

The IPL is stored in ROM (Read Only Memory). This ROM address space is different from that of the main memory, and it is automatically activated when the power is turned on. See FIGURE 1.2.

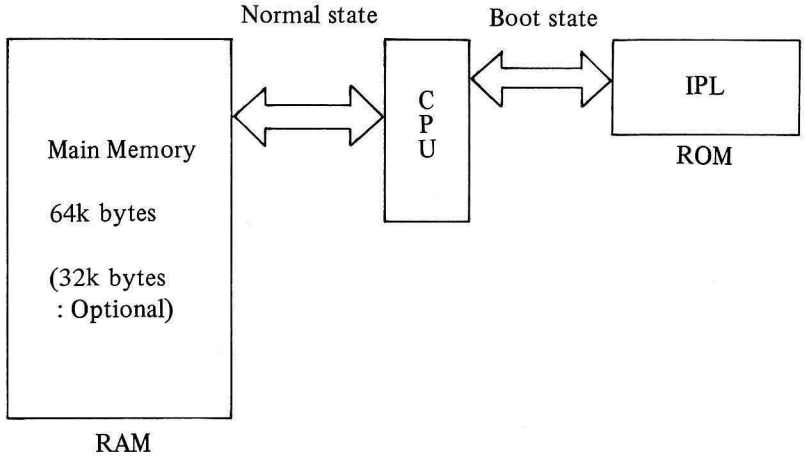


FIGURE 1.2

1.1.2 Superb operability

The MZ-80B becomes a BASIC language computer after the SB-5510 BASIC interpreter has been loaded and activated by the IPL. You can now perform a wide variety of operations with the MZ-80B, such as data input and output, text file generation, debugging and file access.

The MZ-80B's superb operability and expandability will help you to perform such operations with ease.

- Keys on the console are divided into groups according to their functions. The main typewriter keyboard and the numeric pad are located at the front of the console. The special function keys, cursor control keys and cassette tape deck control keys are located under the CRT display screen and cassette tape deck.
  All ordinary operations other than power on/off can be performed with these keys.

- Alphabetic characters, numerics and symbols are all input from the typewriter keyboard. The [RVS] key allows input of reverse characters and the [GRPH] key enables input of graphic patterns from the keyboard.
  Small letters are normally input from the console of the MZ-80B by pressing the [SHIFT] key. A command is provided, however, which makes it possible to reverse the shift function so that capital letters are input when the [SHIFT] key is pressed. Tabulation settings can also be made by the program.
  These functions improve the efficiency of message coding and table and graph editing. The cursor control keys allow these tasks to be performed even more efficiently.

- A separate numeric pad including [0] ~ [9], [00], [+], [−] and [·] keys is also provided. This is convenient when input of large amounts of numeric data is required. The numeric keys are scanned by a different scan signal than that which scans the numeric keys on the typewriter keyboard. This makes various applications possible. For example, keys on the numeric pad can be easily operated with the right hand as real time operation interruption keys.

- Functions of the 10 special function keys are all user definable. Therefore, by defining a special function key as a frequently used command, the command can be executed just by pressing the key once.

The MZ-80B uses the high speed Z80A-CPU which allows instructions to be executed in half the time required by the Z80-CPU.

- The cassette tape deck is controlled by software. All cassette tape operations, i.e., storing, loading and verifying data and rewinding, fast-forwarding and stopping the tape, are performed by the program.

  The APSS (Automatic Program Search System) fast-forwards the tape until the specified file is found.

  Automatic functions allow the cassette tape deck to be operated much more efficiently than has been possible in the past.

  Manual operation keys, [REW], [FF], [STOP] and [EJECT], are provided on the console.

- The MZ-80B has a superior display system with the following features; it displays all characters and patterns input from the keyboard in any mode; it operates in either the 40 or 80 characters/line mode; the scrolling area can be restricted to a part of the screen; and black and white can be reversed.

  Further, 2 optional graphic memories which enable graphic display of 320 x 200 dots per frame can be added to the MZ-80B. With this high resolution, the range of possible applications for the MZ-80B becomes very wide indeed.

1.2 Expansion equipments

A variety of peripheral devices is available for expanding the MZ-80B personal computer system. FIGURE 1.3 shows a typical expanded system configuration. With the floppy disk drive, numerous data and program files can be stored and accessed at high speed. With the printer, hard copies of listings and printed graphic patterns can be obtained. This improved processing efficiency, resulting in a wider range of applications.

The MZ-80B dual floppy disk drive uses a double density mini-floppy diskette (286K bytes/diskette) with a diameter of 5.25 inches, both sides of which are used for recording. It enables use of the DISK BASIC interpreter, which is suitable for practical business applications of the double precision DISK BASIC interpreter, which performs 16 digit BCD operations. Thus, the expanded system exhibits an ability which is comparable with that of larger computers with the aid of a variety of the floppy disk operating system software.

The compact MZ-80P5 line printer enables not only program listing, but also graphic pattern printing in the image mode.

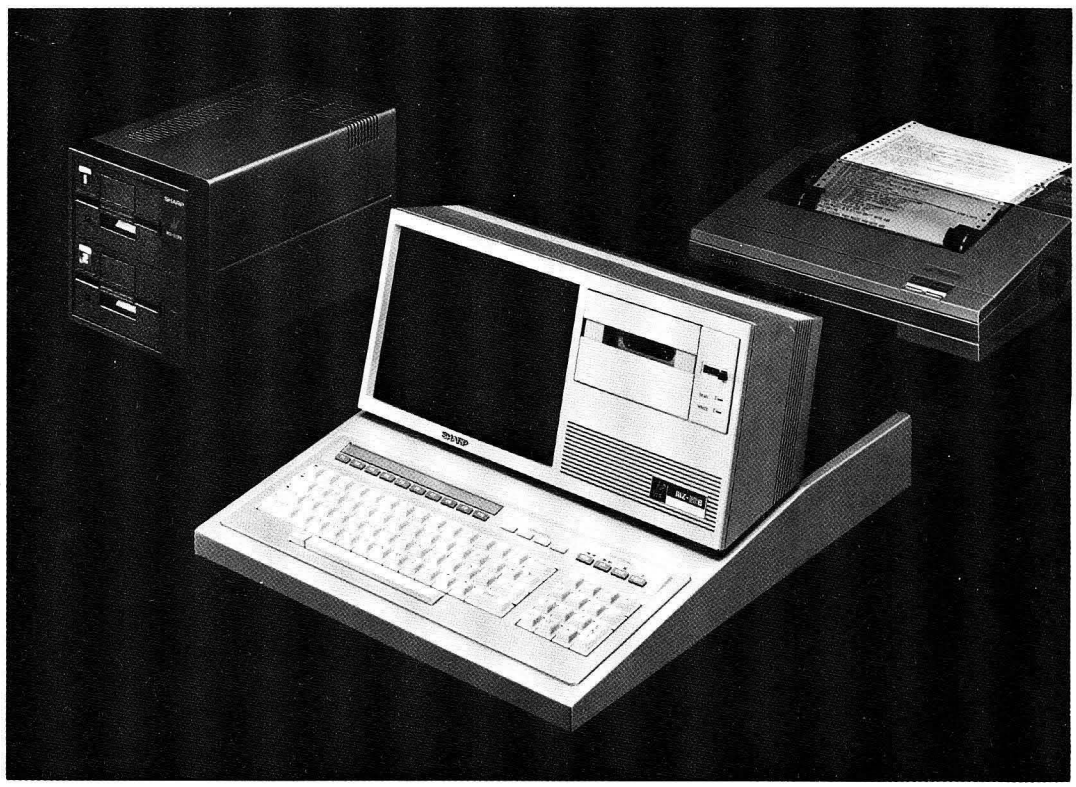


FIGURE 1.3 Typical expansion system

FIGURE 1.4 shows peripheral devices which can be connected to the MZ-80B. Devices which are enclosed in a thick solid line are connected to the expansion I/O port via interface cards or connected to the specified connectors in the main cabinet.

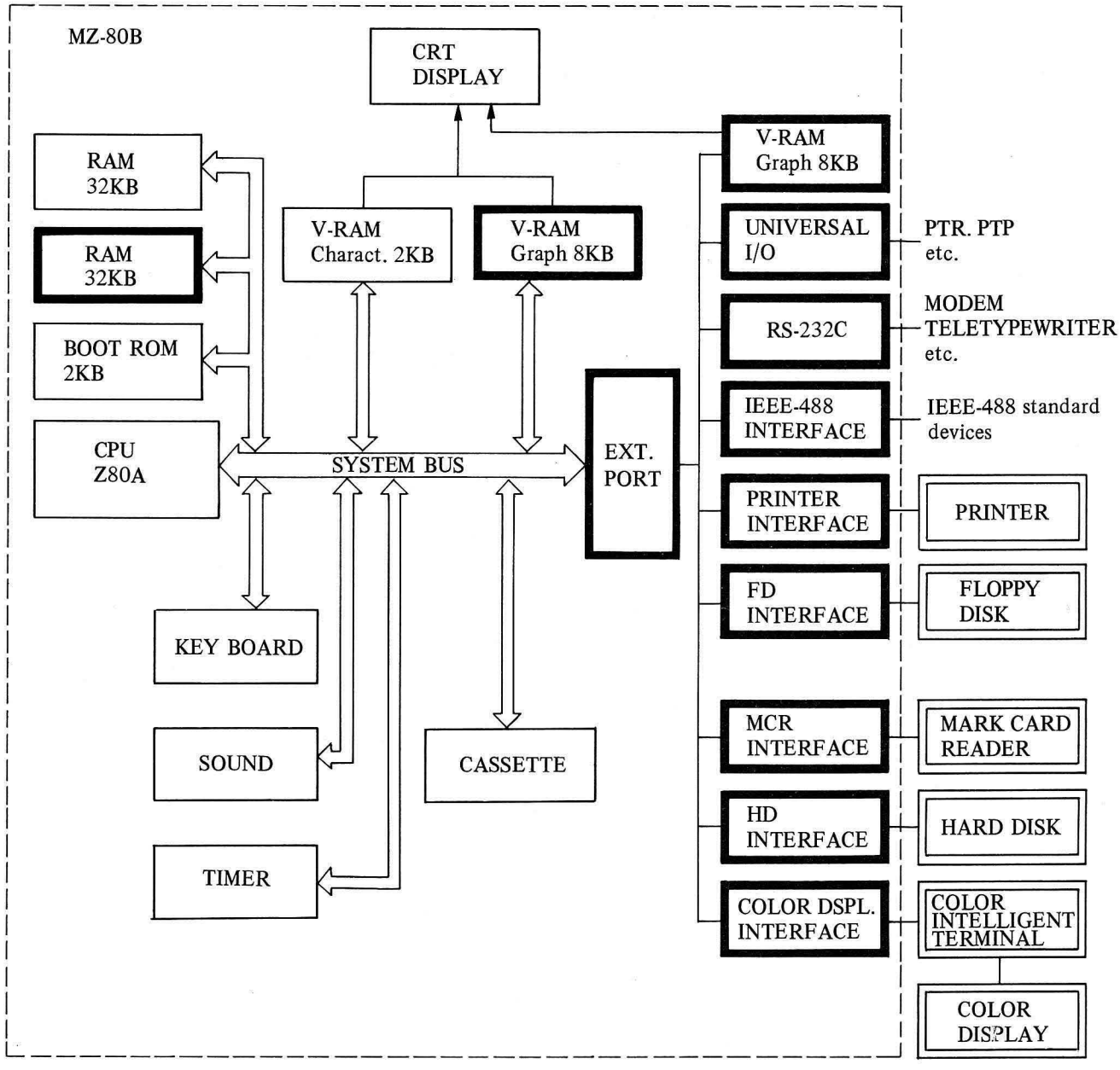


FIGURE 1.4 MZ-80B system extension

Chapter 2
Using the MZ-80B

This chapter describes the constituent units of the MZ-80B and their functions.

- *Locations of constituent units*
- *Use and function of the Initial Program Loader*
- *Functions of keys on the keyboard*
- *Outline of display control systems*

■ Top view of the MZ-80B

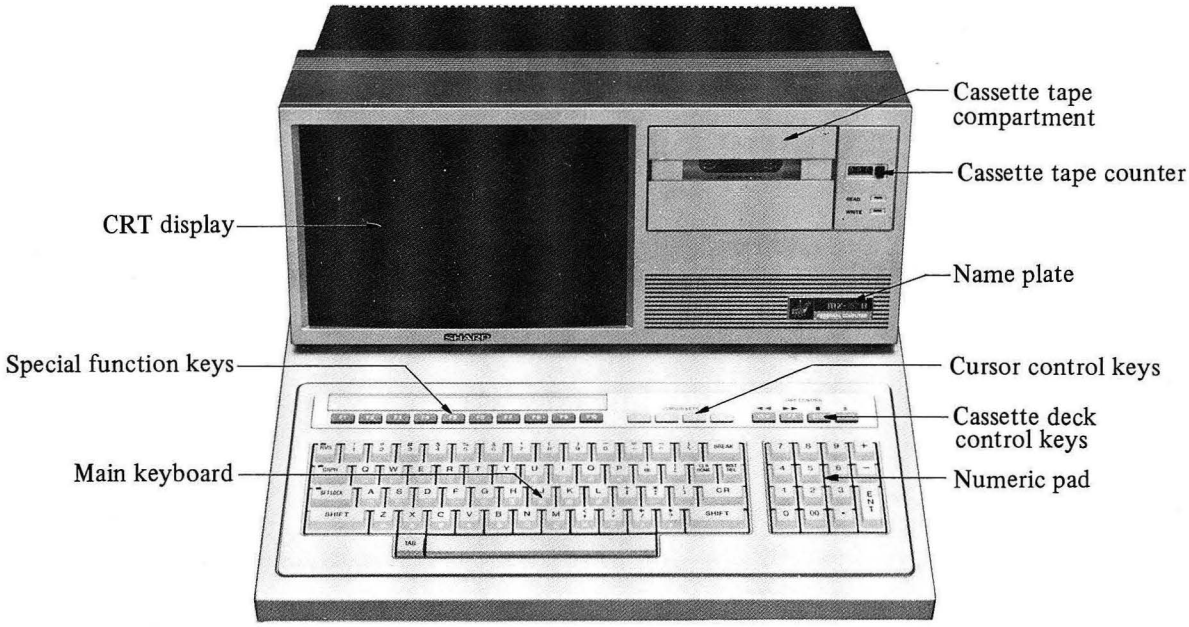


FIGURE 2.1

■ Rear view of the MZ-80B

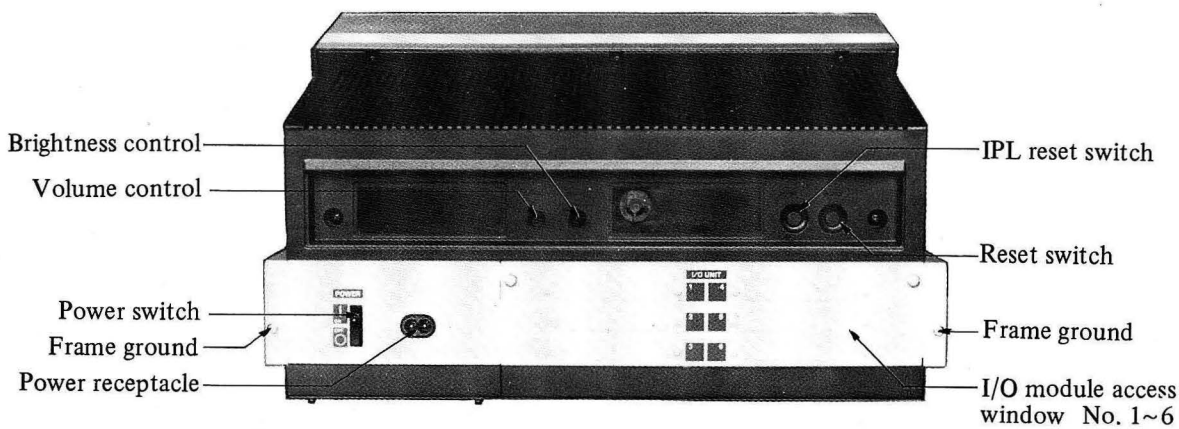


FIGURE 2.2

2.1 Initial program loading

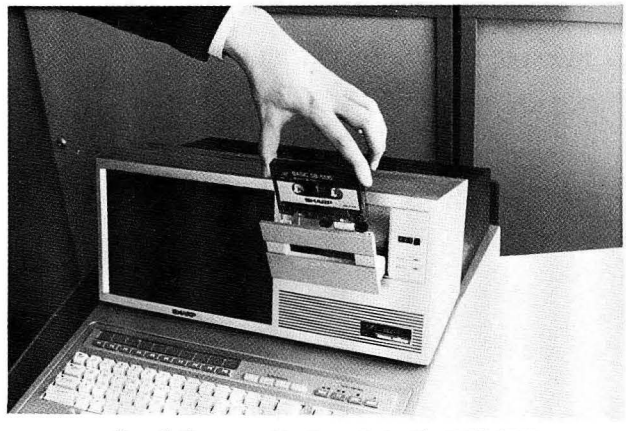
All MZ-80B system software is supported by cassette tape or diskette files.

When the power switch of the MZ-80B is turned on, the Initial Program Loader (a file reading program mandatory for activation of system software) starts. The loader reads the system software from cassette tape or diskette files and, upon completion of loading, transfers system control to the loaded program.

This action takes place automatically the instant the power switch of the MZ-80B is turned on. Accordingly, in order to activate system software stored in a cassette tape file, you must load the cassette tape recorder with the corresponding cassette before turning on the MZ-80B; to activate system software stored in a diskette file, the corresponding diskette must be placed in drive No. 1 of the floppy disk unit connected to the MZ-80B before the power is turned on.

2.1.1 Activating system software contained in a cassette tape file

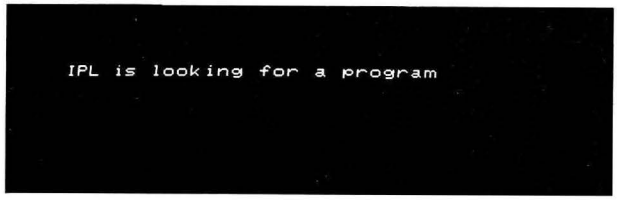
Load the cassette tape into the cassette tape recorder and energize the MZ-80B. See FIGURE 2.3.



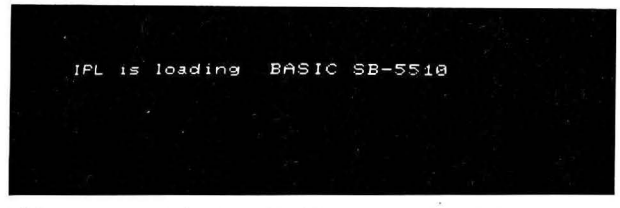
Load the cassette tape into the MZ-80B

FIGURE 2.3

The MZ-80B searches and loads the system software automatically. In this state the following messages are shown. See FIGURE 2.4.



This message indicates that the MZ-80B is searching for the system software on the tape.



This message indicates that loading of the BASIC interpreter is in progress.

FIGURE 2.4

FIGURE 2.5 shows that the loaded BASIC interpreter SB-5510 has been started.

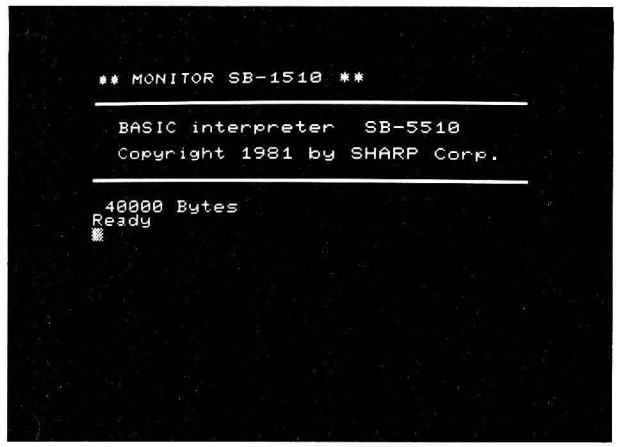


FIGURE 2.5 Message indicating that BASIC interpreter SB-5510 has been started Subsequently, the cassette tape is automatically rewound.

2.1.2 Activating system software stored in a diskette file

Energize the floppy disk unit and place the master diskette in drive No. 1; energize the MZ-80B. The MZ-80B loads the system software automatically.

After a few seconds, a message should appear indicating that DISK BASIC interpreter SB-6510 has been activated.

A special method of loading system programs from a ROM card connected to the expansion I/O port is possible. The IPL of the MZ-80B enables system program loading in this manner; when the IPL is started with the "/" (slash) key depressed, it loads the program from the memory connected to the expansion I/O port.

2.1.3 General flow chart for Initial Program Loading

Initial Program Loading is normally accomplished by the above simple operation.

Individual operations needed to perform Initial Program Loading in special cases (for example, when loading from a cassette tape file with the floppy disk unit connected to the MZ-80B; or when loading from a drive other than drive No. 1) and measures to prevent errors are described later.

FIGURE 2.6 depicts the general flow chart for Initial Program Loading. Execution of Initial Program Loading normally progresses as indicated by the solid line; however, manual operations may be required depending upon conditions at the branchpoints.

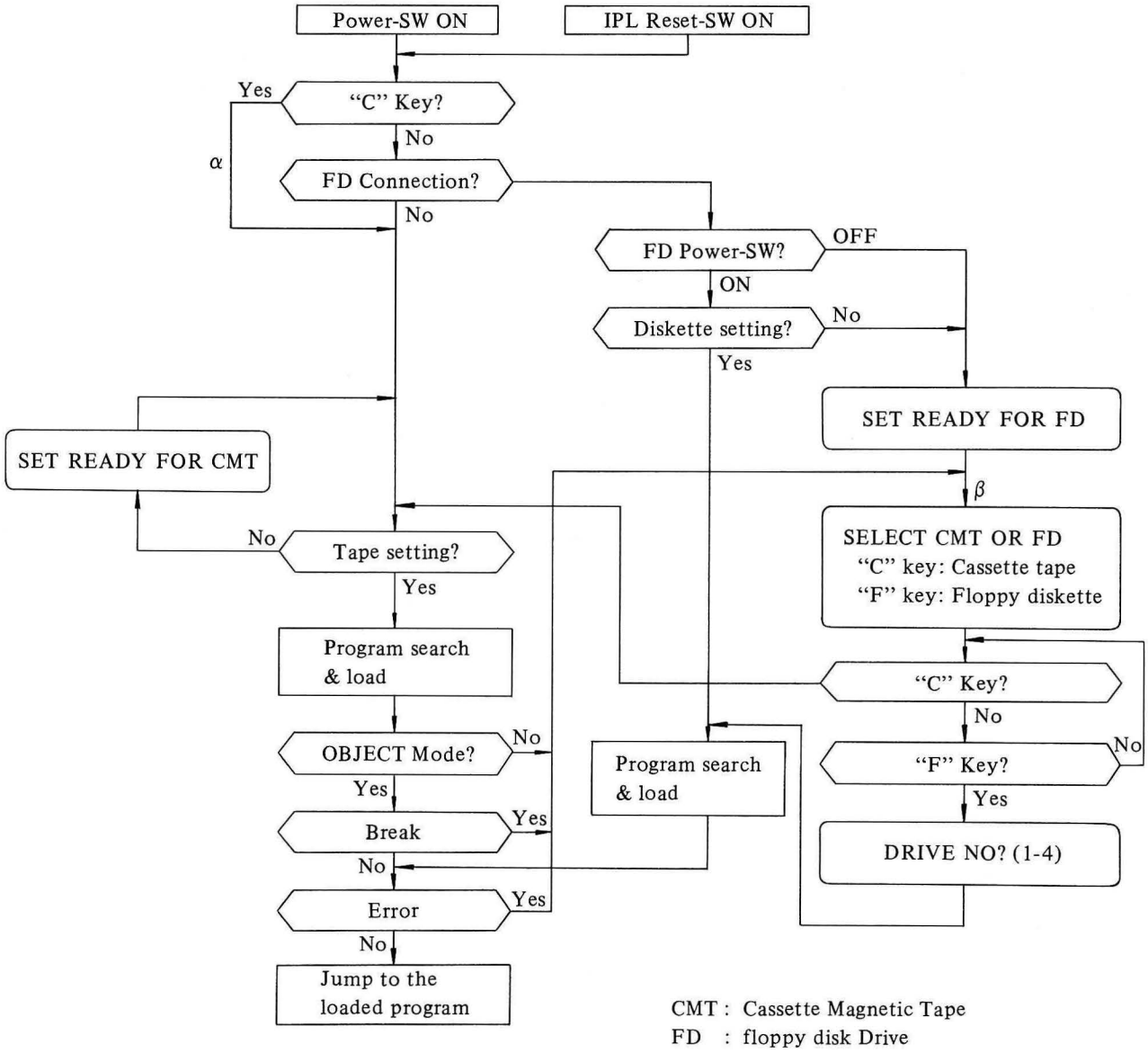


FIGURE 2.6 General flow chart for IPL

To read system software from a cassette tape with the floppy disk unit connected to the MZ-80B (or with the floppy disk interface card inserted in the I/O port of the MZ-80B), switch on the MZ-80B while pressing the [C] (cassette tape) key. (Loading control proceeds along flowline α.) Energizing the MZ-80B without pressing the [C] key drives the master diskette if it is contained in drive No. 1. When drive No. 1 is inoperative, however, when branch point β is reached the MZ-80B asks whether loading is to be made from cassette tape or a diskette. If the [C] key is then pressed, the cassette tape will undergo initial program loading.

If you intend to perform initial program loading from any drive other than drive No. 1, make drive No. 1 inoperative before turning the power on. The drive can be made inoperative by not inserting a diskette, by leaving its cover open or by switching it off.

Program loading will then proceed to branch point β, at which time the system asks whether cassette tape or diskette is specified. Press the [F] (floppy diskette) key. The system further asks which drive number is desired. Input the desired number by pressing the corresponding key.

When you must rewind the cassette tape before initial program loading, first initiate cassette-based loading, then press the [BREAK] key. This will cause loading control to move to branch point β, causing the tape to be rewound. When the tape is completely rewound, press the [C] key.

Pressing the [C] key before the tape is completely rewound causes the system to begin the file search immediately.

When you must fast forward the tape, first initiate cassette-based loading, then press the [BREAK] key. This will cause the tape to be rewound as described above. Press the [STOP] key of the cassette recorder to stop tape travel and press the [FF] key to fast forward. Interrupt tape travel again by pressing the [STOP] key, then press the [C] key to start the file search.

Initial program loading does not provide for discrimination between program texts according to file name. When loading from cassette tape the system reads the first OBJECT mode file it encounters. If the system encounters any file other than one in the OBJECT mode, it displays the error message "FILE MODE MISMATCH ERROR".

The memory map and other references for initial program loading are given in item 3 of the Appendix. The assembly listing for the initial program loader is shown in the **SUPPLEMENT**.

The MZ-80B system can, of course, read (through IPL) any system software you have worked out on the MONITOR SB-1510 or other systems. The MONITOR SB-1510 Reference Manual describes procedures for creating system software with the aid of MONITOR.

2.2 Keyboard

The keyboard of the MZ-80B is arranged as shown in Figure 2.7, and is divided into 5 areas according to function.

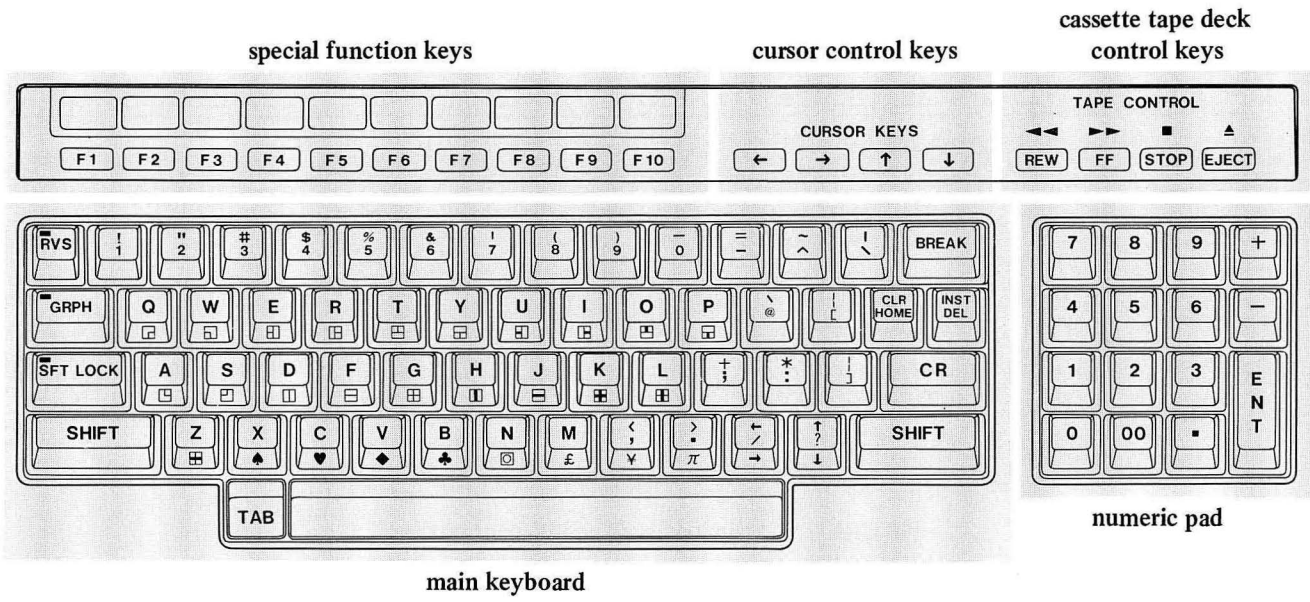


FIGURE 2.7 Locations of 5 areas of the keyboard

The main keyboard (typewriter keyboard) conforms to ASCII standards and includes character keys and control keys (such as the carriage return key and the break key).

The numeric pad is for entering numeric data and is similar to that of an ordinary electronic calculator.

The ten blue keys in the upper left are keys whose functions are defined by the user.

The four yellow keys in the upper center are cursor control keys, and the four green keys in the upper right are cassette tape deck control keys.

The functions of each element of the keyboard are explained in the following pages.

2.2.1 Main keyboard

The main keyboard is operated in a manner similar to that of an ordinary typewriter. One difference is that the main keyboard has three operating modes; another is that several control keys are provided (the stippled keys in Figure 2.8 are the control keys).

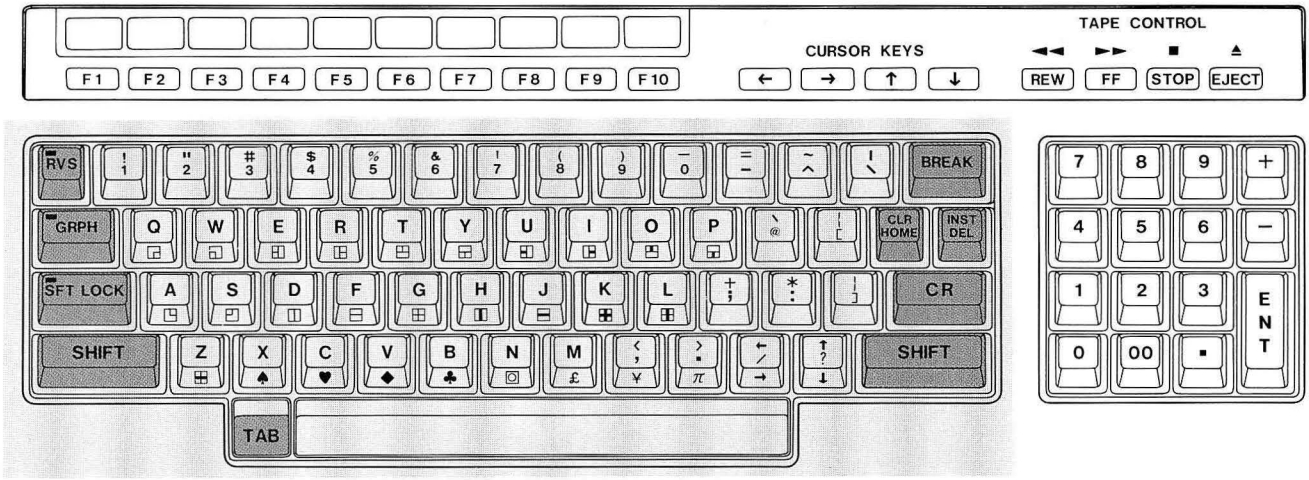


FIGURE 2.8 Main keyboard and its control keys

Three operation modes are as follows:

[1] Normal mode
[2] Graphic mode
[3] Reverse mode

Some of these keys produce different characters according to operation mode, as shown in Figure 2.9. Except under special circumstances, characters input from the main keyboard are displayed on the screen in the position where cursor is located.

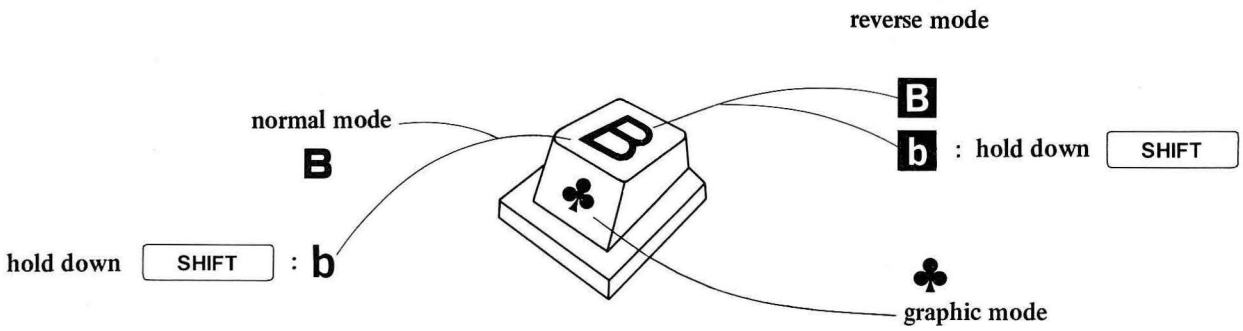


FIGURE 2.9 Different characters of a key

The functions of the control keys which are independent of operation mode are explained below.

[SHIFT] : Similar to the shift key of an ordinary typewriter; when this key is depressed, the character keys and some of the control keys are shifted.

[CR] : Carriage return key. When pressed, the cursor moves to the beginning of the following line.

CR: Abbreviation for carriage return.

[CLR HOME] : HOME returns the cursor to the upper left hand corner of the display screen. CLR clears the display screen and also returns the cursor to the screen's upper left hand corner.

CLR: clear

[INST DEL] : DEL erases the character at the left of the cursor location, shifting all following characters of the string to the left one space. INST inserts a space where the cursor is located by shifting all following characters of the string to the right one space.

DEL: delete, INST: insert

[SFT LOCK] : Shift lock key. When this key is pressed with the [SHIFT] key depressed, the [SHIFT] key is locked. When the [SHIFT] key is locked, [SFT LOCK] lamp lights. Pressing this key again without pressing the [SHIFT] key releases the shift lock.

SFT LOCK: shift lock

[GRPH] : With this key depressed the character keys which have graphic characters produce these graphic characters. If this key is pressed with the [SHIFT] key depressed, the graphic mode is entered and locked, and the [GRPH] key lamp lights.

GRPH: graphic

[RVS] : With this key depressed the character keys produce reversed characters. If this key key is pressed with the [SHIFT] key depressed, the reverse mode is entered and locked, and the [RVS] key lamp lights.

RVS: reverse

[BREAK] : When this key is pressed, a break code is generated. Pressing this key halts execution of BASIC programs.

[TAB] : Tabulation control key.

TAB: tabulation

[1] Normal mode

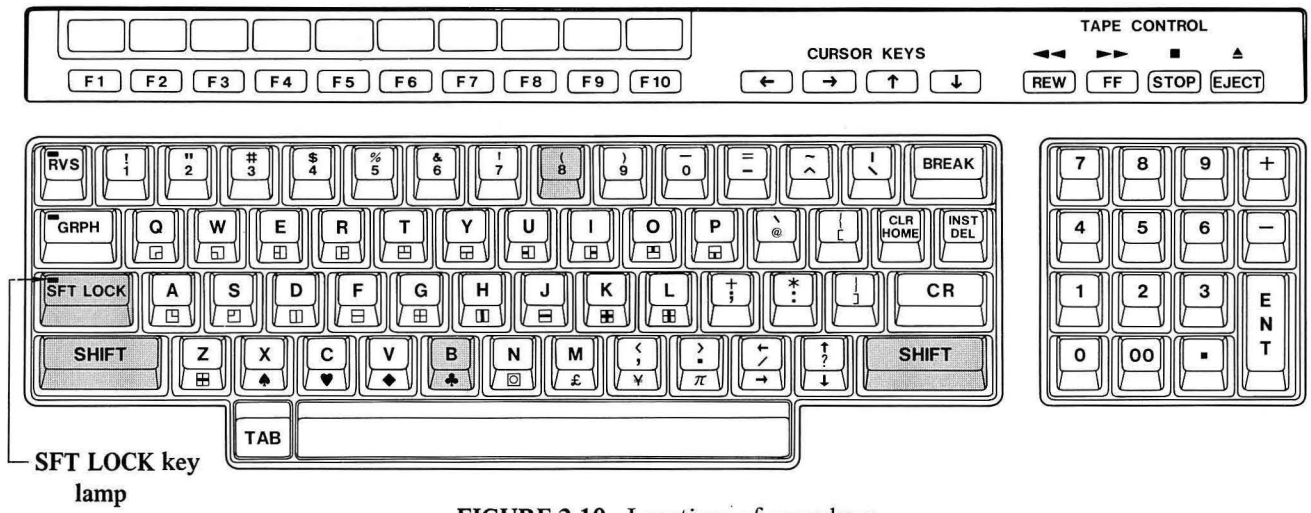


FIGURE 2.10 Locations of some keys

When the BASIC interpreter or another system program is started, keyboard operation is automatically set in the normal mode. Alphanumeric characters and symbols are input in the normal mode. For example, to input a B, press the [B] key (See Figure 2.10) in the same manner as on an ordinary typewriter. Note that the letter keys normally produce capital letters. To enter lower case letters, hold down the [SHIFT] key then press the letter key - just the opposite of an ordinary typewriter.

The reason for this is that capital letters are generally easier to read on the screen, so most people prefer to write their programs in capital letters. When a key has two non-alphabetic symbols on it, such as [(/8] (above the [U] key. See Figure 2.10.), pressing the key alone enters "8". If you hold down the [SHIFT] key while pressing [(/8], "(" will be entered. Only the 26 letter keys are shifted in the opposite direction from a standard typewriter.†

The [SFT LOCK] key locks the [SHIFT] key so that it does not need to be held down. When the [SHIFT] key is locked, the [SFT LOCK] lamp (See Figure 2.10) lights and pressing the [B] key inputs "b". Characters and symbols which can be input in the normal mode correspond to ASCII codes 20H to 7EH. (See Figure 2.22, ASCII code table.)

† The BASIC interpreter SB-5510 and DISK BASIC interpreter SB-6510 are provided with the CHANGE statement. With this statement, the shift direction of the 26 alphabetic characters, A to Z, entered from the keyboard can be changed.

[2] Graphic mode

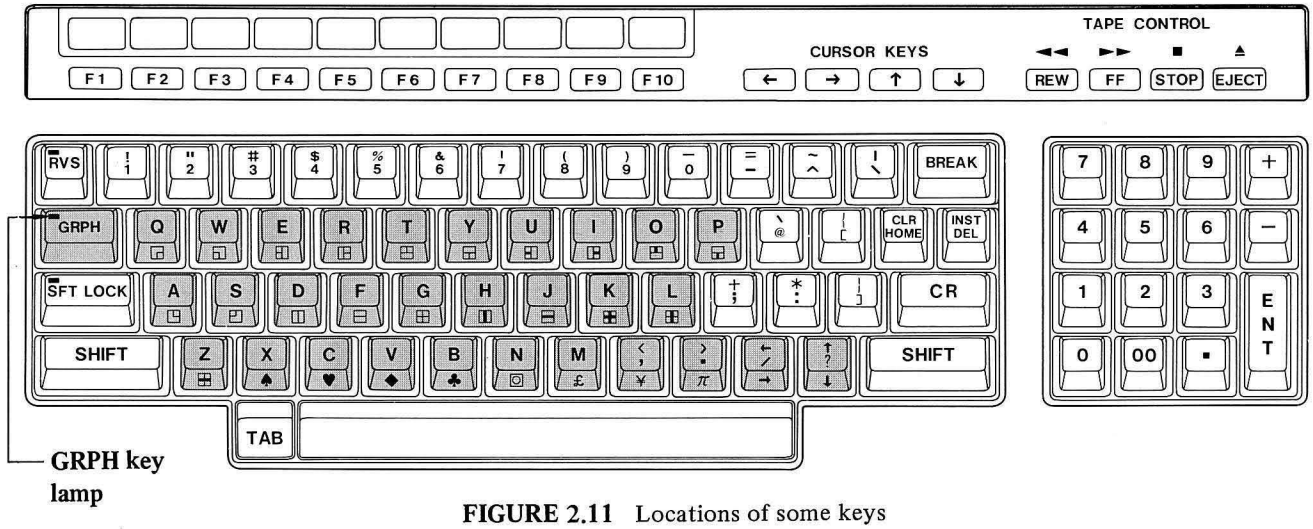


FIGURE 2.11 Locations of some keys

Graphic patterns produced by the stippled keys shown in Figure 2.11 may be input when the system is in the graphic mode. Each graphic pattern is printed in white on the front of each of these 30 keys.

For example, pressing the [B] key inputs the graphic pattern. When any key other than one of these 30 keys is pressed, the character assigned to the key is input. Note that graphic patterns cannot be input when the [SHIFT] key is held down.

Included in the graphic patterns are ruled line patterns which are provided for generating tables. Figure 2.12 shows an example of a table generated using ruled line patterns.

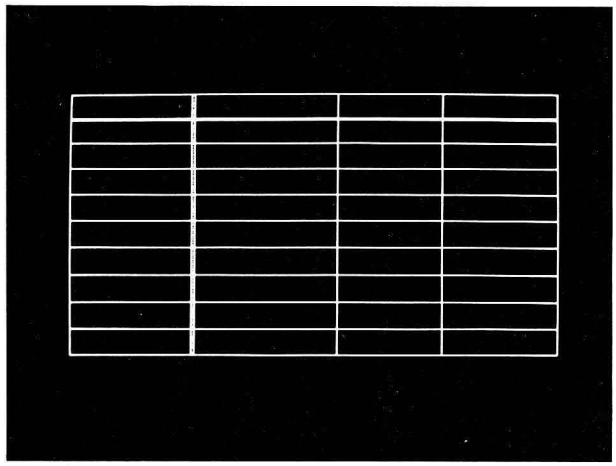


FIGURE 2.12 A table generated in the graphic mode

These graphic patterns correspond to ASCII code 80H to 9FH, (See Figure 2.22, ASCII code table.) Graphic patterns can be processed as string data in the same manner as other characters and symbols.

[3] Reverse mode

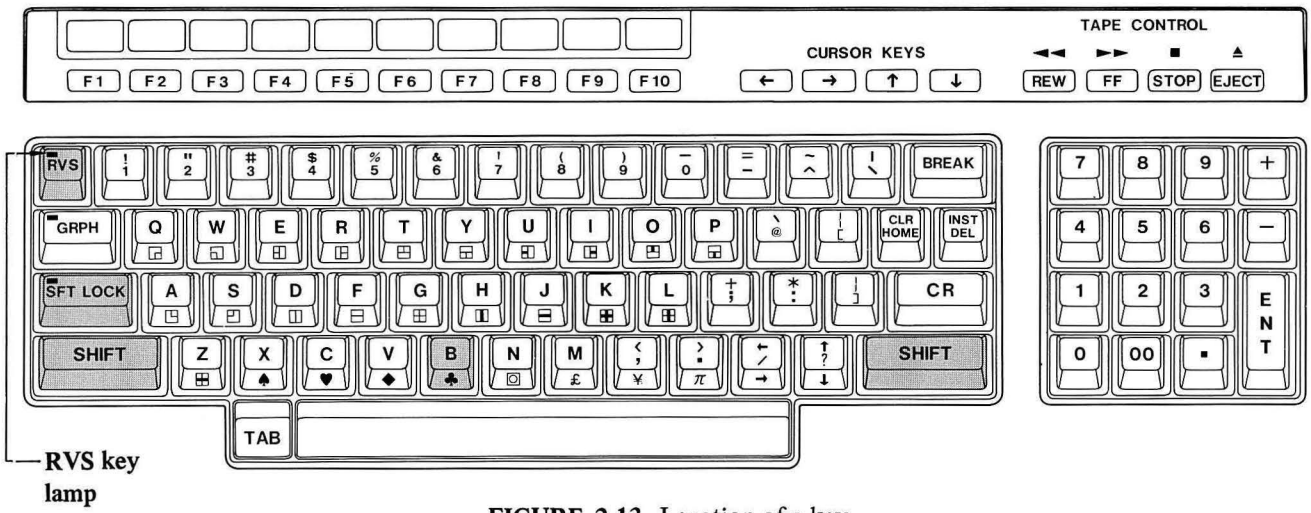


FIGURE 2.13 Location of a key

In the reverse mode, all characters and symbols which can be input in the normal mode appear on the screen in reverse highlighting.

For example, pressing the [B] key in the reverse mode inputs the reverse upper case character and pressing it with the [SHIFT] key depressed inputs the reverse lower case character.

The reverse characters and symbols correspond to ASCII codes AOH to FEH. See Figure A.1, ASCII code table.

As shown in Figure 2.21, the dot patterns constituting reverse characters are set/reset in the exact opposite state of those comprising normal characters.

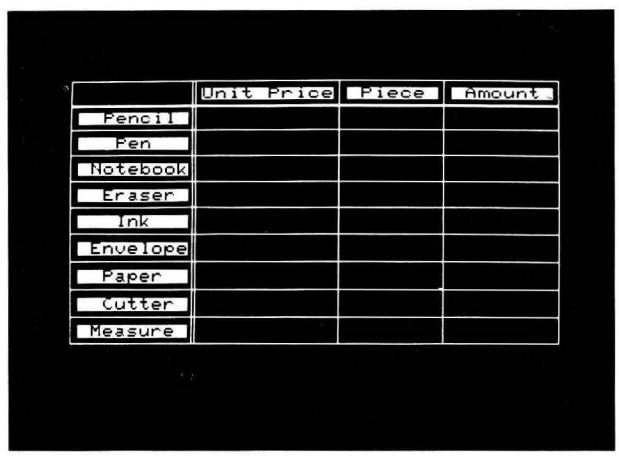


FIGURE 2.14 A title generated in the reverse mode

The entire display may be reversed by setting terminal PA_4 of programmable peripheral interface 8255 to high. For details, see Paragraph 4.3.1.

2.2.2 Numeric Pad

The group of keys on the right of the main keyboard is referred to as the numeric pad. It includes the numeral keys (0 through 9), and the [00] key, [▪] , [+] , [−] and [ENT] keys. These keys are provided on the numeric pad for the convenience of users who frequently enter numeric data.

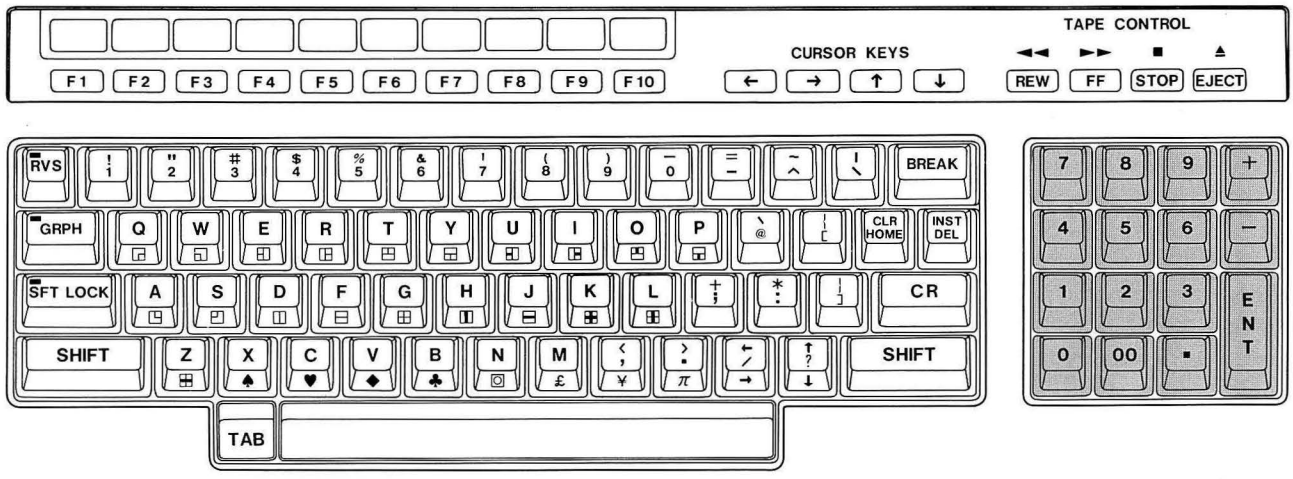


FIGURE 2.15 Location of number pad

When the [00] key is pressed once, two zeros are entered, just as if the [0] key were pressed twice.

A small projection is provided on the face of the [5] key so that the operator can enter numeric data without constantly looking at the keyboard.

All of the keys on the numeric pad operate without relation to the main keyboard operation mode or the [SHIFT] key.

[▪] , [+] and [−] keys are also provided on the main keyboard, along with the [CR] key, which has the same function as the [ENT] key on the numeric pad.†

† The [ENT] key is scanned by a different strobe signal than that which scans the [CR] key. This enables the machine language to differentiate between these keys. (See Paragraph 4.3.3)

2.2.3 Special Function Keys

The ten blue keys in the upper left of the keyboard, marked F1 through F10, are called special function keys. See Figure 2.16.

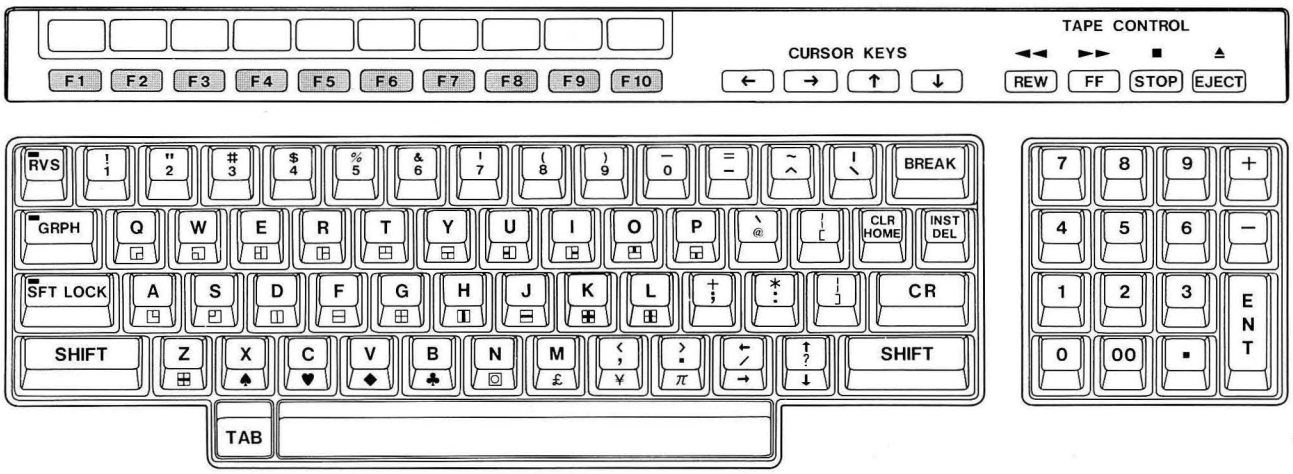


FIGURE 2.16 Location of Special Function keys

These keys are undefined when the MZ-80B is activated. The user can define a function for each of these keys by using the BASIC SB-5510 DEF KEY statement.

To define the function of special function key 1 as the BASIC command RUN, execute the following statement:

```
DEF KEY (1) = RUN
```

Once this statement is executed, special function key 1 performs the function of the RUN command until it is redefined. Thus, when special function key 1 is pressed in the direct mode, the following appears on the display. Then, by pressing [CR] key, the RUN command is executed.

The [CR] key can be defined together with the RUN command as the function of a special function key, if desired.

Execute

```
DEF KEY (1) = RUN ¬
```

The symbol " ¬ " represents the carriage return function, but there is no key on which this symbol appears. To enter this symbol, press the [SFT LOCK] and [GRPH] keys simultaneously. When special function key 1 is defined in this manner, the command may be executed just by pressing the key once.

It is convenient to define the functions of special function keys as direct mode commands and statements. However, numerical data and string data can also be assigned to these keys. The following statement assigns the character string "Personal Computer MZ-80B" to special function key 8.

```
DEF KEY (8) = Personal Computer MZ-80B
```

To obtain a listing of the definitions of the special function keys, execute the KLIST command.

When KLIST is executed, a list such as that shown in Figure 2.17 is displayed.

```
KLIST
1  LIST↵
2  RUN↵
3  RUN 100↵
4  AUTO↵
5  CONT↵
6  2.7182818
7  3.1415927
8  Personal Computer MZ-80B
9
10 KLIST↵
Ready
```

FIGURE 2.17 List of special function keys

This list shows that special function keys 1 through 5 are defined as commands plus the CR key; special function keys 6 through 8 are defined as data; special function key 9 is undefined and special function key 10 is defined as KLIST CR .

Labels are provided to enable the user to indicate the definition of each key under the transparent cover above each key.

These labels are useful when the same functions are assigned to the special function keys every time the MZ-80B is activated.

2.2.4 Cursor Control Keys

The tour yellow keys beside the special function keys are called cursor control keys. An arrow appears on the face of each key. See Figure 2.18.

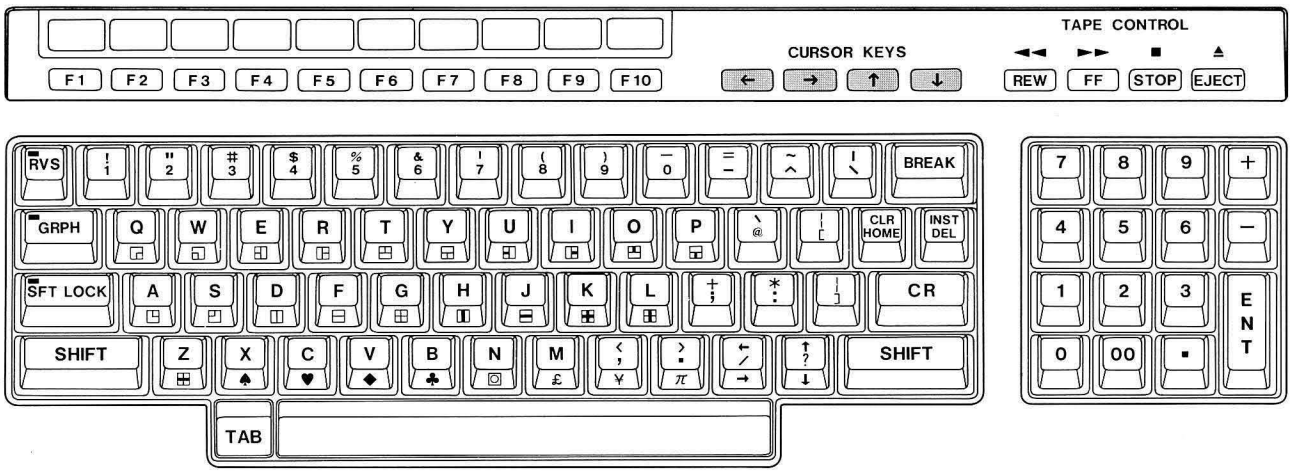


FIGURE 2.18 Location of Cursor Control Keys

Each key moves the cursor in the direction indicated by the arrow.

The cursor moves one position every time a cursor control key is pressed. Therefore, to move the cursor to the right 3 positions, the [→] key must be pressed three times.

To move the cursor repetitively, hold down the [SHIFT] key and press the appropriate cursor control key. The cursor will then move continuously until either of these keys is released.

When the cursor is to be moved to a position nearer to the upper left corner of the screen than to its current position, first press the [CLR HOME] key to move the cursor to the home position, then move it with the cursor control keys to save time.

2.2.5 Cassette Tape Deck Control Keys

The four green keys above the number pad are called cassette tape deck control keys. See Figure 2.19.

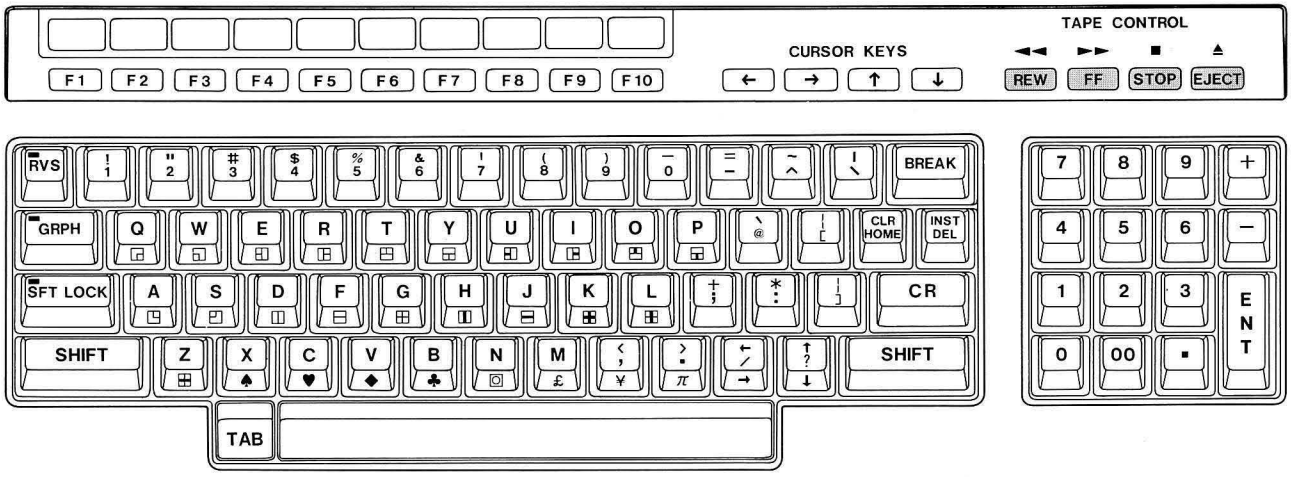


FIGURE 2.19 Location of Cassette Tape Deck Control Keys

These keys are connected directly to the cassette tape deck and perform a different role than the other keys.†

The functions of these keys are as follows:

- [REW] Rewinds the cassette tape.
- [FF] Fast forwards the cassette tape.
- [STOP] Stops the cassette tape.
- [EJECT] Ejects the cassette.

These functions have no relation to the mode in which the computer is operating.

Recording and reading data to/from the cassette tape are controlled by the software.

With BASIC SB-5510, recording or reading of program text is performed by the SAVE and LOAD commands, respectively. Recording or reading of data files is performed by the PRINT/T and INPUT/T statements, respectively.

Instructions for recording and reading are provided with all system software.

† All other keys are scanned by the Z80-PIO and processed by the software (See Paragraph 4.3.3), but the cassette tape deck control keys directly control the motor and eject mechanisms of the tape deck.

2.3 Display

There are two display control system: character display control and graphic display control. The character display control system displays character on the CRT screen using the character V-RAM and character generator. The graphic display control system displays optional curves and dot patterns of high resolution using the graphic V-RAM.

2.3.1 Character display control system

Characters generated by the character generator of the MZ-80B are shown in FIGURE 2.21 along with their corresponding ASCII codes.

As shown in the figure, character from "▩" ($1F) through "π" ($FF) can be displayed on the CRT screen.† Input of these characters from the keyboard was explained previously. Characters entered are displayed at the position where the cursor is located. The cursor pointer is controlled by the monitor program. The cursor position is changed by one of control codes $01~$06. See ASCII code table: FIGURE 2.22.

FIGURE 2.20 shows forms in which the character "A" can be displayed control of the character display control system.

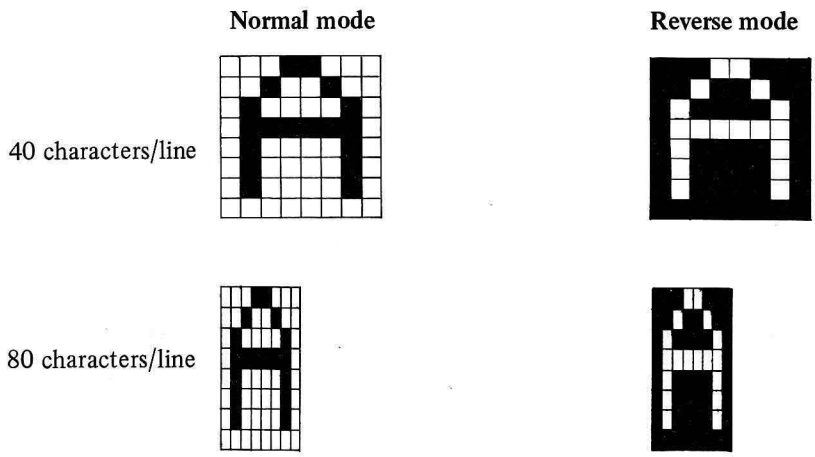


FIGURE 2.20 Character "A" displayed in various modes

† Some special characters, such as ⇦, ⊞ and ⓒ which indicate movement of cursor are displayed when a BASIC program file generated by the Sharp MZ-80K personal computer is converted.

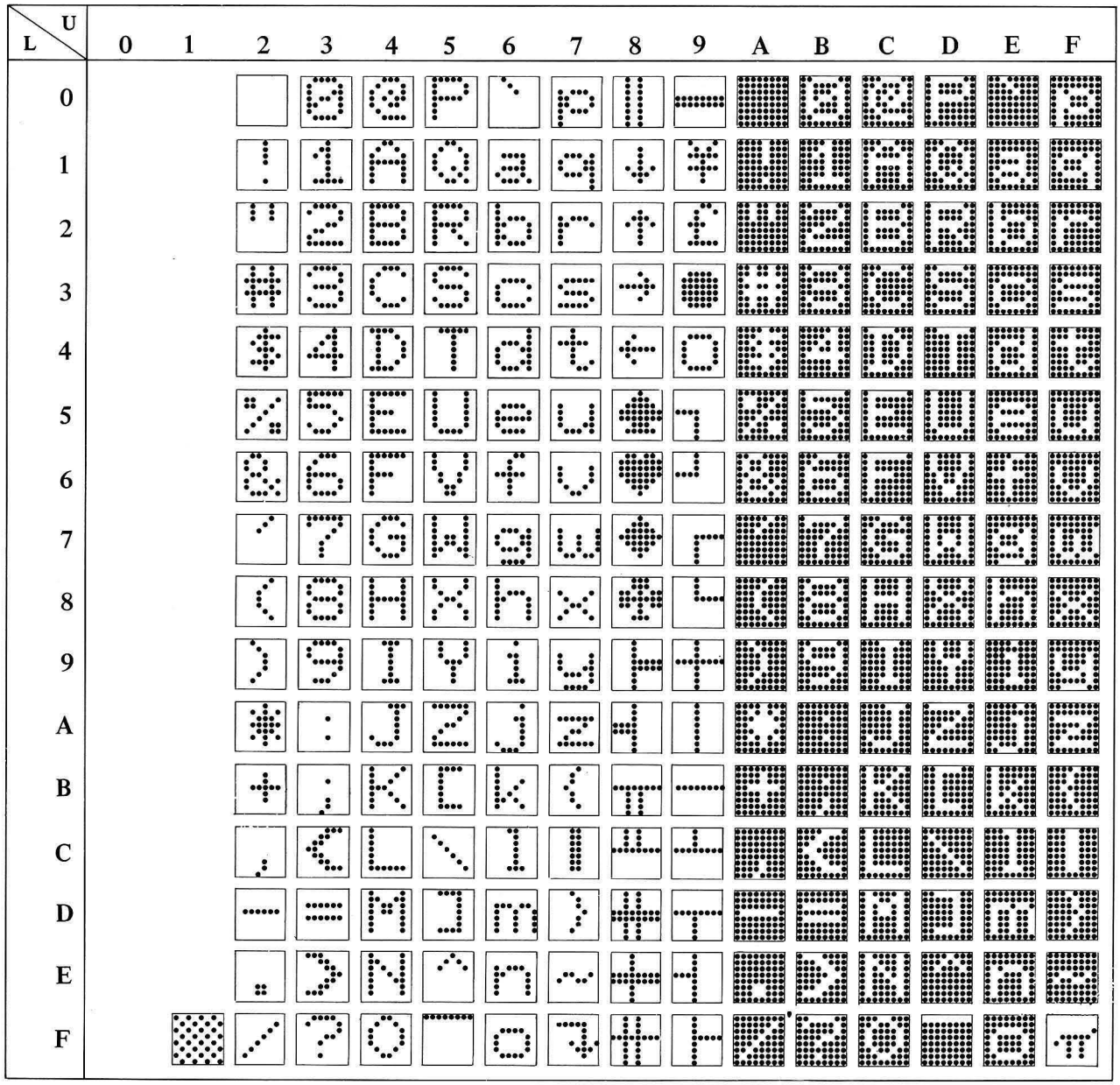


FIGURE 2.21 All characters along with corresponding ASCII Code

Note: U . . . . . Upper 4 bits
L . . . . . Lower 4 bits

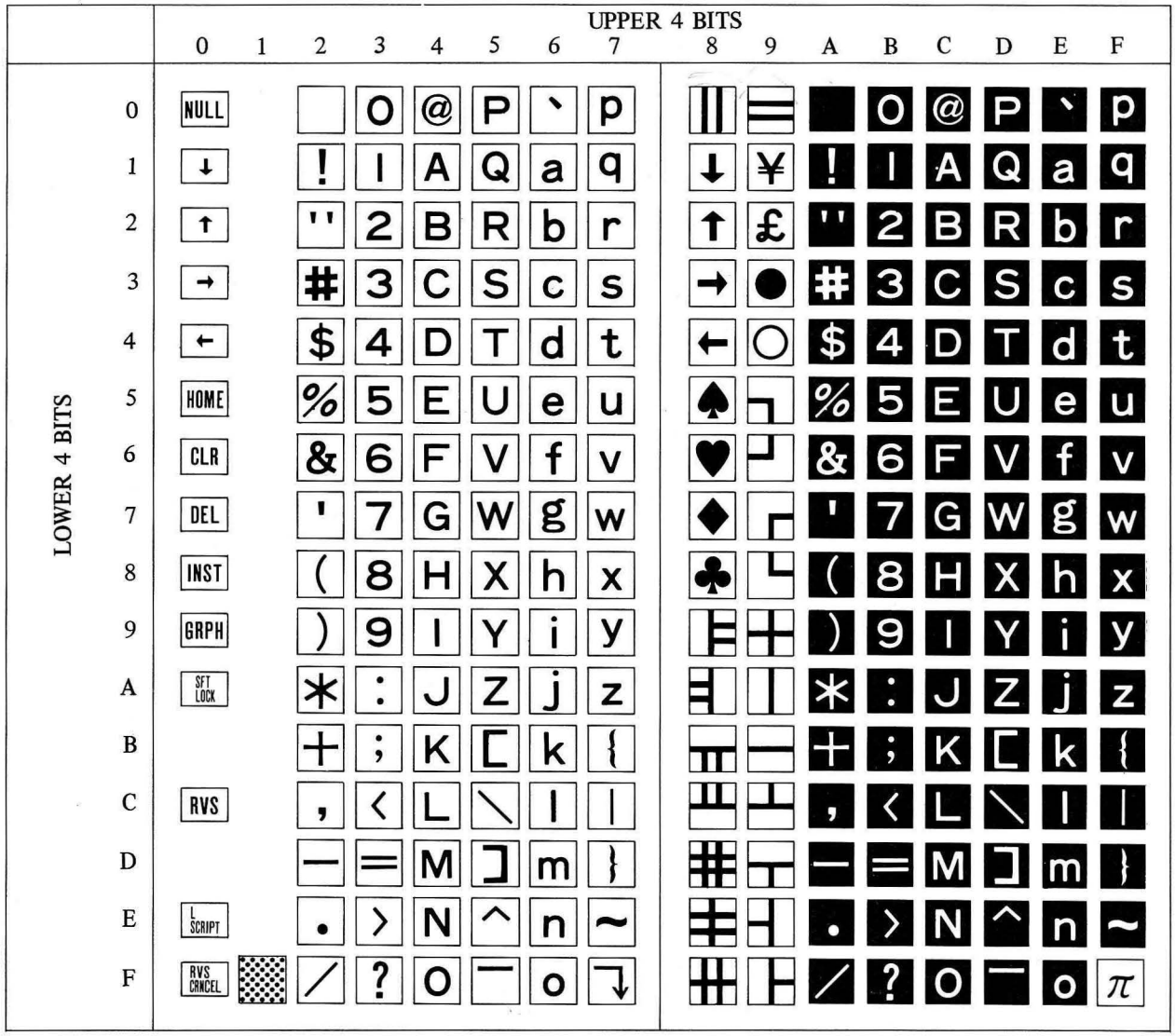


FIGURE 2.22 ASCII Codes of characters and control codes

2.3.2 Graphic display control system

FIGURE 2.23 shows an example of a projection of a three-dimensional object displayed using BASIC graphic control statements. Refer to BASIC Language Manual.

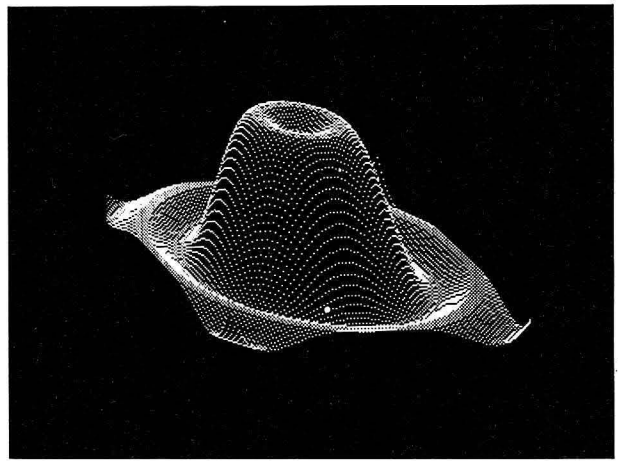


FIGURE 2.23

Chapter 3

Option Device Installation

This chapter describes procedures for installing optional devices in the main cabinet of the MZ-80B.

MZ-80RM	*Expansion Memory Module: 32K byte RAM card*
MZ-80GM	*Graphic Memory I: 8K byte RAM card*
MZ-80EU	*Expansion I/O Port*

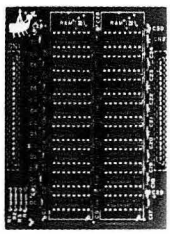
These optional devices must be installed properly according to procedures explained in this chapter.

Other optional devices are connected via the expansion I/O port. General procedures and notes on connecting optional devices via the expansion I/O port are contained in the last part of this chapter.

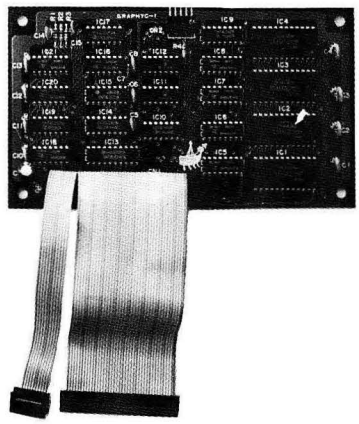
3.1 Installation of optional devices in the main cabinet of MZ-80B

Optional devices which can be installed in the main cabinet of the MZ-80B are expansion memory module MZ-80RM, graphic memory I MZ-80GM and expansion I/O port MZ-80EU.

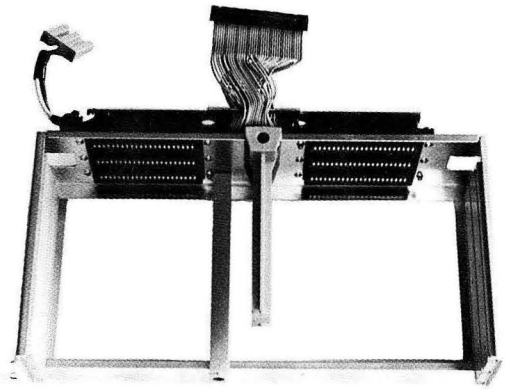
FIGURE 3.1 shows the locations in which these devices are installed.



MZ-80RM



MZ-80GM



MZ-80EU

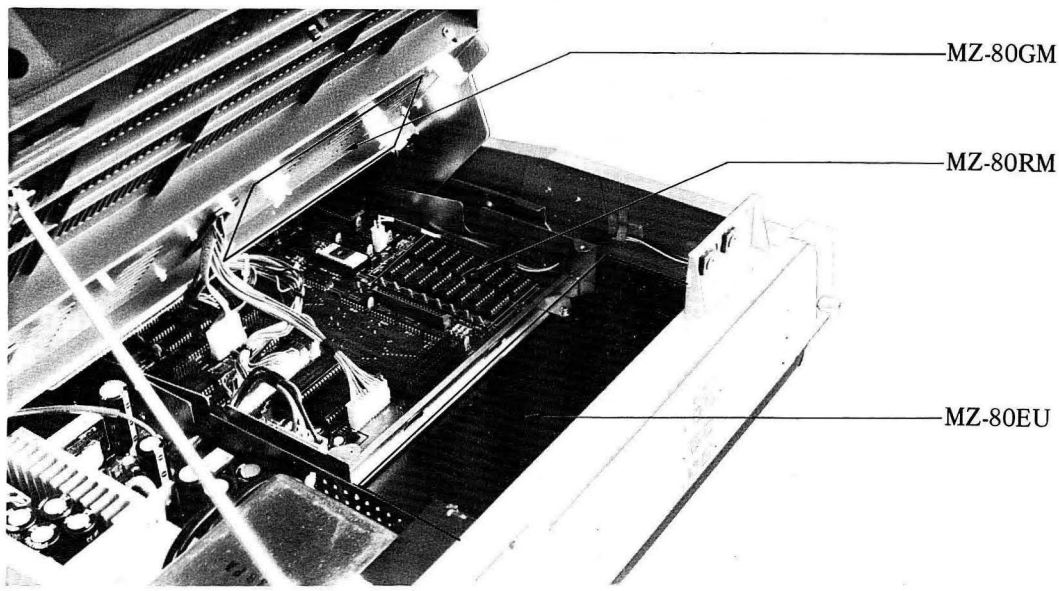


FIGURE 3.1

Before installing optional devices, the upper part of the MZ-80B, that is, the display and cassette tape deck section must be removed.

First, turn the MZ-80B power switch off and pull the power plug out of the AC outlet. Remove the two retaining screws on the rear side of the main cabinet. See FIGURE 3.2.

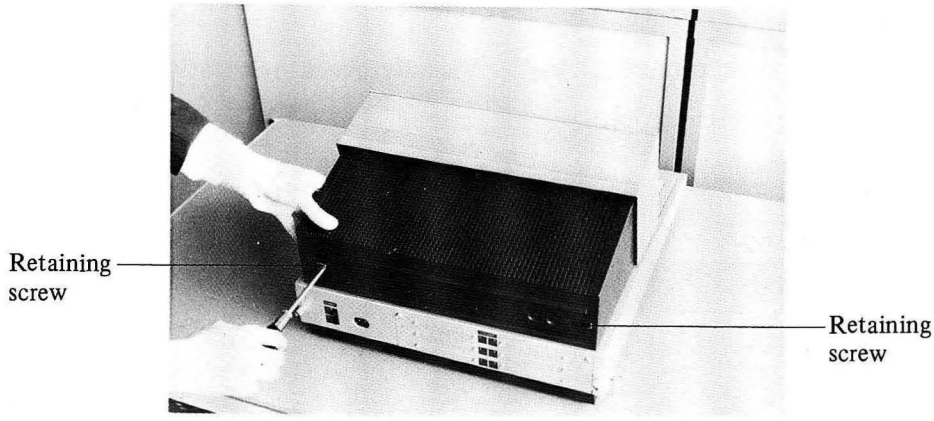


FIGURE 3.2

Gently lift the upper part of the main cabinet and support it with the supporting arm. See FIGURE 3.3.

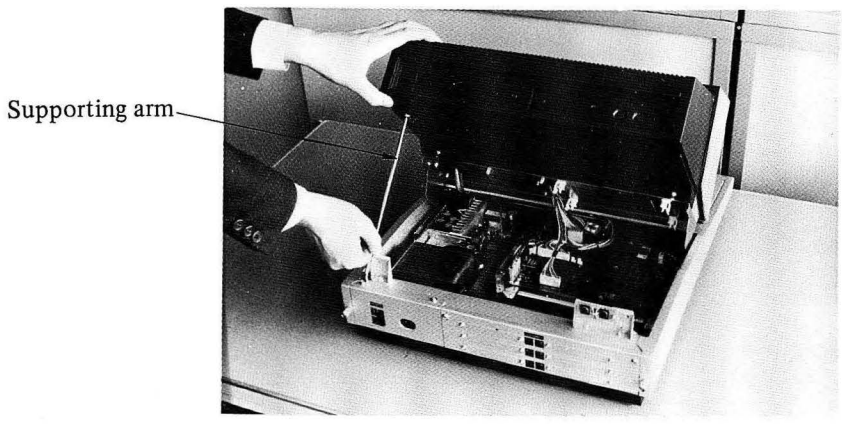


FIGURE 3.3

CAUTION: If the power is turned on with the upper part of the main cabinet lifted, electrical parts may be damaged.

Metal articles remaining in the cabinet can cause serious trouble.

Ensure that no paper clips or other metallic articles fall into cabinet.

3.1.1 Installing the Expansion RAM

The 32K byte expansion RAM card, MZ-80RM, is inserted in the 20 pin connector on the CPU board as shown in FIGURE 3.1. This connector is located on the right rear side of the CPU board as viewed from the rear. The standard 32K byte RAM card is already installed beside the expansion RAM connector. The connector pins on the bottom of the expansion RAM card can be inserted into the 20 pin connector on the CPU board.

The connector cannot be inserted backwards. Visually check orientation of the expansion RAM card before inserting it. See FIGURE 3.4.

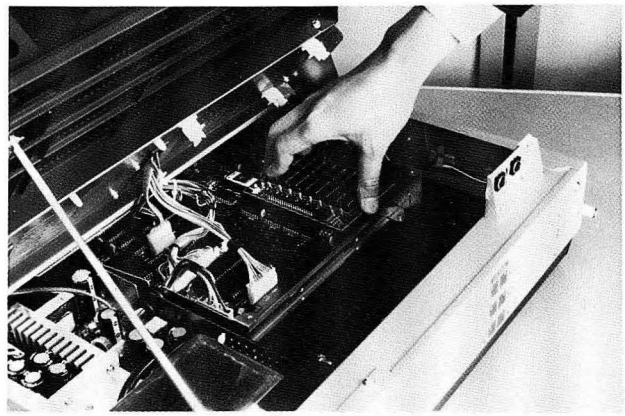


FIGURE 3.4

2.1 Initial program loading

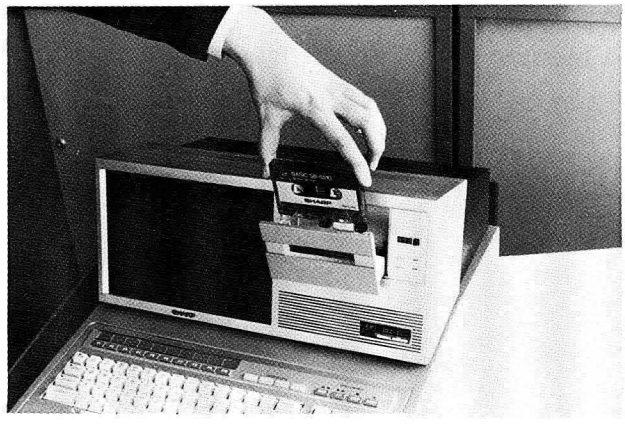
All MZ-80B system software is supported by cassette tape or diskette files.

When the power switch of the MZ-80B is turned on, the Initial Program Loader (a file reading program mandatory for activation of system software) starts. The loader reads the system software from cassette tape or diskette files and, upon completion of loading, transfers system control to the loaded program.

This action takes place automatically the instant the power switch of the MZ-80B is turned on. Accordingly, in order to activate system software stored in a cassette tape file, you must load the cassette tape recorder with the corresponding cassette before turning on the MZ-80B; to activate system software stored in a diskette file, the corresponding diskette must be placed in drive No. 1 of the floppy disk unit connected to the MZ-80B before the power is turned on.

2.1.1 Activating system software contained in a cassette tape file

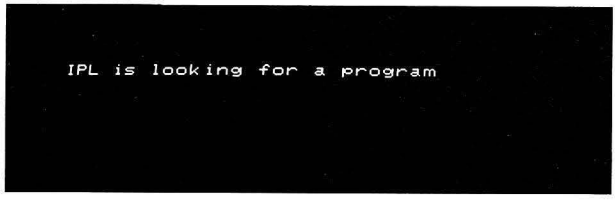
Load the cassette tape into the cassette tape recorder and energize the MZ-80B. See FIGURE 2.3.



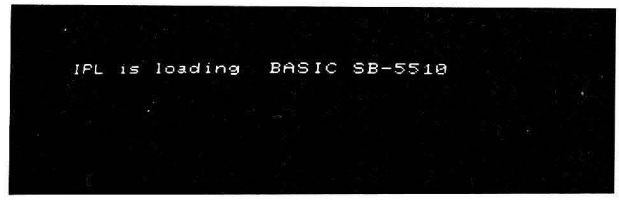
Load the cassette tape into the MZ-80B

FIGURE 2.3

The MZ-80B searches and loads the system software automatically. In this state the following messages are shown. See FIGURE 2.4.



This message indicates that the MZ-80B is searching for the system software on the tape.



This message indicates that loading of the BASIC interpreter is in progress.

FIGURE 2.4

FIGURE 2.5 shows that the loaded BASIC interpreter SB-5510 has been started.

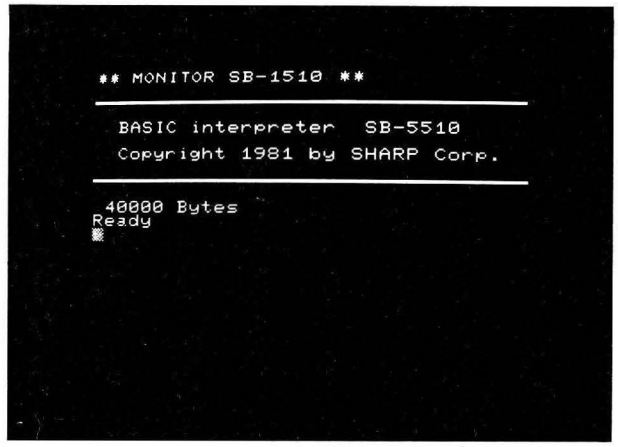


FIGURE 2.5 Message indicating that BASIC interpreter SB-5510 has been started Subsequently, the cassette tape is automatically rewound.

2.1.2 Activating system software stored in a diskette file

Energize the floppy disk unit and place the master diskette in drive No. 1; energize the MZ-80B. The MZ-80B loads the system software automatically.

After a few seconds, a message should appear indicating that DISK BASIC interpreter SB-6510 has been activated.

A special method of loading system programs from a ROM card connected to the expansion I/O port is possible. The IPL of the MZ-80B enables system program loading in this manner; when the IPL is started with the "/" (slash) key depressed, it loads the program from the memory connected to the expansion I/O port.

2.1.3 General flow chart for Initial Program Loading

Initial Program Loading is normally accomplished by the above simple operation.

Individual operations needed to perform Initial Program Loading in special cases (for example, when loading from a cassette tape file with the floppy disk unit connected to the MZ-80B; or when loading from a drive other than drive No. 1) and measures to prevent errors are described later.

FIGURE 2.6 depicts the general flow chart for Initial Program Loading. Execution of Initial Program Loading normally progresses as indicated by the solid line; however, manual operations may be required depending upon conditions at the branchpoints.

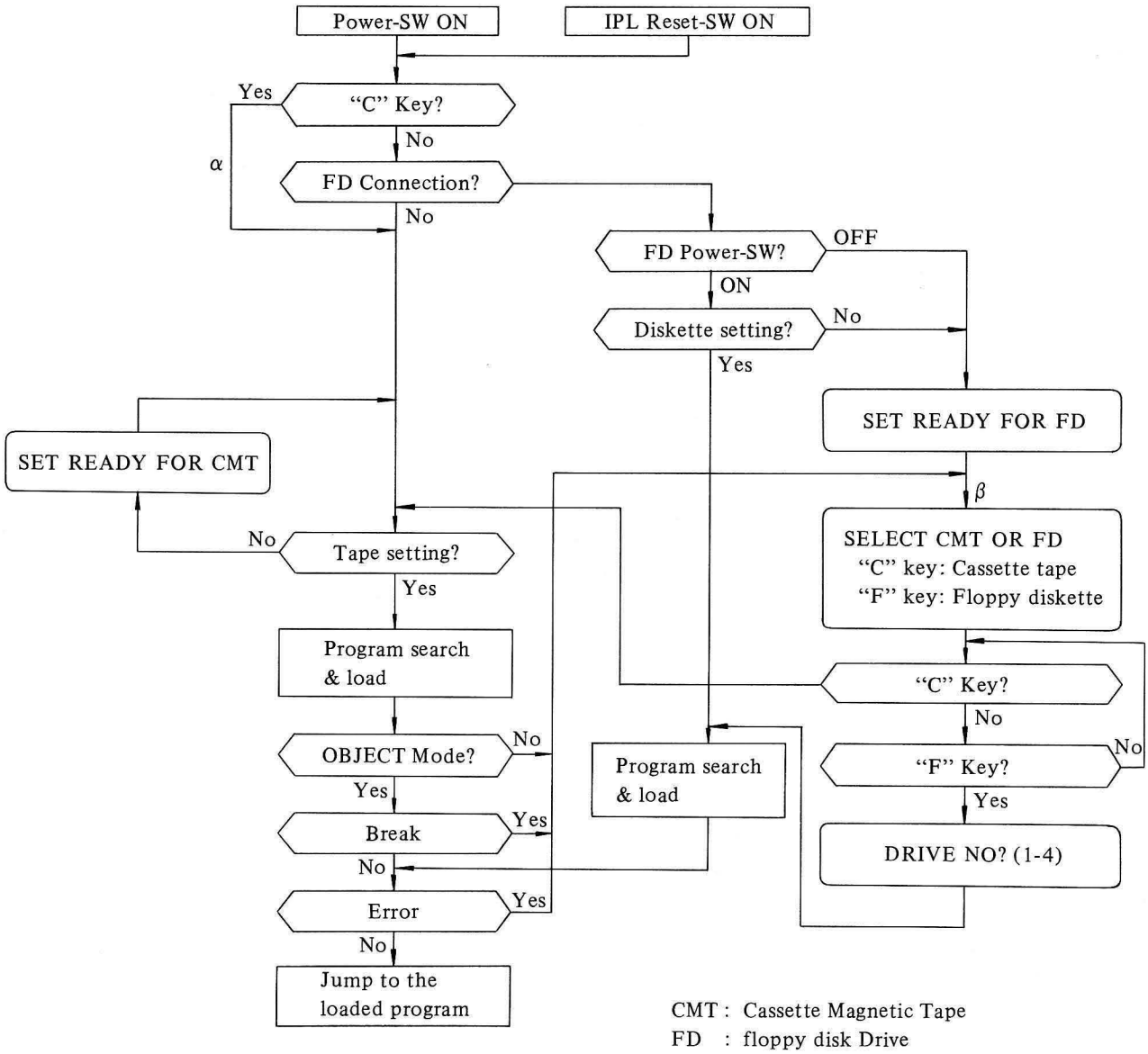


FIGURE 2.6 General flow chart for IPL

To read system software from a cassette tape with the floppy disk unit connected to the MZ-80B (or with the floppy disk interface card inserted in the I/O port of the MZ-80B), switch on the MZ-80B while pressing the [C] (cassette tape) key. (Loading control proceeds along flowline α.) Energizing the MZ-80B without pressing the [C] key drives the master diskette if it is contained in drive No. 1. When drive No. 1 is inoperative, however, when branch point β is reached the MZ-80B asks whether loading is to be made from cassette tape or a diskette. If the [C] key is then pressed, the cassette tape will undergo initial program loading.

If you intend to perform initial program loading from any drive other than drive No. 1, make drive No. 1 inoperative before turning the power on. The drive can be made inoperative by not inserting a diskette, by leaving its cover open or by switching it off.

Program loading will then proceed to branch point β, at which time the system asks whether cassette tape or diskette is specified. Press the [F] (floppy diskette) key. The system further asks which drive number is desired. Input the desired number by pressing the corresponding key.

When you must rewind the cassette tape before initial program loading, first initiate cassette-based loading, then press the [BREAK] key. This will cause loading control to move to branch point β, causing the tape to be rewound. When the tape is completely rewound, press the [C] key.

Pressing the [C] key before the tape is completely rewound causes the system to begin the file search immediately.

When you must fast forward the tape, first initiate cassette-based loading, then press the [BREAK] key. This will cause the tape to be rewound as described above. Press the [STOP] key of the cassette recorder to stop tape travel and press the [FF] key to fast forward. Interrupt tape travel again by pressing the [STOP] key, then press the [C] key to start the file search.

Initial program loading does not provide for discrimination between program texts according to file name. When loading from cassette tape the system reads the first OBJECT mode file it encounters. If the system encounters any file other than one in the OBJECT mode, it displays the error message "FILE MODE MISMATCH ERROR".

The memory map and other references for initial program loading are given in item 3 of the Appendix. The assembly listing for the initial program loader is shown in the **SUPPLEMENT**.

The MZ-80B system can, of course, read (through IPL) any system software you have worked out on the MONITOR SB-1510 or other systems. The MONITOR SB-1510 Reference Manual describes procedures for creating system software with the aid of MONITOR.

2.2 Keyboard

The keyboard of the MZ-80B is arranged as shown in Figure 2.7, and is divided into 5 areas according to function.

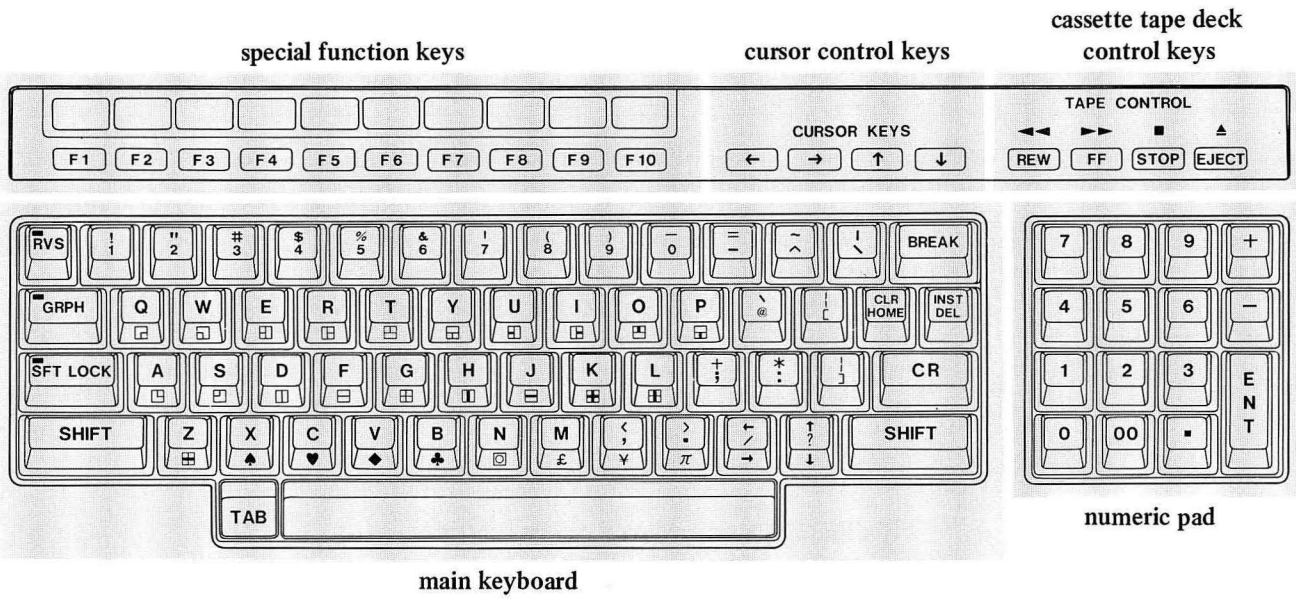


FIGURE 2.7 Locations of 5 areas of the keyboard

The main keyboard (typewriter keyboard) conforms to ASCII standards and includes character keys and control keys (such as the carriage return key and the break key).

The numeric pad is for entering numeric data and is similar to that of an ordinary electronic calculator.

The ten blue keys in the upper left are keys whose functions are defined by the user.

The four yellow keys in the upper center are cursor control keys, and the four green keys in the upper right are cassette tape deck control keys.

The functions of each element of the keyboard are explained in the following pages.

2.2.1 Main keyboard

The main keyboard is operated in a manner similar to that of an ordinary typewriter. One difference is that the main keyboard has three operating modes; another is that several control keys are provided (the stippled keys in Figure 2.8 are the control keys).

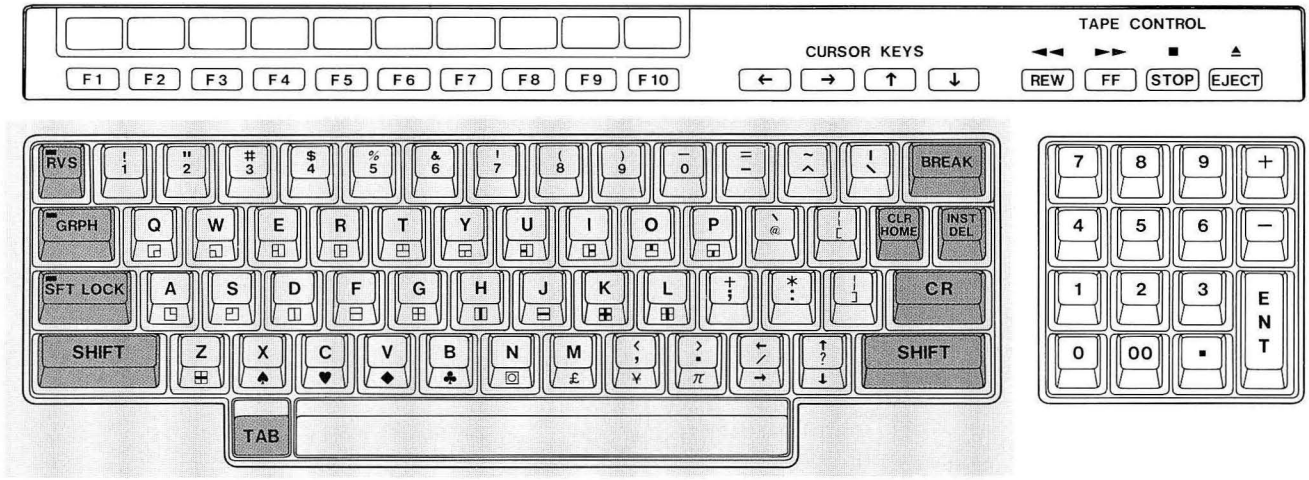


FIGURE 2.8 Main keyboard and its control keys

Three operation modes are as follows:

[1] Normal mode
[2] Graphic mode
[3] Reverse mode

Some of these keys produce different characters according to operation mode, as shown in Figure 2.9. Except under special circumstances, characters input from the main keyboard are displayed on the screen in the position where cursor is located.

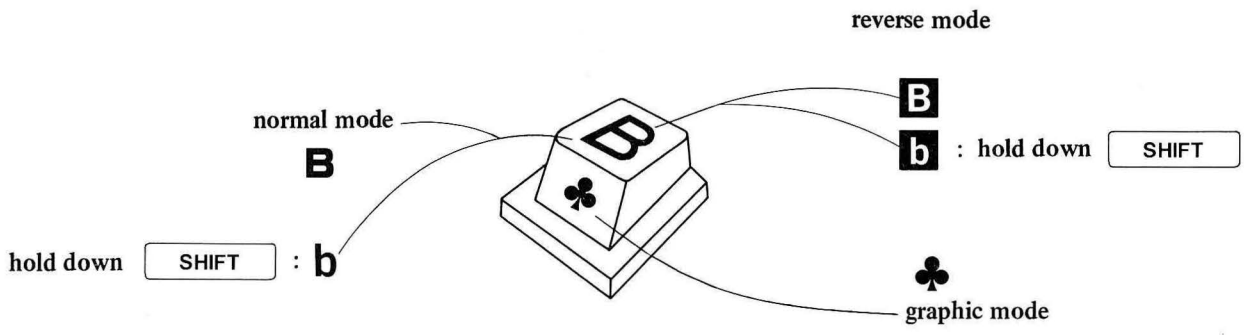


FIGURE 2.9 Different characters of a key

The functions of the control keys which are independent of operation mode are explained below.

SHIFT : Similar to the shift key of an ordinary typewriter; when this key is depressed, the character keys and some of the control keys are shifted.

CR : Carriage return key. When pressed, the cursor moves to the beginning of the following line.

CR: Abbreviation for carriage return.

CLR HOME : HOME returns the cursor to the upper left hand corner of the display screen. CLR clears the display screen and also returns the cursor to the screen's upper left hand corner.

CLR: clear

INST DEL : DEL erases the character at the left of the cursor location, shifting all following characters of the string to the left one space. INST inserts a space where the cursor is located by shifting all following characters of the string to the right one space.

DEL: delete, INST: insert

SFT LOCK : Shift lock key. When this key is pressed with the SHIFT key depressed, the SHIFT key is locked. When the SHIFT key is locked, SFT LOCK lamp lights. Pressing this key again without pressing the SHIFT key releases the shift lock.

SFT LOCK: shift lock

GRPH : With this key depressed the character keys which have graphic characters produce these graphic characters. If this key is pressed with the SHIFT key depressed, the graphic mode is entered and locked, and the GRPH key lamp lights.

GRPH: graphic

RVS : With this key depressed the character keys produce reversed characters. If this key key is pressed with the SHIFT key depressed, the reverse mode is entered and locked, and the RVS key lamp lights.

RVS: reverse

BREAK : When this key is pressed, a break code is generated. Pressing this key halts execution of BASIC programs.

TAB : Tabulation control key.

TAB: tabulation

[1] Normal mode

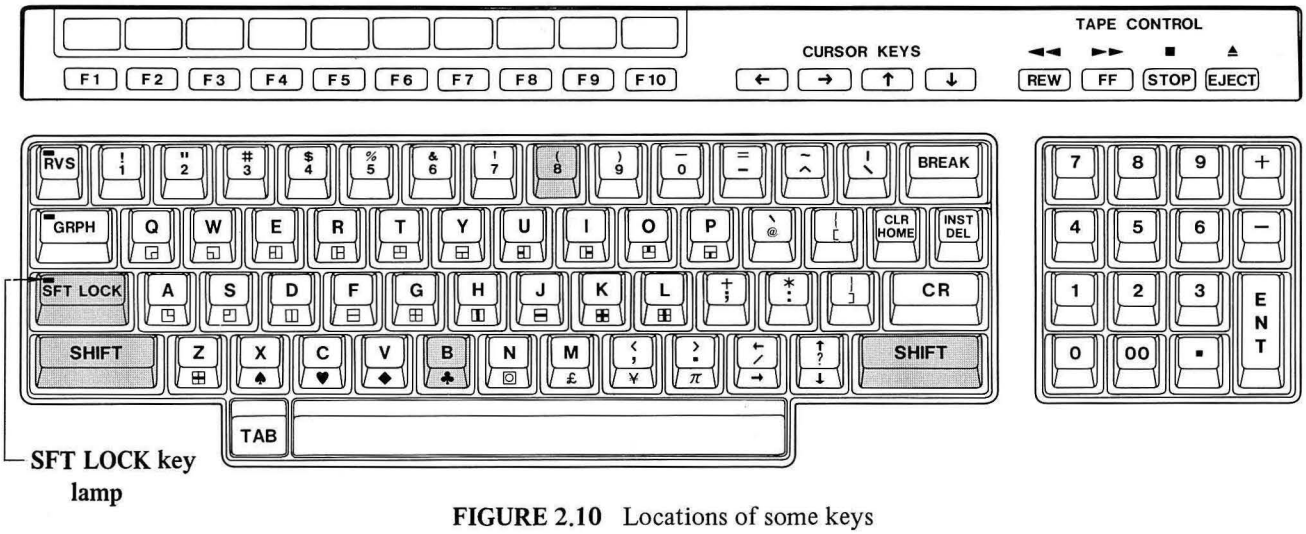


FIGURE 2.10 Locations of some keys

When the BASIC interpreter or another system program is started, keyboard operation is automatically set in the normal mode. Alphanumeric characters and symbols are input in the normal mode. For example, to input a B, press the [B] key (See Figure 2.10) in the same manner as on an ordinary typewriter. Note that the letter keys normally produce capital letters. To enter lower case letters, hold down the [SHIFT] key then press the letter key - just the opposite of an ordinary typewriter.

The reason for this is that capital letters are generally easier to read on the screen, so most people prefer to write their programs in capital letters. When a key has two non-alphabetic symbols on it, such as [(/8] (above the [U] key. See Figure 2.10.), pressing the key alone enters "8". If you hold down the [SHIFT] key while pressing [(/8], "(" will be entered. Only the 26 letter keys are shifted in the opposite direction from a standard typewriter.†

The [SFT LOCK] key locks the [SHIFT] key so that it does not need to be held down. When the [SHIFT] key is locked, the [SFT LOCK] lamp (See Figure 2.10) lights and pressing the [B] key inputs "b". Characters and symbols which can be input in the normal mode correspond to ASCII codes 20H to 7EH. (See Figure 2.22, ASCII code table.)

† The BASIC interpreter SB-5510 and DISK BASIC interpreter SB-6510 are provided with the CHANGE statement. With this statement, the shift direction of the 26 alphabetic characters, A to Z, entered from the keyboard can be changed.

4.2.2 Memory map for normal state

When the machine shifts from the IPL state to the normal state, the memory map becomes as shown in FIGURE 4.3.

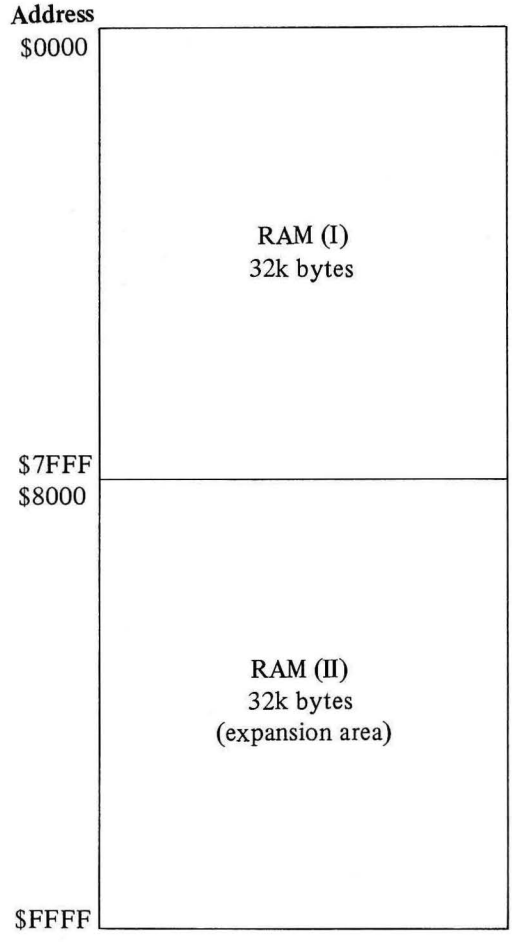


FIGURE 4.3 Memory map for normal state

In the case of the 32 k bytes standard RAM, the addresses range from $0000 to $7FFF; in the case of RAM 64 k bytes full equipment, the address space will be full area of $0000 to $FFFF.

When addresses are changed from IPL state to normal state or vice versa, the execution is controlled by C_1, C_3 output terminal signals of port C of the 8255 as described later.

4.2.3 Memory map for V-RAM accessing state

The memory addresses are changed over also when accessing the V-RAM. The RAM addresses in normal state are from $0000 to $7FFF for RAM (I), and from $8000 to $FFFF for RAM (II). In the case of V-RAM access, addressing of $D000 to $FFFF in RAM (II) is disabled, so that the V-RAM will be the object of access.

This changeover is effected by A_7 of PIO. When two pages of graphic V-RAM are used, selection of graphic pages (I) and (II) will be done by OUT port $F4. This operation is shown in FIGURE 4.4.

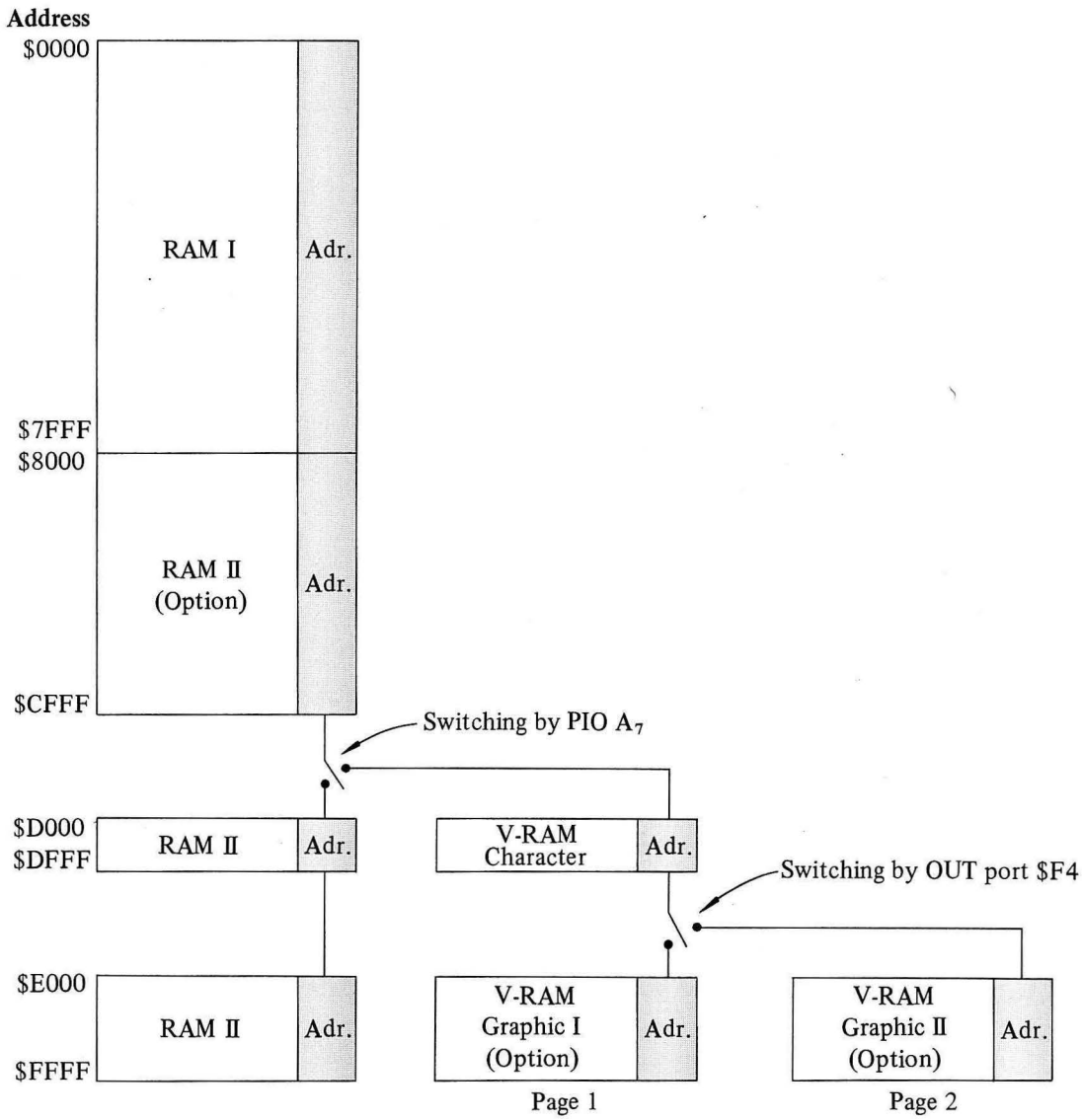


FIGURE 4.4 Switching of main memory and V-RAM (1)

The address switching of V-RAM shown above is effected by the monitor subroutine PRINT or the like.

On the other hand, when accessing the V-RAM with the program in main memory following $D000, the V-RAM addresses may be changed to $5000 to $7FFF, which is realized by setting the A_6 terminal of PIO to high state.

That is, the following instructions are executed. (At this time, the A_7 terminal of PIO may be either high or low state.)

```
IN      A, (E8H)
SET     6, A
OUT     (E8H), A
```

This address switching is shown in FIGURE 4.5.

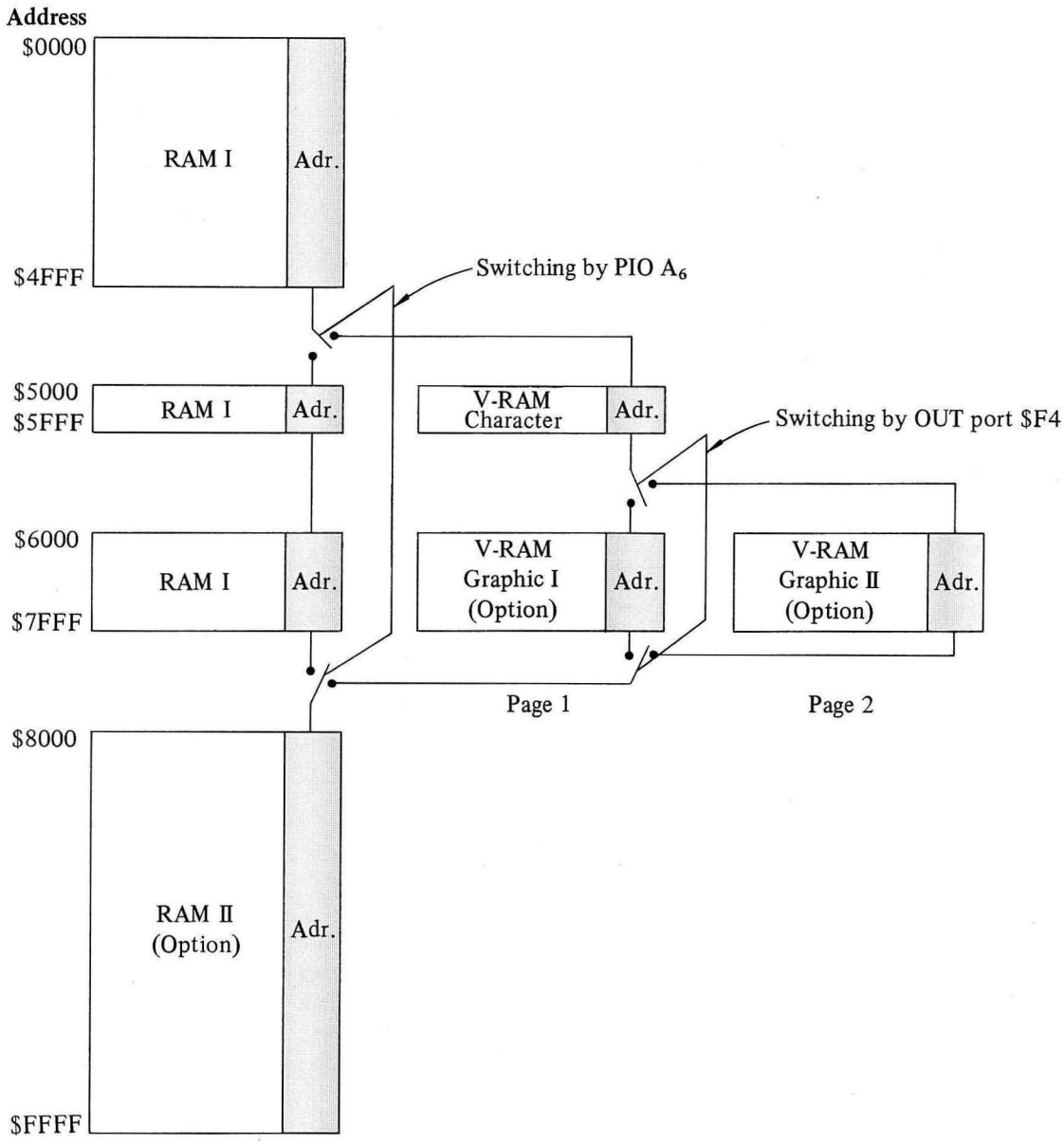


FIGURE 4.5 Switching of main memory and V-RAM (2)

The relation between the V-RAM output and CRT display is shown in FIGURE 4.6.

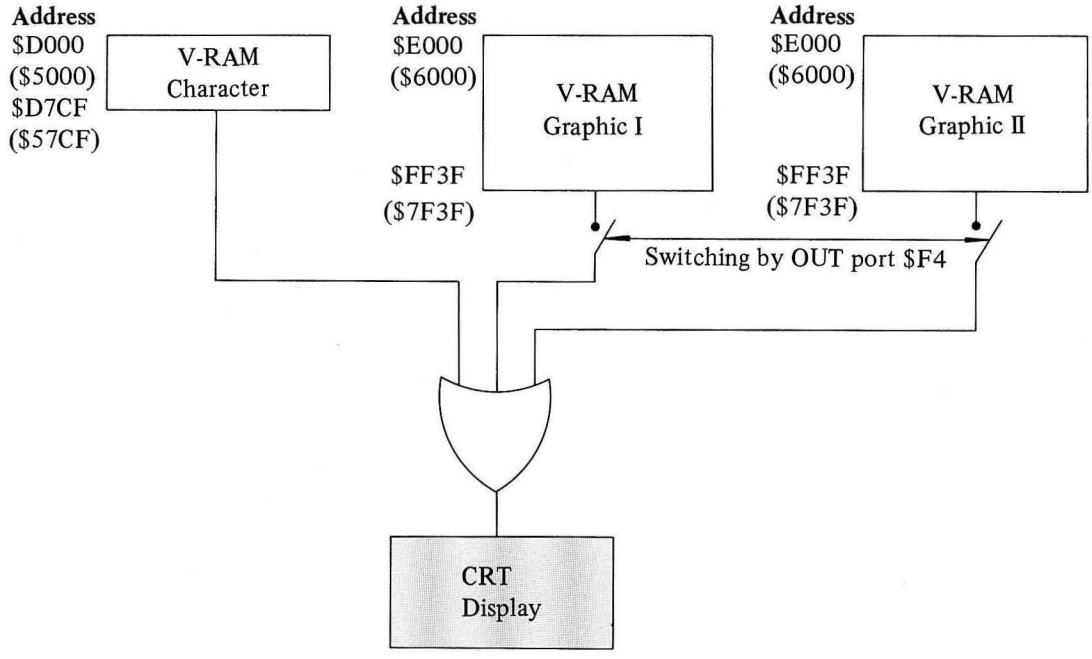


FIGURE 4.6 Relation between V-RAM and CRT display.

As shown in the figure, the V-RAM characters and graphic (I) or (II) can be displayed simultaneously.

The relation between the V-RAM addresses and corresponding positions on the CRT display is shown in FIGURE 4.7.

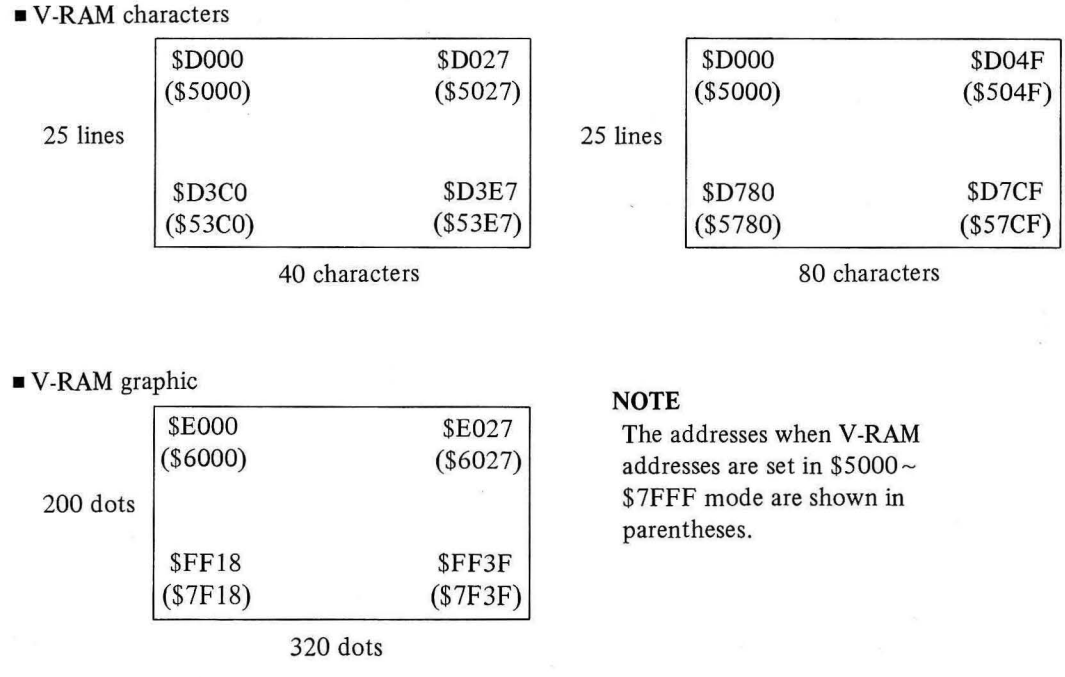


FIGURE 4.7 V-RAM addresses and CRT display

The input and output of V-RAM for graphic can be controlled as follows by means of the data delivered to the OUT port $F4.

Output data to port $F4	V-RAM GRPH I		V-RAM GRPH II	
	Input	Output	Input	Output
00	○	×	×	×
01	×	×	○	×
02	○	○	×	×
03	×	○	○	×
0C	○	×	×	○
0D	×	×	○	○
0E	○	○	×	○
0F	×	○	○	○

Note Input ○: V-RAM transfer enabled
×: V-RAM transfer disabled
Output ○: shown on CRT display
×: not shown on CRT display

Suppose 03H is delivered to $F4 port with 01H being stored in $E000, then 01H is transferred to V-RAM G-II $E000, but not shown on CRT display because the display indicator is set at V-RAM G-I.

4.3 Signal system for the 8255 block, the 8253 block and the PIO block

In this paragraph, the constitutions signal systems for the 8255 block, the 8253 block and the PIO block – which are responsible for essential roles in the system control – are illustrated.

Before description of each section, below are shown the settings of the 8255, 8253 and PIO in the input/output ports, and also summarized are the service modes of the port of each controller.

Table 4.2

CPU's Input/Output port	Controller	Service mode of each port
$E0 $E1 $E2 $E3	8255	P_A : output P_B : input P_C : output mode control
$E4 $E5 $E6 $E7	8253	C_0 : mode 2 (16 bit rate generator) C_1 : mode 2 (16 bit rate generator) C_2 : mode 2 (16 bit rate generator) mode control
$E8 $E9 $EA $EB	Z80A-PIO	A : output mode 3 (bit control) mode control A B : input mode 3 (bit control) mode control B

4.3.1 Signal system for the 8255 block

The 8255 (Programmable peripheral interface) is responsible for control of automatic cassette deck, reverse operation of CRT display, blank control, memory switching between IPL state and normal state, output control of source pulse for generating sound, and lighting control of keyboard LEDs.

FIGURE 4.8 summarizes the signal system for the 8255.

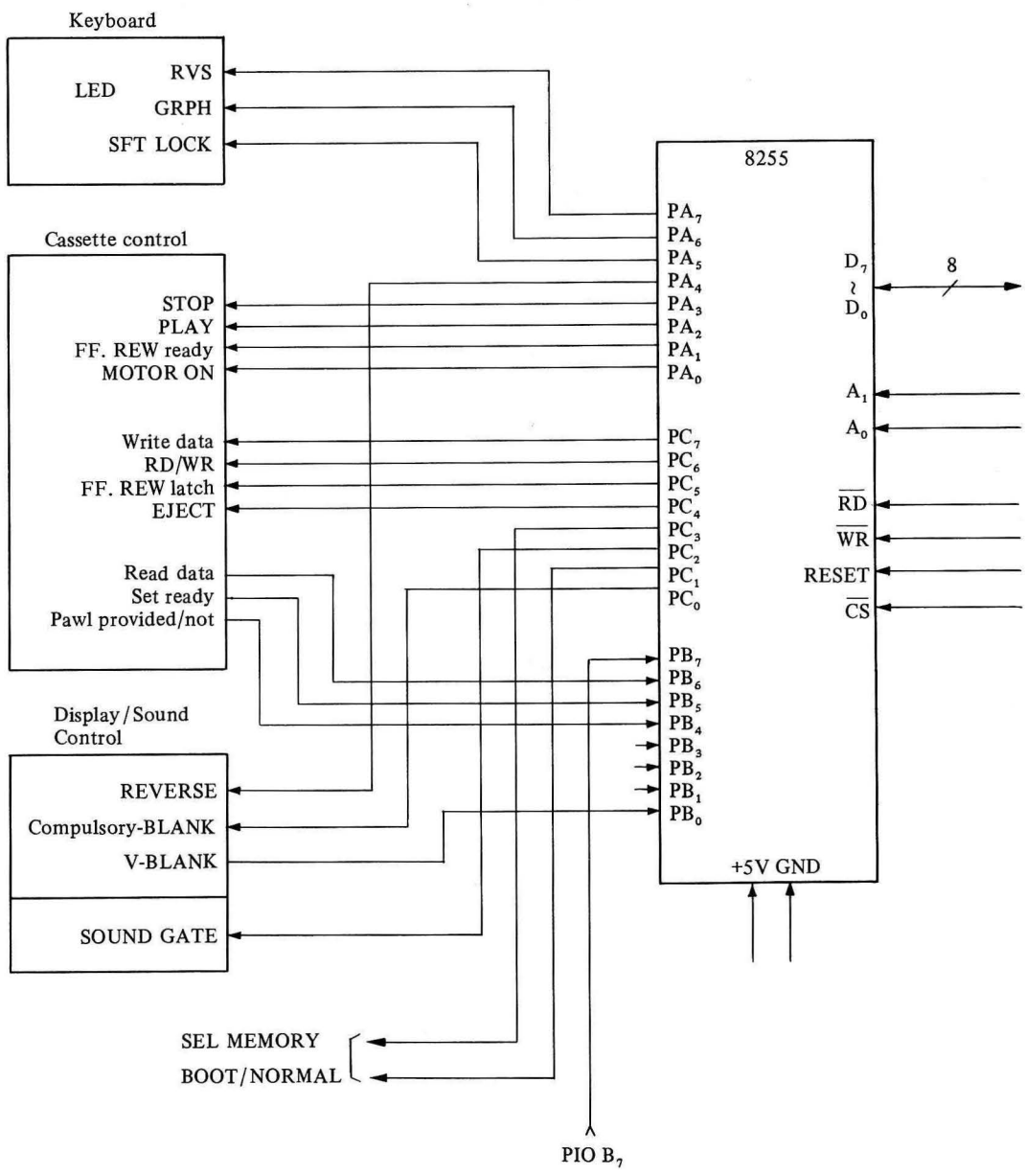


FIGURE 4.8 Signal system for the 8255 block

The control contents of each port are listed below:

Table 4.3

Port A

Port terminal	Active	Control function
PA_7	H	Lights up LED for [RVS]
PA_6	H	Lights up LED for [GRPH]
PA_5	H	Lights up LED for [SFT LOCK]
PA_4	L	Reverses B/W of entire display screen.
PA_3	H	Stops cassette operation.
PA_2	H	Plays cassette.
PA_1	H	Makes ready for FF state (makes ready for REW with L).
PA_0	H	Reel motor ON.

Port B

Port terminal	Active	Control function
PB_7	H	Detects break key while playing cassette.
PB_6		Cassette reading data.
PB_5	L	Cassette being set.
PB_4	L	Applies pawl for prohibiting writing of cassette tape.
PB_3		Reserve input ports.
PB_2		
PB_1		
PB_0	H	Blanking period of display.

Port C

Port terminal	Active	Control function
PC_7		Data to be written into cassette.
PC_6	H	Head amp setting to READ state (WRITE with L).
PC_5	H	Latches ready state for FF and REW.
PC_4	L	Eject operation.
PC_3	L	IPL starts.
PC_2		Source pulse output for generating sound.
PC_1	H	Sets memory in normal state, starting $0000.
PC_0	H	Forces display to be blank.

4.3.2 Signal system for the 8253 block

The 8253 (programmable interval timer) works as built-in clock with its counters #0, #1 and #2. These counters are used as mode 2: rate generators, and are all 16-bit binary counters.

Counter #0 counts input pulses of 31.25 kHz, and delivers a pulse to OUT 0 every one second; counter #1 counts its output pulses, and delivers a pulse to OUT 1 every 12 hours; counter #2 counts its output pulses, and repeats 0 and 1, thus working as AM/PM flag. See FIGURE 4.9.

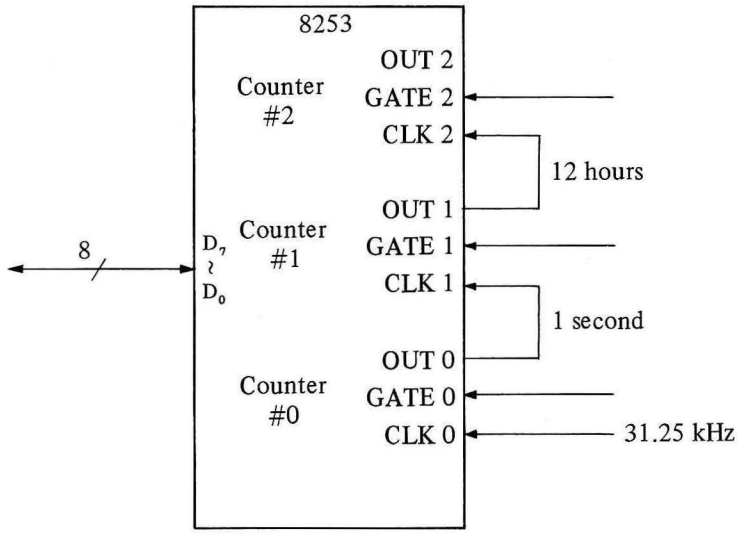


FIGURE 4.9 Signal system for the 8253 block

4.3.3 Signal system for the Z80A-PIO block

The Z80A-PIO (Parallel Input/Output interface controller) is responsible for output of strobe signal for keyboard scan, input of key data, operation to set key strobe to low level, address switching for V-RAM, and output of 40/80 character mode selection control signal.

FIGURE 4.10 summarizes the signal system for the Z80A-PIO.

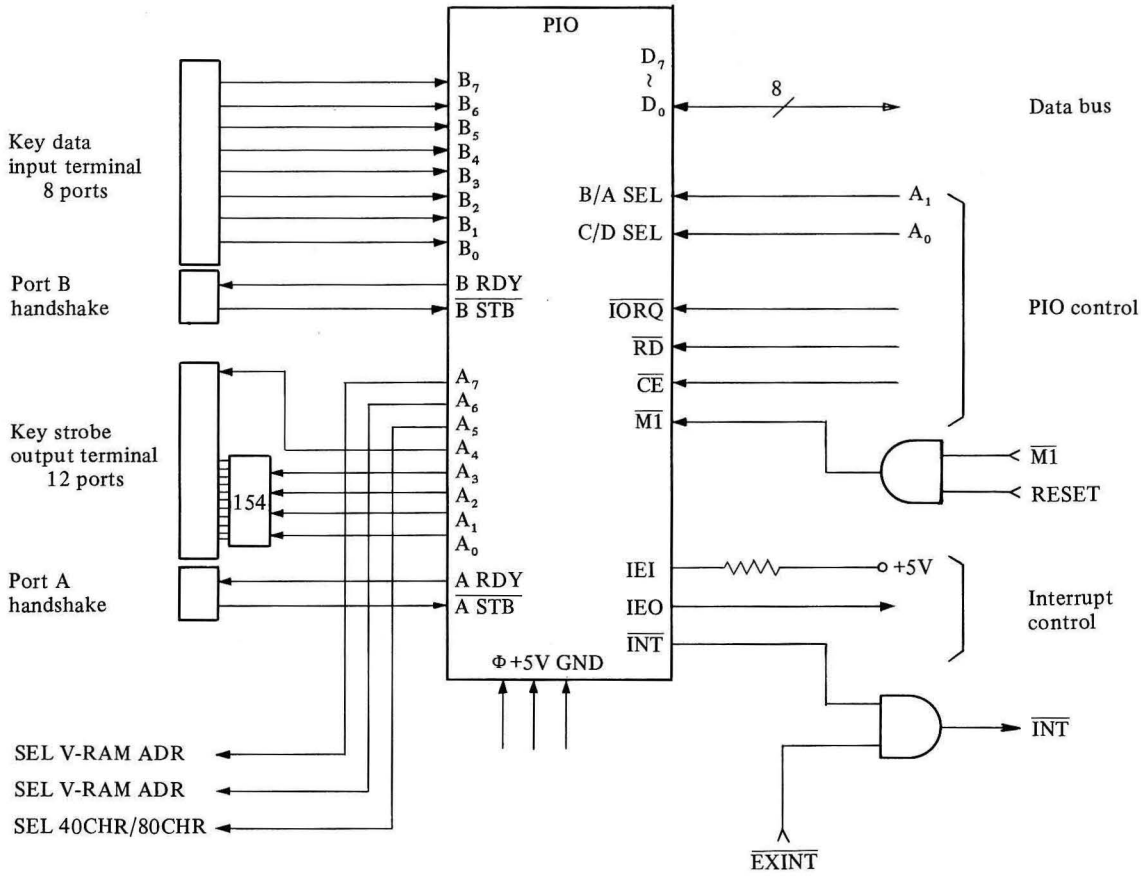


FIGURE 4.10 Signal system for the PIO

The control contents of each port are listed below:

Port A

Port terminal	Active	Control function
A_7	H	Switches addresses $D000-$FFFF to V-RAM.
A_6	H	Set addresses of V-RAM apparently to $50000-$7FFF.
A_5	H	Changes screen to 80-character mode (L: 40-character mode).
A_4	L	Turn all key strobe signals to L.
A_3		Outputs of strobe signals for keyboard scan.
A_2		
A_1		
A_0		

Port B

Port terminal	Active	Control function
B_7		Data inputs for keyboard scan.
B_6		
B_5		
B_4		
B_3		
B_2		
B_1		
B_0		

The relation between the strobe signals and bit data in keyboard scan is shown in Table 4.5.

Strobe signals are delivered to four terminals (A_3, A_2, A_1, A_0), and are fed into the demultiplexer 154, then delivered to 12 terminals of strobe inputs of keyboard. Keys are discriminated by strobe signals and key data.

For instance, when the strobe is '6H' and key data 'FFH', then it is found that key 'S' is being depressed.

Table 4.5 Key scanning strobe signals and bit data

E	MODE	0	1	2	3	4	5	6	7	STROB
0	1	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	0
		F_9	F_{10}	8	9	00	▪	+	−	1
1		0	1	2	3	4	5	6	7	2
		TAB	(SP)	CR	↑	↓	←	→	BREAK	3
2	3	/ ← →	A a	B b ♣	C c ♠	D d	E e	F f	G g	4
		H h	I i	J j	K k	L l	M m £	N n ○	O o	5
3		P p	Q q	R r	S s	T t	U u	V v ◆	W w	6
		X x ♠	Y y	Z z	~ ∧	\ | \	? ↑ ↓	. > π	, < ¥	7
4	2	0 −	1 !	2 "	3 #	4 $	5 %	6 &	7 '	8
		8 (	9)	: *	; +	_ =	@ \	[{		9
5		] }		HOME CLR	DEL INST					A
	SPCL	GRPH	SFT LOCK	SHIFT	RVS					B

■ **For key interruption**

In the MZ-80B, as stated above, key interrupt can be received by the PIO control. Illustrated below are examples of setting of simple key interrupt and response.

Try to program the machine so that the interrupt process routine $5080 is called the moment the [BREAK] key is pressed. In this programming the address table of this interrupt process routine is set to $3370 and the vector interrupt of interrupt mode 2 is used. Also, the PIO is set to mode 3 and no handshake bus is used.

As shown in Table 4.5, the [BREAK] key is, in key scanning, detected by strobe signal 3H and bit data 7 (7FH). Therefore, the strobe signal must be set at 3H, and the interrupt mask must be programmed in 7FH. It is also necessary to set the interrupt vector lower 8-bit (LSB being '0') data 70H.

The setting codes used so far are as follows:

LD A, 33H LD I, A	Setting of vector register
IM 2	Setting of interrupt mode 2
LD HL, 5080H LD (3370H), HL	Setting the address of interrupt process routine in the interrupt address table
LD A, 70H OUT (EBH), A	Setting of interrupt vector (lower 8 bits)
LD A, CFH OUT (EBH), A	Setting port B to mode 3
LD A, FFH OUT (EBH), A	Handling all port B as input
LD A, 97H OUT (EBH), A	Setting of interrupt control words, or enabling interrupt
LD A, 7FH OUT (EBH), A	Setting of mask words; bit data 7 (MSB being 'L') is masked by writing 7FH
IN A, (E8H) AND E0H OR 13H OUT (E8H), A	Of the data delivered to port A, key strobe is set to 3H. The strobe gate is opened with A4 set at 'H'

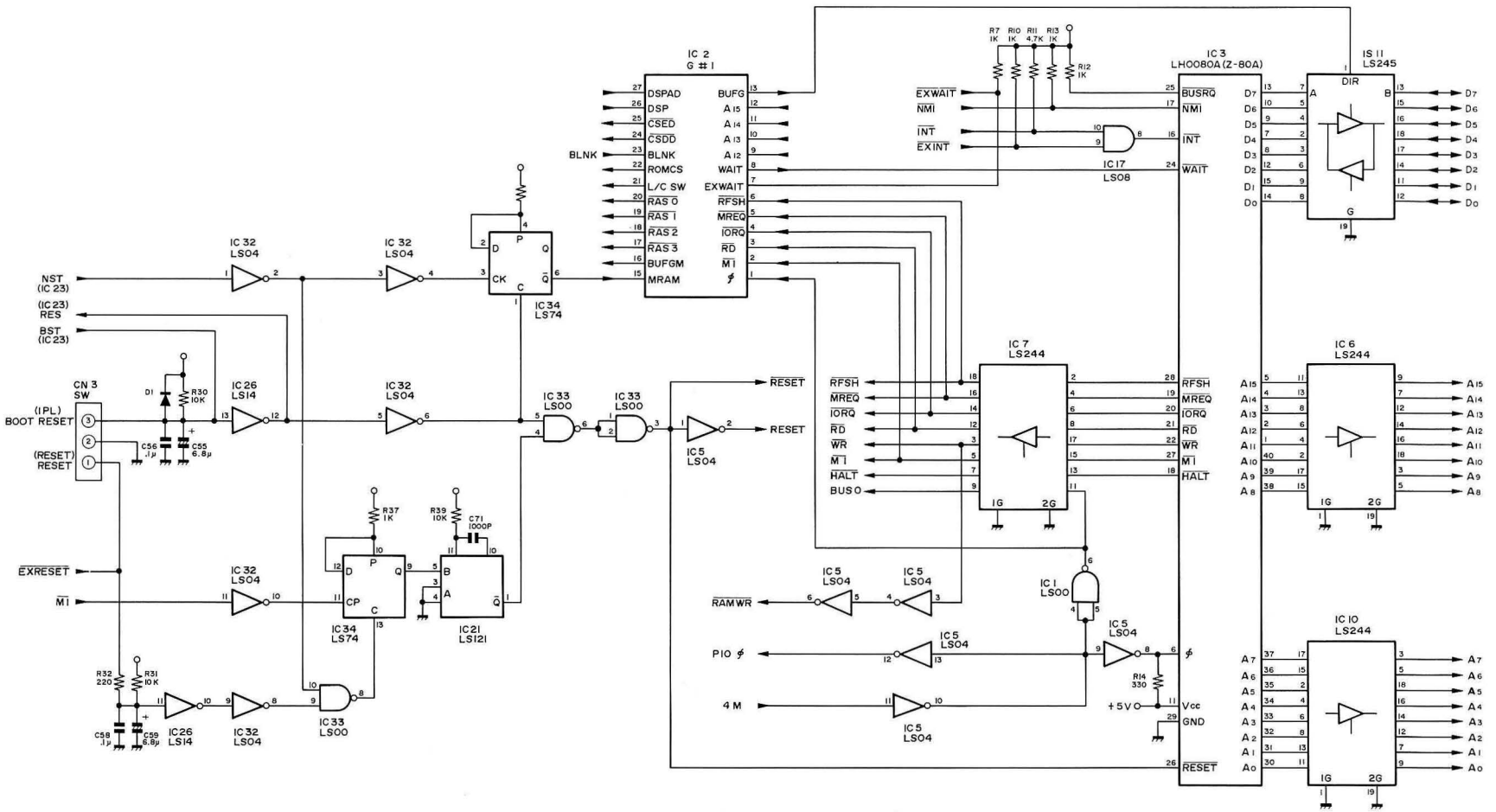
After the mode setting above, when the [BREAK] key is pushed at an arbitrary point, the $3370 is referenced by vector interrupt, and the interrupt process routine $5080 is called.

The instruction to return, after terminating the interrupt process routine, to the program being executed before interruption is RETI.

4.4 The MZ-80B circuit diagrams

This section includes all MZ-80B circuit diagrams for reference. These diagrams are arranged as follows;

(1) CPU board, block 1 : CPU signal system
(2) CPU board, block 2
(3) CPU board, block 3 : 8255 and PIO signal system
(4) CPU board, block 4 : RAM signal system
(5) CPU board, block 5
(6) CRT display control
(7) Cassette tape deck control
(8) Power supply
(9) Graphic Memory 1 card (optional)
(10) Expansion I/O port (optional)
(11) Graphic Memory 2 card (optional)



MZ-80B (1)

FIGURE 4.11 CPU board, block 1 : CPU signal system

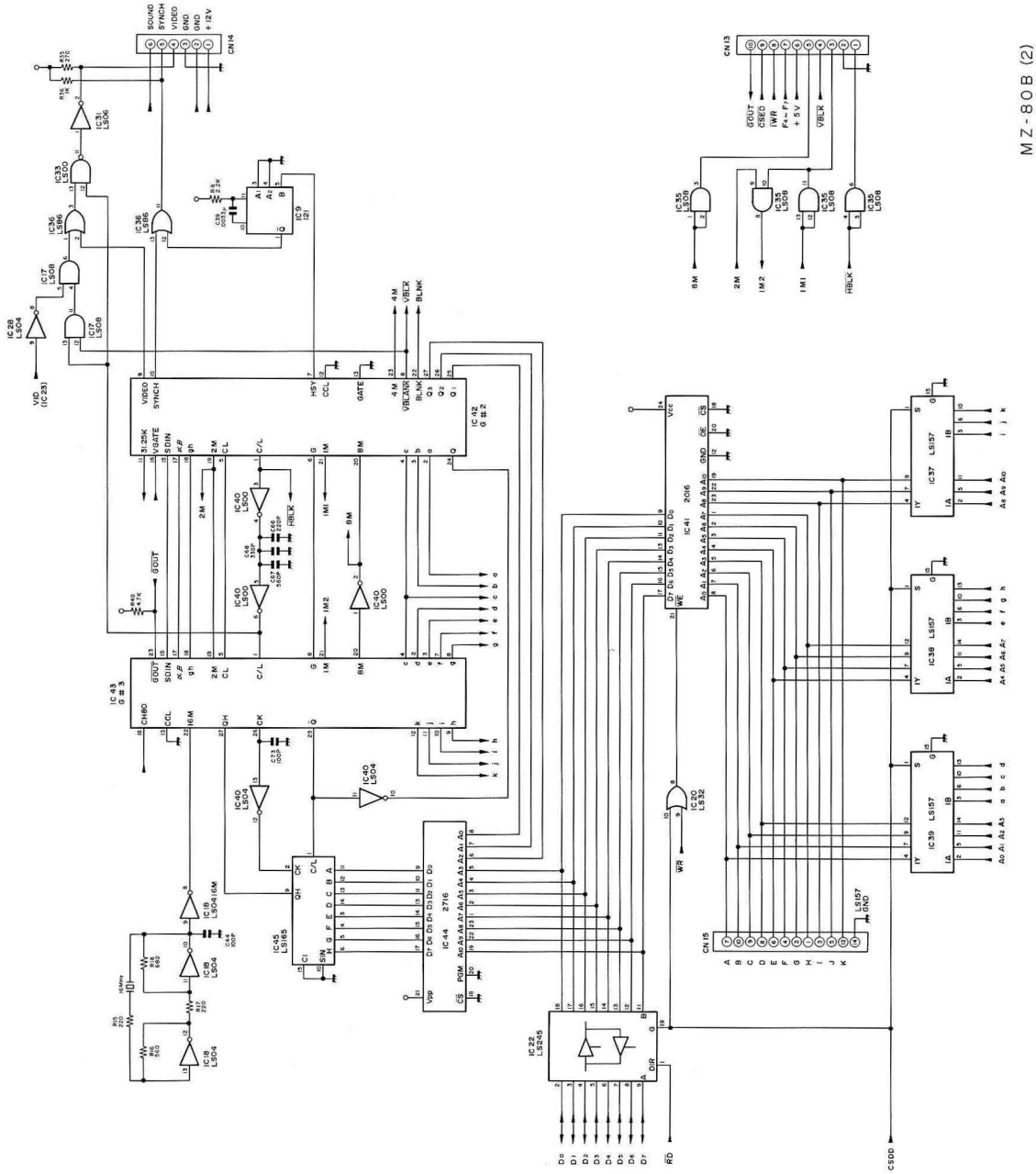


FIGURE 4.12 CPU board, block 2

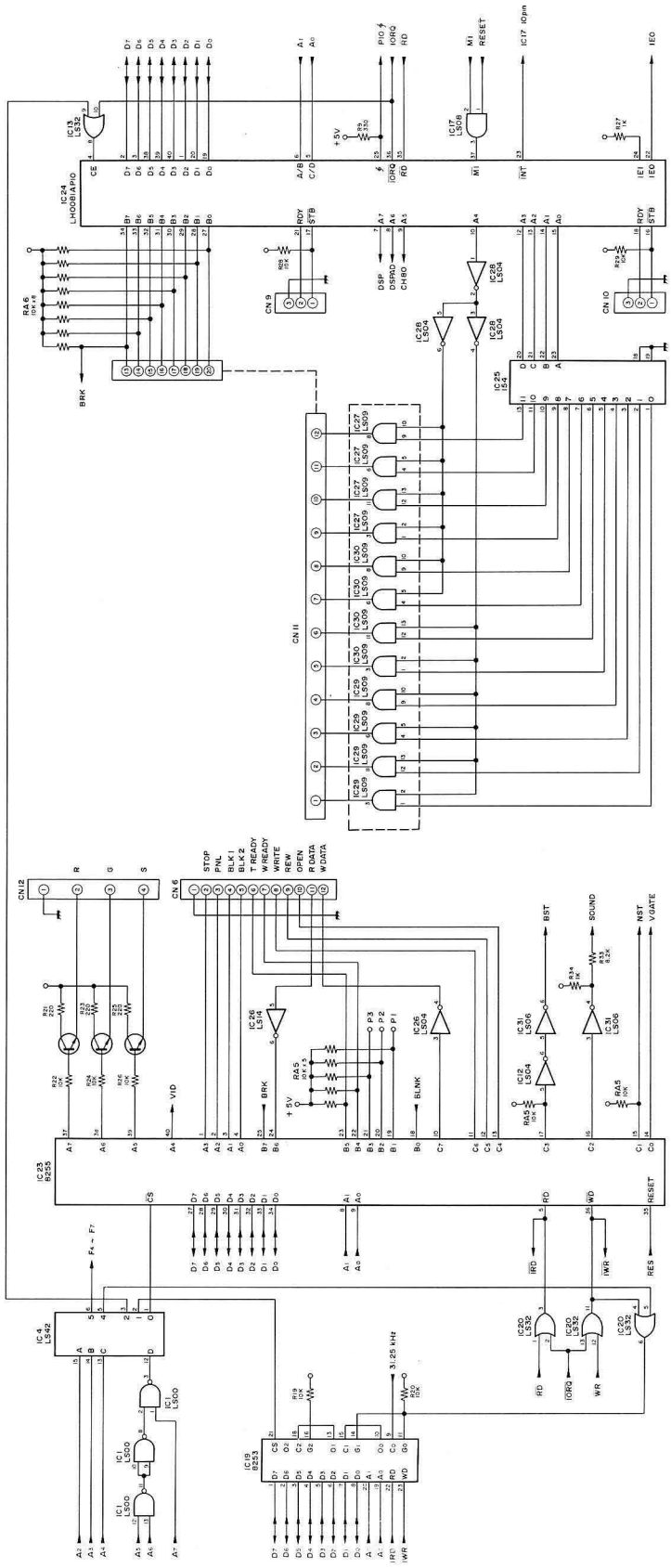


FIGURE 4.13 CPU board, block 3 : 8255 and PIO signal system

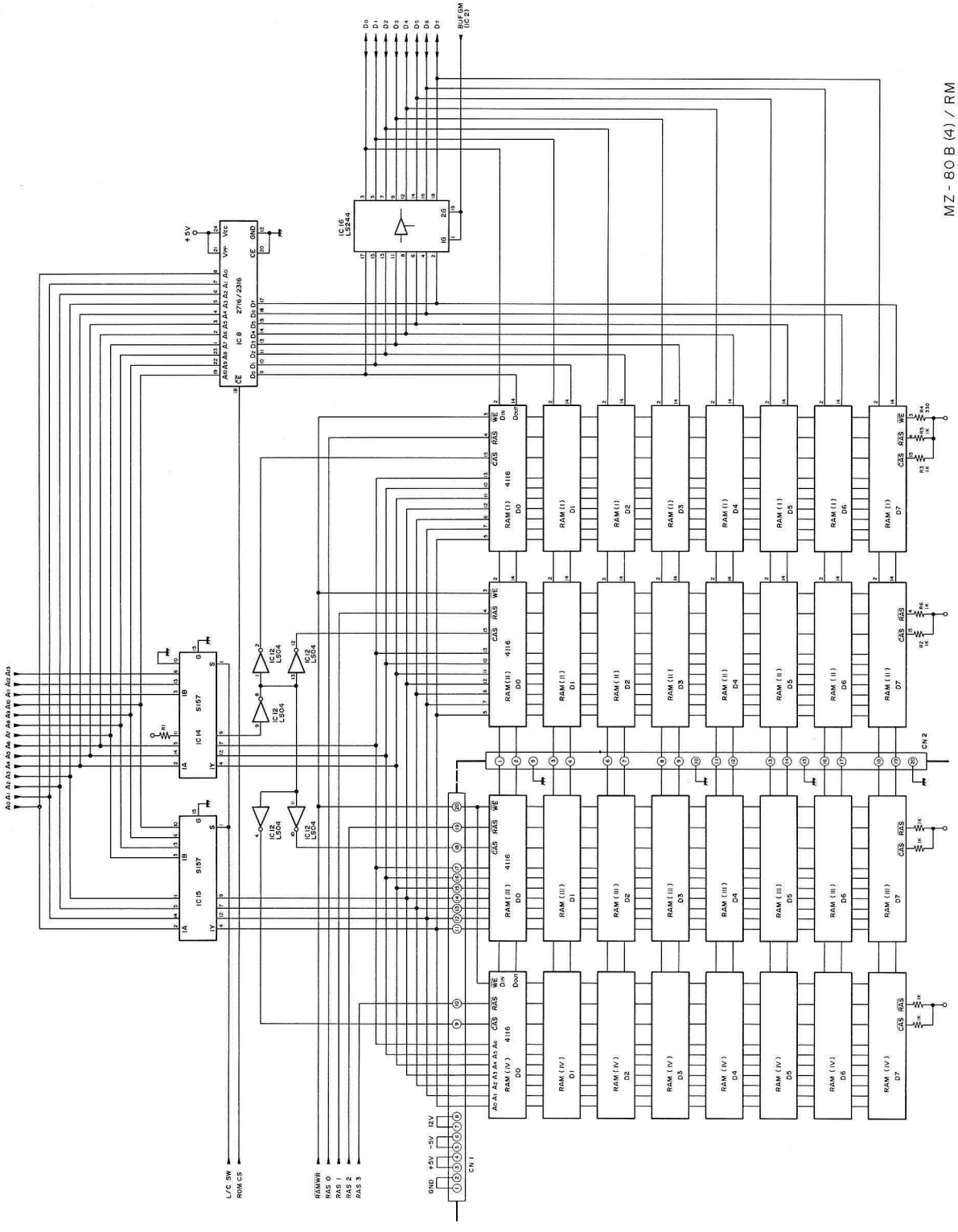


FIGURE 4.14 CPU board, block 4 : RAM signal system

CN5 ⟶ MZ-80GM

CN4, 5 40P			
1	A15	2	A14
3	A13	4	A12
5	A11	6	A10
7	A9	8	A8
9	GND	10	A7
11	A6	12	A5
13	A4	14	A3
15	A2	16	A1
17	AØ	18	GND
19	D7	20	D6
21	D5	22	D4
23	D3	24	D2
25	D1	26	DØ
27	GND	28	$\overline{\text{NMI}}$
29	$\overline{\text{EX WAIT}}$	30	$\overline{\text{EX INT}}$
31	$\overline{\text{EX RESET}}$	32	RESET
33	IEO	34	$\overline{\text{HALT}}$
35	$\overline{\text{MREQ}}$	36	$\overline{\text{IOREQ}}$
37	$\overline{\text{RD}}$	38	$\overline{\text{WR}}$
39	$\overline{\text{M1}}$	40	BUSØ

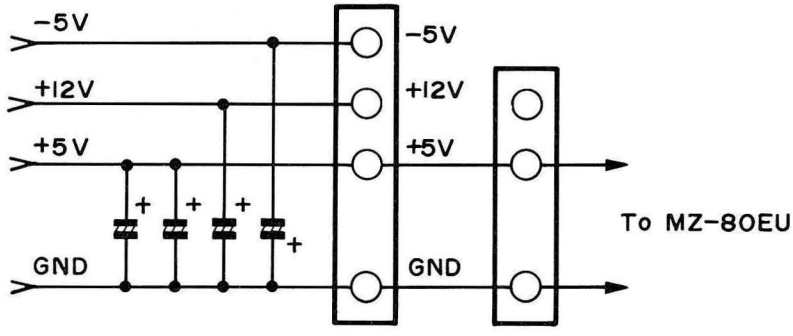


FIGURE 4.15 CPU board, block 5

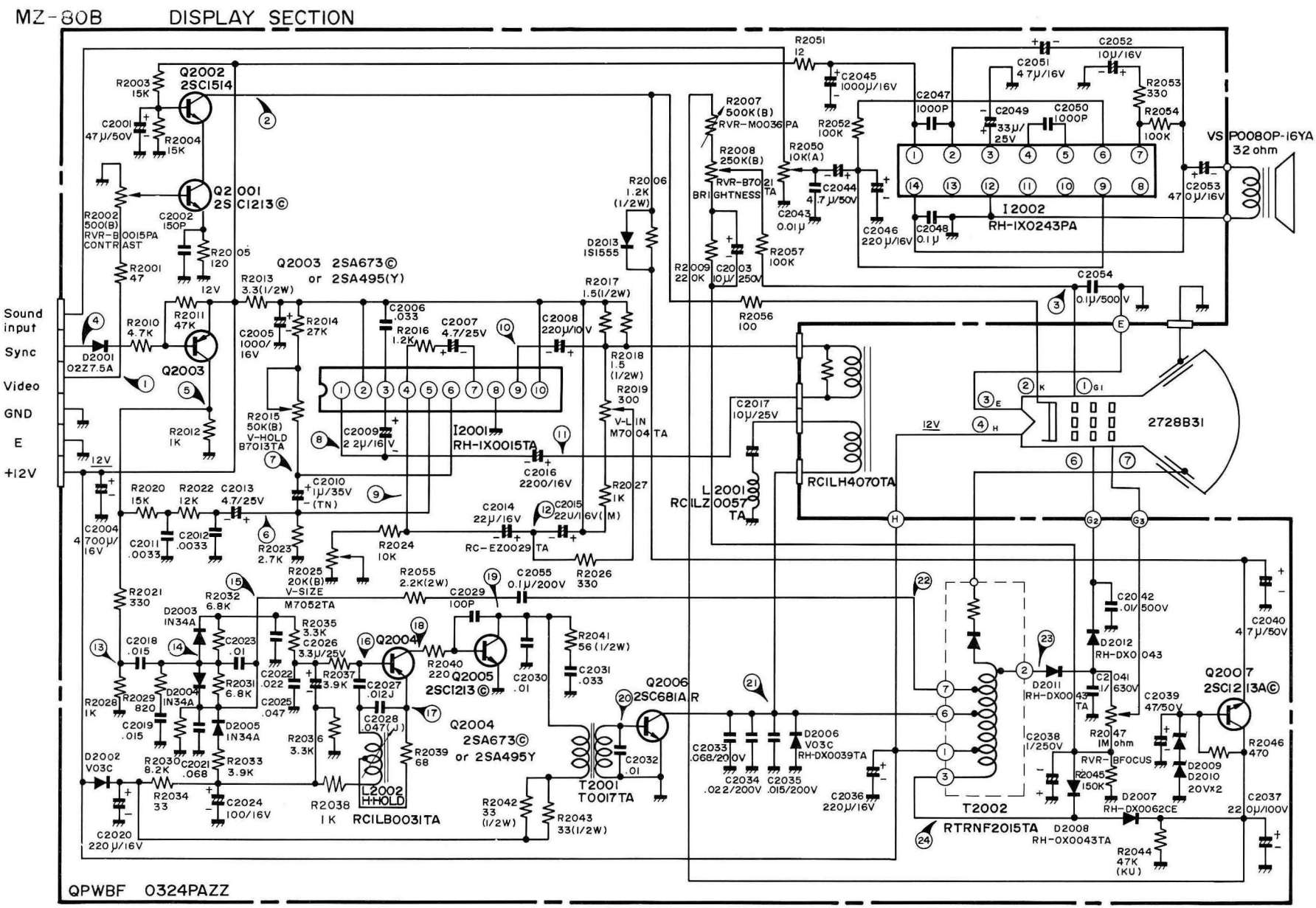


FIGURE 4.16 CRT display control

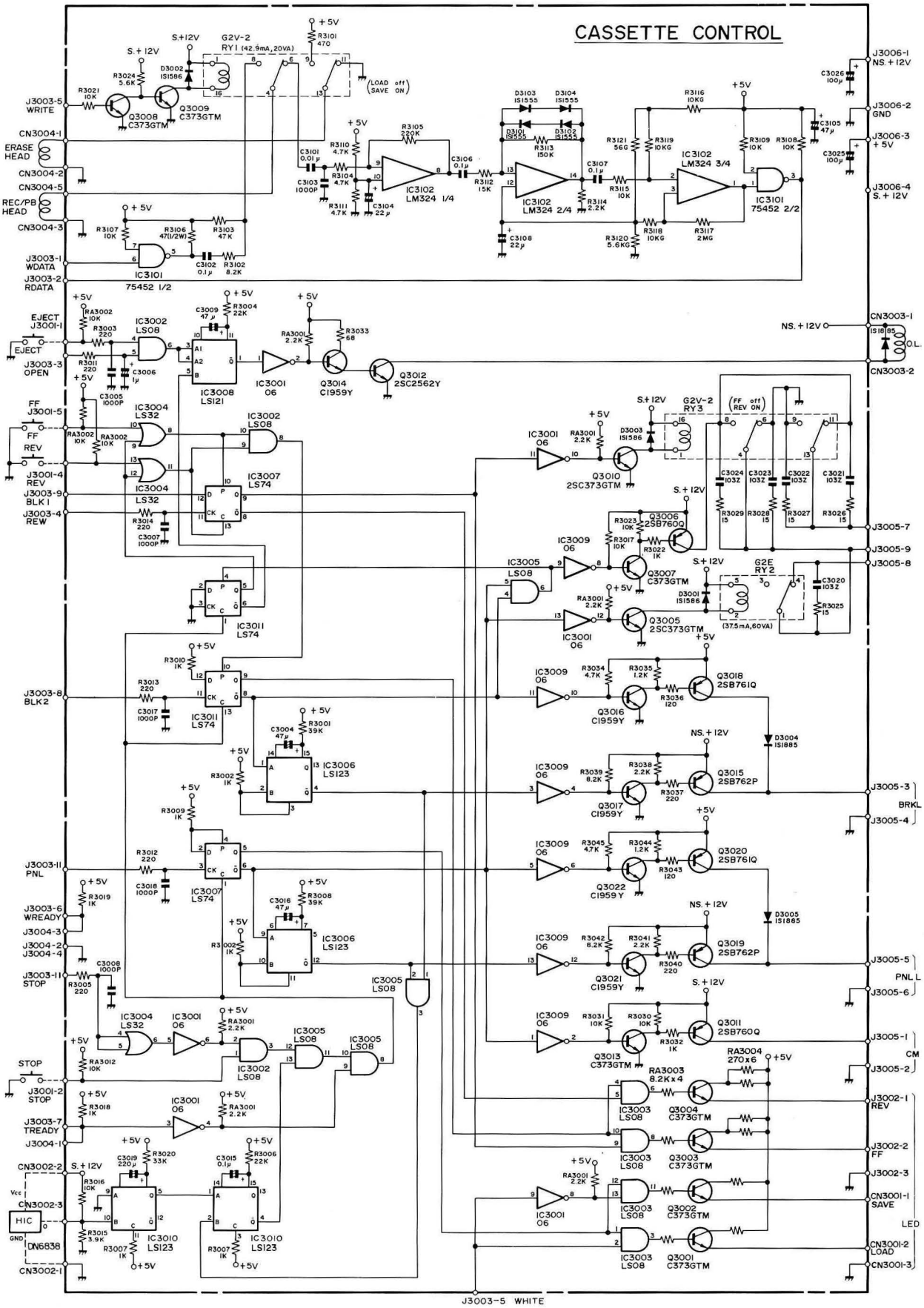


FIGURE 4.17 Cassette tape deck control

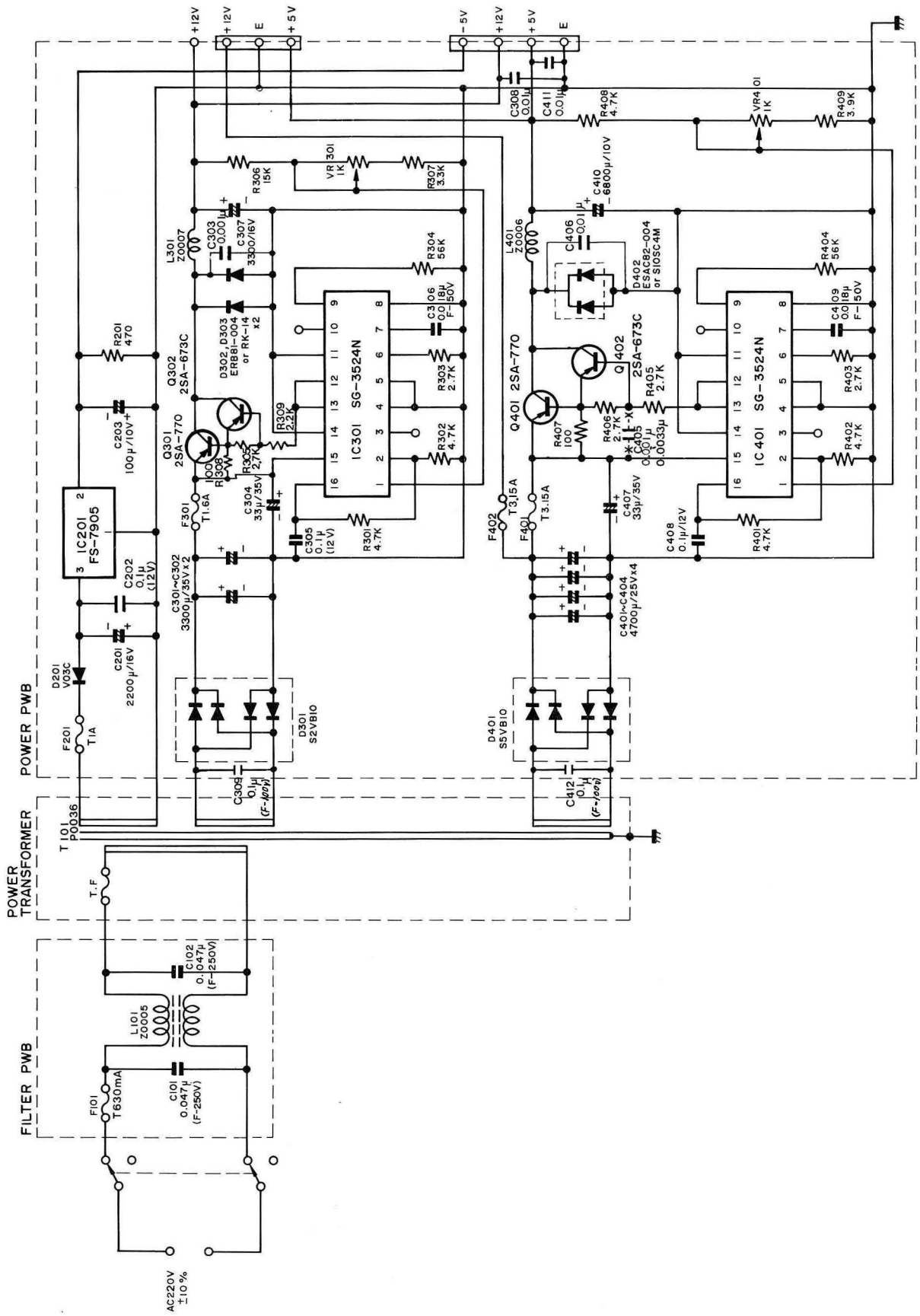


FIGURE 4.18 Power supply

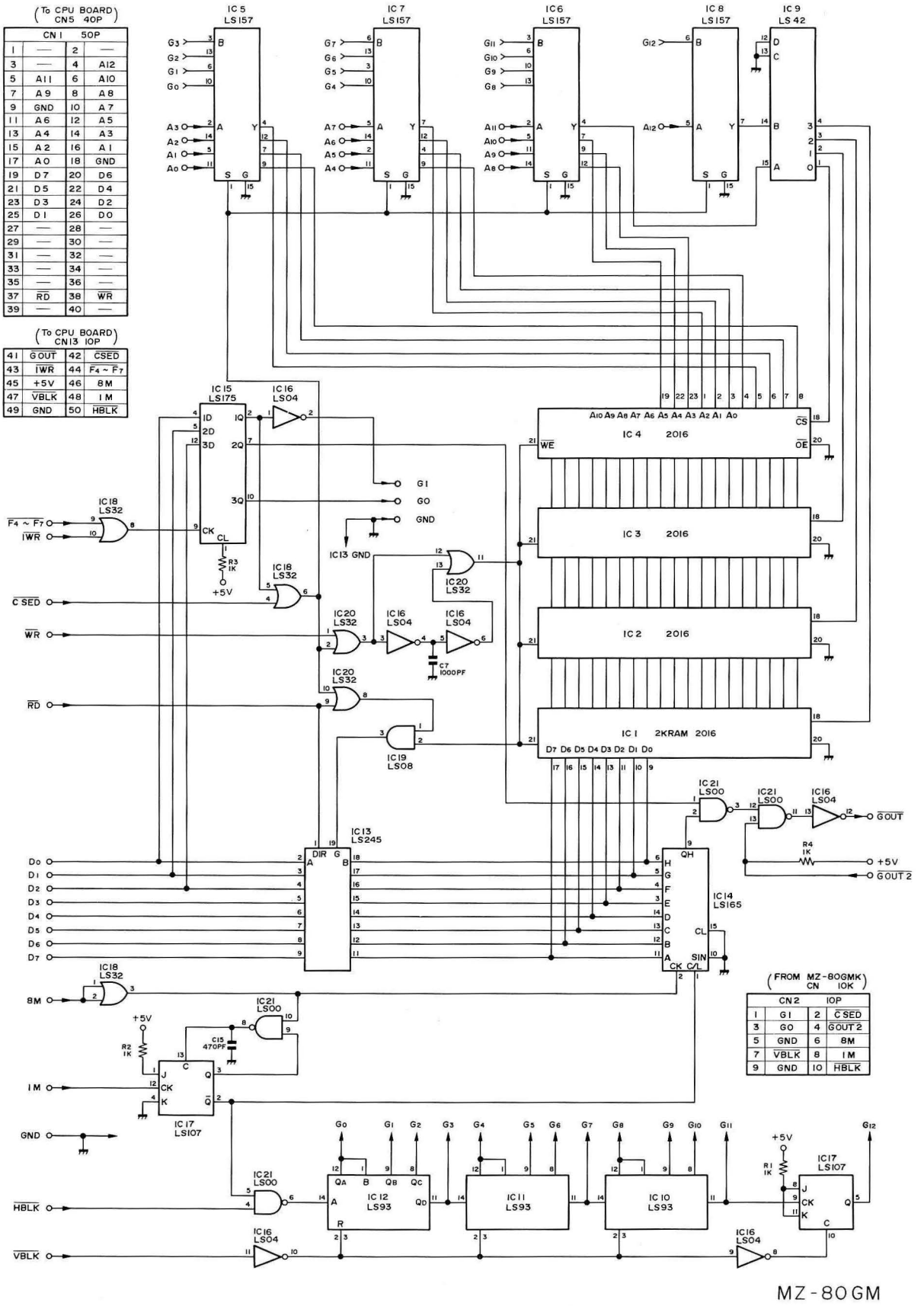
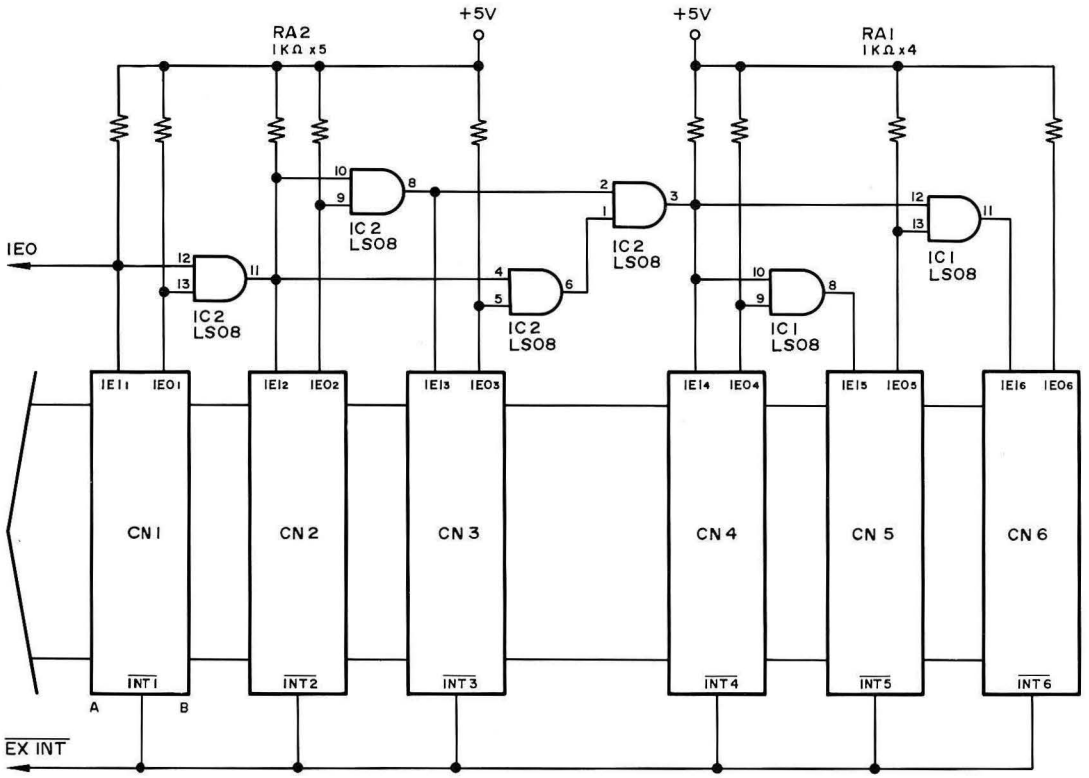


FIGURE 4.19 Graphic Memory 1 card (optional)

(To CPU BOARD)
CN4 40P

CN7, 8		40P	
1	A15	2	A14
3	A13	4	A12
5	A11	6	A10
7	A9	8	A8
9	GND	10	A7
11	A6	12	A5
13	A4	14	A3
15	A2	16	A1
17	A0	18	GND
19	D7	20	D6
21	D5	22	D4
23	D3	24	D2
25	D1	26	D0
27	GND	28	$\overline{NMI}$
29	$\overline{EX\ WAIT}$	30	$\overline{EX\ INT}$
31	$\overline{EX\ RESET}$	32	RESET
33	IEO	34	$\overline{HALT}$
35	$\overline{MREQ}$	36	$\overline{IOREQ}$
37	$\overline{RD}$	38	$\overline{WR}$
39	$\overline{M1}$	40	BUS Ø



CN1 ~ CN6

A		B
+5V	1	+5V
D2	2	D3
D1	3	D4
D0	4	D5
GND	5	D6
A15	6	D7
A14	7	BUS Ø
A13	8	$\overline{M1}$
A12	9	$\overline{WR}$
A11	10	$\overline{RD}$
A10	11	$\overline{IOREQ}$
A9	12	$\overline{MREQ}$
A8	13	GND
A7	14	$\overline{HALT}$
A6	15	IEI
A5	16	IEO
A4	17	RESET
A3	18	$\overline{EX\ RESET}$
A2	19	$\overline{EX\ INT}$
A1	20	$\overline{EX\ WAIT}$
A0	21	$\overline{NMI}$
GND	22	GND

A : PARTS SIDE

FIGURE 4.20 Expansion I/O port (optional)

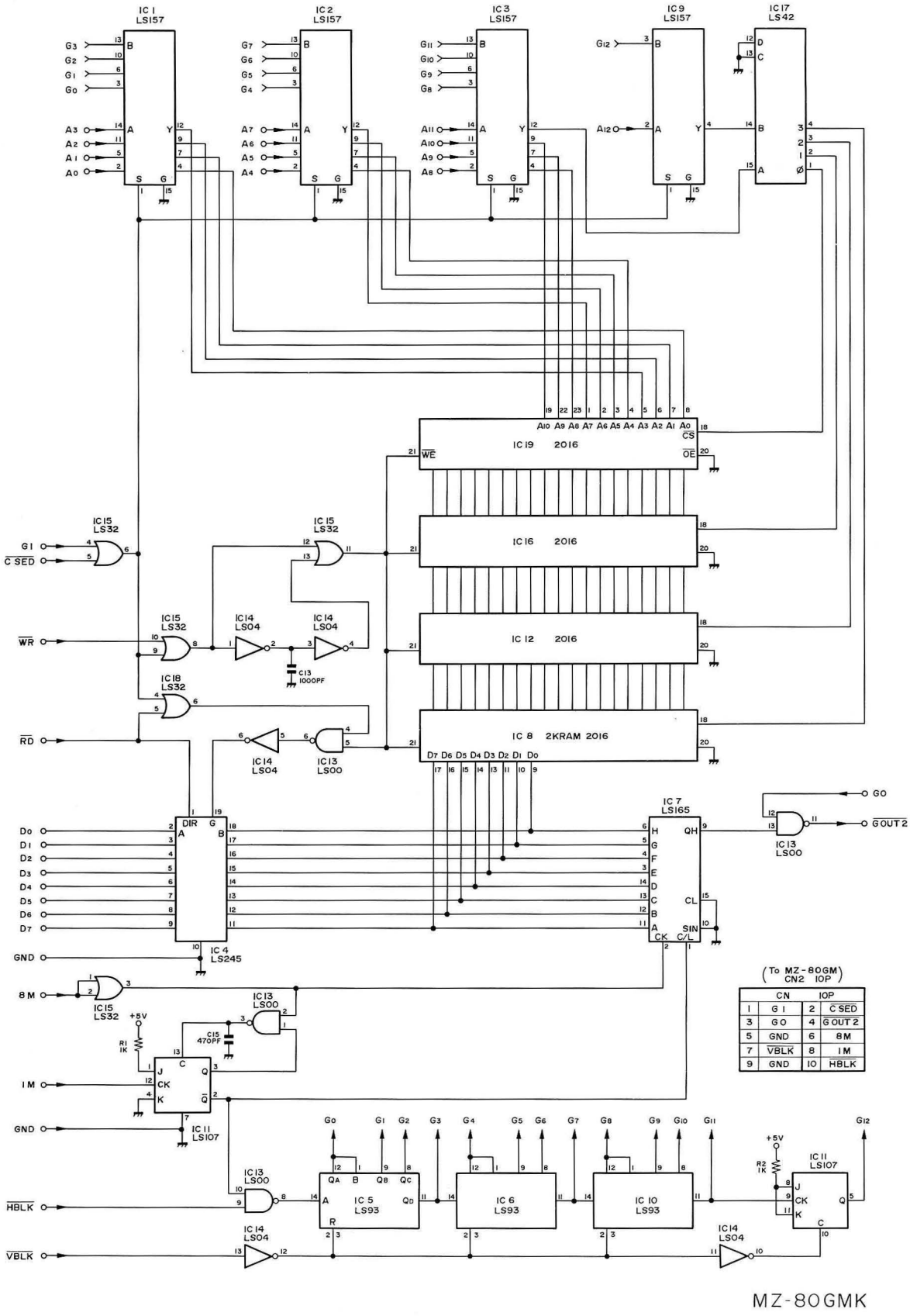


FIGURE 4.21 Graphic Memory 2 card (optional)

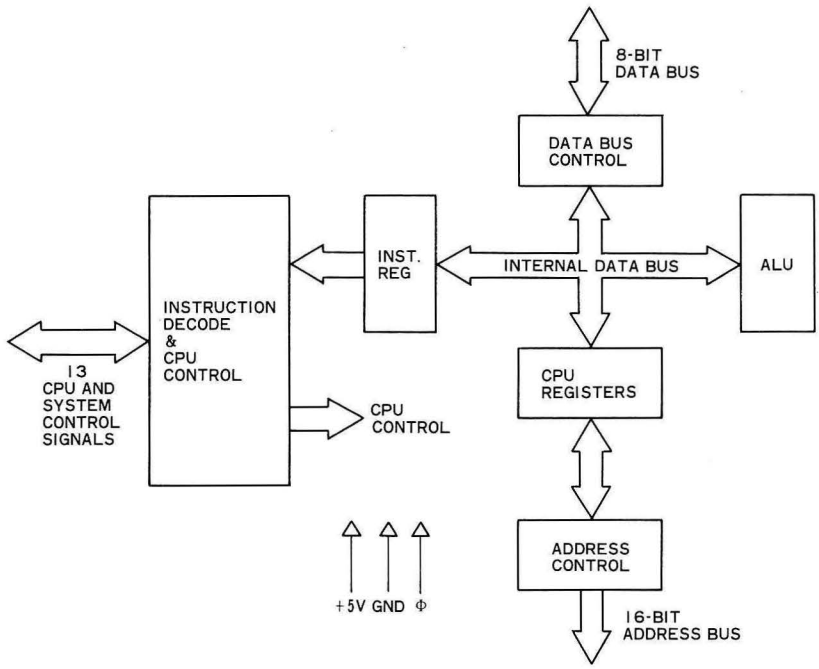
APPENDIX

The Appendix includes technical data on the Z80A-CPU and Z80A-PIO for reference. This information will be helpful to you in expanding the system.

A.1 Technical Data of Z80A-CPU

1.0 ARCHITECTURE

A block diagram of the internal architecture of the Z-80A CPU is shown in Figure 1.0-1. The diagram shows all of the major elements in the CPU and it should be referred to throughout the following description.



Z-80A CPU BLOCK DIAGRAM
FIGURE 1.0-1

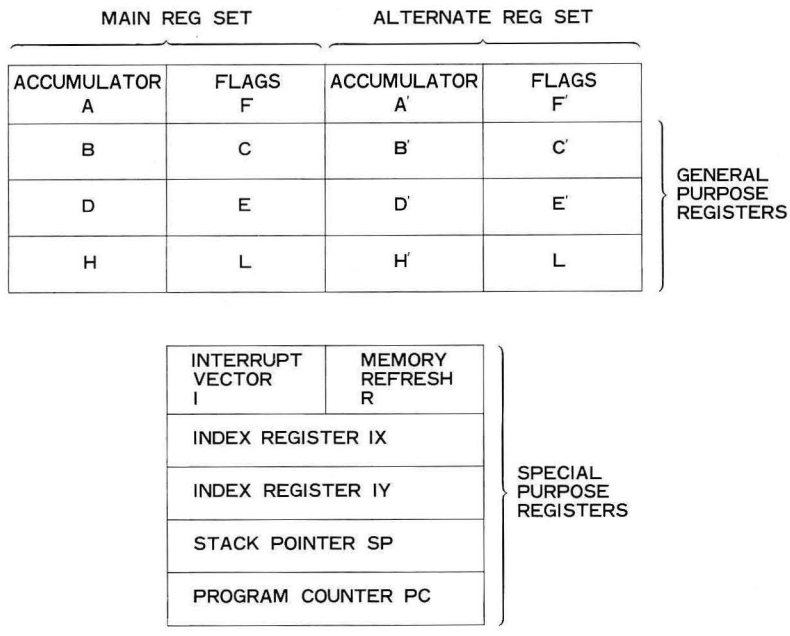
1.1 CPU REGISTERS

The Z-80A CPU contains 208 bits of R/W memory that are accessible to the programmer. Figure 1.0-2 illustrates how this memory is configured into eighteen 8-bit registers and four 16-bit registers. All Z-80A registers are implemented using static RAM. The registers include two sets of six general purpose registers that may be used individually as 8-bit registers or in pairs as 16-bit registers. There are also two sets of accumulator and flag registers.

Special Purpose Registers

1. **Program counter (PC).** The program counter holds the 16-bit address of the current instruction being fetched from memory. The PC is automatically incremented after its contents have been transferred to the address lines. When a program jump occurs the new value is automatically placed in the PC, overriding the incrementer.

2. **Stack Pointer (SP).** The stack pointer holds the 16-bit address of the current top of a stack located anywhere in external system RAM memory. The external stack memory is organized as a last-in first-out (LIFO) file. Data can be pushed onto the stack from specific CPU registers or popped off of the stack into specific CPU registers through the execution of PUSH and POP instructions. The data popped from the stack is always the last data pushed onto it. The stack allows simple implementation of multiple level interrupts, unlimited subroutine nesting and simplification of many types of data manipulation.



Z-80A CPU REGISTER CONFIGURATION
FIGURE 1.0-2

3. **Two Index Registers (IX & IY).** The two independent index registers hold a 16-bit base address that is used in indexed addressing modes. In this mode, an index register is used as a base to point to a region in memory from which data is to be stored or retrieved. An additional byte is included in indexed instructions to specify a displacement from this base. This displacement is specified as a two's complement signed integer. This mode of addressing greatly simplifies many types of programs, especially where tables of data are used.

4. **Interrupt Page Address Register (I).** The Z-80A CPU can be operated in a mode where an indirect call to any memory location can be achieved in response to an interrupt. The I Register is used for this purpose to store the high order 8-bits of the indirect address while the interrupting device provides the lower 8-bits of the address. This feature allows interrupt routines to be dynamically located anywhere in memory with absolute minimal access time to the routine.

5. **Memory Refresh Register (R).** The Z-80A CPU contains a memory refresh counter to enable dynamic memories to be used with the same ease as static memories. Seven bits of this 8-bit register are automatically incremented after each instruction fetch. The eighth bit will remain as programmed as the result of an LD R, A instruction. The data in the refresh counter is sent out on the lower portion of the address but along with a refresh control signal while the CPU is decoding and executing the fetched instruction. This mode of refresh is totally transparent to the programmer and does not slow down the CPU operation. The programmer can load the R register for testing purposes, but this register is normally not used by the programmer. During refresh, the contents of the I register are placed on the upper 8 bits of the address bus.

Accumulator and Flag Registers

The CPU includes two independent 8-bit accumulators and associated 8-bit flag registers. The accumulator holds the results of 8-bit arithmetic or logical operations while the flag register indicates specific conditions for 8 or 16-bit operations, such as indicating whether or not the result of an operation is equal to zero. The programmer selects the accumulator and flag pair that he wishes to work with a single exchange instruction so that he may easily work with either pair.

General Purpose Registers

There are two matched sets of general purpose registers, each set containing six 8-bit registers that may be used individually as 8-bit registers or as 16-bit register pairs by the programmer. One set is called BC, DE and HL while the complementary set is called BC′, DE′ and HL′. At any one time the programmer can select either set of registers to work with through a single exchange command for the entire set. In systems where fast interrupt response is required, one set of general purpose registers and an accumulator/flag register may be reserved for handling this very fast routine. Only a simple exchange commands need be executed to go between the routines. This greatly reduces interrupt service time by eliminating the requirement for saving and retrieving register contents in the external stack during interrupt or subroutine processing. These general purpose registers are used for a wide range of applications by the programmer. They also simplify programming, especially in ROM based systems where little external read/write memory is available.

1.2 ARITHMETIC AND LOGIC UNIT (ALU)

The 8-bit arithmetic and logical instructions of the CPU are executed in the ALU. Internally the ALU communicates with the registers and the external data bus on the internal data bus. The type of functions performed by the ALU include:

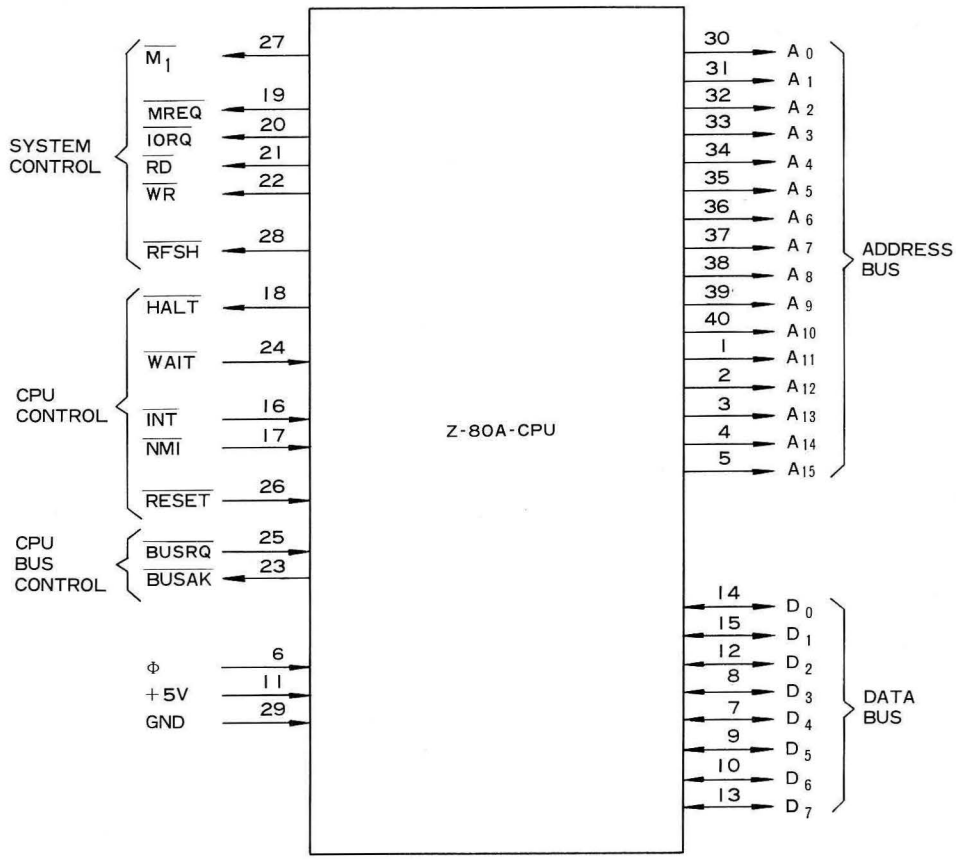
Add	Left or right shifts or rotates (arithmetic and logical)
Subtract	Increment
Logical AND	Decrement
Logical OR	Set bit
Logical Exclusive OR	Reset bit
Compare	Test bit

1.3 INSTRUCTION REGISTER AND CPU CONTROL

As each instruction is fetched from memory, it is placed in the instruction register and decoded. The control section performs this function and then generates and supplies all of the control signals necessary to read or write data from or to the registers, control the ALU and provide all required external control signals.

2.0 PIN DESCRIPTION

The Z-80A CPU is packaged in an industry standard 40 pin Dual In-Line Package. The I/O pins are shown in Figure 2.0-1 and the function of each is described below.



Z-80A PIN CONFIGURATION
FIGURE 2.0-1

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 exchanger 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, $\overline{M_1}$ is generated as each op code byte is fetched. These two byte op-codes always begin with CBH, DDH, EDH or FDH. $\overline{M_1}$ 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.

$\overline{IORQ}$ (Input/Output Request)

Tri-state output, active low. The $\overline{IORQ}$ signal indicates that the lower half of the address bus holds a valid I/O address for a I/O read or write operation. An IORQ signal is also generated with an $\overline{M_1}$ signal when an interrupt is being acknowledged to indicate that an interrupt response vector can be placed on the data bus. Interrupt Acknowledge operations occur during M_1 time while I/O operations never occur during M_1 time.

$\overline{RD}$ (Memory Read)

Tri-state output, active low. $\overline{RD}$ indicates that the CPU wants to read data from memory or an I/O device. The addressed I/O device or memory should use this signal to gate data onto the CPU data bus.

$\overline{WR}$ (Memory Write)

Tri-state output, active low. $\overline{WR}$ indicates that the CPU data bus holds valid data to be stored in the addressed memory or I/O device.

$\overline{RFSH}$ (Refresh)

Output, active low. $\overline{RFSH}$ indicates that the lower 7 bits of the address bus contain a refresh address for dynamic memories and the current $\overline{MREQ}$ signal should be used to do a refresh read to all dynamic memories.

$\overline{HALT}$ (Halt state)

Output, active low. $\overline{HALT}$ indicates that the CPU has executed a HALT software instruction and is awaiting either a non maskable or a maskable interrupt (with the mask enabled) before operation can resume. While halted, the CPU executes NOP's to maintain memory refresh activity.

$\overline{WAIT}$ (Wait)

Input, active low. $\overline{WAIT}$ indicates to the Z-80A CPU that the addressed memory or I/O devices are not ready for a data transfer. The CPU continues to enter wait states for as long as this signal is active. This signal allows memory or I/O devices of any speed to be synchronized to the CPU.

$\overline{INT}$ (Interrupt Request)

Input, active low. The Interrupt Request signal is generated by I/O devices. A request will be honored at the end of the current instruction if the internal software controlled interrupt enable flip-flop (IFF) is enabled and if the $\overline{BUSRQ}$ signal is not active. When the CPU accepts the interrupt, an acknowledge signal ($\overline{IORQ}$ during M_1 time) is sent out at the beginning of the next instruction cycle. The CPU can respond to an interrupt in three different modes.

$\overline{NMI}$ (Non Maskable Interrupt)

Input, negative edge triggered. The non maskable interrupt request line has a higher priority than $\overline{INT}$ and is always recognized at the end of the current instruction, independent of the status of the interrupt enable flip-flop. $\overline{NMI}$ automatically forces the Z-80A CPU to restart to location 0066_H. The program counter is automatically saved in the external stack so that the user can return to the program that was interrupted. Note that continuous WAIT cycles can prevent the current instruction from ending, and that a $\overline{BUSRQ}$ will override a $\overline{NMI}$.

$\overline{\text{RESET}}$

Input, active low, $\overline{\text{RESET}}$ forces the program counter to zero and initializes the CPU. The CPU initialization includes:

1) Disable the interrupt enable flip-flop
2) Set Register I = 00_H
3) Set Register R = 00_H
4) Set Interrupt Mode 0

During reset time, the address bus and data bus go to a high impedance state and all control output signals go to the inactive state.

$\overline{\text{BUSRQ}}$
(Bus Request)

Input, active low. The bus request signal is used to request the CPU address bus, data bus and tri-state output control signals to go to a high impedance state so that other devices can control these buses. When BUSRQ is activated, the CPU will set these buses to a high impedance state as soon as the current CPU machine cycle is terminated.

$\overline{\text{BUSAK}}$
(Bus Acknowledge)

Output, active low. Bus acknowledge is used to indicate to the requesting device that the CPU address bus, data bus and tri-state control bus signals have been set to their high impedance state and the external device can now control these signals.

Φ

Single phase TTL level clock which requires only a 330 ohm pull-up resistor to +5 volts to meet all clock requirements. (4 MHz)

3.0 TIMING

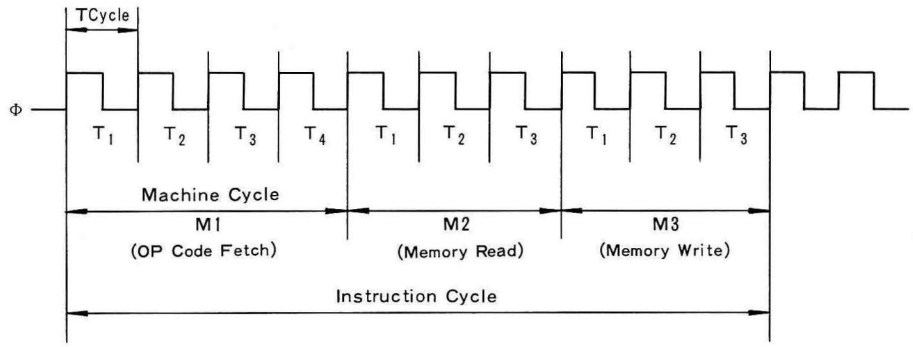
The Z-80A CPU executes instructions by stepping through a very precise set of a few basic operations. These include:

Memory read or write

I/O device read or write

Interrupt acknowledge

All instructions are merely a series of these basic operations. Each of these basic operations can take from three to six clock periods to complete or they can be lengthened to synchronize the CPU to the speed of external devices. The basic clock periods are referred to as T cycles and the basic operations are referred to as M (for machine) cycles. Figure 3.0-0 illustrates how a typical instruction will be merely a series of specific M and T cycles. Notice that this instruction consists of three machine cycles (M1, M2 and M3). The first machine cycle of any instruction is a fetch cycle which is four, five or six T cycles long (unless lengthened by the wait signal which will be fully described in the next section). The fetch cycle (M1) is used to fetch the OP code of the next instruction to be executed. Subsequent machine cycles move data between the CPU and memory or I/O devices and they may have anywhere from three to five T cycles (again they may be lengthened by wait states to synchronize the external devices to the CPU). The following paragraphs describe the timing which occurs within any of the basic machine cycles.



BASIC CPU TIMING EXAMPLE
FIGURE 3.0-0

All CPU timing can be broken down into a few very simple timing diagrams as shown in Figure 3.0-1 through 3.0-7. These diagrams show the following basic operations with and without wait states (wait states are added to synchronize the CPU to slow memory or I/O devices).

3.0-1. Instruction OP code fetch (M1 cycle)

3.0-2. Memory data read or write cycles

3.0-3. I/O read or write cycles

3.0-4. Bus Request/Acknowledge Cycle

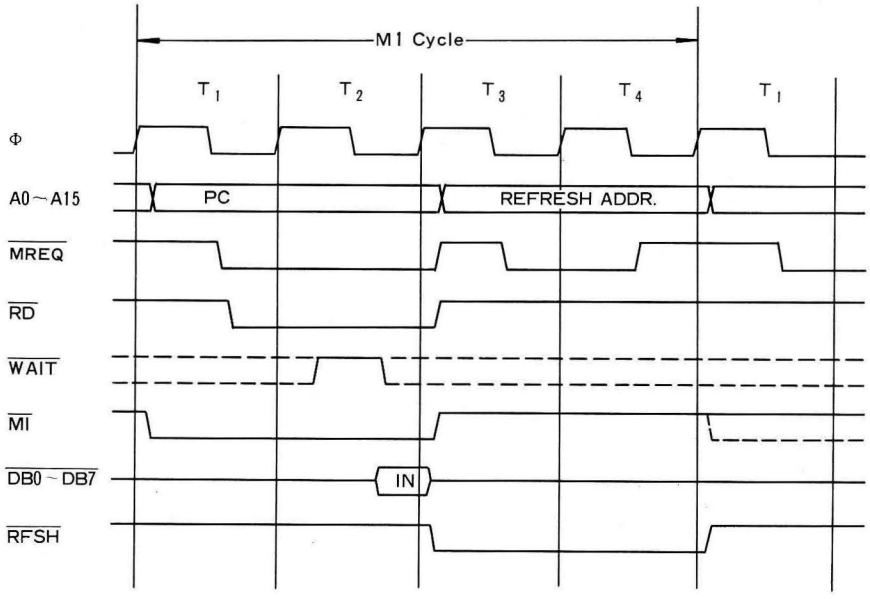
3.0-5. Interrupt Request/Acknowledge Cycle

3.0-6. Non maskable Interrupt Request/Acknowledge Cycle

3.0-7. Exit from a HALT instruction

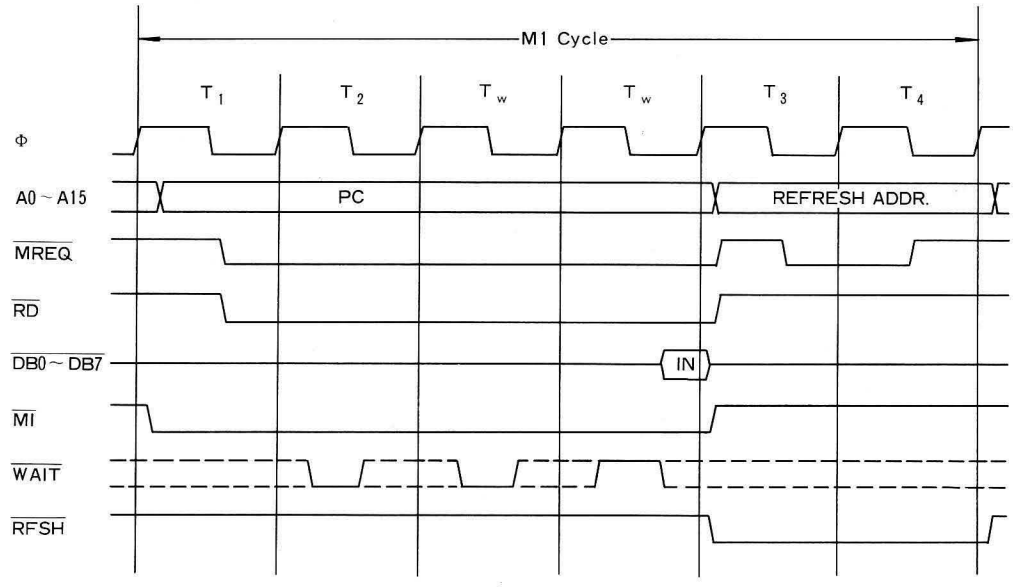
INSTRUCTION FETCH

Figure 3.0-1 shows the timing during an M1 cycle (OP code fetch). Notice that the PC is placed on the address bus at the beginning of the M1 cycle. One half clock time later the $\overline{\text{MREQ}}$ signal goes active. At this time the address to the memory has had time to stabilize so that the falling edge of $\overline{\text{MREQ}}$ can be used directly as a chip enable clock to dynamic memories. The $\overline{\text{RF}}$ line also goes active to indicate that the memory read data should be enabled onto the CPU data bus. The CPU samples the data from the memory on the data bus with the rising edge of the clock of state T3 and this same edge is used by the CPU to turn off the $\overline{\text{RD}}$ and $\overline{\text{MRQ}}$ signals. Thus the data has already been sampled by the CPU before the $\overline{\text{RD}}$ signal becomes inactive. Clock state T3 and T4 of a fetch cycle are used to refresh dynamic memories. (The CPU uses this time to decode and execute the fetched instruction so that no other operation could be performed at this time). During T3 and T4 the lower 7 bits of the address bus contain a memory refresh address and the $\overline{\text{RFSH}}$ signal becomes active to indicate that a refresh read of all dynamic memories should be accomplished. Notice that a $\overline{\text{RD}}$ sgianl is not generated during refresh time to prevent data from different memory segments from being gated onto the data bus. The $\overline{\text{MREQ}}$ signal during refresh time should be used to perform a refresh read of all memory elements. The refresh signal can not be used by itself since the refresh address is only guaranteed to be stable during $\overline{\text{MREQ}}$ time.



INSTRUCTION OP CODE FETCH
FIGURE 3.0-1

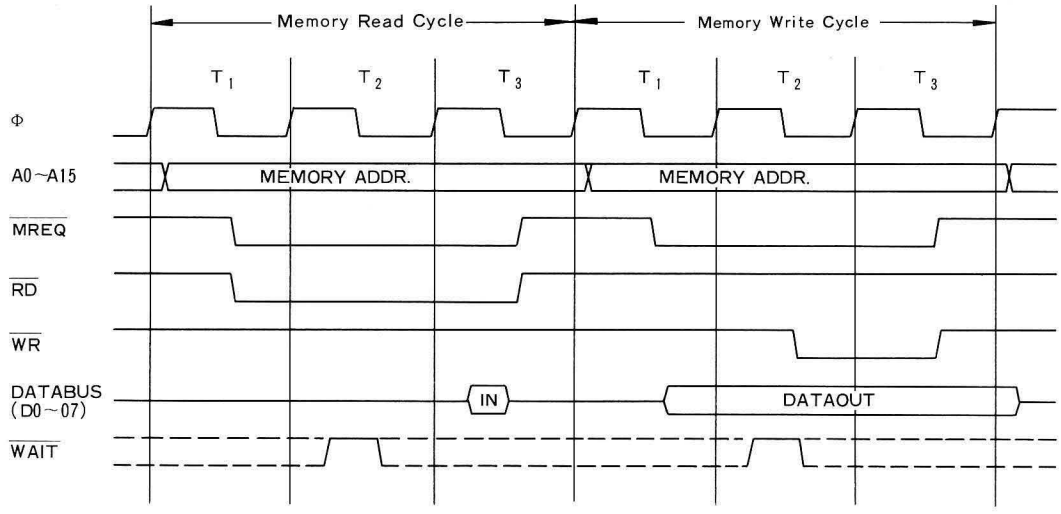
Figure 3.0-1A illustrates how the fetch cycle is delayed if the memory activates the $\overline{\text{WAIT}}$ line. During T2 and every subsequent Tw, the CPU samples the $\overline{\text{WAIT}}$ line with the falling edge of Φ. If the WAIT line is active at this time, another wait state will be entered during the following cycle. Using this technique the read cycle can be lengthened to match the access time of any type of memory device.



INSTRUCTION OP CODE FETCH WITH WAIT STATES
FIGURE 3.0-1A

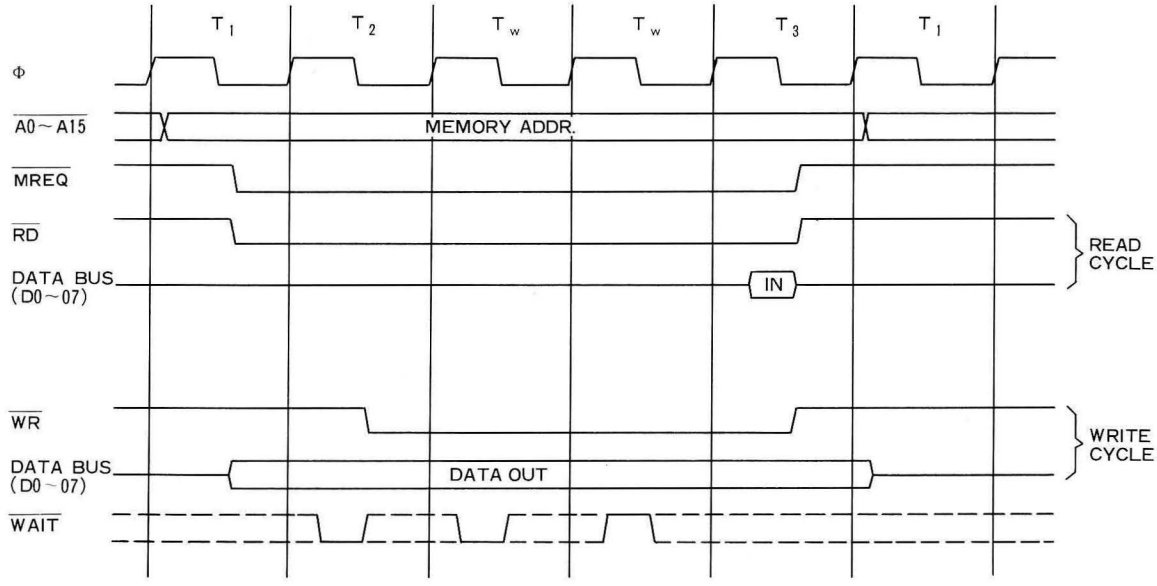
MEMORY READ OR WRITE

Figure 3.0-2 illustrates the timing of memory read or write cycles other than an OP code fetch (M1 cycle). These cycles are generally three clock periods long unless wait states are requested by the memory via the $\overline{WAIT}$ signal. The $\overline{MREQ}$ signal and the $\overline{RD}$ signal are used the same as in the fetch cycle. In the case of a memory write cycle, the $\overline{MREQ}$ also becomes active when the address bus is stable so that it can be used directly as a chip enable for dynamic memories. The $\overline{WR}$ line is active when data on the data bus is stable so that it can be used directly as a R/W pulse to virtually any type of semiconductor memory. Furthermore the $\overline{WR}$ signal goes inactive one half T state before the address and data bus contents are changed so that the overlap requirements for virtually any type of semiconductor memory type will be met.



MEMORY READ OR WRITE CYCLES
FIGURE 3.0-2

Figure 3.0-2A illustrates how a $\overline{WAIT}$ request signal will lengthen any memory read or write operation. This operation is identical to that previously described for a fetch cycle. Notice in this figure that a separate read and a separate write cycle are shown in the same figure although read and write cycles can never occur simultaneously.



MEMORY READ OR WRITE CYCLES WITH WAIT STATES
FIGURE 3.0-2A

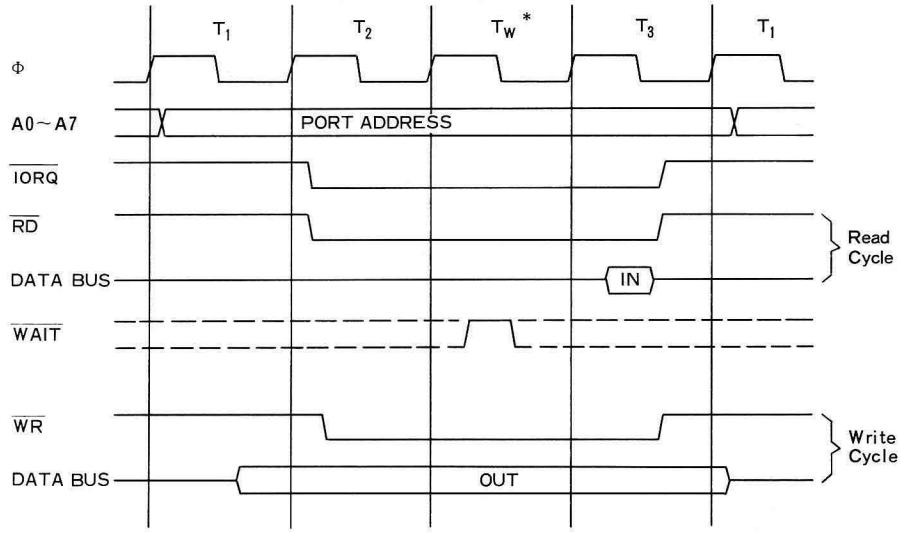
INPUT OR OUTPUT CYCLES

Figure 3.0-3 illustrates an I/O read or I/O write operation. Notice that during I/O operations a single wait state is automatically inserted. The reason for this is that during I/O operations, the time from when the $\overline{IORQ}$ signal goes active until the CPU must sample the $\overline{WAIT}$ line is very short and without this extra state sufficient time does not exist for an I/O port to decode its address and activate the $\overline{WAIT}$ line if a wait is required. Also, without this wait state it is difficult to design MOS I/O devices that can operate at full CPU speed. During this wait state time the $\overline{WAIT}$ request signal is sampled. During a read I/O operation, the $\overline{RD}$ line is used to enable the addressed port onto the data bus just as in the case of a memory read. For I/O write operations, the $\overline{WR}$ line is used as a clock to the I/O port, again with sufficient overlap timing automatically provided so that the rising edge may be used as a data clock.

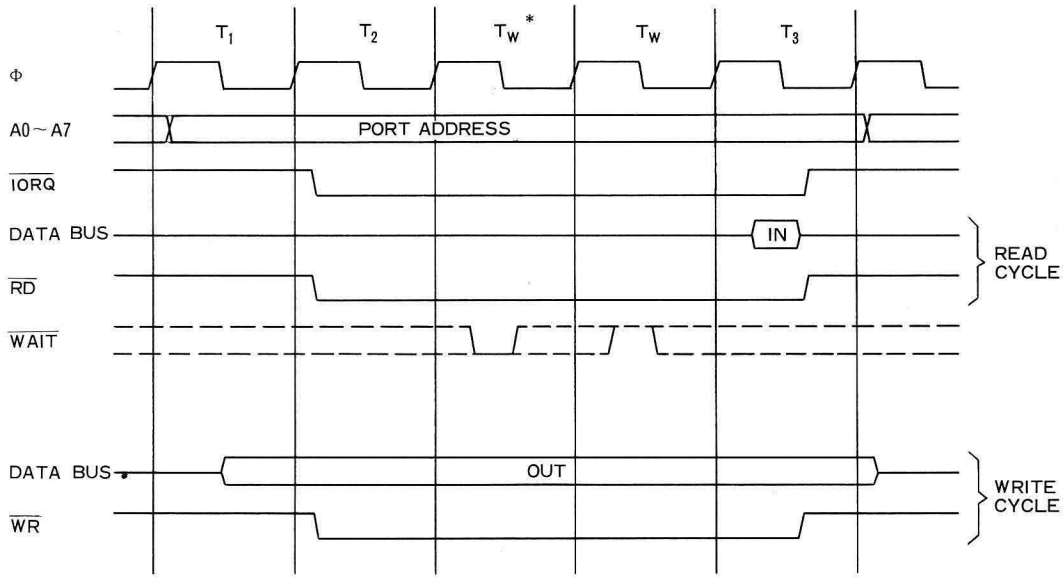
Figure 3.0-3A illustrates how additional wait states may be added with the $\overline{WAIT}$ line. The operation is identical to that previously described.

BUS REQUEST/ACKNOWLEDGE CYCLE

Figure 3.0-4 illustrates the timing for a Bus Request/Acknowledge cycle. The $\overline{BUSRQ}$ signal is sampled by the CPU with the rising edge of the last clock period of any machine cycle. If the $\overline{BUSRQ}$ signal is active, the CPU will set its address, data and tri-state control signals to the high impedance state with the rising edge of the next clock pulse. At that time any external device can control the buses to transfer data between memory and I/O devices. (This is generally known as Direct Memory Access [DMA] using cycle stealing). The maximum time for the CPU to respond to a bus request is the length of a machine cycle and the external controller can maintain control of the bus for as many clock cycles as is desired. Note, however, that if very long DMA cycles are used, and dynamic memories are being used, the external controller must also perform the refresh function. This situation only occurs if very large blocks of data are transferred under DMA control. Also note that during a bus request cycle, the CPU cannot be interrupted by either a $\overline{NMI}$ or an $\overline{INT}$ signal.

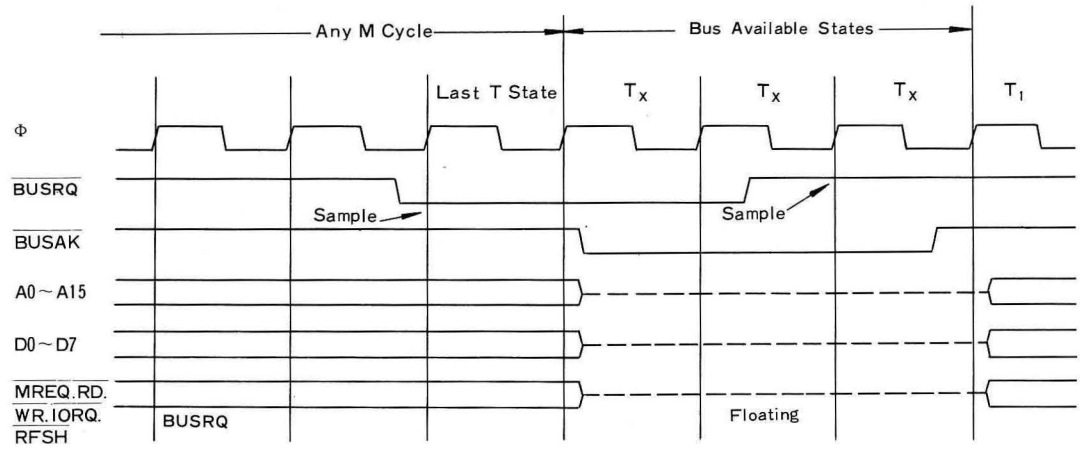


INPUT OR OUTPUT CYCLES
FIGURE 3.0-3



* Automatically inserted WAIT state

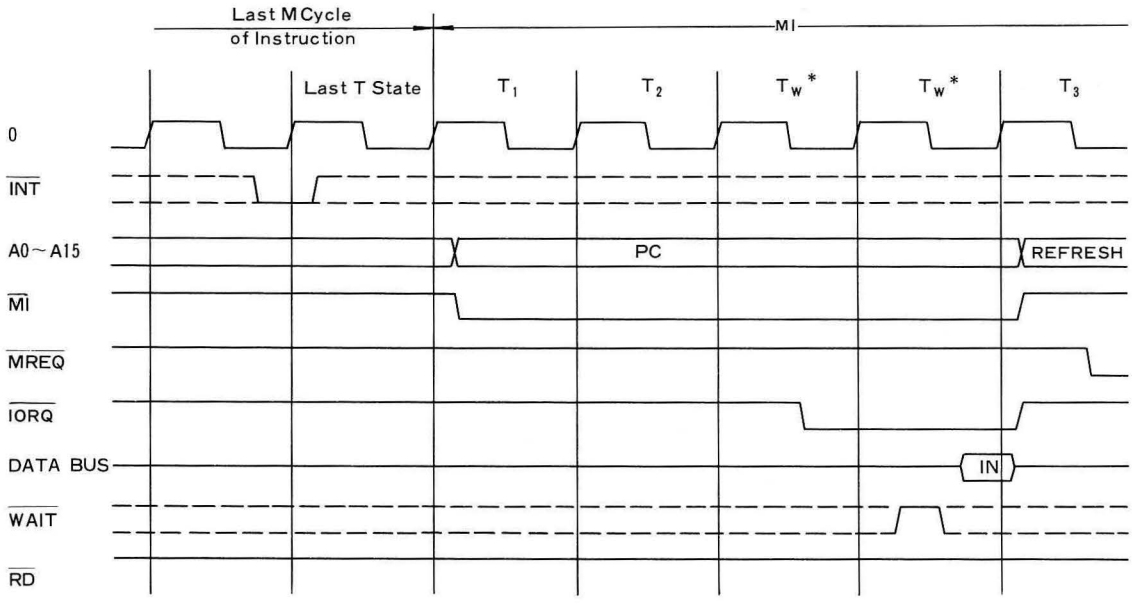
INPUT OR OUTPUT CYCLES WITH WAIT STATES
FIGURE 3.0-3A



BUS REQUEST/ACKNOWLEDGE CYCLE
FIGURE 3.0-4

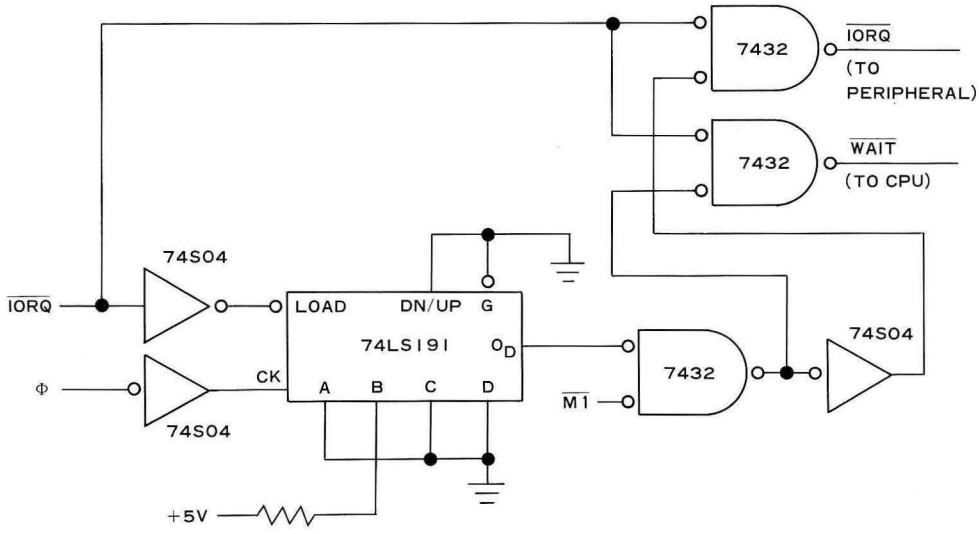
INTERRUPT REQUEST/ACKNOWLEDGE CYCLE

Figure 3.0-5 illustrates the timing associated with an interrupt cycle. The interrupt signal ($\overline{INT}$) is sampled by the CPU with the rising edge of the last clock at the end of any instruction. The signal will not be accepted if the internal CPU software controlled interrupt enable flip-flop is not set or if the $\overline{BUSRQ}$ signal is active. When the signal is accepted a special M1 cycle is generated. During this special M1 cycle the $\overline{IORQ}$ signal becomes active (instead of the normal $\overline{MREQ}$) to indicate that the interrupting device can place an 8-bit vector on the data bus. Notice that two wait states are automatically added to this cycle. These states are added so that a ripple priority interrupt scheme can be easily implemented. The two wait states allow sufficient time for the ripple signals to stabilize and identify which I/O device must insert the response vector. Refer to section 5.0 for details on how the interrupt response vector is utilized by the CPU.

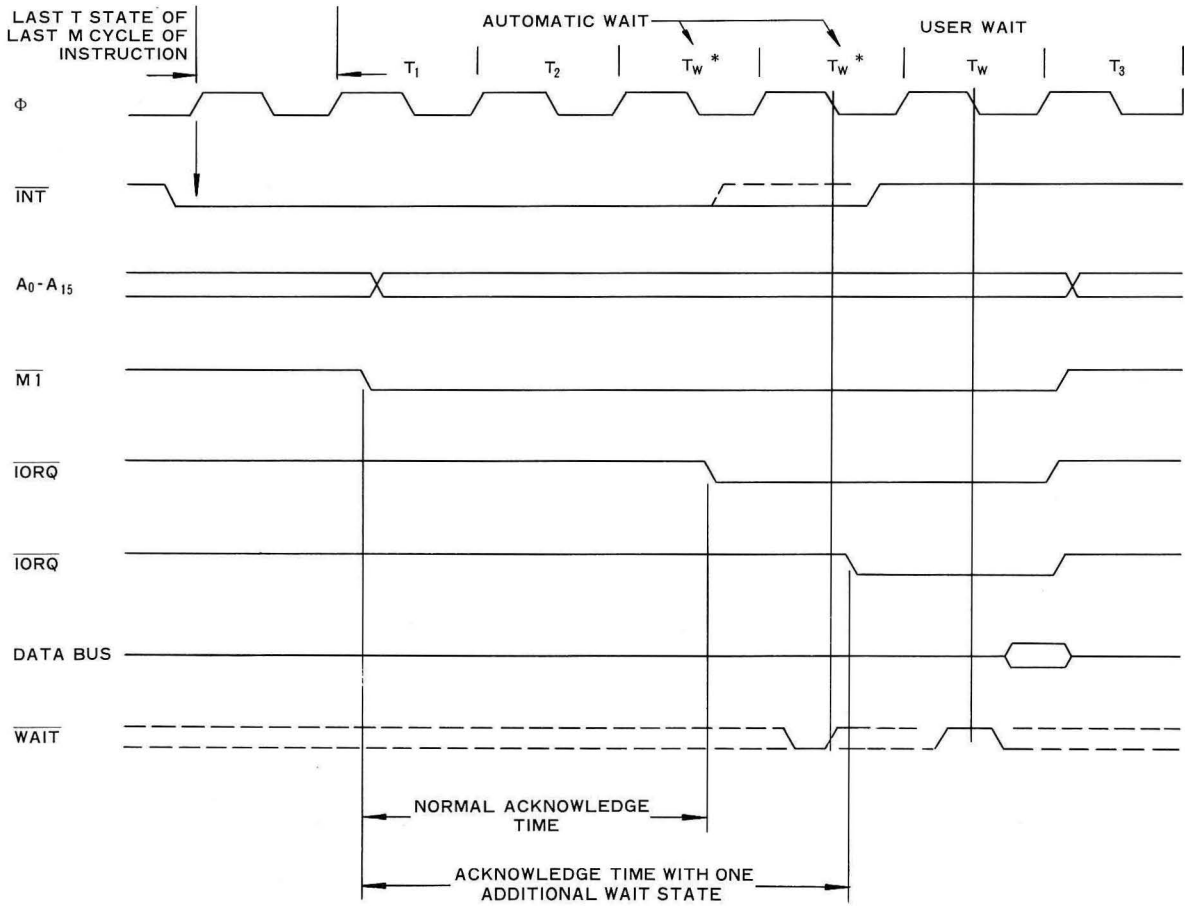


INTERRUPT REQUEST/ACKNOWLEDGE CYCLE
FIGURE 3.0-5

Figures 3.0-5A and 3.0-5B illustrate how a programmable counter can be used to extend interrupt acknowledge time. (Configured as shown to add one wait state)



EXTENDING INTERRUPT ACKNOWLEDGE TIME WITH WAIT STATE
FIGURE 3.0-5A



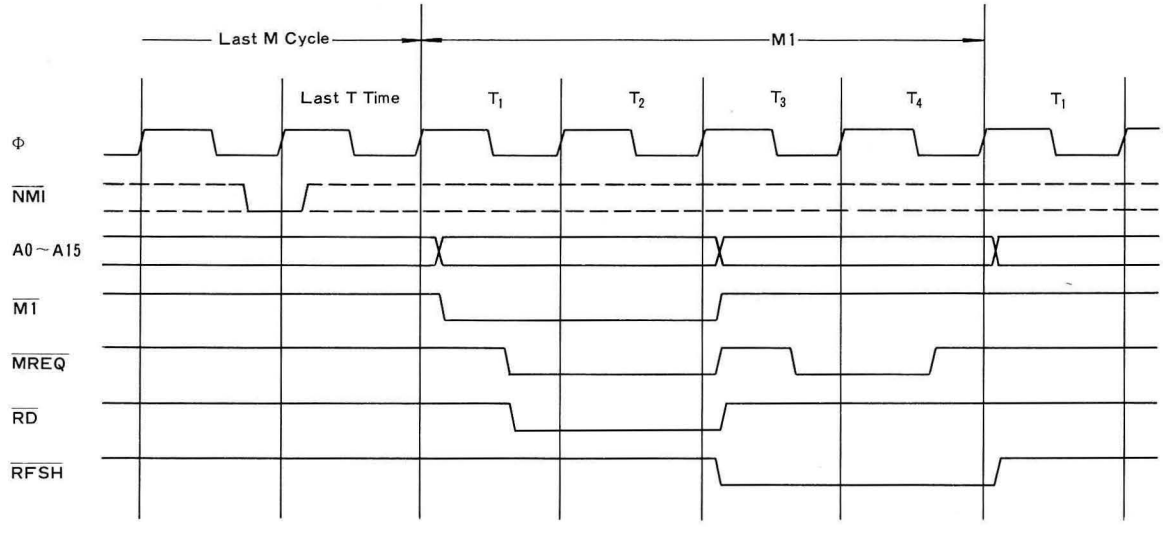
REQUEST/ACKNOWLEDGE CYCLE WITH ONE ADDITIONAL WAIT STATE
FIGURE 3.0-5B

NON MASKABLE INTERRUPT RESPONSE

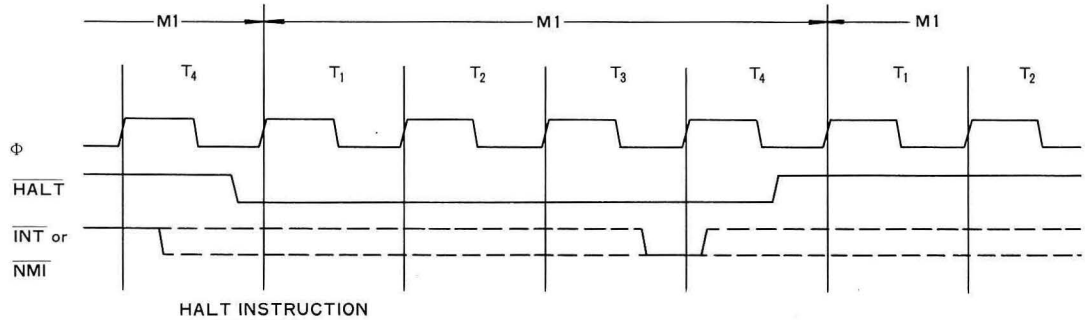
Figure 3.0-6 illustrates the request/acknowledge cycle for the non maskable interrupt. This signal is sampled at the same time as the interrupt line, but this line has priority over the normal interrupt and it can not be disabled under software control. Its usual function is to provide immediate response to important signals such as an impending power failure. The CPU response to a non maskable interrupt is similar to a normal memory read operation. The only difference being that the content of the data bus is ignored while the processor automatically stores the PC in the external stack and jumps to location 0066_H. The service routine for the non maskable interrupt must begin at this location if this interrupt is used.

HALT EXIT

Whenever a software halt instruction is executed the CPU begins executing NOP's until an interrupt is received (either a non maskable or a maskable interrupt while the interrupt flip flop is enabled). The two interrupt lines are sampled with the rising clock edge during each T4 state as shown in Figure 3.0-7. If a non maskable interrupt has been received or a maskable interrupt has been received and the interrupt enable flip-flop is set, then the halt state will be exited on the next rising clock edge. The following cycle will then be an interrupt acknowledge cycle corresponding to the type of interrupt that was received. If both are received at this time, then the non maskable one will be acknowledged since it has highest priority. The purpose of executing NOP instructions while in the halt state is to keep the memory refresh signals active. Each cycle in the halt state is a normal M1 (fetch) cycle except that the data received from the memory is ignored and a NOP instruction is forced internally to the CPU. The halt acknowledge signal is active during this time to indicate that the processor is in the halt state.



NON MASKABLE INTERRUPT REQUEST OPERATION
FIGURE 3.0-6



HALT INSTRUCTION IS RECEIVED DURING THIS MEMORY CYCLE

HALT EXIT
FIGURE 3.0-7

4.0 INSTRUCTION SET

The Z-80A CPU can execute 158 different instruction types including all 78 of the 8080A CPU. The instructions can be broken down into the following major groups:

- Load and Exchange
- Block Transfer and Search
- Arithmetic and Logical
- Rotate and Shift
- Bit Manipulation (set, reset, test)
- Jump, Call and Return
- Input/Output
- Basic CPU Control

4.1 INTRODUCTION TO INSTRUCTION TYPES

The load instructions move data internally between CPU registers or between CPU registers and external memory. All of these instructions must specify a source location from which the data is to be moved and a destination location. The source location is not altered by a load instruction. Examples of load group instructions include moves between any of the general purpose registers such as move the data to Register B from Register C. This group also includes load immediate to any CPU register or to any to Register B from Register C. This group also includes load immediate to any CPU register or to any external memory location. Other types of load instructions allow transfer between CPU registers and memory locations. The exchange instructions can trade the contents of two registers.

A unique set of block transfer instructions is provided in the Z-80A. With a single instruction a block of memory of any size can be moved to any other location in memory. This set of block moves is extremely valuable when large strings of data must be processed. The Z-80A block search instructions are also valuable for this type of processing. With a single instruction, a block of external memory of any desired length can be searched for any 8-bit character. Once the character is found or the end of the block is reached, the instruction automatically terminates. Both the block transfer and the block search instructions can be interrupted during their execution so as to not occupy the CPU for long periods of time.

The arithmetic and logical instructions operate on data stored in the accumulator and other general purpose CPU registers or external memory locations. The results of the operations are placed in the accumulator and the appropriate flags are set according to the result of the operation. An example of an arithmetic operation is adding the accumulator to the contents of an external memory location. The results of the addition are placed in the accumulator. This group also includes 16-bit addition and subtraction between 16-bit CPU registers.

The rotate and shift group allows any register or any memory location to be rotated right or left with or without carry either arithmetic or logical. Also, a digit in the accumulator can be rotated right or left with two digits in any memory location.

The bit manipulation instructions allow any bit in the accumulator, any general purpose register or any external memory location to be set, reset or tested with a single instruction. For example, the most significant bit of register H can be reset. This group is especially useful in control applications and for controlling software flags in general purpose programming.

The jump, call and return instructions are used to transfer between various locations in the user's program. This group uses several different techniques for obtaining the new program counter address from specific external memory locations. A unique type of call is the restart instruction. This instruction actually contains the new address as a part of the 8-bit OP code. This is possible since only 8 separate addresses located in page zero of the external memory may be specified. Program jumps may also be achieved by loading register HL, IX or IY directly into the PC, thus allowing the jump address to be a complex function of the routine being executed.

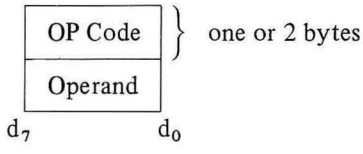
The input/output group of instructions in the Z-80A allow for a wide range of transfers between external memory locations or the general purpose CPU registers, and the external I/O devices. In each case, the port number is provided on the lower 8 bits of the address bus during any I/O transaction. One instruction allows this port number to be specified by the second byte of the instruction while other Z-80A instructions allow it to be specified as the content of the C register. One major advantage of using the C register as a pointer to the I/O device is that it allows different I/O ports to share common software driver routines. This is not possible when the address is part of the OP code if the routines are stored in ROM. Another feature of these input instructions is that they set the flag register automatically so that additional operations are not required to determine the state of the input data (for example its parity). The Z-80A CPU includes single instructions that can move blocks of data (up to 256 bytes) automatically to or from any I/O port directly to any memory location. In conjunction with the dual set of general purpose registers, these instructions provide for fast I/O block transfer rates. The value of this I/O instruction set is demonstrated by the fact that the Z-80A CPU can provide all required floppy disk formatting (i.e., the CPU provides the preamble, address, data and enables the CRC codes) on double density floppy disk drives on an interrupt driven basis.

Finally, the basic CPU control instructions allow various options and modes. This group includes instructions such as setting or resetting the interrupt enable flip flop or setting the mode of interrupt response.

4.2 ADDRESSING MODES

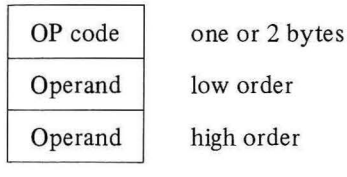
Most of the Z-80A instructions operate on data stored in internal CPU registers, external memory or in the I/O ports. Addressing refers to how the address of this data is generated in each instruction. This section gives a brief summary of the types of addressing used in the Z-80A while subsequent sections detail the type of addressing available for each instruction group.

Immediate. In this mode of addressing the byte following the OP code in memory contains the actual operand.



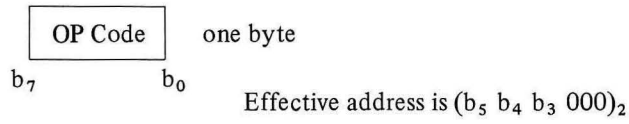
Examples of this type of instruction would be to load the accumulator with a constant, where the constant is the byte immediately following the OP code.

Immediate Extended. This mode is merely an extension of immediate addressing in that the two bytes following the OP codes are the operand.

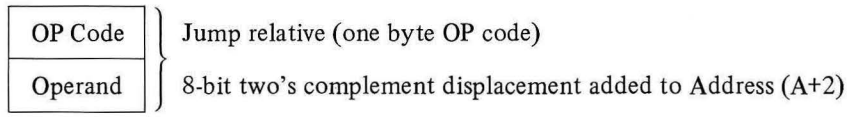


Examples of this type of instruction would be to load the HL register pair (16-bit register) with 16 bits (2 bytes) of data.

Modified Page Zero Addressing. The Z-80A has a special single byte CALL instruction to any of 8 locations in page zero of memory. This instruction (which is referred to as a restart) sets the PC to an effective address in page zero. The value of this instruction is that it allows a single byte to specify a complete 16-bit address where commonly called subroutines are located, thus saving memory space.

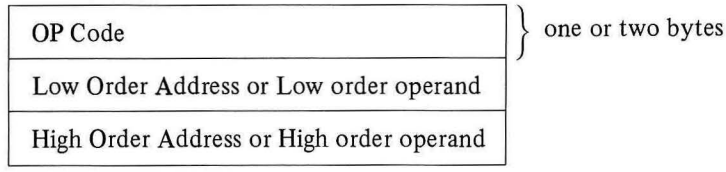


Relative Addressing. Relative addressing uses one byte of data following the OP code to specify a displacement from the existing program to which a program jump can occur. This displacement is a signed two's complement number that is added to the address of the OP code of the following instruction.



The value of relative addressing is that it allows jumps to nearby locations while only requiring two bytes of memory space. For most programs, relative jumps are by far the most prevalent type of jump due to the proximity of related program segments. Thus, these instructions can significantly reduce memory space requirements. The signal displacement can range between +127 and −128 from A+2. This allows for a total displacement of +129 to −126 from the jump relative OP code address. Another major advantage is that it allows for relocatable code.

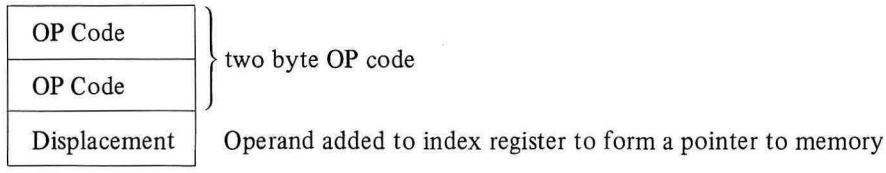
Extended Addressing. Extended Addressing provides for two bytes (16 bits) of address to be included in the instruction. This data can be an address to which a program can jump or it can be an address where an operand is located.



Extended addressing is required for a program to jump from any location in memory to any other location, or load and store data in any memory location.

When extended addressing is used to specify the source or destination address of an operand, the notation (nn) will be used to indicate the content of memory at nn, where nn is the 16-bit address specified in the instruction. This means that the two bytes of address nn are used as a pointer to a memory location. The use of the parentheses always means that the value enclosed within them is used as a pointer to a memory location. For example, (1200) refers to the contents of memory at location 1200.

Indexed Addressing. In this type of addressing, the byte of data following the OP code contains a displacement which is added to one of the two index registers (the OP code specifies which index register is used) to form a pointer to memory. The contents of the index register are not altered by this operation.



An example of an indexed instruction would be to load the contents of the memory location (Index Register + Displacement) into the accumulator. The displacement is a signed two's complement number. Indexed addressing greatly simplifies programs using tables of data since the index register can point to the start of any table. Two index registers are provided since very often operations require two or more tables. Indexed addressing also allows for relocatable code.

The two index registers in the Z-80A are referred to as IX and IY. To indicate indexed addressing the notation:

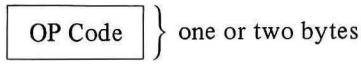
(IX + d) or (IY + d)

is used. Here d is the displacement specified after the OP code. The parentheses indicate that this value is used as a pointer to external memory.

Register Addressing. Many of the Z-80A OP codes contain bits of information that specify which CPU register is to be used for an operation. An example of register addressing would be to load the data in register B into register C.

Implied Addressing. Implied addressing refers to operations where the OP code automatically implies one or more CPU registers as containing the operands. An example is the set of arithmetic operations where the accumulator is always implied to be the destination of the results.

Register Indirect Addressing. This type of addressing specifies a 16-bit CPU register pair (such as HL) to be used as a pointer to any location in memory. This type of instruction is very powerful and it is used in a wide range of applications.



An example of this type of instruction would be to load the accumulator with the data in the memory location pointed to by the HL register contents. Indexed addressing is actually a form of register indirect addressing except that a displacement is added with indexed addressing. Register indirect addressing allows for very powerful but simple to implement memory accesses. The block move and search commands in the Z-80A are extensions of this type of addressing where automatic register incrementing, decrementing and comparing has been added. The notation for indicating register indirect addressing is to put parentheses around the name of the register that is to be used as the pointer. For example, the symbol

(HL)

specifies that the contents of the HL register are to be used as a pointer to a memory location. Often register indirect addressing is used to specify 16-bit operands. In this case, the register contents point to the lower order portion of the operand while the register contents are automatically incremented to obtain the upper portion of the operand.

Bit Addressing. The Z-80A contains a large number of bit set, reset and test instructions. These instructions allow any memory location or CPU register to be specified for a bit operation through one of three previous addressing modes (register, register indirect and indexed) while three bits in the OP code specify which of the eight bits is to be manipulated.

ADDRESSING MODE COMBINATIONS

Many instructions include more than one operand (such as arithmetic instructions or loads). In these cases, two types of addressing may be employed. For example, load can use immediate addressing to specify the source and register indirect or indexed addressing to specify the destination.

4.3 INSTRUCTION OF OP CODES AND EXECUTION TIMES

The following section gives a summary of the Z-80A instructions set. The instructions are logically arranged into groups as shown on tables 4.3-1 through 4.3-11. Each table shows the assembly language mnemonic OP code, the actual OP code, the symbolic operation, the content of the flag register following the execution of each instruction, the number of bytes required for each instruction as well as the number of memory cycles and the total number of T states (external clock periods) required for the fetching and execution of each instruction. Care has been taken to make each table self-explanatory without requiring any cross reference with the test or other tables.

Mnemonic	Symbolic Operation	Flags						OP-Code	No. of Bytes	No. of M Cycles	No. of T States	Comments
		C	Z	P/V	S	N	H	76 543 210				
LD r,r′	r←r′	•	•	•	•	•	•	01 r r′	1	1	4	
LD r,n	r←n	•	•	•	•	•	•	00 r 110 ← n →	2	2	7	
LD r,(HL)	r←(HL)	•	•	•	•	•	•	01 r 110	1	2	7	
LD r,(IX+d)	r←(IX+d)	•	•	•	•	•	•	11 011 101 01 r 110 ← d →	3	5	19	
LD r,(IY+d)	r←(IY+d)	•	•	•	•	•	•	11 111 101 01 r 110 ← d →	3	5	19	
LD (HL),r	(HL)←r	•	•	•	•	•	•	01 110 r	1	2	7	
LD (IX+d),r	(IX+d)←r	•	•	•	•	•	•	11 011 101 01 110 r ← d →	3	5	19	
LD (IY+d),r	(IY+d)←r	•	•	•	•	•	•	11 111 101 01 110 r ← d →	3	5	19	
LD (HL),n	(HL)←n	•	•	•	•	•	•	00 110 110 ← n →	2	3	10	
LD (IX+d),n	(IX+d)←n	•	•	•	•	•	•	11 011 101 00 110 110 ← d → ← n →	4	5	19	
LD (IY+d),n	(IY+d)←n	•	•	•	•	•	•	11 111 101 00 110 110 ← d → ← n →	4	5	19	
LD A,(BC)	A←(BC)	•	•	•	•	•	•	00 001 010	1	2	7	
LD A,(DE)	A←(DE)	•	•	•	•	•	•	00 011 010	1	2	7	
LD A,(nn)	A←(nn)	•	•	•	•	•	•	00 111 010 ← n → ← n →	3	4	13	
LD (BC),A	(BC)←A	•	•	•	•	•	•	00 000 010	1	2	7	
LD (DE),A	(DE)←A	•	•	•	•	•	•	00 010 010	1	2	7	
LD (nn),A	(nn)←A	•	•	•	•	•	•	00 110 010 ← n → ← n →	3	4	13	
LD A,I	A←I	•	‡	IFF2	‡	0	0	11 101 101 01 010 111	2	2	9	
LD A,R	A←R	•	‡	IFF2	‡	0	0	11 101 101 01 011 111	2	2	9	
LD I,A	I←A	•	•	•	•	•	•	11 101 101 01 000 111	2	2	9	
LD R,A	R←A	•	•	•	•	•	•	11 101 101 01 001 111	2	2	9	

r, r′	Reg.
000	B
001	C
010	D
011	E
100	H
101	L
111	A

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.

8-BIT LOAD GROUP
TABLE 4.3-1

Mnemonic	Symbolic Operation	Flags						OP-Code	No. of Bytes	No. of M Cycles	No. of T States	Comments
		C	Z	P/V	S	N	H	76 543 210				
LD dd,nn	dd←nn	•	•	•	•	•	•	00 dd0 001 ← n → ← n →	3	3	10	dd / Pair 00 BC 01 DE 10 HL 11 SP
LD IX,nn	IX←nn	•	•	•	•	•	•	11 011 101 00 100 001 ← n → ← n →	4	4	14	
LD IY,nn	IY←nn	•	•	•	•	•	•	11 111 101 00 100 001 ← n → ← n →	4	4	14	
LD HL,(nn)	H←(nn+1) L←(nn)	•	•	•	•	•	•	00 101 010 ← n → ← n →	3	5	16	
LD dd,(nn)	dd_H←(nn+1) dd_L←(nn)	•	•	•	•	•	•	11 101 101 01 dd1 011 ← n → ← n →	4	6	20	
LD IX,(nn)	IX_H←(nn+1) IX_L←(nn)	•	•	•	•	•	•	11 011 101 00 101 010 ← n → ← n →	4	6	20	
LD IY,(nn)	IY_H←(nn+1) IY_L←(nn)	•	•	•	•	•	•	11 111 101 00 101 010 ← n → ← n →	4	6	20	
LD (nn),HL	(nn+1)←H (nn)←L	•	•	•	•	•	•	00 100 010 ← n → ← n →	3	5	16	
LD (nn),dd	(nn+1)←dd_H (nn)←dd_L	•	•	•	•	•	•	11 101 101 01 dd0 011 ← n → ← n →	4	6	20	
LD (nn),IX	(nn+1)←IX_H (nn)←IX_L	•	•	•	•	•	•	11 011 101 00 100 010 ← n → ← n →	4	6	20	
LD (nn),IY	(nn+1)←IY_H (nn)←IY_L	•	•	•	•	•	•	11 111 101 00 100 010 ← n → ← n →	4	6	20	
LD SP,HL	SP←HL	•	•	•	•	•	•	11 111 001	1	1	6	
LD SP,IX	SP←IX	•	•	•	•	•	•	11 011 101 11 111 001	2	2	10	
LD SP,IY	SP←IY	•	•	•	•	•	•	11 111 101 11 111 001	2	2	10	

Mnemonic	Symbolic Operation	Flags						OP-Code	No. of Bytes	No. of M Cycles	No. of T States	Comments
		C	Z	P/V	S	N	H	76 543 210				
PUSH qq	$(SP-2)\leftarrow qq_L$ $(SP-1)\leftarrow qq_H$	•	•	•	•	•	•	11 qq0 101	1	3	11	
PUSH IX	$(SP-2)\leftarrow IX_L$ $(SP-1)\leftarrow IX_H$	•	•	•	•	•	•	11 011 101 11 100 101	2	4	15	
PUSH IY	$(SP-2)\leftarrow IY_L$ $(SP-1)\leftarrow IY_H$	•	•	•	•	•	•	11 111 101 11 100 101	2	4	15	
POP qq	$qq_H\leftarrow(SP+1)$ $qq_L\leftarrow(SP)$	•	•	•	•	•	•	11 qq0 001	1	3	10	
POP IX	$IX_H\leftarrow(SP+1)$ $IX_L\leftarrow(SP)$	•	•	•	•	•	•	11 011 101 11 100 001	2	4	14	
POP IY	$IY_H\leftarrow(SP+1)$ $IY_L\leftarrow(SP)$	•	•	•	•	•	•	11 111 101 11 100 001	2	4	14	

qq	Pair
00	BC
01	DE
10	HL
11	AF

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.

16-BIT LOAD GROUP
TABLE 4.3-2

Mnemonic	Symbolic Operation	Flags						OP-Code	No. of Bytes	No. of M Cycles	No. of T States	Comments
		C	Z	P/V	S	N	H	76 543 210				
EX DE, HL	DE↔HL	•	•	•	•	•	•	11 101 011	1	1	4	
EX AF, AF′	AF↔AF′	•	•	•	•	•	•	00 001 000	1	1	4	
EXX	$\begin{pmatrix}BC\\DE\\HL\end{pmatrix}\leftrightarrow\begin{pmatrix}BC'\\DE'\\HL'\end{pmatrix}$	•	•	•	•	•	•	11 011 001	1	1	4	Register bank and auxiliary register bank exchange
EX (SP), HL	H↔(SP+1) L↔(SP)	•	•	•	•	•	•	11 100 011	1	5	19	
EX (SP), IX	IX_H↔(SP+1) IX_L↔(SP)	•	•	•	•	•	•	11 011 101 11 100 011	2	6	23	
EX (SP), IY	IY_H↔(SP+1) IY_L↔(SP)	•	•	•	•	•	•	11 111 101 11 100 011	2	6	23	
LDI	(DE)←(HL) DE←DE+1 HL←HL+1 BC←BC−1	•	•	↕ ①	•	0	0	11 101 101 10 100 000	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	•	•	0 ①	•	0	0	11 101 101 10 110 000	2 2	5 4	21 16	If BC≠0 If BC=0
LDD	(DE)←(HL) DE←DE−1 HL←HL−1 BC←BC−1	•	•	↕ ①	•	0	0	11 101 101 10 101 000	2	4	16	
LDDR	(DE)←(HL) DE←DE-1 HL←HL-1 BC←BC-1 Repeat until BC=0	•	•	0 ①	•	0	0	11 101 101 10 111 000	2 2	5 4	21 16	If BC≠0 If BC=0
CPI	A−(HL) HL←HL+1 BC←BC−1	•	↕ ②	↕ ①	↕	1	↕	11 101 101 10 100 001	2	4	16	

Mnemonic	Symbolic Operation	Flags						OP-Code	No. of Bytes	No. of M Cycles	No. of T States	Comments
		C	Z	P/V	S	N	H	76 543 210				
			②	①								
CPIR	A−(HL) HL←HL+1 BC←BC−1 Repeat until A=(HL) or BC=0	•	↕	↕	↕	1	↕	11 101 101 10 110 001	2 2	5 4	21 16	If BC≠0 and A≠(HL) If BC=0 or A=(HL)
			②	①								
CPD	A−(HL) HL←HL−1 BC←BC−1	•	↕	↕	↕	1	↕	11 101 101 10 101 001	2	4	16	
			②	①								
CPDR	A−(HL) HL←HL−1 BC←BC−1 Repeat until A=(HL) or BC=0	•	↕	↕	↕	1	↕	11 101 101 10 111 001	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.

EXCHANGE GROUP AND BLOCK TRANSFER AND SEARCH GROUP
TABLE 4.3-3

Mnemonic	Symbolic Operation	Flags						OP-Code	No. of Bytes	No. of M Cycles	No. of T States	Comments
		C	Z	P/V	S	N	H	76 543 210				
ADD A,r	A←A+r	‡	‡	V	‡	0	‡	10 [000] r	1	1	4	r: B, C, D, E, H, L, A
ADD A,n	A←A+n	‡	‡	V	‡	0	‡	11 [000] 110 ← n →	2	2	7	
ADD A,(HL)	A←A+(HL)	‡	‡	V	‡	0	‡	10 [000] 110	1	2	7	
ADD A,(IX+d)	A←A+(IX+d)	‡	‡	V	‡	0	‡	11 011 101 10 [000] 110 ← d →	3	5	19	
ADD A,(IY+d)	A←A+(IY+d)	‡	‡	V	‡	0	‡	11 111 101 10 [000] 110 ← d →	3	5	19	
ADC A,s	A←A+s+CY	‡	‡	V	‡	0	‡	[001]				s is any of r, n, (HL), (IX+d), (IY+d) as shown for ADD instruction
SUB s	A←A−s	‡	‡	V	‡	1	‡	[010]				The indicated bits replace the [000] in the ADD set above.
SBC A,s	A←A−s−CY	‡	‡	V	‡	1	‡	[011]				
AND s	A←A∧s	0	‡	P	‡	0	1	[100]				
OR s	A←A∨s	0	‡	P	‡	0	0	[110]				
XOR s	A←A∀s	0	‡	P	‡	0	0	[101]				
CP s	A−s	‡	‡	V	‡	1	‡	[111]				
INC r	r←r+1	•	‡	V	‡	0	‡	00 r [100]	1	1	4	
INC (HL)	(HL)←(HL)+1	•	‡	V	‡	0	‡	00 110 [100]	1	3	11	
INC (IX+d)	(IX+d)←(IX+d)+1	•	‡	V	‡	0	‡	11 011 101 00 110 [100] ← d →	3	6	23	
INC (IY+d)	(IY+d)←(IY+d)+1	•	‡	V	‡	0	‡	11 111 101 00 110 [100] ← d →	3	6	23	
DEC m	m←m−1	•	‡	V	‡	1	‡	[101]				m is any of r, (HL), (IX+d), (IY+d) as shown for INC. Same format and states as INC. Replace [100] with [101] in OP-code.

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 panty 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.

8-BIT ARITHMETIC AND LOGICAL GROUP
TABLE 4.3-4

Mnemonic	Symbolic Operation	Flags						OP-Code	No. of Bytes	No. of M Cycles	No. of T States	Comments
		C	Z	P/V	S	N	H	76 543 210				
DAA	Converts acc content into packed BCD following add or subtract with packed BCD operands	‡	‡	P	‡	•	‡	00 100 111	1	1	4	Decimal adjust accumulator
CPL	$A \leftarrow \bar{A}$	•	•	•	•	1	1	00 101 111	1	1	4	Complement accumulator (one's complement) N
NEG	$A \leftarrow \bar{A}+1$	‡	‡	V	‡	1	‡	11 101 101 01 000 100	2	2	8	Negate acc. (two's complement)
CCF	$CY \leftarrow \overline{CY}$	‡	•	•	•	0	X	00 111 111	1	1	4	Complement carry flag
SCF	CY←1	1	•	•	•	0	0	00 110 111	1	1	4	Set carry flag
NOP	No operation PC←PC+1	•	•	•	•	•	•	00 000 000	1	1	4	
HALT	CPU halted	•	•	•	•	•	•	01 110 110	1	1	4	
DI	IFF←0	•	•	•	•	•	•	11 110 011	1	1	4	
EI	IFF←1	•	•	•	•	•	•	11 111 011	1	1	4	
IM 0	Sst interrupt mode 0	•	•	•	•	•	•	11 101 101 01 000 110	2	2	8	
IM 1	Set interrupt mode 1	•	•	•	•	•	•	11 101 101 01 010 110	2	2	8	
IM 2	Set interrupt mode 2	•	•	•	•	•	•	11 101 101 01 011 110	2	2	8	

Notes: IFF indicates the interrupt enable flip-flop
CY indicates the carry flip-flop.

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.

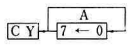
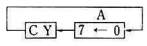
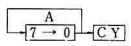
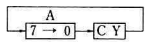
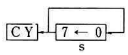
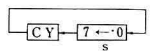
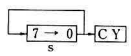
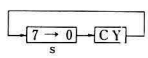
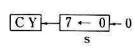
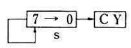
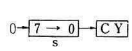
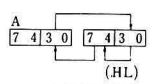
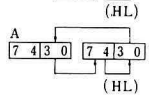
GENERAL PURPOSE ARITHMETIC AND CPU CONTROL GROUPS
TABLE 4.3-5

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

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.

16-BIT ARITHMETIC GROUP
TABLE 4.3-6

Mnémonic	Symbolic Operation	Flags C	Z	P/V	S	N	H	OP-Code 76 543 210	No. of Bytes	No. of M Cycles	No. of T States	Comments
RLC A		‡	•	•	•	0	0	00 000 111	1	1	4	Rotate left circular accumulator
RL A		‡	•	•	•	0	0	00 010 111	1	1	4	Rotate left accumulator
RRC A		‡	•	•	•	0	0	00 001 111	1	1	4	Rotate right circular accumulator
RR A		‡	•	•	•	0	0	00 011 111	1	1	4	Rotate right accumulator
RLC r		‡	‡	P	‡	0	0	11 001 011 00 [000] r	2	2	8	Rotate left circular register r
RLC (HL)		‡	‡	P	‡	0	0	11 001 011 00 [000] 110	2	4	15	
RLC (IX+d)		‡	‡	P	‡	0	0	11 011 101 11 001 011 ← d → 00 [000] 110	4	6	23	
RLC (IY+d)		‡	‡	P	‡	0	0	11 111 101 11 001 011 ← d → 00 [000] 110	4	6	23	
RL s		‡	‡	P	‡	0	0	[010]				Instruction format and states are as shown for RLC, m. To form new OP-code replace [000] of RLC, m with shown code.
RRC s		‡	‡	P	‡	0	0	[001]				
RR s		‡	‡	P	‡	0	0	[011]				
SLA s		‡	‡	P	‡	0	0	[100]				
SRA s		‡	‡	P	‡	0	0	[101]				
SRL s		‡	‡	P	‡	0	0	[111]				
RLD		•	‡	P	‡	0	0	11 101 101 01 101 111	2	5	18	Rotate digit left and right between the accumulator and location (HL).
RRD		•	‡	P	‡	0	0	11 101 101 01 100 111	2	5	18	The content of the upper half of the accumulator is unaffected.

r	Reg.
000	B
001	C
010	D
011	E
100	H
101	L
111	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.

ROTATE AND SHIFT GROUP
TABLE 4.3-7

Mnemonic	Symbolic Operation	Flags						OP-Code	No. of Bytes	No. of M Cycles	No. of T States	Comments
		C	Z	P/V	S	N	H	76 543 210				
BIT b,r	$Z \leftarrow \overline{r}_b$	•	‡	X	X	0	1	11 001 011 01 b r	2	2	8	
BIT b,(HL)	$Z \leftarrow \overline{(HL)}_b$	•	‡	X	X	0	1	11 001 011 01 b 110	2	3	12	
BIT b,(IX+d)	$Z \leftarrow \overline{(IX+d)}_b$	•	‡	X	X	0	1	11 011 101 11 001 011 ← d → 01 b 110	4	5	20	
BIT b,(IY+d)	$Z \leftarrow \overline{(IY+d)}_b$	•	‡	X	X	0	1	11 111 101 11 001 011 ← d → 01 b 110	4	5	20	
SET b,r	$r_b \leftarrow 1$	•	•	•	•	•	•	11 001 011 [11] b r	2	2	8	
SET b,(HL)	$(HL)_b \leftarrow 1$	•	•	•	•	•	•	11 001 011 [11] b 110	2	4	15	
SET b,(IX+d)	$(IX+d)_b \leftarrow 1$	•	•	•	•	•	•	11 011 101 11 001 011 ← d → [11] b 110	4	6	23	
SET b,(IY+d)	$(IY+d)_b \leftarrow 1$	•	•	•	•	•	•	11 111 101 11 001 011 ← d → [11] b 110	4	6	23	
RES b,s	$s_b \leftarrow 0$ s ≡ r,(HL), (IX+d), (IY+d)							[10]				To form new OP-code replace [11] of SET b, m with [10]. Flags and time states for SET instruction.

r	Reg.
000	B
001	C
010	D
011	E
100	H
101	L
111	A

b	Bit Tested
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

Notes: The notation s_b indicates bit b (0 to 7) or location s.

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 SET, RESET AND TEST GROUP
TABLE 4.3-8

Mnemonic	Symbolic Operation	Flags C	Z	P/V	S	N	H	OP-Code 76 543 210	No. of Bytes	No. of M Cycles	No. of T States	Comments
JP nn	PC←nn	•	•	•	•	•	•	11 000 011 ← n → ← n →	3	3	10	
JP cc, nn	If condition cc is true PC← nn, otherwise continue	•	•	•	•	•	•	11 cc 010 ← n → ← n →	3	3	10	cc: 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	•	•	•	•	•	•	00 011 000 ← e-2 →	2	3	12	
JR C, e	If C=0 continue	•	•	•	•	•	•	00 111 000 ← e-2 →	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	•	•	•	•	•	•	00 110 000 ← e-2 →	2	2	7	If condition not met
	If C=0 PC←PC+e								2	3	12	If condition is met
JR Z, e	If Z=0 continue	•	•	•	•	•	•	00 101 000 ← e-2 →	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	•	•	•	•	•	•	00 100 000 ← e-2 →	2	2	7	If condition not met
	If Z=0 PC←PC+e								2	3	12	If condition is met
JP (HL)	PC←HL	•	•	•	•	•	•	11 101 001	1	1	4	
JP (IX)	PC←IX	•	•	•	•	•	•	11 011 101 11 101 001	2	2	8	
JP (IY)	PC←IY	•	•	•	•	•	•	11 111 101 11 101 001	2	2	8	
DJNZ, e	B←B−1 If B=0 continue	•	•	•	•	•	•	00 010 000 ← e-2 →	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 op-code 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.

JUMP GROUP
TABLE 4.3-9

Mnemonic	Symbolic Operation	Flags C	Z	P/V	S	N	H	OP-Code 76 543 210	No. of Bytes	No. of M Cycles	No. of T States	Comments
CALL nn	$(SP-1)\leftarrow PC_H$ $(SP-2)\leftarrow PC_L$ $PC\leftarrow nn$	•	•	•	•	•	•	11 001 101 ← n → ← n →	3	5	17	
CALL cc, nn	If condition cc is false continue, otherwise same as CALL nn	•	•	•	•	•	•	11 cc 100 ← n → ← n →	3 3	3 5	10 17	If cc is false If cc is true
RET	$PC_L\leftarrow(SP)$ $PC_H\leftarrow(SP+1)$	•	•	•	•	•	•	11 001 001	1	3	10	
RET cc	If condition cc is false continue, otherwise same as RET	•	•	•	•	•	•	11 cc 000	1 1	1 3	5 11	If cc is false If cc is true
RETI	Return from interrupt	•	•	•	•	•	•	11 101 101 01 001 101	2	4	14	
RETN	Return from non maskable interrupt	•	•	•	•	•	•	11 101 101 01 000 101	2	4	14	
RST p	$(SP-1)\leftarrow PC_H$ $(SP-2)\leftarrow PC_L$ $PC_H\leftarrow 0$ $PC_L\leftarrow P$	•	•	•	•	•	•	11 t 111	1	3	11	

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

t	P
000	00H
001	08H
010	10H
011	18H
100	20H
101	28H
110	30H
111	38H

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 GROUP
TABLE 4.3-10

Mnemonic	Symbolic Operation	Flags						OP-Code	No. of Bytes	No. of M Cycles	No. of T States	Comments
		C	Z	P/V	S	N	H	76 543 210				
IN A,(n)	A←(n)	•	•	•	•	•	•	11 011 011 ← n →	2	3	10	n to A_0~A_7 Acc to A_8~A_{15}
IN r,(C)	r←(C) If r=110 only the flags will be affected	•	↕ ①	P	↕	0	0	11 101 101 01 r 000	2	3	11	C to A_0~A_7 B to A_8~A_{15}
INI	(HL)←(C) B←B−1 HL←HL+1	•	↕	X	X	1	X	11 101 101 10 100 010	2	4	15	C to A_0~A_7 B to A_8~A_{15}
INIR	(HL)←(C) B←B−1 HL←HL+1 Repeat until B=0	•	1 ①	X	X	1	X	11 101 101 10 110 010	2 2	5 (If B≠0) 4 (If B=0)	20 15	C to A_0~A_7 B to A_8~A_{15}
IND	(HL)←(C) B←B−1 HL←HL−1	•	↕	X	X	1	X	11 101 101 10 101 010	2	4	15	C to A_0~A_7 B to A_8~A_{15}
INDR	(HL)←(C) B←B−1 HL←HL−1 Repeat until B=0	•	1	X	X	1	X	11 101 101 10 111 010	2 2	5 (If B≠0) 4 (If B=0)	20 15	C to A_0~A_7 B to A_8~A_{15}
OUT (n),A	(n)←A	•	•	•	•	•	•	11 010 011	2	3	11	n to A_0~A_7 Acc to A_8~A_{15}
OUT (C),r	(C)←r	•	• ①	•	•	•	•	11 101 101 01 r 001	2	3	12	C to A_0~A_7 B to A_8~A_{15}
OUTI	(C)←(HL) B←B−1 HL←HL+1	•	↕	X	X	1	X	11 101 101 10 100 011	2	4	15	C to A_0~A_7 B to A_8~A_{15}
OTIR	(C)←(HL) B←B−1 HL←HL+1 Repeat until B=0	•	1 ①	X	X	1	X	11 101 101 10 110 011	2 2	5 (If B≠0) 4 (If B=0)	20 15	C to A_0~A_7 B to A_8~A_{15}
OUTD	(C)←(HL) B←B−1 HL←HL−1	•	↕	X	X	1	X	11 101 101 10 101 011	2	4	15	C to A_0~A_7 B to A_8~A_{15}
OTDR	(C)←(HL) B←B−1 HL←HL−1 Repeat until B=0	•	1	X	X	1	X	11 101 101 10 111 011	2 2	5 (If B≠0) 4 (If B=0)	20 15	C to A_0~A_7 B to A_8~A_{15}

Notes: 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.

INPUT AND OUTPUT GROUP
TABLE 4.3-11

4.4 FLAGS

Each of the two Z-80A CPU Flag registers contains six bits of information which are set or reset by various CPU operations. Four of these bits are testable; that is, they are used as conditions for jump, call or return instructions. For example a jump may be desired only if a specific bit in the flag register is set. The four testable flag bits are:

1) **Carry Flag (C)** – This flag is the carry from the highest order bit of the accumulator. For example, the carry flag will be set during an add instruction where a carry from the highest bit of the accumulator is generated. This flag is also set if a borrow is generated during a subtraction instruction. The shift and rotate instructions also affect this bit.

2) **Zero Flag (Z)** – This flag is set if the result of the operation loaded a zero into the accumulator. Otherwise it is reset.

3) **Sign Flag (S)** – This flag is intended to be used with signed numbers and it is set if the result of the operation was negative. Since bit 7 (MSB) represents the sign of the number (A negative number has a 1 in bit 7), this flag stores the state of bit 7 in the accumulator.

4) **Parity/Overflow Flag (P/V)** – This dual purpose flag indicates the parity of the result in the accumulator when logical operations are performed (such as AND A, B) and it represents overflow when signed two's complement arithmetic operations are performed. The Z-80A overflow flag indicates that the two's complement number in the accumulator is in error since it has exceeded the maximum possible (+127) or is less than the minimum possible (–128) number than can be represented in two's complement notation. For example consider adding:

```
+120 =    0111 1000
+105 =    0110 1001
  C = 0   1110 0001  = –95 (wrong) Overflow has occurred
```

Here the result is incorrect. Overflow has occurred and yet there is no carry to indicate an error. For this case the overflow flag would be set. Also consider the addition of two negative numbers:

```
 –5 =    1111 1011
–16 =    1111 0000
  C = 1   1110 1011  = –21 correct
```

Notice that the answer is correct but the carry is set so that this flag can not be used as an overflow indicator. In this case the overflow would not be set.

For logical operations (AND, OR, XOR) this flag is set if the parity of the result is even and it is reset if it is odd.

There are also two non-testable bits in the flag register. Both of these are used for BCD arithmetic. They are:

1) **Half carry (H)** – This is the BCD carry or borrow result from the least significant four bits of operation. When using the DAA (Decimal Adjust Instruction) this flag is used to correct the result of a previous packed decimal add or subtract.

2) **Subtract Flag (N)** – Since the algorithm for correcting BCD operations is different for addition or subtraction, this flag is used to specify what type of instruction was executed last so that the DAA operation will be correct for either addition or subtraction.

The Flag register can be accessed by the programmer and its format is as follows:

S	Z	X	H	X	P/V	N	C

X means flag is indeterminate.

Table 4.4-1 lists how each flag bit is affected by various CPU instructions. In this table a '•' indicates that the instruction does not change the flag, an 'X' means that the flag goes to an indeterminate state, a '0' means that it is reset, a '1' means that it is set and the symbol '↕' indicates that it is set or reset according to the previous discussion. Note that any instruction not appearing in this table does not affect any of the flags.

Table 4.4-1 includes a few special cases that must be described for clarity. Notice that the block search instruction sets the Z flag if the last compare operation indicated a match between the source and the accumulator data. Also, the parity flag is set if the byte counter (register pair BC) is not equal to zero. This same use of the parity flag is made with the block move instructions. Another special case is during block input or output instructions, here the Z flag is used to indicate the state of register B which is used as a byte counter. Notice that when the I/O block transfer is complete, the zero flag will be reset to a zero (i.e. B = 0) while in the case of a block move command the parity flag is reset when the operation is complete. A final case is when the refresh or 1 register is loaded into the accumulator, the interrupt enable flip flop is loaded into the parity flag so that the complete state of the CPU can be saved at any time.

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

The following notation is used in this table:

Symbol	Operation
C	Carry/link flag. C = 1 if the operation produced a carry from the MSB of the operand or result.
Z	Zero flag. Z = 1 if the result of the operation is zero.
S	Sign flag. S = 1 if the MSB of the result is one.
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 into bit 4 of the accumulator.
N	Add/Subtract flag. N = 1 if the previous operation was a subtract.
	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.
↕	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>
m	Any 8-bit location for all the addressing modes allowed for the particular instruction.

SUMMARY OF FLAG OPERATION
TABLE 4.4-1

5.0 INTERRUPT RESPONSE

The purpose of an interrupt is to allow peripheral devices to suspend CPU operation in an orderly manner and force the CPU to start a peripheral service routine. Usually this service routine is involved with the exchange of data, or status and control information, between the CPU and the peripheral. Once the service routine is completed, the CPU returns to the operation from which it was interrupted.

INTERRUPT ENABLE – DISABLE

The Z-80A CPU has two interrupt inputs, a software maskable interrupt and a non maskable interrupt. The non maskable interrupt (NMI) can *not* be disabled by the programmer and it will be accepted whenever a peripheral device requests it. This interrupt is generally reserved for very important functions that must be serviced whenever they occur, such as an impending power failure. The maskable interrupt (INT) can be selectively enable or disabled by the programmer. This allows the programmer to disable the interrupt during periods where his program has timing constraints that do not allow it to be interrupted. In the Z-80A CPU there is an enable flip flop (called IFF) that is set or reset by the programmer using the Enable Interrupt (EI) and Disable Interrupt (DI) instructions. When the IFF is reset, an interrupt can not be accepted by the CPU.

Actually, for purposes that will be subsequently explained, there are two enable flip flops, called IFF_1 and IFF_2.

IFF_1	IFF_2
Actually disables interrupts from being accepted.	Temporary storage location for IFF_1.

The state of IFF_1 is used to actually inhibit interrupts while IFF_2 is used as a temporary storage location for IFF_1. The purpose of storing the IFF_1 will be subsequently explained.

A reset to the CPU will force both IFF_1 and IFF_2 to the reset state so that interrupts are disabled. They can then be enabled by an EI instruction at any time by the programmer. When an EI instruction is executed, any pending interrupt request will not be accepted until after the instruction following EI has been executed. This single instruction delay is necessary for cases when the following instruction is a return instruction and interrupts must not be allowed until the return has been completed. The EI instruction sets both IFF_1 and IFF_2 to the enable state. When an interrupt is accepted by the CPU, both IFF_1 and IFF_2 are automatically reset, inhibiting further interrupts until the programmer wishes to issue a new EI instruction. Note that for all of the previous cases, IFF_1 and IFF_2 are always equal.

The purpose of IFF_2 is to save the status of IFF_1 when a non maskable interrupt occurs. When a non maskable interrupt is accepted, IFF_1 is reset to prevent further interrupts until reenabled by the programmer. Thus, after a non maskable interrupt has been accepted, maskable interrupts are disabled but the previous state of IFF_1 has been saved so that the complete state of the CPU just prior to the non maskable interrupt can be restored at any time. When a Load Register A with Register I (LD A, I) instruction or a Load Register A with Register R (LD A, R) instruction is executed, the state of IFF_2 is copied into the parity flag where it can be tested or stored.

A second method of restoring the status of IFF_1 is thru the execution of a Return From Non Maskable Interrupt (RETN) instruction. Since this instruction indicates that the non maskable interrupt service routine is complete, the contents of IFF_2 are now copied back into IFF_1, so that the status of IFF_1 just prior to the acceptance of the non maskable interrupt will be restored automatically.

Figure 5.0-1 is a summary of the effect of different instructions on the two enable flip flops.

Action	IFF_1	IFF_2	
CPU Reset	0	0	
DI	0	0	
EI	1	1	
LD A, I	•	•	$IFF_2 \rightarrow$ Parity flag
LD A, R	•	•	$IFF_2 \rightarrow$ Parity flag
Accept NMI	0	•	
RETN	IFF_2	•	$IFF_2 \rightarrow IFF_1$

"•" indicates no change

FIGURE 5.0-1
INTERRUPT ENABLE/DISABLE FLIP FLOPS

CPU RESPONSE

Non Maskable

A nonmaskable interrupt will be accepted at all times by the CPU. When this occurs, the CPU ignores the next instruction that it fetches and instead does a restart to location 0066H. Thus, it behaves exactly as if it had received a restart instruction but, it is to a location that is not one of the 8 software restart locations. A restart is merely a call to a specific address in page 0 of memory.

Maskable

The CPU can be programmed to respond to the maskable interrupt in any one of three possible modes.

Mode 0

This mode is identical to the 8080A interrupt response mode. With this mode, the interrupting device can place any instruction on the data bus and the CPU will execute it. Thus, the interrupting device provides the next instruction to be executed instead of the memory. Often this will be a restart instruction since the interrupting device only need supply a single byte instruction. Alternatively, any other instruction such as a 3 byte call to any location in memory could be executed.

The number of clock cycles necessary to execute this instruction is 2 more than the normal number for the instruction. This occurs since the CPU automatically adds 2 wait states to an interrupt response cycle to allow sufficient time to implement an external daisy chain for priority control. Section 3.0 illustrates the detailed timing for an interrupt response. After the application of RESET the CPU will automatically enter interrupt Mode 0.

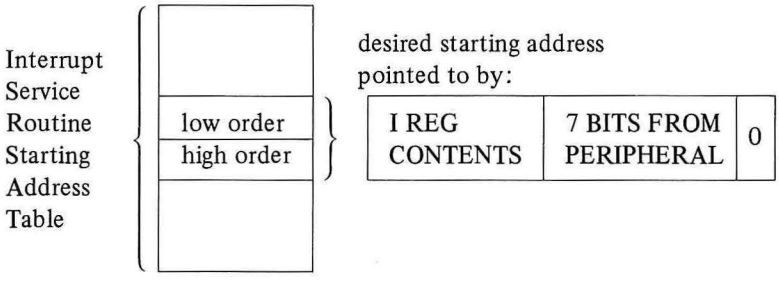
Mode 1

When this mode has been selected by the programmer, the CPU will respond to an interrupt by executing a restart to location 0038H. Thus the response is identical to that for a non maskable interrupt except that the call location is 0338H instead of 0066H. Another difference is that the number of cycles required to complete the restart instruction is 2 more than normal due to the two added wait states.

Mode 2

This mode is the most powerful interrupt response mode. With a single 8 bit byte from the user an indirect call can be made to any memory location.

With this mode the programmer maintains a table of 16 bit starting addresses for every interrupt service routine. This table may be located anywhere in memory. When an interrupt is accepted, a 16 bit pointer must be formed to obtain the desired interrupt service routine starting address from the table. The upper 8 bits of this pointer is formed from the contents of the I register. The I register must have been previously loaded with the desired value by the programmer, i.e. LD I, A. Note that a CPU reset clears the I register so that it is initialized to zero. The lower eight bits of the pointer must be supplied by the interrupting device. Actually, only 7 bits are required from the interrupting device as the least significant bit must be a zero. This is required since the pointer is used to get two adjacent bytes to form a complete 16 bit service routine starting address and the addresses must always start in even locations.



The first byte in the table is the least significant (low order) portion of the address. The programmer must obviously fill this table in with the desired addresses before any interrupts are to be accepted.

Note that this table can be changed at any time by the programmer (if it is stored in Read/Write Memory) to allow different peripherals to be serviced by different service routines.

Once the interrupting devices supplies the lower portion of the pointer, the CPU automatically pushes the program counter onto the stack, obtains the starting address from the table and does a jump to this address. This mode of response requires 19 clock periods to complete (7 to fetch the lower 8 bits from the interrupting device, 6 to save the program counter, and 6 to obtain the jump address.)

Note that the Z-80A peripheral devices all include a daisy chain priority interrupt structure that automatically supplies the programmed vector to the CPU during interrupt acknowledge. Refer to the Z-80A-PIO, manual for details.

A.2 Technical Data of Z80A-PIO

1.0 INTRODUCTION

The Z-80A Parallel I/O (PIO) Circuit is a programmable, two port device which provides a TTL compatible interface between peripheral devices and the Z-80A-CPU. The CPU can configure the Z-80A-PIO to interface with a wide range of peripheral devices with no other external logic required. Typical peripheral devices that are fully compatible with the Z-80A-PIO include most keyboards, paper tape readers and punches, printers, PROM programmers, etc. The Z-80A-PIO utilizes N channel silicon gate depletion load technology and is packaged in a 40 pin DIP. Major features of the Z-80A-PIO include:

- Two independent 8 bit bidirectional peripheral interface ports with 'handshake' data transfer control
- Interrupt driven 'handshake' for fast response
- Any one of four distinct modes of operation may be selected for a port including:

  Byte output

  Byte input

  Byte bidirectional bus (Available on Port A only)

  Bit control mode

  All with interrupt controlled handshake
- Daisy chain priority interrupt logic included to provide for automatic interrupt vectoring without external logic
- Eight outputs are capable of driving Darlington transistors
- All inputs and outputs fully TTL compatible
- Single 5 volt supply and single phase clock are required.

One of the unique features of the Z-80A-PIO that separates it from other interface controllers is that all data transfer between the peripheral device and the CPU is accomplished under total interrupt control. The interrupt logic of the PIO permits full usage of the efficient interrupt capabilities of the Z-80A-CPU during I/O transfers. All logic necessary to implement a fully nested interrupt structure is included in the PIO so that additional circuits are not required. Another unique feature of the PIO is that it can be programmed to interrupt the CPU on the occurrence of specified status conditions in the peripheral device. For example, the PIO can be programmed to interrupt if any specified peripheral alarm conditions should occur. This interrupt capability reduces the amount of time that the processor must spend in polling peripheral status.

2.0 ARCHITECTURE

A block diagram of the Z-80A-PIO is shown in Figure 2.0-1. The internal structure of the Z-80A-PIO consists of a Z-80A-CPU bus interface, internal control logic, Port A I/O logic, Port B I/O logic, and interrupt control logic. The CPU bus interface logic allows the PIO to interface directly to the Z-80A-CPU with no other external logic. However, address decoders and/or line buffers may be required for large systems. The internal control logic synchronizes the CPU data bus to the peripheral device interfaces (Port A and Port B). The two I/O ports (A and B) are virtually identical and are used to interface directly to peripheral devices.

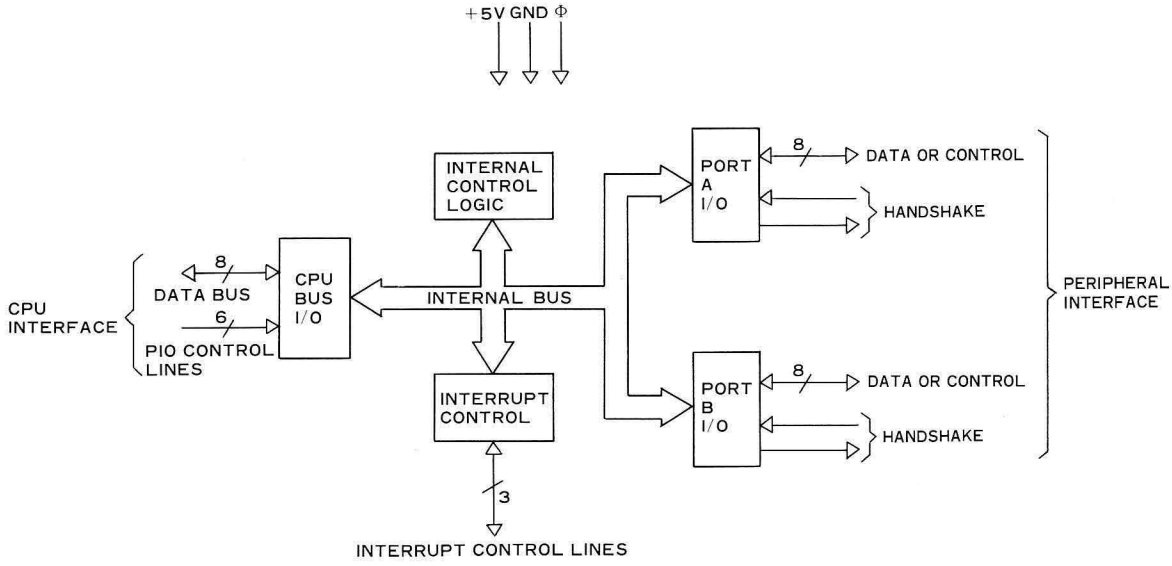


FIGURE 2.0-1
PIO BLOCK DIAGRAM

The Port I/O logic is composed of 6 registers with "handshake" control logic as shown in Figure 1.0-2. The registers include: an 8 bit data input register, an 8 bit data output register, a 2 bit mode control register, an 8 bit mask register, an 8 bit input/output select register, and a 2 bit mask control register.

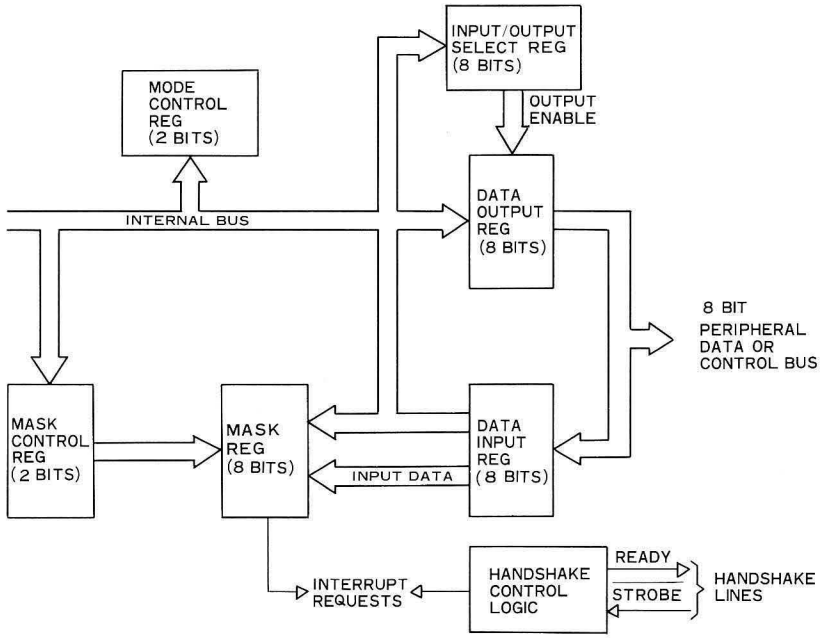


FIGURE 2.0-2
PORT I/O BLOCK DIAGRAM

The 2-bit mode control register is loaded by the CPU to select the desired operating mode (byte output, byte input, byte bidirectional bus, or bit control mode). All data transfer between the peripheral device and the CPU is achieved through the data input and data output registers. Data may be written into the output register by the CPU or read back to the CPU from the input register at any time. The handshake lines associated with each port are used to control the data transfer between the PIO and the peripheral device.

The 8-bit mask register and the 8-bit input/output select register are used only in the bit control mode. In this mode any of the 8 peripheral data or control bus pins can be programmed to be an input or an output as specified by the select register. The mask register is used in this mode in conjunction with a special interrupt feature. This feature allows an interrupt to be generated when any or all of the unmasked pins reach a specified state (either high or low). The 2-bit mask control register specifies the active state desired (high or low) and if the interrupt should be generated when *all* unmasked pins are active (AND CPU status checking of the peripheral by allowing an interrupt to be automatically generated on specific peripheral status conditions. For example, in a system with 3 alarm conditions, an interrupt may be generated if any one occurs or if all three occur.

The interrupt control logic section handles all CPU interrupt protocol for nested priority interrupt structures. The priority of any device is determined by its physical location in a daisy chain configuration. Two lines are provided in each PIO to form this daisy chain. The device closest to the CPU has the highest priority. Within a PIO, Port A interrupts have higher priority than those of Port B. In the byte input, byte output or bidirectional modes, an interrupt can be generated whenever a new byte transfer is requested by the peripheral. In the bit control mode an interrupt can be generated when the peripheral status matches a programmed value. The PIO provides for complete control of nested interrupts. That is, lower priority devices may not interrupt higher priority devices that have not had their interrupt service routine completed by the CPU. Higher priority devices may interrupt the servicing of lower priority devices.

When an interrupt is accepted by the CPU in mode 2, the interrupting device must provide an 8-bit interrupt vector for the CPU. This vector is used to form a pointer to a location in the computer memory where the address of the interrupt service routine is located. The 8-bit vector from the interrupting device forms the least significant 8 bits of the indirect pointer while the I Register in the CPU provides the most significant 8 bits of the pointer. Each port (A and B) has an independent interrupt vector. The least significant bit of the vector is automatically set to a 0 within the PIO since the pointer must point to two adjacent memory locations for a complete 16-bit address.

The PIO decodes the RETI (Return from interrupt) instruction directly from the CPU data bus so that each PIO in the system knows at all times whether it is being serviced by the CPU interrupt service routine without any other communication with the CPU.

3.0 PIN DESCRIPTION

A diagram of the Z-80A-PIO pin configuration is shown in Figure 3.0-1. This section describes the function of each pin.

D_7-D_0 Z-80A-CPU Data Bus (bidirectional, tristate)
This bus is used to transfer all data and commands between the Z-80A-CPU and the Z-80A-PIO. D_0 is the least significant bit of the bus.

B/A Sel Port B or A Select (input, active high)
This pin defines which port will be accessed during a data transfer between the Z-80A-CPU and the Z-80A-PIO. A low level on this pin selects Port A while a high level selects Port B. Often Address bit A_0 from the CPU will be used for this selection function.

C/D Sel Control or Data Select (input, active high)
This pin defines the type of data transfer to be performed between the CPU and the PIO. A high level on this pin during a CPU write to the PIO causes the Z-80A data bus to be interpreted as a *command* for the port selected by the B/A Select line. A low level on this pin means that the Z-80A data bus is being used to transfer data between the CPU and the PIO. Often Address bit A_1 from the CPU will be used for this function.

$\overline{CE}$ Chip Enable (input, active low)
A low level on this pin enables the PIO to accept command or data inputs from the CPU during a write cycle or to transmit data to the CPU during a read cycle. This signal is generally a decode of four I/O port numbers that encompass port A and B, data and control.

Φ 4 MHz System Clock (input)
The Z-80A-PIO uses the standard Z-80A system clock to synchronize certain signals internally. This is a single phase clock.

$\overline{M1}$ Machine Cycle One Signal from CPU (input, active low)
This signal from the CPU is used as a sync pulse to control several internal PIO operations. When $\overline{M1}$ is active and the $\overline{RD}$ signal is active, the Z-80A-CPU is fetching an instruction from memory. Conversely, when $\overline{M1}$ is active and $\overline{IORQ}$ is active, the CPU is acknowledging an interrupt. In addition, the $\overline{M1}$ signal has two other functions within the Z-80A-PIO.

1. $\overline{M1}$ synchronizes the PIO interrupt logic.
2. When $\overline{M1}$ occurs without an active $\overline{RD}$ or $\overline{IORQ}$ signal the PIO logic enters a reset state.

$\overline{IORQ}$ Input/Output Request from Z-80A-CPU (input, active low)
The IORQ signal is used in conjunction with the B/A Select, C/D Select, $\overline{CE}$, and $\overline{RD}$ signals to transfer commands and data between the Z-80A-CPU and the Z-80A-PIO. When $\overline{CE}$, $\overline{RD}$ and $\overline{IORQ}$ are active, the port addressed by B/A will transfer data to the CPU (a read operation). Conversely, when CE and $\overline{IORQ}$ are active but RD is not active, then the port addressed by B/A will be written into from the CPU with either data or control information as specified by the C/D Select signal. Also, if $\overline{IORQ}$ and $\overline{M1}$ are active simultaneously, the CPU is acknowledging an interrupt and the interrupting port will automatically place its interrupt vector on the CPU data bus if it is the highest priority device requesting an interrupt.

$\overline{RD}$ Read Cycle Status from the Z-80A-CPU (input, active low)
If $\overline{RD}$ is active a MEMORY READ or I/O READ operation is in progress. The $\overline{RD}$ signal is used with B/A Select, C/D Select, $\overline{CE}$, and $\overline{IORQ}$ signals to transfer data from the Z-80A-PIO to the Z-80A-CPU.

IEI Interrupt Enable In (input, active high)
This signal is used to form a priority interrupt daisy chain when more than one interrupt driven device is being used. A high level on this pin indicates that no other devices of higher priority are being serviced by a CPU interrupt service routine.

IEO Interrupt Enable Out (output, active high)
The IEO signal is the other signal required to form a daisy chain priority scheme. It is high only if IEI is high and the CPU is not servicing an interrupt from this PIO. Thus this signal blocks lower priority devices from interrupting while a higher priority device is being serviced by its CPU interrupt service routine.

$\overline{INT}$ Interrupt Request (output, open drain, active low)
When INT is active the Z-80A-PIO is requesting an interrupt from the Z-80A-CPU.

A_0 - A_7 Port A Bus (bidirectional, tristate)
This 8 bit bus is used to transfer data and/or status or control information between Port A of the Z-80A-PIO and a peripheral device. A_0 is the least significant bit of the Port A data bus.

$\overline{A\ STB}$ Port A Strobe Pulse from Peripheral Device (input, active low)

1) Output mode: The positive edge of this strobe is issued by the peripheral to acknowledge the receipt of data made available by the PIO.
2) Input mode: The strobe is issued by the peripheral to load data from the peripheral into the Port A input register. Data is loaded into the PIO when this signal is active.
3) Bidirectional mode: When this signal is active, data from the Port A output register is gated onto Port A bidirectional data bus. The positive edge of the strobe acknowledges the receipt of the data.
4) Control mode: The strobe is inhibited internally.

A RDY Register A Ready (output, active high)
The meaning of this signal depends on the mode of operation selected for Port A as follows:

1) Output mode: This signal goes active to indicate that the Port A output register has been loaded and the peripheral data bus is stable and ready for transfer to the peripheral device.
2) Input mode: This signal is active when the Port A input register is empty and is ready to accept data from the peripheral device.
3) Bidirectional mode: This signal is active when data is available in the Port A output register for transfer to the peripheral device. In this mode data is not placed on the Port A data bus unless $\overline{A\ STB}$ is active.
4) Control mode: This signal is disabled and forced to a low state.

B_0 - B_7 Port B Bus (bidirectional, tristate)
This 8 bit bus is used to transfer data and/or status or control information between Port B of the PIO and a peripheral device. The Port B data bus is capable of supplying 1.5ma @ 1.5V to drive Darlington transistors. B_0 is the least significant bit of the bus.

$\overline{B\ STB}$ Port B Strobe Pulse from Peripheral Device (input, active low)
The meaning of this signal is similar to that of $\overline{A\ STB}$ with the following exception:
In the Port A bidirectional mode this signal strobes data from the peripheral device into the Port A input register.

B RDY Register B Ready (output, active high)
The meaning of this signal is similar to that of A Ready with the following exception:
In the Port A bidirectional mode this signal is high when the Port A input register is empty and ready to accept data from the peripheral device.

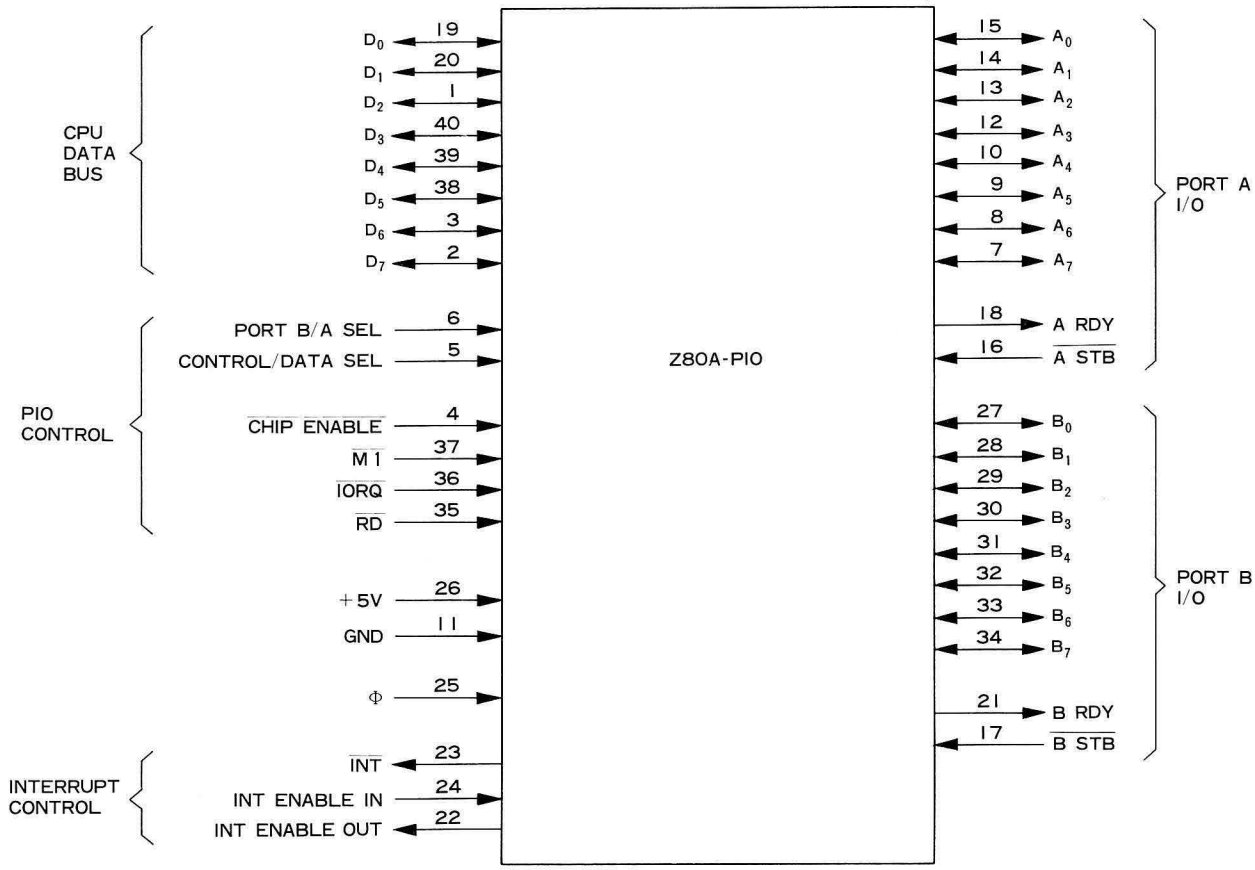


FIGURE 3.0-1
PIO PIN CONFIGURATION

4.0 PROGRAMMING THE PIO

4.1 RESET

The Z-80A-PIO automatically enters a reset state when power is applied. The reset state performs the following functions:

1) Both port mask registers are reset to inhibit all port data bits.
2) Port data bus lines are set to a high impedance state and the Ready "handshake" signals are inactive (low). Mode 1 is automatically selected.
3) The vector address registers are *not* reset.
4) Both port interrupt enable flip flops are reset.
5) Both port output registers are reset.

In addition to the automatic power on reset, the PIO can be reset by applying an $\overline{M1}$ signal without the presence of a $\overline{RD}$ or $\overline{IORQ}$ signal. If no $\overline{RD}$ or $\overline{IORQ}$ is detected during $\overline{M1}$ the $\overline{PIO}$ will enter the reset state immediately after the $\overline{M1}$ signal goes inactive. The purpose of this reset is to allow a single external gate to generate a reset without a power down sequence. This approach was required due to the 40 pin packaging limitation.

Once the PIO has entered the internal reset state it is held there until the PIO receives a control word from the CPU.

4.2 LOADING THE INTERRUPT VECTOR

The PIO has been designed to operate with the Z-80A-CPU using the mode 2 interrupt response. This mode requires that an interrupt vector be supplied by the interrupting device. This vector is used by the CPU to form the address for the interrupt service routine of that port. This vector is placed on the Z-80A data bus during an interrupt acknowledge cycle by the highest priority device requesting service at that time. (Refer to the Z-80A-CPU Technical Manual for details on how an interrupt is serviced by the CPU). The desired interrupt vector is loaded into the PIO by writing a control word to the desired port of the PIO with the following format:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
V_7	V_6	V_5	V_4	V_3	V_2	V_1	0

signifies this control word is an interrupt vector

D_0 is used in this case as a flag bit which when low causes V_7 thru V_1 to be loaded into the vector register. At interrupt acknowledge time, the vector of the interrupting port will appear on the Z-80A data bus exactly as shown in the format above.

4.3 SELECTING AN OPERATING MODE

Port A of the PIO may be operated in any of four distinct modes: Mode 0 (output mode), Mode 1 (input mode), Mode 2 (bidirectional mode), and Mode 3 (control mode). Note that the mode numbers have been selected for mnemonic significance; i.e. 0 = Out, 1 = In, 2 = Bidirectional. Port B can operate in any of these modes except Mode 2.

The mode of operation must be established by writing a control word to the PIO in the following format:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
M_1	M_0	X	X	1	1	1	1

X = unused bit

mode word

signifies mode word to be set

Bits D_7 and D_6 form the binary code for the desired mode according to the following table:

D_7	D_6	Mode
0	0	0 (output)
0	1	1 (input)
1	0	2 (bidirectional)
1	1	3 (control)

Bits D_5 and D_4 are ignored. Bits D_3-D_0 must be set to 1111 to indicate "Set Mode".

Selecting Mode 0 enables any data written to the port output register by the CPU to be output to the port data bus. The contents of the output register may be changed at any time by the CPU simply by writing a new data word to the port. Also, the current contents of the output register may be read back to the Z-80A-CPU at any time through execution of an input instruction.

With Mode 0 active, a data write from the CPU causes the Ready handshake line of that port to go high to notify the peripheral that data is available. This signal remains high until a strobe is received from the peripheral. The rising edge of the strobe generates an interrupt (if it has been enabled) and causes the Ready line to go inactive. This very simple handshake is similar to that used in many peripheral devices.

Selecting Mode 1 puts the port into the input mode. To start handshake operation, the CPU merely performs an input read operation from the port. This activates the Ready line to the peripheral to signify that data should be loaded into the empty input register. The peripheral device then strobes data into the port input register using the strobe line. Again, the rising edge of the strobe causes an interrupt request (if it has been enabled) and deactivates the Ready signal. Data may be strobed into the input register regardless of the state of the Ready signal if care is taken to prevent a data overrun condition.

Mode 2 is a bidirectional data transfer mode which uses all four handshake lines. Therefore only Port A may be used for Mode 2 operation. Mode 2 operation uses the Port A handshake signals for output control and the Port B handshake signals for input control. Thus, both A RDY and B RDY may be active simultaneously. The only operational difference between Mode 0 and the output portion of Mode 2 is that data from the Port A output register is allowed on to the port data bus only when $\overline{\text{A STB}}$ is active in order to achieve bidirectional capability.

Mode 3 operation is intended for status and control applications and does not utilize the handshake signals. When Mode 3 is selected, the next control word sent to the PIO must define which of the port data bus lines are to be inputs and which are outputs. The format of the control word is shown below:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
I/O_7	I/O_6	I/O_5	I/O_4	I/O_3	I/O_2	I/O_1	I/O_0

If any bit is set to a one, then the corresponding data bus line will be used as an input. Conversely, if the bit is reset, the line will be used as an output.

During Mode 3 operation the strobe signal is ignored and the Ready line is held low. Data may be written to a port or read from a port by the Z-80A-CPU at any time during Mode 3 operation. When reading a port, the data returned to the CPU will be composed of input data from port data bus lines assigned as inputs plus port output register data from those lines assigned as outputs.

4.4 SETTING THE INTERRUPT CONTROL WORD

The interrupt control word for each port has the following format:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
Enable Interrupt	AND/ OR	High/ Low	Masks follows	0	1	1	1
	used in Mode 3 only			signifies interrupt control word			

If bit $D_7 = 1$ the interrupt enable flip flop of the port is set and the port may generate an interrupt. If bit $D_7 = 0$ the enable flag is reset and interrupts may not be generated. If an interrupt is pending when the enable flag is set, it will then be enabled onto the CPU interrupt request line. Bits D_6, D_5, and D_4 are used only with Mode 3 operation. However, setting bit D_4 of the interrupt control word during any mode of operation will cause any pending interrupt to be reset. These three bits are used to allow for interrupt operation in Mode 3 when any group of the I/O lines go to certain defined states. Bit D_6 (AND/OR) defines the logical operation to be performed in port monitoring. If bit $D_6 = 1$, an AND function is specified and if $D_6 = 0$, an OR function is specified. For example, if the AND function is specified, all bits must go to a specified state before an interrupt will be generated while the OR function will generate an interrupt if any specified bit goes to the active state.

Bit D_5 defines the active polarity of the port data bus line to be monitored. If bit $D_5 = 1$, the port data lines are monitored for a high state while if $D_5 = 0$ they will be monitored for a low state.

If bit $D_4 = 1$ the next control word sent to the PIO must define a mask as follows:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
MB_7	MB_6	MB_5	MB_4	MB_3	MB_2	MB_1	MB_0

Only those port lines whose mask bit is zero will be monitored for generating an interrupt.

5.0 TIMING

5.1 OUTPUT MODE (MODE 0)

Figure 5.0-1 illustrates the timing associated with Mode 0 operation. An output cycle is always started by the execution of an output instruction by the CPU. A WR* pulse is generated by the PIO during a CPU I/O write operation and is used to latch the data from the CPU data bus into the addressed port's (A or B) output register. The rising edge of the WR* pulse then raises the Ready flag after the next falling edge of Φ to indicate that data is available for the peripheral device. In most systems the rising edge of the Ready signal can be used as a latching signal in the peripheral device if desired. The Ready signal will remain active until: (1) a positive edge is received from the strobe line indicating that the peripheral has taken the data, or (2) if already active, Ready will be forced low 1½ Φ cycles after the leading edge of IORQ if the port's output register is written into. Ready will return high on the first falling edge of Φ after the trailing edge of IORQ. This guarantees that Ready is low when port data is changing. The Ready signal will not go inactive until a falling edge occurs on the clock (Φ) line. The purpose of delaying the negative transition of the Ready signal until after a negative clock transition is that it allows for a very simple generation scheme for the strobe pulse. By merely connecting the Ready line to the Strobe line, a strobe with a duration of one clock period will be generated with no other logic required. The positive edge of the strobe pulse automatically generates an INT request if the interrupt enable flip flop has been set and this device is the highest priority device requesting an interrupt.

If the PIO is not in a reset state, the output register may be loaded before mode 0 is selected. This allows the port output lines to become active in a user defined state.

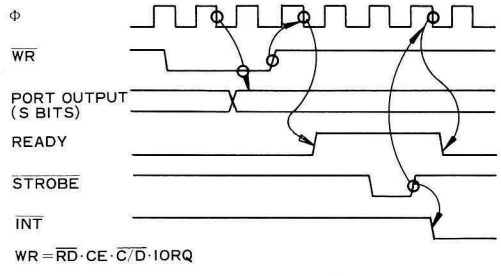


FIGURE 5.0-1
MODE 0 (OUTPUT) TIMING

5.2 INPUT MODE (MODE 1)

Figure 5.0-2 illustrates the timing of an input cycle. The peripheral initiates this cycle using the strobe line after the CPU has performed a data read. A low level on this line loads data into the port input register and the rising edge of the strobe line activates the interrupt request line (INT) if interrupt enable is set and this is the highest priority requesting device. The next falling edge of the clock line (Φ) will then reset the Ready line to an inactive state signifying that the input register is full and further loading must be inhibited until the CPU reads the data. The CPU will, in the course of its interrupt service routine, read the data from the interrupting port. When this occurs, the positive edge of the CPU RD signal will raise the Ready line with the next low-going transition of Φ, indicating that new data can be loaded into the PIO. If already active, Ready will be forced low one and one-half Φ periods following the leading edge of IORQ during a read of a PIO port. If the user strobes data into the PIO only when Ready is high, the forced state of Ready will prevent input register data from changing while the CPU is reading the PIO. Ready will go high again after the trailing edge of the IORQ as previously described.

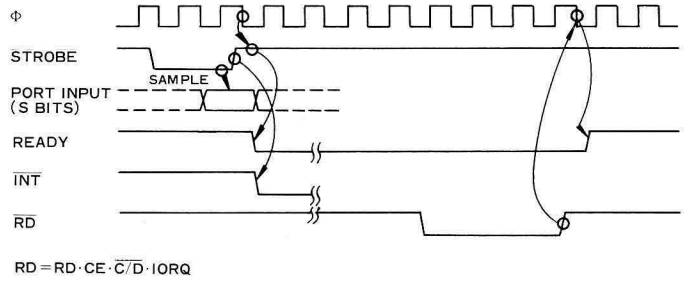


FIGURE 5.0-2
MODE 1 (INPUT) TIMING

5.3 BIDIRECTIONAL MODE (MODE 2)

This mode is merely a combination of Mode 0 and Mode 1 using all four handshake lines. Since it requires all four lines, it is available only on Port A. When this mode is used on Port A, Port B must be set to the Bit Control Mode. The same interrupt vector will be returned for a Mode 3 interrupt on Port B and an input transfer interrupt during Mode 2 operation of Port A. Ambiguity is avoided if Port B is operated in a polled mode and the Port B mask register is set to inhibit all bits.

Figure 5.0-3 illustrates the timing for this mode. It is almost identical to that previously described for Mode 0 and Mode 1 with the Port A handshake lines used for output control and the Port B lines used for input control. The difference between the two modes is that, in Mode 2, data is allowed out onto the bus only when the A strobe is low. The rising edge of this strobe can be used to latch the data into the peripheral since the data will remain stable until after this edge. The input portion of Mode 2 operates identically to Mode 1. Note that both Port A and Port B must have their interrupts enabled to achieve an interrupt driven bidirectional transfer.

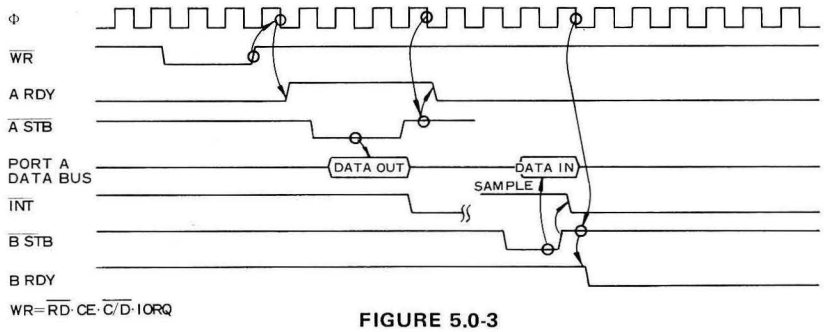


FIGURE 5.0-3
PORT A, MODE 2 (BIDIRECTIONAL) TIMING

The peripheral must not gate data onto a port data bus while A STB is active. Bus contention is avoided if the peripheral uses B STB to gate input data onto the bus. The PIO uses the B STB low level to latch this data. The PIO has been designed with a zero hold time requirement for the data when latching in this mode so that this simple gating structure can be used by the peripheral. That is, the data can be disabled from the bus immediately after the strobe rising edge.

5.4 CONTROL MODE (MODE 3)

The control mode does not utilize the handshake signals and a normal port write or port read can be executed at any time. When writing, the data will be latched into output registers with the same timing as Mode 0. A RDY will be forced low whenever Port A is operated in Mode 3. B RDY will be held low whenever Port B is operated in Mode 3 unless Port A is in Mode 2. In the latter case, the state of B RDY will not be affected.

When reading the PIO, the data returned to the CPU will be composed of output register data from those port data lines assigned as outputs and input register data from those port data lines assigned as inputs. The input register will contain data which was present immediately prior to the falling edge of RD. See Figure 5.0-4.

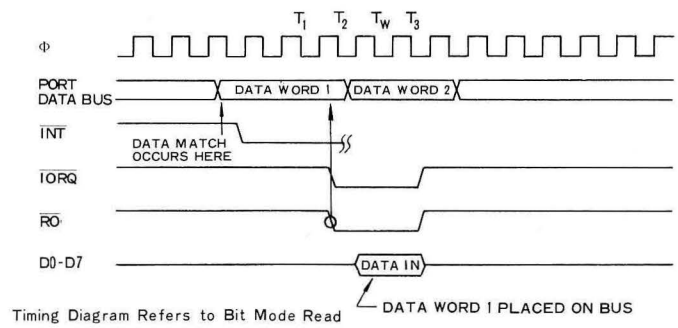


FIGURE 5.0-4

An interrupt will be generated if interrupts from the port are enabled and the data on the port data lines satisfies the logical equation defined by the 8-bit mask and 2-bit mask control registers. Another interrupt will not be generated until a change occurs in the status of the logical equation. A Mode 3 interrupt will be generated only if the result of a Mode 3 logical operation changes from false to true. For example, assume that the Mode 3 logical equation is an "OR" function. An unmasked port data line becomes active and an interrupt is requested. If a second unmasked port data line becomes active concurrently with the first, a new interrupt will not be requested since a change in the result of the Mode 3 logical operation has not occurred.

If the result of a logical operation becomes true immediately prior to or during M1, an interrupt will be requested after the trailing edge of M1.

6.0 INTERRUPT SERVICING

Sometime after an interrupt is requested by the PIO, the CPU will send out an interrupt acknowledge ($\overline{M1}$ and $\overline{IORQ}$). During this time the interrupt logic of the PIO will determine the highest priority port which is requesting an interrupt. (This is simply the device with its Interrupt Enable Input high and its Interrupt Enable Output low). To insure that the daisy chain enable lines stabilize, devices are inhibited from changing their interrupt request status when $\overline{M1}$ is active. The highest priority device places the contents of its interrupt vector register onto the Z80 data bus during interrupt acknowledge.

Figure 6.0-1 illustrates the timing associated with interrupt requests. During $\overline{M1}$ time, no new interrupt requests can be generated. This gives time for the Int Enable signals to ripple through up to four PIO circuits. The PIO with IEI high and IEO low during $\overline{INTA}$ will place the 8-bit interrupt vector of the appropriate port on the data bus at this time.

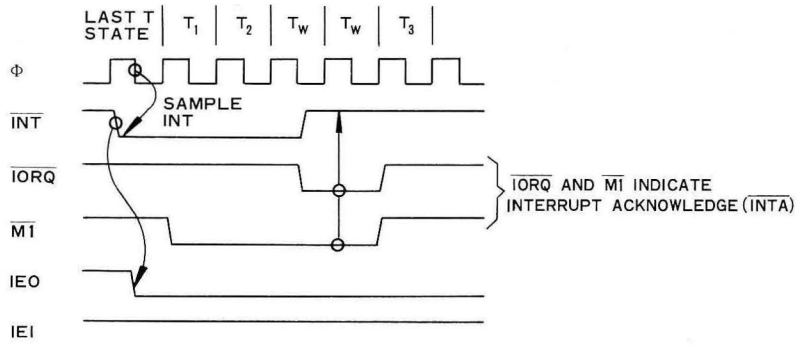


FIGURE 6.0-1
INTERRUPT ACKNOWLEDGE TIMING

If an interrupt requested by the PIO is acknowledged, the requesting port is 'under service'. IEO of this port will remain low until a return from interrupt instruction (RETI) is executed while IEI of the port is high. If an interrupt request is not acknowledged, IEO will be forced high for one $\overline{M1}$ cycle after the PIO decodes the opcode 'ED'. This action guarantees that the two byte RETI instruction is decoded by the proper PIO port. See Figure 6.0-2.

Figure 6.0-3 illustrates a typical nested interrupt sequence that could occur with four ports connected in the daisy chain. In this sequence Port 2A requests and is granted an interrupt. While this port is being serviced, a higher priority port (1B) requests and is granted an interrupt. The service routine for the higher priority port is completed and a RETI instruction is executed to indicate to the port that its routine is complete. At this time the service routine of the lower priority port is completed.

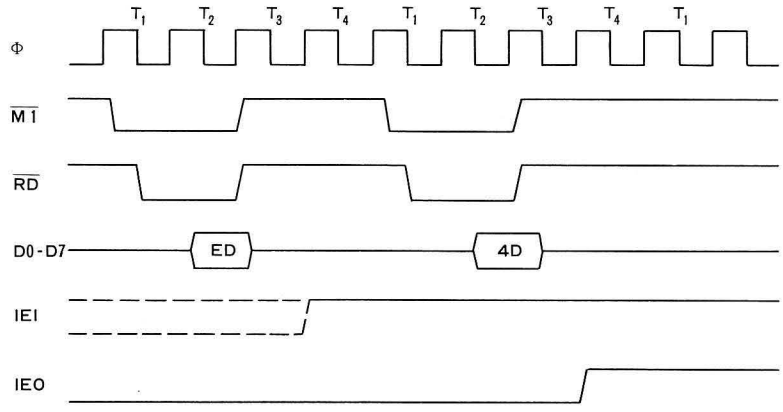


FIGURE 6.0-2
RETURN FROM INTERRUPT CYCLE

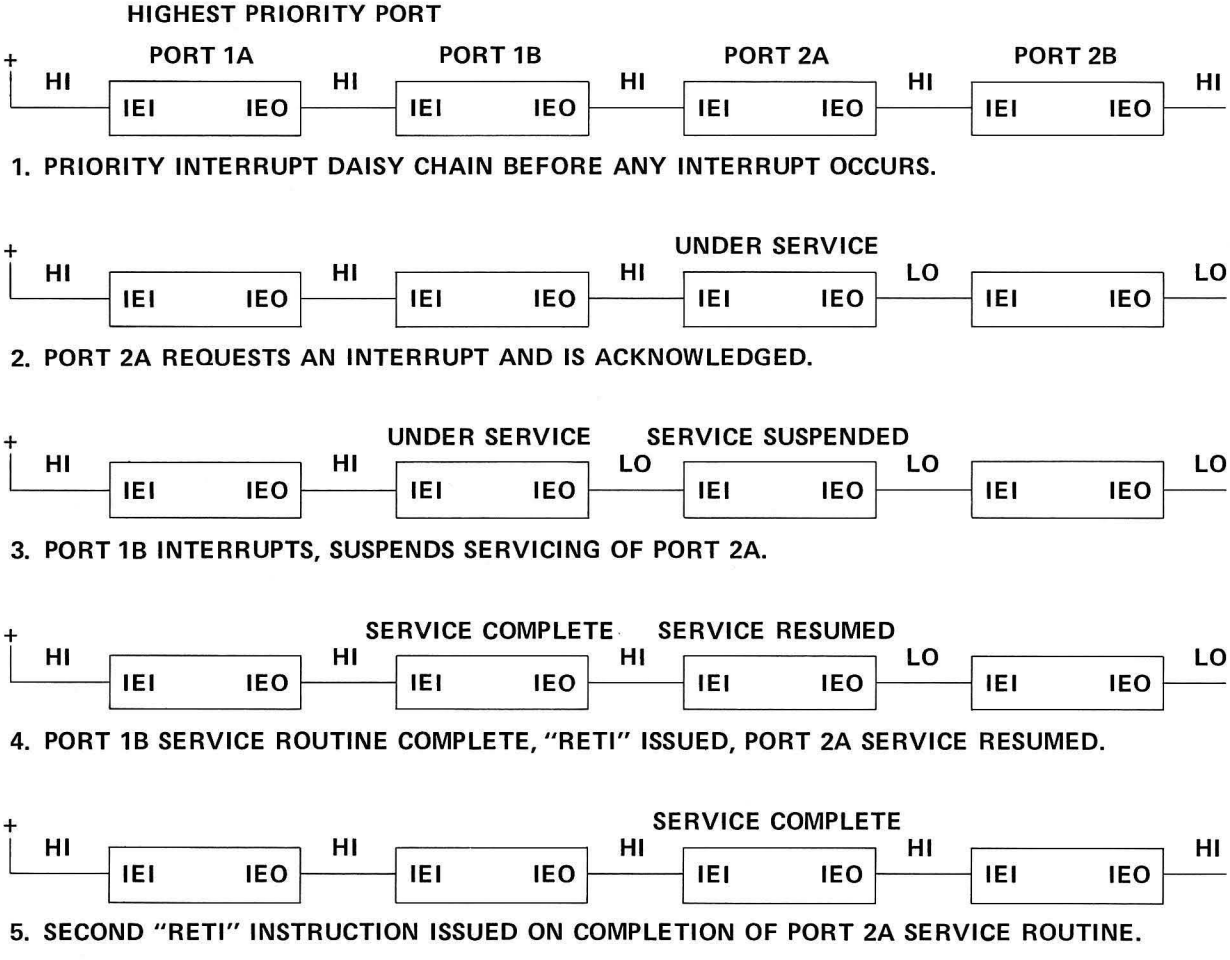


FIGURE 6.0-3
DAISY CHAIN INTERRUPT SERVICING

7.0 APPLICATIONS

7.1 EXTENDING THE INTERRUPT DAISY CHAIN

Without any external logic, a maximum of four Z-80A-PIO devices may be daisy chained into a priority interrupt structure. This limitation is required so that the interrupt enable status (IEO) ripples through the entire chain between the beginning of M1, and the beginning of IORQ during an interrupt acknowledge cycle. Since the interrupt enable status cannot change during M1, the vector address returned to the CPU is assured to be from the highest priority device which requested an interrupt.

If more than four PIO devices must be accommodated, a "look-ahead" structure may be used as shown in Figure 7.0-1. With this technique more than thirty PIO's may be chained together using standard TTL logic.

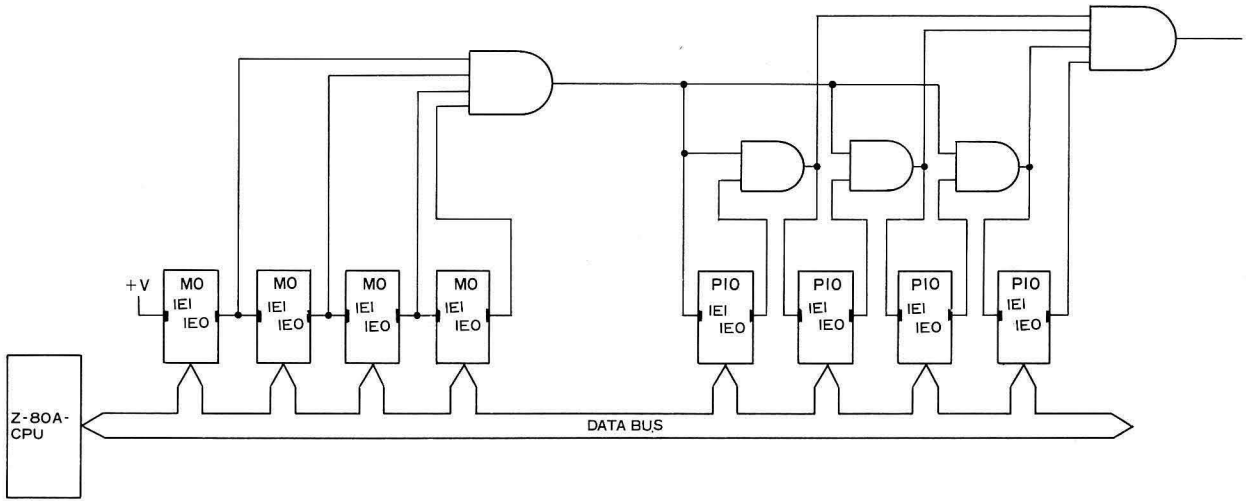


FIGURE 7.0-1
A METHOD OF EXTENDING THE INTERRUPT PRIORITY DAISY CHAIN

7.2 I/O DEVICE INTERFACE

In this example, the Z-80A-PIO is connected to an I/O terminal device which communicates over an 8 bit parallel bidirectional data bus as illustrated in Figure 7.0-2. Mode 2 operation (bidirectional) is selected by sending the following control word to Port A:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	X	X	1	1	1	1
				Mode Control			

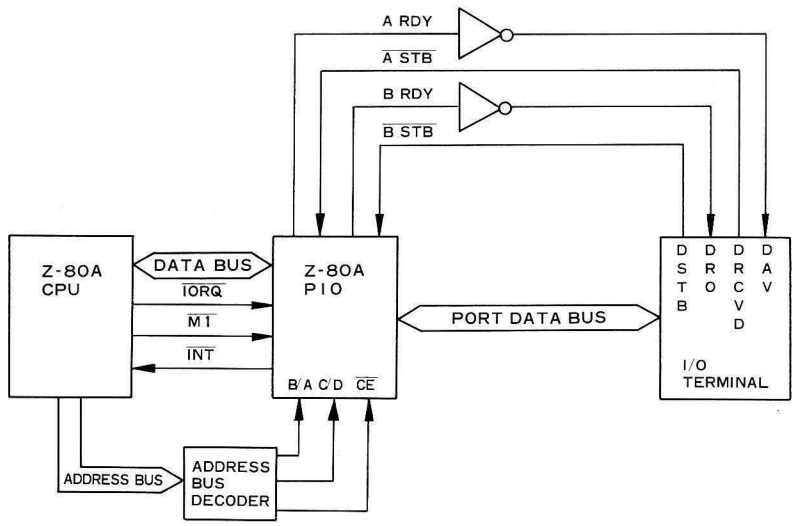


FIGURE 7.0-2
EXAMPLE I/O INTERFACE

Next, the proper interrupt vector is loaded (refer to CPU Manual for details on the operation of the interrupt).

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
V_7	V_6	V_5	V_4	V_3	V_2	V_1	0

Interrupts are then enabled by the rising edge of the first M1 after the interrupt mode word is set unless that M1 defines an interrupt acknowledge cycle. If a mask follows the interrupt mode word, interrupts are enabled by the rising edge of the first M1 following the setting of the mask.

Data can now be transferred between the peripheral and the CPU. The timing for this transfer is as described in Section 5.0.

7.3 CONTROL INTERFACE

A typical control mode application is illustrated in Figure 7.0-3. Suppose an industrial process is to be monitored. The occurrence of any abnormal operating condition is to be reported to a Z-80A-CPU based control system. The process control and status word has the following format:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
Special Test	Turn On Power	Power Failure Alarm	Halt Process-ing	Temp. Alarm	Turn Heaters On	Pressur-ize System	Pressure Alarm

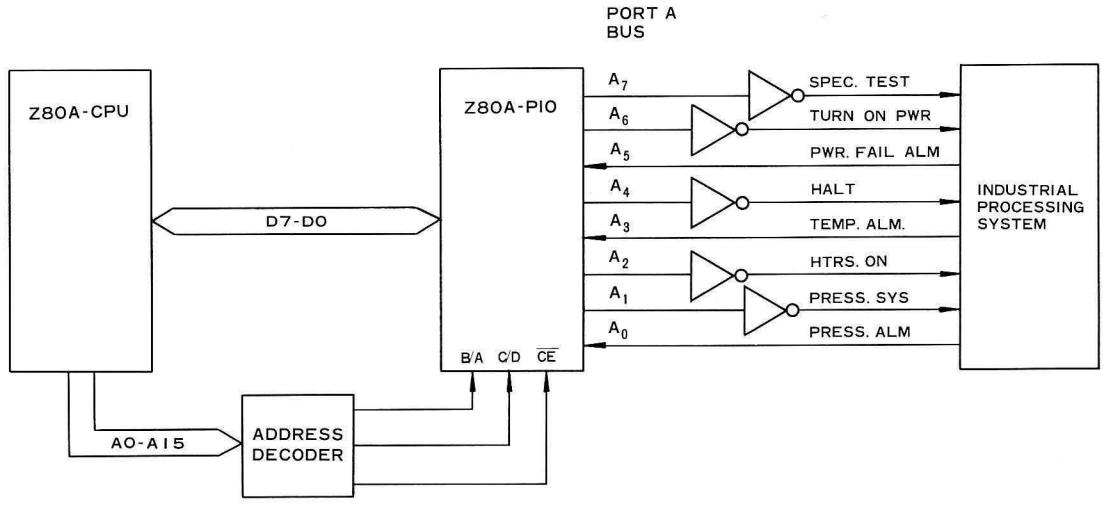


FIGURE 7.0-3
CONTROL MODE APPLICATION

The PIO may be used as follows. First Port A is set for Mode 3 operation by writing the following control word to Port A.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	1	X	X	1	1	1	1

Whenever Mode 3 is selected, the next control word sent to the port must be an I/O select word. In this example we wish to select port data lines A5, A3 and A0 as inputs and so the following control word is written:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	0	1	0	1	0	0	1

Next the desired interrupt vector must be loaded (refer to the CPU manual for details);

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
V_7	V_6	V_5	V_4	V_3	V_2	V_1	0

An interrupt control word is next sent to the port:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	1	1	0	1	1	1

Enable Interrupts | OR Logic | Active High | Mask Follows | Interrupt control

The mask word following the interrupt mode word is:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	1	0	1	0	1	1	0

Selects A5, A3 and A0 to be monitored

Now, if a sensor puts a high level on line A_5, A_3, or A_0, an interrupt request will be generated. The mask word may select any combination of inputs or outputs to cause an interrupt. For example, if the mask word above had been:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	1	0	1	0	1	1	0

then an interrupt request would also occur if bit A_7 (Special Test) of the output register was set.

Assume that the following port assignments are to be used:

$E0_H$ = Port A Data
$E1_H$ = Port B Data
$E2_H$ = Port A Control
$E3_H$ = Port B Control

All port numbers are in hexadecimal notation. This particular assignment of port numbers is convenient since A_0 of the address bus can be used as the Port B/A Select and A_1 of the address bus can be used as the Control/Data Select. The Chip Enable would be the decode of CPU address bits A_7 thru A_2 (1110 00). Note that if only a few peripheral devices are being used, a Chip Enable decode may not be required since a higher order address bit could be used directly.

8.0 PROGRAMMING SUMMARY

8.1 LOAD INTERRUPT VECTOR

V_7	V_6	V_5	V_4	V_3	V_2	V_1	0

8.2 SET MODE

M_1	M_0	X	X	1	1	1	1

M_1	M_0	Mode
0	0	Output
0	1	Input
1	0	Bidirectional
1	1	Bit Control

When selecting Mode 3, the next word must set the I/O Register:

I/O_7	I/O_6	I/O_5	I/O_4	I/O_3	I/O_2	I/O_1	I/O_0

I/O = 1 Sets bit to Input
I/O = 0 Sets bit to Output

8.3 SET INTERRUPT CONTROL

Int Enable	AND/ OR	High/ Low	Mask Follows	0	1	1	1

Used in Mode 3 only

If the "mask follows" bit is high, the next control word written to the port must be the mask:

MB_7	MB_6	MB_5	MB_4	MB_3	MB_2	MB_1	MB_0

MB = 0, Monitor bit
MB = 1, Mask bit from being monitored

Also, the interrupt enable flip flop of a port may be set or reset without modifying the rest of the interrupt control word by using the following command:

Int Enable	X	X	X	0	0	1	1

A. 3 Specifications

1. MZ-80B GENERAL SPECIFICATIONS

CPU	SHARP LH0080A (Z80A-CPU)
Clock	4 MHz
Memory	ROM 2K bytes (initial program loader) ROM 2K bytes (character generator) RAM 32K bytes (dynamic RAM) Can be expanded to 64K bytes. (option)
Display	9" CRT (green display) Character display 8×8 dot matrix 1) Characters; 1000 (40 characters x 25 lines) 2) Characters; 2000 (80 characters x 25 lines) 1), 2): software change-over Graphic display (option) 320 x 200 dots Two graphic areas
Cassette	Standard audio cassette tape Data transfer speed; 1800 bits/sec. Data transfer system; SHARP PWM Automatic or manual control
Sound output	Max. 400 mW (440 Hz)

Key layout	92 keys ASCII standard main keyboard Numeric pad Special function keys Cursor control keys Cassette tape deck control keys
Editing function	Cursor control; up, down, left, right, home, clear Deletion, insertion
Clock function	Built-in
Power supply	Local supply rating voltage
Temperature	Operating temp; 0° to 35°C Storage temp; −15° to 60°C
Humidity	Lower than 80%
Weight	Approx. 16 kg
Dimensions	Width; 450 mm Depth; 520 mm Height; 270 mm

2. CPU BOARD SECTION SPECIFICATIONS

CPU	SHARP LH0080A (Z80A-CPU) 1 pc.
PIO	SHARP LH0081A (Z80A-PIO) 1 pc.
ROM	IPL ROM (2K bytes) 1 pc. Character generator ROM (2K bytes) 1 pc.
RAM	Standard; 16K bits dynamic RAM (SHARP LH4116) 16 pcs. Video RAM (2K bytes) 1 pc.

Programmable counter	8253 1 pc.
Programmable peripheral interface	8255 1 pc.

3. POWER SUPPLY SECTION SPECIFICATIONS

INPUT	Use a power source with the voltage shown on rating name plate.

OUTPUT	5V, −5V, 12V (stabilizing), 12V (non-stabilizing)

4. DISPLAY SECTION SPECIFICATIONS

Size	9"
Vertical horizontal frequency	60Hz (vertical), 15.75kHz (horizontal)
Power source	DC 12V, 1.1A ±10%
Picture tube	E2728B3; 9" 90° deflection explosion proof type Heater; 12V, 75mA
ICs	2 pcs.
Transistors	7 pcs.
Diodes	13 pcs.
Sound output	400mW max. (440 Hz) Speaker 8cm, round dynamic type (32Ω)
Control knobs	Volume, V-Hold, Contrast, H-Hold, Brightness, Focus
Working temperature	−10°C to 50°C

Video output	40Vp-p standard (35Vp-p limit)
Resolution	Horizontal *The pattern of the left in the  center of the picture must be clear.
Non-linearity distortion	Horizontal; ±8% (±14% max.) Vertical; ±8% (±12% max.)
Geometrical distortion	Pincushion dist.; 1% (2% max.) Barrel dist.; 1% (2% max.) Trapezoidal dist.; 1% (2% max.) Parallelogram dist.; 1° (2.5° max.)
High voltage	Zero beam; 11.0kV (10.0kV, min., 12.0kV, max.)
Power supply	DC 12.0V, 1.05A (1.2A max.)
Working range	12V ±10%
Scan size	Horizontal; 10% (15% max.) Vertical; 10% (15% max.)
Horizontal lock-in range	±300Hz (±100Hz limit)
Vertical lock-in range	−12Hz (−6Hz limit)
Audio frequency characteristic	440Hz (0dB) −10dB ±4dB at 100Hz −12dB ±4dB at 10kHz

5. CASSETTE TAPE DECK SECTION SPECIFICATIONS

System	PWM recording
Power source	5V ±5% 12V ±5% (stabilizing) 9.5V~16.5V (non-stabilizing)
Semiconductors	22 transistors 13 ICs 9 diodes
Tape	From C30 to C60
Tape speed	4.75 cm/sec.
Track	2-track monaural

Biasing	DC system
Erasing	DC system
Playback sensitivity	667 μsec. to 333 μsec. (standard)
Working temperature	−10°C to +40°C
Storage temperature	−25°C to +65°C

Specifications and design subject to change without prior notice for product improvement.
In such cases, items mentioned may be partially different from the product.

A.4 Caring for the system

- **Power cable**
  Don't place heavy objects such as desks or chairs, on the power cable and do not damage the covering of the power cable or a severe accident may occur. Be sure to pull the plug (not the cable) when disconnecting the unit from the AC outlet.

- **Line voltage**
  The correct line voltage is shown on rating name plate. Extremely high or low line voltages may cause trouble or result in incorrect operation. Contact your dealer if such trouble occurs.

- **Ventilation**
  Ventilation holes are provided in the cabinet. Never place the unit on a carpet or the like because the holes on the bottom plate of the cabinet will be covered. Place the set in a well ventilated location.

- **Moisture and dust**
  Place the unit in a location which is free from moisture and dust.

- **Temperature**
  Do not expose the unit to direct sunlight and do not place it near heaters to prevent its temperature from rising.

- **Water and other foreign substances**
  Operating the unit when it is wet or contains foreign articles such as clips, pins or other metallic items is dangerous. If water or other liquid enters the unit, immediately pull the power plug and contact your dealer.

- **Shock**
  If the unit is subjected to shock the sensitive electronic parts may be damaged.

- **Trouble**
  If any trouble occurs, stop operating the unit immediately and contact your dealer.

- **Long periods of disuse**
  When the unit is not operated for a long time, be sure to pull the power plug from the AC outlet.

- **Connection of peripheral devices**
  When connecting peripheral devices, use only parts and devices designated by the Sharp Corporation. Use of parts and devices other than those designated (or modification of the set) may result in trouble.

- **Stains**
  Remove stains from the cabinet with a soft cloth moistened with water or detergent. Never use solvents such as benzine, or discoloration will result.

- **Noise**
  When the unit is used in locations where there are high electrical noise levels induced in the AC line, use a line filter to remove the noise. Keep the signal cables away from power cables and other electric equipment.

- **Use and storage**
  Do not use or store the unit with the upper cabinet open, or trouble may occur.

- **Radio wave interference**
  When a radio or TV set is used near the MZ-80B, noise may interfere with broadcast reception. Equipment causing a strong magnetic field may interfere with operation of the MZ-80B.
  Keep such equipment at least 2 to 3 meters away from the MZ-80B.

- **Power switch operation**
  Once the power switch is turned off, wait at least 10 seconds before turning it on again. This ensures correct operation of the microprocessor. Never insert the power plug into an AC outlet with the power switch set to ON.

- **Cassette deck maintenance**
  Dirty cassette deck recording and reproducing heads may result in incorrect data recording the reproduction. Be sure to clean the heads every month. Commercially available cleaning tape is convenient.

- **Discoloration of CRT screen**
  If a certain spot of the CRT screen is lit an external period of time the spot may become discolored. (If it is necessary for certain spot to be lit for an extended time, turn down the brightness control on the display control unit.)