APPLICAZIONI

AP

APPLICAZIONI

Conversione di dati analogici

Possiamo affermare che il PIC16F84 è un microcontroller povero ma essenziale, perché nonostante le scarse risorse di cui dispone al suo interno, permette la realizzazione di un'infinità di applicazioni molto utili. Comunque, che cosa succede quando le risorse di cui abbiamo bisogno non ci sono nel PIC? Anche se la filosofia dei microcontroller è funzionare come sistemi chiusi, dipendendo cioè, unicamente dalle risorse interne e senza connessione ad altri dispositivi, esiste la possibilità di comunicare con periferiche che apportino quello che i PIC non hanno.

Proviamo a fare uso di queste caratteristiche, per disporre di un convertitore che si incarichi di trasformare valori analogici in valori digitali che possano essere processati dal PIC16F84.

In ogni caso, la scelta più sensata sarebbe selezionare un PIC che contenga un convertitore, ad esempio uno della gamma dei PIC16F87x.

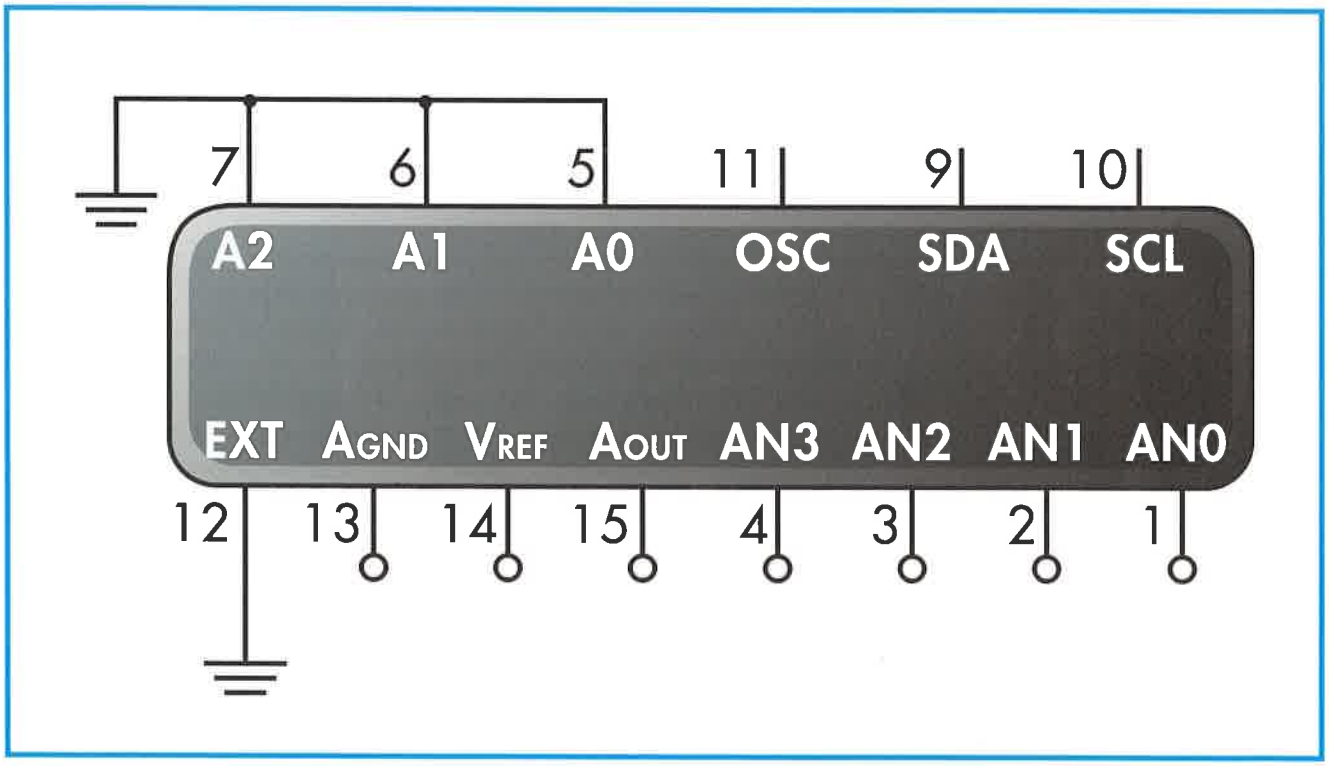
IL MODULO ADC/DAC

Il convertitore che utilizzeremo ha come sigla PCF8591 e si collega al PIC tramite le linee del bus I2C, SCL e SDA.

Nella figura possiamo osservarne l'aspetto. L'indirizzo a cui dovrà fare riferimento il master è determinato dal codice 1001, stabilito in fabbrica, ha tre bit che dipendono dai valori assegnati alle linee A0, A1 e A2, che nella figura sono collegati a massa.

Il bit meno significativo degli indirizzi indica se si va a leggere (1) o se si va a scrivere (0) sul dispositivo.

Il circuito si comporterà come un convertitore analogico/digitale a 8 bit, convertendo i valori che arrivano tramite le linee AN0, AN1, AN2 e AN3, e passando il



Il modulo ADC/DAC PCF8591.

risultato come valore binario al PIC. Inoltre può funzionare come un convertitore digitale/analogico, convertendo il valore binario che fornisce il PIC, in una tensione che esce tramite AOUT. I segnali di massa e di riferimento si collegano ai pin AGND e VREF.

A seconda delle caratteristiche della tensione di riferimento si otterrà maggiore o minore precisione.

ESERCIZIO PROPOSTO

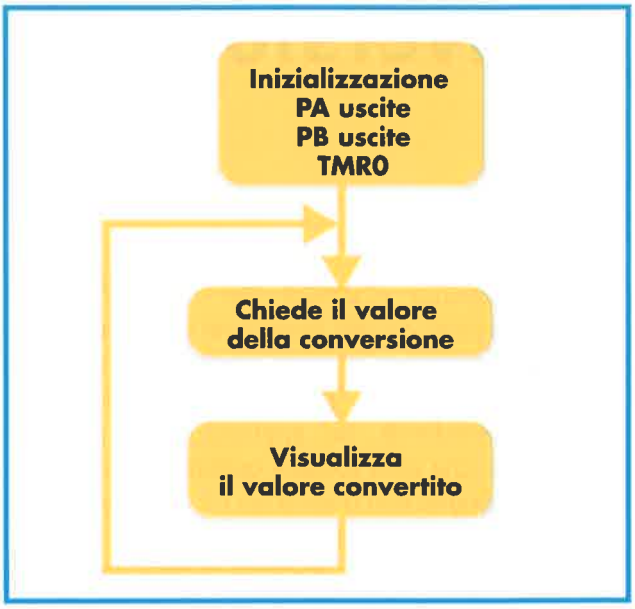
Il programma deve solo prendere un valore e mostrarlo convertito. Immaginate che il parametro sia quello di una temperatura o di un sensore di luminosità, e che in funzione del valore si debba attivare un ventilatore, un termoconvettore, una luce, ecc.

Affrontiamo quindi il lavoro per parti, e dedichiamoci, per ora, solamente alla conversione.

Sarà tanto semplice quanto realizzare un ciclo di lettura sul dispositivo PCF8591. Se si prende come tensione di riferimento 5 V, la risoluzione per bit sarà di 20 mV, come conferma l'equazione della figura e come risulta anche dalla tabella in cui ogni incremento di 20 mV aumenterà di 1 il valore binario.

Il programma legge il valore analogico presente sul canale 0, lo converte in digitale e lo mostra tramite i diodi LED. Si utilizza il bus I2C per mettere in comunicazione il PIC con il dispositivo convertitore.

Enunciato dell'esercizio proposto.



Organigramma dell'esercizio proposto.

$$\text{Risoluzione} = \frac{\text{Vref+} - \text{Vref-}}{2^{\text{N° bit}}} = \frac{5 - 0}{2^8} = \frac{5}{256} \simeq 20\ \text{mV/bit}$$

Tensione analogica	Valore Binario
0V	0000 0000
20mV	0000 0001
40mV	0000 0010
60mV	0000 0011
...	...
5V	1111 1111

Equazione che dà la risoluzione di un convertitore e la relativa tabella.

Come mostra l'organigramma, dal modulo principale si configurano solo le variabili che si vogliono utilizzare e si eseguono le chiamate alle differenti funzioni, che saranno quelle che supporteranno il carico reale del programma. Il modo in cui queste funzioni si realizzeranno dipenderà dal dispositivo: raccomandiamo di leggere sempre le specifiche tecniche per seguire i passi ed i tempi indicati dal costruttore.

PROGRAMMA

Dopo la definizione delle variabili dovremo includere un file con le routines di gestione per il bus I2C che saranno utilizzate più avanti. Configureremo la porta A come uscita, utilizzando le quattro linee meno significative per acquisire parte del valore convertito, cioè i

```
                LIST            P=16F84A        ; Tipo di processore
                INCLUDE         "P16F84A.INC"   ; Definizione dei registri interni

CLKIN           EQU             .4000000        ; Frequenza di lavoro

I2c_var         EQU             0x0c            ; Inizio delle variabili per le routines I2C_LOW
A_inp           EQU             0x12            ; Risultato della conversione AD di ingresso
aux             EQU             0x13            ; Variabile ausiliaria dedicata al dato binario

                ORG             0x00            ; Vector reset
                goto            Inizio

                INCLUDE         "I2CLOW.INC"    ; Include le routines di gestione del bus I2C

A_Input         call            Txt_Start_Bit   ; Trasmette la sequenza di inizio
                movlw           b'10010001'
                movwf           DataByte
                call            Txt_Byte        ; Indirizzo del PCF8591 (modo lettura)
                bsf             _Ultimo_Byte    ; Riceve un unico byte per volta
                call            RcvByte         ; Riceve il risultato della conversione AD
                call            Txt_Stop_Bit    ; Trasmette sequenza di STOP
                movf            DataByte,W
                movwf           A_inp           ; Salva il valore della conversione
                return

Visualizzare    movf            A_inp,0         ; Spostiamo il risultato in W
                andlw           b'00001111'     ; Lavoriamo con la parte bassa del registro
                movwf           PORTA           ; e la spostiamo sulla Porta·A
                movf            A_inp,0         ; Spostiamo nuovamente il risultato in W
                andlw           b'11110000'     ; e lavoriamo con la parte alta
                movwf           aux             ; spostandola su di un registro ausiliario
                swapf           aux,0           ; Cambiamo di posizione ai bit
                movwf           PORTB           ; e li portiamo sulla Porta B
                return

Inizio          clrf            PORTA
                clrf            PORTB
                bsf             STATUS,RP0      ; Seleziona il banco 1
                movlw           b'00000000'
                movwf           TRISA           ; RA0-RA4 uscite
                clrf            TRISB           ; Porta B uscita
                movlw           b'10000110'     ; Prescaler di 128 per il TMR0. Pull_up Off
                movwf           OPTION_REG
                bcf             STATUS,RP0      ; Seleziona il banco 0

                call            Inizia_Bus      ; Inizializza il bus I2C
```

Programma risolto.

```
Loop    clrwdt                          ; Resetta il WDT
        call    A_Input                 ; Ottiene il valore di conversione ADC
        call    Visualizzare            ; Visualizza sui LED il risultato

        goto    Loop

        END                             ; Fine del programma
```

Programma risolto (continuazione).

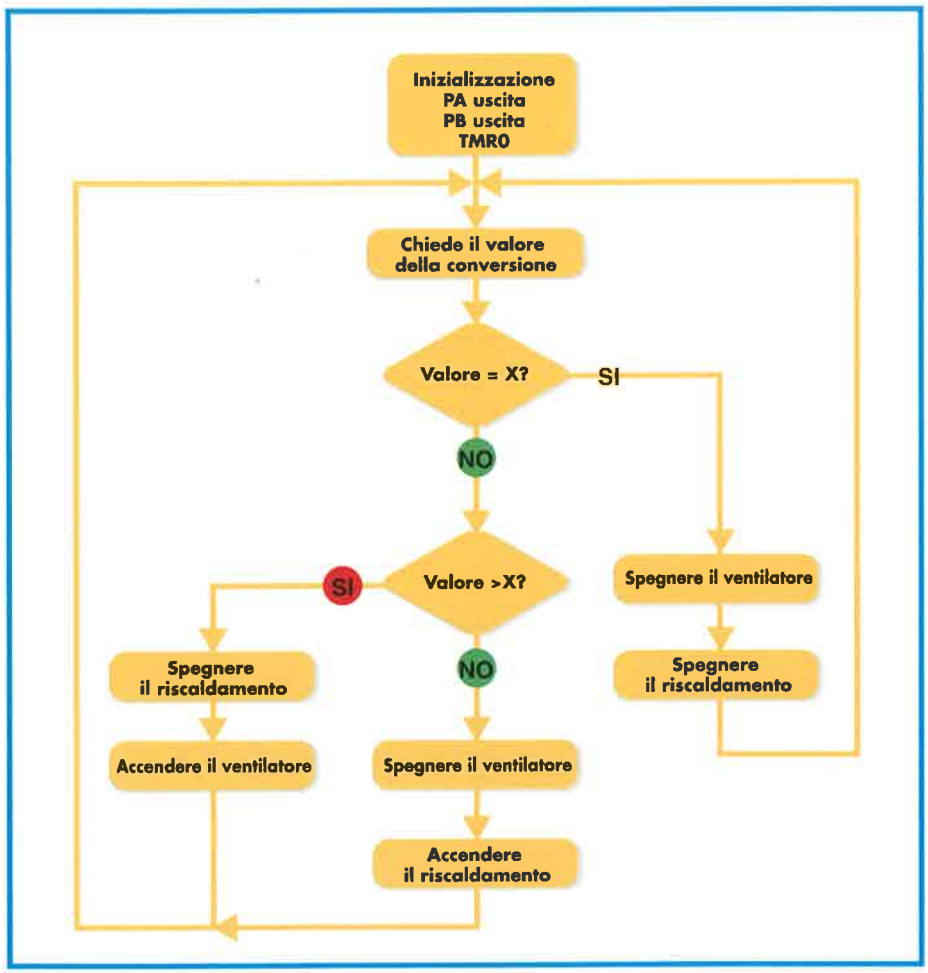
quattro bit meno significativi del numero. Allo stesso modo la porta B acquisirà la restante parte del numero, vale a dire i 4 bit più significativi. Inoltre, le linee RB6 e RB7 saranno utilizzate per la comunicazione con il convertitore I2C.

La routine "A_input" legge il valore binario che arriva dal canale 0 dell'ADC contenuto nel dispositivo I2CPCF8591. La subroutine "Visualizzare" prende il valore convertito dalle porte A e B, nel modo precedentemente descritto. Il modo di realizzare la comunicazione con il convertitore viene indicata dal fabbricante di quest'ultimo.

Dopo un bit di inizio dobbiamo indirizzarlo, indicando se vogliamo leggere o scrivere. Nel nostro caso stiamo leggendo, dato che stiamo cercando di acquisire la conversione di una tensione analogica a binaria. L'opzione di scrittura per questo dispositivo consiste nel fargli produrre la tensione corrispondente al valore binario che introduciamo.

Dato che inviamo solo un byte, se ne vogliamo conoscere il valore, e leggere il valore convertito, possiamo fermare il seguito della sequenza.

Tutte queste routines sono raccolte nel file "I2CLOW.INC". Invece di utilizzare le porte A e B per acquisire il valore della conversione in un modo così ricercato, si potrebbe optare per l'utilizzo di un altro dispositivo, anch'esso I2C, a cui passare il dato con due soli fili, e quest'ultimo visualizzerebbe il dato su dei LED.



Organigramma della nuova proposta.

NUOVA PROPOSTA

Proponiamo ora una modifica all'esercizio precedente.

A seconda che il valore sia superiore, inferiore o uguale ad uno preso come riferimento, si accenderà un riscaldatore, un ventilatore o si spegnerà tutto.

Comunicazione fra dispositivi

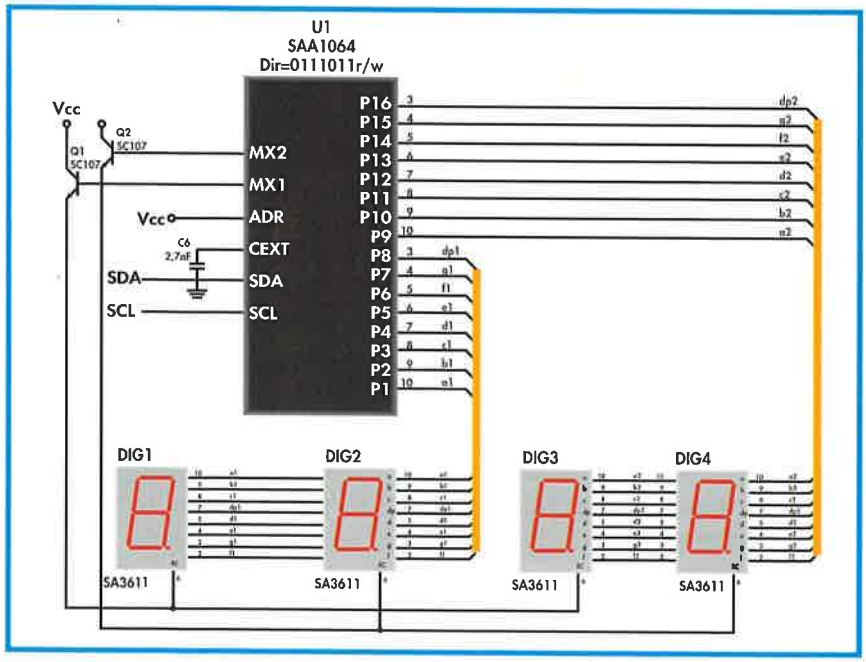
Così come abbiamo visto con il convertitore DAC/ADC PCF8591, esistono molti altri dispositivi predisposti per comunicare con il bus I2C. Questo ci fornisce la possibilità di lavorare con 256 elementi con un impiego minimo di linee, nella fattispecie due. In questo caso vogliamo aggiungere alla funzione di conversione, quella di visualizzazione. Per fare questo utilizzeremo contemporaneamente due dispositivi I2C, in modo che i dati generati da uno, passino all'altro. La comunicazione fra entrambi i dispositivi si realizza tramite un microcontroller, il PIC16F84.

IL CONTROLLORE DI DISPLAY SAA1064

Questo integrato è capace di pilotare fino a 4 display a 7 segmenti. Il suo impiego è semplice: sostanzialmente, dobbiamo inviare la combinazione binaria corrispondente ai segmenti che vanno attivati per ognuno dei 4 digit. Per farlo funzionare con i suddetti display si può collegare come mostrato nella figura. Tramite le linee SCL e SDA il microcontroller manda al dispositivo l'informazione da visualizzare. Dei sette bit di indirizzo, i cinque più significativi sono predeterminati in fabbrica come "01110" e i due successivi sono stabiliti dall'utente secondo la tensione applicata al pin 1 (ADR), la cui corrispondenza si trova nella tabella allegata. L'ultimo bit, è quello di lettura (1) o scrittura (0) del dispositivo. Il SAA1064 funziona come un multiplexer, attivando consecutivamente i display 1 e 3 e i display 2 e 4. Se questo si fa con una velocità adeguata, si provoca la sensazione ottica che i quattro display siano attivati in modo continuo. Il circuito dispone inoltre di funzioni di reset, test, controllo di luminosità, ecc.

Tensione applicata ad ADR	Due bit meno significativi dell'indirizzo
GND	00
3/8 di Vcc	01
5/8 di Vcc	10
Vcc	11

Tabella che determina i due bit meno significativi dell'indirizzo.



Il SAA1064 collegato a quattro display a 7 segmenti.

ESERCIZIO PROPOSTO

Questo programma è simile al primo che abbiamo realizzato con il convertitore, cambia solo il modo di offrire il risultato finale, che sarà un valore compreso fra 0000 e 0255, rappresentato sui quattro display a 7 segmenti. Ancora una volta il programma principale è incaricato di eseguire le chiamate alle diverse funzioni, che realizzano realmente il lavoro. Le specifiche di questo dispositivo indicheranno il modo con cui si realizzerà la comunicazione fra essi e il PIC.

PROGRAMMA

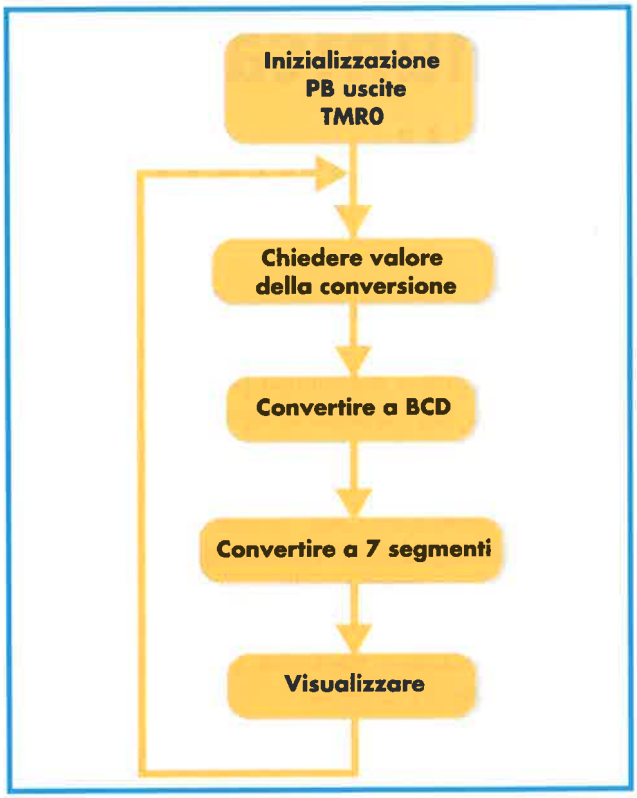
Il programma principale è molto simile a quello del convertitore che abbia-

Il programma legge il valore analogico presente sul canale 0, lo converte in BCD e lo mostra tramite quattro display a 7 segmenti. Si utilizza il bus I2C per mettere in comunicazione il PIC con il dispositivo convertitore e con il controllore di display.

Enunciato dell'esercizio proposto.

mo già visto. Si richiede l'acquisizione di un valore analogico convertito a binario, questo valore binario si converte in BCD, che torna a convertirsi questa volta in codice a 7 segmenti, per rappresentarlo sui display. Si utilizzano come linee di I/O solo le due corrispondenti al bus I2C, situate sulla porta B, che sono condivise con i due dispositivi utilizzati. Come routines nuove abbiamo quelle di visualizzazione del dato, di conversione del dato del codice binario BCD e quella di decodifica del numero BCD.

Per la conversione del dato a BCD si parte da un valore memorizzato nel registro di lavoro W che è convertito in codice BCD e memorizzato nei due registri da 8 bit, in modo che in ogni registro ci siano due digit.



Organigramma dell'esercizio proposto.

```
                LIST        P=16F84A        ; Tipo di processore
                INCLUDE     "P16F84A.INC"   ; Definizione dei registri interni

CLKIN           EQU         .4000000        ; Frequenza di lavoro

I2c_var         EQU         0x0c            ; Inizio delle variabili per la routine I2C_LOW
Digit_1         EQU         0x12            ; Buffer di uscita dei 4 digit per il SAA1064
Digit_2         EQU         0x13
Digit_3         EQU         0x14
Digit_4         EQU         0x15
A_inp           EQU         0x16            ; Risultato della conversione AD di ingresso
Buffer_H        EQU         0x17            ; Risultato della conversione in BCD
Buffer_L        EQU         0x18

                ORG         0x00            ; Vector reset
                goto        Inizio

                INCLUDE     "I2CLOW.INC"    ; Include la routine di gestione del bus I2C

A_Input         call        Txt_Start_Bit   ; Trasmette la sequenza di inizio
                movlw       b'10010001'
                movwf       DataByte
                call        Txt_Byte        ; Indirizzo del PCF8591 (modo lettura)
                bsf         _Ultimo_Byte    ; Riceve un unico byte per volta
                call        RcvByte         ; Riceve il risultato della conversione AD
                call        Txt_Stop_Bit    ; Trasmette la sequenza di STOP
                movf        DataByte,W
                movwf       A_inp           ; Salva il valore risultante dalla conversione
                return
```

```
Visualizzare    call    Inizia_Bus      ;Inizializza la linea del bus I2C
                call    Txt_Start_Bit   ; Invia il codice di inizio
                movlw   b'01110110'
                movwf   DataByte
                call    Txt_Byte        ; Trasmette l'indirizzo del dispositivo (scrittura)
                clrf    DataByte
                call    Txt_Byte        ; Trasmette il byte di istruzione
                movlw   b'01000111'
                movwf   DataByte
                call    Txt_Byte        ; Trasmette byte di controllo
                movf    Digit_1,W
                movwf   DataByte
                call    Txt_Byte        ; Trasmette Digit 1
                movf    Digit_2,W
                movwf   DataByte
                call    Txt_Byte        ; Trasmette Digit 2
                movf    Digit_3,W
                movwf   DataByte
                call    Txt_Byte        ; Trasmette Digit 3
                movf    Digit_4,W
                movwf   DataByte
                call    Txt_Byte        ; Trasmette Digit 4
                call    Txt_Stop_Bit    ; Trasmette la sequenza di STOP
                return

Bin_bcd         clrf    Buffe_L         ; Cancella i registri di lavoro
                clrf    Buffer_H

Bin_bcd_1       addlw   0xf6            ; Sottrae 10 per somma di complemento a 2
                btfss   STATUS,C        ; È presente il Carry?
                goto    Bin_bcd_3       ; NO
                movwf   A_inp           ; SI. Scrivilo nel registro temporale
                incf    Buffer_L,F      ; Incrementare byte basso, scriverlo e inviarlo...
                movf    Buffer_L,W       ... al registro W...
                xorlw   b'00001010'     ; ... xor di 0Ah con W (BCDL)
                btfss   STATUS,Z        ; Il risultato di questa operazione è zero?
                goto    Bin_bcd_2       ; NO.
                clrf    Buffer_L        ; SI. Cancella il byte basso e...
                incf    Buffer_H,F      ; incrementare quello alto.
Bin_bcd_2       movf    A_inp,W         ; Recupera il dato.
                goto    Bin_bcd_1       ; Continua le operazioni.

Bin_bcd_3       addlw   H'0A'           ; TEMPO + 0Ah
                swapf   Buffer_L,F      ; <3:0> <==> <7:4> e ==> BCDL
                iorwf   Buffer_L,F       ; W o BCDL ==> BCDL
                return

Un_Pack         swapf   Buffer_H,W
                call    BCD_7seg
                movwf   Digit_1         ; Converte il nibble più significativo del Buffer_H
                movf    Buffer_H,W
                call    BCD_7seg
                movwf   Digit_2         ; Converte il nibble meno significativo del Buffer_H
                swapf   Buffer_L,W
                call    BCD_7seg
                movwf   Digit_3         ; Converte il nibble più significativo del Buffer_L
                movf    Buffer_L,W
                call    BCD_7seg
                movwf   Digit_4         ; Converte il nibble meno significativo del Buffer_L
                return
```

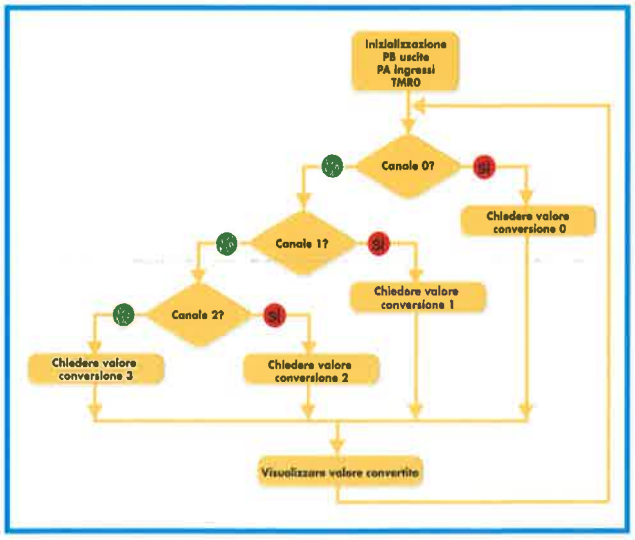
```
BCD_7seg    andlw   b'00001111'
            addwf   PCL,F           ; Spostamento sulla tabella
            retlw   b'00111111'     ; 0
            retlw   b'00000110'     ; 1
            retlw   b'01011011'     ; 2
            retlw   b'01001111'     ; 3
            retlw   b'01100110'     ; 4
            retlw   b'01101101'     ; 5
            retlw   b'01111101'     ; 6
            retlw   b'00000111'     ; 7
            retlw   b'01111111'     ; 8
            retlw   b'01100111'     ; 9
            retlw   b'01110111'     ; A
            retlw   b'01111100'     ; B
            retlw   b'00111001'     ; C
            retlw   b'01011110'     ; D
            retlw   b'01111001'     ; E
            retlw   b'01110001'     ; F

Inizio      clrf    PORTB
            bsf     STATUS,RP0      ; Seleziona il banco 1
            clrf    TRISB           ; Porta B uscita
            movlw   b'10000110'     ; Prescaler da 128 per il TMR0. Pull_Up Off
            movwf   OPTION_REG
            bcf     STATUS,RP0      ; Seleziona il banco 0

            call    Inizia_Bus      ; Inizializza il bus I2C

Loop        clrwdt                  ; Resetta il WDT
            call    A_Input         ; Acquisisce il valore della conversione ADC
            movf    A_inp,W
            call    Bin_bcd         ; Lo converte in BCD
            call    Un_Pack         ; Spacchetta e converte in codice 7 segmenti
            call    Visualizzare    ; Visualizza sui 4 display del SAA1064
            goto    Loop

            END                     ; Fine del programma
```

Programma risolto (continuazione).

Prima di poter visualizzare il risultato, dobbiamo dividerlo in quattro digit o decodificarlo, e trasformare ognuno di essi nel suo corrispondente codice a 7 segmenti. La routine di visualizzazione è quella che utilizza il dispositivo SAA1964. In essa, dopo il bit di inizio e il suo indirizzamento per la scrittura, si invia la funzione che si vuole eseguire. Di seguito si trasmettono, ad uno ad uno, i digit da visualizzare, già convertiti in codice a 7 segmenti.

NUOVA PROPOSTA

Pensate a una nuova possibilità: ad esempio, che ve ne pare di disporre di diversi sensori, uno per ogni canale del convertitore, e scegliere quale convertire a seconda del valore dei due interruttori? Il risultato può essere mostrato tramite dei LED, o con l'aiuto del nuovo dispositivo che abbiamo appena conosciuto.



Organigramma della nuova proposta.

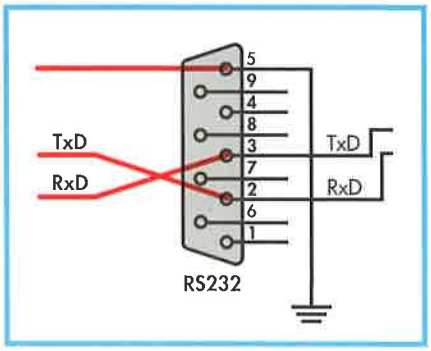
Applicazione pratica: comunicazione con il computer

Se l'utilizzo della tastiera e dell'LCD come periferiche di un PIC vi sono sembrate una grande invenzione, aspettate di vedere cosa abbiamo preparato in questa occasione: niente più e niente meno che la comunicazione del PIC con il computer. Per questo utilizzeremo un canale RS-232 che comunica in modo seriale con la porta del PC. Quest'ultimo, grazie al software HyperTerminal (Hypertrm) fornito da Windows, si comporterà come un terminale di comunicazione.

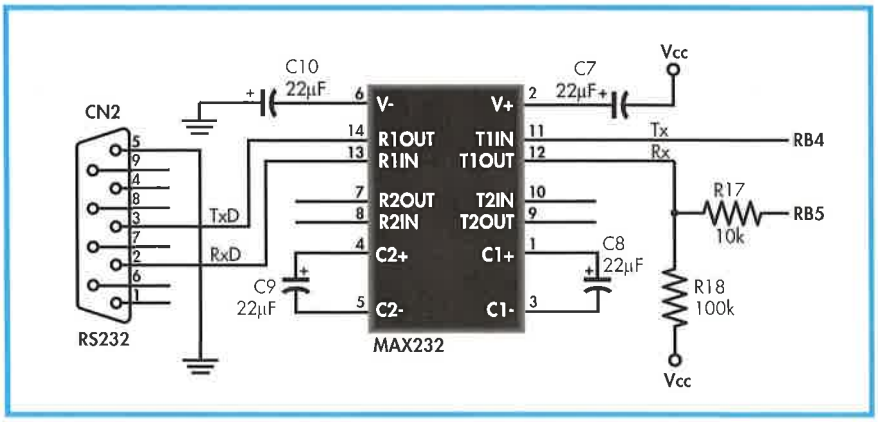
IL CANALE SERIALE RS-232

Per fare comunicare il PIC con il computer non basta utilizzare lo stesso "linguaggio", ma devono essere uguali anche i suoi livelli elettrici.

Per ottenere questo si utilizza il circuito integrato MAX232, il quale ha il compito di convertire i livelli logici TTL, provenienti dal PIC, a livelli logici RS-232, che si applicano al connettore del computer.



Il cavo seriale deve avere le linee di trasmissione e ricezione incrociate.



Il piedino RB4 del PIC sarà la linea di trasmissione dei dati, e il piedino RB5 la linea di ricezione. Nel connettore DB9 del computer saranno i pin 3 e 2 rispettivamente. Il cavo seriale che unisce il computer con il PIC tramite il MAX232 deve avere le linee di trasmissione e di ricezione incrociate, dato che quello inviato dal PIC tramite la linea di trasmissione deve arrivare alla periferica sulla linea di ricezione e viceversa.

Si desidera controllare, tramite un terminale, una serie di dispositivi di allarme quali possono essere i diodi LED, che mostrano in ogni momento la situazione di diversi sensori. Si seguirà la seguente tabella:

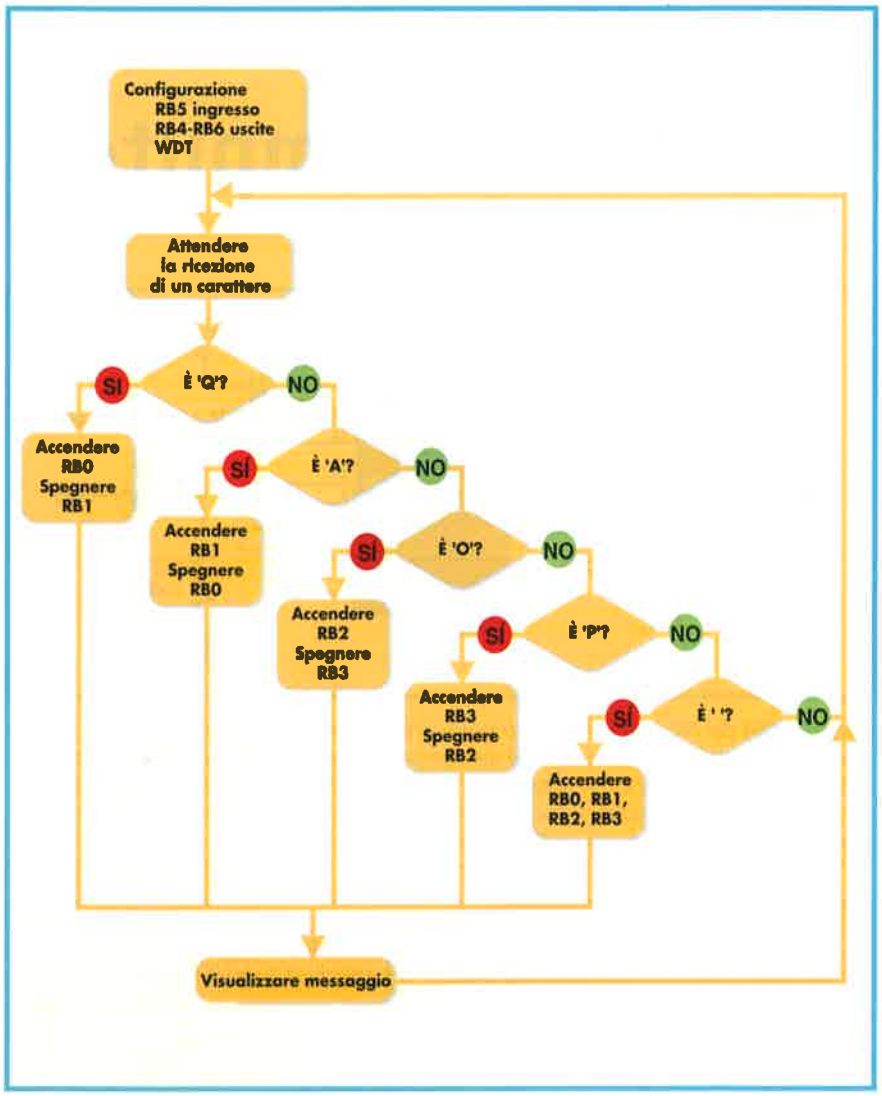
TASTO	AZIONE
Q	pericolo di incendio
A	NO pericolo di incendio
O	pericolo intrusi
P	NO pericolo intrusi
SPC	scollegamento

Premere uno di questi pulsanti farà cambiare lo stato delle linee RB3-RB0, quelle che ci mostrano lo stato dei sensori. Dato che abbiamo due LED per ogni situazione di allarme al posto di uno, dobbiamo assicurarci che i LED spenti lo siano per errore del sistema e non per problemi di montaggio del circuito. Solo nel caso di "disconnessione" non ci sarà alcun LED acceso; in tutti gli altri casi dovranno esserci due LED su quattro accesi nella posizione adeguata.

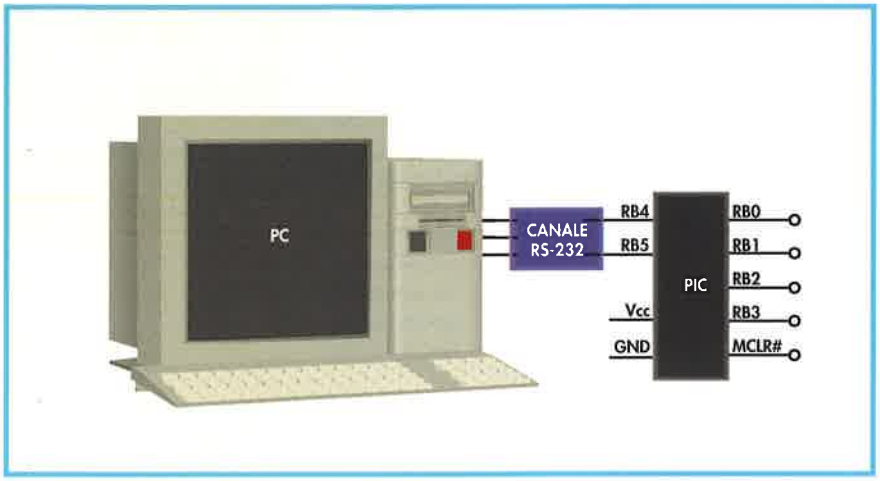
USCITA	ALLARME
RB0	pericolo incendio
RB1	NO pericolo incendio
RB2	pericolo intrusi
RB3	NO pericolo intrusi

Il PIC oltre a mostrare le informazioni tramite i LED fornirà il tipo di situazione al terminale perché sia mostrata in video.

Enunciato dell'esercizio proposto.



Schema dell'esercizio proposto.



Organigramma dell'esercizio proposto.

Quando è il computer che manda i dati al PIC, il MAX232 ha il compito di fare la conversione inversa.

Nella figura possiamo vedere i pochi componenti esterni di cui ha bisogno questo chip per lavorare. Come vedete, le possibilità che offre la comunicazione con un computer sono enormi. Il PC potrebbe trovarsi in una abitazione, e il sistema di controllo in un'altra, anche a notevole distanza.

Oltre a gestire dei semplici LED, potremo anche movimentare un microrobot per realizzare un lavoro, o un sistema di riscaldamento che si attiverà o disattiverà a seconda degli ordini dell'utente o delle ore del giorno.

Lo schema, in questo caso, non si limita agli elementi da collegare al PIC, ma si estende sino alla connessione con il computer. Il modulo che appare nella figura come canale RS-232 è mostrato in dettaglio in una figura precedente.

ORGANIGRAMMA

Dopo aver configurato le linee del PIC come ingressi/uscite, e il resto delle periferiche in modo adeguato, si attende di ricevere un carattere.

Si verificano ad una ad una tutte le possibilità di attivazione di una tastiera, e a seconda di quali tasti sono stati premuti, si accenderanno o si spegneranno i LED adeguati e si manderà un messaggio sul monitor che visualizza la situazione attuale del sistema.

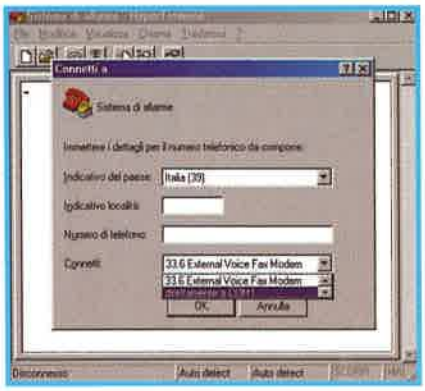
PROGRAMMA

Il programma, anche se lungo, è facile da capire.

Le variabili definite all'inizio del programma sono necessarie per la comunicazione seriale. In seguito si definiscono i differenti messaggi da visualizzare, in modo simile a quan-

```
                LIST        P=16F84A        ; Tipo di processore
                INCLUDE     "P16F84A.INC"   ; Definizione dei registri interni
CLKIN           EQU         .4000000        ; Frequenza di lavoro
BAUDE           EQU         .9600           ; Velocità di comunicazione
T_MODO          EQU         1               ; Trasmette il primo bit LSB
R_MODO          EQU         1               ; Riceve il primo bit LSB
T_Nbit          EQU         8               ; Trasmette caratteri da 8 bit
R_Nbit          EQU         8               ; Riceve caratteri da 8 bit
Sbit            EQU         2               ; Trasmette 2 bit di stop
RS232_var       EQU         0x0C            ; Inizio delle variabili per routines RS232
Contatore       EQU         0x11            ; Variabile contatore di messaggi
Offset          EQU         0x12            ; Spostamento dei messaggi
Temporale_1     EQU         0x13            ; Variabile temporale
Temporale_2     EQU         0x14            ; Variabile temporale
                ORG         0x00            ; Vector Reset
                goto        INIZIO
                ORG         0x05            ; Salta il vector di interrupt
Mess_0          EQU                         $
                DT                          "Scollegamento ",0x0d,0x00
Mess_1          EQU                         $
                DT                          "Pericolo Incendio ",0x0d,0x00
Mess_2          EQU                         $
                DT                          "No pericolo incendio ",0x0d,0x00
Mess_3          EQU                         $
                DT                          "Condizione nulla ",0x0d,0x00
Mess_4          EQU                         $
                DT                          "Pericolo intrusi ",0x0d,0x00
Mess_5          EQU                         $
                DT                          "Incendio/Intrusi ",0x0d,0x00
Mess_6          EQU                         $
                DT                          "Intrusi/No incendio ",0x0d,0x00
Mess_7          EQU                         $
                DT                          "Condizione nulla ",0x0d,0x00
Mess_8          EQU                         $
                DT                          "No Pericolo intrusi ",0x0d,0x00
Mess_9          EQU                         $
                DT                          "Incendio/No intrusi ",0x0d,0x00
Mess_10         EQU                         $
                DT                          "No incendio/Intrusi ",0x0d,0x00

                INCLUDE     "RS232LOW.INC"  ; Include le routines di comunicazione

                                            ; A seconda del valore contenuto
                                            nel registro W, viene fornito
                                            il carattere da visualizzare
Tabella_Messaggi movwf      PCL             ; Calcola lo spostamento sulla tabella
                                            ; dei messaggi: Questa routine invia
                                            tramite RS232 il messaggio
                                            il cui INIZIO è indicato
                                            ; nell'accumulatore. La fine di un
                                            messaggio si determina mediante
                                            il codice 0x00
MESSAGGIO       movwf       Temporale_1     ; Salva posizione della tabella
Messaggio_1     movf        Temporal_1,W    ; Recupera posizione della tabella
                call        Tabella_messaggi ; Cerca carattere di uscita
                movwf       Temporale_2     ; Memorizza il carattere
                movf        Temporale_2,F
                btfss       STATUS,Z        ; Controlla se è l'ultimo
                goto        Non_è_ultimo
                return
Non_è_ultimo    movwf       Txdreg
                call        TxD             ; Trasmette il carattere
                incf        Temporale_1,F   ; Carattere successivo
                goto        Messaggio_1

INIZIO          clrf        PORTB           ; Scollega uscita
                bsf         STATUS,RP0      ; Selezione banco 1 dei dati
                movlw       b'10001111'     ; Prescaler da 128 per il WDT. Pull_up

                movwf       OPTION_REG
                movlw       b'11100000'
                movwf       TRISB           ; RB5 ingresso e RB4-RB0 uscite
                bcf         STATUS,RP0      ; Seleziona pagina 0 dei dati
LOOP            clrwdt
                call        RxD             ; Attendi la ricezione di un carattere
                movlw       'Q'
```

Secondo passo nella configurazione del programma terminale "Hypertrm".



Secondo passo nella configurazione del programma terminale "Hypertrm".

to fatto lavorando con il display LCD.

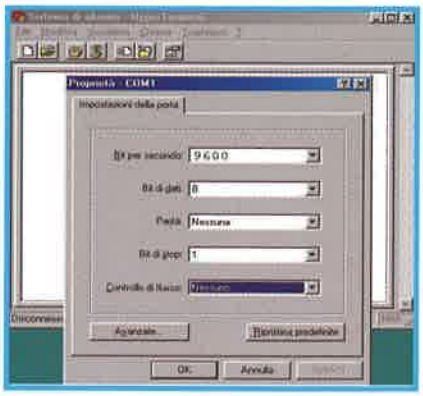
Il file "RS232LOW.INC" contiene le routines di comunicazione, per l'invio e ricezione dei dati; dato che il PIC16F84 non ha queste risorse implementate nell'hardware, queste operazioni devono essere gestite via software.

In seguito troviamo la routine incaricata di trasmettere il messaggio che fa uso della routine "TxD", contenuta in "RS232LOW", e invia un byte (un solo carattere) ogni volta che si chiama. La chiamata alla routine inversa, la "RxD", si trova nel programma principale: essa ci informerà del tasto premuto.

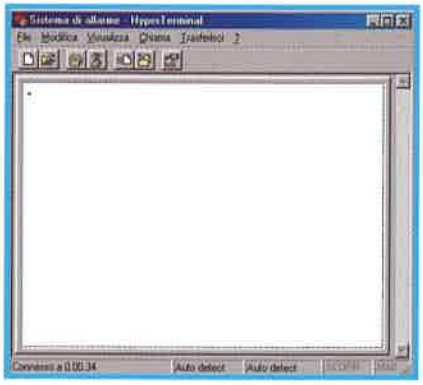
Dopo aver verificato di quale

```
                subwf   Rxdreg,W
                btfsc   STATUS,Z        ; È 'Q'?
                goto    INCENDIO        ; Si
                movlw   'A'
                subwf   Rxdreg,W
                btfsc   STATUS,Z        ; È 'A'?
                goto    NO_INCENDIO     ; Si
                movlw   'O'
                subwf   Rxdreg,W
                btfsc   STATUS,Z        ; È 'O'?
                goto    INTRUSI         ; Si
                movlw   'P'
                subwf   Rxdreg,W
                btfsc   STATUS,Z        ; È 'P'?
                goto    NO_INTRUSI      ; Si
                movlw   ' '
                subwf   Rxdreg,W
                btfsc   STATUS,Z        ; È ' '?
                goto    LOOP            ; No

SCOLLEGAMENTO   bcf     PORTB,0
                bcf     PORTB,1
                bcf     PORTB,2
                bcf     PORTB,3         ; Scollega uscite
                goto    Test_Mess       ; Determina il messaggio da dare
INCENDIO        bcf     PORTB,1         ; Scollega RB1
                bsf     PORTB,0         ; Collega RB0
                goto    Test_Mess       ; Determina il messaggio da dare
NO_INCENDIO     bcf     PORTB,0         ; Scollega RB0
                bsf     PORTB,1         ; Collega RB1
                goto    Test_Mess       ; Determina il messaggio da dare
INTRUSI         bcf     PORTB,3         ; Scollega RB3
                bsf     PORTB,2         ; Collega RB2
                goto    Test_Mess       ; Determina il messaggio da dare

NO_INTRUSI      bcf     PORTB,2         ; Scollega RB2
                bsf     PORTB,3         ; Collega RB3
                goto    Test_Mess       ; Determina il messaggio da dare

                                        ; Ora procediamo a determinare il messaggio
                                        di uscita sul terminale
                                        ; Per questo in base al contenuto della
                                        PORTA B si calcola l'offset
                                        ; del messaggio da fornire tenendo conto che
                                        i messaggi iniziano a
                                        ; partire dall'indirizzo 0x05 ed hanno una
                                        lunghezza fissa di 20 caratteri.

Test_mess       movlw   0x0a
                movwf   Contatore       ; Inizializza contatore di messaggi
                movlw   Mens_10
                movwf   Offset          ; Indirizzo dell'ultimo messaggio
Mess_loop       movf    PORTB,W
                andlw   b'00001111'     ; Legge il valore attuale della PORTA B
                subwf   Contatore,W
                btfsc   STATUS,Z        ; Corrisponde al messaggio da fornire?
                goto    Uscita_mess     ; Sì
                movlw   .20
                subwf   Offset          ; No, punta al messaggio precedente
                decf    Contatore,F
                goto    Mess_loop
Uscita_mess     movf    Offset,W
                call    MESSAGGIO       ; Uscita del messaggio appropriato
                goto    LOOP

END                                     ; Fine del programma sorgente
```

Programma risolto.



Terzo passo nella configurazione del programma terminale "Hypertrm".



Ora si può iniziare a inviare o ricevere i dati.

tasto si tratta, si impostano i bit di segnalazione secondo la tabella vista nell'enunciato del programma, e si procede a determinare quale sia il messaggio da inviare al terminale.

PROVA DEL PROGRAMMA

Per fare funzionare il programma sono necessari, oltre al PIC e ai LED, gli elementi visti nello schema generale.

Dopo aver collegato il PIC con il computer, chiameremo il programma "Hypertrm", e lo configureremo per fare in modo che segua gli stessi parametri del PIC nella comunicazione (porta, velocità, ecc.).

In seguito potremo iniziare a premere pulsanti e vedere che cosa ci risponde il PIC.