Does Anybody Really Know What Time it is?

Add a real time, non-interruptable hardware clock/calendar to your 6502 system using a new clock chip and you will be as close to knowing as anyone can be.

A hardware real-time clock has several advantages over a software real-time clock. First, keeping time does not require interrupt driver software, thereby saving machine time overhead and RAM space. Next, the circuit described here can generate its own interrupts to the microprocessor if regularly spaced interrupts are needed. Finally, and perhaps most significant is that being non-interruptable with its battery backup, the time only has to be set when starting up the first time. Neither turning off the microprocessor system nor power outages affect the keeping of time.

The MSM5832

The MSM5832 from OKI Semiconductor is a CMOS clock/calendar chip made especially for bus-oriented microprocessor applications. Due to its special design, it offers many advantages over other types of conventional clock circuits when used with a microprocessor as a non-interruptable clock/calendar.

The MSM5832 keeps track of seconds, minutes, hours, day of the week, date, month, and year. Data is read and written by using a four bit bi-directional bus, when addressed by a four bit address bus. Table 1 shows the function of each address. Notice that in

ADDRESS INPUTS A0	A1	A2	A3	INTERNAL COUNTER	DATA I/O D0	D1	D2	D3	DATA LIMITS	NOTES
0	0	0	0	S 1	*	*	*	*	0-9	S1 or S10 reset to zero whenever write is executed
1	0	0	0	S 10	*	*	*		0-5	
0	1	0	0	MI 1	*	*	*	*	0-9	
1	1	0	0	MI 10	*	*	*		0-5	
0	0	1	0	H 1	*	*	*	*	0-9	
1	0	1	0	H 10	*	*	†	†	0-1 0-2	D2="1",PM D3="1",24 Hour D2="0",AM D3="0",12 Hour
0	1	1	0	W	*	*	*		0-6	
1	1	1	0	D 1	*	*	*	*	0-9	
0	0	0	1	D 10	*	*	†		0-3	D2="1",29 days in month 2(2) D2="0",29 days in month 2
1	0	0	1	MO 1	*	*	*	*	0-9	
0	1	0	1	MO 10	*				0-1	
1	1	0	1	Y 1	*	*	*	*	0-9	
0	0	1	1	Y 10	*	*	*	*	0-9	

(1) * Data valid as "0" or "1".
Blank does not exist(unrecognized during WRITE and held at "0" during READ).
† Data bits used for AM/PM, 12/24 Hour and leap year.
(2) If D2 previously set to "1", upon completion of month 2 day 29, D2 will be internally reset to "0"

Table 1: **Functions**

CONDITIONS	OUTPUT	REFERENCE FREQUENCY	PULSE WIDTH
HOLD = L	D0(1)	1024 Hz	duty 50%
READ = H	D1	1 Hz	122.1 us
CS = H	D2	1/60 Hz	122.1 us
A0-A3 = H	D3	1/3600 Hz	122.1 us

(1) 1024 Hz signal at D0 not dependent on HOLD input level.

Table 2: **Reference Signal Outputs**

addition to being able to program through software either a 12 or 24 hour format, leap years are handled quite easily. Leap years are controlled by bit D2 of address 8. When set, it gives the second month of the year a 29th day, and after the 29th day has elapsed, the bit is automatically cleared. The bit may be set any time after the second month of the previous year, and before the end of the second month of the leap year.

Another feature is a manual ±30 second correction input. Perhaps the most unique and useful feature is the HOLD control which allows read/write operations to occur with the counters being held static, without disturbing the accuracy of the real time. Additionaly, four different interrupt outputs are available to the microprocessor, as shown in table 2. Finally, the chip will operate on a battery back up as low as 2.2V with a power dissipation of less than 90uW, making long term backup quite attractive and economical.

Functions

The functions of the clock/calendar are best described on an individual basis as follows.

Oscillator (XT,$\overline{XT}$)

A 32.768KHz(2^{15}) crystal is connected to an internal, stable oscillator to form an accurate time base. The two parallel capacitors, one of which is a trimmer, allow the oscillator to be tuned quite precisely.

A0-A3

These are the address inputs which are used to select the internal counters to be set or read on a read or write operation.

D0-D3

These are data inputs or outputs, depending on whether a read or write operation is being done. They are tri-state bi-directional ports controlled by the READ and WRITE controls.

Chip Select

This determines whether the inputs and outputs are active or inactive. Connecting the CS to Vcc activates the inputs and outputs, while connection to ground disables them. In the circuit in figure 1, the CS is permanently connected to +5V from the microprocessor system for battery backup configuration. When the main system is turned off, this disables all

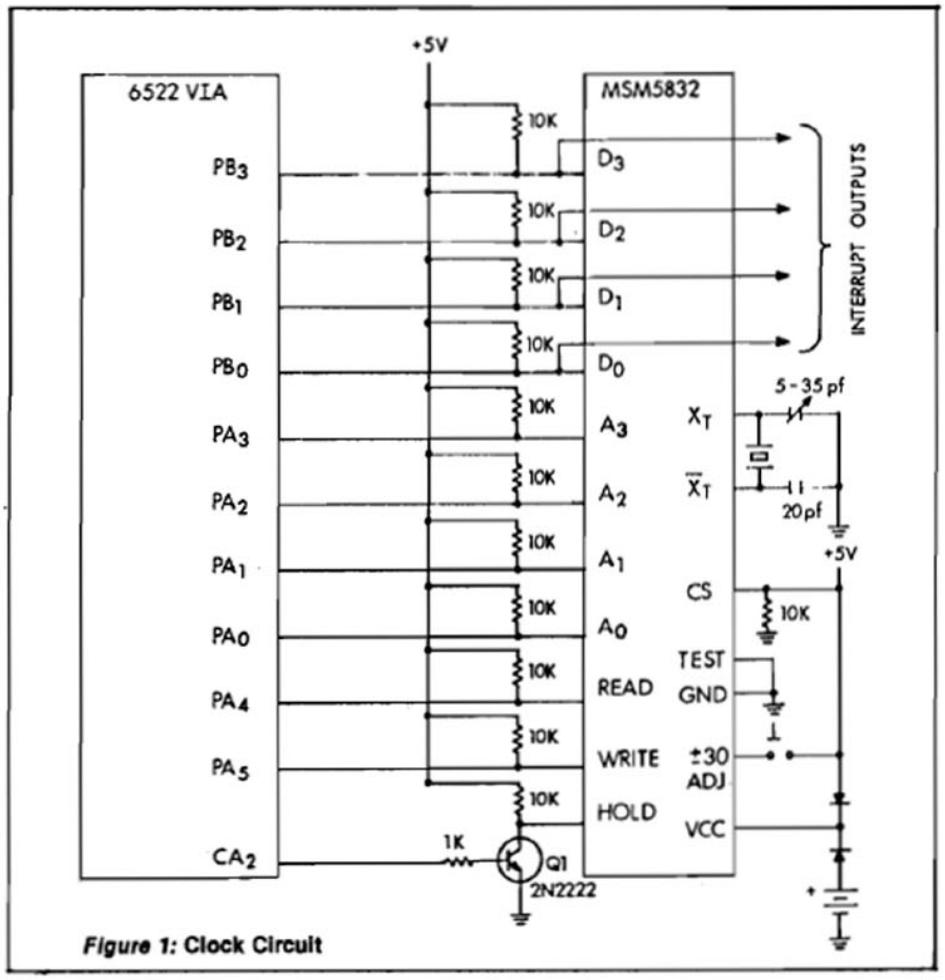

Figure 1: **Clock Circuit**

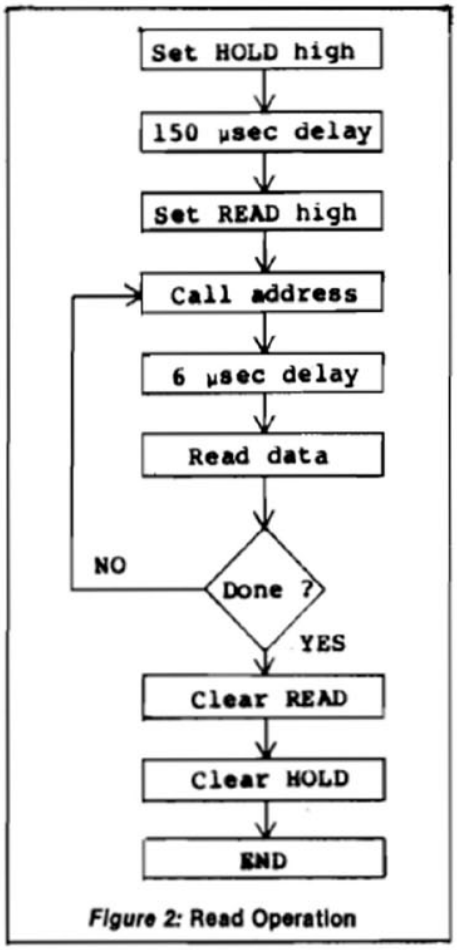

Figure 2: **Read Operation**

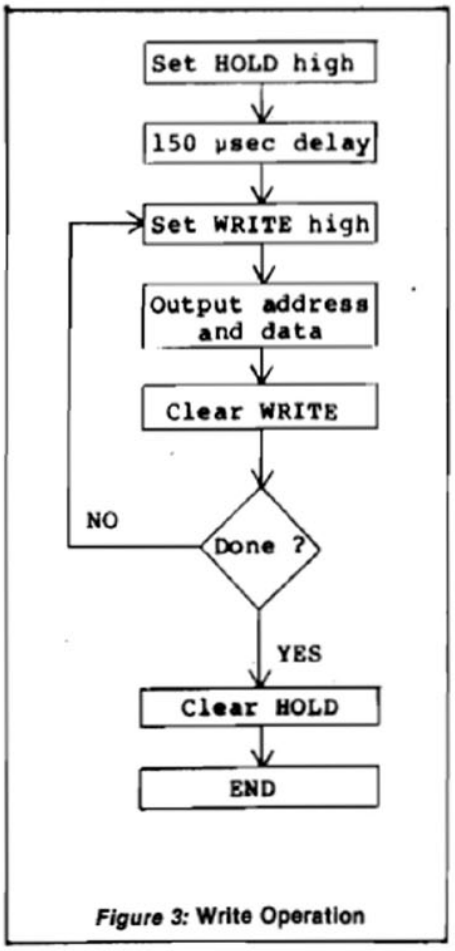

Figure 3: **Write Operation**

functions except the counting circuits, allowing very low power consumption while still keeping time by the battery backup.

Hold

A high on this line keeps the seconds counter from being incremented by the 1Hz clock output. After the initial set up time (150 microseconds), all counters will be in a static state, allowing error-free read and write operations as long as the HOLD time is less than one second. Other clock circuits do not have this feature, and operations have to be done twice and compared to assure no error has been made.

Consider the following example with a conventional clock circuit. Suppose you are reading a time of 12 hours and 59 minutes. If the seconds count should be 59, and after the hours (and before the minutes) are read, the seconds counter clears and sends a carry pulse, the time is then 13 hours, 0 minutes and 0 seconds. But the read operation has resulted in 12 hours, 0 minutes and 0 seconds—a full hour off. It is for this reason that with conventional clock circuits two reads have to be made to insure proper information has been received.

***Listing 1:* Machine Language Routines**

```
                    Machine language routines for MSM5832
                    Clock/Calendar circuit
                    Randy Sebra July, 1980

                  ACCESS EQ   $8B86  Un-write protect system RAM
                    ORB EQ    $A800  Output register B
                    IORA EQ   $A801  Input/output register A
                    DDRB EQ   $A802  Data direction register B
                    DDRA EQ   $A803  Data direction register A
                     PCR EQ   $A80C  Peripheral Control Register

                         ORG  $0FA7  Start of routines

                    Routine to configure Port B and
                    set HOLD high

0FA7-  20 86 8B SETUP  JSR   ACCESS  Remove write protect
0FAA-  A9 3F           LDA   #3F     Set up PA0-PA5 as outputs
0FAC-  8D 03 A8        STA   DDRA     for address and control
0FAF-  A9 0C           LDA   #0C     Set CA2 low for high
0FB1-  8D 0C A8        STA   PCR      input to HOLD
0FB4-  A0 25           LDY   #25     Delay 150 microsec for
0FB6-  88       DELAY  DEY           HOLD time set up
0FB7-  D0 FD           BNE   DELAY
0FB9-  60              RTS           Return

                    Read routine

0FBA-  20 A7 0F READ   JSR   SETUP   Set up HOLD
0FBD-  A9 00           LDA   #00     Configure PB0-PB3 as data
0FBF-  8D 02 A8        STA   DDRB     inputs for read
0FC2-  A2 0C           LDX   #0C     Initial address
0FC4-  8A       RDLOOP TXA           Transfer to accumulator and
0FC5-  09 10           ORA   #10     combine with READ high
0FC7-  8D 01 A8        STA   IORA    Issue RFAD
0FCA-  EA              NOP           Small delay
0FCB-  EA              NOP           for read access time
0FCC-  EA              NOP
0FCD-  AD 00 A8        LDA   ORB     Read data
0FD0-  29 0F           AND   #0F     Mask off high 4 bits
0FD2-  95 E9           STA   E9,X    Store to Page Zero
0FD4-  CA              DEX           Decrement address
0FD5-  10 ED           BPL   RDLOOP  Loop until through
0FD7-  A9 0E           LDA   #0E     Then set HOLD low
0FD9-  8D 0C A8        STA   PCR      by CA2 high
0FDC-  60              RTS           Return

                    Write routine

0FDD-  20 A7 0F WRITE  JSR   SETUP   Set up HOLD
0FE0-  A9 0F           LDA   #0F     Configure PB0-PB3 as
0FE2-  8D 02 A8        STA   DDRB     data outputs for write
0FE5-  A2 0C           LDX   #0C     Set initial address
0FE7-  B5 E9    WRLOOP LDA   E9,X    and fetch data
0FE9-  8D 00 A8        STA   ORB     Write data
0FEC-  8A              TXA           Combine address with
0FED-  09 20           ORA   #20     WRITE high and
0FEF-  8D 01 A8        STA   IORA    issue write
0FF2-  29 0F           AND   #0F     Toggle WRITE control
0FF4-  8D 01 A8        STA   IORA
0FF7-  CA              DEX           Decrement address
0FF8-  10 ED           BPL   WRLOOP  Loop until through
0FFA-  A9 0E           LDA   #0E     Set CA2 high for
0FFC-  8D 0C A8        STA   PCR      low on HOLD
0FFF-  60              RTS           Return
```

Read

This input, when taken to Vcc, signals a read operation.

Write

This input, when taken to Vcc, signals a write or set operation. This method of being able to directly set the time is far easier to use than conventional circuits in which pulses must be directed to either a fast set or slow set input, and the clock must read between pulses until the desired time has been set.

±30 Adjust

Momentarily taking this input to Vcc will reset the seconds count to zero. If the seconds count was 30 or more before this action, a carry is sent to the minutes counter. If less than 30, the minutes count remains unchanged. This means that keeping the time accurate is a very simple matter. If the switch in figure 1 is momentarily pressed at the start of a minute, this will automatically reset the time to the correct value as long as the clock is less than 30 seconds either fast or slow.

Test

This input allows testing of the operation of the clock. Pulses to this input will directly clock the S1, MI10, W, D1, or Y counters, depending on which one is addressed by A0-A3.

Reference Signal Outputs

Outputs are available from D0-D3 when READ, CS, and A0-A3 are at Vcc. These can be used as interrupts to the microprocessor. Table 2 presents the conditions for these signals.

Operation

Figures 2 and 3 present the flow diagrams for read and write operations. Although self-explanatory, there are several aspects of the operations which should be emphasized, especially for applications other than the specified one given in this article. First, the HOLD control must always be given at least 150 microseconds set up time, and *must* be used for WRITE operations. Next, since the read access time

of the chip may be as long a 6 microseconds, a delay must be built in before reading data. Additionally, notice that although the READ control may be held high for as many read operations as desired, the WRITE control must be pulsed between each write operation.

Interfacing

There are many ways which the MSM5832 can be interfaced with a 6502 or other microprocessor. The only requirement is eleven I/O lines, with four being bi-directional. For myself, the most convenient method was through the use of the #2 6522 VIA on my SYM-1. Figure 1 shows this configuration. If you do not have a 6522 available on your system, it is a relatively simple matter to add one. See "An Additional I/O Interface for the PET", by Kevin Erler, MICRO, December 1979 (19:40). This is also applicable for Apple.

Transistor *Q1* in figure 1 at the HOLD pin is used to invert the CA2 input to HOLD. The reason for this is as follows. On power up and reset, all registers in the 6522 are cleared. This causes all I/O lines to be configured as inputs with a high voltage on the pins, and the HOLD would be held high. When using a battery back up, this would cause the clock to stop until the HOLD is pulled low, since the hold time would always be longer than one second. With the HOLD being control - led separately by the CA2 output and inverted, this will always keep HOLD low unless intentionally taken high by software.

For battery back up, the chip select is connected to +5V from the 6502 bus, which disables all inputs and outputs when the system is off and the clock is on back up. The batteries used here are dry cells and the setup is a rather simple battery back up. A more elaborate setup could be used with NI-CADs and with the +5V trickle charging the batteries when the system is up. This could give many years of continuous operation before having to replace batteries. The battery life in both cases, of course, is a function of how frequently (or infrequently) the main system is used.

***Listing 2:* Basic Routine and Sample Run**

```
100 REM
110 REM      MSM5832 CLOCK/CALENDAR
120 REM      SET/READ PROGRAM
130 REM      RANDY SEBRA  JULY,1980
140 REM
150 DEF FNS(X)=INT(X/10)
160 DEF FNT(Y)=Y-FNS(Y)*10
170 INPUT"SET(S) OR READ(R) ? ";I$
180 IF I$<>"S" THEN 430
190 REM
200 REM      GET INPUT AND STORE INTO LOCATIONS $E9-$F5
210 REM
220 INPUT"MONTH,DAY,YEAR(2 DIGITS) ? ";M2,D,Y
230 POKE 245,FNS(Y)
240 POKE 244,FNT(Y)
250 POKE 243,FNS(M2)
260 POKE 242,FNT(M2)
270 POKE 241,FNS(D)
280 POKE 240,FNT(D)
290 INPUT"DAY OF THE WEEK(1-7) ? ";W
300 POKE 239,W-1
310 INPUT"HOURS,MINUTES(24 HOUR TIME) ? ";H,M
320 POKE 238,FNS(H)+8
330 POKE 237,FNT(H)
340 POKE 236,FNS(M)
350 POKE 235,FNT(M)
360 REM
370 REM      CALL MACHINE LANGUAGE WRITE ROUTINE
380 REM
390 S=USR(&"0FDD",0)
400 REM
410 REM      CALL MACHINE LANGUAGE READ ROUTINE
420 REM
430 R=USR(&"0FBA",0)
440 DIM D$(6),M$(11)
450 DEF FNR(I)=PEEK(I)*10+PEEK(I-1)
460 DATA SUNDAY,MONDAY,TUESDAY,WEDNESDAY,THURSDAY,FRIDAY,SATURDAY
470 DATA JANUARY,FEBRUARY,MARCH,APRIL,MAY,JUNE,JULY,AUGUST,SEPTEMBER
480 DATA OCTOBER,NOVEMBER,DECEMBER
490 FOR I=0 TO 6
500 READ D$(I)
510 NEXT I
520 FOR I=0 TO 11
530 READ M$(I)
540 NEXT I
550 REM
560 REM      CONVERT PAGE ZERO DATA INTO APPROPRIATE UNITS
570 REM
580 Y=FNR(245)+1900
590 M2=FNR(243)
600 D=FNR(241)
610 W=PEEK(239)
620 REM
630 REM      CONVERT HOURS, MINUTES, AND SECONDS INTO A
640 REM      STRING FOR "CLEANER" OUTPUT"
650 REM
660 T$=""
670 FOR I=0 TO 4
680 T$=T$+RIGHT$(STR$(PEEK(237-I)),1)
690 NEXT I
700 T$=RIGHT$(STR$(PEEK(238)-8),1)+T$
710 T$=LEFT$(T$,2)+":"+MID$(T$,3,2)+":"+RIGHT$(T$,2)
720 PRINT"TODAY IS ";D$(W);" ";M$(M2-1);D;Y;T$
730 END
OK
RUN
SET(S) OR READ(R) ? S
MONTH,DAY,YEAR(2 DIGITS) ? 7,18,80
DAY OF THE WEEK(1-7) ? 6
HOURS,MINUTES(24 HOUR TIME) ? 13,8
TODAY IS FRIDAY JULY 18  1980 13:08:00

OK
RUN
SET(S) OR READ(R) ? R
TODAY IS FRIDAY JULY 18  1980 13:09:10
```

Machine Language Routines

Listing 1 presents machine language routines to set the clock/calendar and read the time. As shown by figure 1, PA0-PA3 go to address lines A0-A3, PA4-PA5 go to the READ and WRITE, PB0-PB3 go to data lines D0-D3, and CA2 is inverted and goes to the HOLD input.

Data to be written to the clock, and the data received from a read are stored in Page Zero locations $E9-$F5. These are "safe" Page Zero locations which are not used either by BASIC nor by the SYM monitor. For computers other than the SYM, other locations may have to be used, but virtually all 6502 computers will have Page Zero locations available.

The routines themselves are general routines which may be used for any 6502 computer since they do not use any monitor routines, except the routine necessary to remove the write-protect from system RAM. Of course, the locations for the 6522 will probably be different. The machine language is located so as to occupy the highest part of memory in a 4K system. They can be easily relocated, with the only changes required being the JSR's at locations $0FBA and $0FDD in listing 1.

Applications

The obvious use for the clock/calendar interface is setting the time and getting the time output upon request. Using the machine language routines in listing 1 in conjunction with a BASIC driver is perhaps the most convenient method of accomplishing this. Listing 2 is an example of one such BASIC program, along with two typical resultant runs. The program, in order to set the clock/calendar, merely requests the necessary data and stores it into the proper Page Zero locations and then calls the machine language routine to do the actual setting. To insure that the input data was correct, a read is done after the setting as a check. For the read operation, the program calls the machine language to do the actual reading, and then merely arranges the data obtained from $E9-$F5 to be output in a convenient manner.

The memory size of 4006 is for a 4K system. The dummy tape save, SAVE D, is needed to overcome a bug in SYM BASIC. The program is loaded in as file "C". The machine language routines were saved as file $4D, so that they can be loaded by the LOAD M command.

As mentioned previously, the clock can generate interrupts to the microprocessor. Since we are using a 6522 VIA in the interface, either of the two on-board timers can be used to generate precise interrupts of up to 0.65 seconds apart. With the MSM5832, we then can generate interrupts at one second, one minute, or one hour apart (see figure 1 and table 2).

Listing 3 presents a machine language routine to use any one of these three interrupts. Although the D1, D2 or D3 outputs could be tied directly to the $\overline{IRQ}$ line of the 6502 system, in this example one of the outputs goes to the CB2 input of the 6522. The routine has been set up so that the interrupt occurs on the negative-going edge of the 122.1 microsecond pulse. If setting on the positive edge is desired, merely write a $20 in the PCR register of the 6522. In whatever interrupt routine used with this setup, the IFR register bit can be cleared, either by directly writing into the IFR, or by reading or writing to Port B. This may be accomplished by reading the time with the routines in listing 1. Note, however, that reading the time reconfigures Port A, and this must be reset to the configuration in listing 3.

An obvious use of this type of operation would be to use an interrupt of one second when using a video display or terminal with an addressable cursor, to continually write the time in the upper right hand corner of the screen. A much more effective use would be in a polled environment where it would be desirable to get data from input ports or status of peripherals every second, minute, or hour.

Special note: The MSM5832 is a fairly new chip, first introduced in the first quarter of 1980. For this reason, it is not commonly available, except in quantity from the manufacturer. If there is sufficient reader interest in using the circuit described in this article, I can supply the chip and the 32.768 KHz crystal for $17 plus postage. Delivery may take 4 to 6 weeks.

***Listing 3:* Interrupt Set-up Routines**

```
                    Machine language routines for
                    Setting up interrupts from the MSM5832
                    Randy Sebra July, 1980

                ACCESS EQ   $8B86  Un-write protect system RAM
                   ORA EQ   $A801  Input/output register A
                  DDRB EQ   $A802  Data direction register B
                  DDRA EQ   $A803  Data direction register A
                   IFR EQ   $A80D  Interrupt flag register
                   IER EQ   $A80E  Interrupt enable register

                        ORG  $0F8C  Start of routine

0F8C-  20 86 8B SETUP   JSR  ACCESS  Remove write protect RAM
0F8F-  A9 00            LDA  #00     Set up B0-B3 (data
0F91-  8D 02 A8         STA  DDRB    lines) as inputs
0F94-  A9 1F            LDA  #1F     Set up A0-A4 (address
0F96-  8D 03 A8         STA  DDRA    lines and READ) as
0F99-  8D 01 A8         STA  ORA     outputs and set all high
0F9C-  A9 08            LDA  #08     Set up IFR for interrupt
0F9E-  8D 0D A8         STA  IFR     from CB2
0FA1-  A9 88            LDA  #88     Enable interrupt for
0FA3-  8D 0E A8         STA  IER     CB2
0FA6-  60               RTS          Return
```