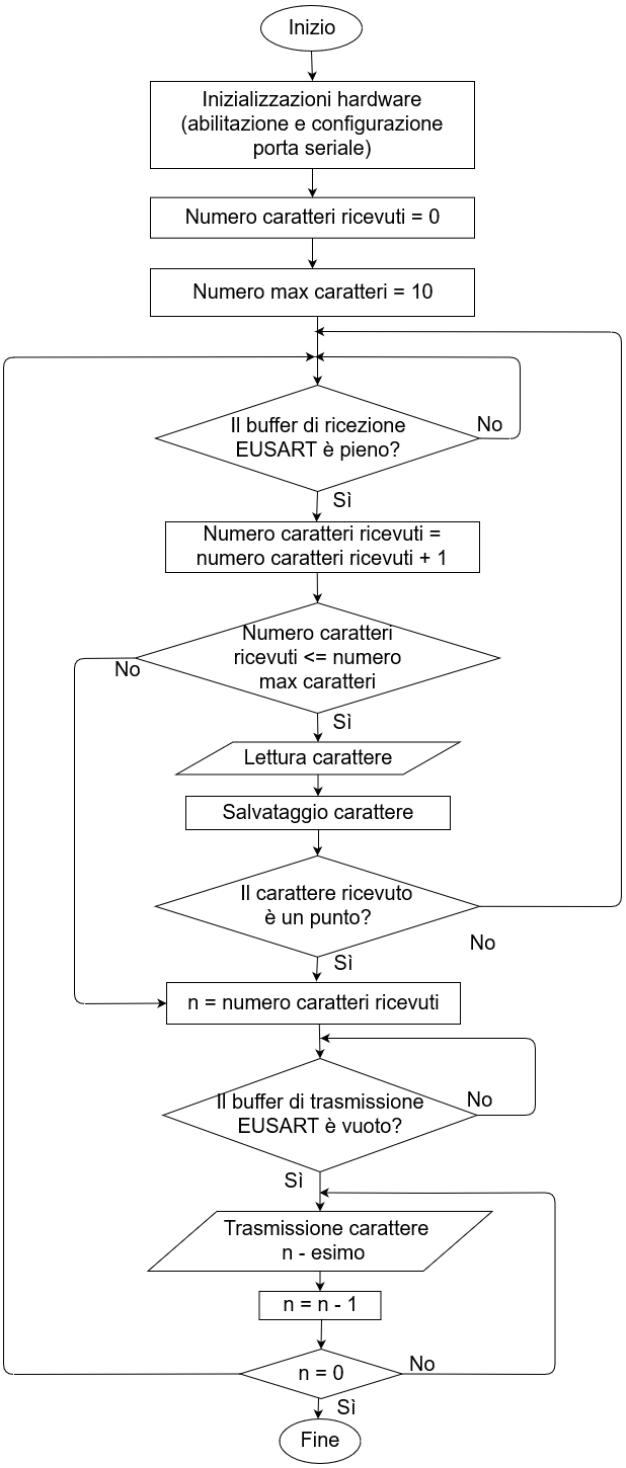
Relazione Sistemi Elettronici

Descrizione della tesina: si realizzi un firmware che riceve dal computer (tramite porta seriale) una parola, come sequenza di codici ascii dei singoli caratteri. La parola è terminata da un punto ed è di lunghezza massima fissata a priori. Dopo aver ricevuto la parola, il programma deve reinviarla sulla porta seriale scritta al contrario.

Matteo Orlandini
Corso di Laurea Triennale in Ingegneria Elettronica
Matricola 1079505

Indice

1 Diagramma di flusso



2 Codice

2.1 Direttive

La direttiva list imposta il tipo di processore usato, cioè il PIC16F887. I "configuration bits" possono essere programmati per selezionare varie configurazioni del dispositvo. Questi bit sono mappati all'indirizzo 2007h della program memory. L'indirizzo 2007h è oltre lo spazio della program memory, appartiene infatti allo spazio special configuration memory compreso tra 2000h e 3FFFh, che può essere indirizzato solo durante la programmazione. I configuration bits sono impostati come riportato di seguito:

- *CONFIG1*:
  - *_INTRC_OSC_NOCLKOUT* → oscillatore interno: pin RA6/OSC2/CLKOUT e RA7/OSC1/CLKIN con funzioni di I/O
  - *_WDT_OFF* → Watch Dog Timer disabilitato, può essere abilitato dal bit *SWDTEN* del registro *WDTCON*.
  - *_LVP_OFF*→ pin RB3 usato come I/O, High Voltage su MCLR usato per programmare.
- *CONFIG2*:
  - *_BOR21V* → Brown-out Reset settato a 2.1 V.

```
#include "p16f887.inc"
list p=16f887

; configuration bits
__CONFIG _CONFIG1, _INTRC_OSC_NOCLKOUT & _WDT_OFF & _LVP_OFF
__CONFIG _CONFIG2, _BOR21V

```

2.2 Inizializzazione variabili

1. numeroChar contiene il numero attuale di caratteri ricevuti sulla seriale, viene inizializzato a zero.
2. numeroMassimo contiene il numero massimo di caratteri che si possono inviare sulla seriale, viene inizializzato a 10.
3. rxData contiene il dato ricevuto, letto da *RCREG*.
4. Inizializzo il puntatore File Select Register (FSR) al fine di puntare all'indirizzo 0x20, cioè al primo General Purpose Register disponibile.

```
;variabili in RAM (shared RAM)
                    udata_shr
numeroChar          res             .1
numeroMassimo       res             .1
rxData              res             .1

; reset vector
```

```
8  rst_vector              code    0x0000
9                          pagesel start
10                         goto start
11
12
13         ; programma principale
14         code
15 start
16         ;inizializzo la variabile che conta i caratteri ricevuti
17         movlw .0
18         movwf numeroChar
19         ;inizializzo la variabile che contiene il numero massimo di
20         ;caratteri che si possono ricevere
21         movlw .10
22         movwf numeroMassimo
23
24         ;inizializzo il puntatore al vettore che contiene i caratteri
25         ;per contenere l'indirizzo 0x20, cioe' il primo general purpose
26         ;file register
27         movlw 0x20
28         movwf FSR
29
30         pagesel initHw
31         ;initHw contiene le inizializzazioni hardware
32         call initHw
33
```

2.3 Configurazioni hardware

1. Gli interrupt sono tutti disabilitati, quindi il registro *INTCON* ha tutti i bit a zero, perché il controllo sulla seriale viene fatto in polling. Questa scelta è dovuta al fatto che sarebbe possibile mandare il PIC in sleep e poi farlo svegliare tramite wake up da seriale, ma si perderebbe il primo byte ricevuto che servirebbe per l'uscita dallo stato di sleep del micro.

2. Disabilito tutti gli interrupt dalle periferiche resettando tutti i bit del registro PIE1.

3. Dal registro *OSCCON* imposto il clock a 8 MHz portando ad 1 i bit [6:4] e il bit 0 per usare l'oscillatore interno come clock di sistema.

4. Nel registro *OPTION_REG* imposto il prescaler del timer 0 a 256 e l'interrupt sul fronte di discesa del pin INT. Il timer 0 viene impostato ma non è usato nel progetto.

5. Tutte le porte sono settate in input perché non sono necessari degli output nel progetto.

6. Il protocollo USART è configurato nei registri *TXSTA*, *RXSTA*, *BAUDCTL* e *SPBRG*.

   (a) In *TXSTA*, i bit *TXEN* e *BRGH* sono ad 1 per indicare rispettivamente l'abilitazione della trasmissione e la trasmissione ad alta velocità. Il bit *SYNC* è a zero perché viene selezionata la modalità asincrona.

   (b) Nel registro *RXSTA*, vengono abilitate la porta seriale e la ricezione continua settando i bit *SPEN* e *CREN*.

   (c) Si resettano tutti i bit di *BAUDCTL*, tra cui anche *BRG16*, impostando così il baud rate a 8 bit.

(d) Con la configurazione citata, cioè $SYNC = 0$, $BRGH = 1$ e $BRG16 = 0$, la formula per calcolare il Baud Rate è la seguente:

$$BaudRate = \frac{F_{osc}}{16 \cdot (n+1)}$$

dove $F_{osc} = 8MHz$ e n è il numero da inserire nel registro *SPBRG*. Per avere un Baud Rate di 19200 bps, n deve essere pari a 25.

```
initHw
        ;*********** inizio configurazioni hardware  ******************


        ;**********  inizio configurazione interrupt  *****************
        movlw B'00000000'
        banksel INTCON
        ;INTCON:
        ;bit 7 = 0 -> disabilitazione di tutti gli interrupt
        ;bit 6 = 0 -> disabilitazione interrupt dalle periferiche
        ;bit 5 = 0 -> disabilitazione interrupt timer 0
        ;bit 4 = 0 -> disabilitazione interrupt esterno
        ;bit 3 = 0 -> disabilitazione interrupt porta B
        ;bit 2 = 0 -> reset del flag interrupt timer 0
        ;bit 1 = 0 -> reset del flag interrupt esterno
        ;bit 0 = 0 -> reset del flag interrupt porta B
        movwf INTCON

        movlw B'00000000'
        banksel PIE1
        ;PIE1
        ;bit 7 = 0 -> non implementato
        ;bit 6 = 0 -> disabilitazione interrupt ADC
        ;bit 5 = 0 -> disabilitazione interrupt EUSART in ricezione
        ;bit 4 = 0 -> disabilitazione interrupt EUSART in trasmissione
        ;bit 3 = 0 -> disabilitazione interrupt MSSP
        ;bit 2 = 0 -> disabilitazione interrupt CCP1
        ;bit 1 = 0 -> disabilitazione interrupt Timer 2 = PR2
        ;bit 0 = 0 -> disabilitazione interrupt overflow Timer 1
        movwf PIE1
        ;***********  fine configurazione interrupt  ******************

        ;***********  inizio configurazione clock  ********************
        movlw B'01110001'
        banksel OSCCON
        ;OSCCON:
        ;bit 7 = non implementato
        ;bit 6-4 = 111 -> oscillatore interno a 8 MHz
        ;bit 3 = 0 -> il PIC lavore con l'oscillatore interno (solo     lettura)
        ;bit 2 = 0 -> HFINTOSC non stabile (solo lettura)
        ;bit 1 = 0 -> LFINTOSC non stabile (solo lettura)
        ;bit 0 = 1 -> oscillatore interno usato come clock di sistema
        movwf OSCCON

        movlw B'00000111'
        banksel OPTION_REG
        ;OPTION_REG:
        ;bit 7 = 0 -> pull up abilitato sulla porta b
        ;bit 6 = 0 -> interrupt sul fronte di discesa
        ;bit 5 = 0 -> clock interno (Fosc/4)
        ;bit 4 = 0 -> incremento del Timer 0 sulla transizione basso-alto
```

```
51          ;del pin T0CKI
52          ;bit 3 = 0 -> prescaler assegnato al Watch Dog Timer
53          ;bit 2-0 = 111 Prescaler 1:256
54          ; -> Ftick = (8 MHz / 4) / 256 = 7812.5 Hz, tick = 128us, periodo = 32.768
        ms
55          movwf OPTION_REG
56          ;************  fine configurazione clock  **********************
57
58          ;************  inizio configurazione porte  ********************
59          ;port A: non usata, input
60          movlw B'11111111'
61          banksel TRISA
62          movwf TRISA
63
64          ;port B: non usata, input
65          movlw B'11111111'
66          banksel TRISB
67          movwf TRISB
68
69          ;port C: RC6 e RC7 usate per la seriale
70          movlw B'11111111'
71          banksel TRISC
72          movwf TRISC
73
74          ;port D: non usata, input
75          movlw B'11111111'
76          banksel TRISD
77          movwf TRISD
78
79          ;port E: non usata, input
80          movlw B'00001111'
81          banksel TRISE
82          movwf TRISE
83          ;**************  fine configurazione porte  ********************
84
85          ;**************  inizio configurazioni USART  ******************
86          movlw B'00100100'
87          banksel TXSTA
88          ;TXSTA:
89          ;bit 7 = 0 -> don't care perche' usart in modalita' asincrona
90          ;bit 6 = 0 -> trasmissione a 8 bit
91          ;bit 5 (TXEN) = 1 -> trasmissione abilitata
92          ;bit 4 (SYNC) = 0 -> modalita' asincrona
93          ;bit 3 = 0 -> Sync Break transmission completata
94          ;bit 2 (BRGH) = 1 -> Baud rate alta velocita'
95          ;bit 1 = 0 -> Transmit Shift Register Status bit (solo lettura)
96          ;bit 0 = 0 -> contenuto del nono bit (non abilitato)
97          movwf TXSTA
98
99          movlw B'10010000'
100         banksel RCSTA
101         ;RCSTA:
102         ;bit 7 (SPEN) = 1 -> porta seriale abilitata
103         ;bit 6 = 0 -> ricezione a 8 bit
104         ;bit 5 = 0 -> don't care perche' usart in modalita' asincrona
105         ;bit 4 (CREN) = 1 -> ricezione continua abilitata
106         ;bit 3 = 0 -> don't care perche' ricezione a 8 bit
107         ;bit 2 = 0 -> Framing Error bit (solo lettura)
108         ;bit 1 = 0 -> Overrun Error bit (solo lettura)
109         ;bit 0 = 0 -> Nono bit ricevuto, non usato (solo lettura)
110         movwf RCSTA
111
```

```
112         movlw B'00000000'
113         banksel BAUDCTL
114         ;BAUDCTL:
115         ;bit 7 = 0 -> overflow del baud timer (solo lettura)
116         ;bit 6 = 0 -> ricezione dello start bit (solo lettura)
117         ;bit 5 = 0 (non implementato)
118         ;bit 4 = 0 -> Transmissione dei dati non invertiti al pin RB7
119         ;bit 3 (BRG16) = 0 -> baud rate a 8 bit
120         ;bit 2 = 0 (non implementato)
121         ;bit 1 = 0 -> wake up enable bit disabilitato
122         ;bit 0 = 0 -> Auto-Baud Detect disabilitato
123         movwf BAUDCTL
124
125         ;per avere un baud rate di 19200 occorre scrivere .25 nel
126         ;registro SPBRG perche' con Fosc = 8 MHz, SYNC = 0, BRGH = 1,
127         ;BRG16 = 0 si ha BaudRate = Fosc/[16 * (n+1)] e con n = 25,
128         ;BaudRate = 19230 bps
129         movlw .25
130         banksel SPBRG
131         movwf SPBRG
132         ;*************  fine configurazione USART  ********************
133         return
134
135         ;************* fine configurazioni hardware  ******************
136
```

2.4 Ricezione da seriale

1. Nel main loop si abilitano i bit *SPEN* e *CREN* di *RCSTA* nel caso in cui si fossero verificati precedentemente un framing error o un over run.
2. Si aspetta in polling la ricezione di un byte da seriale controllando il bit *RCIF* di *PIR1*.
3. Si legge dunque il framing error bit e se è alto si può resettare portando a zero il bit *SPEN*.
4. Si ottiene il byte ricevuto dal buffer *RCREG* copiandolo sia nella variabile *rxData* sia in *INDF*.
5. Vengono poi incrementati il puntatore alla cella di memoria che contiene i dati ricevuti e la variabile *numeroChar* che conta i caratteri inviati dal PC.
6. Se si è verificato un over run, segnalato dal bit *OERR* di *RCSTA*, si resetta il flag portando a zero *CREN*.
7. Viene fatta la sottrazione tra *numeroMassimo* a *numeroChar*, questo ha effetto sul bit *Z* di *STATUS*.
   (a) Se *numeroMassimo* - *numeroChar* = 0, cioè $Z = 1$, si è raggiunto il numero massimo di caratteri ricevibili, viene chiamata la subroutine *TXEUSART* che si occupa della trasmissione dei byte su seriale.
   (b) Se $Z = 0$ si procede a verificare se il carattere ricevuto sia un punto.
8. Per controllare il punto si sottrae l'equivalente numerico del codice ASCII del punto a *rxData* che contiene il carattere ricevuto. Facendo lo stesso controllo sul bit *Z* di *STATUS* descritto precedentemente, si verifica se è necessario chiamare *TXEUSART* o ritornare al *mainLoop*.

```
mainLoop
        ;abilitazione seriale e ricezione continua
        banksel RCSTA
        bsf RCSTA, SPEN
        bsf RCSTA, CREN
        banksel PIR1
        ;se RCIF e' a 1 allora e' stato ricevuto un byte da seriale
        btfss PIR1, RCIF
        goto $-1
        ;lettura del bit di Framing Error, se e' ad uno si puo' resettare
        ;portando a zero il bit SPEN di RCSTA che resetta la EUSART
        banksel RCSTA
        btfsc RCSTA, FERR
        bcf RCSTA, SPEN
        ;lettura del contenuto di RCREG
        ;per resettare l'interrupt RCIF
        banksel RCREG
        movf RCREG,w
        ;metto il contenuto di RCREG in rxData
        movwf rxData
        ;metto il contenuto di RCREG nel vettore che contiene i caratteri
        movwf INDF
        ;incremento il puntatore
        incf FSR, f
        ;incremento la variabile che conta i caratteri
        incf numeroChar, f
        ;se si e' verificato un over run, cioe' il bit OERR di RCSTA e'
        ;a uno, si resetta il flag portando a zero il bit CREN di RCSTA
        banksel RCSTA
        btfsc RCSTA, OERR
        bcf RCSTA, CREN
        ;se i caratteri ricevuti sono maggiori del numeroMassimo allora
        ;li invio sulla seriale
        movf numeroMassimo, w
        subwf numeroChar, w
        btfsc STATUS, Z
        call TXEUSART

        ;se il byte ricevuto e' un punto la parola e' terminata
        movlw '.'
        ;confronta il dato ricevuto con '.'
        subwf rxData, w
        btfsc STATUS, Z
        ;se il carattere ricevuto e' '.', chiama TXEUSART
        call TXEUSART
        goto mainLoop

```

2.5 Trasmissione da seriale

1. Per trasmettere da seriale occorre portare a zero il bit *SYNC* di *TXSTA* e settare *CREN* e *TXEN* rispettivamente di *RCSTA* e *TXSTA*.

2. Viene decrementato il puntatore per ottenere l'ultimo byte arrivato, poiché nella parte di codice che gestisce la ricezione da seriale viene aumentato il puntatore dopo che è stato salvato il dato e quindi si andrebbe a puntare una locazione di memoria dopo l'ultimo byte effettivamente ricevuto.

3. Si mette il contenuto di *INDF* nel registro accumulatore *W*.
4. Solo quando il bit *TXIF* è alto, cioè il buffer di trasmissione è vuoto, si invia un nuovo carattere scrivendo *W* in *TXREG*.
5. Si decrementa dunque la variabile *numeroChar*.
6. Si esegue questo loop finché *numeroChar* = 0.

```
TXEUSART
        ;resetto il bit SYNC di TXSTA (trasmissione asincrona)
        banksel TXSTA
        bcf TXSTA, SYNC
        banksel RCSTA
        bsf RCSTA, CREN
        ;abilitazione della trasmissione seriale
        banksel TXSTA
        bsf TXSTA, TXEN

        invioDati
        ;decremento il puntatore
        decf FSR, f
        ;metto il contenuto di ogni elemento del vettore in w
        movf INDF, w
        ;TXIF e' a 1 quando il buffer di trasmissione EUSART e' vuoto
        banksel PIR1
        btfss PIR1, TXIF
        goto $-1
        ;scrivo w (numeroChar) in TXREG
        banksel TXREG
        movwf TXREG
        decfsz numeroChar
        goto invioDati
        return

```