FSK
Modulation-Demodulation
Protocollo di lavoro

Massimo Maiolo & Stefano Camozzi

25 maggio 2006

Indice

Elenco delle figure

1 Introduzione

Programmare in assembler un modulatore di frequenza per PIC16F877.
Il programma deve modulare con 3 diverse frequenze:

- 1050 Hz in stato Idle
- 900 Hz se invia uno '0' logico
- 1200 Hz se invia un '1' logico

1.1 Esecuzione del compito

Il programma modula le 3 frequenze richieste con un'eccezione: lo '0' logico viene modulato con una frequenza leggermente piú alta (976 Hz anziché 900 Hz), questo per motivi di settaggio di parametri.

L'utente, una volta inviato il programma si trova a terminale un header e il prompt dei comandi. Quando invia da tastiera dei caratteri (via USART tramite RS232) il PIC risponde con un echo a terminale e contemporaneamente modula il segnale sul pin 1 di PORTC.
Dall'altra parte un altro PIC riceve il segnale sempre sul pin 1 di PORTC. Il segnale viene demodulato e il testo viene inviato (con la USART tramite RS232) sul terminale attaccato a questo PIC.

2 Flow-chart

Qui di seguito i Flow-charts dei 2 programmi.

2.0.1 FSK modulation

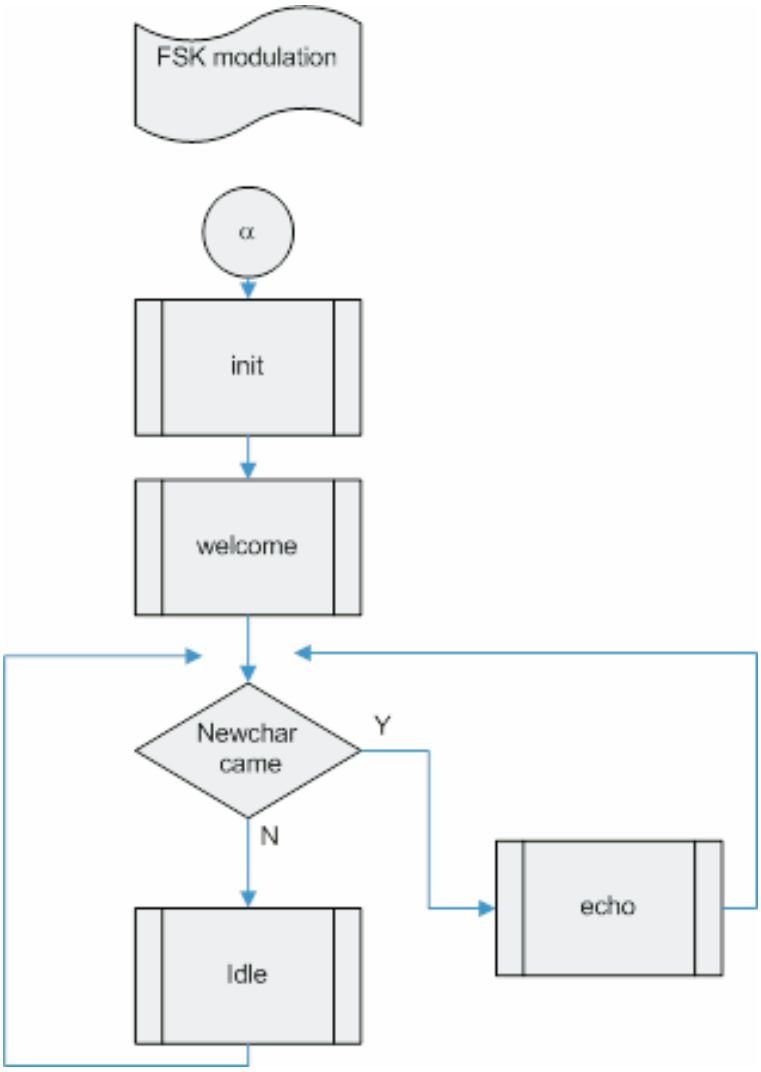

Figura 1: Main program modulation

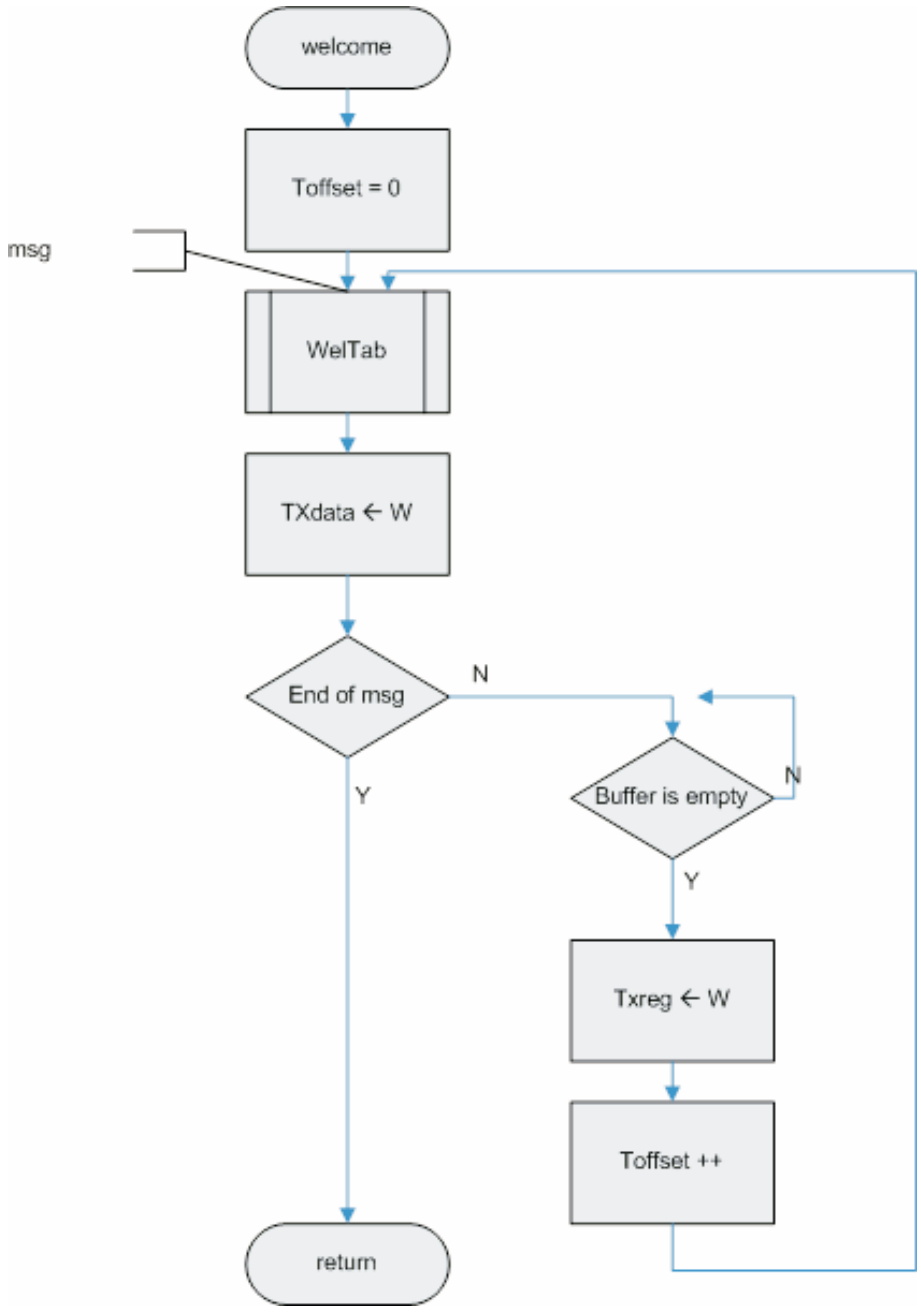

Figura 2: Welcome task modulation

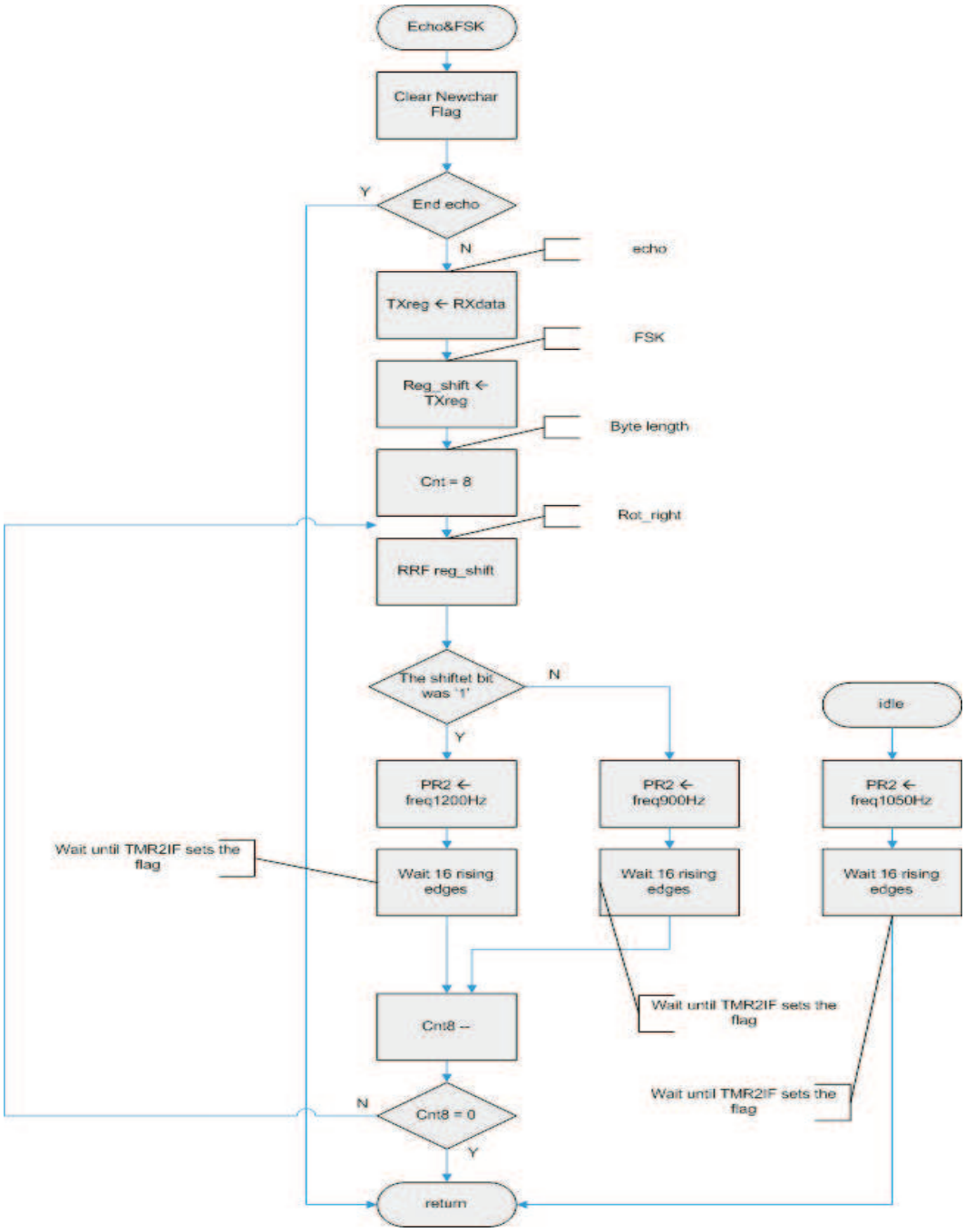

Figura 3: Echo&FSK task modulation

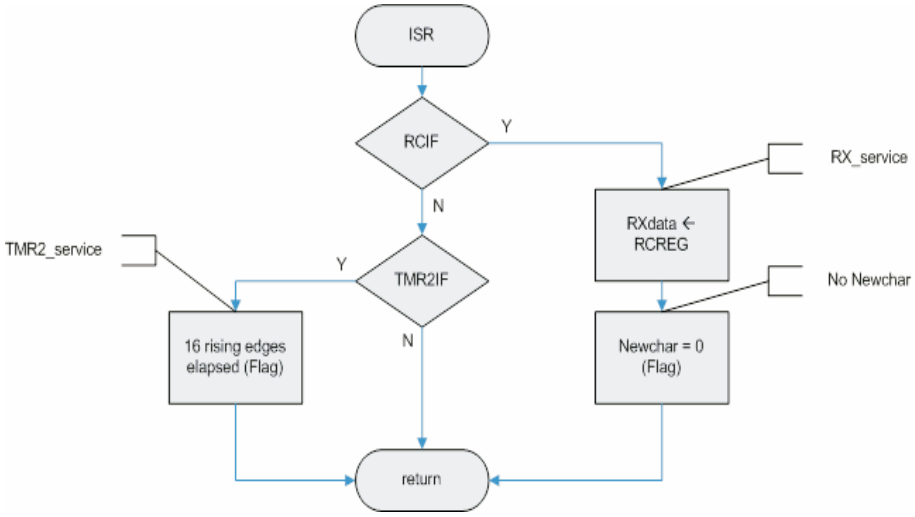

Figura 4: Interrupt Service Routine modulation

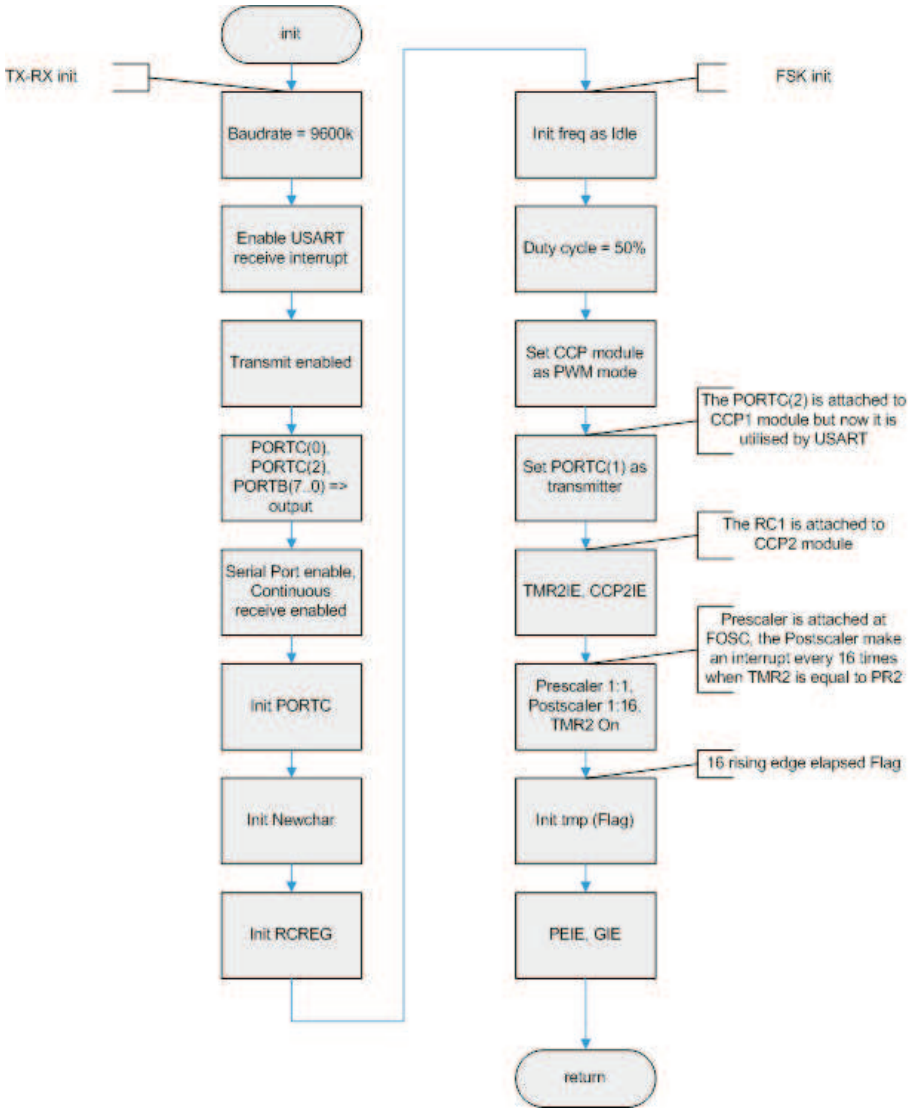

Figura 5: Initialisation modulation

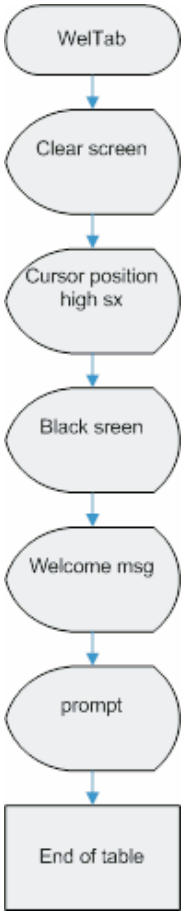

Figura 6: Table modulation

2.0.2 FSK demodulation

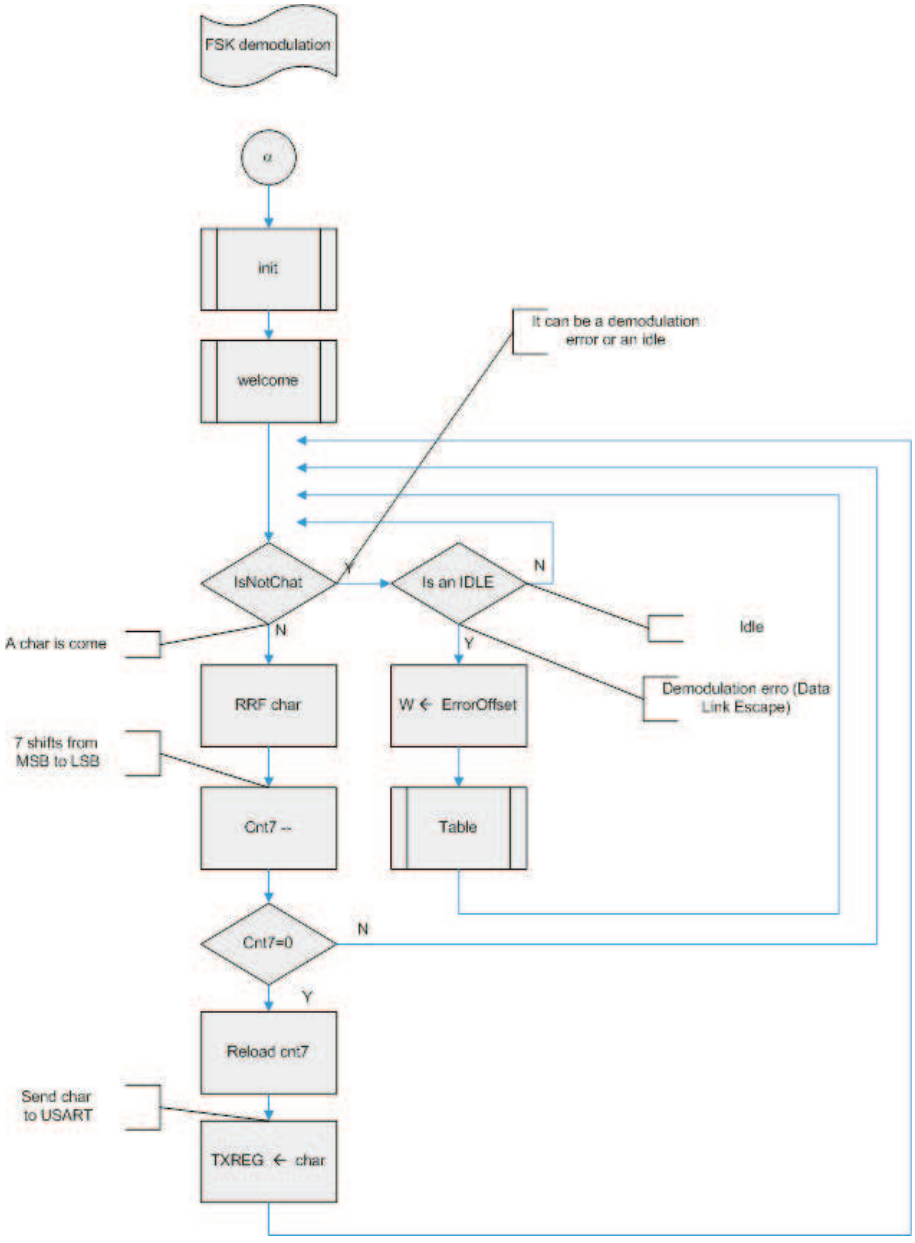

Figura 7: Main program demodulation

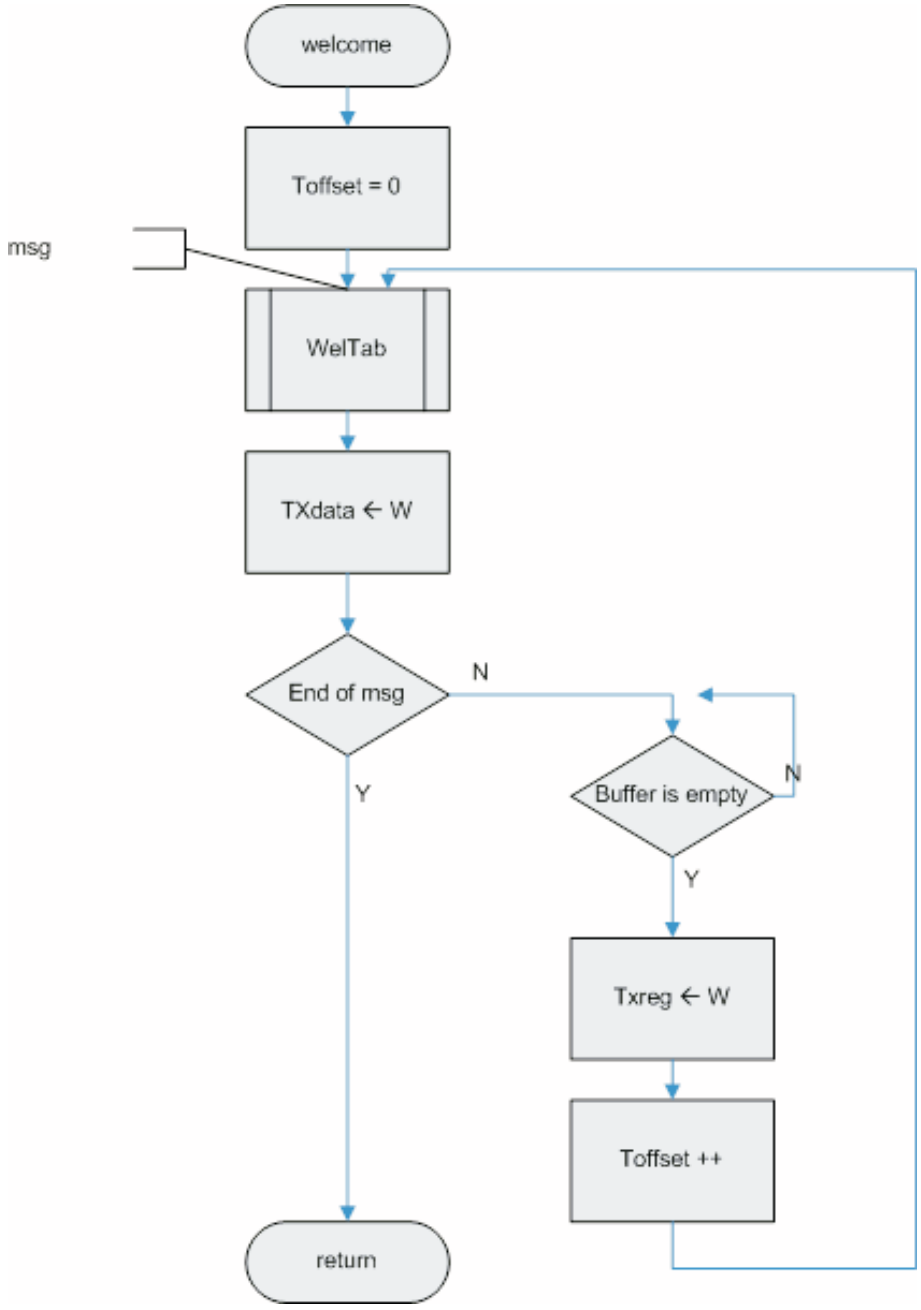

Figura 8: Welcome task demodulation

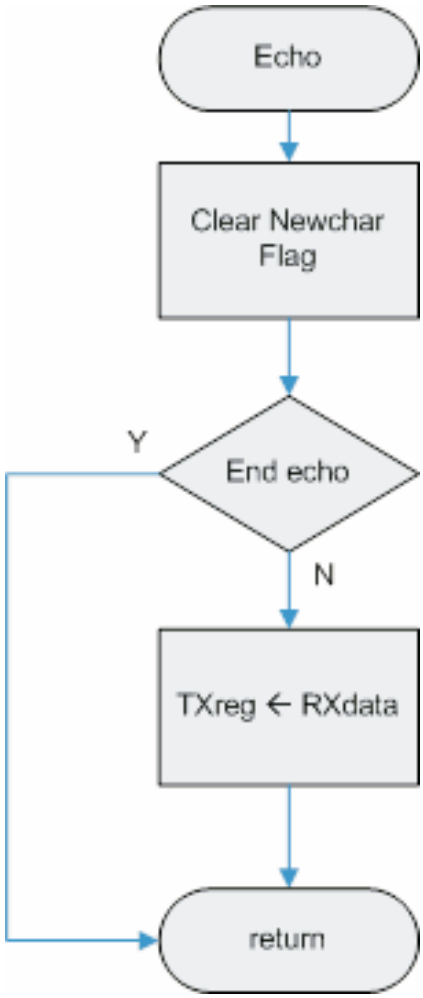

Figura 9: Echo task demodulation

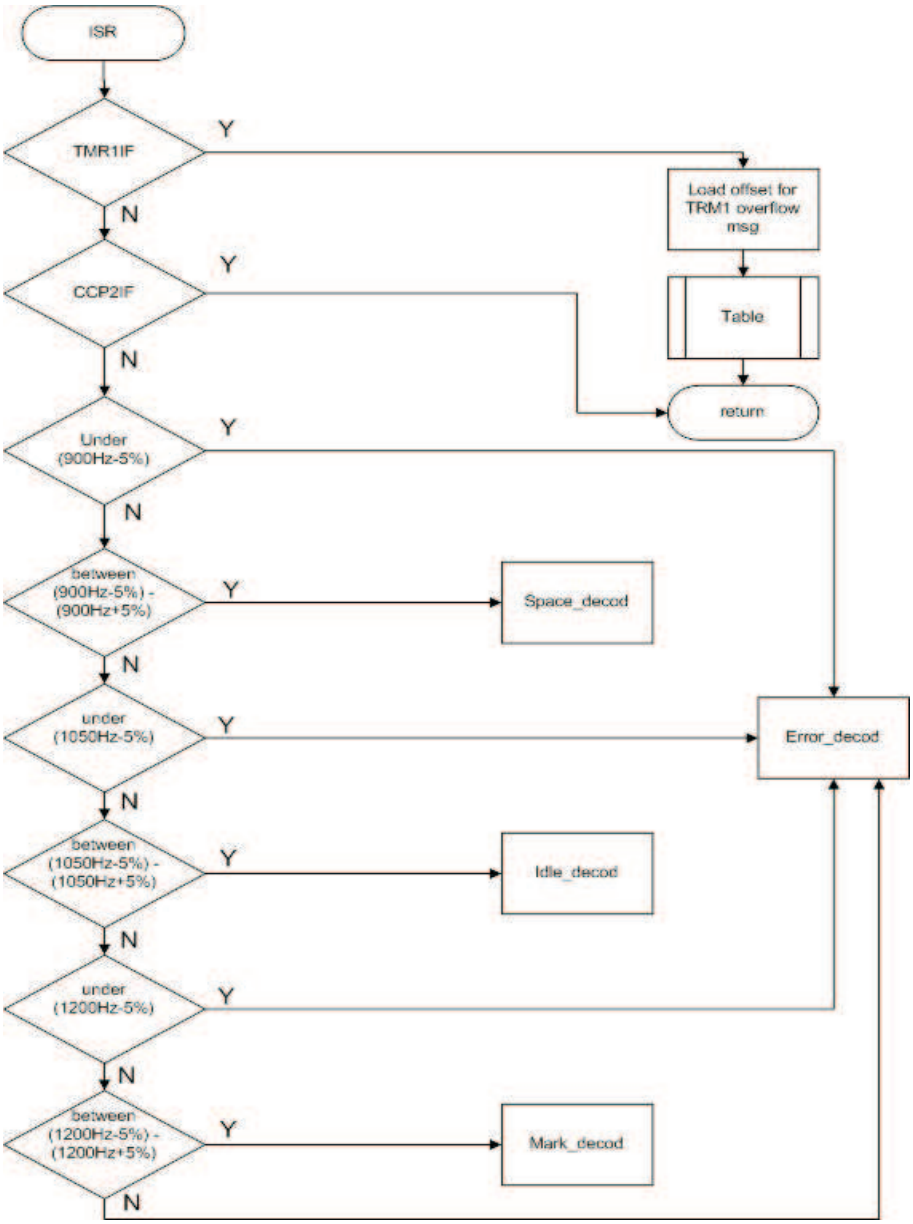

Figura 10: Interrupt Service Routine demodulation

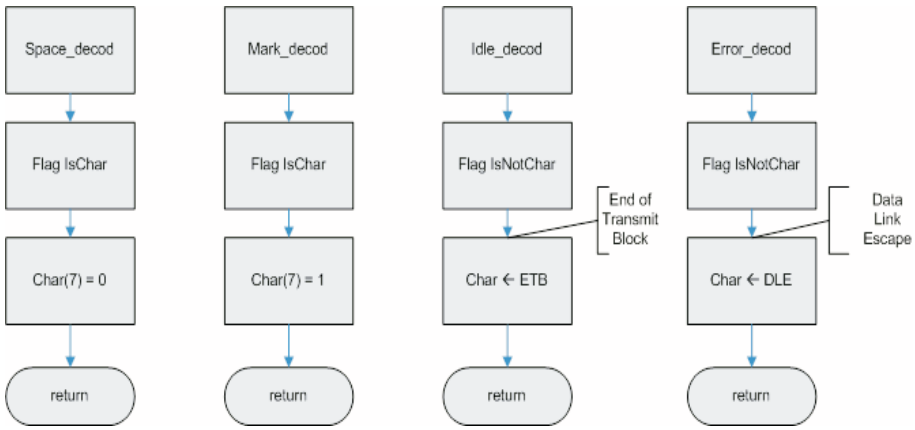

Figura 11: Interrupt Service Routine cont. demodulation

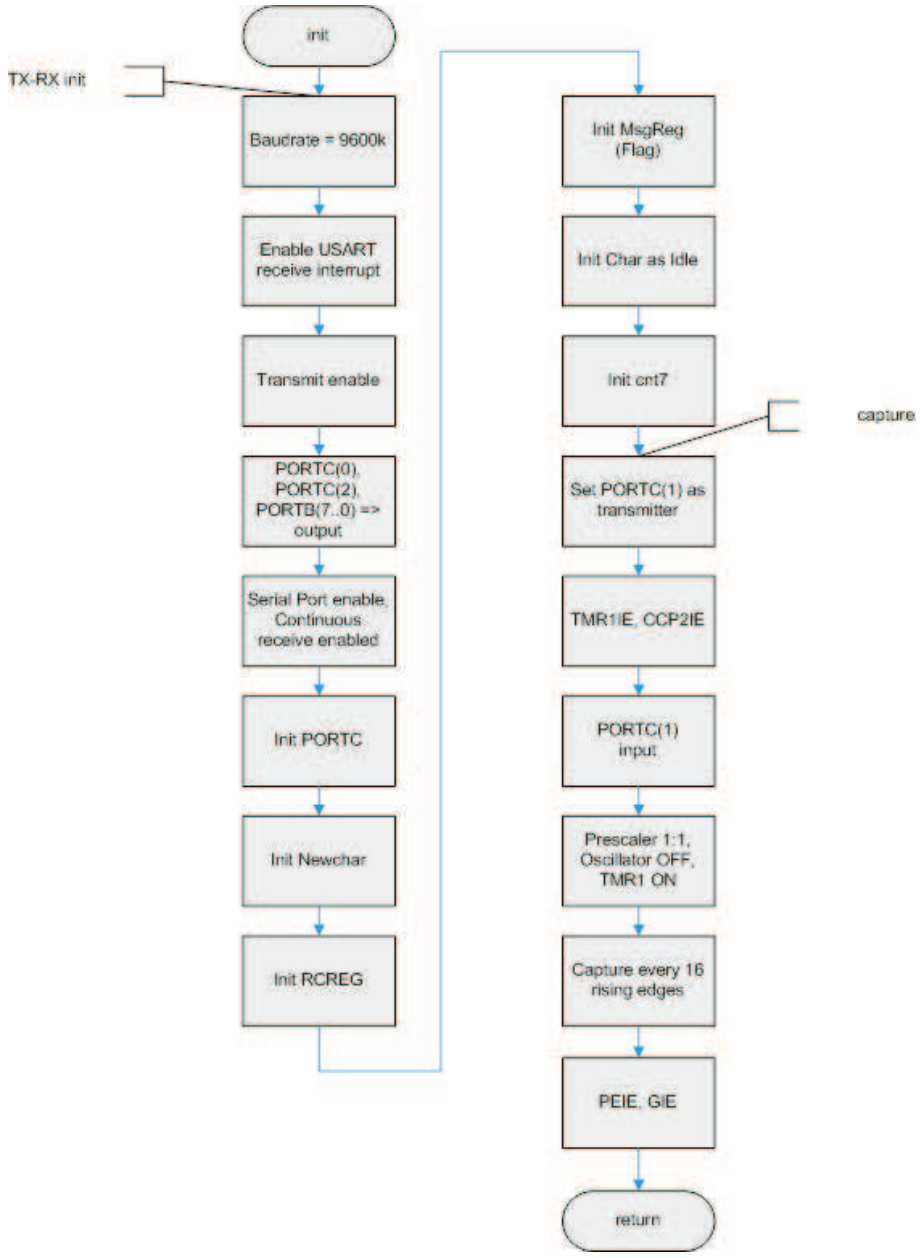

Figura 12: Initialisation demodulation

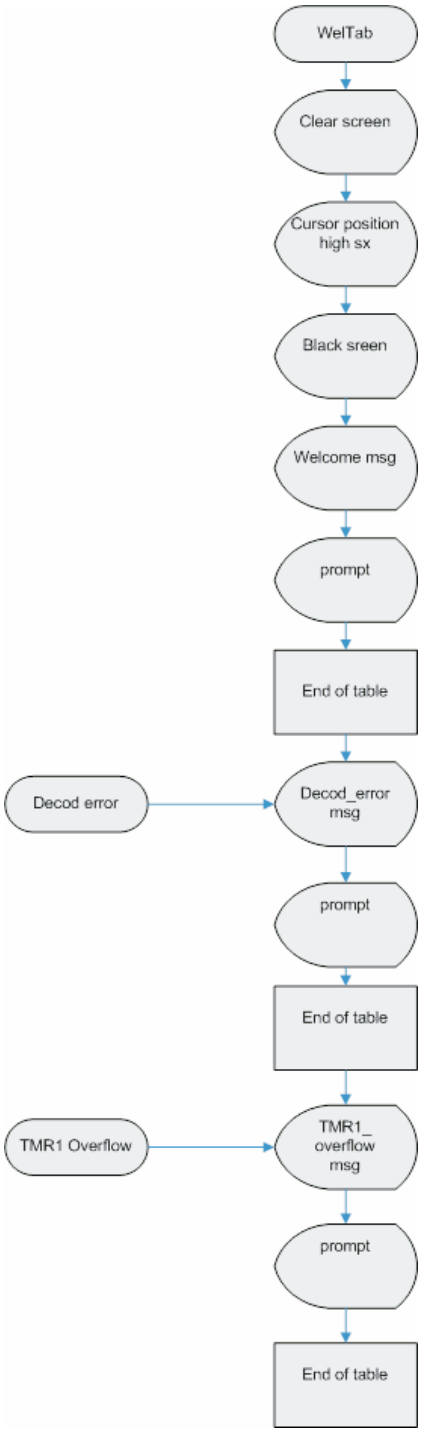

Figura 13: Table demodulation

3 Codice

Il codice del programma é il seguente.

3.1 FSK modulation

```
;*****************************************************************
;    Filename:           fsk_m.asm
;    Date:               May 2006
;    File Version:       v6.0
;
;    Authors:            Maiolo & Camozzi
;    Company:            SUPSI_DTI
;    Project:            FSK modulation
;    Module:             SSL
;*****************************************************************
;    Files required:     mydef.inc, P16F877.INC, myVXdef.inc
;*****************************************************************
;    Notes: FSK MODULATION: first it appears a
;    header on the terminal, then the user can write a text,
;    the PIC does the eco on the terminal and it modules the text.
;    If nothing is sent (IDLE), the PIC modules with 1050Hz, if
;    a text is sent, instead, it modules from the LSB to the MSB
;    the ASCII code of that character.
;    The '1' is codificaded with 1200HZ, the '0' with 900Hz.
;    We transmit with PORTC(1), because PORTC(2) is utilised
;    by the USART, so we must utilise the CCP2 module.
;    The PWM duty-cycle is set to 50%.
;    PWM period=(PR2+1)*4*TOSC*(TRM2 prescaler)
;    PWM dutycycle=(CCPR2L:CCP2CON<5:4>*TOSC*(TRM2 prescaler)
;
;*****************************************************************
;    State: Simulated seems to work, but not completed guaranteed
;*****************************************************************
;*****************************************************************
;------------   Specs definition for the Assembler  -------
;*****************************************************************

    LIST  p=16F877, t=ON
#include "P16F877.INC"
#include "mydef.inc"

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

```
;------------   Specs definition for the application  -----------
;*****************************************************************

; Variables    for myVXdef.inc
    cblock        0x20
        w_temp
        status_temp
        pclath_temp
        fsr_temp
    endc
;TX-RX variables
    cblock        0x24
        RXdata
        TXdata
        Toffset
        Newchar
    endc
;FSK variables
    cblock        0x28
        tmp
        cnt8
        reg_shift
    endc

;General constants
LSB                equ        0
RC1                equ     1                    ;PORTC(1) transmit

;TX-RX constants
TX9600             equ     .25
CTS                equ     2
CR                 equ     H'0D'
LF                 equ     H'0A'
ESC                equ     H'1B'

;FSK constants
PR2_900HZ          equ     .255
PR2_1050HZ         equ     .237
PR2_1200HZ         equ     .207
duty_cycleL        equ     .8

ByteLength         equ     .8
```

```
;*****************************************************************
;-----------    some specs  ------------------------------------
;*****************************************************************

#include "myVXdef.inc"

;*****************************************************************
;-------------  Application  ------------------------------------
;*****************************************************************

main            call    init
appl            call    welcome
loop            btfss   Newchar, LSB    ;test if a newchar is come
                goto    idle_freq
                call    echo            ;echo on Terminal and fsk
                goto    label_loop      ;modulation
idle_freq       call    idle
label_loop      goto    loop

;*****************************************************************
;------------   Tasks  -----------------------------------------
;*****************************************************************

;---------------------------------------
;-------------  Header  ----------------
;---------------------------------------
welcome         clrf    Toffset
msg             movf    Toffset, w
                call    WelTab
                movwf   TXdata
                movf    TXdata, same
                btfsc   STATUS,Z
                goto    endSend
                movf    TXdata, w
TXbusy          btfss   PIR1, TXIF      ;if TXIF='1', USART tran-
                goto    TXbusy          ;smit buffer is empty
                movwf   TXREG
                incf    Toffset, same
                goto    msg
endSend         return

;---------------------------------------
;-------------  Echo&FSK  --------------
```

```
;--------------------------------------
echo            bcf     Newchar, LSB
                movlw   H'0A'    ;H'30'
                xorwf   RXdata, w
                btfsc   STATUS, Z
                goto    EndEcho
                movf    RXdata, w      ;send (echo) of char
                movwf   TXREG

                movwf   reg_shift
                movlw   ByteLength
                movwf   cnt8
rot_right       RRF     reg_shift,same
                btfsc   STATUS, C
                goto    f1200Hz
                goto    f900Hz

cnt             decfsz  cnt8, same
                goto    rot_right
EndEcho         return

;--------------------------------------
;-------------  Frequencys  ------------
;--------------------------------------
f1200Hz         movlw    PR2_1200HZ
                banksel  PR2
                movwf    PR2
                banksel  PORTA
                bcf      tmp, 0
wait1           btfss    tmp, 0
                goto     wait1
                goto     cnt
;--------------------------------
f900Hz          movlw    PR2_900HZ
                banksel  PR2
                movwf    PR2
                banksel  PORTA
                bcf      tmp, 0
wait2           btfss    tmp, 0
                goto     wait2
                goto     cnt
;--------------------------------
idle            nop
```

```
f1050           movlw    PR2_1050HZ
                banksel  PR2
                movwf    PR2
                banksel  PORTA
                bcf      tmp, 0
wait3           btfss    tmp, 0
                goto     wait3
                return

;*****************************************************************
;------------   interrupt service routine  ----------------------
;*****************************************************************

ISR             btfsc    PIR1, RCIF
                goto     RX_service

                btfsc    PIR1, TMR2IF
                goto     TMR2_service
                goto     EISR

RX_service      bcf      PIR1, RCIF
                movf     RCREG, w
                movwf    RXdata
                bsf      Newchar, LSB
                goto     EISR

TMR2_service    bcf      PIR1, TMR2IF
                bcf      PIR2, CCP2IF
                bsf      tmp, 0

EISR            return

;*****************************************************************
;-------------  INIT  ------------------------------------------
;*****************************************************************

;---------------------------------------
;-------------  TX-RX initialisation  --
;---------------------------------------


init            nop
                banksel  TXSTA
```

```
                bsf     TXSTA, BRGH       ;high speed Baud rate
                movlw   TX9600
                movwf   SPBRG             ;FOSC=4MHz,Baudrate 9.6K
                bsf     PIE1, RCIE        ;enable USART Receive Int.
                bsf     TXSTA, TXEN       ;transmit enable
                bcf     TRISC, CTS        ;PORTC(2)=>output
                bcf     TRISC, LSB        ;PORTC(0)=>output
                clrf    TRISB             ;PORTB(7..0)=>output
                banksel RCSTA
                movlw   H'90'
                movwf   RCSTA             ;SPEN:enable,CREN:enable
                bsf     RCSTA, SPEN       ;continuous receive
                bcf     PORTC, CTS
                bsf     PORTC, LSB

                clrf    Newchar

                movf    RCREG, w           ;flush register
                movf    RCREG, w
                movf    RCREG, w

;--------------------------------------
;-------------  FSK initialisation  ----
;--------------------------------------
                movlw   PR2_1050HZ         ;PWM period=(PR2+1)
                banksel PR2                ;*4*TOSC*(TRM2 prescaler)
                movwf   PR2                ;init as idle
                banksel PORTA

                movlw   duty_cycleL        ;PWM duty cycle =
                movwf   CCPR2L             ;CCPR2L:CCP2CON<5:4>*
                bsf     CCP2CON, CCP2Y     ;TOSC*(TRM2 prescaler)
                bsf     CCP2CON, CCP2X
                bsf     CCP2CON, CCP2M3    ;PWM mode
                bsf     CCP2CON, CCP2M2
                banksel TRISC
                bcf     TRISC, RC1         ;PORTC(1)=>output
                bsf     PIE1, TMR2IE       ;TMR2 enable interrupt
                bsf     PIE2, CCP2IE       ;enable CCP2 interrupt
                banksel PORTA
                movlw   H'7C'
                movwf   T2CON              ;Postscale 1:16, TMR2 ON,
                                           ;Prescaler 1:1
```

```
                clrf     tmp

;--------------------------------------
;-------------  General initialisation
;--------------------------------------

                bsf      INTCON, PEIE
                bsf      INTCON, GIE
Einit           return

;*****************************************************************
;-------------  TABLE  -----------------------------------
;*****************************************************************
WelTab          addwf    PCL, same
                dt       ESC, "[2J"
                dt       ESC, "[1;1H"
                dt       ESC,"[?5h"
                dt       "*******************************",CR,LF
                dt       "   ---------  FSK  ----------   ",CR,LF
                dt       CR, LF, LF
                dt       "fsk>", 0

    end
```

3.2 FSK demodulation

```
;*****************************************************************
;    Filename:        fsk_d.asm
;    Date:            May 2006
;    File Version:    v6.0
;
;    Authors:         Maiolo & Camozzi
;    Company:         SUPSI_DTI
;    Project:         FSK demodulation
;    Module:          SSL
;*****************************************************************
;    Files required:  mydef.inc, P16F877.INC, myVXdef.inc
;*****************************************************************
;    Notes: FSK DEMODULATION: the receiver-PIC attached
;    to the transmitter-PIC send an header and the prompt,
;    then it put itself in wait mode. When it receive the
;    charachter, it does the eco on the terminal and when
;    it has reiceived 16 rising edges it calculates the
;    frequency of the received signal.
;    If the frequency is a '0' or '1' logical it wait until
;    it has received 8 of them it put and then it gave back
;    the ASCII character corrispondenting. In case of IDLE
;    frequence nothing will be written, instead if the
;    IDLE frequence is neighter a '0' or a '1' an error is shown.
;
;*****************************************************************
;    State:     Simulated but it doesn't work correctly
;*****************************************************************
;*****************************************************************
;------------   Specs definition for the Assembler  -------------
;*****************************************************************

    LIST  p=16F877, t=ON         ;default settings
#include "P16F877.INC"           ;must include this packages
#include "mydef.inc"

;*****************************************************************
;------------   Specs definition for the application  -----
;*****************************************************************

; Variables    for myVXdef.inc
    cblock        0x20
        w_temp
```

```
        status_temp
        pclath_temp
        fsr_temp
    endc
;TX-RX variables
    cblock        0x24
        RXdata
        TXdata
        Toffset
        Newchar
    endc
;FSK variables
    cblock        0x28
        MsgReg
        cnt7
        char
    endc

;General constants
LSB             equ     0
MSB             equ     7
RC1             equ     1       ;PORTC(1) receive

;TX-RX constants
TX9600          equ     .25
CTS             equ     2
CR              equ     H'0D'
LF              equ     H'0A'
ESC             equ     H'1B'

;FSK constants

s1              equ     .197    ;available range og freq
s2              equ     .217
s3              equ     .225
s4              equ     .248
s5              equ     .242
s6L             equ     .11
s6H             equ     .1

IsChar          equ     H'00'
IsNotChar       equ     H'FF'
DLE             equ     H'10'   ;Data Link Escape
```

```
ETB             equ     H'17'   ;End of Transmit Block
SByteLenght     equ     .7

;*****************************************************************
;-----------    some specs  ------------------------------------
;*****************************************************************

#include "myVXdef.inc"

;*****************************************************************
;-------------  Application  ------------------------------------
;*****************************************************************

main            call    init
                call    welcome
loop            movlw   IsNotChar
                xorwf   MsgReg, w
                btfsc   STATUS,Z
                goto    InC         ;IsNotChar
;- - - - - - - - - - - - - - - - - - ;...else
IsC             RRF     char, same ;IsChar
                decfsz  cnt7, same
                goto    Endloop
                movlw   SByteLenght
                movwf   cnt7
                movf    char, w
                movwf   TXREG       ;send char
                goto    Endloop
;- - - - - - - - - - - - - - - - - -
InC             movlw   DLE         ;test if a demodulation error
                xorwf   char, w     ;is occurred
                btfsc   STATUS, Z
                goto    Endloop     ;idle
;               movlw   .55
;               call    send_E

Endloop         goto    loop

;*****************************************************************
;------------   Tasks  -----------------------------------------
;*****************************************************************

;-------------------------------------
```

```
;-------------  Header  ----------------
;----------------------------------------
welcome         clrf    Toffset
msg             movf    Toffset, w
send_E          call    WelTab
                movwf   TXdata
                movf    TXdata, same
                btfsc   STATUS,Z
                goto    endSend
                movf    TXdata, w
TXbusy          btfss   PIR1, TXIF   ;if TXIF='1', USART transmit
                goto    TXbusy       ;buffer is empty
                movwf   TXREG
                incf    Toffset, same
                goto    msg
endSend         return

;----------------------------------------
;-------------  Echo  -----------------
;----------------------------------------
echo            bcf     Newchar, LSB
                movlw   H'0A'
                xorwf   RXdata, w
                btfsc   STATUS, Z
                goto    EndEcho
                movf    RXdata, w     ;send (echo) of char
                movwf   TXREG

EndEcho         return

;*****************************************************************
;------------   interrupt service routine  ----------------------
;*****************************************************************

ISR             btfsc   PIR1, TMR1IF
                goto    TMR1_service

                btfsc    PIR2, CCP2IF
                goto    CCP2_service
                goto    EISR

TMR1_service    bcf     PIR1, TMR1IF
;               movlw   .75            ;not so clean
```

```
;               call    send_E
                goto    EISR

CCP2_service    bcf     PIR2, CCP2IF
                movf    CCPR2H, same    ;value of TMR1 are <255
                btfss   STATUS, Z
                goto    over1200

                movlw   s1
                subwf   CCPR2L, w
                btfsc   STATUS, C       ;too little=>out of sx_bound
                goto    Error_decod

                movlw   s2
                subwf   CCPR2L, w
                btfsc   STATUS, C
                goto    Space_decod

                movlw   s3
                subwf   CCPR2L, w
                btfsc   STATUS, C
                goto    Error_decod

                movlw   s4
                subwf   CCPR2L,w
                btfsc   STATUS, C
                goto    Idle_decod

                movlw   s5
                subwf   CCPR2L, w
                btfsc   STATUS, C
                goto    Error_decod

over1200        movlw   s6H
                subwf   CCPR2H, w
                btfsc   STATUS, C
                goto    Error_decod
                movlw   s6L
                subwf   CCPR2L, w
                btfsc   STATUS, C
                goto    Mark_decod
                                        ;else...
Error_decod     movlw    IsNotChar
```

```
                movwf    MsgReg
                movlw    DLE
                movwf    char
                goto     EISR

Idle_decod      movlw    IsNotChar
                movwf    MsgReg
                movlw    ETB
                movwf    char
                goto     EISR

Space_decod     movlw    IsChar
                movwf    MsgReg
                bcf      char, MSB     ;it comes before the LSBs...
                goto     EISR          ;then for 7 RRF => MSB

Mark_decod      movlw    IsChar
                movwf    MsgReg
                bsf      char, MSB

EISR            return

;*****************************************************************
;-------------  INIT  ----------------------------------------
;*****************************************************************

;---------------------------------------
;-------------  TX-RX initialisation  --
;---------------------------------------

init            nop
                banksel  TXSTA
                bsf      TXSTA, BRGH    ;high speed Baud rate
                movlw    TX9600
                movwf    SPBRG          ;FOSC=4MHz,Baudrate 9.6K
                bsf      PIE1, RCIE     ;enable USART Receive Interrupt
                bsf      TXSTA, TXEN    ;transmit enable
                bcf      TRISC, CTS     ;PORTC(2)=>output
                bcf      TRISC, LSB     ;PORTC(0)=>output
                clrf     TRISB          ;PORTB(7..0)=>output
                banksel  RCSTA
                movlw    H'90'
                movwf    RCSTA           ;SPEN:serial port enable,
```

```
                bsf     RCSTA, SPEN     ;continuous receive CREN:enable
                bcf     PORTC, CTS
                bsf     PORTC, LSB

                clrf    Newchar

                movf    RCREG, w        ;flush register
                movf    RCREG, w
                movf    RCREG, w

                movlw   IsNotChar       ;init as idle
                movwf   MsgReg
                movlw   DLE
                movwf   char
                movlw   SByteLenght
                movwf   cnt7
;---------------------------------------
;-------------  init capture   ---------
;---------------------------------------
                banksel PIE2
                bsf     PIE2, CCP2IE
                bsf     PIE1, TMR1IE
                bsf     TRISC, RC1      ;PORTC(1)=>input
                banksel PORTA
                movlw   H'01'
                movwf   T1CON           ;Prescaler 1:1,Oscillator shut-off,
                movlw   H'07'           ;internal clock, TMR1 on
                movwf   CCP2CON         ;capture every 16 rising edges
;---------------------------------------
;-------------  General initialisation
;---------------------------------------

                bsf     INTCON, PEIE
                bsf     INTCON, GIE
Einit           return

;*****************************************************************
;-------------  TABLE  -----------------------------------
;*****************************************************************
WelTab          addwf   PCL, same
                dt      ESC, "[2J"
                dt      ESC, "[1;1H"
                dt      ESC,"[?5h"
```

```
                dt      "*------------  FSK  -------------*",CR,LF
                dt      CR, LF, LF
                dt      "fsk>", 0

;Table_error    dt      CR, LF
;               dt      "Decod_error",CR,LF
;               dt      "fsk>", 0
;
;TMR1_overflow  dt      CR, LF
;               dt      "*TO*",CR,LF, 0
    end
```

4 Osservazioni

Lo stato attuale del programma é il seguente: I programmi compilano entrambi, la modulazione é stata testata con l'oscilloscopio, al primo impatto sembra corretto ma bisogna trovare il modo di verificare che effettivamente la modulazione avvenga come desiderato. La demodulazione é in a fase di test ma i primi risultati ci mostrano che vi sono errori. Il PIC scrive a terminale anche in caso di Idle. Come discusso assieme le prime modifiche verteranno a scrivere a terminale il valore della frequenza misurata. Di seguito riportiamo l'utilizzo della memoria e il rapporto di compilazione di entrambi i programmi ma per il momento é presto tirare conclusioni visto che non é ancora tutto sistemato.

4.0.1 FSK modulation

```
MEMORY USAGE MAP ('X' = Used,  '-' = Unused)

0000 : XXX-XXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0040 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0080 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
00C0 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXX-----------

All other memory blocks unused.

Program Memory Words Used:   244
Program Memory Words Free:  7948

Errors   :     0
Warnings :     0 reported,     0 suppressed
Messages :    14 reported,     0 suppressed
```

4.0.2 FSK demodulation

```
MEMORY USAGE MAP ('X' = Used,  '-' = Unused)

0000 : XXX-XXXXXXXXXXX XXXXXXXXXXXXXXX XXXXXXXXXXXXXXX XXXXXXXXXXXXXXX
0040 : XXXXXXXXXXXXXXX XXXXXXXXXXXXXXX XXXXXXXXXXXXXXX XXXXXXXXXXXXXXX
0080 : XXXXXXXXXXXXXXX XXXXXXXXXXXXXXX XXXXXXXXXXXXXXX XXXXXXXXXXXXXXX
00C0 : XXXXXXXXXXXXXXX XXXXXXXXXXXXXXX XXXXXXXXXXXXXXX XXXXXXXXXXXXXXX
0100 : XXXXXXXXXXXXXXX XXX------------ ---------------- ----------------

All other memory blocks unused.

Program Memory Words Used:   274
Program Memory Words Free:  7918

Errors   :     0
Warnings :     0 reported,     0 suppressed
Messages :    10 reported,     0 suppressed
```

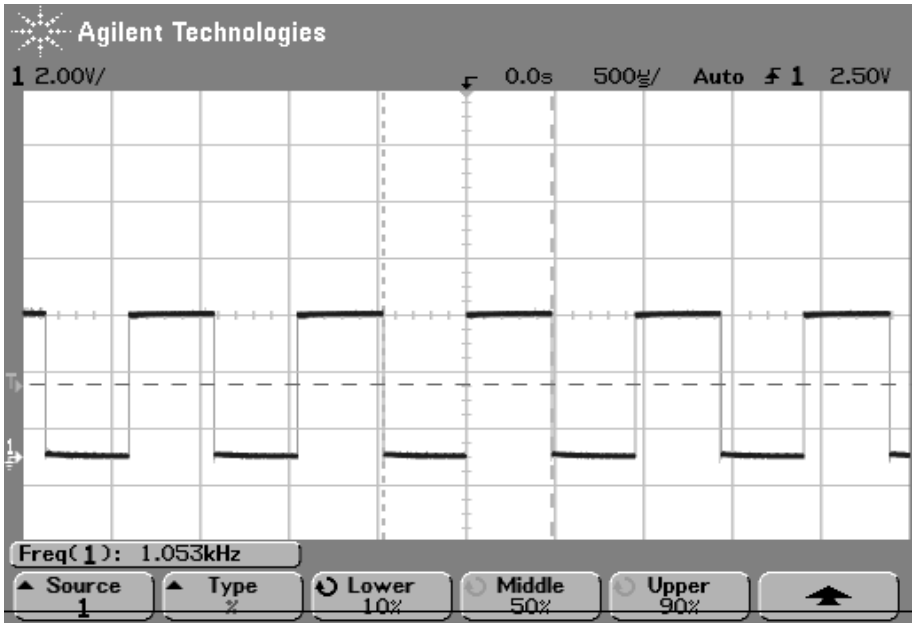

Figura 14: Frequenza di Idle

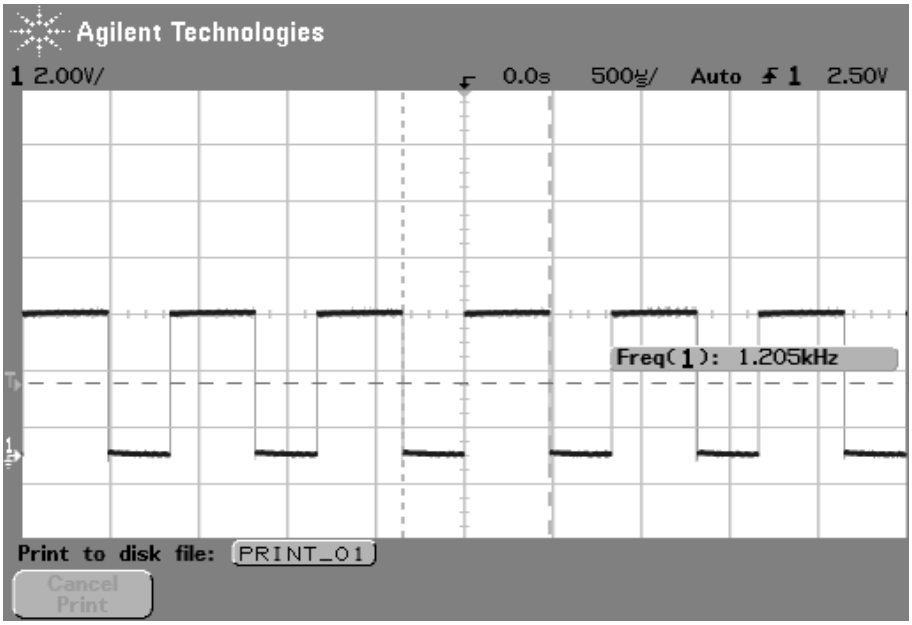

Figura 15: Frequenza di '1' logico

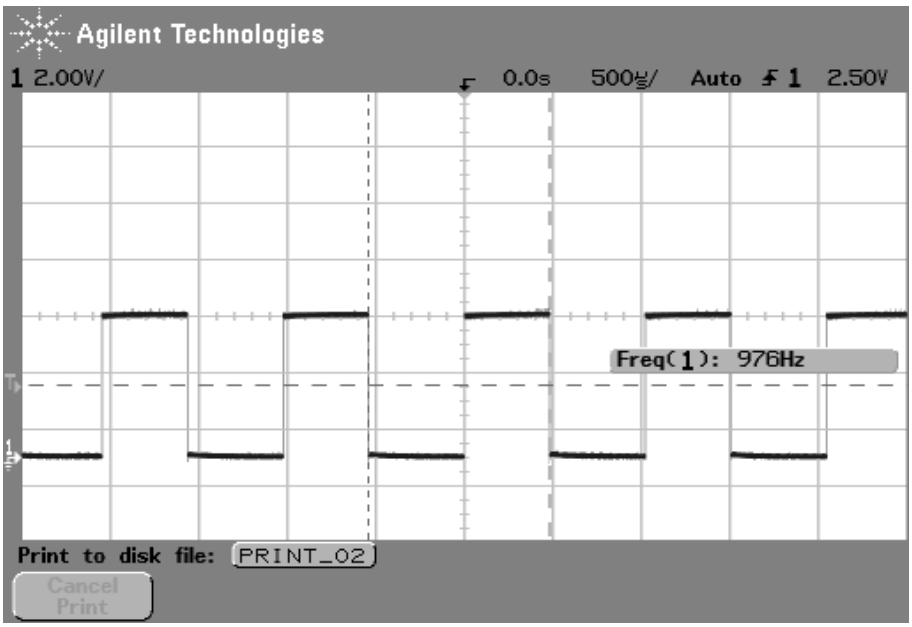

Figura 16: Frequenza di '0' logico

5 Conclusioni

I programmi non sono ancora funzionanti al 100% perció questa parte verrá discussa piú avanti.