PC-8201A Technical Manual

First edition

Mar 15th 1984

NEC Corporation
Personal Computer development division

IMPORTANT NOTICE

(1) All rights reserved. This manual is protected by copyright. No part of this manual may be reproduced in any form whatsoever without the written permission of the copyright owner.

(2) All efforts have been make to ensure that the contents of this manual are correct; however, should any errors be detected, NEC would greatly appreceate being informed.

(3) NEC can assume no responsibility for errors in this manual or their consequences. The entire risk as to the results of and performance of this manual is assumed by you.

(c) 1984 NEC Corporation
Personal Computer Development Division
Tokyo, Japan

The numeric notation and rules in illustration

In this manual, all numbers are expressed in Decimal unless preceded by special radix prefix ^X, ^O ,^B and ^D. ^X, ^O and ^B represent hexadecimal, Octal and Binary numbers respectively. For instance, ^X1000 is hexadecimal number 1000, which is 4096 in Decimal. Similarly ^O400 is octal number 400, which is 256 in Decimal. ^B10000000 is Binary number 10000000, 128 in Decimal. ^D is used to explicitly tell that is the decimal number.

In the illustration, the upper side of the memory map is near ^XFFFF. So the part drawn below another part is allocated at lower address area.

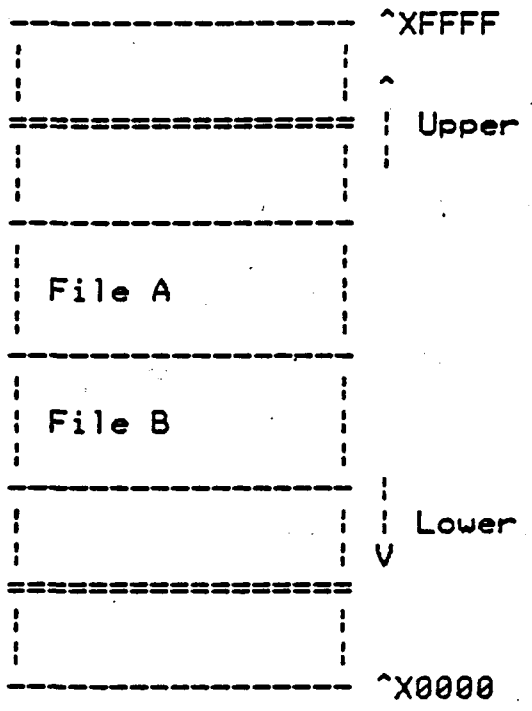


Fig 0.1 The 'File A' is located upper the 'File B'. ('=' means 'skipped'.)

And the address and the pointers written in the illustration points the just above the line.

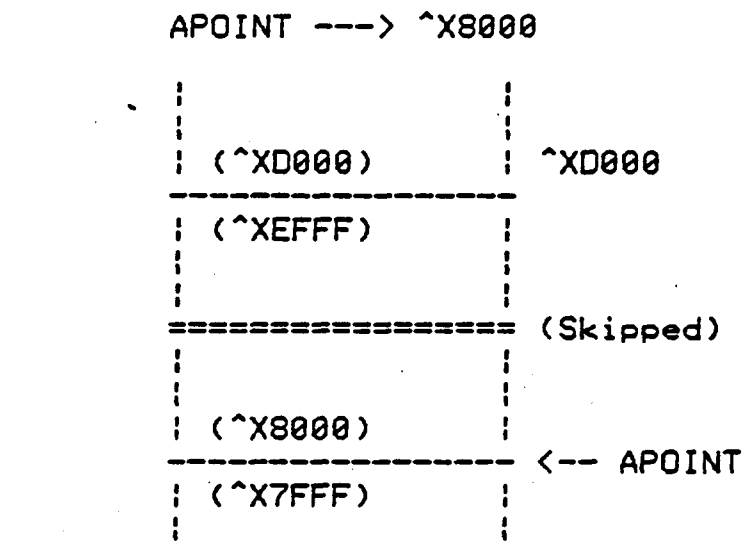


Fig 0.2 Pointer and its contents

Authors:

Chapter 2 -- 4

Mr.Youshiro Hayashi

Software Engineer

Application Technology Department
Personal Computer Sales Promotion Division

Chapter 1,5 -- 8

Mr.Hiroaki Yokoyama

Software Engineer

Development department
Personal Computer Development Division

Chapter 9 -- 14

Mr.Akio Takagi

Software Engineer

Development Department
Personal Computer Development Division

Chapter 15

Mr.Moriharu Seki

Hardware Engineer

Development Department
Personal Computer Development Division

Rewritten and edited by Mr.Hiroaki Yokoyama

CHAPTER 1 INTRODUCTION

CHAPTER 2 MEMORY MAP

2.1 OVERVIEW 13
2.2 BANK SWITCHING ARCHITECTURE 16
2.2.1 Bank Switching Hardware 17
2.2.2 Bank Switching Software 20
2.3 GENERAL MEMORY MAPPING OF INTERNAL SOFTWARE USE . 21
2.4 SAMPLE 25

CHAPTER 3 HOW TO USE 2ND ROM

3.1 CONSIDERATION OF INTERRUPT 28
3.1.1 Power Off Trap (ADDRESS ^X4CFA) 29
3.1.2 Barcode Reader 30
3.1.3 UART 31
3.1.4 Interval Timer (ADDRESS ^X1EBE With Disable Interrupt) 33
3.2 ROM SWAPPING METHOD 35
3.3 THE METHOD TO USE 1ST ROM ENTRY FROM 2ND ROM . . . 36
3.3.1 Sample 37
3.4 SEQUENCES IN THE 2ND ROM 39
3.5 SUMMARY -- IMPORTANT NOTICE 42
3.6 SAMPLE 44

CHAPTER 4 HOW TO USE 2ND/3RD RAM

4.1 READ AND WRITE TO ANOTHER RAM BANK 51
4.1.1 Method 1 [USING 1st ROM] 51
4.1.1.1 GETBNK [^X7EEC] 51
4.1.1.2 PUTBNK [^X7EEB] 52
4.1.2 Method 2 [USING YOUR ORIGINAL CODE] 53

CHAPTER 5 UNDERSTANDING THE RAM FILE CONCEPT

5.1 SUMMARY 60
5.2 WHAT IS RAM FILE? 62
5.2.1 DO File (ASCII File) 62
5.2.2 BA File 66
5.2.3 CO File 74
5.2.4 The Order Of The Files In RAM 75

CHAPTER 6 DIRECTORY STRUCTURE

6.1 DIRECTORY CONFIGURATION PER ENTRY 76

CHAPTER 7 RAM ORGANIZATION

7.1 MEMORY MAP ABOUT RAM FILES 79
7.2 BOOKKEEPING AREA 86
7.2.1 Part I (For RAM File Handling And BASIC) . . . 87
7.2.1.1 FSIDSV 88
7.2.1.2 HIMEM 88
7.2.1.3 TXTTAB 89
7.2.1.4 STKTOP 89
7.2.1.5 DIRTBL 90
7.2.1.6 NULDIR 90
7.2.1.7 SCRDIR 91
7.2.1.8 EDTDIR 91
7.2.1.9 USRDIR 91
7.2.1.10 BOTTOM 92
7.2.1.11 MEMSIZ 92
7.2.1.12 FRETOP 92
7.2.1.13 ASCTAB 93
7.2.1.14 BINTAB 93
7.2.1.15 VARTAB 93
7.2.1.16 ARYTAB 94
7.2.1.17 STREND 94
7.2.1.18 FILTAB 94
7.2.1.19 NULBUF 95
7.2.2 Part II (VRAM Area For LCD) 98
7.2.3 Part III (Bookkeeping Area For BIOS) 98
7.2.4 FCB (file Control Block) 99

CHAPTER 8 RAM FILE HANDLING

8.1 WHAT SHOULD WE DO IN RAM FILE HANDLING 103
8.2 HOW TO MAKE NEW FILE 106
8.2.1 How To Register The New File Name 106
8.2.2 How To Make DO File 106
8.2.3 How To Make A BA File 109
8.2.4 How To Make A CO File 111
8.3 HOW TO DELETE A FILE 113
8.3.1 How To Delete A DO File 113
8.3.2 How To Delete A BA File 114
8.3.3 How To DELETE A CO File 117
8.4 HOW TO APPEND DATA TO DO FILE 122
8.5 HOW TO INSERT DATA TO DO FILE 123
8.6 HOW TO DELETE DATA FROM DO FILE 124
8.7 USEFUL ROUTINES FOR RAM FILE HANDLING IN ROM #0 . 125

8.7.1 MAKHOL 126
8.7.2 LNKFIL 129
8.7.3 MASDEL 132
8.7.4 CHEAD 133
8.8 SAMPLE PROGRAM 134
8.8.1 Make A New DO File (ASCii File) 135
8.8.2 Save Data Into DO File 139
8.8.3 DELETE SOME DATA FROM DO FILE 142
8.8.4 DELETE DO FILE 144
8.8.5 DELETE BA FILE 146
8.8.6 MAKE NEW CO FILE 149
8.8.7 DELETE A CO FILE 152

CHAPTER 9 LCD INTERFACE

9.1 OVER VIEW 154
9.2 CONSTRUCTION OF LCD 154
9.3 I/O PORT RELATED TO LCD 156
9.3.1 BLOCK SELECT --- PPI 81C55 PORT A/B 156
9.3.2 LCD COMMAND SET 157
9.3.2.1 Display ON/OFF. 157
9.3.2.2 Set Address Counter 158
9.3.2.3 Set Starting Page. 160
9.3.2.4 Select Address Counter Mode 162
9.3.3 Read Status --- Read The Status Of Segment Driver. 163
9.3.4 Write/Read Display Data 164
9.4 SOFTWARE FOR LCD 165
9.4.1 How To Initialize The LCD. 165
9.4.1.1 Sample Program For LCD Initialization. 166
9.4.2 How To Write A Character. 168
9.4.2.1 Sample Program Of Writing A Character On The LCD. 169
9.4.3 How To Set/reset A Dot On The LCD. 172
9.4.3.1 Sample Program For SET/RESET Dot. 172
9.4.4 How To Define A Character 177
9.4.4.1 Structure Of Character And How To Define It. . 177
9.4.5 How To Store The Your Own CG 179
9.5 AVAILABLE SYSTEM WORK AREA 180
9.5.1 How To Use The CG In System ROM. 180
9.5.2 VRAM AREA IN SYSTEM Work Area 182
9.5.3 Reverse The Attribute Of The Specified Area. . . 183

CHAPTER 10 KEYBOARD INTERFACE

10.1 THE KEYBOARD MATRIX 184
10.1.1 I/O Port For Keyboard 186
10.1.2 KEYBOARD STROBE ----- PART A/B Of 81C55 186

10.1.3 KEYIN ----- Read Keyboard Data 187
10.1.4 Keyboard Scanning 188
10.2 SOFT WARE FOR KEYBOARD OPERATION. 189
10.2.1 How To Read The Keyboard 189
10.2.1.1 Sample Program Reading Keyboard. 190

CHAPTER 11 CMT INTERFACE

11.1 HARDWARE FOR CMT 193
11.1.1 Writing Operation. 194
11.1.2 Reading Operation. 195
11.1.3 Baud Rate Generation. 196
11.1.4 I/O Port For CMT 197
11.1.4.1 SCP ---- SYSTEM CONTROL PORT 197
11.1.4.2 PPI 81C55 Command Set 197
11.2 SOFTWARE FOR CMT 199
11.2.1 CMT MOTOR CONTROL 199
11.2.2 Baud Rate Generation 200
11.2.3 Write A Data To The CMT 201
11.2.4 Reading A Data From The CMT 203

CHAPTER 12 SERIAL INTERFACE

12.1 HARDWARE OF SERIAL INTERFACE 206
12.1.1 I/O Port 207
12.1.1.1 Channel Select -- (System Control Port) . . . 207
12.1.1.2 UART Mode Control 208
12.1.1.3 UART Status Read 209
12.1.1.4 Set UART Baud Rate (PPI 81C55 Timer Section) . 210
12.1.1.5 UART DATA I/O Port 212
12.2 SOFTWARE DESCRIPTION. 213
12.2.1 How To Initialize Serial Port 213
12.2.1.1 Sample Program ... How To Initialize SERIAC Port 214
12.2.2 SEND A Data To The Serial Port 217
12.2.3 Read A Data From Serial Port. 218
12.3 AVAILABLE SYSTEM AREA. 219

CHAPTER 13 BARCODE READER

13.1 ELECTRIC SPECIFICATION 221
13.2 THEORY OF OPERATION 222

CHAPTER 14 PARALLEL INTERFACE

14.1 HARDWARE SPECIFICATION 223

14.1.1 Physical Interface Of PC-8201A 223
14.1.2 I/O Port For PRINTER Interface. 223
14.1.2.1 Port A ---- Data Out Put Port For Printer. . . 223
14.1.2.2 Port C ---- BUSY,SLCT Signal Read 224
14.1.2.3 SPC(System Control Port) --- STROBE Output Port 224
14.1.3 Basic Theory Of Writing A Data To Centronics . . 225
14.2 SOFTWARE SPECIFICATION 226
14.2.1 How To Write A Byte To The Printer. 226

CHAPTER 15 HARDWARE

15.1 SYSTEM SLOT 229
15.1.1 Assignment Of Signal 229
15.1.2 Explanation Of Pin 232
15.1.2.1 Function Of Signal 232
15.1.3 DC Characteristics 236
15.1.4 AC Characteristics 237
15.2 MEMORY CONTROL CIRCUIT 240
15.3 I/O ADDRESS 245
15.3.1 Detail Information About I/O 247
15.3.1.1 Reserve Area 247
15.3.1.2 System Control 247
15.3.1.3 Bank Control 248
15.3.1.4 Bank Status 249
15.3.1.5 PIO 81C55 Address 250
15.3.1.6 UART Data I/O Port 254
15.3.1.7 UART Control Port 254
15.3.1.8 Keyboard Input 257
15.3.1.9 LCDC Address 257

CHAPTER 1

INTRODUCTION

The portable personal computer, PC-8201A is a unique and practical computer. It has many special capabilities in it. For example, it uses large LCD (Liquied Crystal Display), CMOS (Complementary Metal Oxide Semiconductor) technology and special built-in Software.

The built-in software features are very powerful and useful. But for using PC-8201A fully in particular purpose, new Software written in Machine language might be requested. One of the built-in software, N82-BASIC is very useful to make a small utility, but it's not enough to make a large size utility, for instance, Spread Sheet or new Word Processor.

In order to support the programmers who want to make such a large programs, and to support the programmers who want to manage the hardware features directly, this document describes not only the detail hardware features of PC-8201A, but also the know-how to use these features without any trouble.

The most important thing is 'compatibility'. The built-in Software features keep the promise in using the memory, I/O interface and interrupt functions. The built-in Software checks many critical points at Power-on automatically as far as you don't remove the ROM #0. So if you break the promise, PC-8201A begins 'Cold start' to initialize the all contents in RAM. In this case, the important files and data which you stored are flushed.

The promise for using PC-8201A's features is described in each chapter. Before making a your special program, please

refer to the corresponding chapters. The previous INDEX will help you.

The built-in Software uses a small part of the PC-8201A's special features. With this manual, may you make a super programs for your own purpose!!

CHAPTER 2

MEMORY MAP

2.1 OVERVIEW

The PC-8201A has the following memory capacity. The value specified with 'Max' means the maximum capacity that is greatly expanded by adding RAM/ROM chips or RAM cartridge.

ROM 32K bytes

(Max 64K bytes)

RAM 16K bytes

(Max 96 K : 32k bytes x 3 bank)
2 banks are equipped on Main board of PC-8201A and 1 bank is provided with RAM cartridge.

And PC-8201A has three useful programs in the standard ROM, ROM #0. These programs are (N82-)BASIC , TEXT and TELCOM.

N82-BASIC: Microsoft BASIC, specialized for PC-8201A.

TEXT: Simple and powerful word processor

TELCOM: Communication program with other digital computers via RS-232C.

The simple memory map of PC-8201A is illustrated in

the next figure. This illustration is a one of the standard pattern. Refer to Chapter 15 to understand the hardware expansibility, the detail configuration of memory and how to change the memory configuration.

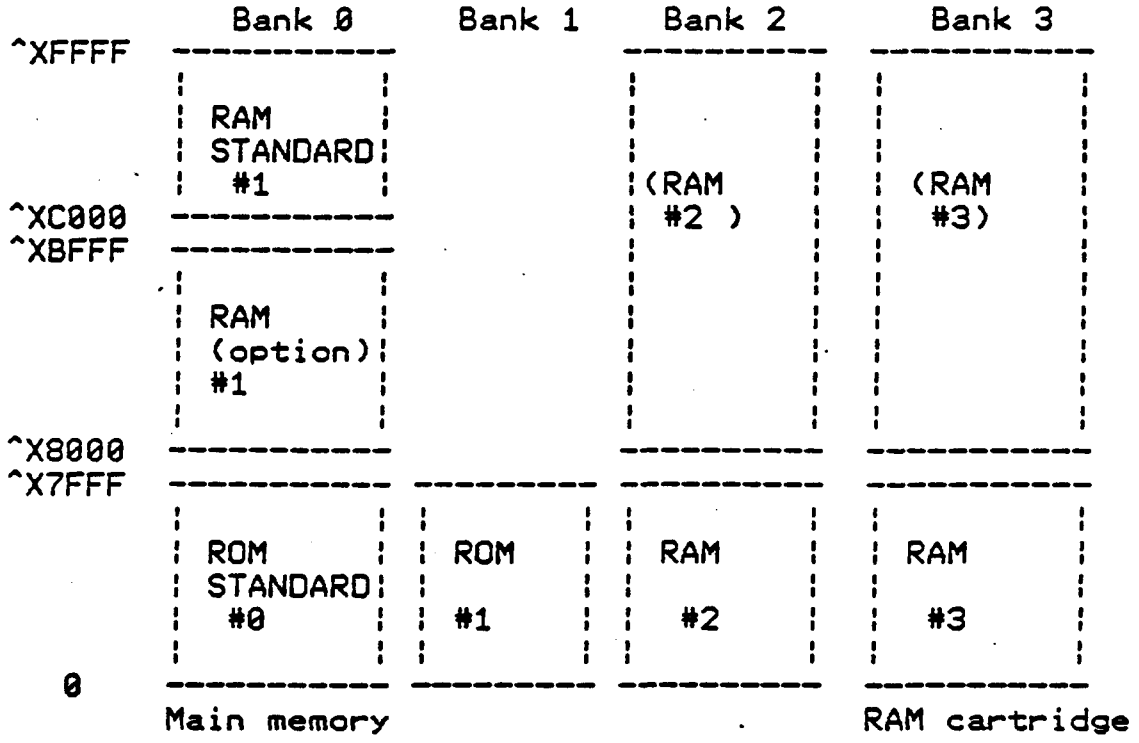


Fig 2.1 P C - 8 2 0 1 A M E M O R Y M A P

The RAM #2 and RAM #3 can be located both low address, from 0 to ^X7FFF, and high address, from ^X8000 to ^XFFFF. This selection can be done by PORT access. Refer to chapter 2.3.

2.2 BANK SWITCHING ARCHITECTURE

The heat of PC-8201A is the Intel 80C85, which is 8 bit processor and whose address bus is 16. Thus, the 80C85 can access 64K of memory at a time. In PC-8201A, however, special memory access function called memory-bank switching is supported. So the 64K barrier in 8-bit microprocessor can be tricked in PC-8201A.

The RAM in the PC-8201A is divided into units referred to as 'BANKS'. One bank can contain a maximum of 32K bytes of memory, while the RAM can be expanded to hold a maximum of three banks. (RAM #1, RAM #2, RAM #3)

The RAM #2 and RAM #3 can be located in two different positions, lower position is from ^X0000 to ^X7FFF and higher position is from ^X8000 to ^XFFFF) And RAM #3 is detachable, because it is provided in RAM cartridge. The bank-switching is executed every 32K bytes. For the sake of this limitation it is impossible to access the half part of RAM #1 and half part of RAM #2 at a time. In other words, you cannot set up the this kind of memory allocation, lower half of RAM #2, from ^X8000 to ^XBFFF, and higher part of RAM #1, from ^XC000 to ^XFFFF as 32K of memory. The variety of memory allocation is illustrated and explained kindly in Chapter 15. The explanation about the software specification in bank-switching is shown in the next section.

The RAM #2 and RAM #3 can be protected by a 'PROTECT SWITCH'. The 'PROTECT SWITCH' for RAM #2 is equipped at the real panel. Refer to the page 1-3 in PC-8201A User's guide. The RAM #3 has it at the side of the cartridge. But unfortunately, RAM #1 has no such a protect function. When you use this protect switch, you cannot use that RAM bank in usual way, for instance, BASIC. Because, PC-8201A uses the highest RAM area, from ^XF380 to ^XFFFF to save the current status of PC-8201A every time.

All RAM chips consists of CMOS and are back-uped by battery. All data and program files stored in RAM will be kept, even if the power switch is turned off. If you make a special utility for 2nd ROM or special RAM configuration, you have to consider about this Power-down sequence. Refer to chapter 3 to understand the Power-off trap in ROM #0.

2.2.1 Bank Switching Hardware

The 'bank-switching' is performed by OUT instruction. The OUT instruction outputs 8 bit data to the I/O port. The port address and that bit assign of the 8 bit data is shown below.

PORT ADDRESS ^XA1 (OUT)
Bank control

MSB	7	6	5	4	3	2	1	0

Bit 7 --- not used

Bit 6 --- not used

Bit 5 --- not used

Bit 4 --- not used

Bit 3 --- High address
(^X8000 - ^XFFFF) selection #2

Bit 2 --- High address
(^X8000 - ^XFFFF) selection #1

Bit 1 --- Low address
(^X0000 - ^X7FFF) selection #2

Bit 0 --- Low address
(^X0000 - ^X7FFF) selection #1

High address #2	High address #1	
0	0	Bank #0 (RAM #1)
0	1	not used
1	0	Bank #2 (RAM #2)
1	1	Bank #3 (RAM #3)

Low address #2	Low address #1	
0	0	Bank #0 (ROM #0)
0	1	Bank #1 (ROM #1)
1	0	Bank #2 (RAM #2)
1	1	Bank #3 (RAM #3)

The current status of the memory, the status of bank-switching, can be examined by IN instruction. The IN instruction reads a 8 bit data from the specified I/O port. See next figure about the Port address and bit assignment of the data.

PORT ADDRESS ^XA0H (IN)
Bank status

MSB	7	6	5	4	3	2	1	0

Bit 7 --- Serial interface status #2
Bit 6 --- Serial interface status #1
Bit 5 --- Not used
Bit 4 --- Not used
Bit 3 --- High address (^X8000 - ^XFFFF) status #2
Bit 2 --- High address (^X8000 - ^XFFFF) status #1
Bit 1 --- Low address (^X0000 - ^X7FFF) status #2
Bit 0 --- Low address (^X0000 - ^X7FFF) status #1

Serial I/F #2	Serial I/F #1	
0	0	Not used
0	1	SIO port
1	0	Floppy disk port
1	1	RS-232C port

High address #2	High address #1	
0	0	Bank #0 (RAM #1)
0	1	Not used
1	0	Bank #2 (RAM #2)
1	1	Bank #3 (RAM #3)

Low address #2	Low address #1	
0	0	Bank #0 (ROM #0)
0	1	Bank #1 (ROM #1)
1	0	Bank #2 (RAM #2)
1	1	Bank #3 (RAM #3)

Refer to Chapter 12 about Serial Interface.

2.2.2 Bank Switching Software

The bank-switching capability is used in Menu mode. The 'BANK' command, arranged in Function key 10 (Shift + F.5) uses this function. This function falls into the Bank handler routine, CHGBNK, ^X7EAB. The CHGBNK checks the current bank status, tests whether the bank really exists, save the new bank # in BANK (^XF3DB), changes the bank status and jumps to the address 0. Jumping to address 0 causes 'COLD START' if the bank has not ever used or the flag named FSIDSV has a wrong value. (Refer to the section 3.2 Bookkeeping area.) Otherwise, Jumping 0 does 'WARM START'.

In order to test the existence of the another bank, CHGBNK reads the contents of the address, ^XE000, in that destination bank, modifies that value, restores it, and re-reads it. If that bank were really in exist, the value read first and the value re-read last are not identical.

The reason why CHGBNK jumps into the address 0 is, you might already notice, to set up the bookkeeping area. As described in Chapter 7, all standard programs and operating system uses this area every time to keep the current status. This area contains very important pointers, flags and interrupt routines. So without setting up this area, that bank cannot be handled with ROM #0 correctly.

If you use a bank only with your special application program, which does not use the pointers on interrupt routines in the bookkeeping area, you might think that you need not care about the bookkeeping area. But please do not forget that 'SHIFT+F.5' in menu level can change the bank any time. I recommend that you will keep the current rules about Bank-switching in ROM #0, and set up the bookkeeping area.

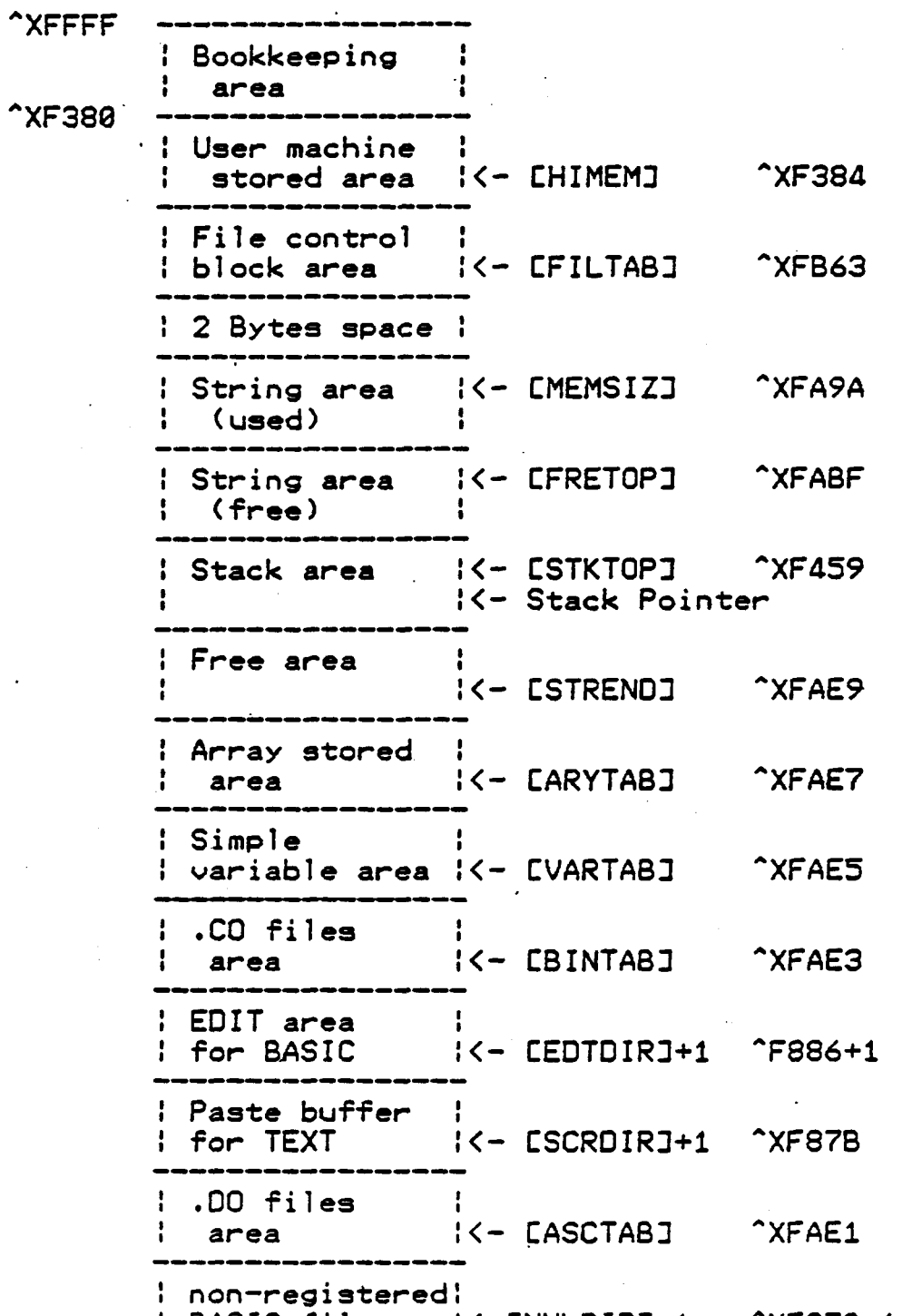
Refer chapter 4 'HOW TO USE 2ND/3RD RAM' to get more detail documents.

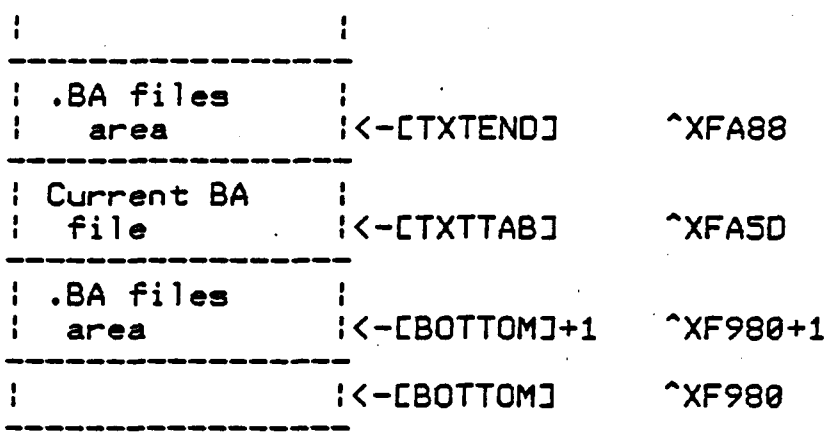
2.3 GENERAL MEMORY MAPPING OF INTERNAL SOFTWARE USE

You know that the ROM #0 addressed from 0 to ^X7FFF is used for standard programs and operating systems. (Sometimes, 'standard programs' represents BASIC, TEXT and TELCOM especially. 'Operating system' also represents 'Menu'. But there is no explicit border line between the 'standard programs' and 'operating system'. But I do sometimes use these words to explain the concept of the PC-8201A's built-in software.) Also, Some parts of the RAM memory area are reserved and used by that standard programs and operating system. The memory map about the RAM area is figured at next page. The each part of the reserved area is pointed by pointers in the 'book-keeping area', located at the highest part of the RAM memory, from ^XF380 to ^XFFFF. And the following 2 items are included in the book-keeping area, too.

Interrupt routine
System work area

Fig 2.2 PC - 8201A RAM AREA MEMORY MAP




.BA files area
<-[TXTEND]
^XFA88
Current BA file
<-[TXTTAB]
^XFA5D
.BA files area
<-[BOTTOM]+1
^XF980+1
<-[BOTTOM]
^XF980

Brief explanation about pointers which appear at the previous page.

[BOTTOM] Bottom address of RAM
[TXTTAB] Beginning of the current BASIC program
[TXTEND] End of the current BASIC program
[NULDIR] Non-registered BASIC program
[ASCTAB] Lowest address of ASCII files
[SCRDIR] SCRAP file
[BINTAB] Lowest address of binary files
[VARTAB] Simple variable space
[ARYTAB] Start of array table
[STREND] End of Array table
[STKTOP] Top of stack space
[FRETOP] Top of string free space
[MEMSIZ] Highest location in memory
[HIMEM] Highest memory available to BASIC
(The same as CLEAR's 2nd parameter)

rf. Chapter 5 'UNDERSTANDING THE RAM FILE CONCEPT', 'DIRECTORY STRUCTURE' and 'RAM ORGANIZATION'. In those chapters, the concept of the files and detail explanation about the pointers are described.

2.4 SAMPLE

```
        TITLE   Bank switching program


;
;       This sample will only change the bank of
;       RAM addressed from ^X8000 to ^XFFFF.
;       You had better check that the bank which
;       you want to switch really exists. And you
;       should save the next bank # at the
;       bookkeeping area, BANK.
;
;       Entry   None
;       Exit    None
;               Bank will be changed
;
;       Bank rotation #1 -> #2 -> #3 -> #1 -> ...
;

; <<< SYSTEM labels >>>
SYSTEM  EQU     ^X0000          ; Reset address
CONTRL  EQU     ^X0A1           ; Bank control port
STATUS  EQU     ^X0A0           ; Bank status port

; <<< Bank switching program >>>

        ORG     ^X0100          ; This program must be
                                ; stored between ^X0000
                                ; and ^X7FFF

CHECK:  DI                      ; Disable interrupt
        IN      STATUS          ; Read current bank status
        MOV     B,A             ; Save current bank status
        ANI     ^B00001100      ; Pick up high bank status
                                ;  only
NEXTB:
        ADI     ^B00000100      ; Set next bank data
        CPI     ^B00000100      ; This pattern was not used!
        JZ      NEXTB           ; Set up next bank data
                                ; for lap around

        MOV     C,A             ; Save new bank data
        MOV     A,B             ; Remember old bank status
        ANI     ^B11110011      ; Do not change bit data
                                ;  without RAM bank data
        ORA     C               ; Set new RAM bank
        OUT     CONTRL          ; Select bank
```

```
        EI                      ; Enable interrupt
        JMP     SYSTEM          ; We must update book
                                ;  keeping area.
                                ; Jump ^X0000 is the
                                ;  best way.

        END
```

CHAPTER 3

HOW TO USE 2ND ROM

When you want to make some programs stored in 2nd ROM, there are a lot of matters should be attended and stored in the 2nd ROM. The matters are interrupt jump tables and power on/power off sequences. You have to implement these tables and sequences in order to process the ROM bank switching smoothly. Otherwise, PC-8201A will run away on switching the ROM bank. First half sections describe the interrupt functions and power sequence.

And you have to know the rules to handle the files and data in RAM, too. If you will use the routines in ROM #0 to handle the RAM, you need not to care about the detail rules. (You can get the information about the RAM file handling routines in ROM #0 at the Chapter 8 and another technical manual that has already been available by NEC HE in Chicago. Please request it if you have not gotten it yet.) The last half of this chapter describes how to use the routines in ROM #0 from 2nd ROM, ROM #1. (Hereafter ROM #1 sometimes represents 2nd ROM.)

If you want to make I/O control routines and store them in 2nd ROM, you have to understand Chapter 9 to 14. If you utilize the ROM #0's I/O routines, the last half of this chapter and another manual will help you.

3.1 CONSIDERATION OF INTERRUPT

Basically, PC-8201A has some interrupt service routines in that system. The main purpose of interrupts are smooth processing in Power off trap, reading data from Bar-code reader, communicating through UART(RS-232C) and using Interval timer.

The interrupt table is located in the zero page area.

POWER OFF TRAP	NMI	^X0024
BARCODE READER	RST 5.5	^X002C
UART	RST 6.5	^X0034
INTERVAL TIMER	RST 7.5	^X003C

The Interval timer interrupt has the highest priority, and UART has the second one. The lowest interrupt is used for Barcode reader. The reason why the internal timer has the highest priority is to scan the key and to count the auto-power off counter for saving the battery power. PC-8201A has the 'Auto-Power Off' function'. Usually, this function is executed after 10 minutes has past since last key stroke was detected. (This interval can be set by the 'POWER' command in BASIC. Refer 'PC-8201A Reference Manual'.) The interval time is used to count this period.

The interrupt hook table is located from ^XF386 to ^XF394. And that table is constructed in the following fig.

Interrupt hook table in RAM area

^XF386	POWER ON SEQUENCE
^XF389	BARCODE READER INPUT SEQUENCE
^XF38C	UART INPUT SEQUENCE
^XF38F	TIMER SEQUENCE and KEY SCANNING SEQUENCE
^XF392	POWER FAILURE SEQUENCE

3.1.1 Power Off Trap (ADDRESS ^X4CFA)

This interrupt is Non maskable. When power switch is turned off, this interrupt occurs. The following sequence is the algorism of this interrupt.

1: Disable the interrupt
2: Call hook table
3: Reset Key wait counter
4: Cancel Time counter
5: Out a data to the Auto power off port
6: HLT

The detail bit assignment of the auto power off port is following.

PORT ADDRESS ^XBA (OUT)
81C55 port B

MSB	7	6	5	4	3	2	1	0

Bit 7 --- RTS output
Bit 6 --- DTR output
Bit 5 --- BELL
0:Ring bell
1:Stop bell

Bit 4 --- Auto power off
0:Off
1:On

Bit 3 --- DCD/RD select
Bit 2 --- Melody control
0:On
1:Off

Bit 1 --- LCD chip select #1
Bit 0 --- LCD chip select #0

rf.Chapter 9 to 15 about more detail information of this port.

3.1.2 Barcode Reader

(ADDRESS ^XF389 with Disable interrupt)

This interrupt is using RST 5.5. If you do not use barcode reader program, this interrupt should do 'RETURN' soon.

3.1.3 UART

(ADDRESS ^X6E00 with Disable interrupt)

This interrupt is using RST 6.5. This interrupt is caused by UART. (Serial communication device 6402) This interrupt occurs when the data in 6402 receive buffer is available.

The algorism of this interrupt is shown below.

1: Disable the interrupt
2: Call hook table
3: Read data from 6402
4: Read error status from 6402
5: Xon/Xoff control check
6: SI/SO control check
7: Return to previous process

PORT ADDRESS ^XD8 (OUT)
UART control port

MSB	7	6	5	4	3	2	1	0

Bit 7 --- Not used
Bit 6 --- Not used
Bit 5 --- Not used
Bit 4 --- Character length select #2
Bit 3 --- Character length select #1
Bit 2 --- Parity inhibit
0:Parity generation Check
1:Parity generation check,
Inhibit

Bit 1 --- Even parity enable
0:Odd parity
1:Even parity

Bit 0 --- Stop bit select
0:Stop bit 1 bit
1:Stop bit 1.5 bit
in case of DATA Length is 5
1:Stop bit 2 bit
in case of DATA Length
is not 5

PORT ADDRESS ^XC8 (OUT)

UART data I/O port

MSB	7	6	5	4	3	2	1	0

Bit 7 --- Data #7
Bit 6 --- Data #6
Bit 5 --- Data #5
Bit 4 --- Data #4
Bit 3 --- Data #3
Bit 2 --- Data #2
Bit 1 --- Data #1
Bit 0 --- Data #0

rf. Chapter 12 and 15 about more detail information about UART.

3.1.4 Interval Timer (ADDRESS ^X1EBE With Disable Interrupt)

This interrupt is using RST 7.5. This is the interrupt from interval timer. (Timer device 1990) This interrupt is also used for the key scanning.

In the system's initialization, the interval timer which is controlled by 1990, is set up as 4m second mode. The port for 1990 is illustrated below.

PORT ADDRESS ^XB9 (OUT)

Calendar clock (1990) control port

MSB	7	6	5	4	3	2	1	0

Bit 7	---	Not used
Bit 6	---	Not used
Bit 5	---	Not used
Bit 4	---	Data output
Bit 3	---	Shift clock
Bit 2	---	Command output #2
Bit 1	---	Command output #1
Bit 0	---	Command output #0

Command #2	Command #1	Command #0	
1	0	0	timing 64Hz
1	0	1	Timing 256Hz
1	1	0	Timing 2048Hz
1	1	1	TEST mode

In the initialization routine, the command is set up as ^X05. It means 4m second interval.

rf. Chapter 15 for more information about 1990

The following step is the algorism about interval timer sequence.

1: Disable the interrupt
2: Call hook table
3: Mask RST 7.5,RST 5.5
4: Reverse cursor character for cursor blink
5: Key matrix scanning
6: Return to the interrupted process

3.2 ROM SWAPPING METHOD

When you would like to use 2nd ROM, you must write the following information into the 2nd ROM's special reserved area. The special reserved area is located from ^X0000 to ^X0047. These area will be used for 2nd ROM starting jump instruction and ID code, and the file name of 2nd ROM. This name is displayed like a one of the RAM files on Menu screen by 1st ROM, ROM #0. The following figure is the explanation about 2nd ROM special reserved area.

```
ADDRESS          CODE
^X0000 JMP       START    ; 2nd ROM start address
^X0003 ....
^X0024           RET      ; Non maskable interrupt
^X002C           RET      ; Barcode reader interrupt
^X0034           RET      ; UART interrupt
^X003C           RET      ; Interval timer interrupt
^X003F                    ; Reserved for RST interrupt
^X0040           DB       'A'
^X0041           DB       'B'      ; ID code for 2nd ROM
^X0042           DB       '2NDROM' ; File name which displayed in
                                    ; the menu

^X0048 START:                       ; 2nd ROM code
```

S P E C I A L R E S A R V E D A D D R E S S

If these data are implemented correctly, the name will appears on the 1st ROM's menu screen. So it's easy to switch the ROM and execute the program in it. When you want to start the programs in 2nd ROM from the Menu mode of ROM #0, move the cursor to 2nd ROM's file name on the screen. Then please press return key. The system will fall into the 2nd ROM program.

3.3 THE METHOD TO USE 1ST ROM ENTRY FROM 2ND ROM

If you want to use the routines in 1st ROM from 2nd ROM, at the first, you have to create a special routine in the higher memory location of RAM (^X8000-^XFFFF) and use it. That routine switches the ROM bank with using bank switching method, and calls the routine in 1st ROM. It is very important for you that the interrupts must be disabled before you change the ROM banks. And in addition, as the following sections will tell you, you have to change the hook table for Power down interrupt that was changed by 2nd ROM to restart the current process in 2nd ROM program at next power-on. With this hook table for 2nd ROM, the power down in ROM #0 will cause the fatal error. Power-off interrupt can not be prohibited. And you have to consider about the contents of the routine which you will call. The reason is that some routines in the 1st ROM routine may enable the interrupts in some parts of their code even if you disable the interrupts just before switching the ROM banks to call 1st ROM entry. Therefore you had better change the all hook tables in the current book keeping area. I suggest that all hook table should be replaced with previout contents which were stored by 1st ROM, just before calling ROM bank-switching routine ,and restored just after coming back from 1st ROM.

The following program is the sample which uses 1st ROM entry points from 2nd ROM.

3.3.1 Sample

```
;       TITLE   Using 1st ROM entry from 2nd ROM
;
;
;       This sample will enable to use 1st ROM entry from
;       2nd ROM.
;       Some routines in 1st ROM might enable interrupts,
;       so all interrupt
;       hook table should be replaced with RET code.
;       And restore them after done the 1st ROM calling.
;
;       Entry   [ENTRY]:1st ROM entry address
;       Exit    for return condition of 1st ROM
;

; <<< SYSTEM define label >>>
BNKCRL  EQU     ^X0A1           ; Bank control port
STATUS  EQU     ^X0A0           ; Bank status port

; <<< Main routine >>>
        ORG     ^X8000          ; This routine must be stay
                                ;  ^X8000-^XFFFF

ROM1ST: SHLD    WORKH           ; Save register HL
        LXI     H,RET2ND        ; Return address from 1st ROM
        PUSH    H               ; Push stack top
        LHLD    ENTRY           ; Pick up 1st ROM entry
                                ; address
        PUSH    H               ; Push stack top
        LHLD    WORKH           ; Restore HL
        PUSH    PSW             ; Save all register
        DI                      ; Disable interrupt
        IN      STATUS          ; Get current bank status
        ANI     ^B11111110      ; Switch 1st ROM data set up
        OUT     BNKCRL          ; Bank select
                                ; Now ^X0000-^X7FFF are
                                ; 1st ROM
        EI                      ; Enable interrupt
        POP     PSW             ;
        RET                     ; Jump 1st ROM entry

; <<< Return from 1st ROM >>>
RET2ND: PUSH    PSW             ; Save all register
        IN      STATUS          ; Get current bank status
        ORI     ^B00000001      ; Switch 2nd ROM data set up
        OUT     BNKCRL          ; Bank select
                                ; Now ^X0000-^X7FFF are
                                ; 2nd ROM
        POP     PSW             ; Pick up all register
```

```
        RET                             ;

; <<< SYSTEM WORK AREA >>>
ENTRY:  DW      ^X0000                  ; 1st ROM entry address
WORKH:  DW      ^X0000                  ; HL register saving area

        END
```

3.4 SEQUENCES IN THE 2ND ROM

1. INITIALIZE

This sequence sets up [SP](Stack Pointer), power-on trap and other interrupt routines. Then it copies the book-keeping area and system area. finally some peripherals will be initialized by this routine.

2. RETURN TO MENU

At the first, this sequence selects the standard RAM, RAM #0 and resets the power-off trap. Then it jumps to 1st ROM's menu mode.

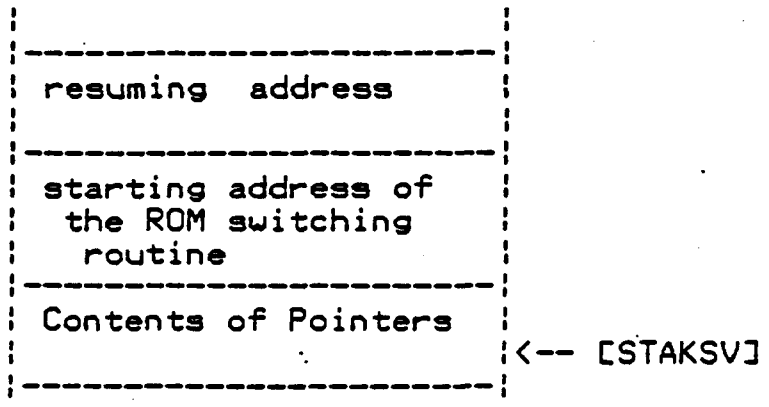
3. POWER DOWN

When power is turned off, the control is transferred to this sequence. In this sequence, you must save all registers and circumstances which should be saved in the stack. So the stack-pointer is most important to resume the current processing on the next power-on.

The RAM bank number is always stored in RAM #0. On turning on, the 1st ROM and RAM #0 is selected automatically. And the bank-switching procedure will be called in Power on sequence if the number of the RAM bank was not identical to the RAM #0 in the power down sequence. After changing the RAM bank, all registers will be restored and pending procedure will be resumed. Therefore in the stack, the address of the process which was abandoned by Power down trap should be stored.

In addition, in order to resume the abandoned process with 2nd ROM, you have to do special power on/power off sequence. In power off trap, you should the set the start routine of the special power-on sequence which switches the ROM bank. I recommend to use the hook, ^XF38F. Usually, 'JUMP to POWER FAIL SEQUENCE' command is stored here. In 2nd ROM, however, you have to rewrite this hook table and call the special power down routine here. In

it, the address of special power-on routine on the stack. In this case, the following information should be stacked before 'HLT' command is executed.



[STAKSV] keeps the SP's value at 'HLT'.

Fig 3.1

4. POWER ON

At the first, the initializing routine in ROM #0 checks the RAM bank number in BANK (^XF3DB) when power-off was executed. When power-off was done in non-standard RAM bank, RAM bank-switching routine is called and switched. Then, the registers' contents will be restored. If the address of the process which should be resumed was stacked, the address will be picked up and executed. When the power-down was detected in ROM #1, the address of the special ROM switching routine ought to be stacked above the address of the process should be resumed. Therefore, after switching the ROM, the abandoned process will be resumed.

The following figure are the general 2nd ROM routine control sequence.

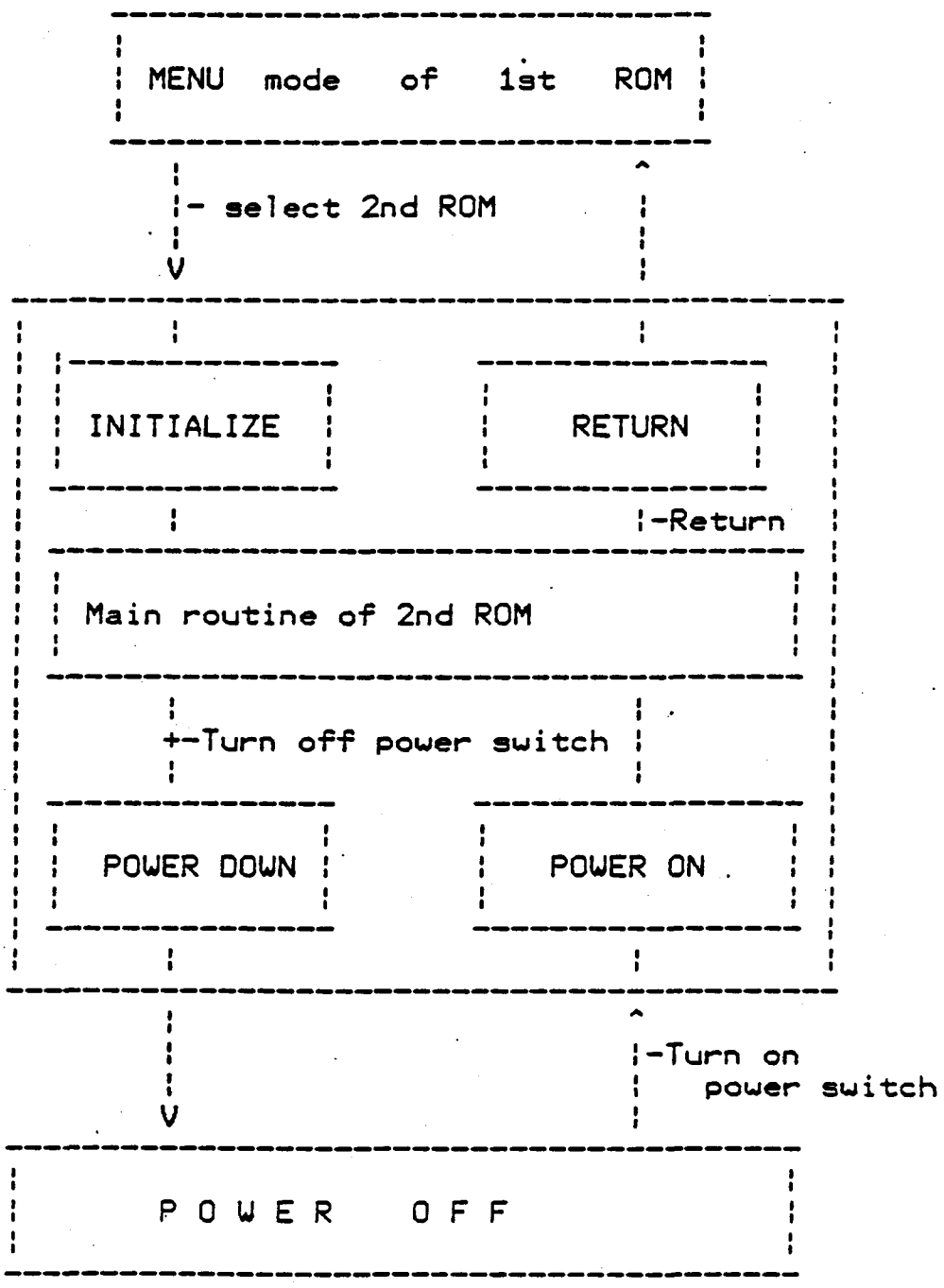


Fig 3.2