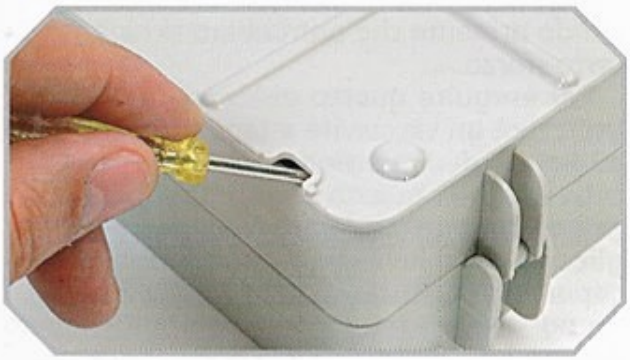


Metodo per sganciare i fermagli.

Estrazione del fermaglio.

La costruzione stessa del manico evita un'eventuale apertura accidentale del laboratorio, sia della parte che appartiene al pannello principale sia quella del pannello superiore.

Esperimenti

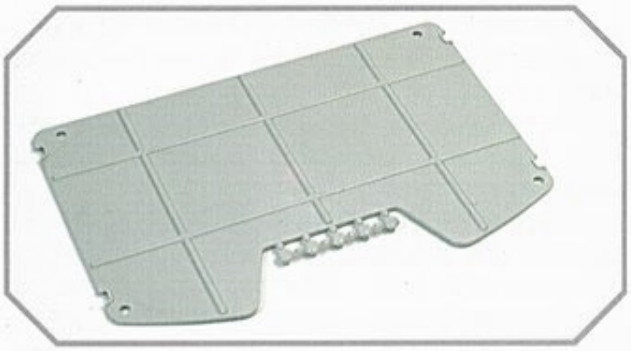
La capacità di questo laboratorio è molto grande, e può risultare estremamente utile non solo agli appassionati di elettronica, ma anche agli studenti che possono apprezzare la sua utilità sia per eseguire gli esperimenti proposti sia per progettarne dei nuovi.

Il fatto di disporre di una alimentazione variabile e di una scheda Bread Board universale, oltre al generatore di frequenza e un circuito ausiliario audio, semplifica molto il lavoro.

Il microcontroller utilizzato inoltre è piuttosto potente ed è stato spiegato in modo abbastanza dettagliato, per questo gli esperimenti e i programmi che è possibile eseguire con lo stesso sono numerosi e possono essere più complessi di quelli da noi proposti.

Vista generale del laboratorio.

Coperchio inferiore



Coperchio e fermagli.

Dettaglio di un fermaglio.

Con questo fascicolo viene fornito il coperchio del pannello principale del laboratorio e i quattro fermagli che lo fissano.

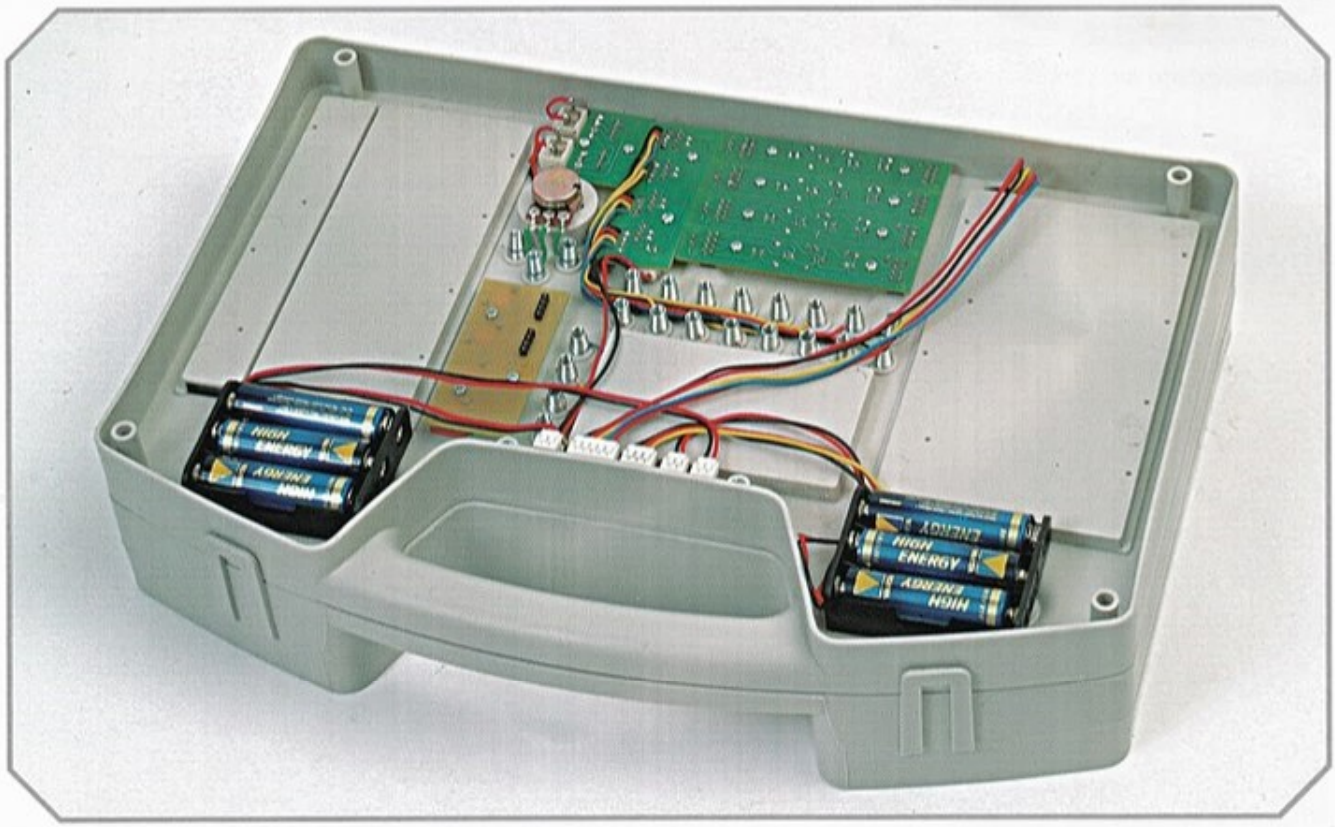
Presentazione

Prima di installare il coperchio è consigliabile dare un'ultima occhiata all'interno del pannello, per verificare che non ci sia ancora qualche lavoro da eseguire.

Il coperchio si fissa con quattro fermagli, come vedremo in modo più dettagliato in seguito. Questi fermagli possono essere estratti per aprire il coperchio, operazione che è necessario eseguire per sostituire le batterie.

Installazione

Prima di montare il coperchio del pannello principale si deve chiudere il laboratorio



Interno del pannello frontale.

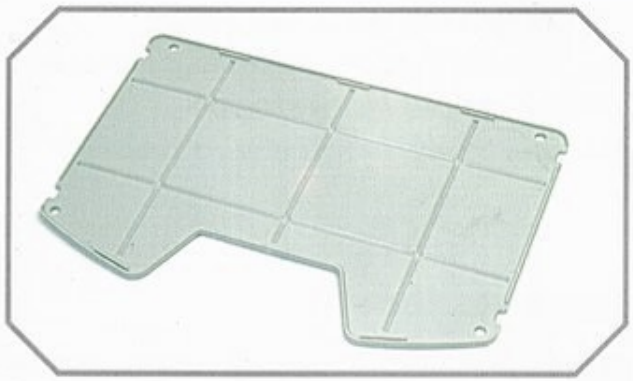


Foto dell'interno del coperchio.

Posizionamento del coperchio.

e collocarlo su un tavolo, in modo che si possa osservare l'interno del pannello principale.

I fermagli devono essere separati dal coperchio, inoltre è necessario tagliare i pezzi di plastica rimanenti.

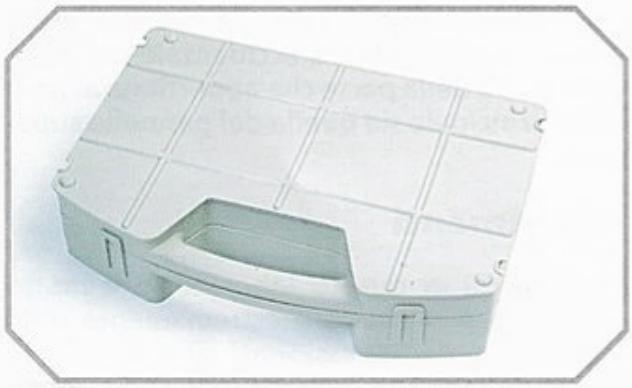
Il coperchio deve essere posizionato in modo tale che il suo bordo rimanga incastrato nel pannello principale del laboratorio, in questo modo rimarranno allineati i quattro fori del coperchio con le corrispondenti colonne perforate che utilizzeremo per il fissaggio.

Il fissaggio si ottiene grazie ai quattro fermagli di materiale plastico che si inseriscono nei fori del coperchio, affinchè rimangano "agganciati" nelle colonne, dovranno essere fissati con una semplice pressione; non è necessario utilizzare nessun tipo di colla, in quanto dovrà essere possibile rimuoverli nuovamente per sostituire le batterie scariche, o per poter effettuare un qualsiasi intervento di manutenzione o riparazione all'interno del laboratorio.

Asportazione del coperchio

Per togliere il coperchio dal laboratorio è necessario estrarre prima i quattro fermagli, te-

Inserimento di uno dei fermagli.



Coperchio installato e fissato.

Fermaglio installato.

nendo presente che potrebbero richiedere un certo sforzo.

Per eseguire questo distacco è necessario utilizzare un cacciavite a taglio di circa 5 o 6 millimetri di dimensione.

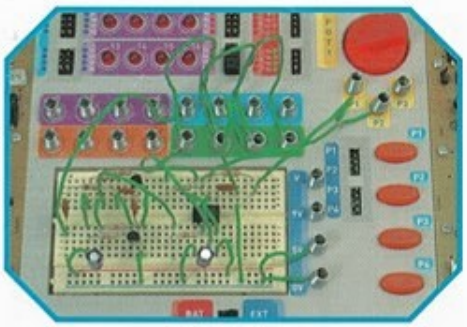
La punta di questo cacciavite si deve inserire nel foro rettangolare più vicino al fermaglio, per fare leva con molta attenzione sul coperchio dall'interno in una zona il più vicino possibile al fermaglio stesso, utilizzando come punto di appoggio il bordo del laboratorio.

Questa operazione si deve ripetere per ognuno dei quattro fermagli, fino a farli fuoriuscire per poterli togliere direttamente con la mano.

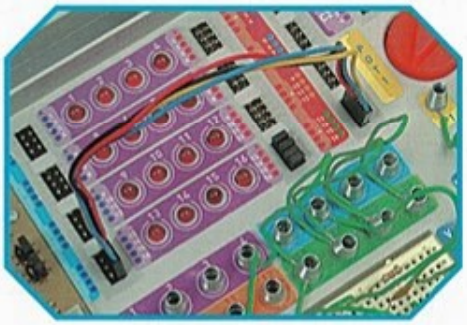
Con il laboratorio chiuso

Con il prossimo fascicolo verrà fornito l'altro coperchio, il quale, una volta installato, vi permetterà di conservare il laboratorio anche in modo verticale in una libreria, come se si trattasse di un libro, con il manico posto verso l'esterno.

Aspetto del laboratorio con la base chiusa.

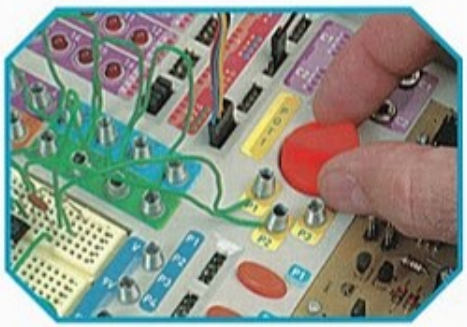


Dettaglio del collegamento alle molle.

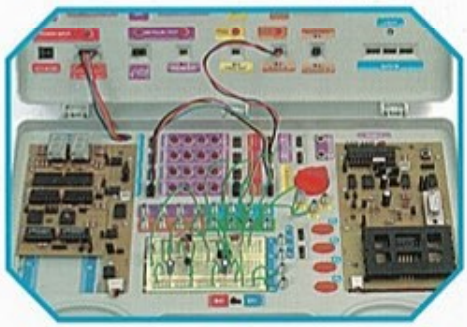


Dettaglio del collegamento ai LED.

Collegamenti all'amplificatore audio.



Ruotando POT 1 cambia la modulazione.



Esperimento completato.

La tensione di uscita terminale 3 del 555, si utilizza per illuminare la quarta fila di LED, cosa che avviene solamente quando questo livello di uscita è basso. L'uscita del VCO è portata verso l'amplificatore audio inserendo un circuito seriale formato dalla resistenza R6 e dal condensatore C6, in modo che ascoltando i suoni sull'altoparlante si possa verificare l'effetto delle variazioni di livello sull'uscita del 555.

Montaggio

In questo montaggio è necessario inserire diversi componenti sulla scheda Bread Board, quindi dovremo essere ordinati ed eseguire il lavoro lentamente per evitare errori.

I collegamenti dei catodi dei LED dal 13 al 16, si eseguono con un cavetto terminato su due connettori a quattro vie cadauno, che arriva al connettore corrispondente alle molle dalla 13 alla 16, inoltre dovremo inserire i quattro ponticelli sugli anodi di questi LED.

I collegamenti dell'uscita del VCO all'amplificatore audio si eseguono con un cavetto terminato su due connettori a due vie. Uno di questi connettori si collega a quello corrispondente alle molle 5 e 6 e l'altro capo al connettore AUDIO IN, in modo che il filo rosso corrisponda al punto rosso e quello nero al punto nero. Ricordate che l'uscita dell'amplificatore per essere collegata deve avere i due ponticelli inseriti in senso orizzontale, fra AUDIO OUT e SPEAKER IN.

Prova

Dopo aver verificato il montaggio possiamo dare alimentazione al circuito, includendo in questa operazione il commutatore AUDIO ON, in modo che funzioni l'amplificatore.

Dopo aver messo in marcia il circuito si udirà un suono sull'altoparlante, la cui frequenza si potrà modulare azionando il comando di POT 1. Ogni volta che cambia il suono i LED dal 13 al 16 si spengono o si accendono.

LISTA DEI COMPONENTI

U1	Circuito integrato 555
Q1, Q2	Transistor BC547 o BC548
R1, R2	Resistenza 47 K (giallo, viola, arancio)
R3, R6	Resistenza 100 K (marrone, nero, giallo)
R4, R5	Resistenza 1K8 (marrone, grigio, rosso)
R6, R7	Resistenza 4K7 (giallo, viola, rosso)
C1, C6	Condensatore 22 nF
C2, C3	Condensatore 10 µF elettrolitico
C4, C5	Condensatore 22 nF

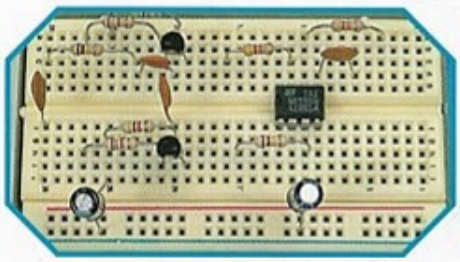
Generatore audio modulato

Questo circuito corrisponde a un VCO (Oscillatore Controllato in Voltaggio), la cui tensione di controllo è generata da un altro oscillatore.

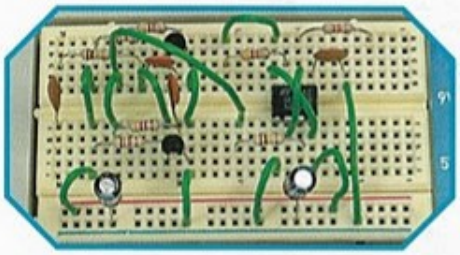
Il circuito

Osservando il circuito possiamo vedere un VCO formato principalmente dai transistor Q1 e Q2 e dai rispettivi componenti passivi associati. L'ingresso di controllo della frequenza di questo oscillatore è il punto di collegamento delle resistenze R6 e R7.

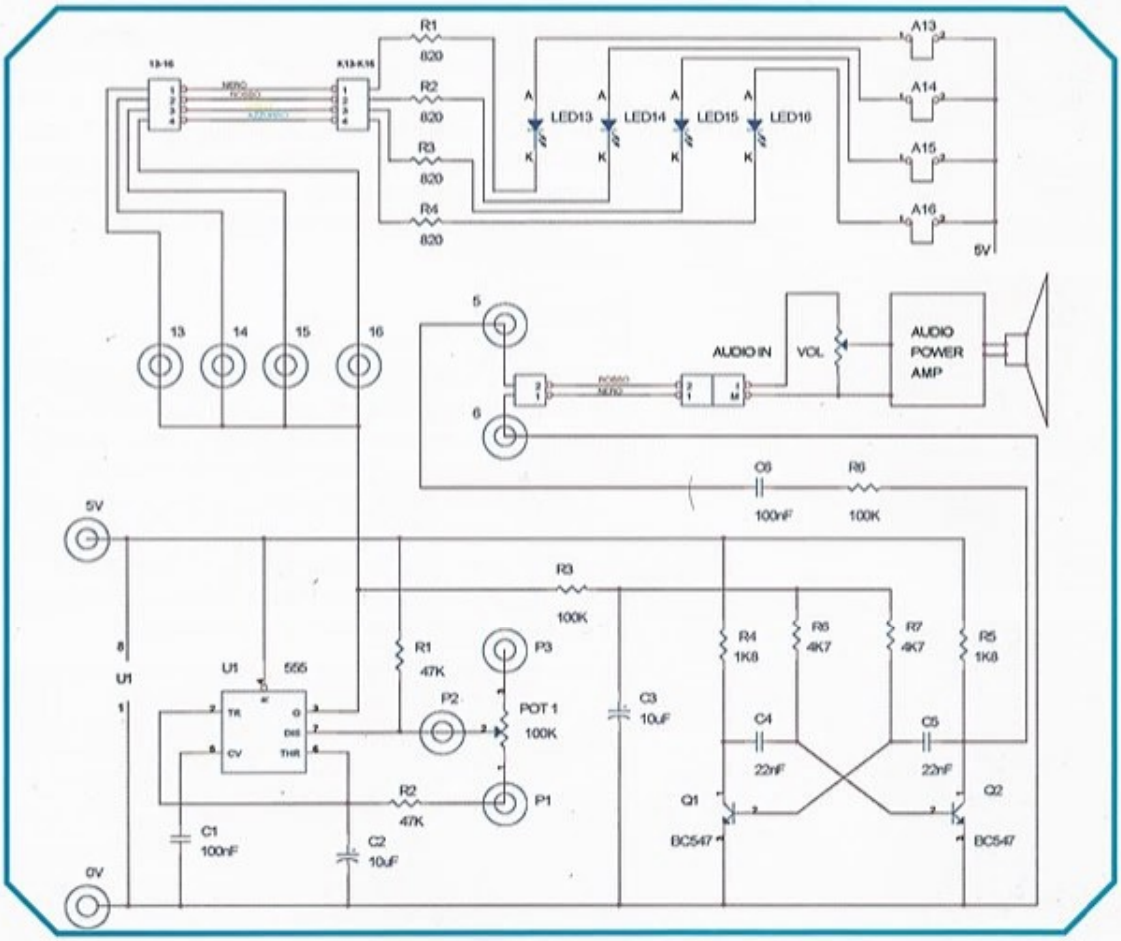
Nello schema possiamo anche vedere un circuito integrato 555 configurato per lavorare come oscillatore astabile, ma a una frequenza molto bassa. La sua uscita si può utilizzare per controllare il VCO e si collega a esso tramite una rete RC. Per fare in modo che la variazione di tensione non sia brusca, la rete RC è formata dalla resistenza R3 e dal condensatore C3.



Distribuzione dei componenti.



Collegamenti della scheda.



Schema dell'oscillatore audio modulato.

Repertorio delle istruzioni nel PicBasic Plus Lite

Vediamo ora le istruzioni e le direttive che formano il repertorio dell'ambiente di programmazione PicBasic Plus Lite. I programmi che realizzeremo con il PicBasic Plus Lite richiederanno la capacità di utilizzo delle espressioni e la conoscenza di queste istruzioni e del loro impiego.

Rappresentazioni numeriche

Il PBPL (PicBasic Plus Lite) riconosce quattro modi di riferimento per i diversi sistemi di rappresentazione:

- Binario: si utilizza il prefisso '%'. Esempio: %01001110.
- Esadecimale: si utilizza il prefisso '$'. Esempio: $1ª.
- Decimale: non si utilizza prefisso.
- Carattere: si definisce utilizzando le virgolette. Esempio: "a".

Per i comandi che accettano una catena di caratteri, questi vengono definiti utilizzando le virgolette, ad esempio: PRINT "Ciao".

Il compilatore discriminerà solamente le maiuscole e le minuscole all'interno della catena di caratteri.

Operatore matematico

Il PicBasic Plus Lite accetta tutti i tipi di operatori aritmetici e inoltre stabilisce un ordine gerarchico al momento di assegnare le priorità fra le operazioni. Come per molti altri ambienti le prime ad essere eseguite sono le espressioni che si trovano all'interno delle parentesi, le operazioni di moltiplicazione e divisione hanno la precedenza su quelle di somma e sottrazione, ecc.

Nella tabella qui sotto potete vedere gli operatori matematici utilizzati nel PBPL e il loro significato.

Quando si esegue una moltiplicazione, se i due operandi sono da 16 bit il risultato può raggiungere i 32 bit. Quando utilizziamo '*' otteniamo come risultato solamente i 16 bit meno significativi della moltiplicazione, perdendo i bit più significativi nel caso fossero presenti.

Se utilizziamo '**' il risultato dell'operazione sarà composto dai 16 bit più significativi e se utilizziamo '*/' il risultato dell'operazione saranno i 16 bit situati al centro, quindi non utilizzeremo il byte più significativo né quello meno significativo.

Istruzioni e direttive

Il repertorio delle istruzioni dell'ambiente PBPL è composto da 74 istruzioni. Durante l'apprendimento della programmazione in Assembler abbiamo studiato le 35 istruzioni con le quali abbiamo lavorato e che erano di basso livello, questo significa che con esse possiamo eseguire solamente un'azione alla volta.

Nel linguaggio di alto livello oltre a poter contare su un maggior numero di istruzioni, esse sono più complesse, cosa che a volte permette di eseguire più azioni con una sola istruzione.

Prendiamo come esempio l'istruzione DELAYMS. Questa istruzione genera un ritardo o temporizzazione di tanti millisecondi quanti ne vengono dichiarati all'utilizzo dell'istruzione stessa. Per fare una temporizzazione in As-

```
PORTA = % 01101010      ; Scrive un valore sulla porta A
TRISB = % 00000000      ; Configura la porta B come uscita
TRISC = % FF            ; Configura la porta C come ingresso
Contatore = 10          ; Inizializziamo la variabile
Testo = "a"             ; Assegniamo un valore alla variabile
```

Esempio di rappresentazioni numeriche.

REPERTORIO DELLE ISTRUZIONI

ADIM	Legge il valore del convertitore A/D
ASM-ENDASM	Inserisce una sezione di codice in assembler
BRANCH	Salto
BRANCH	Salto lungo
BUSIN	Legge i byte del dispositivo I2C
BUSOUT	Scrive i byte sul dispositivo I2C
CALL	Chiamata a una subroutine di un linguaggio assembler
CDATA	Definisce il contenuto iniziale della memoria
CLS	Cancella il display LCD
CONFIG	Configurazione
COUNTER	Conta il numero di impulsi su un pin
CREAD	Legge una parola della memoria di codice
CURSUR	Posiziona il cursore sul display LCD
CWRITE	Scrive una parola nella memoria di codice
DATA	Definisce il contenuto iniziale della memoria
DECLARE	Regola i parametri delle subroutine o librerie
DELAYMS	Ritardo (1 ms di risoluzione)
DELAYUS	Ritardo (2 µs di risoluzione)
DEVICE	Sceglie il tipo di PIC che utilizzerà il compilatore
DIG	Restituisce il valore di un digit decimale
DIM	Crea una variabile
EDATA	Definisce i contenuti iniziali della memoria EEPROM
END	Ferma l'esecuzione
EREAD	Legge un byte o una word nella memoria EEPROM
EWRITE	Scrive un byte nella memoria EEPROM
FOR... TO... NEXT... STEP	Esegue ripetutamente le istruzioni
GOSUB... RETURN	Chiamata a una subroutine in Basic con l'etichetta specificata
GOTO	Continua l'esecuzione all'etichetta specificata
HIGH (ó SET)	Passa a livello alto un pin, porta o registro
IF... THEN... ELSE... ENDIF	Esegue istruzioni in funzione della condizione
INCLUDE	Include un file in Basic nel codice
INKEY	Cerca un tasto
INPUT	Definisce un pin come ingresso
[LET]	Assegna il risultato di una espressione a una variabile
LCDREAD	Legge un byte del display LCD
LCDWRITE	Scrive un byte del display LCD
LOOKDOWN	Cerca un valore in una tabella di costante

REPERTORIO DELLE ISTRUZIONI

LOOKDOWNL	Cerca un valore in una tabella di costanti o variabili
LOOKUP	Cerca una costante in una tabella
LOOKUPL	Cerca una costante o una variabile di una tabella
LOW (o CLEAR)	Imposta a livello basso un pin, porta o registro
ON... INTERRUPT	Esegue una subroutine quando avviene un interrupt hardware
OUTPUT	Definisce un pin come uscita
ORG	Definisce l'origine del programma
PEEK	Legge un byte di un registro
PIXEL	Legge un singolo pixel del display LCD
PLOT	Definisce un singolo pixel del display LCD
POKE	Scrive un byte su un registro
POT	Legge un valore analogico variabile sul pin specificato
PRINT	Visualizza un carattere sul display LCD
PULSIN	Misura l'ampiezza di banda dell'impulso ricevuto su un pin
PULSOUT	Genera un impulso su un pin
PWM	Uscita di un impulso modulato tramite un pin
RANDOM	Genera un numero casuale
RCIN	Misura l'ampiezza di un impulso di un pin
READ	Legge un byte o una parola della memoria
REM	Aggiunge un commento al codice
REPEAT... UNTIL	Ripete le istruzioni fino a quando la condizione è compiuta
RESTORE	Sposta il puntatore nella posizione specificata per leggere
RETURN	Continua l'esecuzione dall'istruzione successiva alla chiamata alla subroutine
RSIN	Ingresso seriale asincrono tramite un pin fisso
RSOUT	Uscita seriale asincrona tramite un pin fisso
SERVO	Controllo di un servomotore
SET... OSCCAL	Calibra l'oscillatore
SHIN	Ingresso seriale sincrono
SHOUT	Uscita seriale sincrona
SNOOZE	Spegne il processore per un breve periodo
SLEEP	Spegne il processore per un periodo
SOUND	Genera un tono sul pin specificato
STOP	Ferma l'esecuzione del programma
SWAP	Interscambia i valori di due variabili
SYMBOL	Crea un alias o un equivalente di una costante, pin, porta o registro
UNPLOT	Cancella un singolo pixel sul display LCD
WHILE... WEND	Esegue le istruzioni quando la condizione è verificata

ISTRUZIONI SPECIALI		
BK label	Salta	GOTO label
BC label	Salta se c'è CARRY	BTFSC 3,0 GOTO label
BDC label	Salta se DIGIT CARRY=1	BTFSC 3,1 GOTO label
BNC label	Salta se non c'è CARRY	BTFSS 3,0 GOTO label
BNDC label	Salta se DIGIT CARRY=0	BTFSS 3,1 GOTO label
BZ label	Salta se è 0	BTFSC 3,2 GOTO label
BNZ label	Salta se è diverso da 0	BTFSS 3,2 GOTO label
CLRC	Azzera CARRY	BCF 3,0
CLRDC	Azzera DIGIT CARRY	BCF 3,1
CLRZ	Resetta ZERO	BCF 3,2
MOVFW file	Sposta file su WREG	MOVF file,W
SETC	Imposta a 1 il CARRY	BSF 3,0
SETDC	Imposta a 1 il DIGIT CARRY	BSF 3,1
SETZ	Imposta a 1 il flag ZERO	BSF 3,2
SKPC	Salta alla successiva se CARRY=1	BTFSS 3,0
SKPNC	Salta alla successiva se CARRY=0	BTFSC 3,0
SKPDC	Salta alla successiva se DIGIT CARRY=1	BTFSS 3,1
SKPNDC	Salta alla successiva se DIGIT CARRY=0	BTFSC 3,1
SKPZ	Salta alla successiva se ZERO=1	BTFSS 3,2
SKPNZ	Salta alla successiva se ZERO=0	BTFSC 3,2

sembler, invece, dobbiamo utilizzare una subroutine con diverse istruzioni e gestire molto bene i dispositivi del PIC come ad esempio i temporizzatori. Con il semplice inserimento di una linea di programma che riporti: DELAYMS 100, otterremo un ritardo di 100 ms.

Nella tabella delle pagine precedenti è riportato un listato delle 74 istruzioni che gestiremo con il PBPL e la loro descrizione. Se si vuole vedere la descrizione in modo più dettagliato e degli esempi di utilizzo, è necessario ricorrere al manuale che si trova nell'opzione Syntax all'interno del menù di aiuto Help del programma stesso. Questo manuale è in inglese.

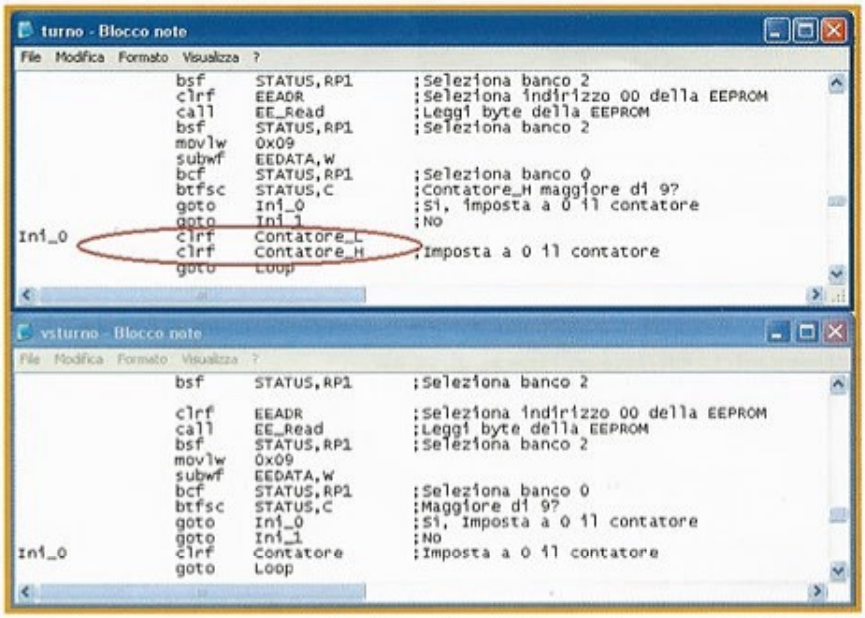
Altre istruzioni

Esistono mnemonici per istruzioni speciali che non sono incluse all'interno della lista delle 74 istruzioni. Queste istruzioni speciali sono subistruzioni di altre più generali, come ad esempio BRANCH, SET o SKIP, che dato il loro elevato utilizzo per le funzioni descritte nella tabella allegata, sono state implementate nel compilatore. Anche il lavoro con la memoria avviene in modo speciale e molto più semplice rispetto all'Assembler, infatti grazie ai nuovi mnemonici non dovremo più fare differenza fra i banchi di memoria dove si trovano i registri.

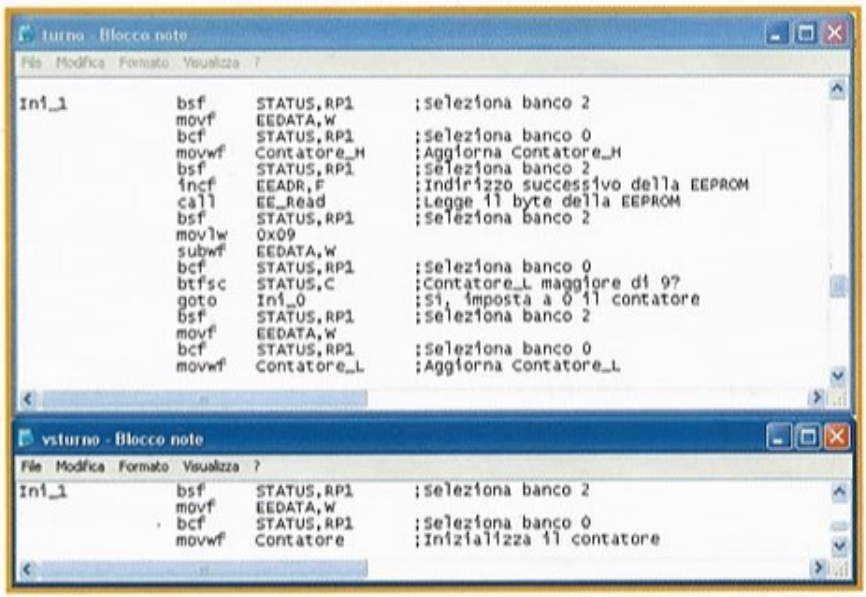
Conclusioni

Abbiamo presentato l'ambiente di programmazione PicBasic Plus Lite, perché la tendenza attuale per quanto riguarda la programmazione dei microcontroller tende alla programmazione di linguaggi con alto livello, principalmente il Basic, benché sia seguito da vicino dal C. I linguaggi di programmazione di alto livello presentano molti vantaggi rispetto all'Assembler, anche se questo rimane il linguaggio per eccellenza nel mondo dei microcontroller.

Se si vuole diventare dei veri esperti nel mondo dei microcontroller è necessario far pratica con questo ambiente e saperlo dominare, sarà indispensabile anche lavorare con il linguaggio C. Su Internet potrete trovare diversi compilatori in C e in Basic per l'Assembler, la maggior parte di essi reperibili nelle versioni demo.



Differenze tra i due programmi nel primo blocco.



Secondo blocco: etichetta "Ini_1":

ne rappresentiamo sull'LCD il messaggio il cui inizio è determinato dal registro di lavoro o accumulatore. Nella figura potete vedere la routine in questione.

Programma principale

Iniziamo il programma principale configurando i dispositivi con cui lavoreremo. La porta A deve essere configurata in parte come ingresso e in parte come uscita, dato che ora abbiamo bisogno dei segnali di E, RS e R/W per l'LCD, e continua a essere necessario un ingresso RA4. Carichiamo quindi '00011000' sul registro TRISA e '00000110' sul registro ADCON1. Configuriamo la porta B come porta di uscita dei dati e il registro OPTION_REG con il predivisore per il Timer 0. Fatto questo, inizializzeremo il display LCD chiamando la routine "LCD_INI" e lo configureremo mandando il valore '00001100' alla routine LCD_REG.

Se manteniamo l'ordine definito nell'organigramma, il passo successivo consiste nel visualizzare il messaggio con il testo fisso "Turno:". A questo scopo caricheremo il messaggio 0 (Mess_0) sull'accumulatore e richiameremo la subroutine di visualizzazione sul display LCD.

A partire da questo punto possiamo utilizzare il codice sviluppato per l'applicazione con il display a 7 segmenti. Osservando la figura vediamo che l'unica cosa cambiata in questo blocco di istruzioni è che invece di resettare un'unica variabile "Contatore" nel nuovo codice dobbiamo resettarne due, Contatore_H (decine) e Contatore_L (unità).

Il secondo blocco di programma, invece, si differenzia abbastanza dal programma che utilizziamo come base, infatti dobbiamo includere la verifica che il dato successivo contenuto nella memoria, ovvero quello delle unità, non sia maggiore di nove. Se lo è impostiamo a zero il contatore, in caso contrario aggiorniamo la variabile con il valore contenuto nella memoria.

Abbiamo quindi letto il valore contenuto nella memoria e lo abbiamo aggiornato nella nostra variabile, il passo successivo consisterà nel visualizzarlo.

Continuate a risolvere l'applicazione e poi verificate che la soluzione che avete adottato coincida con quella presentata.

Esercizio: "Il vostro turno" con il display LCD, il programma

In quest'ultimo esercizio adatteremo il programma "Il Vostro Turno" che abbiamo realizzato per il display a sette segmenti, per visualizzarlo sul modulo LCD. Un programmatore non sviluppa solamente nuove applicazioni ma deve saper adattare anche applicazioni già sviluppate a nuove specifiche o a nuovi miglioramenti tecnologici.

Enunciato

Si tratta di emulare il funzionamento delle macchine tipo "IL VOSTRO TURNO" comuni in molti negozi. Sul display LCD verrà visualizzato il numero del turno attuale che sarà formato da due digit. Questo verrà incrementato a ogni impulso applicato all'ingresso RA4. Nella memoria EEPROM del PIC16F870 viene scritto l'ultimo numero visualizzato, in modo che in caso di mancanza di tensione, ad esempio, si riprenda il conteggio da quest'ultimo numero.

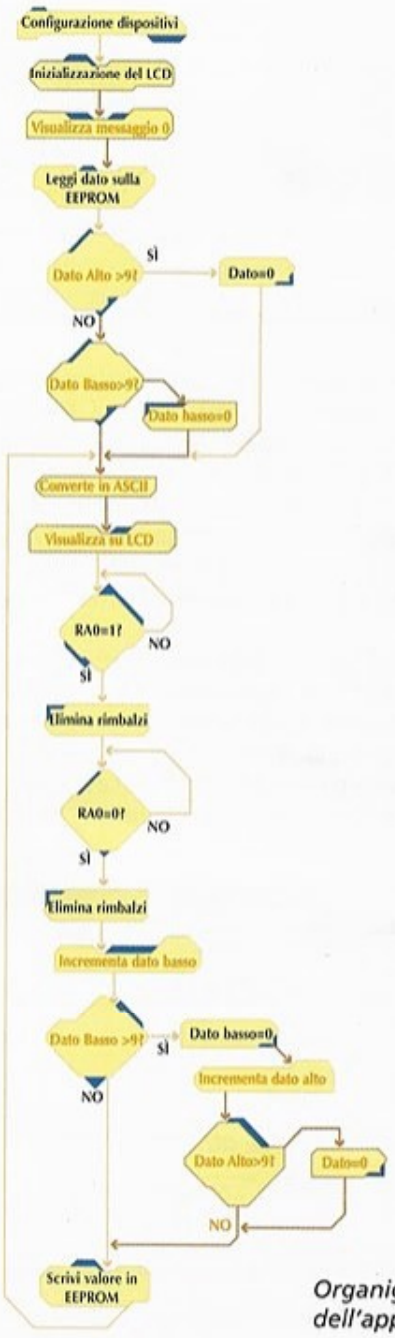
Supponiamo di partire da un sistema che viene utilizzato per la prima volta, quindi si visualizza lo 00.

Come potete vedere l'esercizio che stiamo pianificando non è altro che la macchina "Il Vostro Turno", ma in versione migliorata. Si tratta di un esercizio molto completo dato che lavoreremo con la memoria EEPROM dei dati e il display LCD.

Organigramma

Per risolvere questa applicazione possiamo fare due ipotesi di lavoro, in una di esse svilupperemo il programma dall'inizio, partendo da

Macchina "Il vostro turno".



Organigramma dell'applicazione.

turno - Blocco note

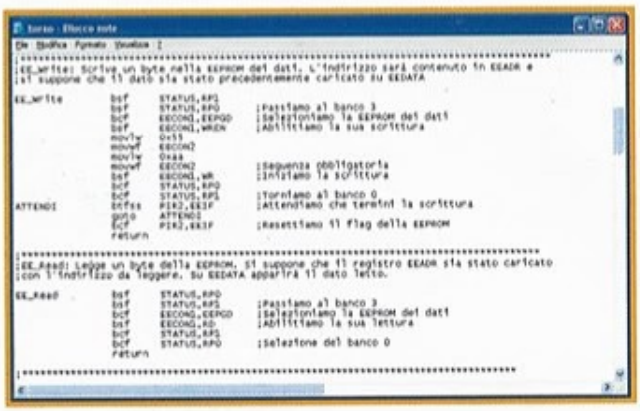
File Modifica Formato Visualizza ?

```
;       ESERCIZIO: Il vostro turno su LCD
;       ---------------------------------
;
;La memoria EEPROM dei dati e il display LCD. La macchina "IL VOSTRO TURNO", versione migliorata.
;
;Si tratta di emulare il funzionamento delle macchine tipo "IL VOSTRO TURNO" comuni in molti
;negozi. Sul display LCD si visualizzerà il numero del turno attuale (2 cifre). Questo numero si
;incrementa ad ogni impulso su RA4. Nella memoria EEPROM del PIC16F870 si scrive l'ultimo
;numero visualizzato, in questo modo, in caso di mancanza di tensione (ad esempio), si ricomincia
;il conteggio dall'ultimo numero.
;
;Si parte dal sistema che inizia per la prima volta, si visualizza 00

                List    p=16F870        ;Tipo di processore
                include "P16F870.INC"   ;Definizioni dei registri interni

Lcd_var         equ     0x20            ;Variabili (2) della routine di gestione del LCD
Contatore_L     equ     0x21            ;Variabile per il contatore parte bassa
Contatore_H     equ     0x22            ;Variabile per il contatore parte alta
Temporale_1     equ     0x23            ;Variabile Temporale
Temporale_2     equ     0x24            ;Variabile Temporale

                org     0x00            ;Vector di Reset
                goto    Inizio
                org     0x05            ;Salva vector di interrupt

                include "LCD_Cxx.inc"   ;Include la routine di gestione del LCD

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

Intestazione del programma.

zero e nell'altra utilizzeremo il programma che abbiamo creato per i display a 7 segmenti. In alcuni casi, adattare il programma è più complicato che rifarlo da zero, ma in questo caso dobbiamo unicamente inserire il display LCD, quindi scegliamo l'opzione di utilizzare il programma "turno.asm" (l'esercizio è: La memoria EEPROM dei dati. La macchina "IL VOSTRO TURNO").

Partendo dall'organigramma che abbiamo creato per questo programma, lo dobbiamo adattare alle nuove specifiche. Ora il contatore sarà formato da due variabili, una per ogni cifra, quindi per contare bene dovremo verificare che entrambe arrivino a nove per iniziare a contare partendo da zero. Dobbiamo anche inserire la programmazione per il display LCD, cosa che implica variazioni generali nel programma. Le differenze fra questo organigramma e quello dell'applicazione per i display a 7 segmenti sono evidenziate in rosso.



Routine di gestione della EEPROM.

Codice

Utilizzeremo il codice del programma "turno.asm" (l'esercizio è: La memoria EEPROM dei dati. La macchina "IL VOSTRO TURNO") lo adatteremo per visualizzare l'uscita sul modulo LCD. Come per l'organigramma l'adattamento del codice richiederà alcune modifiche, ma partendo dalla radice del programma queste saranno piuttosto semplici.

Impostazioni e variabili

La prima differenza o modifica da eseguire è nell'intestazione del programma. Dato che si tratta di un programma con nuove specifiche le dovremo dettagliare mediante dei commenti. Inoltre il nuovo programma ha delle variabili in più. La variabile contatore ora è doppia, ne serve una per contenere le decine e l'altra per le unità, quindi creeremo "Contatore_H" e "Contatore_L". Dichiareremo anche la variabile "Lcd_var", che sarà formata da due variabili all'interno della libreria di gestione dell'LCD, e

turno - Blocco note

File Modifica Formato Visualizza ?

```
;*******************************************************************************
;In base al valore contenuto nel registro W, si ottiene il carattere ASCII da visua-
;lizzare sul display LCD

Tabella_Messaggi:       movwf   PCL     ;Spostamento sulla tabella

;*******************************************************************************
;La direttiva dt genera tante istruzioni retlw quanti byte o caratteri contiene
;chiusi fra "

Mess_0          equ     $               ;Mess_0 punta al primo carattere
                dt      "TURNO:",0x00

;*******************************************************************************
;Delay_20_ms: Questa routine di temporizzazione ha come obiettivo l'eliminazione
;dell'effetto rimbalzo dei dispositivi elettromeccanici. Realizza un ritardo di 20 ms.
;Se il PIC lavora ad una frequenza di 4 MHz, il TMR0 evolve ogni µs. Se vogliamo tempo-
;rizzare 20000 µs (20 ms) con un prescaler da 128, il TMR0 dovrà contare 156 eventi
;(156 * 128). Il valore 156 equivale a 9c hex. e dato che il TMR0 è ascendente lo
;dovremo caricare con il suo complemento a 1 (63 hex.).

Delay_20_ms:    bcf     INTCON,T0IF     ;Azzera il flag di overflow
                movlw   0x63            ;Complemento hex. di 156
                movwf   TMR0            ;Carica il TMR0
Delay_20_ms_1   clrwdt                  ;Aggiorna il WDT
                btfss   INTCON,T0IF     ;Overflow del TMR0?
                goto    Delay_20_ms_1   ;Non ancora
                bcf     INTCON,T0IF     ;Ora sì, azzera il flag
                return

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

Altre routine associate al programma.

due variabili temporali "Temporale_1" e "Temporale_2" che utilizzeremo per la rappresentazione dei dati sul modulo LCD.

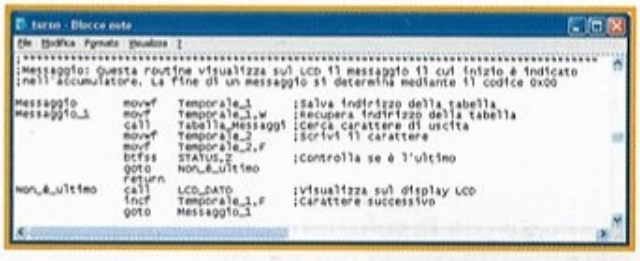
Routine associate al programma

Dopo l'intestazione del programma, includiamo le routine con le quali vogliamo lavorare. Avremo bisogno di quelle di scrittura e lettura della memoria EEPROM, che non variano in nulla rispetto a quelle utilizzate in altri programmi. Avremo altresì bisogno di una routine di temporizzazione fissa per eliminare i "rimbalzi" che possono essere generati da un azionamento meccanico. Come per le routine di gestione della memoria EEPROM, anche questa routine è la stessa già utilizzata in altre occasioni.

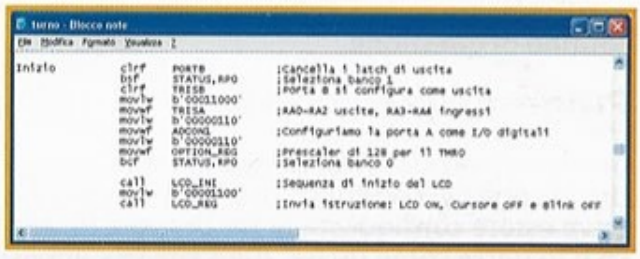
Come si può vedere dalle figure, oltre alle routine per eliminare i rimbalzi, sono state dichiarate due nuove routine. La prima è molto simile alla tabella del programma originale in cui ricerchiamo la corrispondenza del dato al codice a 7 segmenti, ma in questo caso cerchiamo la corrispondenza al codice ASCII.

La routine successiva è una tabella che contiene i caratteri fissi del messaggio da visualizzare. Negli esercizi precedenti i caratteri che formavano un messaggio erano separati e inviati alla visualizzazione uno a uno. In questa routine utilizziamo la direttiva "dt" che genera tanti return quanti caratteri si trovano tra le virgolette (""). In altre parole, questa routine è uguale a quella che contiene tutti i caratteri nella tabella, ma semplificata mediante un'unica direttiva.

Infine avremo bisogno di una nuova routine che non è nient'altro che quella di visualizzazione sul display LCD. Questa routine è stata utilizzata nei programmi precedenti (ad esempio "addio.asm"). Tramite questa routi-



Routine di visualizzazione sull'LCD.



Configurazione dei dispositivi.