Application Note 12

Title: **Memory Requirements for Clock Applications**

Product Family: **4 bits Microcontroller**

Part Number: EM6603, EM6503
Keywords: 4 bits Microcontroller, Memory Requirements, Alarm Clock
Date: January 07, 2002 REV. B/416

Introduction

EM's 4 bits uC family is based on a 4-bit core with a RISC-like architecture. As it is typical for RISC processors all the instructions have a fixed length (16 bits for the 4 bits uC), are all executed in the same number of clock cycles (2 cycles for the 4 bits uC) and use a relatively small but powerful set of instructions.

As a new microcontroller the 4 bits uC has to compete with quite a number of existing 4-bit microcontrollers and is compared to them in terms of speed, memory requirements, features and cost effectiveness. Writing programs for the 4 bits uC involves more than just translating existing programs for 4-bit controllers into 4 bits uC code. To take full advantage of the 4 bits uC architecture the application program's architecture and design has to be optimised likewise, as can be demonstrated by the sample application described in this application note.

Sample Program

The original program in this example was written for an another microcontroller, which has a traditional architecture and uses segmented data and program memories. The instruction set is rather larger and has many register-based instructions.

The program implements a simple clock, counting seconds, minutes and hours. It compares the current time against a predefined termination time and stops when the current time equals the termination time. These functions are typical for applications such as alarm clocks.

Results

The program assembles into 132 instructions. Before this program was translated into a 4 bits uC program, some people who until now worked only with an another processors were asked about what program size they would expect for the 4 bits uC. The estimates were in the range between 1.5 and 2 times the number of instructions required for the another processor.

After translating and optimising the program for the 4 bits uC, it turned out that the 4 bits uC needs only 107 instructions. **The 4 bits uC program size is only 80% of the another program !** This is very encouraging and together with the lean 4 bits uC CPU this may be taken as an indication that the 4 bits uC is capable of delivering very cost effective solutions.

EM66xx Programming Hints

There are several things we can learn from this sample project with respect to the design of optimised programs for the 4 bits uC.

- Do not simply translate an existing program. Optimise the program architecture to match the 4 bits uC

architecture.

- For the 4 bits uC it is sometimes more efficient to use specialised inline code than general purpose subroutine calls, especially if such subroutines would be called from only a few places in your main program.

- Design subroutines application-specifically, i.e. do not pack general purpose features into a subroutine that might we useful for some other applications, but are not required by the current application.

- When using multiple instances of equal data structures (types) in RAM which have to be accessed indirectly, align each instance on a page boundary to simplify index register operations.

- On the 4 bits uC complex programs are probably easier to implement when a state machine model is used for the application. Such a program should be organised around a central event loop.

Test Program Assembler Listing

The following is the 4 bits uC assembler output listing. The listing of the original program is available as a separate document.

```
:;Clock Evaluation Program
:;========================

:;FILE:          Eval.asm
:;VERSION:       1.00
:;DATE:          May 2, 1995

:;PROCESSOR:     EM V6603
:;LANGUAGE:      ASM_EM66

:;SUMMARY:       This program implements a simple clock, counting
:;               seconds, minutes and hours. It compares the current
:;               time against a predefined termination time and stops
:;               when the current time equals the termination time.
:;               The program does not produce any output, but its
:;               operation can be monitored with an appropriate
:;               debugging tool.

:;               The sole purpose of this program is to demonstrate
:;               the basics required to build a simple alarm clock
:;               program and to evaluate the memory requirements of
:;               the V6602 as compared to other CPUs.

:;               The program also contains some optional code that
:;               produces a beep every second and visualizes the
:;               contents of some registers. In order to enable this
:;               option, the Option parameter must be set to 1 below.
:;               Setting the Option parameter to 0 disables this option.
```

```
33    V00000000              :Option  EQU     0
```



```
                    :;REGISTER DECLARATIONS
                    :;--------------------------------------------------------------------

40    V00000060             :LTimLS  EQU 060H ;timer register low, starts counting,
                    :                 ;read before HTimLS, always read HTimLS after LTimLS
42    V00000061             :HTimLS  EQU 061H ;timer register high, must be written before LTimLS
                    :                 ;always read after reading LTimLS
44    V00000062             :TimCtr  EQU 062H ;timer control reg., must be written before LTimLS
45    V00000008             :TimAuto EQU 08H  ;mask, '1' enables auto reload of timer, also
                    :                 ; cleared by writing 0 into LTimLS
47    V00000007             :TEC     EQU 07H  ;mask, timer/event counter mode:
48    V00000000             :TEC_off EQU 00H  ; not active
49    V00000001             :TEC2048 EQU 01H  ; 2048 Hz input
50    V00000002             :TEC_512 EQU 02H  ;  512 Hz input
51    V00000003             :TEC_128 EQU 03H  ;  128 Hz input
52    V00000004             :TEC_32  EQU 04H  ;   32 Hz input
53    V00000005             :TEC_8   EQU 05H  ;    8 Hz input
54    V00000006             :TEC_1   EQU 06H  ;    1 Hz input
55    V00000007             :TEC_PA3 EQU 07H  ; PA3 input

57    V0000006C             :SVLD    EQU 06CH ;supply voltage level detector
58    V00000003             :VLC     EQU 03H  ;mask, select test voltage:
59    V00000000             :VLC_0   EQU 00H  ; inactive
60    V00000001             :VLC_1   EQU 01H  ; 1.38 V
61    V00000002             :VLC_2   EQU 02H  ; 2.00 V
62    V00000003             :VLC_3   EQU 03H  ; 2.50 V
63    V00000004             :SVLDbsy EQU 04H  ;mask (read only), '1' indicates busy during test
64    V00000008             :VLDR    EQU 08H  ;mask (read only), '1' indicates low supply voltage

66    V0000006D             :CIRQD   EQU 06DH ;interrupt and debouncer control register
67    V00000001             :INTEN   EQU 01H  ;mask, '1' enables CPU interrupts generally
68    V00000002             :DebCK   EQU 02H  ;mask, sets debouncer clock time:
69    V00000002             :DebCK2  EQU 02H  ;  2 msec debouncer time
70    V00000000             :DebCK16 EQU 00H  ; 16 msec debouncer time

72    V0000006E             :XL      EQU 06EH ;index register low (110 decimal)
73    V0000006F             :XH      EQU 06FH ;index register high (111 decimal)
74    V00000007             :XHmask  EQU 07H  ;XH mask

76    V00000070             :IntRq   EQU 070H ;Interrupt request register
                    :                 ;reading IntRq clears INTPR and INTTE
78    V00000004             :SLEEP   EQU 04H  ;mask (write only), '1' starts SLEEP mode if
                    :                 ; WD.Slmask = 1
80    V00000001             :INTPA   EQU 01H  ;mask (read only), Port A, cleared by reading IRQpA
81    V00000002             :INTPC   EQU 02H  ;mask (read only), Port C, cleared by reading IRQpC
82    V00000004             :INTTE   EQU 04H  ;mask (read only), Timer/Event Counter
83    V00000008             :INTPR   EQU 08H  ;mask (read only), Prescaler
85    V00000071             :WD      EQU 071H ;Watchdog timer control register
86    V00000008             :WDRST   EQU 08H  ;mask (write only), '1' resets watchdog timer
87    V00000004             :SLmask  EQU 04H  ;mask, '1' enables writing to IntRq.SLEEP
                    :                 ;cleared by POR, not cleared by RESET
89    V00000003             :WDdata  EQU 03H  ;mask (read only), watchdog timer data 0..3 seconds
                    :                 ;incremented at a rate of 1Hz

92    V00000072             :PortA   EQU 072H ;(read only) port A data register
                    :                 ;PA0..2 connected to TestVar1..3
                    :                 ;PA3 connected to event counter
```



```
95    V00000073             :IRQpA   EQU 073H ;(read only), interrupt flags for PA0..3, cleared by
                    :                ;reading this register or initial reset
97    V00000074             :MPortA  EQU 074H ;interrupt mask register for PA0..3

99    V00000075             :PortB   EQU 075H ;port B data register
100   V00000076             :CIOPortB EQU 076H ;IO control register for each bit of PB0..3,
                    :                ;PB0 can be used as buzzer output
                    :                ;the following bits are flags set by the hardware:
103   V00000001             :EnbOut0 EQU 01H  ;mask, '1' enables output mode
104   V00000002             :EnbOut1 EQU 02H  ;mask
105   V00000004             :EnbOut2 EQU 04H  ;mask
106   V00000008             :EnbOut3 EQU 08H  ;mask

108   V00000077             :PortC   EQU 077H ;port C data register, IO controlled by CPIOB.CIOPC
109   V00000078             :IRQpC   EQU 078H ;(read only), interrupt flags for PC0..3, cleared by
                    :                ;reading this register or initial reset
111   V00000079             :MPortC  EQU 079H ;interrupt mask register for PC0..3

113   V0000007A             :PortD   EQU 07AH ;port D data register, IO controlled by CPIOB.CIOPD

115   V0000007C             :CPIOB   EQU 07CH ;IO control (pC,pD) and interrupt mode (pA,pC) flags
116   V00000001             :PA_C    EQU 01H  ;mask, if set to '1' interrupts from PA and PC are
                    :                ;served only if INTPA = 1 AND INTPC = 1
118   V00000002             :CIOPC   EQU 02H  ;mask, '1' enables output mode for port C
119   V00000004             :CIOPD   EQU 04H  ;mask, '1' enables output mode for port D

121   V0000007D             :PRESC   EQU 07DH ;Prescaler control register
122   V00000003             :PSF     EQU 03H  ;mask for interrupt frequency selection:
123   V00000000             :PSF_0   EQU 00H  ; no interrupt
124   V00000001             :PSF_1   EQU 01H  ;  1 Hz interrupt frequency
125   V00000002             :PSF_8   EQU 02H  ;  8 Hz interrupt frequency
126   V00000003             :PSF_32  EQU 03H  ; 32 Hz interrupt frequency
127   V00000008             :MTim    EQU 08H  ;mask, '1' enables timer/event counter interrupt
request
128   V00000004             :PRST    EQU 04H  ;mask (write only), '1' resets prescaler

130   V0000007E             :BEEP    EQU 07EH ;Buzzer control register
131   V00000003             :BCF     EQU 03H  ;mask, beep frequency:
132   V00000000             :BCF_0   EQU 00H  ; silence
133   V00000001             :BCF_1   EQU 01H  ; 1024 Hz
134   V00000002             :BCF_2   EQU 02H  ; 2048 Hz
135   V00000003             :BCF_3   EQU 03H  ; 2667 Hz
136   V00000004             :BUen    EQU 04H  ;mask, '1' enables buzzer output on port PB0,
                    :                ;overrides CIOPortB.0
138   V00000008             :TimEn   EQU 08H  ;mask, '1' enables timer/event counter countdown
                    :                ;mask, '0' disables timer/event counter function

141   V0000007F             :TEST    EQU 07FH ;
```



```
                    :;CONSTANTS
                    :;---------
147   V00000000             :FALSE          EQU 0H
148   V0000000F             :TRUE           EQU 0FH
                    :;VARIABLES
                    :;---------
                    :;NOTE:
                    :;The clock, clock copy and termination time structures have to be
                    :;aligned on page boundaries in the data RAM to optimize indexed copy
                    :;and compare operations. The page size is defined to be 16 nibbles.
                    :;Thus the low nibble of the index register (XL) can remain the same
                    :;when any of the data structures is accessed by indexed copy and
                    :;compare operations.

160   V00000000             :SecOffs        EQU 0
161   V00000002             :MinOffs        EQU 2
162   V00000004             :HourOffs       EQU 4
163   V00000010             :PgSize         EQU 16
165   V00000000             :TimeH          EQU 0H              ;clock data address, high nibble
166   V00000000             :SecL           EQU (PgSize * TimeH) + SecOffs  ;seconds, low nibble
167   V00000001             :SecH           EQU SecL + 1            ;clock seconds, high nibble
168   V00000002             :MinL           EQU SecL + MinOffs      ;clock minutes, low nibble
169   V00000003             :MinH           EQU MinL + 1            ;clock minutes, high nibble
170   V00000004             :HourL          EQU SecL + HourOffs     ;clock hours, low nibble
171   V00000005             :HourH          EQU HourL + 1           ;clock hours, high nibble


174   V00000006             :IntDone        EQU SecL + 6            ;flag to indicate that
                    :                                       ;interrupt has been served
176   V00000007             :DelayReg1_0    EQU SecL + 7            ;parameter for delay routine,
                    :                                       ;nibble 0 (LS)
178   V00000008             :DelayReg1_1    EQU DelayReg1_0 + 1     ;nibble 1
179   V00000009             :DelayReg1_2    EQU DelayReg1_0 + 2     ;nibble 2 (MS)

181   V00000001             :CopyH          EQU 1H          ;clock copy data address, high nibble
182   V00000010             :Copy_SecL      EQU (PgSize * CopyH) + SecOffs  ;clock copy seconds,
                    :                                       ;low nibble
184   V00000011             :Copy_SecH      EQU Copy_SecL + 1       ;high nibble
185   V00000012             :Copy_MinL      EQU Copy_SecL + MinOffs ;clock copy minutes,
                    :                                       ;low nibble
187   V00000013             :Copy_MinH      EQU Copy_MinL + 1       ;high nibble
188   V00000014             :Copy_HourL     EQU Copy_SecL + HourOffs ;clock copy hours,
                    :                                       ;low nibble
190   V00000015             :Copy_HourH     EQU Copy_HourL + 1      ;high nibble

192   V00000002             :TestH          EQU 2H                  ;termination time data addr,
                    :                                       ;high nibble
194   V00000020             :Test_SecL      EQU (PgSize * TestH) + SecOffs  ;term. time seconds,
                    :                                       ;low nibble
196   V00000021             :Test_SecH      EQU Test_SecL + 1       ;high nibble
197   V00000022             :Test_MinL      EQU Test_SecL + MinOffs ;termination time minutes,
                    :                                       ;low nibble
199   V00000023             :Test_MinH      EQU Test_MinL + 1       ;high nibble
200   V00000024             :Test_HourL     EQU Test_SecL + HourOffs ;termination time hours,
                    :                                       ;low nibble
202   V00000025             :Test_HourH     EQU Test_HourL + 1      ;termination time hours,
                    :                                       ;high nibble
                    :;PROCEDURES
```



```
                    :;----------
                    :               ORG 0

                    :Reset:
                    :;=====
                    :;Dispatches to the reset and interrupt service routines.

215   0000  0006     :               JMP Init                ;hardware and watchdog reset
216   0001  0002     :               JMP Handler             ;interrupt service


                    :Handler:
                    :;=======
                    :;Interrupt service routine. Interrupts are generated by prescaler.
                    :;This subroutine resets the watchdog timer and serves the timer
                    :;interrupt by posting an IntDone message for the event loop in the
                    :;main program.

226   0002  C871     :               STI WD, WDRST           ;reset watchdog
227   0003  B870     :               LDR IntRq               ;clear interrupt request
228   0004  CF06     :               STI IntDone, TRUE
229   0005  F27F     :               RTI

                    :Init:
                    :;====
                    :;This section initializes the data structures in RAM.

                    :;Set Clock to 23:59:00

237   0006  C000     :               STI SecL, 0
238   0007  C001     :               STI SecH, 0
239   0008  C902     :               STI MinL, 9
240   0009  C503     :               STI MinH, 5
241   000A  C304     :               STI HourL, 3
242   000B  C205     :               STI HourH, 2

                    :;Set Termination Time to 00:00:00

246   000C  C020     :               STI Test_SecL, 0
247   000D  C021     :               STI Test_SecH, 0
248   000E  C022     :               STI Test_MinL, 0
249   000F  C023     :               STI Test_MinH, 0
250   0010  C024     :               STI Test_HourL, 0
251   0011  C025     :               STI Test_HourH, 0


                    :Main:
                    :;====
256   0012  E025     :               CALL Copy_Data          ;copy clock data to buffer
                    :                                       ;structure
                    :               IF Option <> 0
                    :                CALL Present           ;optional presentation
                    :                                       ;(beep, display)
                    :               ENDIF
262   0013  E02E     :               CALL Test_Data          ;compare clock buffer to
                    :                                       ;termination time
```



```
264     0014  C006      :                STI IntDone,FALSE       ;clear IntDone message
265     0015  C16D      :                STI CIRQD, INTEN        ;enable interrupts
266     0016  C57D      :                STI PRESC, PSF_1 OR PRST ;set prescaler rate to 1 Hz,
                        :                                        ;reset prescaler
                        :Timer:
                        :;=====
                        :;This section increments the clock

274     0017  B806      :Do:             LDR IntDone             ;loop until interrupt handler
                        :                                        ;sends IntDone message
276     0018  6017      :                JPZ Do

                        :                                        ;-- increment seconds
279     0019  C06F      :                STI XH, TimeH           ;set index register high (XH)
                        :                                        ;to clock structure
281     001A  C06E      :                STI XL, SecOffs         ;set index register low (XL)
                        :                                        ;to seconds
283     001B  E03A      :                CALL IncM60             ;increment 2 nibbles mod 60
284     001C  7012      :                JPNZ Main               ;Z = 0 indicates that
                        :                                        ;60 seconds have elapsed

                        :;if 60 seconds have elapsed then ..
                        :;-- increment minutes and wait 100 ms
289     001D  E05E      :                CALL Delay              ;wait 100 ms
290     001E  C26E      :                STI XL, MinOffs         ;set index register low (XL)
                        :                                        ;to minutes
292     001F  E03A      :                CALL IncM60             ;increment 2 nibbles mod 60
293     0020  7012      :                JPNZ Main

                        :;if 60 minutes have elapsed then ..
                        :;-- increment hours and wait 100 ms
297     0021  E05E      :                CALL Delay              ;wait 100 ms
298     0022  C46E      :                STI XL, HourOffs        ;set index register low (XL)
                        :                                        ;to hours
300     0023  E049      :                CALL IncM24             ;increment 2 nibbles mod 24


303     0024  0012      :                JMP Main                ;go to begin of main loop


                        :Copy_Data:
                        :;=========
                        :;This subroutine copies a time structure (6 nibbles)
                        :;from TimeH to CopyH.

311     0025  F805      :                LDI HourOffs + 1        ;set A to nibble count - 1


                        :Copy_Nibbles:
                        :;============
                        :;Copy A + 1 nibbles from TimeH to CopyH.

318     0026  D06E      :                STA XL                  ;set index register low
319     0027  C06F      :                STI XH, TimeH           ;set index register high
                        :                                        ;to source
321     0028  B8FF      :                LDRX                    ;get nibble
322     0029  C16F      :                STI XH, CopyH           ;set index register high
                        :                                        ;to destination
```



```
324     002A  D0FF      :                STAX                    ;store nibble
325     002B  986E      :                DEC XL                  ;get index register and
                        :                                        ;decrement A
327     002C  4026      :                JPC Copy_Nibbles        ;loop until done (A < 0)
328     002D  FA7F      :                RET
                        :Test_Data:
                        :;=========
                        :;This subroutine compares two time structure (6 nibbles)
                        :;from TestH and CopyH.
                        :;If both structures have equal values, the interrupts are disabled
                        :;and the processor is put into halt mode. It returns from halt mode
                        :;when the watchdog timer expires

339     002E  F805      :                LDI HourOffs + 1

                        :Test_Nibbles:
                        :;============
                        :;Compares A + 1 nibbles from TestH and CopyH.

345     002F  D06E      :                STA XL                  ;set index register low
346     0030  C26F      :                STI XH, TestH           ;set index register high
                        :                                        ;to operand_1
348     0031  B8FF      :                LDRX                    ;get nibble
349     0032  C16F      :                STI XH, CopyH           ;set index register high
                        :                                        ;to operand_2
351     0033  8CFF      :                SUBX                    ;calc. operand_2 - operand_1
352     0034  7039      :                JPNZ Test_End           ;if not equal then exit
353     0035  986E      :                DEC XL                  ;get index register and
                        :                                        ;decrement A
355     0036  402F      :                JPC Test_Nibbles        ;loop until done (A < 0)

357     0037  C06D      :                STI CIRQD, 0            ;disable interrupts
358     0038  F4FF      :                HALT                    ;enter HALT mode.
                        :                                        ;Watchdog timer returns
                        :                                        ;control to RESET vector.
361     0039  FA7F      :Test_End:       RET


                        :IncM60:
                        :;======
                        :;This subroutine increments 2 nibbles modulo 60.

                        :;Input Parameters:      X register:     points to the least
                        :;                                       significant nibble
                        :;Output Parameters:     Z flag:         cleared when the result is 0.

372     003A  86FF      :                INCX                    ;get low nibble and increment
373     003B  D0FF      :                STAX                    ;store low nibble
374     003C  F80A      :                LDI 10                  ;if nibble < 10
375     003D  8CFF      :                SUBX
376     003E  7048      :                JPNZ IncM60End          ;then return

378     003F  C0FF      :                STIX 0                  ;else clear low nibble
379     0040  866E      :                INC XL                  ;get index and increment
380     0041  D06E      :                STA XL                  ;store index
381     0042  86FF      :                INCX                    ;get high nibble & increment
```



```
382    0043  D0FF    :               STAX                    ;store high nibble
383    0044  F806    :               LDI 6                   ;if nibble < 6
384    0045  8CFF    :               SUBX
385    0046  7048    :               JPNZ IncM60End          ;then return

387    0047  C0FF    :               STIX 0                  ;else clear high nibble
                     :
389    0048  FA7F    :IncM60End:     RET
                     :IncM24:
                     :;======
                     :;This subroutine increments 2 nibbles modulo 24

                     :;Input Parameters:     X register:     points to the least
                     :;                                      significant nibble
                     :;Output Parameters:    Z flag:         cleared when the result is 0.

399    0049  86FF    :               INCX                    ;get low nibble and increment
400    004A  D0FF    :               STAX                    ;store low nibble
401    004B  F80A    :               LDI 10                  ;if nibble < 10
402    004C  8CFF    :               SUBX
403    004D  7052    :               JPNZ IncM24Test         ;then check modulo 24

405    004E  C0FF    :               STIX 0                  ;else clear low nibble
406    004F  C56E    :               STI XL, HourOffs + 1    ;set index to high nibble
407    0050  86FF    :               INCX                    ;get high nibble & increment
408    0051  D0FF    :               STAX                    ;store high nibble


                     :IncM24Test:
                     :;Check modulo 24
413    0052  C56E    :               STI XL, HourOffs + 1    ;set index to high nibble
414    0053  F802    :               LDI 2                   ;if nibble < 2
415    0054  8CFF    :               SUBX
416    0055  705D    :               JPNZ IncM24End          ;then return
417    0056  C46E    :               STI XL, HourOffs        ;else set index to low nibble
418    0057  F804    :               LDI 4                   ;if nibble < 4
419    0058  8CFF    :               SUBX
420    0059  705D    :               JPNZ IncM24End          ;then return
421    005A  C0FF    :               STIX 0                  ;else clear low nibble
422    005B  C56E    :               STI XL, HourOffs + 1    ;set index to high nibble
423    005C  C0FF    :               STIX 0                  ;clear high nibble

425    005D  FA7F    :IncM24End:     RET


                     :Delay:
                     :;=====
                     :;This subroutine delays program excution by 100 msec

432    005E  C007    :               STI DelayReg1_0, 0      ; 100 msec
433    005F  CF08    :               STI DelayReg1_1, 15
434    0060  C109    :               STI DelayReg1_2, 1
                     :

                     :Delay_Next:
                     :;==========
                     :;For a master oscillator frequency of 32768 Hz, this subroutine
                     :;delays program execution by an arbitrary value of between 625 usec
```



```
                    :;and approximately 768 msec. The delay may be specified in steps of
                    :;187.5 usec by setting DelayReg1_ based on the following equation:

                    :;       Delay/usec = (273 * R2 + 17 * R1 + R0) * 187.5 + 625

446   0061  9807    :                DEC DelayReg1_0         ;get low nibble and decrement
447   0062  D007    :                STA DelayReg1_0         ;store low nibble
448   0063  4061    :                JPC Delay_Next          ;loop until nibble < 0
449   0064  9808    :                DEC DelayReg1_1         ;get next nibble & decrement
450   0065  D008    :                STA DelayReg1_1         ;store nibble
451   0066  4061    :                JPC Delay_Next          ;loop until nibble < 0
452   0067  9809    :                DEC DelayReg1_2         ;get next nibble & decrement
453   0068  D009    :                STA DelayReg1_2         ;store nibble
454   0069  4061    :                JPC Delay_Next          ;loop until nibble < 0
455   006A  FA7F    :                RET
                    :Present:
                    :;=======
                    :;This (optional) section generates a short beep each second and
                    :;visualizes some registers.

                    :              IF Option <> 0
                    :               ;enable ports B,C,D for output
                    :               STI CIOPortB, EnbOut0 OR EnbOut1 OR EnbOut2 OR EnbOut3
                    :               STI CPIOB, CIOPC OR CIOPD
                    :
                    :               LDR MinL                ;get minutes low
                    :               STA PortB               ;output port B
                    :               LDR SecH                ;get seconds high
                    :               STA PortC               ;output port C
                    :               LDR SecL                ;get seconds low
                    :               STA PortD               ;output port D
                    :
                    :               STI BEEP, BCF_3         ;set buzzer to 2667 Hz,
                    :                                       ;start beeping
                    :               CALL Delay              ;wait 100 msec
                    :               STI BEEP, BCF_0         ;set buzzer to 0 Hz,
                    :                                       ;stop beeping
                    :               RET
                    :              ENDIF
```


