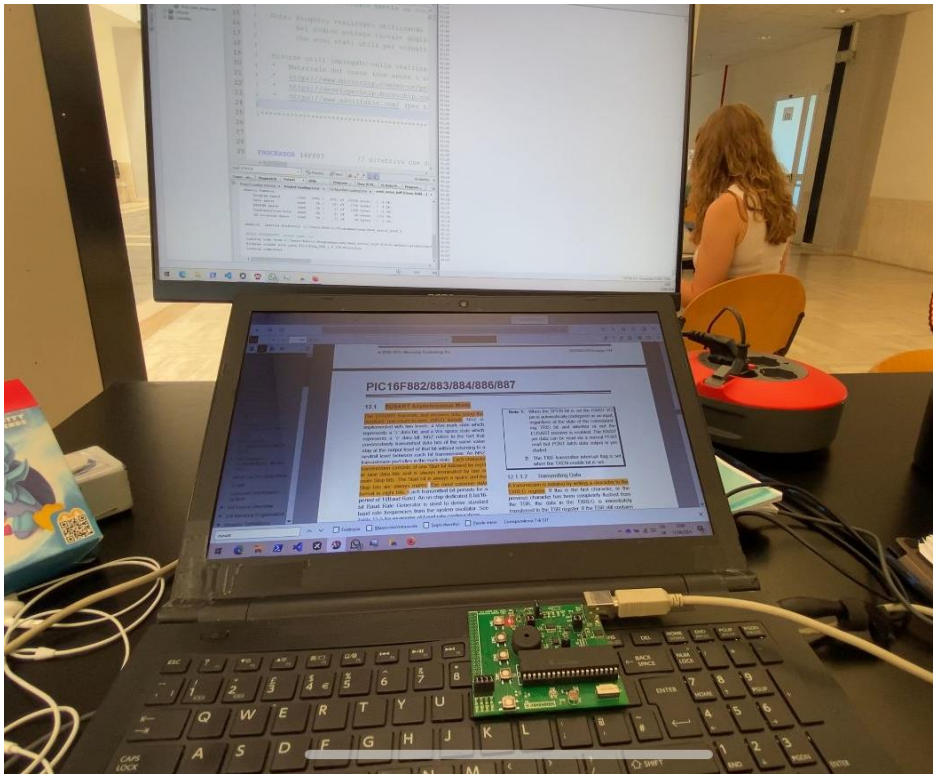
Traccia 14

Corso di Architetture e Progettazione di Sistemi Digitali/Sistemi Elettronici

(A.A. 2023-2024)

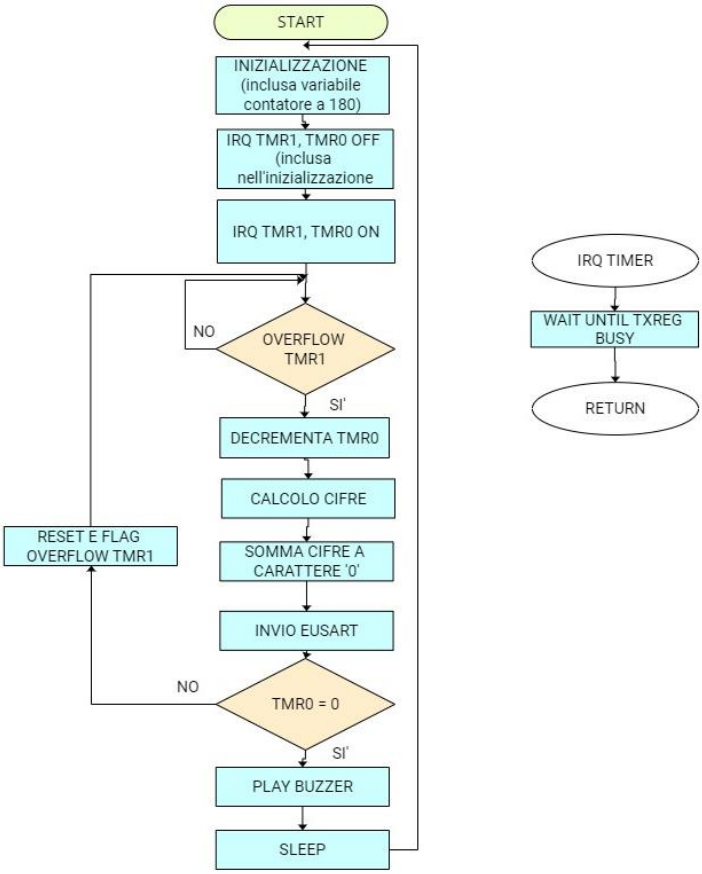


Specifiche assegnate

- Hardware: PIC16F887 (vedere documentazione scheda di riferimento PICBOARD)
- Ambiente di sviluppo: Microchip MPLAB X IDE
- Linguaggio: Assembly
- Gestione eventi: microcontrollore in modalità sleep (se possibile) in assenza di eventi da processare

Descrizione della tesina: si realizzi un firmware che implementi un countdown (valore iniziale 3 minuti) con risoluzione di 1 secondo, visualizzando il tempo tramite porta seriale (EUSART) nel formato "mm:ss" ed al termine del conteggio emetta un suono tramite il buzzer.

Flow chart



Descrizione

Abbiamo gestito il temporizzatore del progetto attraverso TMR1, utilizzando il clock del LP e prescaler 1:1 per contare 1 secondo.
TMR1 è un contatore a 16 bit, che può misurare $2^{16} - 1$ valori.
Essendo il PIC un microcontrollore a 8 bit, TMR1 è diviso in 2 sottotimer: TMR1 High e TMR1 Low.
Abbiamo impostato TMR1 ad un valore ($2^{16} – 1$) – Tick calcolati (in base al datasheet), avendo come risultato il contatore che aumenta grazie al segnale di clock dell'LP, TMR1 arriverà al suo overflow.

Attraverso variabili tmp, sono state calcolate le cifre del contatore del TMR0, timer prefissato a 180 secondi (3 minuti) e poi decrementato, timer che abbiamo utilizzato per contare i secondi trascorsi.
La sub-routine "start" ci reindirizza all'inizializzazione dell'hardware (sub-routine "init_hw").

Nell'init hardware abbiamo configurato l'INTCON per l'interrupt del timer1, il quale e' stato disattivato l'interrupt durante la configurazione; abbiamo fatto il setup del TMR0 come contatore e dell'EUSART, usando come Fosc la frequenza adatta relativamente al BAUDRT desiderato (in questo caso 19200).
TMR2 è stato utilizzato per la generazione della PWM del buzzer: questa grazie all'abilitazione del registro CCP1CON.

Passiamo al "main_loop", dove c'è il controllo dell'overflow del TMR1 (se c'è stato, è passato 1 secondo).

“reload_timer1” è una sub-routine che inizialmente controlla se il buzzer è acceso, in questo caso va in sleep, altrimenti decrementa il valore di TMR0 ed arresta il timer di TMR1 per svolgere le operazioni.

Su “stampa_seriale”, vengono calcolate le cifre dei secondi, delle decine di secondi e dei minuti e vengono sommati al carattare ‘0’; oltre ai caratteri citati, in EUSART, quindi TXREG, vengono inviati i caratteri ‘:’ e ‘\n’, in ASCII.

Il PIC, non potendo svolgere divisioni, è stato programmato per svolgere serie di sottrazioni secondo i rispettivi dividendi, 60 per i minuti e 10 per le decine di secondi. La sub-routine si conclude se il TMR0 è arrivato a zero, con il goto alla “fine_con_buzzer”, dove viene impostato e attivato il buzzer; se TMR0 è maggiore di 0, va al “main_loop”.

Successivamente, si passa alla subroutine “vai_a_dormire”, dove si disattivano globalmente gli interrupt e si manda in sleep il PIC: al risveglio dallo sleep, il codice ricomincia da capo.

Codice

```
;*********************************************************************
*********************************************************************
;
;   Esame di Architetture e Programmazione di Sistemi Digitali/Sistemi
Elettronici
;   Autori Stefano Rossini e Samuele Pulita
;   Corso Ingegneria Elettronica / Ingegneria Elettronica e delle
Tecnologie Digitali
;   Universita' Politecnica delle Marche, Ancona
;
;   Data consegna progetto 17/06/2024
;
;   Traccia 14:
;   Si realizzi un firmware che implementi un countdown (valore iniziale
3 minuti) con risoluzione di 1 secondo,
;   visualizzando il tempo tramite porta seriale (EUSART) nel formato
"mm:ss" in cui
;   al termine del conteggio emetta un suono tramite il buzzer, con sleep
finale
;
;   Note: Progetto realizzato utilizzando la tecnica di polling
;       Nel codice potrete trovare degli elementi copia in Working
Register che non sono utili allo svolgimento del countdown
;       che sono stati utili per visualizzare i valori di quelle
variabili in debug con il simulatore
;
;   Risorse utili impiegato nella realizzazione del progetto:
;   * Materiale del corso (con anche i codici di esempio)
;   * https://www.microchip.com/en-us/product/pic16f887 (in particolare
il Datasheet)
;   * https://developerhelp.microchip.com/xwiki/bin/view/products/mcu-
mpu/8bit-pic/peripherals/eusart-ausart/  (per spiegazione approfondita
rispetto al datasheet)
;   * https://www.asciitable.com/ (per tabella ASCII)
;
;*********************************************************************
*********************************************************************




PROCESSOR 16F887// direttiva che definisce il tipo di processore
#include <xc.inc>    // file che contiene le definizioni dei simboli
(nomi registri, nomi bit dei registri, ecc).

; configuration bits
CONFIG "FOSC = INTRC_NOCLKOUT"// Ext Oscillator Selection->Oscillator not
enabled
CONFIG "CP = OFF"             // PFM and Data EEPROM code protection
disabled
CONFIG "CPD = OFF"            // Data memory code protection is disabled
CONFIG "WDTE = OFF"           // WDT Disabled
CONFIG "BOREN = OFF"       // Brown-out Reset disabled
```

```
CONFIG "PWRTE = OFF"        // Power-up Timer is disabled
CONFIG "LVP = OFF"            // Low voltage programming disabled
                        // LVP disattivato (_LVP_OFF)
CONFIG "DEBUG = OFF"          // Background debugger disabled

; CONFIG2
CONFIG  "BOR4V = BOR21V"       // Brown-out Reset Selection bit (Brown-
out Reset set to 2.1V)
CONFIG  "WRT = OFF"            // Flash Program Memory Self Write Enable
bits (Write protection off)



;Definizione costanti utili per il progetto
tmr_1000ms EQU (65536 - 32768) ;numero di clock per fare reset del timer1
per contare un secondo usando LP e prescaler 1:1
                        ;Dal Datasheet 6.0 Timer1 Module with Gate
control
UART_TX_BUSY EQU 0
EUSART_TRANSMIT_BUFFER_FULL EQU 0 ;TXIF IN PIR1 DA CONTROLLARE BUSY
                       ;Dal Datasheet 12.1.1.3 Transmit Interrupt Flag

; variabili in RAM (shared RAM)
PSECT udata_shr

;Variabili impiegate nel calcolo delle cifre dal contatore
tmp2_secondi:
    DS 1 ; riservato un byte in ram

tmp2b_secondi:
    DS 1

tmp3:
    DS 1

tmp_minuti:
    DS 1

tmp_decine_secondi:
    DS 1

;Variabili che, dopo il calcolo delle cifre, vengono sommate al carattere
ASCII '0' per poi essere inviate in EUSART
secondi:
    DS    1

minuti:
    DS     1

decine_secondi:
    DS 1

minuti_in_secondi:
    DS 1

;Variabili da salvare per "Context Saving During Interrupts"
```

```
;Nome variabili prese
;Dal Datasheet Example 14-1: SAVING STATUS AND W REGISTERS IN RAM

W_TEMP:
    DS     1

STATUS_TEMP:
    DS     1


flags:
    DS 1

; reset vector
;Dal Datasheet FIGURE 2-3: PROGRAM MEMORY MAP AND STACK FOR THE
PIC16F886/PIC16F887
PSECT resetVec,class=CODE,delta=2
resetVec:
          pagesel start
          goto start


; programma principale
PSECT MainCode,global,class=CODE,delta=2
start:
     pagesel init_hw
     call init_hw

     ;Dal Datsheet Nota in 2.2.2.4 PIE1 Register:
     ;Bit PEIE of the INTCON register must be set to enable any
perpheral interrupt
     ;Quindi andiamo sul registro INTCON

     ;Register 2-3: INTCON: INTERRUPT CONTROL REGISTER
     ;I bit sono settati di default a 0, tranne RBIF (nel nostro caso
non e' nel nostro studio)
     ;Indichiamo di seguito i bit che interessano nel nostro progetto

     ;bit 4 TXIE: EUSART Transmit Interrupt Enable bit --> Perche' i
caratteri li invieremo in EUSART
     ;1 = Enables the EUSART Transmit Interrupt
     ;0 = Disables the EUSART Transmit Interrupt

     ;bit 6 PEIE: Peripheral Interrupt Enable bit --> Da 2.2.2.4 PIE1
Register Note: Bit PEIE of the INTCON register must be set to enable any
peripheral interrupt
     ;1 = Enables all unmasked peripheral interrupts
     ;0 = Disables all peripheral interrupts

     ;bit 7 GIE: Global Interrupt Enable bit --> Senza questo bit a 1
non possiamo bloccare il codice quando avvengono gli interrupt
     ;1 = Enables all unmasked
     ;0 = Disables all interrupts

     banksel INTCON
     bsf INTCON, INTCON_PEIE_POSITION
```

```
      bsf INTCON, INTCON_GIE_POSITION

      pagesel main_loop
      goto main_loop


init_hw:
      ;setup dell'INTCON per il Timer1 Interrupt

      ;Configuriamo il clock a 4MHz e settiamolo di sistema
      ;Dal Datasheet
      ;Register 4-1: OSCCON: OSCILLATOR CONTROL REGISTER

      ;bit 0 SCS: System Clock Select bit
      ;1= Internal Oscillator is used for system clock
      ;0= CClock sourse defined by FOSC <2:0> of the CONFIG1 Register

      ;bit 6-4 IRCF<2:0>: Internal Oscillator Frequency Select bits
      ;110 = 4 MHz (default)
      banksel OSCCON
      movlw 01100001B
      movwf OSCCON


      ;Disabilitiamo TUTTI gli interrupt, prima del setup
      banksel INTCON
      clrf INTCON

      ;Da datasheet
      ;6.7 Timer1 Interrupt
      ;Note: The TMR1H:TMR1L register pair and the TMR1IF should be
cleared before enabling interrupts

      banksel TMR1H
      clrf TMR1H
      banksel TMR1L
      clrf TMR1L

      ;Setup del Timer1

      ;T1CON: TIMER1 CONTROL REGISTER

      ;bit 0 TMR1ON: Timer1 ON bit
      ;1 = Enables Timer1
      ;0 = Stops Timer1

      ;bit 1 TMR1CS: Timer1 Clock Source Select bit
      ;0 = Internal clock (Fosc/4)

      ;bit 2 T1SYNC
      ;IF TMR1CS = 0
      ;--> This bit is ignored. Timer1 uses the internal clock
      ;IF TMR1CS = 1
      ;1--> Do not synchronize external clock input
```

```
      ;bit 3 T1OSCEN: LP Oscillator Enable Control bit
      ;1 = LP Oscillator is enabled for Timer1 clock
      ;0 = LP Oscillator is off

      ;bit 5-4 T1CKPS<1:0>: Timer1 Input Clock Prescale Select bits
      ;00 = 1:1 Prescale Value

      ;bit 6 TMR1GE: Timer1 Gate Enable bit
      ;if TMR1 = 0
      ;--> This bit is ignored
      ;if TMR1 = 1
      ;1 = Timer1 counting is controlled by the Timer1 Gate function
      ;0 = Timer1 is always counting

      ;bit 7 T1GNV: Timer1 Gate Invert bit
      ; 1 = Timer1 counts when gate is high
      ; 0 = Timer1 counts when gate is low

      banksel T1CON
      ;movlw 00001001B --> Decommentare per rendere il conteggio piu'
veloce
      movlw 00001111B
      movwf T1CON

      ;Setup del TMR0 come contatore
      banksel TMR0
      movlw 181
      movwf TMR0

      pagesel reload_timer1
      call reload_timer1

      ;Setup dell'EUSART
      ;Baud rate = 19200 (BRGH = 1, BRG16 = 0)
      ;TXEN = 1
      ;SPEN = 1
      ;CREN = 1

      banksel TXSTA
      movlw 00100100B
      movwf TXSTA

      banksel BAUDCTL
      clrf BAUDCTL


      ;Usando Fosc = 4 MHz
      ;Table 12-5: BAUD RATES FOR ASYNCHRONOUS MODES (CONTINUED)
      banksel SPBRG
      movlw 12
      movwf SPBRG


      ; Di default, tutti i pin connessi all'ADC sono settati come input
analogici,
```

```
     ; impedendone l'uso come I/O digitali. L'impostazione seguente
rende I/O digitali
     ; i pin RB0..RB3
     banksel ANSELH             ; banco di ANSELH
     clrf  ANSELH               ; AN8..AN13 disattivati

    ; Timer2 e CCP1: generazione PWM per buzzer
    ;  timer2 prescaler = 16  (valori possibili: 1, 4, 16)
    ;   -> freq. = 62.5 kHz, tick = 16 us, periodo max = 4096 us
    ;  timer2 inizialmente off
    ;  CCP1: modalita' PWM
    banksel     T2CON
    movlw  00000011B                ;TMR2 OFF
    movwf  T2CON
    banksel     CCP1CON
    ; Imposto CCP1CON = B'00001100' dove:
    ; bit 3-0 = 1100 -> PWM mode; P1A,P1C active-high; P1B,P1D active-
high
    ; bit 5-4 = 00 -> PWM mode: These bits are the two LSbs of the PWM
duty cycle. The eight MSbs are found in CCPR1L.
    ; quindi non uso la risoluzione a 10-bit perchè per i nostri scopi
sono sufficienti 8
    ; bit 7-6 = 00 ->  Single output; P1A modulated; P1B, P1C, P1D
assigned as port pins
    ; quindi uso solo il pin P1A che nella piedinatura del mio pin
corrisponde proprio al pin RC2 (ingresso del buzzer)
    movlw  00001100B  ; PWM mode ;PWM enhanced mode: Single PWM
    movwf  CCP1CON

    return

main_loop:
         ;il codice rimane in loop qui

         banksel PIR1
         btfsc PIR1, PIR1_TMR1IF_POSITION ;se ci e' un overflow del TMR1
allora fa il reset del Timer stesso
         call reload_timer1



         goto main_loop

reload_timer1:
         ; ricarica contatore timer1 per ricominciare conteggio.
         ; In modalita' asincrona, occorre arrestare il timer prima
         ; di aggiornare i due registri del contatore

         banksel T2CON
         btfsc T2CON,T2CON_TMR2ON_POSITION
         goto vai_a_dormire


         banksel TMR0
         decfsz TMR0, f ;diminuisce di uno il contatore dei secondi del
countdown
```

```
          banksel     T1CON
          bcf    T1CON,T1CON_TMR1ON_POSITION     ; arresta timer

          pagesel stampa_seriale
          call stampa_seriale

          ; le funzioni "low" e "high" forniscono il byte meno e piu'
          ;  significativo di una costante maggiore di 8 bit
          banksel     TMR1L
          movlw  low  tmr_1000ms
          movwf  TMR1L
          movlw  high tmr_1000ms
          movwf  TMR1H
          banksel     PIR1
          bcf    PIR1,PIR1_TMR1IF_POSITION       ; azzera flag interrupt

          banksel     T1CON
          bsf    T1CON,T1CON_TMR1ON_POSITION          ; riattiva timer



          return

stampa_seriale:

           ;CALCOLO DELLE CIFRE

           ;calcolo dei minuti

           banksel tmp2_secondi
           clrf tmp2_secondi
           ;copiamo il valore di TMR0 che ci servira' dopo per il calcolo
delle decine
           banksel TMR0
           movf TMR0, w
           banksel tmp2_secondi
           movwf tmp2_secondi

           banksel minuti
           clrf minuti

           banksel decine_secondi
           clrf decine_secondi

           banksel tmp_decine_secondi
           clrf tmp_decine_secondi

           banksel TMR0
           movf TMR0, w

           banksel tmp_minuti
           movwf tmp_minuti

           ;Non potendo svolgere le divisioni sul Pic, allora possiamo
reintepretare le divisioni come sottrazioni successive
```

```
           banksel tmp_minuti
           movlw 60; i secondi da 60
           subwf tmp_minuti, f ; tmp_minuti = tmp_minuti-60

           ;Dal datasheet
           ;SUBWF --> f-W --> destination
           ;C= 0 W> f
           ;C=1 W<f
           banksel STATUS
           btfss STATUS, STATUS_C_POSITION ; Se carry e' 0
           goto $+9 ; salta alle prossime nove istruzione

           btfsc STATUS, STATUS_C_POSITION ; Se carry e' 1
           incf minuti

           ;aggiungiamo i 60 secondi a tmp_decine di secondi
           banksel tmp_decine_secondi
           movlw 60
           addwf tmp_decine_secondi, f ;tmp_decine_secondi =
tmp_decine_secondi + 60 (se c'e' un minuti=o)
           goto $-13 ; ritorna indietro al calcolo della divisione



           ;calcolo delle decine di secondi



           banksel tmp_decine_secondi
           movf tmp_decine_secondi, w ; vediamo in debug quanto e'
rimasto e' 255

           ;sottrarre da tmr0 le decine di secondi
           banksel tmp_decine_secondi
           ;sottrarre a TMR0 i minuti calcolati
           ;tmp_minuti = TMR0 - 60*minuti
           banksel tmp_decine_secondi
           movf tmp_decine_secondi
           banksel tmp2_secondi
           subwf tmp2_secondi, f

           banksel tmp2_secondi
           movf tmp2_secondi, w ; vediamo se in debug c'e' 59 quando TMR0
e' 179 YES


     ;*****************************************************************
*****
           ;Adesso dividere tmp2_secondi per 10 per calcolarsi le decine
di secondi

           banksel tmp_decine_secondi
           movwf tmp_decine_secondi
           ;copiare il valore di tmp2_secondi in
tmp2_secondi_se_minore_dieci
```

```
            banksel tmp2_secondi
            movf tmp2_secondi, w
            banksel tmp2b_secondi
            movwf tmp2b_secondi

            banksel tmp2_secondi
            movlw 10; i secondi da 60
            subwf tmp2_secondi, f ; tmp_minuti = tmp_minuti-60

            ;Dal datasheet
            ;SUBWF --> f-W
            ;C= 0 W> f
            ;C=1 W<f
            banksel STATUS
            btfss STATUS, STATUS_C_POSITION ; Se carry e' 0
            goto $+11

            btfsc STATUS, STATUS_C_POSITION ; Se carry e' 1
            incf decine_secondi, f;minuti

            ;Calcolarsi i secondi rimasti e' proprio tmp2_secondi
            banksel tmp2_secondi
            movf tmp2_secondi, w
            banksel secondi
            movwf secondi

            movf secondi, w ;debug

            goto $-21

            ;*****************************************
            ;Calcolo dei secondi, se sono rimasti sotto i 10 secondi (da 9
a 0)

            banksel tmp2b_secondi
            movf tmp2b_secondi, w
            banksel tmp3
            movwf tmp3

            banksel tmp2b_secondi
            movlw 10
            subwf tmp2b_secondi, f ; tmp_minuti = tmp_minuti-60

            banksel STATUS

            btfsc STATUS, STATUS_C_POSITION ; Se carry e' 1
            goto $+5

            btfss STATUS, STATUS_C_POSITION ; Se carry e' 0
            ;tmp3 sono i secondi

            banksel tmp3
            movf tmp3, w
            banksel secondi
            movwf secondi
```

```
     ;*****************************************************************
*****

           banksel minuti
           movf minuti, w

           banksel decine_secondi
           movf decine_secondi, w  ; vediamo in debug se ci sono le
decine di secondi

           banksel secondi
           movf secondi, w



           ;*************************
           ;Convertire i valori in ascii

           banksel minuti
           movlw '0'
           addwf minuti, f ;minuti = minuti + ASCII Code per 0

           banksel decine_secondi
           movlw '0'
           addwf decine_secondi, f ;minuti = minuti + ASCII Code per 0

           banksel secondi
           movlw '0'
           addwf secondi, f ;minuti = minuti + ASCII Code per 0


           ;minuti_in_secondi = 60 * minuti = 60 + 60 se minuti e' > 0
           banksel minuti_in_secondi
           clrf minuti_in_secondi

           addlw 60
           addwf minuti_in_secondi, f ; minuti_in_secondi =
minuti_in_secondi + 60


           ;*********************************************

           bsf PIE1, PIE1_TXIE_POSITION ;avvia l'interrupt dell'invio del
carattere

           ;Buffer di TXREG due caratteri, quindi dividere l'invio in
           ;EUSART in tre inviii

           ;************************************
           banksel TXREG
           movlw '0'
           movwf TXREG
```

```
            ;Ora EUSART occupata. Noi sappiamo che e' occupata dal flag
            ;Dal datasheet
            ;PIR1
            ;bit TXIF
            ;0 = EUSART BUFFER IS FULL

            banksel PIR1
            btfss PIR1, PIR1_TXIF_POSITION
            goto $-1 ;ritorna indietro di un comando finche' non e' stato
inviato il carattere
            ;*****************************

            bsf PIE1, PIE1_TXIE_POSITION ;avvia l'interrupt dell'invio del
carattere


            banksel minuti
            movf minuti, w
            banksel TXREG
            movwf TXREG

            banksel PIR1
            btfss PIR1, PIR1_TXIF_POSITION
            goto $-1
            ;*****************************

            bsf PIE1, PIE1_TXIE_POSITION ;avvia l'interrupt dell'invio del
carattere


            banksel TXREG
            movlw ':'
            movwf TXREG

            banksel PIR1
            btfss PIR1, PIR1_TXIF_POSITION
            goto $-1
            ;*****************************

            ;Ora EUSART occupata. Noi sappiamo che e' occupata dal flag
            ;Dal datasheet
            ;PIR1
            ;bit TXIF
            ;0 = EUSART BUFFER IS FULL

            ;banksel PIR1
            ;btfss PIR1, PIR1_TXIF_POSITION
            ;goto $-1

            bsf PIE1, PIE1_TXIE_POSITION ;avvia l'interrupt dell'invio del
carattere

            banksel decine_secondi
            movf decine_secondi, w
            banksel TXREG
```

```
            movwf TXREG

            banksel PIR1
            btfss PIR1, PIR1_TXIF_POSITION
            goto $-1
            ;*************************************

            bsf PIE1, PIE1_TXIE_POSITION ;avvia l'interrupt dell'invio del
carattere

            banksel secondi
            movf secondi, w
            banksel TXREG
            movwf TXREG

            banksel PIR1
            btfss PIR1, PIR1_TXIF_POSITION
            goto $-1

            ;*****************************
            bsf PIE1, PIE1_TXIE_POSITION ;avvia l'interrupt dell'invio del
carattere

            movlw 10 ;carattere a capo in ASCII code
            movwf TXREG

            banksel PIR1
            btfss PIR1, PIR1_TXIF_POSITION
            goto $-1

            ;***************************

            ;Ora EUSART occupata. Noi sappiamo che e' occupata dal flag
            ;Dal datasheet
            ;PIR1
            ;bit TXIF
            ;0 = EUSART BUFFER IS FULL

            banksel PIR1
            btfss PIR1, PIR1_TXIF_POSITION
            goto $-1


            ;Quando TMR0 e' zero, finisce il codice
            ;Questo accade perche' se TMR0 = 0 e viene sottratto a 1, TMR0
diventa 255 perche' va in underflow
            ;Abbiamo usato l'istruzione subwf perche' decf non da il bit
di Carry (quindi non sappiamo se avviene cio' che' indicato sopra)

            banksel TMR0
            movlw 1
            subwf TMR0, w

            banksel STATUS
```

```
            btfss STATUS, STATUS_C_POSITION ; Se carry e' 0
            goto fine_con_buzzer



            return

vai_a_dormire:

    ;disattivo interrupt globali prima di dormire
    pagesel fine_con_buzzer
    call fine_con_buzzer

    banksel INTCON
    bcf INTCON, INTCON_GIE_POSITION
    sleep
    bcf INTCON, INTCON_GIE_POSITION

    pagesel start
    goto start

fine_con_buzzer:

    ;Buzzzer
    banksel TRISC
    bcf TRISC, TRISC_TRISC2_POSITION ; RCE come output



    ; copia costante appena caricata in W su PR2
    movlw 119
    banksel     PR2
    movwf  PR2
    ; per ottenere un'onda quadra con duty-cycle al 50%,
    ; occorre settare CCPR1L alla meta' di PR2
    bcf    STATUS,STATUS_C_POSITION   ; azzera carry per successivo shift
    rrf    PR2, w          ; W = PR2 shiftato a destra = meta'
    banksel     CCPR1L
    movwf  CCPR1L
    ; attiva timer2 -> suono emesso da buzzer
    banksel     T2CON
    bsf    T2CON,T2CON_TMR2ON_POSITION


    return




;interrupt
;14.4 Context Saving During Interrupts
PSECT isrVec,class=CODE,delta=2
isr:
```

```
            ; context saving (vedere datasheet) INIZIO
            movwf W_TEMP
            swapf STATUS, W
            movwf STATUS_TEMP
            pagesel  test_timer1
            goto test_timer1
            ;Inserire il codice dei bit test
test_timer1:
                ; testa evento overflow timer1 (TMR1IF + TMR1IE)
                banksel    PIR1
                btfss PIR1,PIR1_TMR1IF_POSITION
                goto  irq_end
                banksel    PIE1
                btfss PIE1,PIE1_TMR1IE_POSITION
                goto  irq_end
                ; avvenuto interrupt

                pagesel stampa_seriale
                call stampa_seriale


                ; ricarica contatore timer1
                pagesel    reload_timer1
                call  reload_timer1
                ; fine evento timer1
                goto  irq_end



irq_end:
    ; context saving (vedere datasheet) FINE
    swapf STATUS_TEMP, W
    movwf STATUS
    swapf W_TEMP, F
    swapf W_TEMP, W

    RETFIE; esci dall'interrupt e ritorni nel codice di prima

    END resetVec
```