Chapter 2

Specifications of other BIOS

Chapter 2 BIOS Subroutine Specifications

1. General

This chapter describes specifications for the principle subroutines used within BIOS. Most of these subroutines are located in BIOS2, 3, 4, and 5 in the system bank, and cannot be called directly from user programs.

Note that the addresses shown in the program lists are not fixed, but vary according to the version of MultiFonts CP/M® used.

2. Subroutine Specifications

2.1 Subroutine SIGNMSG (BIOS2)

(1) Description of processing

The SIGNMSG subroutine clears the display screen and displays the MultiFonts CP/M® opening message.

(2) Call procedure

a) Entry parameters: None

CALL SIGMSG

b) Return information: None

2.2 Subroutine KBCOM (BIOS1)

(1) Description of processing

This subroutine outputs a 1-byte command to the keyboard.

(2) Call procedure

a) Entry parameters

Register C = Command to be output

CALL KBCOM

b) Return information: None

2.3 Subroutine CALBRT (BIOS2)

(1) Description of processing

This subroutine outputs a calibrate command to the specified flexible disk drive.

(2) Call procedure

a) Entry parameters

The drive number of the drive to which the command is to be output is set in DRIVE (address 0FE01H) in the BIOS common data area. (0 is specified for drive A, 1 for drive B, 2 for drive

C, and 3 for drive D.)

CALL CALBRT

b) Return information: None

However, the status is saved in FDBSY (address 0FE11H) in the BIOS common area. See the description of the BIOS common data area for details.

2.4 Subroutine NSECRW (BIOS2)

(1) Description of processing

This subroutine outputs or inputs data from the flexible disk specified in parameters set in the BIOS common data area.

(2) Call procedure

a) Entry parameters

Access to the flexible disk drive is governed by settings made in DRIVE (0FE01H, the drive number), TRACK (0FE02H, the track number), HEAD (0FE03H, the head position), SECTOR (0FE04H, the sector number), SECTCT (0FE05H, the number of bytes to be read or written), and RWFLG (0FE00H, set to 0 for writes and to other than 0 for reads) in the BIOS common data area.

CALL NSECRW

b) Return information

$A = 0$	:	Normal completion (Z flag=1)
$A \neq 0$	:	Error (Z flag=0)

If an error occurs, the error status is indicated in FDSTS (address 0FED3H) and FDBSY (address 0FE11H) in BIOS1. See the description of the BIOS common data area for details.

2.5 Subroutine WRITEHST (BIOS2)

(1) Description of processing

This subroutine writes data to a flexible disk drive in accordance with parameters set in the BIOS common data area. Logical data is set in the system table.

(2) Call procedure

a) Entry parameters

Access to the flexible disk drive is governed by settings made in HSTDSK (0FC44H, the drive number), HSTTRK (0FC45H, the track number), and HSTSEC (0FC47H, the number of sectors to be written). For HSTSEC, one logical sector corresponds to four physical sectors; therefore, one logical sector consists of 1024 bytes.

CALL WRITEHST

b) Return information: None

However, a return code is saved in ERFLAG (address 0FC50H) in the BIOS common data area. Normal completion is indicated when the return code is 0, and an error is indicated if it is other than 0. Also, the status at that time is saved in FDBSY (address 0FE11H).

2.6 Subroutine DMASET (BIOS2)

(1) Description of processing

This subroutine makes settings required for making DMA reads or writes to flexible disks as specified by parameters set in the BIOS common data area.

(2) Call procedure

a) Entry parameters

Settings to be made are specified in advance in DBADDR (address 0FE06H, the buffer address) and SECTCT (address 0FE05H, the number of sectors) in the BIOS common data area.

CALL DMASET

b) Return information: None

2.7 Subroutine COMMD (BIOS2)

(1) Description of processing

This subroutine outputs commands set in the BIOS common data area to the flexible disk drive (µPD765).

(2) Call procedure

a) Entry parameters

Commands to be output are set in FDCOM (address 0FE08H) in the BIOS common data area, and the number of commands to be output is set in register B.

CALL COMMD

b) Return information: None

Note:

All interrupts other than those from the flexible disk drive are masked (inhibited) upon execution of this subroutine; therefore, the mask bits must be restored to their original conditions by the flexible disk drive interrupt processing routine or routine for processing execution results.

Chapter 3

Interrupt Processing Routines

Chapter 3 Interrupt Processing Routines

1. General

BIOS provides interrupt processing only for the following four devices.

a) Keyboard (μPD7201)

b) Flexible disk controller (μPD765)

c) RS-232C interface (μPD7201)

d) Light pen (optional) (μPD7220)

However, the MASKI routine uses interrupt processing to check whether or not option cards are installed when it is called to reset an interrupt mask. The interrupt processing routine does the following.

a) Reads the status of devices and sets status flags.

b) Reads data from devices (excluding the flexible disk controller).

There is no way of knowing what states will result in generation of an interrupt, nor or what memory bank will be selected at the time an interrupt occurs. Therefore, the following processing must be performed at the beginning of the interrupt processing routine.

a) The stack pointer must be saved somewhere between addresses 0E000H and 0FFFFH of the common data area and a new stack pointer must be established elsewhere in that area.

b) The number of the currently selected memory bank must be read (by checking the settings of bits 4 to 7 of the data obtained by reading I/O port 030H), then that number must be saved and the bank switched to that containing the program which actually handles the interrupt processing. This is done by writing the bank number to I/O port 018H.

c) Execution must be transferred to the interrupt processing routine by a JP instruction, etc.

d) The interrupt processing routine must save the stack pointer in the common data area of memory and establish a new stack pointer in its own bank.

e) The contents of all registers must be saved.

f) Interrupt processing is done after the above steps have been completed.

Calls to BIOS routines cannot be made from within the interrupt processing routine.

Correct results and continued execution of the program prepared are not assured unless this processing is done.

2. Preparation of interrupt processing routines

The only options supported by BIOS are the RS-232C interfaces and the light pen. Therefore, interrupt processing routines must be prepared when other options are to be used. Points to be considered when preparing such routines are described below.

(1) All option interrupts are inhibited after a cold start has been made; in other words, all interrupt masks are set. Therefore, the interrupt mask bit corresponding to the device to be used must be reset (to enable interrupts). This is done by calling the MASKI routine described in Chapter 1. See Chapter 1 for details.

(2) After calling the MASKI routine and resetting the interrupt mask, set the execution address of the interrupt processing routine in the BIOS interrupt vector table. This vector table is contained in BIOS1, and its starting address is fixed at 0FD80H. The table consists of 16 4-byte blocks.

The option interrupt level is returned in the return information after normal termination of the MASKI routine. Set the exectuion address of the interrupt processing routine in the vector table according to that level.

The interrupt levels and corresponding addresses in the vector table are as follows.

Vector address	Interrupt level/interrupt device
0FD80H	Power failure
0FD84H	Software timer 1
0FD88H	Option level 1 (slots 1∿5)
0FD8CH	Option level 2 (slots 1∿5)
0FD90H	Keyboard or main board RS-232C interface
0FD94H	Light pen
0FD98H	Flexible disk controller
0FD9CH	Reserved (setting inhibited)
0FDA0H	Printer
0FDA4H	Option level 3 (slot 1)
0FDA8H	Calender clock
0FDACH	Option level 4 (slot 2)
0FDB0H	Option level 5 (slot 3)
0FDB4H	Software timer 2
0FDB8H	Option level 6 (slot 4)
0FDBCH	Option level 7 (slot 5)

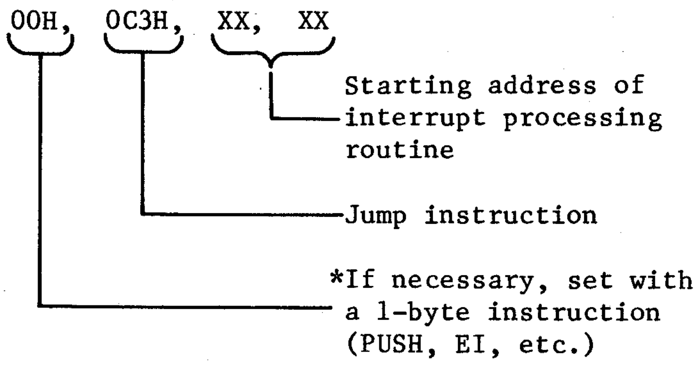
The interrupt sequence is performed as follows.

When an interrupt occurs, the interrupt controller (an Intel 8259A) outputs a CALL instruction to the CPU for the vector address corresponding to the interrupt level; i.e., a CALL is made to the vector address corresponding to the interrupt level.

Example: When a keyboard interrupt is generated, three bytes (0CDH, 090H, and 0FDH) are output to the CPU. These constitute the following instruction.

```
CALL    0FD90H
```

Thus, all that is necessary is to execute a jump instruction to the starting address in the vector table of the interrupt processing routine which corresponds to the interrupt level. The vector table entry consists of 4 bytes, which are set in the following format.


00H, 0C3H, XX, XX
Starting address of interrupt processing routine
Jump instruction
*If necessary, set with a 1-byte instruction (PUSH, EI, etc.)

This procedure passes control to the interrupt processing routine.

(3) The interrupt processing routine must satisfy the requirements outlined in paragraphs a) to e) above. In other words, items a) to c) must be loaded into the common data area in memory.

Normally, CCP, BDOS, and BIOS1 are loaded into the resident area of memory; however, if BDOS is not called, addresses 0E000H to 0F7FFH can be used in any manner desired.

To return control to CP/M upon completion of the processing program, simply call the WBOOT routine.

(4) Before returning from the interrupt processing routine after interrupt processing has been completed, be sure to post completion of interrupt processing to the interrupt controller and reenable interrupts. This is done by coding the following steps.

```
        .
        LD      A,020H          ;NORMAL EOI.
        OUT     (08H),A         ;RESTORE REGISTERS.
        EI                      ;ENABLE INTERRUPT.
        RETI                    ;RETURN.
```

Note that 020H is written to I/O port 08H; this posts completion of interrupt processing to the interrupt controller. However, this applies to vector addresses from 0FD80H to 0FD9CH for 7 interrupt levels; when the vector address is from 0FDA0H to 0FDBCH, it is also necessary to write 020H to I/O port 0CH.

```
        .
        .
        LD      A,020H
        OUT     (08H),A         ;NORMAL EOI
        OUT     (0CH),A         ;
        .
        .
```

The reason for this is that there are two interrupt controllers, which are connected in cascade. As indicated earlier, this provides for up to 15 interrupt control levels. I/O port 08H is the address for the master interrupt controller, and 0CH is the address for the slave interrupt controller.

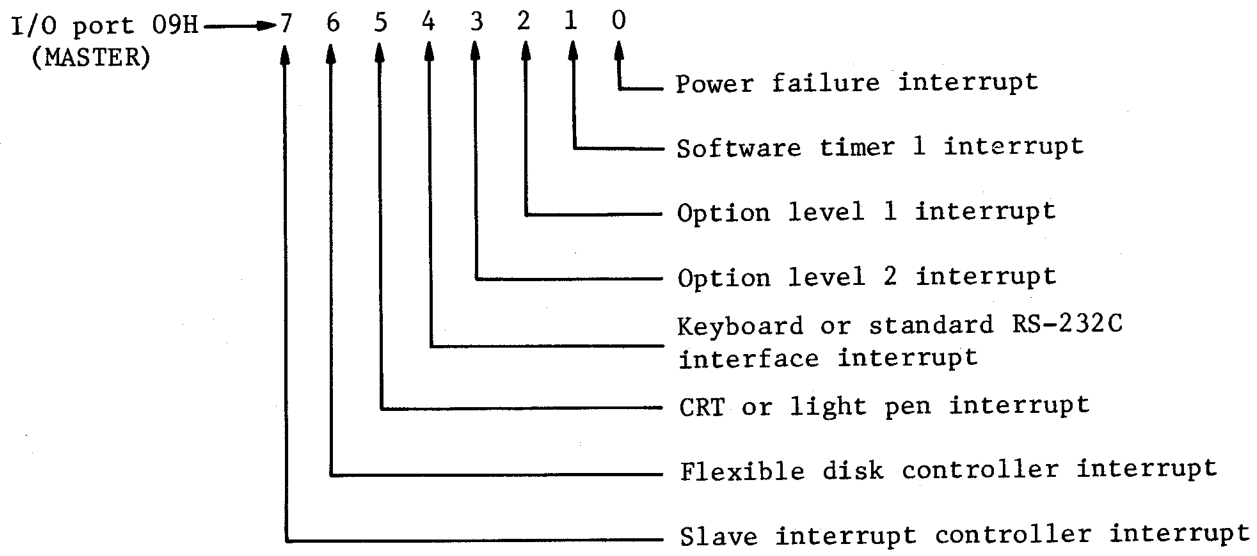
(5) The highest interrupt level is that for power failure interrupts and the lowest is option level 7.
Other interrupts are inhibited when control is passed to an interrupt processing routine. Therefore, all other interrupts are held pending even when an interrupt of the lowest level (option level 7) is being processed. Pending interrupts are processed when the EI (enable interrupts) instruction is executed.

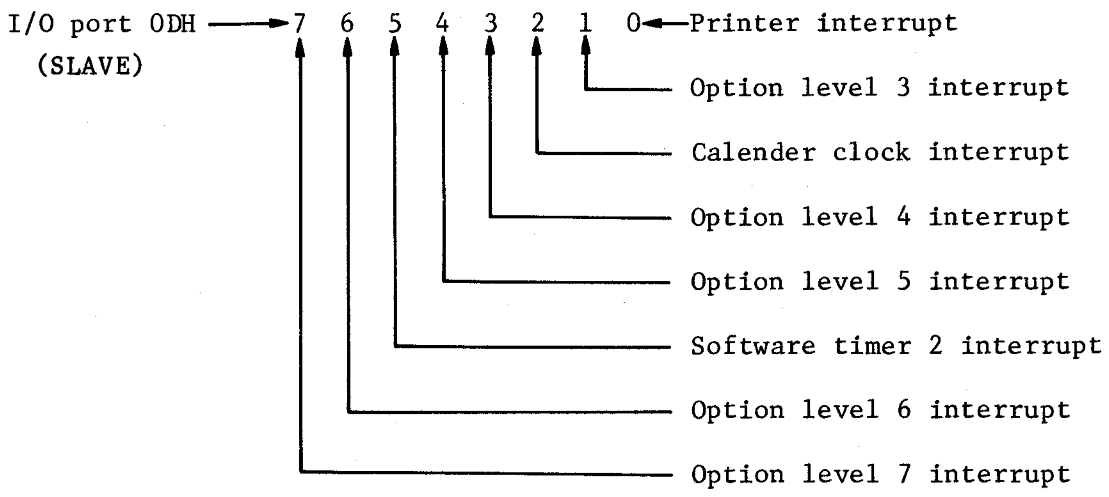
If you wish to perform multi-level interrupt processing, output the EI instruction at the beginning of the interrupt processing routine. After doing this, any higher level interrupts occurring during processing routine execution will be handled immediately.

(6) All interrupts other than those from the flexible disk controller are masked during flexible disk access. Therefore, other interrupts are held pending until completion is posted from the device or a software timeout is detected.

(7) Interrupt masks

Interrupt masks can be set for the I/O ports indicated below. The settings fo the masks can also be read. Interrupts are inhibited when corresponding mask bits are set to "1" and enabled when they are set to "1".


I/O port 09H
(MASTER)
7 6 5 4 3 2 1 0
Power failure interrupt
Software timer 1 interrupt
Option level 1 interrupt
Option level 2 interrupt
Keyboard or standard RS-232C interface interrupt
CRT or light pen interrupt
Flexible disk controller interrupt
Slave interrupt controller interrupt



(8) Power failure interrupt routine

With the QX-10, it is possible to execute from 400 to 1000 program steps between the time a power failure is detected and the time the power actually drops below minimum operating levels. The user can make use of this time to store vital data in CMOS RAM (2048 bytes). However, this operation is not assured by hardware and thus there is no positive assurance that it will be carried out.

The power failure interrupt routine of BIOS performs the following processing.

o Disables CMOS RAM.

o Resets DMA.

The user can use independently developed interrupt processing procedures instead of the above routine. Naturally, such routines must be kept constantly resident in memory; however, it is not necessary for them to be located in the common data area. An example of user setup for a power failure interrupt processing routine is shown below. This program consists of two sections. One sets the jump address to the user interrupt processing routine, and the other is actually executed when a power failure occurs.

(i) Jumps to a user processing routine in BIOS

```
PDINT    EQU    0FD8CH+2
         .
         .
         LD     DE,(PDINT)  ;DE=POWER FAILURE JUMP ADDRESS.
         LD     HL,PWNUC    ;
         LD     BC,PWNUCE-PWNUC;
         LDIR               ;SET EXIT JUMP ROUTINE.
         .
         .
PWNUC:   LD     A,010H
         OUT    (018H),A
         JP     PWEXIT      ;JUMP TO USER ROUTINE.
PWNUCE   EQU    $
```

The above program may be executed to rewrite the BIOS power failure routine with the seven bytes from PWNUC to PWNUCE. Since there are only 12 bytes available in the BIOS area for this purpose, the length of the rewritten portion must fit into that space. The address of the main section of the user program should be indicated for the label in JP PWEXIT. This main section is the program which actually saves data in CMOS RAM.

(ii) Example of program for saving data in CMOS RAM

```
        PWEXIT:  LD     A,01H
                 OUT    (020H),A    ;ENABLE CMOS RAM.
                 LD     DE,CMOS     ;
                 LD     HL,SVDATA   ;HL=DATA ADDRESS.
                 LD     BC,DLNG     ;
                 LDIR               ;SAVE CMOS.
                 HALT               ;STOP!
        ;
        CMOS     EQU    08000H      ;CMOS TOP ADDRESS.
        DLNG     EQU    2048        ;CMOS  LENGTH.
```

CMOS RAM is assigned to the 2048 bytes starting at address 08000H, and is normally disabled. It may be enabled by writing 01H to I/O port 020H. However, once CMOS is enabled, memory from 08800H to 0DFFFH cannot be accessed. This means that the PWEXIT program shown in (ii) above and the data indicated by SVDATA must be located in the area from address 0H to 07FFFH. The program may be located in any memory bank. In (i) above, the bank is selected by LD A,010H of PWNUC; this may be changed as necessary to select the bank desired.

(9) Interrupt processing for option cards prepared by users

When the user builds an interface using the universal card, processing for interrupts from that interface is performed as follows. See the "QX-10 Technical Manual-Hardware operations" concerning the option board interface.

When the option interrupt level is set to 1 or 2, interrupts are generated at that level regardless of the slot into which the card is inserted. However, the level of interrupts generated varies according to slot for interrupt levels 3, 4, 5, 6, and 7.

Correspondence between the option interrupt levels and the option slots is as follows.

Slot 1:	Option interrupt level 3
Slot 2:	Option interrupt level 4
Slot 3:	Option interrupt level 5
Slot 4:	Option interrupt level 6
Slot 5:	Option interrupt level 7

Since the interrupt level is determined by the slot into which the option card is inserted, the corresponding processing address must be set in the vector table at the applicable point. The vector table entry consists of four bytes; these are filled with a NOP instruction and a jump instruction to the processing routine. The program which does this must also reset the corresponding interrupt mask. However, there is no way of predicting when interrupts will be generated or what memory bank will be selected at the time. Therefore, the interrupt processing routine must be loaded into common data area. In order to prevent destroying BIOS1, the routine must be limited to the 5632 bytes from address 0E000H to 0F5FFH; further, CCP and BDOS cannot be used in this case. If BDOS is to be used, it must be limited to the 2048 bytes from 0E000H to 0E7FFH. (This also includes space for the stack.)

Two program examples are shown below. One sets the NOP and jump instructions and resets the interrupt mask in the vector table, and the other is used for entering/exiting the interrupt processing routine.

(i) Setting NOP and the JUMP instruction

```
INTTBL  EQU     0FD80H          ;INTERRUPT VECTOR TOP ADDR.
LV5RST  EQU     01110111 1B     ;LEVEL 5 MASK OFF.
        .
        .
        DI                      ;DISABLE INTERRUPT.
        LD      DE,INTTBL+4*12;LEVEL 5 VECTOR.
        LD      HL,LV5JP        ;
        LD      BC,4            ;
        LDIR                    ;SET JUMP SEQUENCE.
```

```
                    IN      A,(0DH)         ;PIC CURRENT MASK.
                    AND     LV5RST          ;RESET LEVE5 MASK.
                    OUT     (0DH),A         ;
                    EI                      ;
                    .
                    .

          LV5JP:    NOP                     ;
                    JP      LV5INT          ;EXECUTE INTERRUPT HANDLER.
```

After execution of this program, execution jumps to the LV5INT label address when an option level 5 interrupt is generated. See Chapter 4 for details on the instructions marked with an asterisk.

(ii) Interrupt processing routine entry/exit

```
          LV5INT:   LD      (SVSP),SP       ;SAVE CURRENT STACK POINTER.
                    LD      SP,INTSTACK     ;SET NEW STACK.
                    PUSH    AF              ;SAVE ALL REGISTER.
                    PUSH    BC              ;
                    PUSH    DE              ;
                    PUSH    HL              ;
                    PUSH    IX              ;
                    PUSH    IY              ;
                    EX      AF,AF'          ;
                    EXX                     ;
                    PUSH    AF              ;
                    PUSH    BC              ;
                    PUSH    DE              ;
                    PUSH    HL              ;
                    .
                    .

          PROCESSING ROUTINE =>   THIS ROUTINE MUST NOT INCLUDE CALLS TO
                                  BIOS OR BDOS.

                    .
                    .
                    POP     HL              ;RESTORE ALL REGISTERS.
                    POP     DE              ;
                    POP     BC              ;
                    POP     AF              ;
                    EXX                     ;
                    EX      AF,AF'          ;
                    POP     IY              ;
                    POP     IX              ;
                    POP     HL              ;
                    POP     DE              ;
                    POP     BC              ;
                    LD      A,020H          ;NORMAL EOI.
```

```
                OUT     (08H),A       ;MASTER.
                OUT     (0CH),A       ;SLAVE.
                POP     AF            ;
                LD      SP,(SVSP)     ;RECOVER CURRENT STACK POINTER.
                EI                    ;ENABLE INTERRUPT.
                RETI                  ;CONTINUE.
;
SVSP:           DS      2             ;STACK POINTER SAVE AREA.
                .
                .
INTSTACK        EQU     0E800H        ;TEMPORARY STACK AREA.
                .
                .
```

(10) Interrupt processing using the calendar clock

The calendar clock of the QX-10 can be used to generate interrupts at intervals of minutes, seconds, or hours. Preparation of an interrupt processing routine for interrupts from the calendar clock thus makes it possible to perform prescribed processing at fixed intervals. However, all or part of the interrupt processing routine must be included in the common data area of memory (the area from address 0E000H to 0F5FFH, or from 0E000H to 0E7FFH).

(i) Setting the interrupt processing routine address in the vector table

```
INTTBL      EQU     0FD80H         ;INTERRUPT VECTOR TOP ADDR.
CCMRST      EQU     01111011B      ;CALENDAR CLOCK MASK OFF.
            .
            .
            DI                     ;DISABLE INTERRUPT.
            LD      DE,INTTBL+4*10;CALENDAR CLOCK INT. VECTOR.
            LD      HL,CLINT       ;
            LD      BC,4           ;
            LDIR                   ;SET JUMP SEQUENCE.
            IN      A,(0DH)        ;PIC CURRENT MASK.
            AND     CCMRST         ;RESET CALENDAR CLOCK MASK.
            OUT     (0DH),A        ;
            EI                     ;ENABLE INTERRUPT.
            .
            .
CLINT:      NOP                    ;
            JP      CLCKINT        ;EXECUT INTERRUPT HANDLER.
```

CLCKINT is the starting address of the interrupt processing routine. After execution of this program, processing will jump to CLCKINT whenever a calendar clock interrupt is generated.

Procedures for setting the hour, minute, and second are as follows. See (9) above for an example of an interrupt processing routine.

(ii) Setting the calendar clock

This setting must be made whenever the QX-10's power is turned on or the reset button is pressed. This also applies when the TIMDAT routine of BIOS is called.

```
 RSREG     EQU     03DH          ;SELECT RAM ADDRESS.
 RDREG     EQU     03CH          ;RAM DATA.
           .
           .
 CHKUP:    LD      A,0AH         ;SELECT CONTROL REG1.
           OUT     (RSREG),A     ;
           IN      A,(RDREG)     ;READ CONTROL REG1.
           AND     080H          ;UPDATE CYCLE?
           JR      NZ,CHKUP      ;YES.
           LD      A,0BH         ;SELECT CONTROL REG2.
           OUT     (RSREG),A     ;
           LD      A,08AH        ;STOP UPDATE.
           OUT     (RDREG),A     ;
           LD      DE,RAMAD      ;RAM ADDRESS TABLE.
           LD      HL,INTTIME    ;INT. TIME TABLE.
           LD      BC,033CH      ;REG B=3, REG C=RDREG.
 SETTIME:  LD      A,(DE)        ;SET H,M,S.
           OUT     (RSREG),A     ;
           INC     DE            ;
           OUTI                  ;
           JR      NZ,SETTIME    ;
           LD      A,0BH         ;SELECT CONTROL REG2.
           OUT     (RSREG),A     ;
           LD      A,02AH        ;ENABLE CALENDAR CLOCK INT.
           OUT     (RDREG),A     ;
           .
           .
 RAMAD:    DB      5,3,1         ;HOURS, MINUTES, SECONDS (RAM ADDR.)
 ;
 INTTIME:  DB      0,030H,015H   ;0H:30M:15S  (BCD DATA)
           .
```

The setting for the QX-10's calendar clock is made in BCD code using 24-hour representation. Therefore, the setting must be made in BCD code.

The addresses in RAM for the hour, minute, and second are as shown below.

```
    Hour:   05H (set with 00H to 023H)
  Minute:   03H (set with 00H to 059H)
  Second:   01H (set with 00H to 059H)
```

The RAM address is selected by writing a corresponding number to I/O port 03DH.

The example above causes a calendar clock interrupt to be generated each day when the time reaches 00:30:15. Whether or not the hour, minute, or second data is used in controlling the timing at which interrupts are generated is determined by the settings of bits 7 and 6 of each value. If these bits are set to "1", the relevant data is not used in controlling interrupt timing.

Examples:

Generating interrupts at 10 second intervals

```
     INTTIME:  DB  0C0H, 0C0H, 010H
```

Generating interrupts at 15 minute intervals

```
     INTTIME:  DB  0C0H, 015H, 0C0H
```

Generating interrupts at 2 hour intervals

```
     INTTIME:  DB  02H, 0C0H, 0C0H
```

(iii) Control register 3 must be read by the interrupt processing routine as shown below. This serves to reset the calendar clock interrupt flag.

```
        .
        .
        LD      A,0CH          ;SELECT CONTROL REG3.
        OUT     (03DH),A       ;
        IN      A,(03CH)       ;READ CONTROL REG3.
        .
        .
```

(iv) The CMOS real-time clock IC used is a Hitachi HD146818 (compatible with the Motorola MC146818). This is a time-of-day clock and calendar which counts seconds, minutes, and hours of the day, as well as days of the week and the date, month, and year. Correction for leap years is done automatically.

The device includes 50 bytes of static RAM, and operates on a time base of 32.768 kHz. It also provides programmable periodic interrupt generation and square wave output. Backup is provided by a NiCd rechargeable battery.

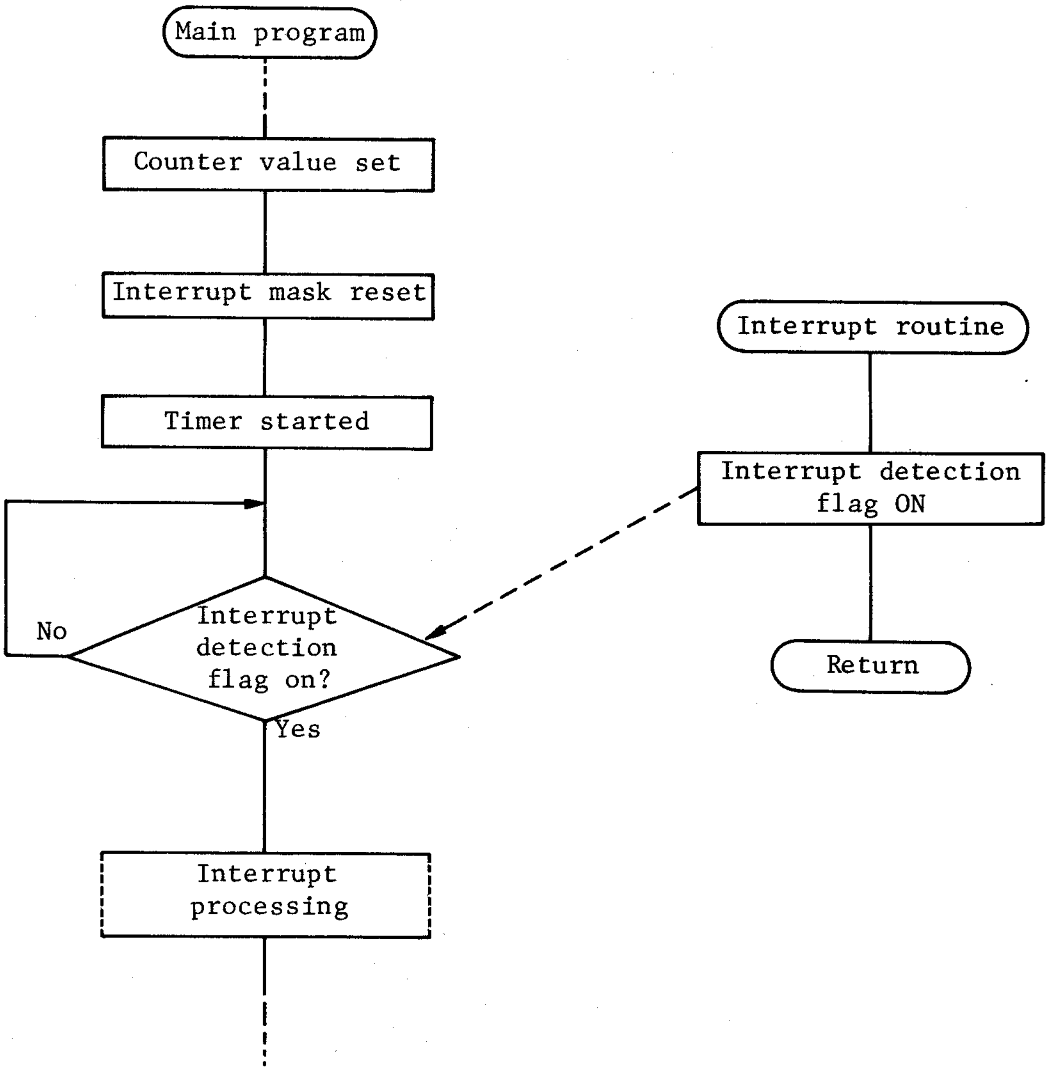
Signals applicable to operation of the real-time calendar clock IC are as shown in the table below.

Pin No.	Signal Name	I/O	Function
2	OSC1	I	External clock signal (32.768 kHz).
3	OSC2	-	External clock signal (32.768 kHz). Open during input.
4 to 11	AD0 to AD7	I/O	Bidirectional bus lines through which the CPU transfers the RTC access address and data. The access address is transferred during the first half of the cycle, and data is transferred during the second half. The address signal level is determined at the falling edge of M. The impedance of the data bus driver and tri-state output buffer is high except during RTC data output.
12	Vss	-	GND
13	$\overline{CS}$	I	Chip select.
14	M	I	Strobe signal used to read the address from the address bus. The address is read into the RTC at the falling edge of this signal.
15	$\overline{W}$	I	Input terminal for the R/W signal from the CPU. The CPU sets W to H to read the RTC, and to L to write data to the RTC.
17	$\overline{G}$	I	System clock input terminal. The CPU reads data from the RTC while $\overline{G}$ is H, and writes data to the RTC at the falling edge of $\overline{G}$.
18	$\overline{RES}$	I	RTC reset signal. Resets the control registers and user RAM. RTC operation resumes when RES returns to L. This signal does not affect the clock or calendar RAM.
19	$\overline{IRQ}$	O	CPU interrupt request signal (active low).
22	PS	I	When this signal becomes L, the Valid RAM and Times (VRT) bit is cleared to 0; the CPU should then initialize the RTC and set the VRT bit to 1 to prepare for a power failure.
24	Vcc	-	Power supply

(11) Software timer interrupts

The QX-10 is equipped with two software timers. One of these counts down in $1/2x10^6$-second units, and the other counts down in 1/1200-second units. Both generate interrupts when 0 is reached. The counters consist of 16 bits, and either a binary count or BCD count can be selected. Therefore, the maximum timer intervals are 32.8 ms or 54.6 seconds, respectively. The timer which counts down in $1/2x10^6$ second units is designated software timer 1, or the fast timer; and that which counts down in 1/1200 second units is designated software timer 2, or the slow timer.

An example of interrupt processing using the software timers is shown below.

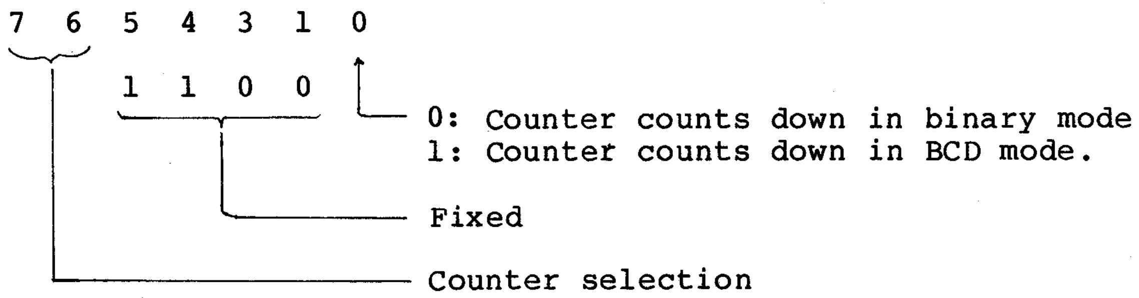


(i) Setting the counter value

The counter value is set as follows.

```
        TCTRL     EQU    03H
        TCNTF     EQU    02H           ;FAST COUNTER.
        TCNTS     EQU    01H           ;SLOW COUNTER.
                  .
                  .
                  LD     A,070H        ;SELECT TIMER SLOW.
                  OUT    (TCTRL),A     ;
                  LD     A,(COUNTSL)   ;SET LOWER VALUE.
                  OUT    (TCNTS),A     ;
                  LD     A,(COUNTSU)   ;SET UPPER VALUE.
                  OUT    (TCNTS),A     ;
        ;
                  LD     A,0B0H        ;SELECT TIMER FAST.
                  OUT    (TCTRL),A     ;
                  LD     A,(COUNTHL)   ;SET LOWER VALUE.
                  OUT    (TCNTF),A     ;
                  LD     A,(COUNTHU)   ;SET UPPER VALUE.
                  OUT    (TCNTF),A     ;
                  .
                  .
        COUNTSL   EQU    $
        COUNTSU   EQU    $+1
                  DW     1200          ;1 SEC.
        ;
        COUNTHL   EQU    $
        COUNTHU   EQU    $+1
                  DW     0FFFFH        ;32.8 MILLI SEC.
                  .
                  .
```

The program above writes 070H and 0B0H to I/O port 070H. The meanings of the corresponding bit patterns are as follows.



0: Counter counts down in binary mode
1: Counter counts down in BCD mode.

Fixed

Counter selection

Software timer

01: Software timer 2 (slow)
10: Software timer 2 (fast)
00 and 11 must not be specified.

(ii) Example of setting the interrupt vector address and
     resetting the interrupt mask

```
MASKI     EQU     0F65AH        ;BIOS ENTRY
;
INTTBL    EQU     0FD80H        ;INTERRUPT VECTOR TOP ADDRESS.
          .
          .
          DI
          LD      DE,INTTBL+4*1;SOFTWARE TIMER FAST VECTOR.
          LD      HL,STINTF     ;
          LD      BC,4          ;
          LDIR                  ;SET JUMP SEQUENCE. (FAST)
          LD      DE,INTTBL+4*13;SOFTWARE TIMER SLOW VECTOR.
          LD      HL,STINTS     ;
          LD      BC,4          ;
          LDIR                  ;SET JUMP SEQUENCE. (SLOW)
          .
          .
;
STINTF:   PUSH    AF            ;
          JP      STIM1         ;SOFTWARE TIMER FAST INT. HANDLER.
;
STINTS:   PUSH    AF            ;
          JP      STIM2         ;SOFTWARE TIMER SLOW INT. HANDLER.
          .
          .
```

(iii) Example of interrupt processing routine

```
STIM1:    LD      A,1           ;SOFTWARE TIMER FAST.
          JR      SETF
;
STIM2:    LD      A,020H        ;SOFTWARE TIMER SLOW.
          OUT     (0CH),A       ;NORMAL EOI. (SLAVE)
          LD      A,2           ;
SETF:     LD      (INTD),A      ;SET INTERRUPT ID.
          LD      A,020H        ;
          OUT     (08H),A       ;NORMAL EOI. (MASTER)
          POP     AF            ;RECOVER REGISTER.
          RETI                  ;RETURN.
;
INTD      EQU     0E7FFH        ;
;
```

The interrupt processing program above must be loaded into the resident area of memory (starting at address 0E000H). The interrupt device (software timer 1 or software timer 2) can be determined from the contents of INTD (address 0E7FFH) by user programs to perform subsequent processing accordingly.

(iv) Starting the software timers

Counter operation is starting by writing data to I/O port 018H. However, since this port is used for switching memory banks, the following procedure must be used.

```
        .
        .
        IN      A,(030H)      ;READ CURRENT BANK NO.
        AND     0F0H          ;
        OR      02H           ;SET SOFTWARE TIMER TRIGGER.
        OUT     (018H)        ;START TIMER.
        .
        .
```

This procedure is effective for both software timer 1 and software timer 2.