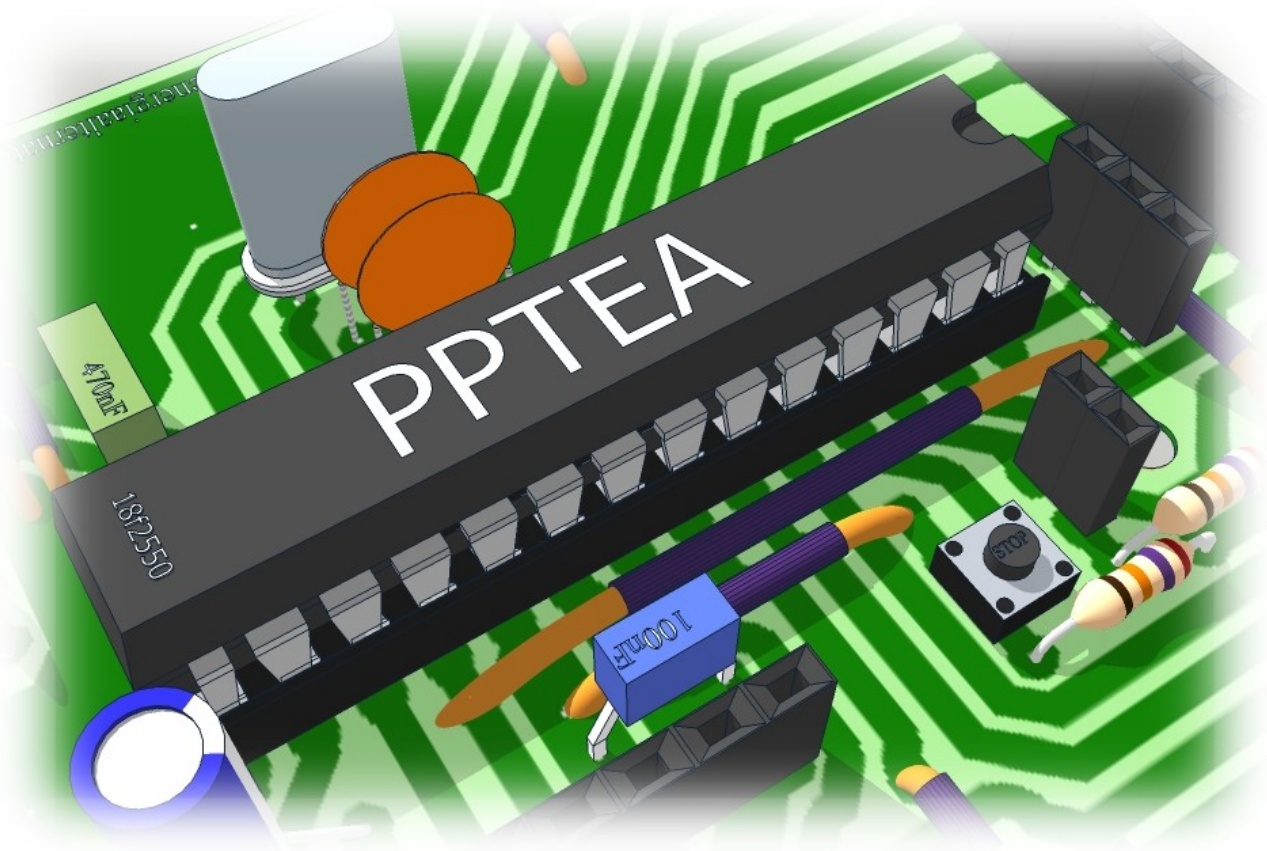
PPTEA

MANUALE DI RIFERIMENTO

Versione Advanced 4.2.6

Autore: ***NonSoloBolleDiAcqua***

Un ringraziamento a :Libero51, Mixtrb , PinoTux, Jumpy75, MarkoZakka, Ronwal e Alessio287



Sito ufficiale
http://pptea.altervista.org

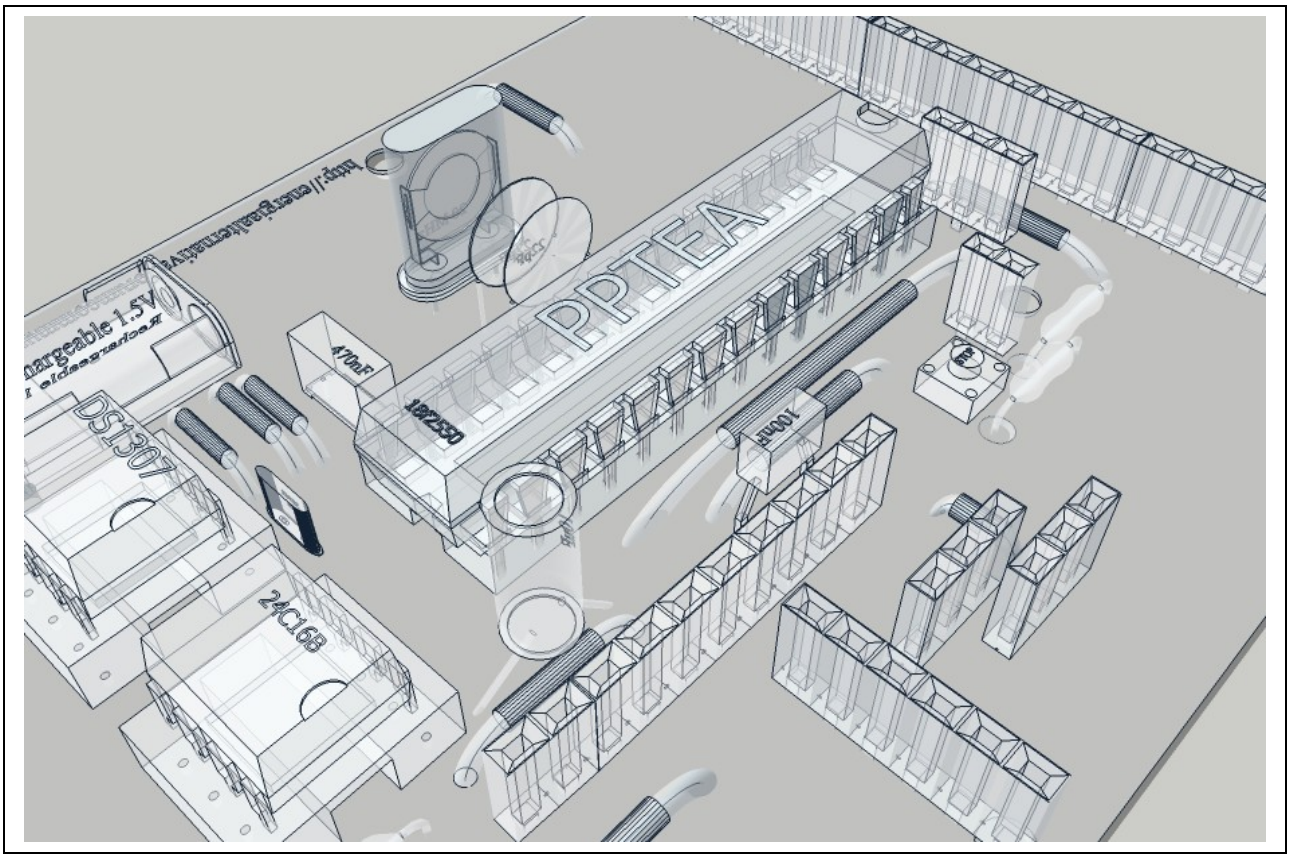
Sito PPTEA
http://www.pptea.it

Storia ed evoluzioni del progetto
http://www.energialternativa.org/Public/NewForum/Sezione.php?8107054

SCOPO DEL PROGETTO

Lo scopo del PPTEA (Processore Per Tutti Energia Alternativa) è quello di aiutare le persone a sviluppare un progetto elettronico a microprocessore con semplicità. L'utilizzo del progetto è completamente gratuito ma ***non può essere utilizzato per scopi commerciali, plagiato o alterato il suo contenuto***. Il progetto nella sua interezza é coperto dal diritto d'autore e ogni persona può partecipare in modo attivo (http://creativecommons.org/licenses/by-nc-nd/3.0/deed.it).

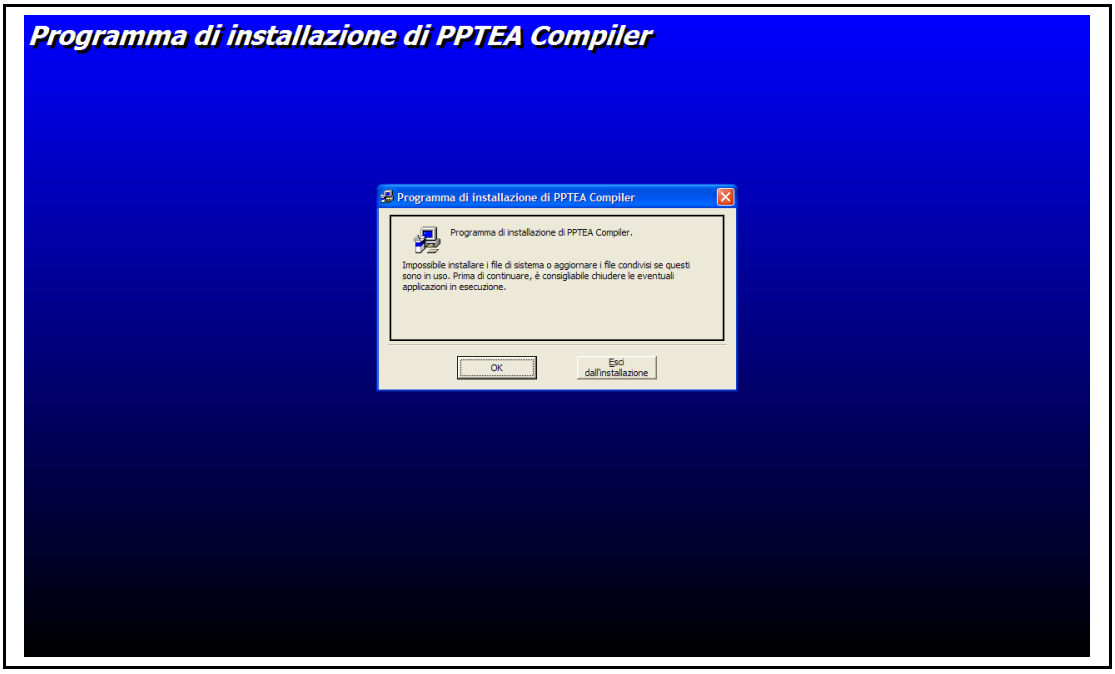
L'obiettivo è stato realizzare un sistema con microprocessore a basso costo; la scelta è caduta sul PIC 18F2550 (4,81 Euro su RS) che corredato da un esiguo numero di componenti permetterà di realizzare i progetti. Per lo sviluppo del Software è stato realizzato un compilatore (gratuito ed utilizzabile su piattaforma Windows) che permetterà la programmazione del linguaggio EABasic (EnergiaAlternativaBasic). La programmazione del linguaggio EABasic è molto semplice e non necessita ne di una conoscenza dell'architettura del processore ne di una particolare conoscenza di programmazione. Il firmware (unico) presente nel processore è il cuore del sistema del PPTEA e permette di interfacciarsi con il compilatore ed eseguire i programmi trasferiti nel PIC senza alcuna riprogrammazione del CHIP. Il consumo del processore è molto ridotto (milliWatt) e nel forum di Energia Alternativa sono presenti diversi esempi su come utilizzare le diverse periferiche (I/O, USB, Wireless, Web Server, PWM, Contatori Esterni, RealTimeClock, Espansione di Memoria, Display LCD, Suoni, RS232, Temperatura, Pressione, etc…)



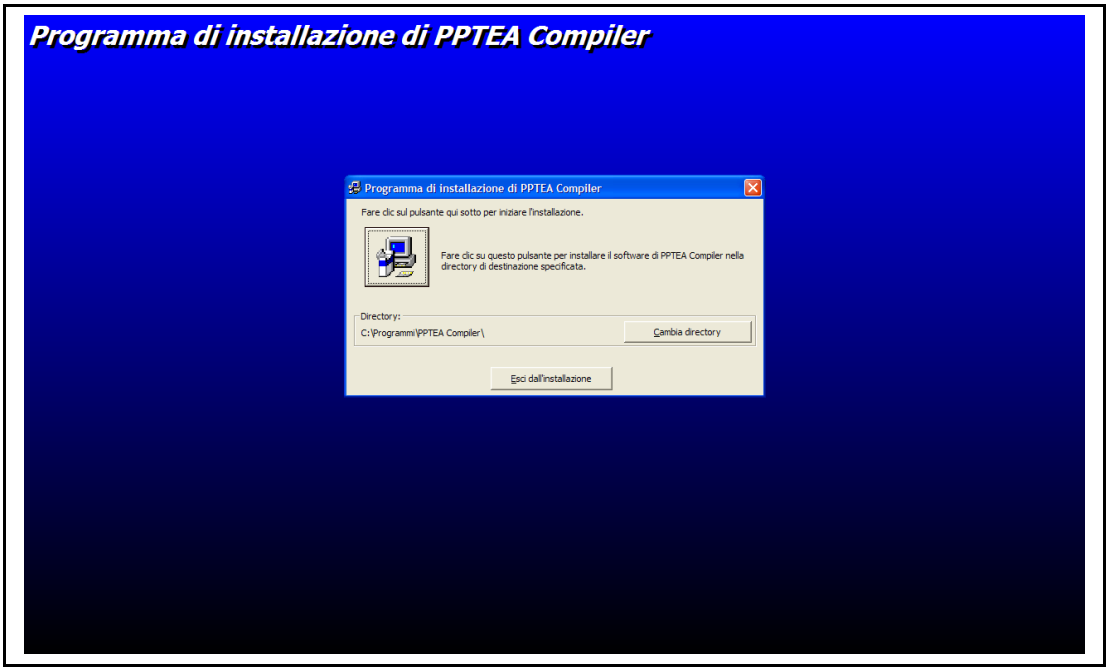
INSTALLAZIONE DEL PPTEA COMPILER SUL PC

Il PPTEA Compiler può essere istallato sul PC mediante Setup o mediante la copia dei file sul Pc stesso.
Vediamo l'***istallazione mediate Setup***. Se abbiamo il PPTEACompiler già istallato occorre procedere alla disintallazione (vedi paragrafo successivo).
Decomprimere i file di consegna in una cartella e lanciare il Setup.exe:

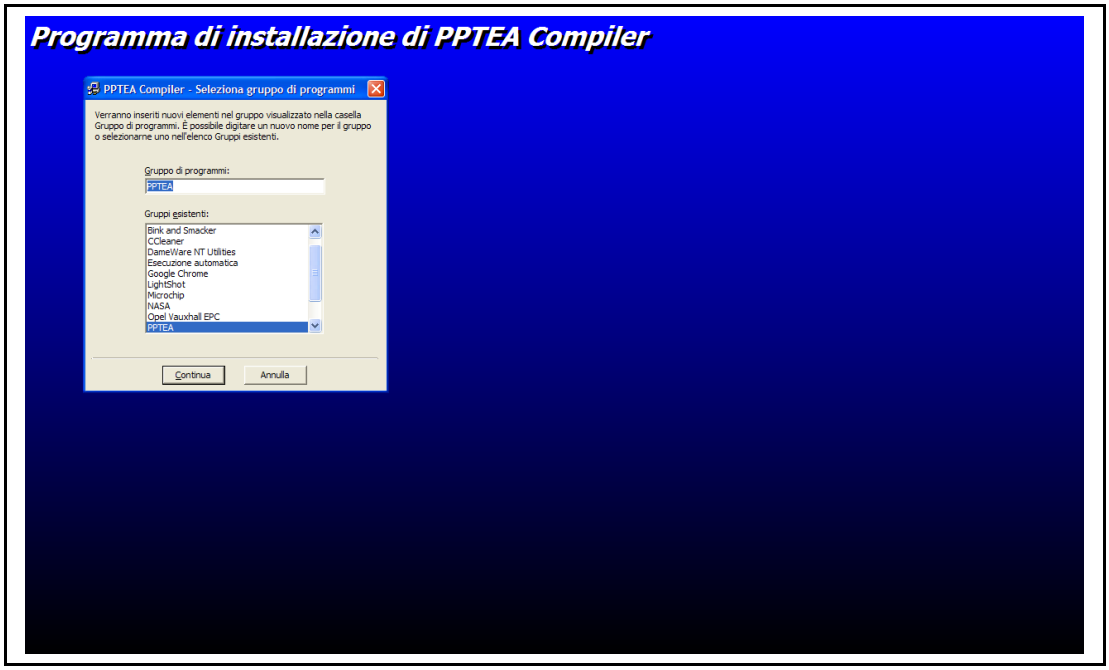
Apparirà la schermata:



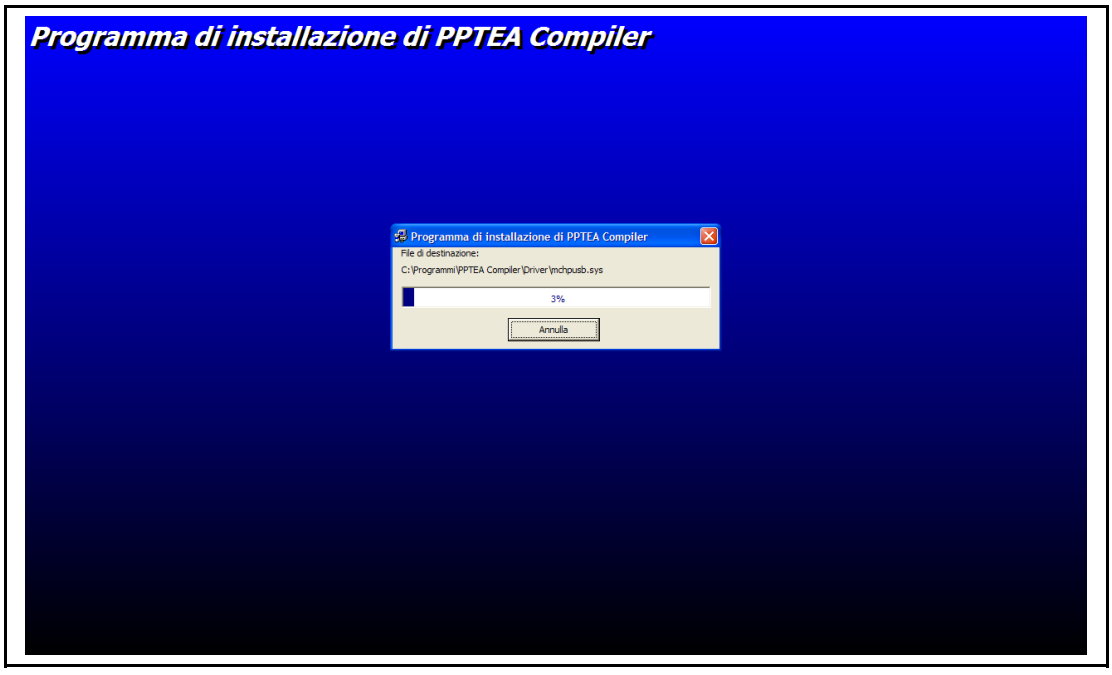
Premere il pulsante ok. Apparirà la schermata:



Se necessario modificare il direttorio e successivamente premere il pulsante con il computerino.

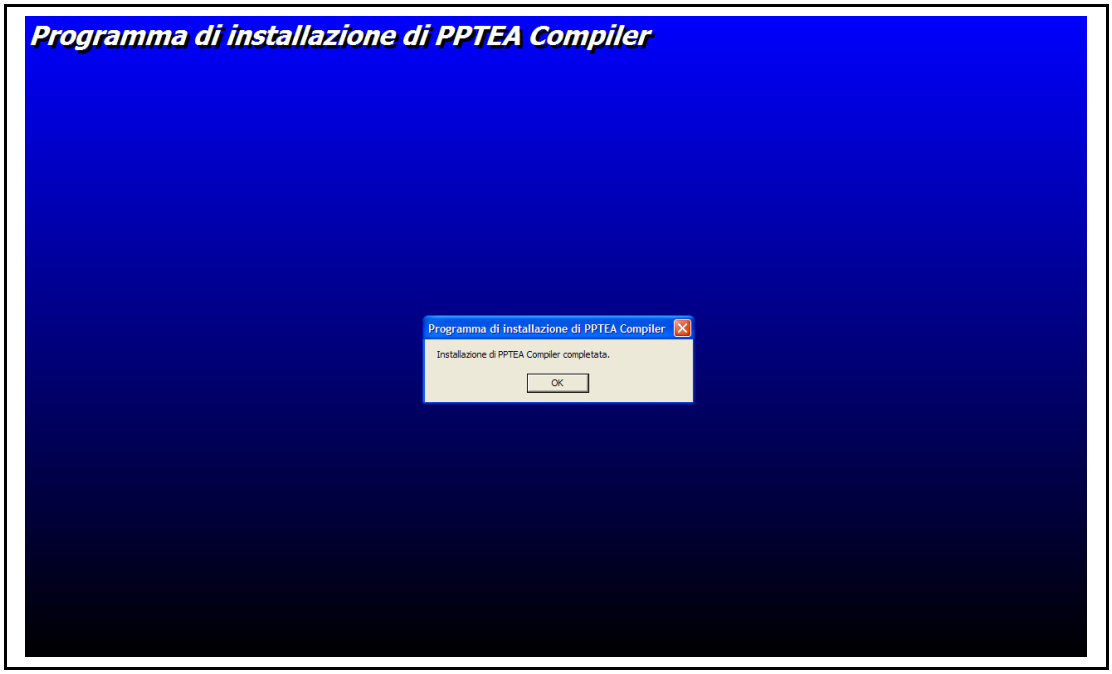


Premere il pulsante ***Continua***



Verranno copiati i file nella destinazione.

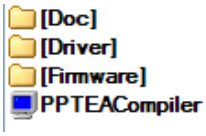
Terminata la copia dei file verrà presentata la finestra di Istallazione Completata.



Premere ok.

Se l'istallazione è avvenuta con successo, sul Menù Dei Programmi di Windows (Start->Tutti i programmi->) sarà presente la voce PPEA Compiler:

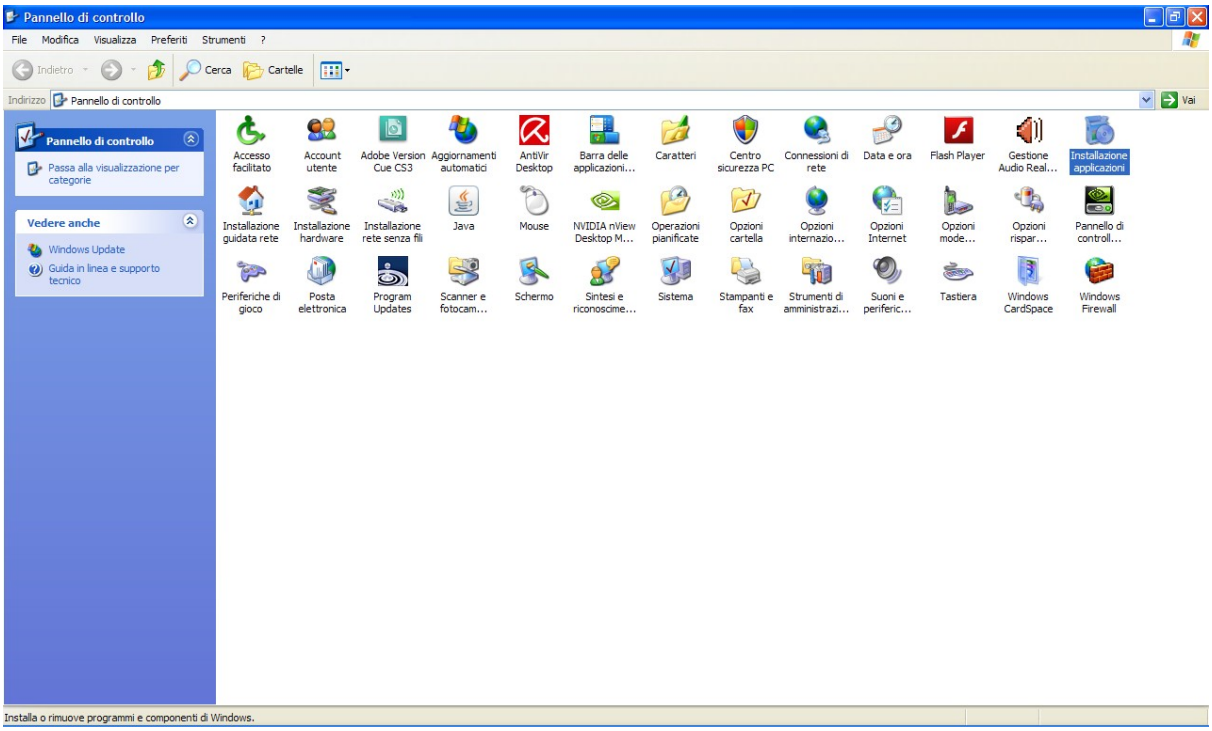
mentre la cartella PPTEA Compiler conterrà tre sottocartelle :

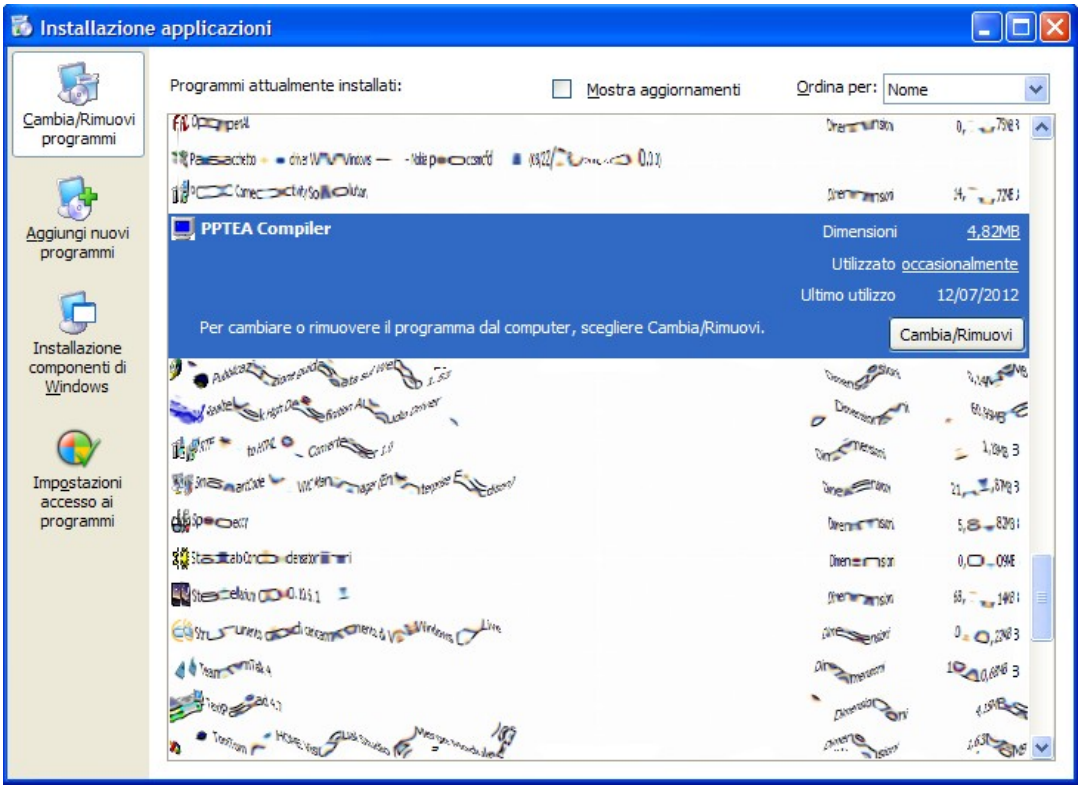


- ***Doc***
  Contiene i docuementi
- ***Driver***
  Contiene i driver di Windows Xp e Windows 7
- ***Firmware***
  Contiene il firmware del PPTEA che andrà caricato nel PIC 18F2550

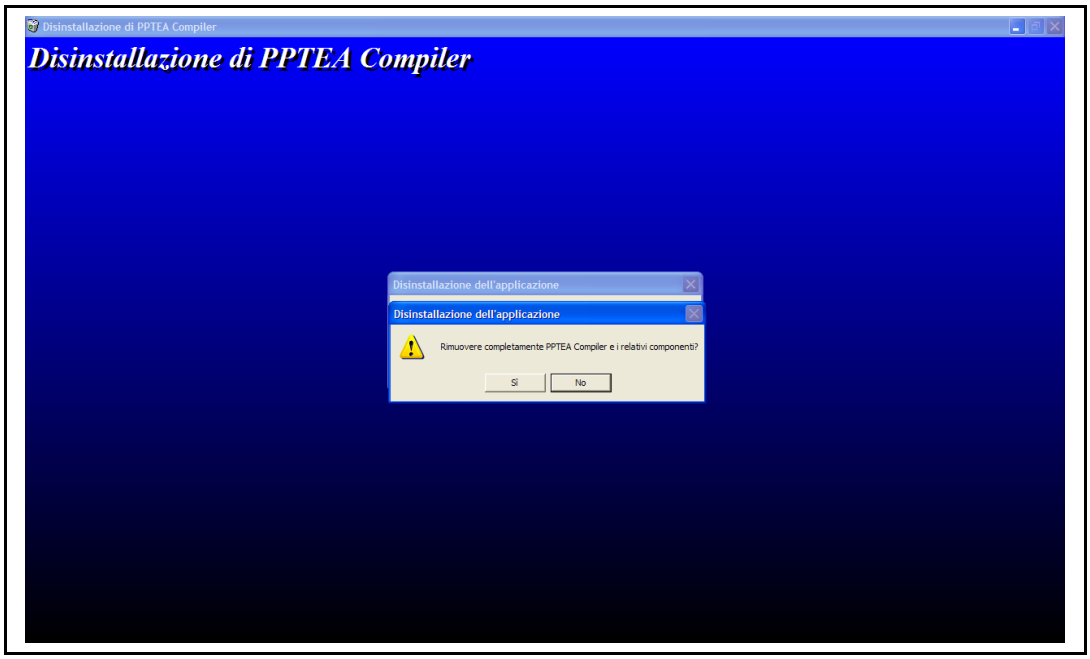
DISINSTALLAZIONE DEL PPTEA COMPILER SUL PC

Andare nel pannello di controllo di windows ed effettuare un doppio click sull'icona ***Installazione applicazioni***



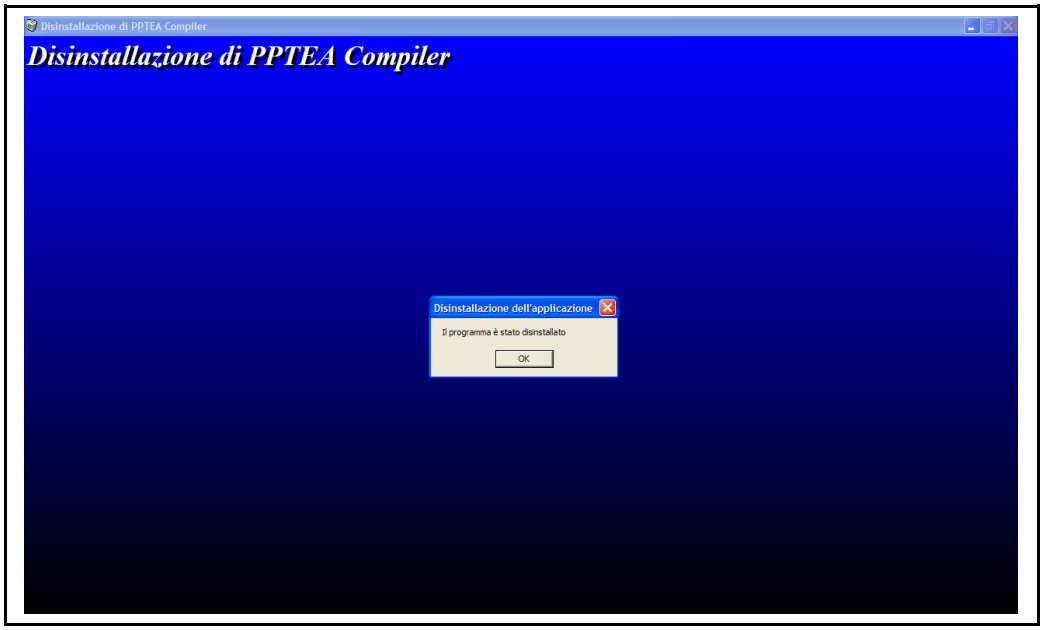


Trovare l'applicazione del PPTEA Compiler e premere il pulsante ***Cambia/Rimuovi.*** Apparirà la finestra:



premere il pulsante ***Si.***

Attendere la comparsa della finestra:

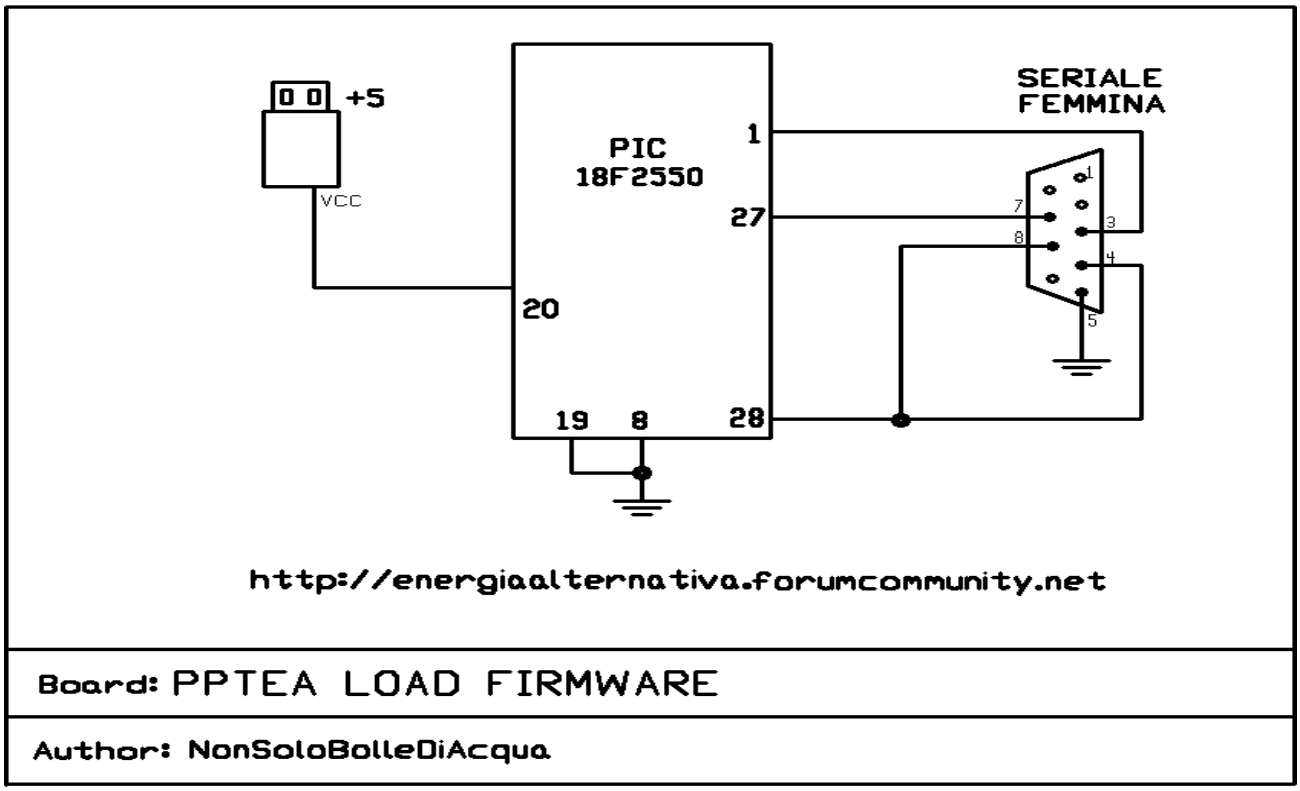


Premendo il pulsante OK il ***PPTEA Compiler*** sarà completamente disinstallato.

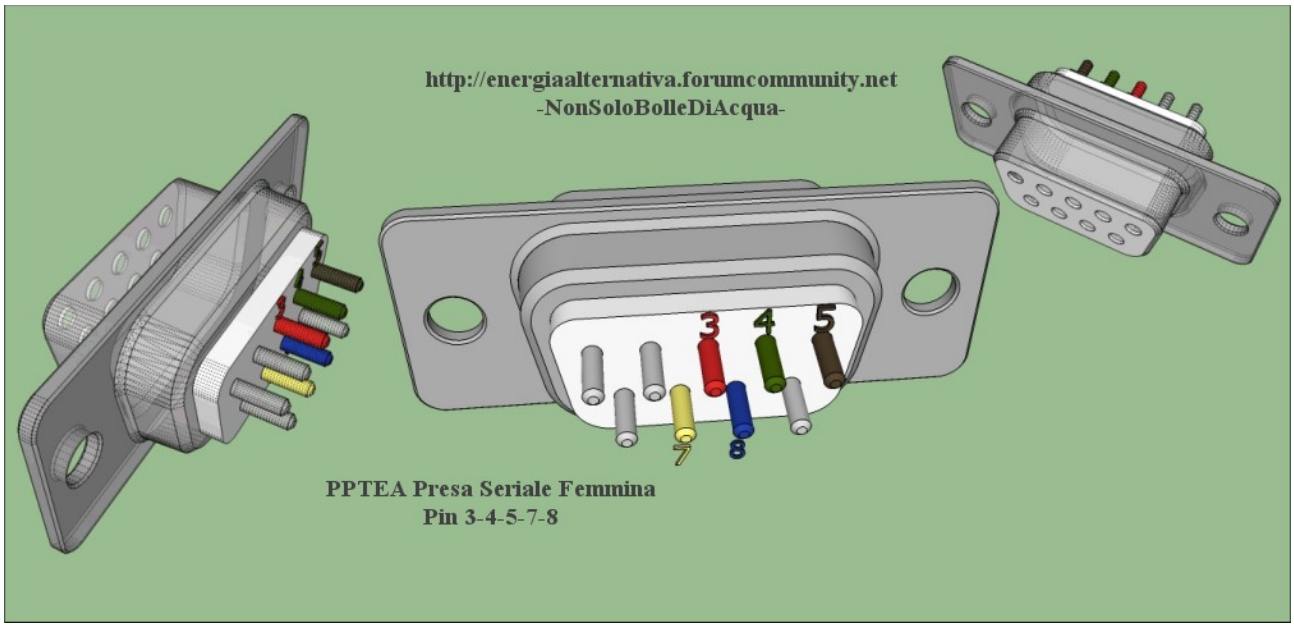
CARICARE IL FIRMWARE DEL PPTEA

La prima operazione da fare,che va effettuata una sola volta, è caricare nel pic 18f2550 il firmware del PPTEA.
Ci sono diversi hardware e software che permettono tale operazione. Proponiamo un circuito fai da te, ridotto ai minimi termini, che necessita di una porta seriale (*) e una porta USB (utile esclusivamente per la +5V) ed utilizza il SW gratuito WINPIC 800. Se non si dispone di una porta USB è sufficiente dare una +5V con un alimentatore collegando la massa in comune.
*) Non utilizzare i convertitori usb/seriale perchè generalmente non funzionano.

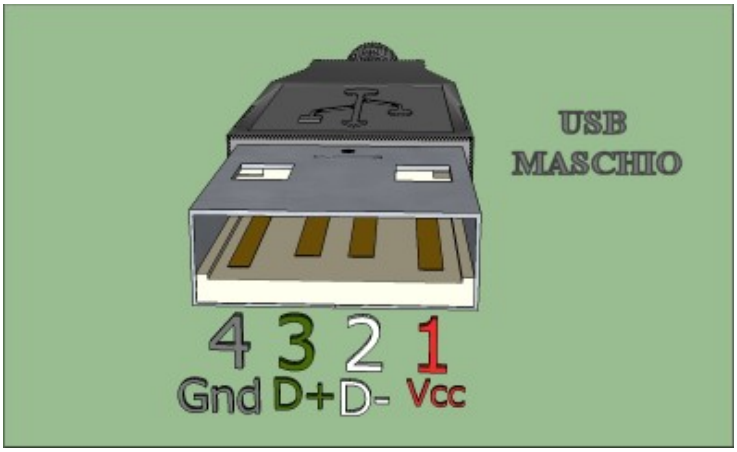
Questo è lo schema:



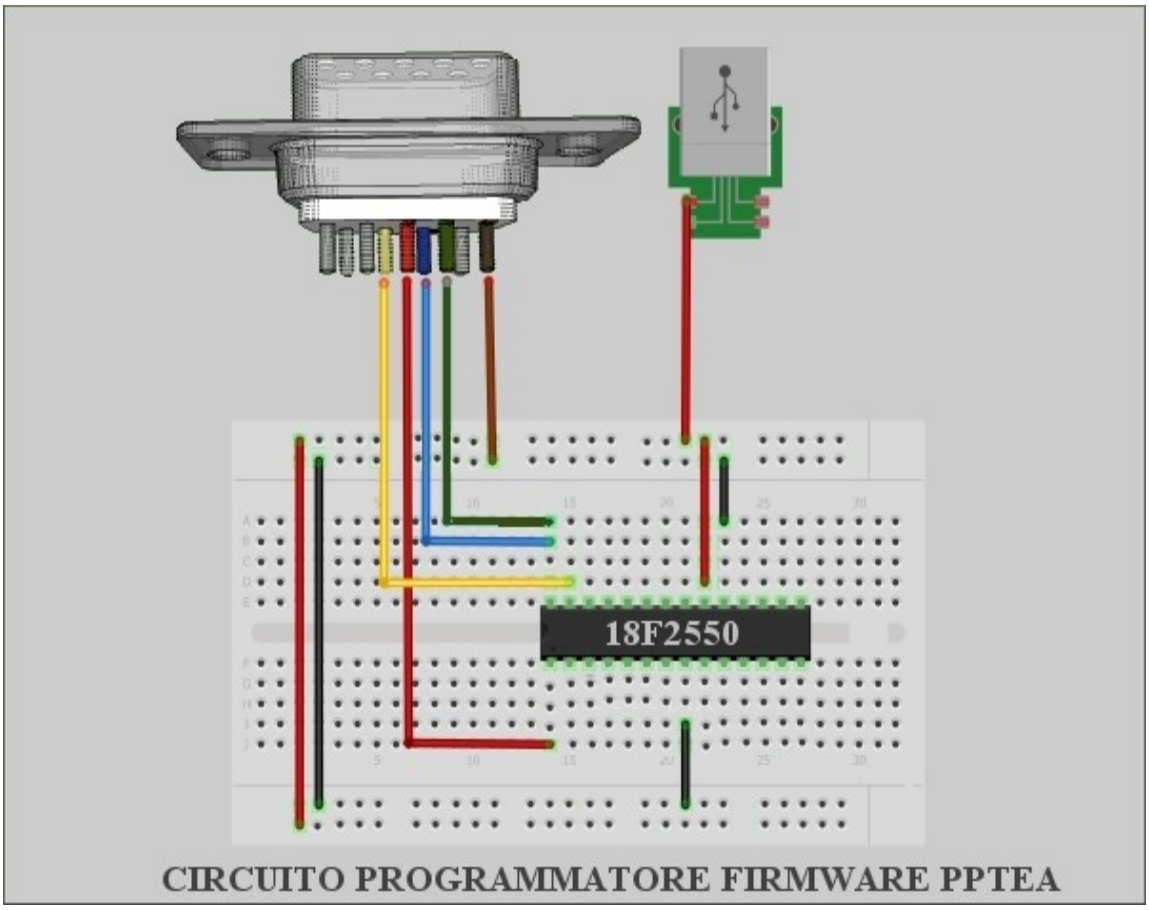
Vista pin presa seriale femmina:



Vista pin presa usb maschio:



Breadboard del circuito per caricare il firmware del PPTEA:



CIRCUITO PROGRAMMATORE FIRMWARE PPTEA

Per caricare il firmware,realizzato il circuito dello schema in alto, occorre seguire la seguente procedura:

1.Istallare **WinPic 800** (Versione 3.59) selezionare lingua italiano

2.Una volta completata l'istallazione lanciare il SW e selezionare Idioma (lingua) italiano.

3.Sulla barra dei menù andare su Impostazioni poi Hardware Selezionare JDM Programmer e la seriale utilizzata.

4.Selezionare PIC 18F in alto a destra e sotto il 18f2550

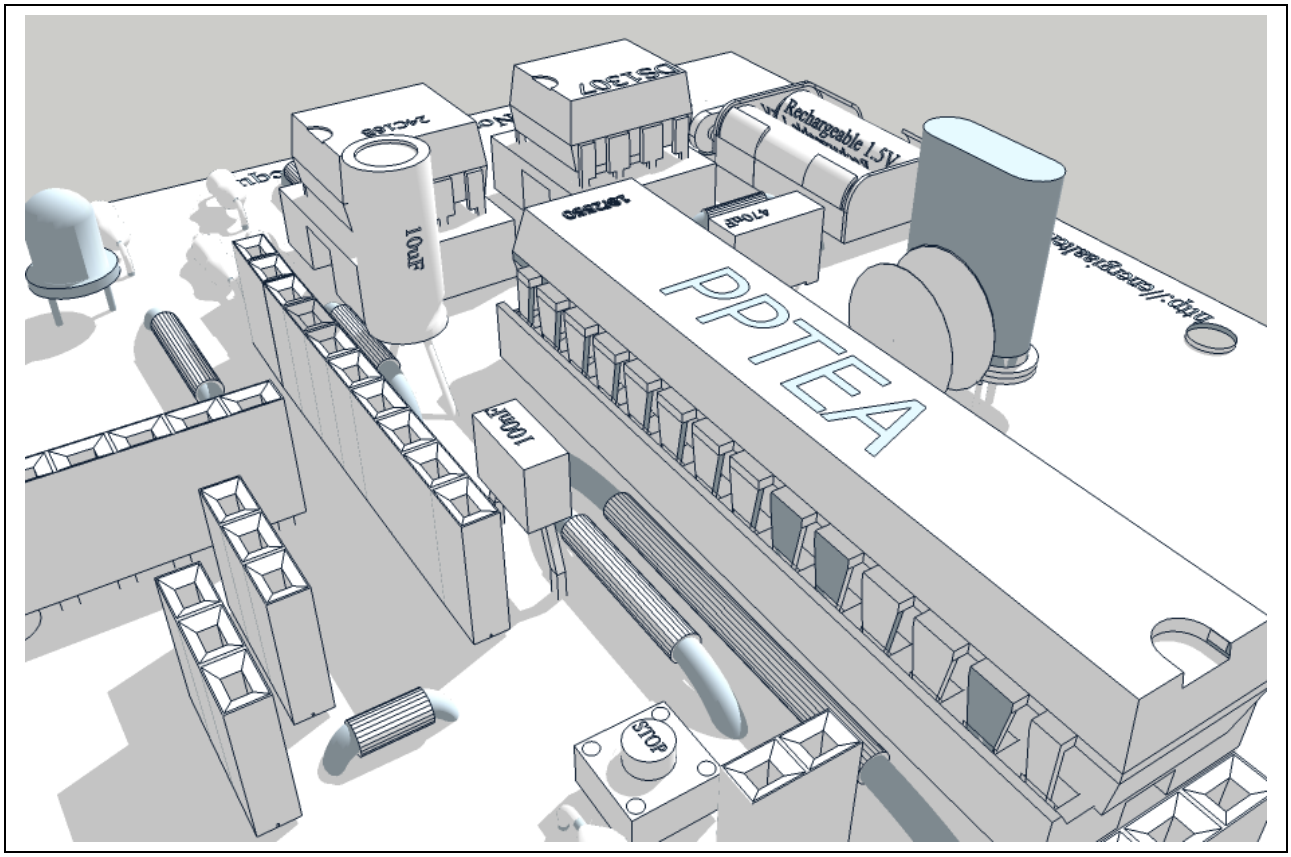
5.Connettere la presa seriale e la presa usb sul PC.

6.Verificare la presenza del pic premendo il chip con sopra il punto interrogativo giallo...apparirà una finestra con su scritto :rilevato->18f2550.

7.Selezionare l'hex da caricare mediante il file di menù File, Apri e indicare il file contenente il firmware del PPTEA.

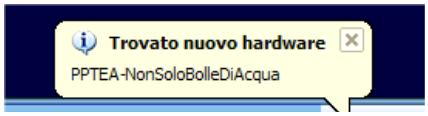
8.Premere il pulsante con la freccia rossa a sinistra premerlo...il pic viene programmato

9. Attendere diversi secondi…se viene visualizzato il verde la programmazione è stata effettuata con successo.

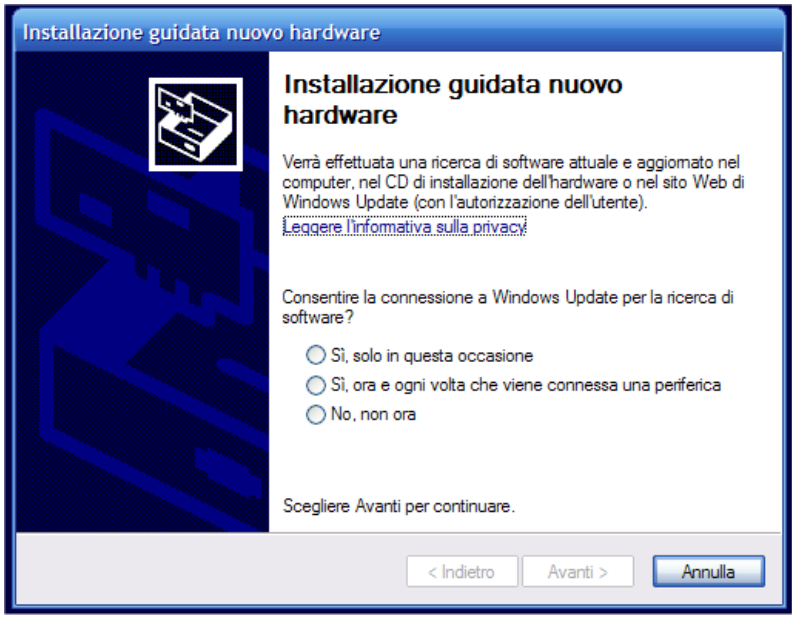


ISTALLAZIONE DRIVER SU PC E CONFIGURAZIONE PORTA USB

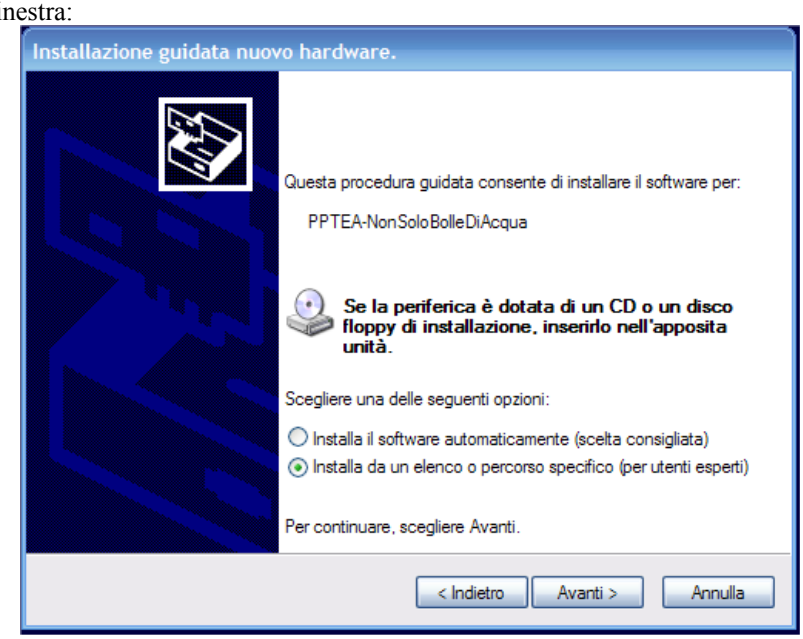
Dopo aver inserito il firmware nel PIC e realizzato la basetta di interfaccia usb (si dovrà agganciare il connettore maschio usb ad una qualsiasi porta usb del PC. Se il circuito e il firmware sono corretti apparirà sul PC la segnalazione di un nuovo hardware:



Dopo qualche secondo apparirà la schermata di istallazione di un nuovo hardware :

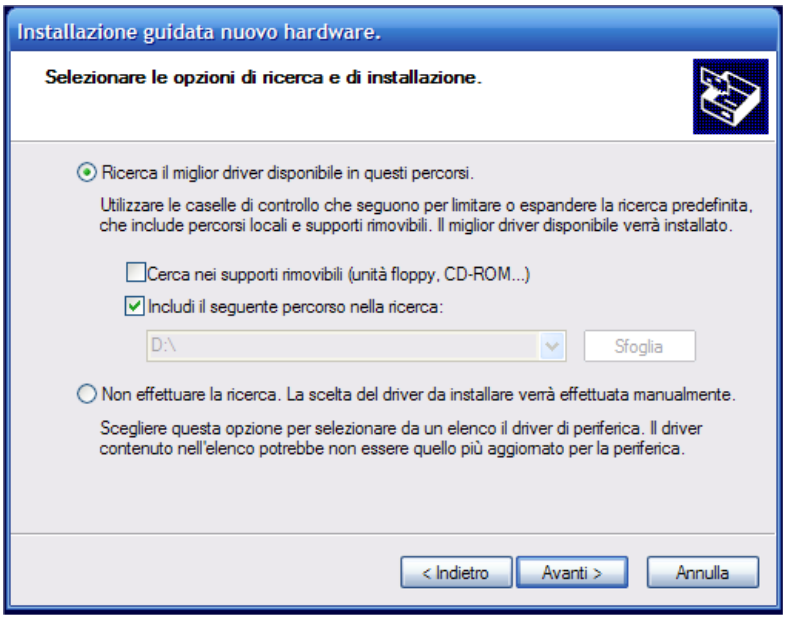


Selezionare "No, non ora" e premere il pulsante avanti.
Apparirà una nuova finestra:

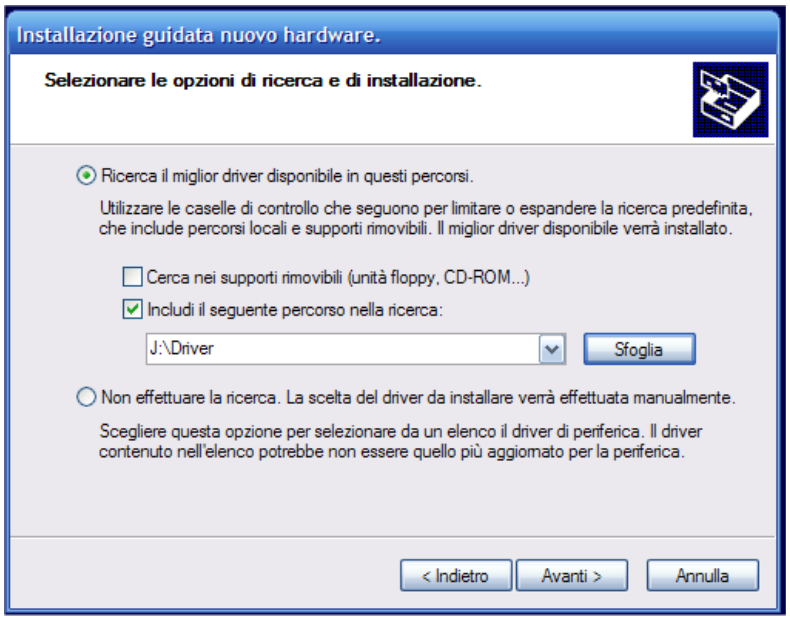


Si dovrà selezionare "Installa da un elenco o percorso specificato (per utenti esperti)"

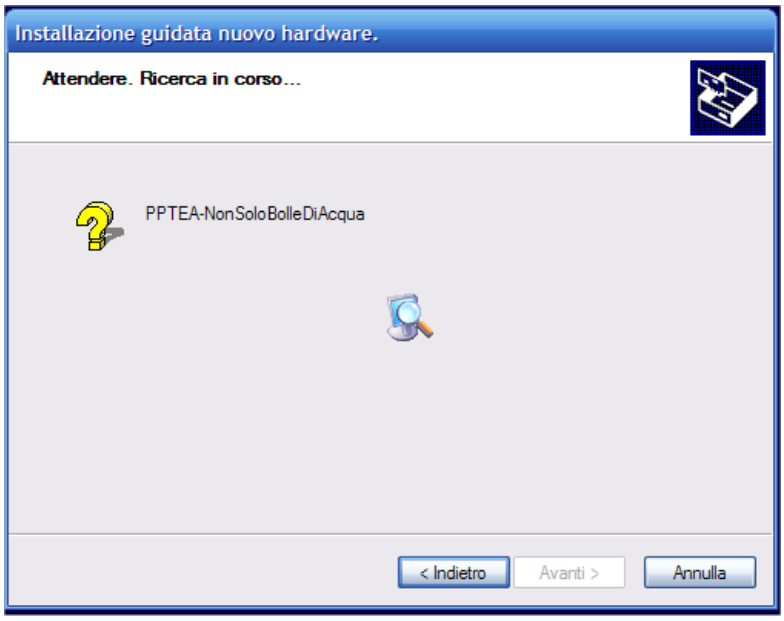
Apparirà una nuova finestra ,si dovrà selezionare "Includi il seguente percorso nella ricerca"

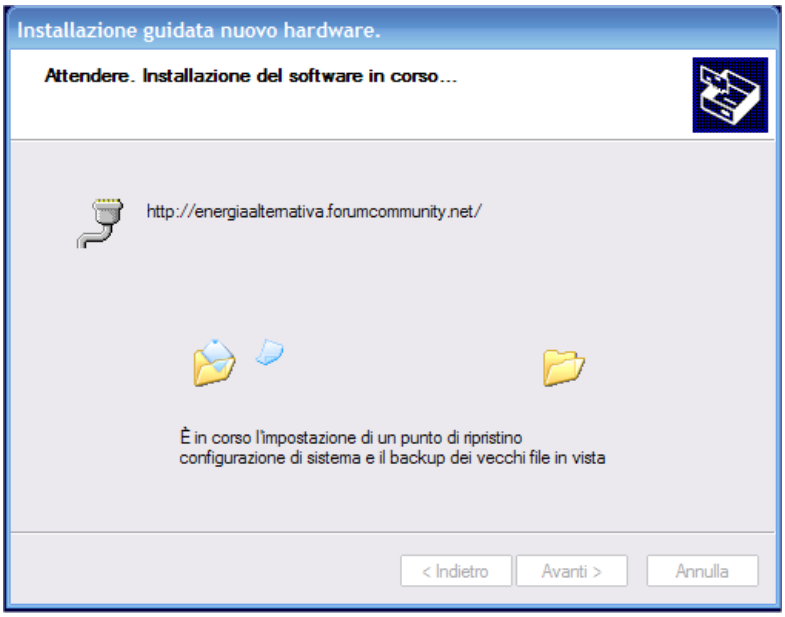


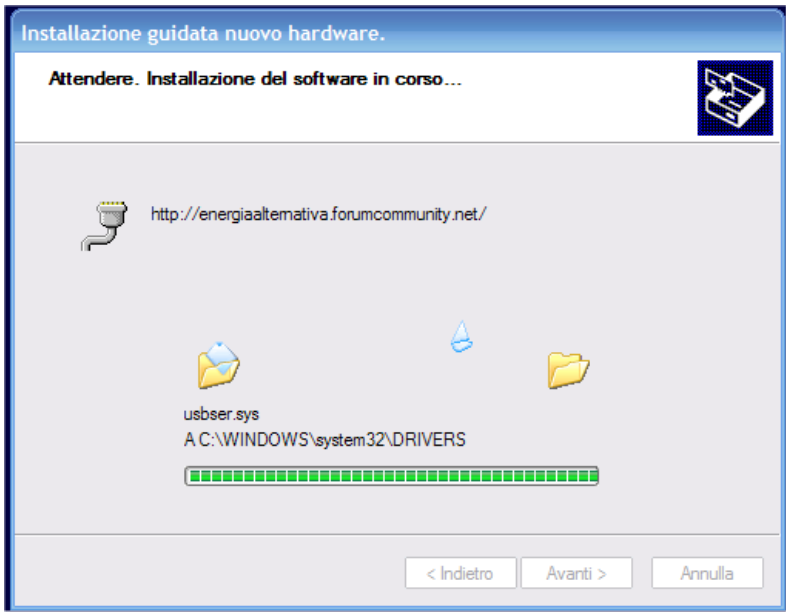
Si dovrà premere sfoglia e selezionare la cartella dove sono presenti i driver del PPTEA.



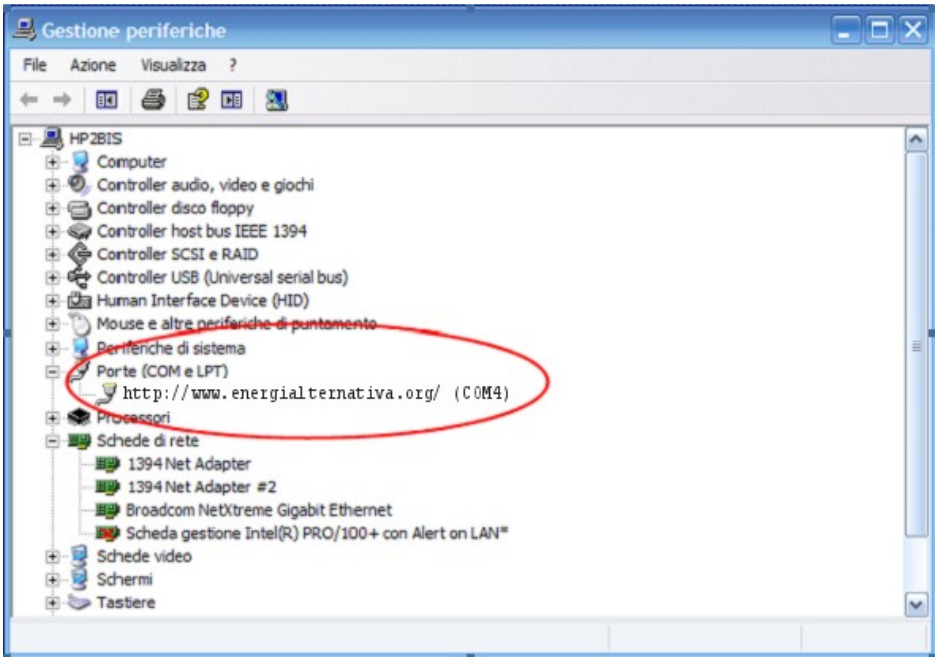
Il Pc inizierà la configurazione dei driver:



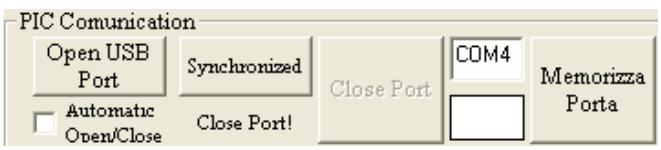




Premere il pulsante fine al termine dell'istallazione dei driver.
Una volta che i driver sono correttamente istallati occorre sapere il numero di porta con cui il sistema operativo vede il dispositivo PPTEA. Dovremo aprire la finestra di gestione delle periferiche e trovare con quale COM è stato 'battezzato' il dispositivo: nell'esempio è la **COM4** (tra parentesi dopo http://www.energialternativa.org).



A questo punto occorre lanciare il compilatore PPTEA premere il pulsante "TRANSFER CODE ON PIC" e inserire la com visualizzata (*in questo caso COM4*) nel campo della porta e dovremo premere il pulsante "Memorizza Porta".



PRIMA CONNESSIONE PPTEA-PC

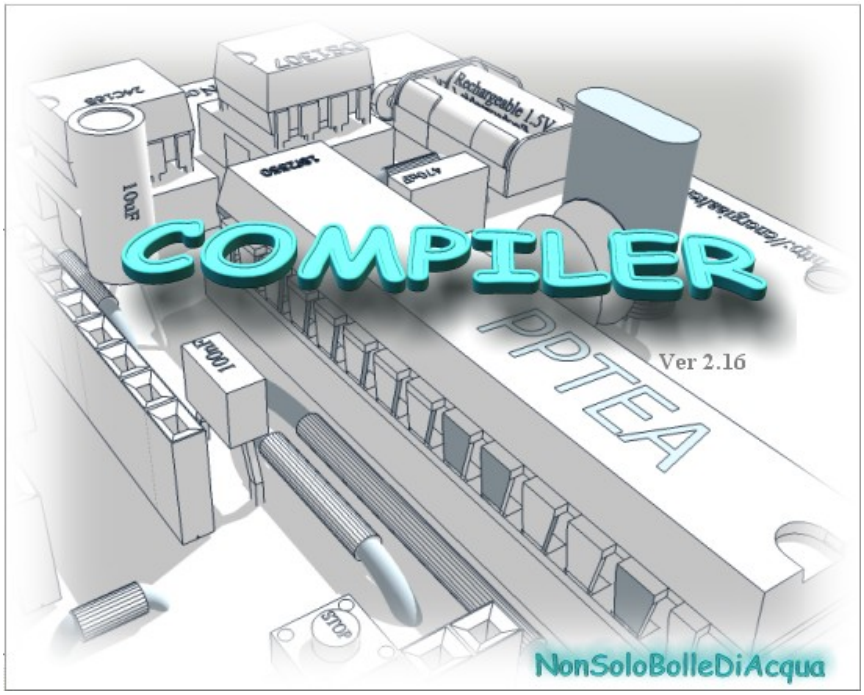
Dopo aver configurato la porta COM (vedi paragrafo **ISTALLAZIONE DRIVER SU PC E CONFIGURAZIONE PORTA**) possiamo far riconoscere il PPTEA al PC.
Ci sono diversi modi, descriviamo due procedure:
PROCEDURA A:
1.Con la porta usb del PPTEA non ancora agganciata alla porta usb del pc lanciare il compilatore del PPTEA
2. Premere il pulsante 'TRANSFER CODE ON PIC' e successivamente il pulsante 'Syncronized'.
3.Inserire la porta usb del PPTEA al PC.
4.Appena si vede lo stato della porta diventare verde occorre premere il pulsante di STOP (può apparire la scritta STOP oppure no).
5.Premere il pulsante del compilatore 'PPTEA VERSION' per visualizzare la versione del firmware presente nel PPTEA.

PROCEDURA B:
1.Inserire l'usb del PPTEA nella porta usb del PC.
2.Appena si sente il suono di windows (blen blen) premere il pulsante di STOP.
3.Lanciare il compilatore del PPTEA, premere 'TRANSFER CODE ON PIC' e aprire la porta seriale del compilatore: lo stato deve diventare VERDE.
4.Premere il pulsante del compilatore 'PPTEA VERSION' per visualizzare la versione del firmware presente nel PPTEA.

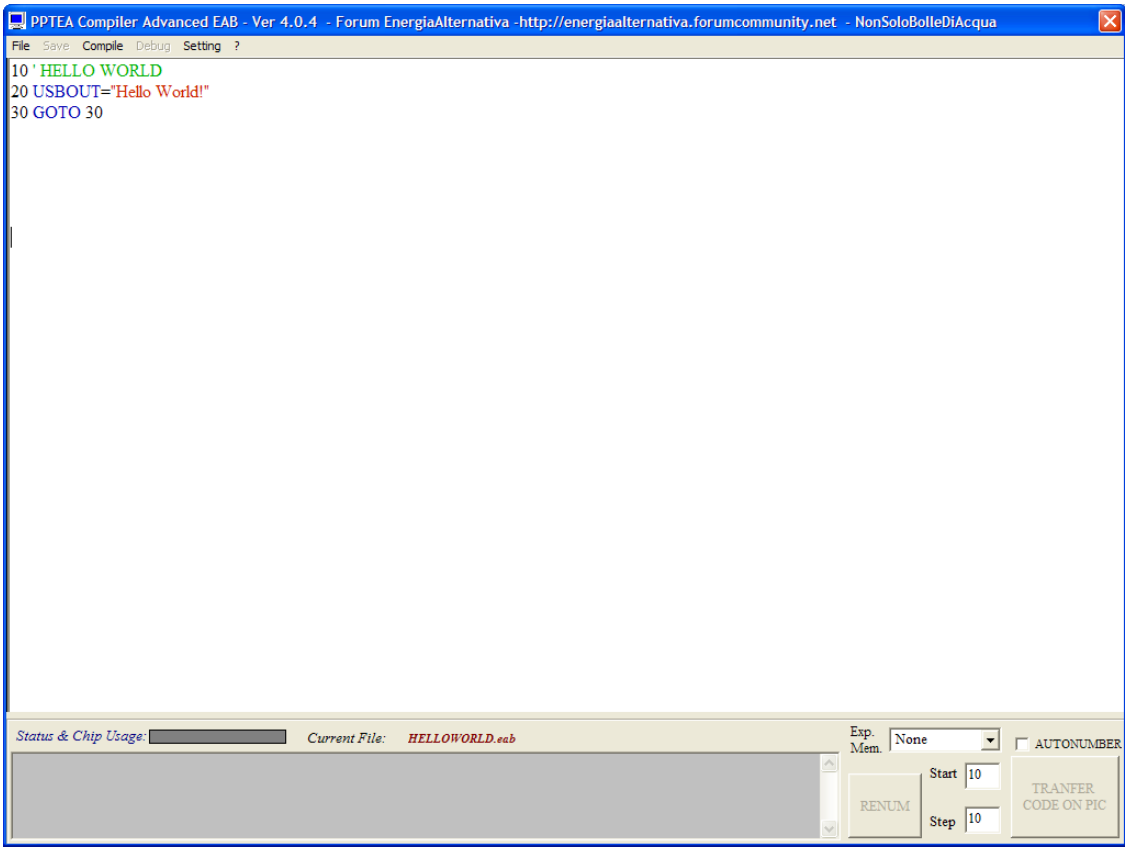
se lo stato diventa verde il PPTEA è stato configurato e riconosciuto correttamente dal PC. Vedere il paragrafo "***IL PRIMO PROGRAMMA:HELLO WORLD!***" per prendere dimestichezza con il compilatore ed eseguire il primo programma.

IL PRIMO PROGRAMMA HELLO WORLD e PROCEDURA DI TRASFERIMENTO CODICE.

Realizziamo insieme il primo programma cercando di compilare, trasferire il codice e far eseguire il programma dal PPTEA. Innanzitutto occorre lanciare il compilatore del PPTEA : PPTEACompiler.exe. La risoluzione dello schermo deve essere uguale o superiore a 800x600.
Mandato in esecuzione verrà visualizzata la schermata di presentazione:



Dopo qualche secondo o dopo un click con il mouse verrà visualizzata la schermata principale:

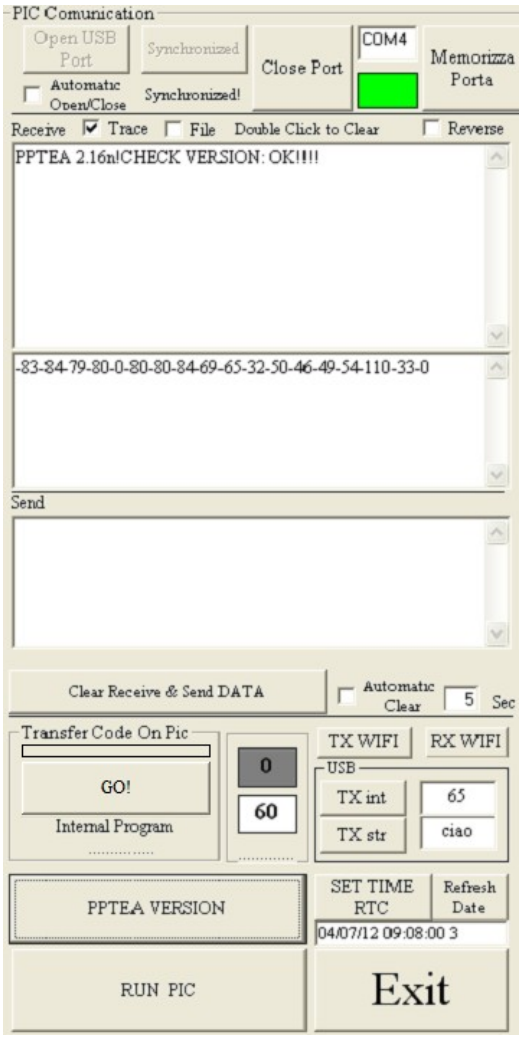


All'interno dell'area testo dovremo scrivere il programma comprensivo di numeri di linea:

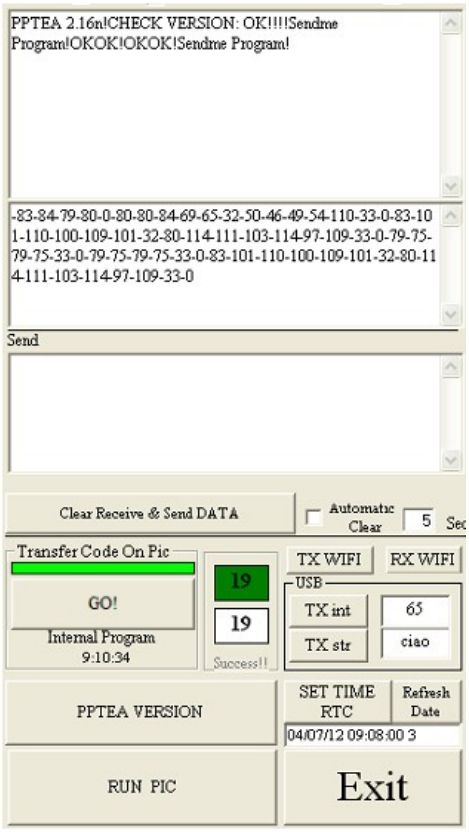
```
10 REM HELLO WORLD
20 USBOUT="Hello World!"
30 GOTO 30
```

Scritto il programma dovremo salvarlo mediante il comando "Save As" presente nel menù a tendine "File" con il nome HelloWorld.eab . Fatto questo dovremo compilare il codice scritto mediante il comando "Compile".
Se la compilazione avrà successo apparirà lo Status di colore verde confermato da un doppio beep di successo.
Verranno utilizzati 19 Token con un occupazione del 7% della memoria.

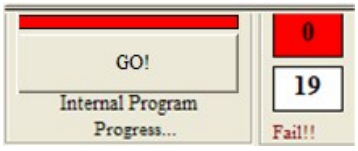
Il pulsante "TRANSFER CODE ON CHIP" sarà abilitato e dovremo premerlo. Si aprirà una finestra sulla destra dello schermo:



Dopo aver configurato la porta COM e averla memorizzata (Vedi CONFIGURAZIONE PORTA COM E DRIVER), dovremo far vedere il PPTEA al PC (vedere il paragrafo **PRIMA CONNESSIONE PPTEA-PC**). Se la porta è configurata correttamente apparirà il riquadro verde. A questo punto occorre premere il pulsante di stop presente nel circuito del PPTEA (Vedi SCHEMA ELETTRICO). Verificare che il flag "Trace" sia settato. Se un programma è in esecuzione verrà scritta nella area Receive la scritta "STOP"...se non gira nessun programma non verrà scritto nulla. Per verificare lo stato di STOP si può inviare la richiesta di Versione al PPTEA, il quale risponderà con la versione corrente del firmware"PPTEA Vx.y.z!"
Se avverrà quanto specificato, occorre premere il pulsante Go per trasferire il codice all'interno del PIC.
Dopo un attesa di un paio di secondi il programma sarà caricato all'interno del PIC.



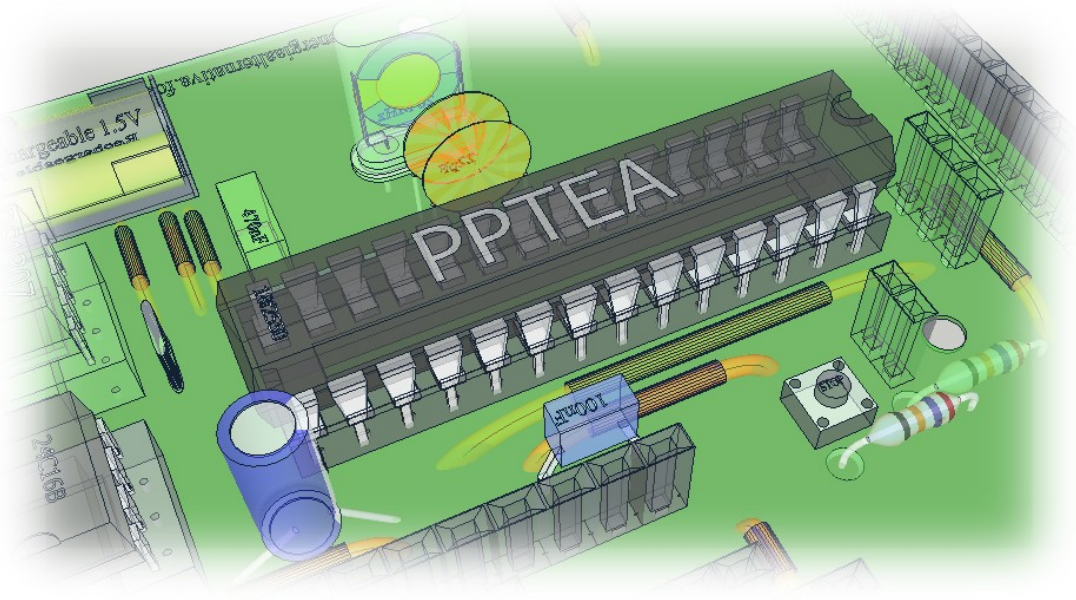
Potremo eseguire il programma premendo il pulsante "RUN PIC" che manderà in esecuzione il programma e scriveà nella finestra di Receive "Hello World!".
Il programma a questo punto è memorizzato all'interno del PPTEA e anche se si dovesse l'evare l'alimentazione varrà conservato all'interno. Il programma verrà automaticamente eseguito dopo una attesa di circa 3 secondi anche senza la presenza del PC...basterà alimentare il PPTEA.C'è la possibilità di evitare questa attesa mediante lo IMMEDIATE "START " (Vedi schema elettrico). Per modificare il programma o reinserirne uno nuovo basterà premere il pulsante "STOP" e ricominciare la procedura. Se si tenta di inviare un programma e il PPTEA è in esecuzione verrà segnalato l'errore “Fail!!”:



In questa situazione occorre premere il pulsante di STOP e reinviare il codice. Se Fail sussiste probabilmente non c’è connessione con la porta USB.
Il flag sul campo Automatic il compilatore gestirà in modo automatico ed autonomo l’aggancio/sgancio del dispositivo, l’opzione del flag viene memorizzata ed inizia ad avere effetto solo dopo la prima operazione di aggancio e/o sgancio porta usb.

CODICE EABASIC

Il codice viene scritto e compilato nell'ambiente del Compilatore PPTEA e il linguaggio di programmazione si chiama *EABASIC*. Il codice EABasic è composto da linee di codice ed ognuna viene compilata e trasformata in un 'tokenizzato' che può essere trasferito nel PPTEA. Ogni un numero linea ed è limitata da 120 caratteri di tokenizzato , se tale linite viene sorpassato viene dato dal compilatore l'errore :" "ERROR LINE TO LONG TOKENIZER". In questo caso occorrerà spezzare la linea di codice in due. Il programma trasferito nel PPTEA rimane residente e viene mantenuto al suo interno anche se non alimentato. Un nuovo programma può essere inserito nel PPTEA e non ci sono limiti al numero di scritture.



LINGUAGGIO EABASIC

Il linguaggio EABasic è il linguaggio con cui si programma il PPTEA. Questo linguaggio è stato realizzato, progettato e implementato per semplificare al massimo la programmazione da parte dell'utente. In questo EABasic ogni tipo di operazione è semplificata e il compilatore tenta di interpretare anche l'intenzione del programma. Ad esempio l'utente può realizzare un codice dove avviene una somma tra una stringa ad un numero, operazione non possibile con un basic tradizionale. Gli spazi non hanno effetto nel codice.

NUMERI DI LINEA

Nel linguaggio EABasic ogni linea di codice deve essere preceduta da un numero di linea che deve essere progressiva e ordinata dal valore più basso al valore più alto. Il numero di linea può assumere valori interi compresi tra 1 a 32767. Ogni altro valore produrrà un errore da parte del compilatore: *LINE NUMBER ERROR!* . Non è importante da quale numero di linea si inizi, l'importante il codice sia composta da numeri di linee ordinate in modo crescente. L'esecuzione parte dal numero di linea più basso. Ogni numero di linea deve essere seguito da del codice o da un commento e non può essere compilato un codice senza numeri di linea, in questo caso il compilatore segnalerà l'errore: NO PROGRAM IN MEMORY!

EABASIC SENZA NUMERI DI LINEA

L'istruzione ad inizio codice *PRAGMA NO_NUM_LINE* permette l'eliminazione dei numeri di linea. I salti incondizionati vengono effettuati esclusivamente con le Label.

TIPI

L'EABasic gestisce automaticamente i tipi e li converte in modo automatico:

a. Interi Numeri interi (32 Bit da *-2147483647* a *+ 2147483647*)

b. Float Numeri in virgola mobile (32 Bit da $2^{-126} \approx 1.17549435e-38$ a $2^{128} * (2-2^{-15}) \approx 6.80564693e+38$)

c. Stringhe Insieme di caratteri alfanumerici (Max 84 caratteri)

COSTANTI

Le costanti sono valori che non possono variare e sono spesso utilizzate per le inizializzazioni.

Esempi di costanti intere:

10 -30 25676 1000000 -212312423

Esempi di costanti float:

10.0 -32.3 0.025676 1000.0 23123.2323

Esempi di costanti stringa:

"CIAO" "GENNAIO" "COME?" "OK!"

Esempi di costanti esadecimali:

&H00 &HFFFF &1232 &HFACD

Esempi di costanti binarie:

&B00 &B1010 &B0110101 &B1111111111

VARIABILI

Le variabili contengono valori che possono variare nel corso dell'esecuzione del programma. I valori che possono assumere sono di 3 tipi: intero, float, stringa. La variabile deve iniziare con un carattere,non deve contenere spazi e non deve avere lo stesso nome delle funzioni e/o comandi.
Esempi di variabili valide:

A=10
B=70.0
VALORE="43.0"

Esempio di variabile MOD errata visto che esiste la funzione MOD (modulo)

10 MOD=25

Nel codice si possono utilizzare al massimo 66 variabili. Queste vengono specificate nella fase 6 del compilatore dove vengono identificate le variabili utilizzate ed il loro numero:

Fase 6: Check Variable!
*Fase 6: Cheched 7 Variable Usage! (**C-A-SF-MENU-MODIFICA_ELEMENTO-G-R**)*

QUANDO IL PPTEA CAMBIA TIPO ALLE VARIABILI

E' importante capire quando e perché il PPTEA cambia il tipo alle variabili. Questa operazione è effettuata automaticamente ma è sempre possibile forzare il tipo effettuando un casting.

Vediamo l'istruzione :

```
10 A=9/2
```

Nell'istruzione abbiamo il rapporto di due numeri interi, ed in questo caso la variabile A è di tipo intero ed assume il valore 4.
L'istruzione :

```
10 A=float(9/2)
```

forza il compilatore ad assegnare a come float che assume il valore 4.0

L'istruzione :

```
10 A=float(9)/2
```

forza il compilatore ad assegnare a come float che assume il valore 4.5

```
10 A=9.0/2
```

la variabile A è un float (perchè trova il 9.0 che è un float) e il valore dopo aver eseguito l'istruzione è uguale ad 4.5.

Eseguendo le istruzioni :

```
10 B="9"
20 A=B/2.0
```

la variabile A è un float (perchè trova il 2.0 che è un float e la variabile stringa B viene convertita in float) ed il valore dopo aver eseguito l'istruzione è uguale ad 4.5.

ESPRESSIONI NUMERICHE

Il linguaggio EABasic mette a disposizione diversi operatori matematici +(somma) – (sottrazione) *(prodotto) / (divisione) etc… l'elenco completo è nel paragrafo **OPERATORI MATEMATICI/STRINGHE.** Vediamo alcuni esempi:

PRINT 3+4	*Somma 3 +4*	*7*
PRINT 86+59	*Somma 86+59*	*145*
PRINT 145.3+12+ 7.2	*Somma 145.3+12+7.2*	*164.500000*
PRINT 45-12	*Sottrae 12 a 45*	*33*
PRINT 12.6-2.6	*Sottrae 2.6 a 12.6*	*10.000000*
*PRINT 7*5*	*Prodotto tra 7 e 5*	*35*
*PRINT 3.23*2*	*Raddoppia 3.23*	*6.460000*
*PRINT 7.98*56.07*	*Prodotto tra 7.98 e 56.07*	*447.438600*
PRINT 35/7	*Divisione tra 35 e 7*	*5*
PRINT 35/7.0	*Divisione tra 35 e 7.0*	*5.000000*
PRINT A+B	*Somma A a B*	
PRINT A-B	*Sottrae B ad A*	
*PRINT A*B*	*Prodotto A a B*	
PRINT A/B	*Divisione tra A e B*	

Funzioni e Comandi dell'*EABASIC*

(F)=Funzione (C)=Comando (S)=Statement

' (Apice alto) (C)	PI (F)
ABS (F)	POW (F)
ACOS (F)	PRAGMA (S)
ASC (F)	PRINT (C)
ASIN (F)	PWMDC1 (C)
ATAN (F)	PWMDC2 (C)
BREAK (C)	PWMDC3 (C)
CADS (C)	PWMFQ1 (C)
CADSEQ (C)	PWMIO1 (C)
CADSIND (F)	PWMIO2 (C)
CADSn (F) n=1…5	PWMIO3 (C)
CALL (F/S)	RANDOMIZE (C)
CASE (S)	RASPSEND (C)
CASEEND (S)	RASPRECEIVE (C)
CELL (S)	READ (S)
CHR (F)	REEPROM (F)
CLR (C)	REM (C)
CONSTANT (S)	REPEAT-UNTIL (S)
CONTINUE (C)	RESCOUNTER (C)
COS (F)	RESTORE (S)
COUNTER (F)	RETURN (C)
DATA (S)	RIGHT (F)
DATE (F)	RND (F)
DEFINE (S)	SERIALCLOSE (C)
DIM (S)	SERIALINP (F)
EEXTERNAL (C)	SERIALINPSTR (F)
END (C)	SERIALOUT (C)
ERROR (F)	SERIALSPEED (C)
EXTINP (F)	SERIALRESTART (C)
EXTOUT (C)	SET (C)
FDATE (C)	SETCOUNTER (C)
FLOAT (F)	SETDATE (C)
FOR (S)	SETIO (C)
FRQ (C)	SGN (F)
GOSUB (C)	SIN (F)
GOTO (S)	SQR (F)
IF-THEN-ELSE (S)	SUB (S)
INCLUDE (C)	SUBEND (S)
INP (F)	SUBEXIT (S)
INPBIT (F)	SWITCH (S)
INSTR (F)	SWITCHEND (S)
INT (F)	TAN (F)
IOMODE (C)	TIMER (T)
LABEL (S)	USBINP (F)
LCDCLEAR (C)	USBINPSTR (F)
LCDINIT (C)	USBOUT (C)
LCDPOS (C)	USBOUTB (C)
LCDWRITE (C)	VOLUME (C)
LEFT (F)	WAITMS (C)
LEN (F)	WAITS (C)
LOG (F)	WEEPROM (C)
MATH_PRECISION (C)	WHILE –LOOP (S)
MEMORY (F)	WIFIDISP (C)
MID (F)	WIFIDISPINP (F)
NEXT (S)	WIFIINP (F)
ON GOTO (C)	WIFIINPNW (F)
OTHER (S)	WIFIOUT (C)
OUT (C)	WEBRECEIVE (C)
OUTBIT (C)	WEBSEND (C)

Questa parte riguarda le funzioni e i comandi del linguaggio di programmazione EABasic.

- **' (Apice alto)**
  Commento sulla linea anche dopo comandi o istruzioni.

  Esempio:
```
10 A=10 ' IDENTIFICA IL GIORNO 10
.....
```

- **ABS(EspressioneNumerica)**
  Ritorna il valore assoluto.

  Esempio:
```
10 A=-30
20 B=ABS(A) ' B= 30
30 C=40
40 D=ABS(C) 'D=40
.....
```

  Esempio:
```
10 '-----------------------------------------
15 'TEST ABS CON OUTPUT SU PC
20 '-----------------------------------------
25 A=-30
30 USBOUT= A &"/"
35 B=ABS(A)
40 USBOUT= B &"/"
45 C=40
50 USBOUT= C &"/"
55 D=ABS(C)
60 USBOUT= D &"/"
65 GOTO  65
```

  Output PC:
  -30/30/40/40/

- **ACOS(radiant-expression)**
  Ritorna il arcocoseno trigonometrico.

  Esempio:
```
10 '-----------------------------------------------
15 'TEST SIN/ASIN CON OUTPUT SU PC
20 '-----------------------------------------------
22 MATH_PRECISION=18
25 A=45*PI/180
40 C=SIN(a)
45 USBOUT=C & " "
50 D=ACOS(C)
55 USBOUT= d*180/PI &""
65 GOTO  65
```

  Output PC: 0.707106 45.000003

  Vedere altro esempio nella funzione ACOS.

- **ASC(Stringa)**
  Ritorna il codice ascii del primo carattere della stringa. Vedi la tabella ascii.

Esempio:

```
10 A=”H”
20 B=ASC(A) ' B= 72
30 C=”BYE”
40 D=ASC(C) 'D=66
50 .....
```

Esempio:

```
10 '-----------------------------------------
15 'TEST ASC/CHR CON OUTPUT SU PC
20 '-----------------------------------------
25 A=ASC("A")
30 FINE=ASC("Z")
35 B=CHR(A)
40 C=ASC(B)
45 USBOUT= B & "=" & C &","
50 A++
55 IF A<=FINE THEN 35
60 GOTO  60
```

Output PC:
A=65,B=66,C=67,D=68,E=69,F=70,G=71,H=72,I=73,J=74,K=75,L=76,M=77,N=78,O=79,P=80,Q=81,R=82,
S=83,T=84,U=85,V=86,W=87,X=88,Y=89,Z=90,

- **ASIN(radiant-expression)**
  Ritorna il arcoseno trigonometrico.

Esempio:

```
10 '-------------------------------------------------
15 'TEST SIN/ASIN CON OUTPUT SU PC
20 '-------------------------------------------------
22 MATH_PRECISION=18
25 A=45*PI/180
40 C=SIN(a)
45 USBOUT=C & " "
50 D=ASIN(C)
55 USBOUT= d*180/PI &""
65 GOTO  65
```

Output PC: 0.707106 44.999996

Vedere altro esempio nella funzione ACOS.

- **ATAN(Espressione In Radianti)**
  Ritorna l’arcotangente trigonometrica . Vedi la funziona TAN.

- **BREAK**
  Interrompe il ciclo While, For o Repeat che è in esecuzione saltando a fine ciclo.

```
10 FOR I=1 TO 10
20    PRINT "I="& I
25    IF I =5 THEN BREAK
30 NEXT I
40 END
```

Output PC: I=1I=2 I=3 I=4 I=5

```
PRAGMA NO_NUM_LINE
SUB TEST_FOR
FOR Y=1 TO 3
 FOR I=1 TO 10
   PRINT "Y="& Y &" I="& I
   IF I =3 THEN BREAK
 NEXT I
NEXT Y
PRINT "FINE"
END
SUBEND
```

Output PC: Y=1 I=1 Y=1 I=2 Y=1 I=3 Y=2 I=1 Y=2 I=2 Y=2 I=3 Y=3 I=1 Y=3 I=2 Y=3 I=3 FINE

```
PRAGMA NO_NUM_LINE
PRINT "INIZIO PROG"
WHILE a< 10
        A++
        PRINT  "A="&A
        IF A=5 THEN BREAK
        WAITMS 250
 LOOP
PRINT "FINE PROG"
END
```

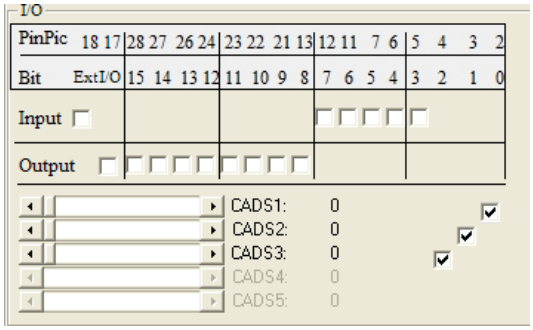
*Output PC:*INIZIO PROG A=1 A=2 A=3 A=4 A=5 FINE PROG

```
05 USBOUT="INIZIO PROG"
06 a=0
07 REPEAT
20   A++
25   WAITMS 250
26   USBOUT="A="&A &"-"
27   IF A=5 THEN BREAK
30 UNTIL a<10
40 USBOUT ="FINE PROG"
50 GOTO 50
```

*Output PC:*INIZIO PROG A=1-A=2-A=3-A=4-A=5-FINE PROG

- **CADS = Expr**

  Configura il numero di convertitori analogico digitali utilizzati e il default è uguale a 3 con un massimo di 5.



Esempio:
10 CADS=4
20 GOTO 20

Porta a 4 il numero di convertitori utilizzabili.

Esempio:
10 CADS=0
20 ….

Disabilita i Convertitori Analogico Digitali e predispone il PPTEA ad avere solo ingressi ed uscite digitali.

- **CADSEQ(VETT, CONVERTITORE, TicWait, NElementi)**

  Legge NElementi prelevati dal/dai convertitori A/D con un delay di tic (di x microsencodi) e li inserisce nel Vettore. Il vettore deve essere adeguato per contenere i dati (Vedi istruzione Dim).

- **CADSIND(n)**

  Torna il valore letto dal Convertitore Analogico Digitale n.Il valore letto ha 10 bit di precisione (0-1023) e il valore di n deve essere compreso tra 1 e 5. Se il convertitore selezionato non è valido viene letto l'ultimo valido selezionato.
  PS:Il debugger segnala un eventuale convertitore n errato con l'errore "SENDORn ADC DISABLE"

Esempio:
10 B=1
20 A=CADSIND(B)
……

- **CADSn**
  CADS1, CADS2,CADS3,CADS4,CADS5

  Torna il valore letto dal Convertitore Analogico Digitale n.Il valore letto ha 10 bit di precisione (0-1023) e sono utilizzabili al massimo cinque convertitori.

  Esempio:
  10 A=CADS1
  20 B=CADS2
  30 C=CADS3
  .....

- **CASE Expr**
  Costrutto dello statement SWITCH. Se è verificata l'espressione esegue il codice sottostante.

- **CASEEND**
  Costrutto dello statement SWITCH. Termina il blocco relativo al CASE.

- **_CALL SUBPROGRAM[(PAR1,PAR2,….)]**
- **VAR=CALL SUBPROGRAM[(PAR1,PAR2,….)]**

La CALL trasferisce il controllo al relativo sottoprogramma SUB. Il programma riprende all'istruzione successiva. La CALL può avere parametri di chiamata e un parametro di ritorno. Si possono avere al massimo 10 Call/Gosub midificate.

Esempio Call INIT senza parametri:

```
PRAGMA NO_NUM_LINE

CALL INIT
END

SUB INIT
   PRINT "INIZIALIZZZAZIONE"
SUBEND
```

il risultato dell'esecuzione sarà:

INIZIALIZZAZIONE

Esempio Call un solo parametro:

```
10 A=25
15 CALL PIPPO(A)
20 PRINT B
21 CALL PIPPO(27)
22 PRINT B
23 CALL PIPPO(PI)
24 PRINT B
25 END

160 'SUBROUTINE
170 SUB PIPPO(PAR1)
175 B=PAR1*3
180 SUBEND
```

il risultato dell'esecuzione sarà:

75
81
9.424778

Esempio Call con tre parametri:

```
10 CALL CIAO(PI, 3, 5*2)
18 PRINT B
20 END
100 SUB CIAO(A,D,C)
110 B=A*2+D+C
120 SUBEND
```

il risultato dell'esecuzione sarà:

19.28318

Esempio Call con parametro di ritorno:

```
10 A=CALL SOMMA(3,5)
20 PRINT A
30 END

40 ' SUBROUTINE SOMMA DI PARAMETRI
50 SUB SOMMA(PAR1,PAR2)
60 SOMMA=PAR1+PAR2
```

```
70 SUBEND
```
il risultato dell'esecuzione sarà:
8

- **CELL= EXPR**
  Setta la cella della eeprom su cui leggere o scrivere. Il valore di default è 255.

  ATTENZIONE: Nella EEPROM è contenuto il codice del PPTEA, occorre quindi assicurarsi che la cella non sia una utilizzata dal programma che scriviamo. I Byte utilizzati vengono specificati nella fase 7 di compilazione e tale numero viene riportato nella cella a fianco del pulsante "Transfer Code On Pic".

- **CHR(Espressione Numerica)**
  Ritorna la stringa composta dal carattere associate al codice ascii Espresione Numerica.

  Esempio:
```
10 A=66
20 B=89
30 C=69
40 D=CHR(A)&CHR(B) &CHR(C) 'D=”BYE”
.....
```

  Vedere altro esempio nella funzione ASC.

- **CLR VARIABILE [, VARIABILE, VARIABILE,…]**
  Azzera il contenuto della variabile della variabile o dell’elemento del vettore.

  Esempio:
```
05 B=37
10 A=25
12 USBOUT= a &"-" & b &"-"
15 CLR A,B
16 USBOUT= a &"-"& b &"-"
20 GOTO 20
```

  Output PC:
  25-37-0-0-

  ATTENZIONE :L’istruzione CLR OUT (oppure outbit) non ha effetto sull’I/O.

- **CONSTANT Nome = espressione**
  Assegna l’espressione alla costante Nome.

  Esempio:
```
10 CONSTANT OK=1
20 CONSTANT VAL2=”HELLO”
30 CONSTANT MOLT=3*4+1
40 A=MOLT+1 ' A=14
50 B=OK      ' B=1
60 C=VAL2    ' C=”HELLO”
.....
```
  Esempio:
```
10
'-----------------------------------------------------
15 'TEST CONSTANT CON OUTPUT SU PC
20
'-----------------------------------------------------
25 CONSTANT OK=1
30 CONSTANT VAL2="HELLO"
35 CONSTANT MOLT=3*4+1
40 A=MOLT+1
45 USBOUT=A & ","
50 B=OK
```

```
55 USBOUT=B & ","
60 C=VAL2
65 USBOUT=C & "!"
70 GOTO  70
```

Output PC: 14,1,HELLO!

- **CONTINUE**

Riprende il ciclo While, For o Repeat non eseguendo le istruzioni presenti fino a fine ciclo.

```
PRAGMA NO_NUM_LINE
SUB TEST_CONTINUE
        FOR Y=1 TO 7
            PRINT "Y="& Y
            IF Y>3 THEN CONTINUE
            PRINT "DOPO IF"
        NEXT Y
        PRINT "FINE"
        END
SUBEND
```

Output PC: Y=1DOPO IF Y=2 DOPO IF Y=3 DOPO IF Y=4 Y=5 Y=6 Y=7 FINE

```
PRAGMA NO_NUM_LINE

PRINT "INIZIO PROG"
WHILE a< 10
        A++
        PRINT  "A="&A
        IF A>=5 THEN CONTINUE
        WAITMS 250
 LOOP
PRINT "FINE PROG"
END
```

Output PC:INIZIO PROG A=1 A=2A=3A=4A=5A=6A=7A=8A=9A=10 FINE PROG
L'esecuzione avviene più velocemente con valori di a maggiori di 5.

```
05 USBOUT="INIZIO PROG"
06 a=0
07 REPEAT
20   A++
23   USBOUT="A="&A &"-"
25   IF A>=5 THEN  CONTINUE
26 WAITMS 250
30 UNTIL a<10
40 USBOUT ="FINE PROG"
50 GOTO 50
```

Output PC:INIZIO PROG A=1 A=2A=3A=4A=5A=6A=7A=8A=9A=10 FINE PROG
L'esecuzione avviene più velocemente con valori di a maggiori di 5.

COS(Espressione In Radianti)

Ritorna il coseno trigonometrico.

Esempio:

```
10 '-------------------------------------------------
15 'TEST SIN/COS CON OUTPUT SU PC
20 '-------------------------------------------------
25 A=45*PI/180
30 B=COS(A)
35 USBOUT=B & ","
40 C=SIN(a/2)
45 USBOUT=C & ","
50 D=23*PI/180
55 E=SIN(a)*SIN(a)+COS(a)*COS(a)
60 USBOUT=E & ""
65 GOTO  65
```

Output PC:

0.707,0.383,1.

- **COUNTER**

Torna il il numero di oscillazioni del contatore esterno. Il contatore esterno convive con il PWM3. Per default è attivo il PWM3 quindi il contatore deve essere attivato con l'istruzione SETCOUNTER. E' possibile azzerare il valore del contatore sempre mediante il comando SETCOUNTER. Per tornare ad utilizzare il PWM3 si utilizza il comando RESCOUNTER. La connessione hardware, come visibile dallo schema elettrico, deve essere effettuata sul pin 11 (BIT 6) per default settata come ingresso.

Esempio:

```
05   A=COUNTER
```

Il valore del contatore viene inserito nella variabile A.

Esempio test contatore con i pin 28 ed 11 in corto:

```
03 ' Dalla versione 2.15 in poi
05 '----- CORTOCIRCUITARE IL PIN 28 CON IL PIN 11 (BIT 16- BIT 4)
10 SETCOUNTER ' ATTIVIAMO IL CONTATORE
15 PWMDC1=50  ' ATTIVIAMO IL PWM
17 PWMFQ1=4  ' AUMENTO LA FREQUENZA DELLE OSCILLAZIONI
20 A=COUNTER
30 USBOUT=A &" " ' VISUALIZZIAMO IL VALORE DEL CONTATORE
40 WAITMS 100
50 GOTO 20
```

- **DATA COSTANT1 [, CONSTANT2….]**

Questo comando permette di memorizzare all'interno del codice dei dati per poi richiamarli mediante il comando READ. L'istruzione DATA deve essere presente prima dell'istruzione READ. E' possibile utilizzare più linee di codice DATA ma queste debbono essere consecutive.

Esempio:

```
10 DIM A(10) AS WORD
20 DATA 25,40,47
30 DATA 19,36,"CIAO",89,34
35 READ A(3),D,E,F
37 USBOUT=a(3) &" " & D & " " & E & " " &F &"-"
38 READ A(4),D,E,F
39 USBOUT=a(4) &" " & D & " " & E & " " &F &"-"
40 GOTO 40
```

Output PC:

25 40 47 19-36 CIAO 89 34-

- **DATE**
  Torna la stringa ricevuta dall'orologio contenente l'orario.

```
10 'L'OROLOGIO DEL PPTEA
20 LCDCLEAR
32 STR=DATE
35 LCDPOS=&H11
38 LCDWRITE=STR
39 USBOUT=STR & CR_LF
40 WAITS 1
45 GOTO 32
```

  Output Display:
  Invia sul display e su USB l'orario del DS1307 (se il display è da 16 colonne non si visualizzano i secondi)

- **DEFINE Macro = definizione**
  Assegna la definizione alla macro. Il compilatore sostituisce la definizione ogni qual volta incontra la macro nel codice. Non può essere ridefinita una macro di una macro.

  Esempio:

```
10 DEFINE PULSANTE_SI = INPBIT(3)
20 IF !PULSANTE_SI THEN 20
30 USBOUT =  "PREMUTO IL TASTO SI" & CR_LF
40 WAITS 1
50 GOTO  20
```

- **DIM Variable(Costante Numerica) as type**
  Dimensiona il vettore Variable con numero di elementi CostanteNumerica (valore compreso tra 0 e 255). Tale dichiarazione deve essere la prima del programma. Viene supportato il tipo BYTE (0-256) , WORD (-32768, 32767), LONG e FLOAT .Il numero di elementi dipende dal codice scritto (vedi *MEMORIA DEL PPTEA*).Dalla versione 2.16d gli elementi vengono inizializzati a zero all'avvio.

  Nell'esempio vengono riempiti i primi dieci elementi del vettore PIPPO con valore doppio rispetto l'indice.

  Esempio:

```
10 DIM PIPPO(10) AS WORD
20 i=i+1
25 IF i= 11 THEN 50
30 PIPPO(i)=i*2
40 GOTO 20
50 END
```

- **EEXTERNAL=EXPR**

  Se il valore dell'espressione è uguale ad 1 le operazioni di scrittura/lettura relative alla EEPROM sono effettuate sulla EXPROM esterna.

  Se il valore dell'espressione è uguale a 0 le operazioni di scrittura/lettura relative alla EEPROM interna al CHIP (valore di default).

  Esempio:

```
10 REM SCRITTURA SULLA CELLA 250 DELLA EEPROM INTERNA ED ESTERNA
15 EEXTERNAL=0 ' EEPROM INTERNA
20 CELL=250
25 WEEPROM=12
30 B=REEPROM
35 USBOUT = "I W="&B
40 EEXTERNAL=1 ' EEPROM ESTERNA
45 CELL=250
50 WEEPROM=50
55 B=REEPROM
60 USBOUT = ",E W="&B
65 EEXTERNAL=0 ' EEPROM INTERNA
70 B=REEPROM
75 USBOUT = ",I W="&B
80 GOTO  80
```

  Output PC: *I W=12,E W=50,I W=12*

  ATTENZIONE: *PER ESEGUIRE IL CODICE OCCORRE AVERE MONTATA UNA EEPROM ESTERNA , COME DA SCHEMA ELETTRICO.*

- **END**

  Termina il programma. Il PPTEA rimane chiuso in un loop infinito. Per mandare in esecuzione il programma effettuare un reset.

  Esempio:

```
20 A=20
30 B=A*2
40 END
```

  Il goto sulla stessa istruzione equivale all'istruzione END.

  Esempio:

```
…….
40 GOTO 40
```

  ATTENZIONE:OGNI PROGRAMMA, SE TERMINA, DEVE TERMINARE CON UNA ISTRUZIONE END O CON UN GOTO SULLA STESSA LINEA. IL COMPILATORE IN TAL CASO DARA' UN WARNING: Compiled but code doesn't close!

  ERRORE:

```
20 A=20
30 B=A*2
```

- **ERROR**

Funzione non disponibile dal firmware 3.1

Ritorna lo stato di errore del PPTEA. Se c'è è presente un errore torna un valore diverso da zero.

Esempio:

```
10 A=""
11 A="ciao" & A
12 B++
15 USBOUT="LEN=" & LEN(A) & cr_lf
20 IF ERROR THEN :PERRORE
30 WAITMS 100
40 GOTO 10
50 :PERRORE
60 USBOUT="ERRORE:"&error
70 GOTO 70
```

Il PPTEA si blocca quando è terminato lo spazio per le stringhe.

- **EXTINP**

Torna il valore del bit di ingresso dell'espansione. Questo bit (pin 18) è sempre di input.

Esempio:

```
10 a=EXTINP
15 USBOUT=a&"-"
20 WAITS 1
30 GOTO 10
```

- **EXTOUT =Val**

Il valore viene inviato sul bit di uscita dell'espansione dell'I/O. Questo bit (pin 17) è sempre di output.

Esempio:

```
10 EXTOUT=1
20 WAITS 1
30 EXTOUT=0
40 WAITS 1
50 GOTO 10
```

Il pin 17 cambia stato ogni secondo.

- **FDATE=Expr**

Setta il formato di uscita dell'orologio esterno. FDATE è posto per dafault a 0.
(gg=giorno, mm=mese, aa=anno, hh=ora, mm=minuti,ss=secondi,gds=giorni della settimana = Lun, Mar, Mer, Giov, Ven, Sab, Dom)

Valore	*Formato Orario*	*Descrizione*
FDATE=0	gg/mm/aa hh:mm:ss	(Data e Orario)
FDATE=1	hh:mm:ss	(Solo Orario)
FDATE=2	gg/mm/aa	(Solo Data)
FDATE=3	gg/mm/aa gds hh:mm:ss	(Data, GiornoDellaSettimana e Orario)
FDATE=4	ss	(Secondi trascorsi dalla mezzanotte)
FDATE=5	ss	(Secondi trascorsi dall'inizio della settimana)
FDATE=6	ss	(Secondi trascorsi dall'inizio del mese)

- **FLOAT(Espressione Numerica)**
  Torna il valore numerico in virgola mobile.
  Il PPTEA inizializza le variabili a seconda degli operatori che gli vengono assegnati.
  Nell'esempio la variabile (nell'istruzione 20) B vale 3 ed è inizializzata come variabile di tipo intero visto che 2 è un valore intero e A è una variabile intera, mentre la variabile C (istruzione 30) vale 3,5 e la variabile D viene inizializzata direttamente in virgola mobile per la presenza del cast *float*.

Esempio:

```
10 A=7
20 B=A/2
30 C=FLOAT(A)/2
40 D=3.2
50 END
```

Esempio:

```
10 '-----------------------------------------------------
15 'TEST FLOAT CON OUTPUT SU PC
20 '-----------------------------------------------------
30 A=7
40 B=A/2
45 USBOUT="B="&B & "=" & A & "/2,"
50 C=FLOAT(A)/2
55 USBOUT="C="&C & "=" & A & "/2,"
60 D=7.0
70 E=D/2
75 USBOUT="E="&E & "=" & D & "/2"
80 GOTO 80
```

Output PC:
B=3=7/2,C=3.5=7/2,E=3.5=7.0/2

- **FOR VAR=EXPR1 TO EXPR2 [STEP EXPR] …. NEXT VAR**

Nel ciclo FOR-NEXT la variabile VAR assume i valori da EXPR1 a EXPR2. Quando l'esecuzione incontra il costrutto NEXT avviene incremento della variabile che inizialmente assume il valore Expr1 e arriva fino al valore Expr2.Il ciclo FOR-NEXT può essere nidificato e ha come limite 127 cicli nidificati.

Esempio:

```
15   FOR A=4 TO  7
25     USBOUT="A="& A & cr_lf
28   NEXT A
80 GOTO 80
```

Output Pc:

A=4
A=5
A=6
A=7

Esempio decremento (step negativo):

```
15   FOR A=4 TO  1 STEP -1
25     USBOUT="A="& A & cr_lf
28   NEXT a
80 GOTO 80
```

Output Pc:

A=4
A=3
A=2
A=1

Esempio cicli nidificati:

```
15 FOR A=1 TO 3
20   FOR B=1 TO 3
21     FOR C=1 TO 3
25       USBOUT="A="& A & " B=" &B  &" C="&c &cr_lf
26       WAITMS 150
27     NEXT c
29   NEXT b
32 NEXT A
35 USBOUT="FINE CICLI"
40 GOTO 40
```

Output Pc:
INIZIO CICLI
A=1 B=1 C=1
A=1 B=1 C=2
.....
A=3 B=3 C=3
FINE CICLI

- **FRQ PERIODO_QUARTI, DURATA**

Questo comando permette di inviare sul pin 28 del PPTEA una NOTA di una determinata durata.
Il valore della nota va inserito in un quarto del periodo della nota stessa. Questa è la tabella di riferimento:

FREQUENZA (Hz)

OTTAVA	DO	DO#	RE	Mib	MI	FA	FA#	SOL	SOL#	LA	Sib	SI
	C	C#	D	Eb	E	F	F#	G	G#	A	Bb	B
0	16,35	17,32	18,35	19,45	20,6	21,83	23,12	24,5	25,96	27,5	29,14	30,87
1	32,7	34,65	36,71	38,89	41,2	43,65	46,25	49	51,91	55	58,27	61,74
2	65,41	69,3	73,42	77,78	82,41	87,31	92,5	98	103,8	110	116,5	123,5
3	130,8	138,6	146,8	155,6	164,8	174,6	185	196	207,7	220	233,1	246,9
4	261,6	277,2	293,7	311,1	329,6	349,2	370	392	415,3	440	466,2	493,9
5	523,3	554,4	587,3	622,3	659,3	698,5	740	784	830,6	880	932,3	987,8
6	1047	1109	1175	1245	1319	1397	1480	1568	1661	1760	1865	1976
7	2093	2217	2349	2489	2637	2794	2960	3136	3322	3520	3729	3951
8	4186	4435	4699	4978	5274	5588	5920	6272	6645	7040	7459	7902

PERIODO_QUARTI (uS)

OTTAVA	DO	DO#	RE	Mib	MI	FA	FA#	SOL	SOL#	LA	Sib	SI
	C	C#	D	Eb	E	F	F#	G	G#	A	Bb	B
0	15291	14434	13624	12853	12136	11452	10813	10204	9630	9091	8579	8098
1	7645	7215	6810	6428	6068	5727	5405	5102	4816	4545	4290	4049
2	3822	3608	3405	3214	3034	2863	2703	2551	2408	2273	2146	2024
3	1911	1804	1703	1607	1517	1432	1351	1276	1204	1136	1073	1013
4	956	902	851	804	758	716	676	638	602	568	536	506
5	478	451	426	402	379	358	338	319	301	284	268	253
6	239	225	213	201	190	179	169	159	151	142	134	127
7	119	113	106	100	95	89	84	80	75	71	67	63
8	60	56	53	50	47	45	42	40	38	36	34	32

Per far emettere un LA con frequenza a 220Hz dobbiamo mettere il periodo diviso quattro, seguito dal tempo di durata della nota.
Il periodo (inverso della frequenza) si calcola così:
Periodo= 1/220=0,004545 cioè 4,5 mS (quattro virgola cinque millisecondi).
Il periodo va diviso diviso 4 e va scritto in microsecondi:
4500 uS / 4 = 1136 uS.

*Viene effettuata una **pausa silenziosa** se si specifica un periodo_quarti uguale a 0 ed il compilatore ha predefinita la costante **PAUSE =0**.*

Il compilatore del PPTEA ha le costanti sonore predefinite :
_DO _DO# _RE _Mib _MI _FA _FA# _SOL _SOL# _LA _Sib _SI

Le costanti predefinite utilizzano sette ottave: partono da zero ed arrivano fino a 6.
Per specificare l'ottava basta far seguire la nota da numero della relativa ottava.
Sintassi: ***_Nota[Ottava]***
Se l'ottava non è specificata viene eseguita la quarta ottava.

COSTANTI PREDEFINITE PPTEA COMPILER _NotaOttava

OTTAVA	DO	DO#	RE	Mib	MI	FA	FA#	SOL	SOL#	LA	Sib	SI
	C	C#	D	Eb	E	F	F#	G	G#	A	Bb	B
0	3822	3608	3405	3214	3034	2863	2703	2551	2408	2273	2146	2024
1	1911	1804	1703	1607	1517	1432	1351	1276	1204	1136	1073	1013
2	956	902	851	804	758	716	676	638	602	568	536	506
3	478	451	426	402	379	358	338	319	301	284	268	253
4	239	225	213	201	190	179	169	159	151	142	134	127
5	119	113	106	100	95	89	84	80	75	71	67	63
6	60	56	53	50	47	45	42	40	38	36	34	32

ATTENZIONE: PERIODO_QUARTI deve essere un valore positivo.

Esempi:

```
05 '----- SAN MARTINO SUONATO IN 4 OTTAVA----
07 CONSTANT NUMERO_NOTE=34
10 DATA _DO3, _RE3, _MI3, _DO3, _DO3, _RE3, _MI3, _DO3, _MI3, _FA3, _SOL3
15 DATA PAUSE, _MI3, _FA3 ,_SOL3 ,_SOL3, _LA3, _SOL3 ,_FA3 ,_MI3 ,_DO3
16 DATA _SOL3, _LA3, _SOL3, _FA3, _MI3, _DO3, _RE3, _SOL3, _DO3, PAUSE
18 DATA _RE3, _SOL3, _DO3
22 READ NOTA
23 A++
24 FRQ NOTA, 250
26 IF A<NUMERO_NOTE THEN 22
40 GOTO  40
```

```
05 '---- LE 5 NOTE DI INCONTRI RAVVICINATI DEL TERZO TIPO----
10 USBOUT= "SUONO"
14 USBOUT="SOL-"
15 FRQ _SOL2,400
16 USBOUT="LA-"
17 FRQ _LA2,400
18 USBOUT="FA-"
19 FRQ _FA2,400
20 USBOUT="FA-"
22 FRQ _FA1,600
23 USBOUT="DO-"
24 FRQ _DO2,1000
25 USBOUT="PAUSA-"
35 WAITS 1
36 GOTO 14
37 USBOUT= "-END"
40 GOTO  40
```

```
10 '----------SUONO COMPOSTO CICLICO---------
11 USBOUT="SUONO COMPOSTO-"
17 FRQ _SI3,30
19 FRQ _LA3,40
25 USBOUT="PAUSA-"
35 WAITS 1
36 GOTO 17
```

```
05 ' ------SAN MARTINO IN 7 OTTAVA------
07 CONSTANT NUMERO_NOTE=34
10 DATA _DO6, _RE6, _MI6, _DO6, _DO6, _RE6, _MI6, _DO6, _MI6, _FA6, _SOL6
15 DATA PAUSE, _MI6, _FA6 ,_SOL6 ,_SOL6, _LA6, _SOL6 , _FA6 ,_MI6 ,_DO6
16 DATA _SOL6, _LA6, _SOL6, _FA6, _MI6, _DO6, _RE6, _SOL6, _DO6, PAUSE
18 DATA _RE6, _SOL6, _DO6
22 READ NOTA
23 A++
24 FRQ NOTA, 250
26 IF A<NUMERO_NOTE THEN 22
40 GOTO  40
```

- **GOSUB NumeroDiLinea**
  Trasferisce il controllo alla subroutine specificata alla linea *NumeroDiLinea*.Quando il programma incontra un *RETURN* il controllo torna alla linea successiva al *Gosub*. Possono essere inseriti al massimo 10 gosub/call nidificate.

  Esempio:
```
10 A=5
20 GOSUB 100
30 A=10
40 GOSUB 100
50 END
100 REM SUBROUTIN
110 B=A/2+150/3
120 RETURN
```

- **GOTO NumeroDiLinea**
  Effettua un salto incondizionato alla linea NumeroDiLinea.Il goto effettuato sulla stessa istruzione equivale all'istruzione END.

  Esempio:
```
20 A=20
30 B=A*2
40 GOTO 40
```

- **IF** *espressione* **THEN** *NumeroDiLinea1* **[ELSE** *NumeroDiLinea2***]**
- **IF** *condizione* **THEN**
  codice
  ENDIF
- **IF** condizione **THEN**
  codice
  ELSE
  codice
  ENDIF

Trasferisce il controllo alla linea *NumeroDiLinea1* se l'*espressione* è vera altrimenti alla linea successiva. Se è presente l'else e se l' *espressione* è falsa il controllo viene trasferito al numero di linea *NumeroDiLinea2.* Se nella condizione non compaiono operatori di confronto la condizione è ritenuta vera se l'espressione è diversa da 0, falsa se è uguale a zero.

Esempio:

```
10 A=50
20 A=A+1
30 IF A = 70 THEN 40 ELSE 50
40 GOTO 60
50 GOTO 20
60 END
```

Esempio:

```
PRAGMA NO_NUM_LINE
SUB IF_MULTILINEA
        A=5
        IF A = 3  THEN
            PRINT "A=3"
            B++
        ELSE
            PRINT "A<>3"
            B=-1
        ENDIF

        IF B>0 THEN
            PRINT "INCREMENTATA B"
        ENDIF
        END
SUBEND
```

Output PC:A<>3

Esempio:

```
PRAGMA NO_NUM_LINE

SUB IF_MULTILINEA_NIDIFICATO
        C=0
        FOR A=1 TO 6
                IF A=5  THEN  ' IF MULTILINEA
                        B++
                        IF B=1 THEN
                                D++
                                IF G=4 THEN
                                        H=0
                                ELSE
                                        H=1
                                ENDIF
                                IF D=1 THEN PRINT "CIAO " ' IF MONOLINEA
                                J--
                        ELSE
                                E++
                        ENDIF
                        J+=2
                ELSE
                        C--
                ENDIF
                J+=3
        NEXT A
       PRINT "A="& A & " B="& B & " C="& C & " D="& D & " H="&H
        END

SUBEND
```

Output PC:CIAO A=7 B=1 C=-5 D=1 H=1

- **INCLUDE FILE**
  Il compilatore include il file specificato elaborandolo. Il file può contenere solo DEFINE e CONSTAT. Il file deve avere estensione .inc.

Esempio programma eabasic:

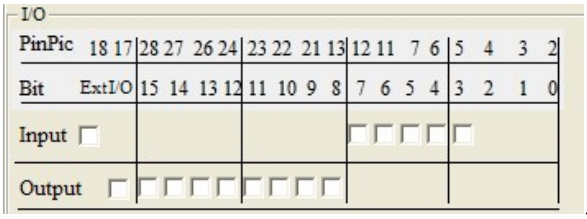
```
05 INCLUDE "MyInclude"
06 A = SETTANTA
07 :CONT
10 A++
15 PRINT A
20 IF  A = 80 THEN :STOP  ELSE :CONT
30 :STOP
35 PRINT CIAO
40 GOTO 40
```

File MyInclude.Inc:

```
DEFINE CIAO = "BYE BYE"
CONSTANT SETTANTA=70
```

- **INP**

Torna il valore degli ingressi (intero a 16 bit) dello stato dei bit.



L'istruzione **SETIO** permette di modificare la tipologia dei pin del PIC che può essere input/output.

Esempio:

```
15 V=INP
30 GOTO  30
```

- **INPBIT(BIT)**

Torna il valore del BIT della porta di ingresso.

Esempio:

```
10 CADS=0
15 SETIO=&HFFFF
20 VAL=INPBIT(A)
25 USBOUT="BIT("&A &")=" &VAL
30 WAITMS 250
35 a++
40 IF a < 16 THEN 20
45 CLR a
50 GOTO  20
```

- **INSTR(STR1,STR2)**

Se la stringa1 è contenuta nella stringa2 torna il numero del carattere della stringa2 dove inizia la stringa1. Se la stringa1 non è contenuta nella stringa2 torna zero.

Esempio:

```
10 A="CIAO COME FORUM!"
11 B="COME"
15 c=INSTR(b,a)
16 USBOUT="POS="& c
40 GOTO  40
```

Output PC:

```
POS=6
```

- **INT(EspressioneNumerica)**
  Torna il valore numerico intero della variabile. Il valore viene troncato.

  Nel primo esempio la variabile B viene definita come float (istruzione 20) ma poi viene convertita in intera e la variabile B a fine programma assume il valore di 25.

  Esempio:
```
10 A=25.2
20 B=A
30 B=INT(A)
40 GOTO 40
```

  Esempio:
```
5  'TEST INT CON OUTPUT SU PC
10 A=25.2
15 B=A
20 USBOUT=b &"<<->>"
25 B=INT(A)
30 USBOUT=b &""
35 GOTO  35
```

  Output PC:
  25.2<<->>25

- **IOMODE(BIT) = MODE**
  Setta l'i/o del singolo BIT alla modalità MODE (MODE : INPUT=1, OUTPUT=0).

  Esempio:
```
IOMODE(14)=INPUT
```

- **LABEL**

Permette l'assegnazione di una LABEL ad un numero di linea.La label risente del maiuscolo/minuscolo. La sintassi è: **:LABEL [ISTRUZIONE]**
Le istruzioni che possono far riferimento alle LABEL sono : goto, if ,gosub.

Esempio:

```
05 :LAB0 B=37
10 IF B= 37 THEN :LAB1
20 B --
30 :LAB1
35  B=0
40 GOTO :LAB0
```

- **LCDCLEAR**

Cancella la schermata del DISPLAY.

- **LCDINIT**

Inizializza il DISPLAY. Questa operazione è gestita automaticamente dal PPTEA. E' utile nel caso in cui si voglia spegnere il display per poi nuovamente alimentarlo.

- **LCDPOS=Expr (Riga*16 + Colonna)**

Posiziona il cursore del Display alla riga e alla colonna specificata.

ATTENZIONE: se il bit 16 del SETIO è posto ad uno la posizione del display è gestita dall'utente.

- **LCDWRITE=Stringa**

Invia la stringa la stringa sul puntatore del Display.

Esempio:

```
10 LCDCLEAR
20 LCDPOS=&H13
25 LCDWRITE="PPTEA-BOLLE"
45 LCDPOS=&H21
46 a++
47 b="V="&a
48 LCDWRITE=b
49 WAITMS 250
50 GOTO 45
```

Output Display:

" PPTEA-BOLLE "
"V=x " Dove il valore di x viene incrementato.

- **LEFT(VAR, EXPR)**

Torna la sottostringa sinistra di VAR lunga EXPR.

Esempio:

```
10 A="CIAO FORUM "
20 c=LEN(a)
25 b=LEFT(a,c)
30 USBOUT=b &"-"
35 c--
40 IF c >0 THEN 25
45 GOTO  45
```

Output PC:
CIAO FORUM-CIAO FORU-CIAO FOR-CIAO FO-CIAO F-CIAO -CIAO-CIA-CI-C-

- **LEN(STRINGA)**
  Ritorna la lunghezza della stringa.

Esempio:

```
11 B="CIAO COME VA?"
20 C=LEN(B)
22 USBOUT="STRINGA LUNGA:"& C
35 GOTO  35
```

Output PC:
STRINGA LUNGA:13

- **LOG(EspressioneNumerica)**
  Ritorna il logaritmo naturale dell'espressione numerica. Il logaritmo è calcolato mediante la serie di Taylor ed ha dei **limiti -1 < x < 1**.

Esempio:

```
10 FOR I=0.0 TO 1.0 STEP 0.1
20   PRINT "I="& I & "LOG=" & LOG(I) & ","
30 NEXT I
35 GOTO  35
```

Output PC:
I=0.100000LOG=-2.083879,
I=0.200000LOG=-1.568137,
I=0.300000LOG=-1.196035,
I=0.400000LOG=-0.914946,
I=0.500000LOG=-0.692967,
I=0.600000LOG=-0.510809,
I=0.700000LOG=-0.356674,
I=0.800000LOG=-0.223143,
I=0.900000LOG=-0.105360,

- **MATH_PRECISION=EXPR**
  Setta la precisione delle funzioni matematiche sin, cos, tan, atan, log, sqr. Il valore di default è 10 e rappresenta l'ordine della serie di Taylor. Più il valore è alto e più i valori sono precisi.

Esempio:

```
10 MATH_PRECISION=2
11 ANG_RAD= 90 * PI/180
15 B=SIN(ANG_RAD)
20 USBOUT="RIS="&B
30 MATH_PRECISION=20
35 B=SIN(ANG_RAD)
40 USBOUT=" RIS="&B
50 GOTO 50
```

Output PC: *RIS=0.925 RIS=1.0*

ATTENZIONE: Per le funzioni Sin, Cos e Tan viene effettuato il fattoriale e quindi il valore massimo per questo tipo di operazione è 85 e non ha senso andare oltre. Per quanto riguarda le altre funzioni , come ad esempio la ATAN, si può andare oltre e ottenere valori sempre più precisi.

Per le funzioni ASIN e ACOS, visto che nella serie di Taylor si calcola un fattoriale con un valore doppio dell'indice della serie il valore di MATH_PRECISION non può passare il valore 18.

- **MEMORY**
  Torna i byte di memoria libera del PPTEA.

ESEMPIO:

```
10 USBOUT=MEMORY & "" ' Risultato 512
20 END
```

In questo esempio torna il valore (massimo) 512 byte, perchè non sono presenti Variabili e la memoria ram del PPTEA non viene utilizzata.

ESEMPIO:

```
05 A=A+1
10 USBOUT=MEMORY & "" ' Risultato 507 -> 512-5
20 END
```

In questo esempio torna il valore 507 byte,perchè viene utilizzata una sola variabile che occupa 5 byte (Mem=512-NUMVARIABILI*5).

ESEMPIO:

```
05 GOSUB :VIS_MEM
10 a="Ciao come va?"
15 GOSUB :VIS_MEM
20 A=""
25 GOSUB :VIS_MEM
100 GOTO 100
1000 :VIS_MEM ' Soubroutine che visualizza la memoria occupata
1100 PRINT "MEM=" & MEMORY
1200 RETURN
```

Il PPTEA risponde in questo modo:

MEM=502
MEM=488
MEM=501

Si può vedere che la memoria nel corso del programma viene allocata (a="Ciao come va?") e successivamente liberata (a="").

ESEMPIO:

```
10 DIM B(20) AS WORD
20 USBOUT=MEMORY &""
30 END
```

Il PPTEA risponde con 467.

ESEMPIO:

```
10 DIM B(20) AS BYTE
20 USBOUT=MEMORY &""
30 END
```

Il PPTEA risponde con 487.

- **MID(VAR, EXPR1, EXPR2)**
  Torna la sottostringa di var lunga EXPR2 che inizia dal carattere EXPR1.

Esempio:

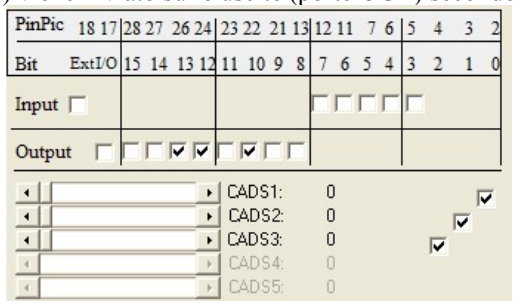
```
10 A="CIAO FORUM!"
15 c=LEN(a)
20 b=MID(a,1,c)
25 USBOUT=b &"-"
30 c--
35 IF c >0 THEN 20
40 GOTO  40
```

Output PC:
CIAO FORUM!-CIAO FORUM-CIAO FORU-CIAO FOR-CIAO FO-CIAO F-CIAO -CIAO-CIA-CI-C-

- **NEXT**
  Chiusura del ciclo FOR (Vedi FOR)

- **OTHER**
  Costrutto dello statement SWITCH. Esegue il codice sottostante se non è verificata l'espressione dei case precedenti.

- **OUT = EXPR**
  Il valore EXPR (intero a 16 bit) viene inviato sulle uscite (porte OUT) secondo la seguente mappatura:



L'istruzione **SETIO** permette di modificare la tipologia dei pin del PIC che può essere input/output.

Esempio:

```
10  OUT=&H3400
20  END
```

- **ON VAR GOTO NLINE1 [,NLINE2,…]**
  Effettua un salto alle linee specificate a seconda del valore della variabile. Salta alla variabile NLINE1 se la variabile vale 1, alla variabile NLINE2 se la variabile vale 2 e così via.
  Se non sono specificate le linee o se il valore della variabile è minore di zero salta alla linea successiva l'istruzione.
  ……

  Esempio:

```
1 Y=0
5 Y++
7 WAITMS 250
20 Y=Y MOD 5
30 ON Y  GOTO 40,  50,  60, 70
32 PRINT "altro"
35 GOTO 5
40 PRINT "1"
45 GOTO 5
50 PRINT "2"
55 GOTO 5
60 :CIAO PRINT "3"
65 GOTO 5
70 PRINT "4"
75 GOTO 5
```

  Output: 1 ,2 ,3, 4, altro, 1 , 2, 3, 4, altro,….

- **OUTBIT(BIT) = EXPR**
  Setta il singolo BIT al valore EXPR (viene settato 1 se EXPR è diverso da zero) sulle porte di uscita.

  Esempio:

```
10 CADS=0
15 SETIO=0
20 V=a MOD 2
25 OUTBIT(A) =V
30 USBOUT="BIT("&A &")=" &V
35 WAITMS 250
40 a++
45 IF a < 16 THEN 20
50 CLR a
55 GOTO  25
```

- **PI**
  Ritorna il valore di Pigreco. Vedi secondo esempio della funzione COS.

- **POW(Espressione Numerica Intera 1, Espressione Numerica Intera 2)**
  Ritorna il l'elevazione a potenza di EN1 alla EN2. Sono accettati solo valori interi.

Esempio:

```
10 A=0
15 B=POW(2,A)
20 USBOUT=B &","
22 A++
25 IF a <= 16  THEN 15
35 GOTO  35
```

Output PC:

1,2,4,8,16,32,64,128,256,512,1024,2048,4096,8192,16384,32768,65536,

- **PRAGMA OPZIONE**

Le possibili opzioni sono:
EXTERNAL_EEPROM o EXTENDED2K_EEPROM -> il codice viene inserito nella eeprom 24C16
EXTENDED4K_EEPROM -> il codice viene inserito nella eeprom 24C32
EXTENDED8K_EEPROM -> il codice viene inserito nella eeprom 24C64
EXTENDED16K_EEPROM -> il codice viene inserito nella eeprom 24C128
EXTENDED32K_EEPROM -> il codice viene inserito nella eeprom 24C256
INTERNAL_EEPROM -> il codice va compilato nella eeprom interna
NO_NUM_LINE -> Istruzione inserita come prima riga del codice che permette l'eliminazione dei numeri di linea.Per effettuare i salti vanno utilizzate esclusivamente le label.

Esempio INTERNAL EEPROM:

```
10 PRAGMA INTERNAL_EEPROM 'codice nella eeprom del pic
15 A=0
20 A++
….
```

Esempio codice senza numeri di linea:

```
PRAGMA NO_NUM_LINE
 A=ASC("A")
 FINE=ASC("Z")
:MAIN
 B=CHR(A)
 C=ASC(B)
 USBOUT= B & "=" & C &","
 A++
 WAITMS 250
 IF A<=FINE THEN :MAIN
 END
```

- **PRINT EXPR**
  Visualizza l'espressione in formato stringa (inviando un a capo a fine linea) inviandola sulla porta USB del PC. E' una macro associata alla USBOUT.

Esempio:

```
15 A=45
20 PRINT "VALORE DI A ="&A
30 END
```

- **PWMDC1=EXPR**
  Modifica il Duty Cycle del PWM 1. Exp va da 0 a 100 e rappresenta la percentuale del Duty Cycle.
  Il bit15 (pin 28) è l'uscita di default del PWM1. Il PWM1 parte quando viene eseguita questa istruzione. Un Duty Cycle =0 blocca il PWM (ed azzera il valore del TIMER) mentre un Duty Cycle=100 mette a positivo i bit associati al PWM.

  Esempio:
  ```
  50  PWMDC1=50
  60  goto 60
  ```

  Il pin 28(bit15) inizia a fare un polling alto/basso con con un duty cycle del 50% (ponendo un led ed una resistenza sul pin 28 (bit 15) lo vedremo lampeggiare nonostante il codice sia bloccato sulla linea 60).

- **PWMDC2=EXPR**
  Modifica il Duty Cycle del PWM 2. Exp va da 0 a 100 e rappresenta la percentuale del Duty Cycle.
  Il bit14 è l'uscita di default del PWM2. Il PWM2 parte quando viene eseguita questa istruzione. Un Duty Cycle =0 blocca il PWM2 mentre un Duty Cycle=100 mette a positivo i bit associati al PWM2.

- **PWMDC3=EXPR**
  Modifica il Duty Cycle del PWM 3. Exp va da 0 a 100 e rappresenta la percentuale del Duty Cycle.
  Il bit13 è l'uscita di default del PWM3. Il PWM3 parte quando viene eseguita questa istruzione. Un Duty Cycle =0 blocca il PWM3 mentre un Duty Cycle=100 mette a positivo i bit associati al PWM3.

- **PWMFQ1=VAL_PERIOD**
  Modifica il periodo del PWM. VAL_PERIOD può assumere valori da 0 (bassa frequenza) a 7(alta frequenza). Per default il valore è posto uguale ad 0. Solo nel primo PWM è possibile modificare la frequenza di oscillazione. Questa è la corrispondenza dei valori:

  VAL_PERIOD = 0 T= 1.67 s
  VAL_PERIOD = 1 T= 0.83 s
  VAL_PERIOD = 2 T= 419 ms
  VAL_PERIOD = 3 T= 209 ms
  VAL_PERIOD = 4 T= 104 ms
  VAL_PERIOD = 5 T= 52 ms
  VAL_PERIOD = 6 T= 26 ms
  VAL_PERIOD = 7 T= 13 ms

  Esempio:
  ```
  10  PWMDC=50
  20  PWMFQ=3
  30  GOTO  30
  ```
  Il pin 28 (bit15) inizia a fare un polling alto/basso molto veloce.

- **PWMIO1=BIT**
  Seleziona i bit del primo PWM. Il default è &H8000 relativo al bit 15.
  Attenzione:occorre assicurarsi che i BIT associati al PWM siano impostati come output.

  Esempio:
  ```
  05   SETIO=&H00FE
  60   PWMFQ=3
  70   PWMIO=&H8001
  80   PWMDC=50
  90   GOTO 90
  ```
  Il bit15 (pin 28) ed il bit0 (pin 2) vengono associati al PWM e lampeggiano insieme.

- **PWMIO2=BIT**
  Seleziona i bit del secondo PWM. Il default è &H4000 relativo al bit 14.
  Attenzione:occorre assicurarsi che i BIT associati al PWM siano impostati come output.

- **PWMIO3=BIT**
  Seleziona i bit del terzo PWM. Il default è &H2000 relativo al bit 13.
  Attenzione:occorre assicurarsi che i BIT associati al PWM siano impostati come output.

- **RANDOMIZE (SEME)**

  Inizializza il generatore di numeri casuali.

  Esempio:
```
10 RANDOMIZE(15)
15 NumeroCasuale=Rnd '
20 PRINT NumeroCasuale&"-"
25 WAITMS 100
30 GOTO  15
```
  Output PC: 3612-30669-3698-14235-18968-20217-….

- **RASPRECEIVE**
  Riceve una stringa dal RaspBerryPI. Evitare di oltrepassare i 50 caratteri di ricezione dati contemporanei.

- **RASPSEND=EXPR**
  Invia un byte o una stringa al modulo SP1.

- **READ VAR1 [, VAR2…..]**
  Questo comando permette di leggere in modo ordinato e progressivo i dati memorizzati mediante il comando DATA. L'istruzione READ deve essere eseguita dopo l'istruzione DATA. E' possibile utilizzare l'istruzione READ in ordine sparso nel codice . (Vedi DATA) .L'istruzione RESTORE permette di effettuare dei salti sui dati da caricare.

- **REEPROM**
  Ritorna il contenuto della cella eeprom settata con il comando CELL.

- **REM**
  Commento ad inizio linea.

  Esempio:
```
10 REM PROGRAMMA DI ESEMPIO
.....
```
- **REPEAT …. UNTIL EXPR**
  Il ciclo REPEAT–UNTIL viene ripetuto fintanto che la condizione Expr sia vera. Se l'espressione è falsa viene eseguita l'istruzione successiva all'UNTIL. Si possono utilizzare fino a 127 cicli nidificati.

  Esempio:
```
05 USBOUT="INIZIO PROG"
06 a=0
07 REPEAT
20     A++
25     WAITMS 250
26     USBOUT="A="&A &"-"
30 UNTIL a<10
40 USBOUT ="FINE PROG"
50 GOTO 50
```
  Output PC:INIZIO PROGA=1-A=2-A=3-A=4-A=5-A=6-A=7-A=8-A=9-A=10-FINE PROG

Esempio di due cicli nidificati.

Esempio:

```
05 USBOUT="INIZIO PROG"
07 REPEAT
08   b++
09   CLR a
10   REPEAT
20     A++
25   WAITMS 250
26   USBOUT="A="&A &" B="& B & "-"
30   UNTIL a<3
35 UNTIL b<5
40 USBOUT ="FINE PROG"
50 GOTO 50
```

Output PC:

INIZIO PROGA=1 B=1-A=2 B=1-A=3 B=1-A=1 B=2-A=2 B=2-A=3 B=2-A=1 B=3-A=2 B=3-A=3 B=3-A=1 B=4-A=2 B=4-A=3 B=4-A=1 B=5-A=2 B=5-A=3 B=5-FINE PROG

- **RESCOUNTER**
  Permette al PPTEA di utilizzare il PWM3, cioè resetta la funzione contatore.

- **RESTORE [NUM_LINEA]**

Questo comando permette di assegnare o riassegnare il punto di inizio dei dati da caricare mediante l'istruzione READ. Se il RESTORE non ha specificato il numero di linea inizia a prendere i dati dalla prima costante dell'istruzione DATA. (Vedi DATA,READ).

Esempio:

```
10 DATA 25,40,47
11 DATA 27,41,49
12 DATA 28,42,43
15 READ A,B
20 USBOUT=A & " " &B &"-"
25 RESTORE 12
30 READ C,D
35 USBOUT=C & " " & D & "-"
40 GOTO  40
```

Output PC:
25 40-28 42-

- **RETURN**

Ritorna da una subroutine (linea successiva alla chiamata effettuata dal GOSUB/CALL). Vedi Esempio GOSUB.

- **RIGHT(VAR, EXPR)**

Torna la sottostringa destra di VAR lunga EXPR.

Esempio:

```
10 A="CIAO FORUM"
20 c=LEN(a)
25 b=RIGHT(a,c)
30 USBOUT=b &"-"
35 c--
40 IF c >0 THEN 25
45 GOTO  45
```

Output PC:
CIAO FORUM- IAO FORUM-AO FORUM-O FORUM- FORUM-FORUM-ORUM-RUM-UM-M-

- **RND**

Torna un numero casuale da 0 a 32767.

Esempio:

```
10 A=Rnd
20 USBOUT=A&"-"
30 WAITMS 250
55 GOTO  10
```

Output PC:
30393-29006-31527-532-21029-31402-14131-11408-2513-23878-11647-9804-27069-21794-18955-19272.....

- **SERIALCLOSE**

Chiude la comunicazione con la serial. Questo comando è utile per passare ad utilizzare la modalità WIFI o per resettare la porta seriale.

- **SERIALINP**

Receve un byte dalla seriale.

Esempio:

```
05 SETIO=&H400FF
25 V=SERIALINP
26 WAITMS 5
27 IF v=-1 THEN 25
30 USBOUT=v&"-"
40 GOTO 25
```

Visualizza i byte che arrivano dalla seriale. Questo esempio non è in grado di gestire un flusso continuo di dati.

- **SERIALINPSTR**

Riceve una stringa dalla seriale. Evitare di oltrepassare i 50 caratteri di ricezione dati contemporanei.

Esempio visualizzazione stringa ricevuta da seriale:

```
05 SETIO=&H400FF
15 WAITMS 250
25 V=SERIALINPSTR
26 IF v="" THEN 15
30 USBOUT=v
40 GOTO 15
```

- **SERIALINPSTR**

Riceve una stringa dalla seriale. Evitare di oltrepassare i 50 caratteri di ricezione dati contemporanei.

Esempio visualizzazione stringa ricevuta da seriale:

```
05 SETIO=&H400FF
15 WAITMS 250
25 V=SERIALINPSTR
26 IF v="" THEN 15
30 USBOUT=v
40 GOTO 15
```

- **SERIALOUT=EXPR**

Invia un byte o una stringa sulla seriale.

Esempio invio stringa su seriale:

```
10 SETIO=&H400FF
20 a="Ciao ciao come va?"
30 SERIALOUT=A
40 GOTO 40
```

Esempio invio byte su seriale:

```
10 SETIO=&H400FF
20 USBOUT= A & "-"
25 SERIALOUT=A
30 WAITMS 250
35 A++
40 GOTO 10
```

- **SERIALSPEED=SPEED**

Permette di settare la velocità di trasmissione della seriale. Utilizzare una delle costanti predefinite per impostare il relativo baud-rate:

SPEED_57600 , SPEED_38400, SPEED_19200, SPEED_9600, SPEED_4800 .

- **SERIALRESTART**

Riavvia la comunicazione con la seriale. Questo comando è necessario per effettuare un reset della seriale che permette lo sblocco dei dati in ricezione.

- **SET VARIABILE [, VARIABILE, VARIABILE,…]**
  Setta ad uno il contenuto della variabile o dell'elemento del vettore.

  Esempio:

```
05 B=37
10 A=25
12 USBOUT= a &"-" & b &"-"
15 SET A,B
16 USBOUT= a &"-"& b &"-"
20 GOTO 20
```

  Output PC:
  25-37-1-1-
  ATTENZIONE :L'istruzione SET OUT (oppure outbit) non ha effetto sull'I/O.

- **SETCOUNTER**
  Setta nel PPTEA la modalità contatore e contemporaneamente lo azzera. Se il contatore è già settato, viene azzerato il contatore.

- **SETDATE=STRING**
  Setta l'orario dell'orologio esterno. La stringa deve avere il seguente formato:
  "GG/MM/AA HH:MM:SS D"
  (GG=giorno, GG=mese, AA=anno, HH=ora, MM=minuti,SS=secondi,
  D= 1 se Lunedi,2 se Martedi, 3 se mercoledi, 4 se giovedi, 5 se venerdi, 6 se Sabato,7 se Domenica)

  Devono essere presenti sempre 2 cifre per ogni dato se il dato è composto da una sola cifra va messo lo zero.

  Esempio:

```
10 ORARIO="01/05/11 16:12:05"
20 SETDATE=ORARIO
…….
```

  Viene settata la data del primo maggio del 2011 alle ore sedici, dodici minuti e cinque secondi.

- **SETIO = Expr**

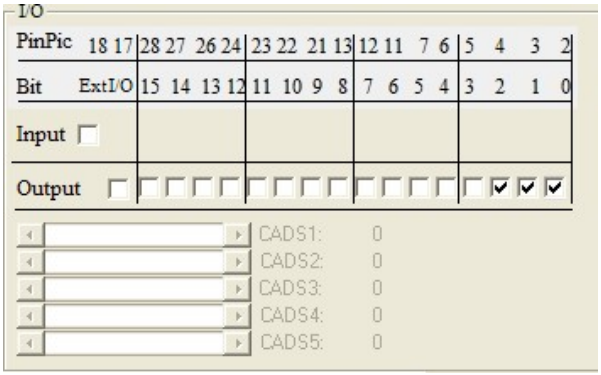
  Configura o modifica l'I/O del PPTEA. I 16 bit di I/O possono essere configurati come ingresso (bit alto) o come uscita (bit basso). Per Default il valore è settato con SETIO=&HFF, cioè i primi 8 bit sono ingressi ed i restanti sono uscite. Il default del PPTEA è configurato per avere ingressi anologici , ingressi digitali ed uscite digitali in particolare sono presenti 3 convertitori analogico digitali (BIT0-BIT2), 5 ingressi digitali (BIT3-BIT7) e 7 uscite digitali (BIT8-BIT15), quindi all'avvio il PPTEA presenta questo I/O:

  BIT0: Ingresso A/D
  BIT1: Ingresso A/D
  BIT2: Ingresso A/D
  BIT3: Ingresso Digitale
  BIT4: Ingresso Digitale
  BIT5: Ingresso Digitale
  BIT6: Ingresso Digitale
  BIT7: Ingresso Digitale
  BIT8 : Uscita Digitale
  BIT9 : Uscita Digitale
  BIT10: Uscita Digitale
  BIT11: Uscita Digitale
  BIT12: Uscita Digitale
  BIT13: Uscita Digitale
  BIT14: Uscita Digitale
  BIT15: Uscita Digitale

Il default può essere modificato in qualsiasi momento. Ad esempio per utilizzare tutte e 16 le porte in digitale occorre settare a zero il numero di convertitori (CADS=0). I bit da 9 a 15 sono connessi a delle resistenze di pull-up che possono essere sfruttati se utilizzati come input. Se viene utilizzata la eeprom esterna o il Real Time Clock i bit 9 e 10 (pin 21 e pin 22) non possono essere utilizzati.
Questa è la rappresentazione di tutti i bit relativi al SETIO:

Valore	Seriale	DISPLAY		INGRESSO USCITE PPTEA BYTE ALTO								INGRESSO USCITE PPTEA BYTE BASSO							
	Bit18	Bit17	Bit16	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Val Dec	262144	131072	65536	32768	16384	8192	4096	2048	1024	512	256	128	64	32	16	8	4	2	1
Val Esa	H40000	H20000	H10000	H8000	H4000	H2000	H1000	H800	H400	H200	H100	H80	H40	H20	H10	H08	H04	H02	H01
0	WIFI	(***)	(*)	OUT	OUT	OUT	OUT	OUT	OUT	OUT	OUT	OUT	OUT	OUT	OUT	OUT	OUT	OUT	OUT
1	RS232	(****)	(**)	INP	INP	INP	INP	INP	INP	INP	INP	INP	INP	INP	INP	INP	INP	INP	INP

(*)	Display in modalità GOTO controllata dal PPTEA
(**)	Display in modalità GOTOcontrollata dall'utente
(***)	Controllo display a 6 fili
(****)	Controllo display a 2 fili



Esempio:
10 SETIO=0
15 CADS=0
20 GOTO 20

Nell'esempio vengono utilizzate i bit solo come uscite.

ATTENZIONE: Dalla versione 2.5 è a disposizione il bit 16 per settare la modalità locate display in modo libero:

a. bit 16=0 Display 2x16 o 4x16
b. bit 16=1 Display con locate gestita dall'utente

ATTENZIONE: Dalla versione 2.14 è a disposizione il bit 17 per settare la modalità display a due fili:

c. bit 17=0 Display con 6 fili di collegamento
d. bit 17=1 Display con 2 fili di collegamento

ATTENZIONE: Tutti i pin collegati al display sono settati come output dal PPTEA al solo INIT del dispositivo.

ATTENZIONE: Dalla versione 2.16b è a disposizione il bit 18 per settare la modalità wifi/seriale:

e. bit 18=0 Abilitata la modalità WIFI
f. bit 18=1 Abilitata la modalità RS232

Esempio:
SETIO=&H10000 ‘←Display free
SETIO=&H00000 ’←Standard Display 2x16 o 4x16

SETIO=&H007F ‘←Utilizzo il solo display a 6 fili
SETIO=&H00BF ‘←Utilizzo il solo display a 2 fili
SETIO=&H003F ‘←Utilizzo il display a due file e a 6 fili

SETIO=&H40000 ‘←Utilizzo la comunicazione seriale

- **SGN(Espressione Numerica)**
  Ritorna il segno del'espressione : +1 se > 0 , 0 se =0 , -1 se <0.

Esempio:

```
10 A=10
15 A--
20 USBOUT=a &","
25 IF SGN(a) <> -1  THEN 15
35 GOTO  35
```

Output PC: *9,8,7,6,5,4,3,2,1,0,-1,*

- **SIN(radiant-expression)**
  Ritorna il seno trigonometrico.

Esempio:

```
10 A=45*PI/180
15 B=SIN(A)
20 C=SIN(A/2)
.....
```

Vedere altro esempio nella funzione COS.

- **SQR(EspressioneNumerica)**
  Ritorna la radice quadrata.

Esempio:

```
10 '-------------------------------------------------
15 'TEST SQR CON OUTPUT SU PC
20 '-------------------------------------------------
25 A=2
30 B=SQR(A)
35 USBOUT=B & ","
40 C=B*B
45 USBOUT=C & ""
50 GOTO  50
```

Output PC:	Output PPTEA
1.414214,2.000000	1.414213,1.999999

ATTENZIONE: QUANDO IL PPTEA EFFETTUA UNA TRASFORMAZIONE IN STRINGA NON E' PRECISO COME IL VALORE CONTENUTO NELLA VARIABILE...COME VISIBILE IN QUESTO ESEMPIO.

- **SUB SUBPROGRAM[(PAR1,PAR2,….)]**

La SUB è la dichiarazione di inizio di un sottoprogramma che termina con l'istruzione **SUBEND**. Si può uscire prematuramente da un sottoprogramma con l'istruzione SUBEXIT. Il numero di parametri debbono essere gli stessi della call. L'eabasic del PPTEA ha solo variabili globali e le variabili parametriche assumono tale valore alla loro chiamata. Per far tornare un valore alla sub basta semplicemente assegnare il valore al nome della variabile che si chiama come la subroutine. Vedere gli esempi nella istruzione CALL.

Programma di Temperatura & Orario Display:

```
PRAGMA NO_NUM_LINE
'-------------- COSTANTI ----------------------
CONSTANT POS_GIORNO_DISPLAY=&H19
CONSTANT POS_ORA_DISPLAY=&H21
CONSTANT POS_TEMPERATURA_DISPLAY=&H2B
CONSTANT SIMB_GRADI_CENTIGRADI=223
CONSTANT NUM_ELEMENTI_MEDIA=20
'-------------- VETTORE DI MEMORIA ---------
DIM MEM(NUM_ELEMENTI_MEDIA) AS WORD
'------------------ PROGRAMMA ---------------
SUB MAIN
  CALL PRESENTAZIONE
  WHILE TRUE
    CALL RILEVA_TEMPERATURA
    CALL MEDIA_DELLA_TEMPERATURA
    CALL VISUALIZZA_SUL_DISPLAY
    WAITS 1
  LOOP
SUBEND
'--------------INIZIO SUBROUTINE ---------------
'-----------PRESENTAZIONE INIZIALE -------
SUB PRESENTAZIONE
  LCDWRITE="PPTEA"
SUBEND
'---RILEVA LA TEMPERATURA SUL CADS1---
SUB RILEVA_TEMPERATURA
  TEMP=CADS1*CAD_TO_TEMP
  IND=(INDICE MOD NUM_ELEMENTI_MEDIA) +1
  MEM(IND)=TEMP*10
  INDICE++
SUBEND
'---EFFETTUA LA MEDIA DELLA TEMPERATURA---
SUB MEDIA_DELLA_TEMPERATURA
  CLR G
  T=0.0
  :ELAB G++
  T+=MEM(G)
  IF G>=NUM_ELEMENTI_MEDIA THEN :SAL_TEMP
  IF G<indice THEN :ELAB
  :SAL_TEMP T/=G
  T/=10
SUBEND
'---VISUALIZZA I DATI SUL DISPLAY---
SUB VISUALIZZA_SUL_DISPLAY
  LCDPOS=POS_GIORNO_DISPLAY
  LCDWRITE=LEFT(DATE, 8) ' GIORNO
  LCDPOS=POS_ORA_DISPLAY
  LCDWRITE=MID(DATE, 10,8) ' ORA
  LCDPOS=POS_TEMPERATURA_DISPLAY
  LCDWRITE=LEFT(T&"",4) & CHR(SIMB_GRADI_CENTIGRADI) & "C"
SUBEND
'--------------FINE SUBROUTINE ---------------
```

Esempio Sub Nidificate:

```
10 A=25
15 B=CALL PIPPO(A)
20 PRINT B
25 END

170 SUB PIPPO(PAR1)
175       C=PAR1*3
178       PIPPO=CALL PEPPE(C)
180 SUBEND

200 SUB PEPPE(PAR2)
210        PEPPE=PAR2*2
      220 SUBEND
```

Esempio SUBEXIT:

```
PRAGMA ADVANCED
          A=25
          B= CALL CONTA(A)
          PRINT "VALORE TORNATO="& B
END

SUB CONTA(PAR1)
  WHILE TRUE
        PAR1++
       CONTA=PAR1
         PRINT PAR1&" "
        WAITMS 100
        IF PAR1=35 THEN SUBEXIT
 LOOP
SUBEND
```

- **SUBEND**
  Termina la subroutine.

- **SUBEXIT**
  Permette l'uscita prematura da una subroutine.

- **SWITCH EXPR**
  A seconda del valore che assume Expr *viene eseguito il relativo **CASE**. Il blocco **CASE** termina con **CASEEND**. Se **CASEEND** non è presente nel blocco CASE, viene eseguito anche il CASE successivo. Se nessun **CASE** è valido , viene eseguito il blocco **OTHER**.*

```
PRAGMA NO_NUM_LINE ' SI OMETTONO I NUMERI DI LINEA
Y=0
WHILE TRUE ' LOOP INFINITO
      Y++
     WAITMS 250
     SWITCH Y ' CONDIZIONE DI CASE
          CASE 1 ' ESEGUE IL CASE SE Y=1
                A++
               PRINT "1"
               CASEEND ' INTERROMPE IL CASE 1

          CASE 2 ' ESEGUE IL CASE SE Y=2
                A++
               PRINT "2"
               CASEEND

          OTHER ' ESEGUE OTHER SE Y<> 1 e Y<>2
                A--
               PRINT "ALTRO"

     SWITCHEND
     PRINT "SWITCHEND"
LOOP
END 'FINE PROGRAMMA
```

Output PC:

```
1
SWITCHEND
2
SWITCHEND
ALTRO
SWITCHEND
ALTRO
SWITCHEND
ALTRO
SWITCHEND
ALTRO
SWITCHEND
ALTRO
SWITCHEND
ALTRO
SWITCHEND
ALTRO
SWITCHEND
.....
```

- **SWITCHEND**
  Costrutto dello statement SWITCH. Termina il blocco dello SWITCH.

- **TAN(Espressione In Radianti)**
  Ritorna la tangente trigonometrica .

Esempio:

```
10 A=45*PI/180
15 B=TAN(A)              'B=1
20 C=ATAN(B)*180/PI   'C=45
.....
```

Esempio:

```
10
'-----------------------------------------------------
15 'TEST TAN/ATAN CON OUTPUT SU PC
20
'-----------------------------------------------------
25 A=45*PI/180
30 B=TAN(A)
35 USBOUT="TAN(" & A &")="&B
40 C=ATAN(B)*180/PI
45 USBOUT=", ATAN(" & B &")="&C
50 GOTO  50
```

Output PC:
Tan(0.785)=1.0, Atan(1.0)=45.0

- **TIMER**
  Torna il numero di secondi trascorsi dall'accensione del PPTEA. La parte intera rappresenta i secondi mentre la parte decimale ha la precisione dei microsecondi. Il timer può essere resettato mediante il comando PWMDC=0 (o mediante la macro RESET_TIMER) e il suo conteggio ripartirà da 0. Il Timer non è utilizzabile se si usa il PWM1 perchè condividono le stesse risorse, è utilizzabile se si usano i PWM2 e PWM3.

Esempio:

```
150  A=TIMER
205  USBOUT= A & " "
206  WAITMS  100
210  GOTO 150
```

Output PC: 0.400646 0.402592 0.492169 0.576269 0.671323 0.760900 0.845000 0.940032 1.029630 1.113816 1.208934 1.293119 1.402806 1.476081 1.560309 1.655470 1.739698 1.834880 1.924586 2.008792

- **USBINP**

  Torna il valore intero (max 4 byte) ricevuto dal PC sulla porta USB. L'istruzione è non bloccante e se non riceve nessun dato dalla porta USB torna -1 (***ATTENZIONE: dalla versione la 2.15 o inferiore tornava 0***). Se il PIC non è agganciato alla PORTA USB l'istruzione non ha effetto. Per debug può essere utilizzato il pulsante "SEND USB" presente sul PPTEA COMPILER.

  Esempio:

```
10 B=-1 ' modificato dalla 2.16 ..prima era B=0
15 A=USBINP
20 IF A<>B THEN 35
25 WAITMS 50
30 GOTO  15
35 USBOUT="Ricevuto :"&A
40 WAITMS 50
45 GOTO  15
```

  (Viene inserito il valore 78 nel campo "SEND USB" e premuto il pulsante)

  Output PC:

  Ricevuto :78

- **USBINPSTR**

  Torna la stringa ricevuta dal PC sulla porta USB. L'istruzione è non bloccante e se non riceve nessun dato dalla porta USB torna una stringa nulla. Se il PIC non è agganciato alla PORTA USB l'istruzione non ha effetto. La stringa ricevuta può essere al massimo lunga 63 caratteri.

  Esempio:

```
10 a++
20 USBOUT=a & "-"
30 WAITMS 100
31 gg=USBINPSTR
35 IF LEN(gg)<>0 THEN 50
40 GOTO 10
50 USBOUT="USB=" & gg &"-"
60 GOTO 10
```

  Output PC:

  1-2-3-…… (premendo il pulsante "TX STR" visualizza la stringa contenuta nell'area testo "USB=ciao")

- **USBOUT = EXPR**
  Manda il valore dell'espressione sulla porta USB. L'espressione può essere di tipo:intera, a virgola mobile o stringa. Se il PIC non è agganciato alla PORTA USB l'istruzione è bloccante, mentre se è agganciata la porta usb ma la connessione non è aperta l'istruzione non ha effetto.

  ATTENZIONE: Dalla versione 2.5 il comando anche in assenza di connessione USB non è più bloccante!

  Esempio:
```
10 B="CIAO!"
15 USBOUT=B
20 A=50
25 USBOUT=a
30 USBOUT=PI&""
35 GOTO  35
```
  Output PC:

  CIAO! 3.142

  Output PC Byte: -67-73-65-79-33-2-1-51-46-49-52-50

- **VOLUME = EXPR**
  Setta il volume dell'altoparlante dove 0 è il Volume minimo e 100 volume massimo ma l'andamento del volume è esponenziale, suddividiamo il volume in 10 livelli:

  LIVELLO = VALORE

  VOLUME_0 = 0 (VOLUME MINIMO)
  VOLUME_1 = 1
  VOLUME_2 = 2
  VOLUME_3 = 4
  VOLUME_4 = 8
  VOLUME_5 = 16
  VOLUME_6 = 30
  VOLUME_7 = 60
  VOLUME_8 = 80
  VOLUME_9 = 100 (VOLUME MAX)

- **USBOUTB = EXPR**
  Manda un byte sulla porta USB. Se il PIC non è agganciato alla PORTA USB l'istruzione non ha effetto.

  Esempio:
```
10 A=50
15 USBOUTB=a
20 b="A"
5 USBOUT=b
30 GOTO  30
```
  Output PC:

  2A

  Output PC Byte: -50-65

- **WAITMS EXPR**
  Attende i millisecondi specificati nell'espressione.

  Esempio:
```
10 WAITMS 50
.....
```
- **WAITS EXPR**
  Attende i secondi specificati nell'espressione.

Esempio:

```
10 WAITS 1
.............
```

- **WEEPROM=EXPR**

  Scrive il valore dell'espressione nella cella eeprom settata con il comando CELL.

Esempio:

```
10 CELL=255
15 B=REEPROM
20 USBOUT=B&"-"
25 WEEPROM=50
30 C=REEPROM
35 USBOUT=C&"-"
40 WEEPROM=B
45 D=REEPROM
50 USBOUT=D&"-"
55 GOTO  55
```

- **WHILE EXPR … LOOP**

  Il ciclo WHILE-LOOP viene ripetuto fintanto che la condizione Expr sia vera. Se l'espressione è falsa viene eseguita l'istruzione successiva al LOOP. Si possono utilizzare fino a 127 cicli nidificati.
  Attenzione ogni While che viene eseguito deve essere chiuso con il loop, se il loop non viene eseguito è come se si effettuassero While nidificati.

Esempio:

```
05 USBOUT="INIZIO PROG"
07 WHILE a< 10
20   A++
25   'waitms 250
26   USBOUT="A="&A &"-"
30 LOOP
40 USBOUT ="CIAO"
50 GOTO 50
```

Output PC:

INIZIO PROGA=1-A=2-A=3-A=4-A=5-A=6-A=7-A=8-A=9-A=10-CIAO

Esempio di due cicli nidificati.

Esempio:

```
10 USBOUT="INIZIO PROG"
12 WHILE B<3
13   B++
15   CLR A
20   WHILE a<5
25     A++
30     WAITMS 250
35     USBOUT="A="&A &" B="& B
40   LOOP
41 LOOP
45 USBOUT ="FINE PROG"
50 GOTO  50
```

Output PC:

INIZIO PROGA=1 B=1A=2 B=1A=3 B=1A=4 B=1A=5 B=1A=1 B=2A=2 B=2A=3 B=2A=4 B=2A=5 B=2A=1 B=3A=2 B=3A=3 B=3A=4 B=3A=5 B=3FINE PROG

- **WIFIOUT=EXPR (Non disponibile dalla versione PPTEA 4.0)**

Invia un dato (max 4 byte) in radiofrequenza mediante dispositivo WIFI.

Esempio:

```
10 '  INVIA IL BYTE 44 SU WIFI
15 ' SE SI CHIUDE UN CONTATTO (Massa pin 28)
20 ' ACCENDENDO IL LED (Pin 27)
25 OUT=&H6000
30 WAITMS 100
35 OUT=0
40 WAITMS 100
45 OUT=&H4000
50 WAITMS 100
55 OUT=0
60 WAITMS 100
65 SETIO=&H80FF
70 A=INP
75 A=A AND &H8000
80 IF A = 0 THEN 90
85 GOTO  70
90 D=44
95 USBOUT=D & "-"
100 WIFIOUT=D
105 OUT=&H4000 ' Accendo il led (pin 27)
115 WAITMS 250
120 OUT=0      ' Spengo il led (pin 27)
125 GOTO  70
```

- **WIFIINP (Non disponibile dalla versione PPTEA 4.0)**

Torna un dato ricevuto dal dispositivo WIFI. L'istruzione è bloccante. Questa istruzione, a differenza della WIFIINPNW, non necessita del bit 15 e quindi può essere utilizzato dall'eabasic.

Esempio:

```
10 ' RICEVITORE WIFI
11 ' CHIUDE UN CONTATTO (pin 21) PER 250 ms
12 ' SE RICEVE IL BYTE 44 SCRIVENDO SULLA PORTA USB ON SEGUITO DA OFF
13  OUT=512
14 WAITMS 250
15 OUT=0
16 WAITMS 250
17  OUT=512
18 WAITMS 250
19 OUT=0
20 A=WIFIINP
30 IF A = 44 THEN 50
40 GOTO 80
50 'RICEVUTO IL VALORE 127
60 USBOUT = "ON"
65 OUT=512
66 WAITMS 250
```

```
67 OUT=0
68 USBOUT = "OFF"
70 GOTO 20
80 USBOUT = A &""
85 WAITMS 100
90 GOTO 20 20 USBOUT=A&"-"
30 WAITMS 250
55 GOTO  10
```

Output PC:

ONOFF

Il programma rimane in attesa sulla istruzione 20, se riceve il segnale con il byte 44 manda ad 1 il pin 21 per 100 millisecondi.

- **WIFIINPNW (Non disponibile dalla versione PPTEA 4.0)**

Torna un byte in arrivo dal dispositivo WIFI. L'istruzione non è bloccante. Se non arriva nessun segnale torna il valore -1. La gestione dell'I/O sul bit 15 viene gestito automaticamente , il PPTEA definisce il bit 15 come ingresso.

ATTENZIONE:Per utilizzare questa istruzione occorre cortocircuitare il pin 18 con il pin 28 perdendo il bit15 del PPTEA.

- **WIFIDISP=EXPR (Non disponibile dalla versione PPTEA 4.0)**

Setta il dispositivo WIFI. Il valore di default è uguale ad 1.

- **WIFIDISPINP (Non disponibile dalla versione PPTEA 4.0)**

Torna l'identificativo dell'ultimo dispositivo WIFI da cui si è ricevuto un messaggio.

- **WEBRECEIVE**

Riceve una stringa dal modulo SP1. Evitare di oltrepassare i 50 caratteri di ricezione dati contemporanei.

- **WEBSEND=EXPR**

Invia un byte o una stringa al modulo SP1.

EABASIC ADVANCED

La versione avanzata del PPTEA nasce per sopperire alle problematiche relative la scrittura di grandi codici dove occorre rimappare in continuazione i numeri di linea. Inoltre sono state aumentate il numero di variabili (max 66) ed aumentata la meroria di codice (max 32kB). Chi ha le capacità può fare a meno dei numeri di linea e fare riferimento alle Label per effettuare i salti. La prima istruzione ***PRAGMA NO_NUM_LINE*** fa capire al compilatore che non si vogliono utilizzare i numeri di linea. Per velocizzare la stesura del codice il compilatore mette a disposizione due acceleratori CTRL-L che crea una nuova label :Lxxx (si incrementa ad ogni CTRL-L) e CTRL-G che replica l'ultima label creata. Vediamo come si può trasformare un codice eliminando i numeri di linea:

```
10 A=ASC("A")
13 FINE=ASC("Z")
15 B=CHR(A)
25 C=ASC(B)
30 USBOUT= B & "=" & C &","
35 A++
37 WAITMS 250
40 IF A<=FINE THEN 15
50 END
```

Utilizzando label numeriche:

```
PRAGMA NO_NUM_LINE
A=ASC("A")
FINE=ASC("Z")
:15 B=CHR(A)
C=ASC(B)
USBOUT= B & "=" & C &","
A++
WAITMS 250
IF A<=FINE THEN :15
END
```

Utilizzando label alfanumeriche:

```
PRAGMA NO_NUM_LINE
A=ASC("A")
FINE=ASC("Z")
:MAIN
B=CHR(A)
C=ASC(B)
USBOUT= B & "=" & C &","
A++
WAITMS 250
IF A<=FINE THEN :MAIN
END
```

WIFI del PPTEA

Questi comandi non sono disponibili dalla versione 4.0 del PPTEA, sono sostituiti da dispositivi che permettono l'utilizzo del protocollo seriale.

Il PPTEA gestisce fino a 255 dispositivi WIFI connessi tra loro. E' possibile implementare :

1. Ricetrasmettitore (AUREL RT-DATA-SAW 433)
2. Ricevitore (AUREL RX BC-NBX)
3. Trasmettitore (AUREL R073A)

Il primo consentirà la ricezione e la trasmissione dei dati, mentre gli altri potranno solo ricevere o trasmettere. Esempio:in un telecomando del cancello si utilizzerà il (3), nel cancello si userà il (2), mentre per applicazioni dove si vuole sia trasmettere che ricevere si utilizzerà (1).
Ogni dispositivo trasmette sulla frequenza di 433 MHz ed avrà bisogno di una antenna (va bene anche un filo di rame lungo 17.5 cm).
Per la ricezione dei dati WIFI si potranno utilizzare funzioni bloccanti e non bloccanti.

I comandi relativi al WIFI sono:

1. WIFIDISP
2. WIFIDISPINP
3. WIFIINP
4. WIFIINPWN
5. WIFIOUT

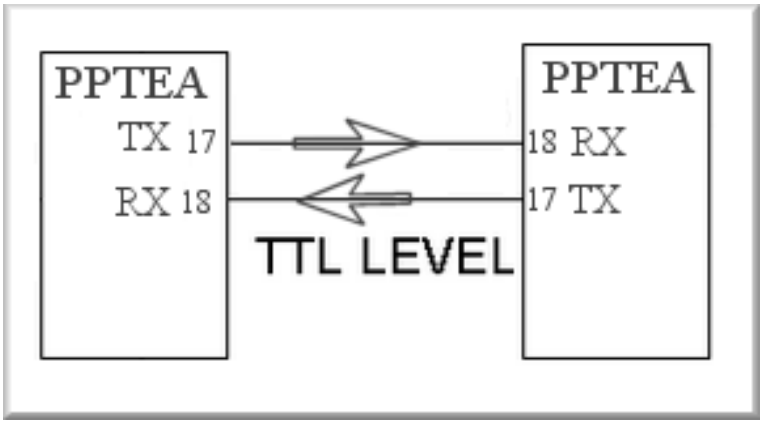
SERIALE RS232

Il PPTEA gestisce la comunicazione seriale con i seguenti parametri di default (baud-rate 9600, 8 bit ,no parity, 1 bit stop) : Velocità 9600, numero bit dati:8, parità :nessuna, Bit Stop :1

Il pin TX del PPTEA è il numero 17 (pin dove il PPTEA trasmette i dati)
Il pin RX del PPTEA è il numero 18 (pin dove il PPTEA riceve i dati)

L'Apertura della porta con i parametri di configurazione avviene al primo comando che utilizza la seriale.

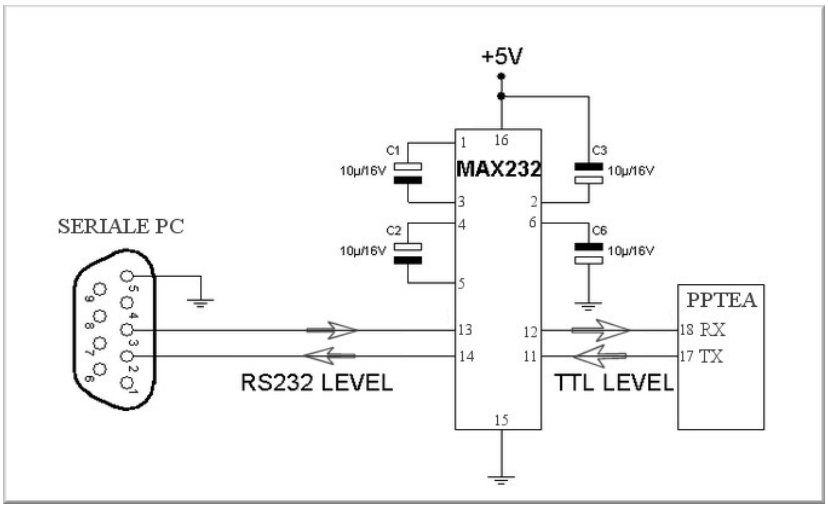
Il PPTEA dispone sui pin 17 e 18 una uscita seriale di tipo TTL. Due PPTEA possono comunicare incrociando i dati TX/RX:



I comandi relativi alla seriale sono:

1. SERIALCLOSE
2. SERIALINP
3. SERIALINPSTR
4. SERIALOUT
5. SERIALSPEED

Se si vuole interfacciare una seriale con livelli RS232 occorre inserire un MAX232 tra il PPEA e il dispositivo. Questo esempio rappresenta lo schema di una connessione seriale RS232 tra un PC e il PPTEA.



DISPLAY

Il PPTEA può gestire un display tipo Hitachi44780 da 2 righe (oppure 4 righe) per 16 colonne . Se il numero di righe/colonne è diverso si può scrivere sul display settando la modalità libera (vedi SETIO).
Il display viene connesso al PPTEA come da schema elettrico e l'I/O viene gestito automaticamente alla prima chiamata che riguarda il display. E' possibile connettere un display mediante 6 fili o mediante due fili utilizzando un integrato intermedio (CD4096) e qualche componente.
I comandi relativi al display sono:

1. LCDCLEAR
2. LCDINIT
3. LCDPOS
4. LCDWRITE

Vediamo alcuni esempi:

Esempio Gioco:

```
02 BL=0
03 BL++
05 LCDCLEAR
07 LCDWRITE=" PPTEA-GAME-LCD"
08 LCDPOS=&H25
09 LCDWRITE="(FORUM)"
10 WAITMS 250
11 LCDCLEAR
12 WAITMS 100
13 IF BL MOD 3 THEN 03
15 BLANK="               "
16 LCDPOS=17
17 LCDWRITE="*"
18 WAITMS 100
25 a++
30 LCDPOS=16+a
40 WAITMS 100
50 LCDWRITE=" *"
55 IF a < 16 THEN 25
60 a--
70 LCDPOS=16+a
80 WAITMS 100
90 LCDWRITE="* "
100 IF a>2 THEN 60
103 a=0
104 NUM++
106 IF NUM < 5 THEN 11
107 num=0
110 GOTO 2
```

Output Display:
Appare la scritta lampeggiante:

```
" PPTEA-GAME-LCD"
"     (FORUM)"
```

successivamente appare una palla che rimbalza sul muro…e dopo 5 giri si ripete tutto.

Visualizzazione caratteri del display:

```
10 LCDCLEAR
15 LCDPOS=&H11
20 LCDWRITE="ALL-CHAR-DISPLAY"
25 LCDPOS=&H21
30 a++
35 b="c="&CHR(a)& "->"&a
40 USBOUT=b&""
45 LCDWRITE=b
50 WAITMS 255
55 GOTO  25
```

Output Display:
"ALL-CHAR-DISPLAY"
"C=X->nn"

Display in configurazione 2 fili:

```
06 SETIO=&H200DF ' il 17 BIT (partendo da 0 ) identifica la modalita 2 fili del display
10 LCDCLEAR
20 LCDWRITE="CIAO CIAO"
```

Output Display:
"CIAO CIAO"

Gestione due Display contemporanei:

```
05 SETIO=&H006F
10 LCDCLEAR
25 LCDWRITE="DISPLAY 1"
35 SETIO=&H2006F
40 LCDCLEAR
65 LCDWRITE="DISPLAY 2"
66 WAITS 1
67 LCDCLEAR
75 LCDWRITE="CIAO2"
76 WAITS 1
85 SETIO=&H006F
90 LCDCLEAR
95 LCDWRITE="CIAO1"
161 GOTO 161
```

Output Display 6 File:
"DISPLAY1" sostituita poi da "CIAO1"

Output Display 2 File:
"DISPLAY2" sostituita poi da "CIAO2"

REAL TIME CLOCK:L'orologio del PPTEA

Il PPTEA ha la possibilità di gestire un orologio esterno mediante un colloquio a due fili (BUS I2C). Con l'aggiunta di questo dispositivo vengono persi due bit di I/O:il bit 9 e bit 10. Il dispositivo in questione è il DS1307 che può essere configurato direttamente con il compilatore del PPTEA mediante il pulsante *SET TIME RTC* (vene inviata presente nel campo testo avente formato "GG/MM/AA HH:MM:SS D" (vedi SETDATE)) quando il PPEA si trova in modalità apprendimento. Se l'operazione ha avuto successo il PPTEA risponde con l'echo dell'orario inviato.L'orario presente nel campo si può modificare ma occorre verificare la correttezza dei dati. Per poter aggiornare l'orario al'orario corrente del pc va premuto il pulsante *Refresh Date*.

Il dispositivo dispone di un quarzo (32.768 kHz) e di una batteria tampone (3V) che mantiene l'ora anche se il PIC non è alimentato. Il dispositivo può funzionare contemporaneamente con l'espansione di memoria eeprom. Il dispositivo viene interrogato con l'istruzione DATE, mentre il formato può essere modificato mediante l'istruzione FDATE. L'istruzione DATE può andare in blocco se non sono presenti le resistenze di pul-up sul circuito del Timer esterno. Dalla versione 2.16c il Pin 7 del DS1307 oscilla con una frequenza di 1 Hz , utile per visualizzare i secondi in modo preciso.

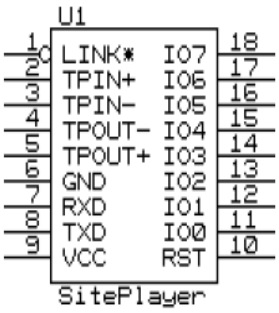
I comandi relativi all'orologio del PPTEA sono:

1. DATE
2. FDATE
3. SETDATE

WEB SERVER

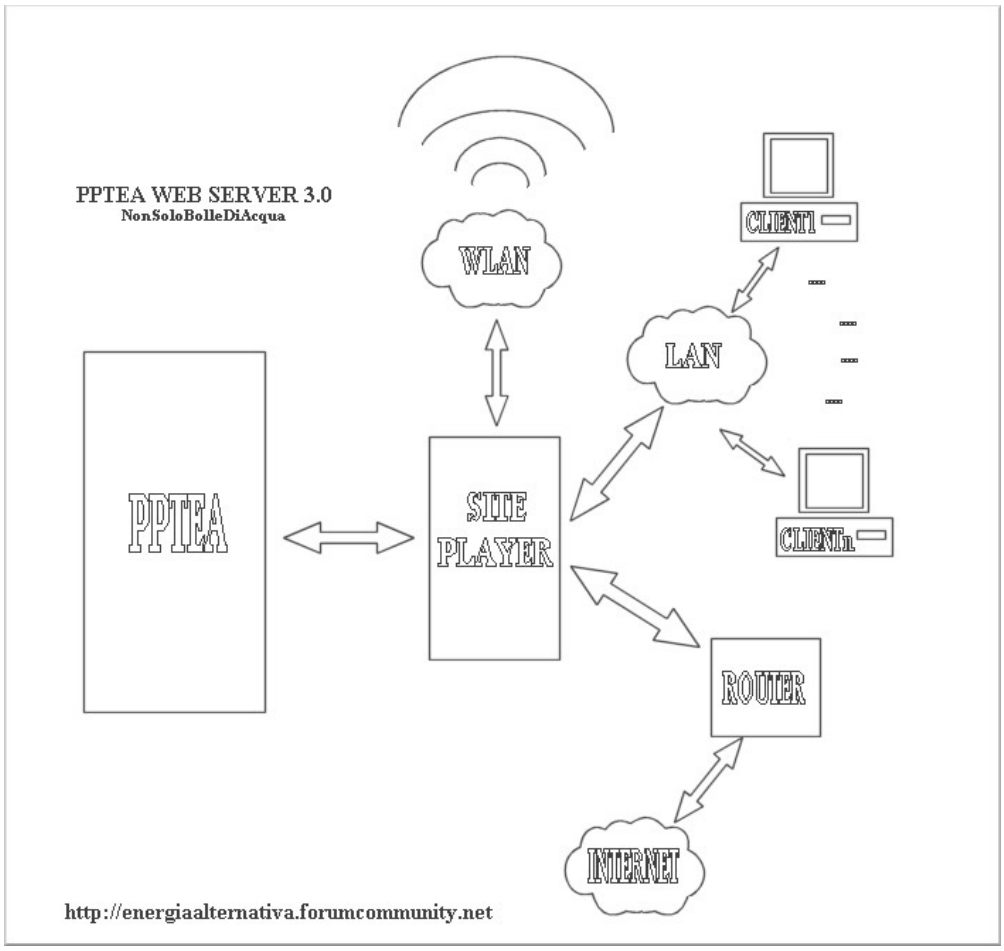
Il PPTEA, integrato con il SitePlayer SP1, ha la possibilità di essere utilizzato come un web server e gestire dati ed informazioni. Il colloquio PPTEA/SP1 avviene mediante la seriale (vedi schema elettrico).

(SitePlayer Sp1 http://www.siteplayer.com/)



Pin Name	Description
1 Link LED	Pin low when link has been established, typically resistor to LED to VCC
2 RX+	10BaseT receive + typically connects to filter/transformer
3 RX-	10BaseT receive - typically connects to filter/transformer
4 TX-	10BaseT transmit - typically connects to filter/transformer
5 TX+	10BaseT transmit + typically connects to filter/transformer
6 VSS	Ground
7 RXD	Receive Data to UART Can direct connect to device UART TXD
8 TXD	Transmit Data to UART Can direct connect to device UART RXD
9 VCC	+5 Volts, typically 75mA
10 Reset	High - Reset, Ground or No Connect - Run
11 through 18	Hardware I/O port

Questo è lo schema a blocchi delle possibili connessioni:



I comandi EABASIC relativi al WEB SERVER sono:

1. WEBSEND
2. WEBRECEIVE

ESEMPIO:

Nell'esempio è stata realizzata una pagina html , caricata nel modulino SP1e in internet è possibile leggere la temperatura e modificare lo stato di un bit, cioè accendere/spegnere un led. Tutto questo non necessita di un pc e l'assorbimento è di circa mezzo Watt.


```
05 SETIO=&H400FF
07 PRINT "PPTEA WEB SERVER 3.0"
08 BSEND=5
10 OUTBIT(15)=1
12 WAITMS 250
13 OUTBIT(15)=0
14 INDIRIZZO=1
15 TEMP=CADS1*CAD_TO_TEMP
16 GOSUB :INVIA_TEMPERATURA


35 T=TIMER
36 STR=WEBRECEIVE
37 IF STR <> "" THEN 50
38 WAITMS 50
40 IF T-TS >= 5 THEN 15 ELSE 35
50 PRINT "<"& STR &">"
51 IF LEFT(STR,10)="PPTEABIT16" THEN 55 ELSE 40
55 OUTBIT(15)= MID(STR,11,2)="ON"
60 GOTO 40


100 :INVIA_TEMPERATURA
110 STR_TEMP=LEFT(TEMP,5)
120 STR_SEND=CHR(128+BSEND-1)
121 STR_SEND=STR_SEND&CHR(INDIRIZZO)
122 STR_SEND=STR_SEND&STR_TEMP
130 WEBSEND=STR_SEND
131 TS=TIMER
134 PRINT STR_TEMP
135 RETURN
```

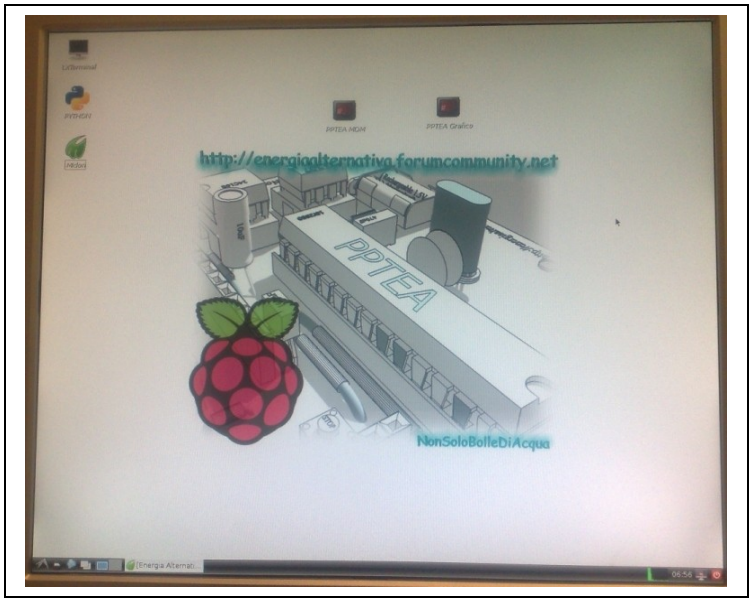
RASPBERRY PI

Il PPTEA, integrato con il Raspberry Pi, ha molteplici possibilità . Il colloquio PPTEA/RaspBerry avviene mediante la seriale (vedi schema elettrico).

3.3V	1	2	5V
I2C0 SDA	3	4	DNC
I2C0 SCL	5	6	GROUND
GPIO4	7	8	UART TXD
DNC	9	10	UART RXD
GPIO 17	11	12	GPIO 18
GPIO 21	13	14	DNC
GPIO 22	15	16	GPIO 23
DNC	17	18	GPIO 24
SP10 MOSI	19	20	DNC
SP10 MISO	21	22	GPIO 25
SP10 SCLK	23	24	SP10 CE0 N
DNC	25	26	SP10 CE1 N

(http://www.raspberrypi.org/)

Verranno consegnate delle immagini del Raspberry Pi per consentire un utilizzo immediato del PPTEA.



(Desktop Image PPTEA_RASPBERRY.img)

Precisione Matematica del PPTEA

Si può decidere la precisione matematica mediante l'istruzione MATH_PRECISION, per calcoli che non richiedono valori molto precisi il default va più che bene.

Vediamo la precisione del PPTEA mediante un esempio sulla radice quadrata.

Esempio:

```
25 A=2
30 B=SQR(A)
35 PRINT "Radice di " &a &"=" & B
40 C=B*B
45 PRINT "Radice*Radice="& C
46 D=C-A
47 PRINT "Radice*Radice-"&A& "=" &D
50 GOTO  50
```

Risultato del PPTEA:

Radice di 2=1.414213
*Radice*Radice=1.999999*
*Radice*Radice-2=-0.000000*

Possiamo vedere che la conversione float-stringa di per se non è molto precisa visto che riporta un valore preciso fino a 6 cifre dopo la virgola. Il valore del prodotto della radice per se stessa, sempre da un punto di vista della stringa ottenuta, è quasi 2 (si ottiene lo stesso valore iniziale) , mentre se il valore ottenuto sottraiamo 2 , otteniamo una stringa di valore 0...ma solo perchè la precisione è molto più alta e i soli caratteri visualizzati non permettono di valutare la precisione. Possiamo quindi verificare che il valore all'interno della variabile ha una precisone che va oltre la conversione del float-stringa.

Elenco errori del compilatore

DUPLICATE LABEL : Label già presente

NO PROGRESSIVE NUMBER: I numeri di linea nel codice sorgente non sono progressivi.

LINE NUMBER ERROR: La linea è sprovvista di numero o è un valore non consentito o non esistente

WRONG LINE NUMBER: Non esiste il numero di linea

NO END OF LINE: Si attendeva la fine della linea

NO END STRING: Si attendeva la fine della stringa

DIM MUST FIRST ISTRUCTION: L'istruzione DIM deve essere la prima istruzione

DIMENSION ERROR : Errore nella dichiarazione della dimensione del vettore

SINTAX ERROR: Sintassi Errata

SINTAX ERROR LABEL : Label non presente

MISSING = , *MISSING (*,*MISSING)*: Manca l'uguale la parentesi tonda aperta la parentesi tonda chiusa

MISSING , : Manca la virgola

MISSING VARIABLE : Manca la variabile

MISSING VECTOR: Manca il Vettore

*MISSING , OR END OF LINE :*Manca la virgola o la fine della linea

MISSING AS : Manca il costrutto AS

MISSING TYPE : Manca il Tipo

PRAGMA UNKNOW: Pragma non riconosciuto

PRAGMA TOO MUTCH: Troppi Pragma nel codice

SINTAX VARIABLE ERROR : Errore nella definizione della Variabile

NO PROGRAM IN MEMORY: Nessun programma in memoria

VARIABLES OVERFLOW: Sono state inserite troppe variabili massimo 26 variabili

ERROR LINE TO LONG TOKENIZER: La linea è più lunga di 127 caratteri

STRING OVERFLOW: La stringa contiene più di 63 caratteri

EXPRESSION OVERFLOW: Occorre semplificare l'espressione inserita

VECTOR OVERFLOW: Si è andati oltre lo spazio del Vettore

CHANGE NAME VARIABLE: Il nome della variabile non è permesso

NO LOOP: Manca l'istruzione LOOP

NO REPEAT: Manca l'istruzione REPEAT

WRONG VALUE …… : Valore errato

DATA BEFORE READ : L'istruzione DATA deve essere presente prima dell'istruzione READ

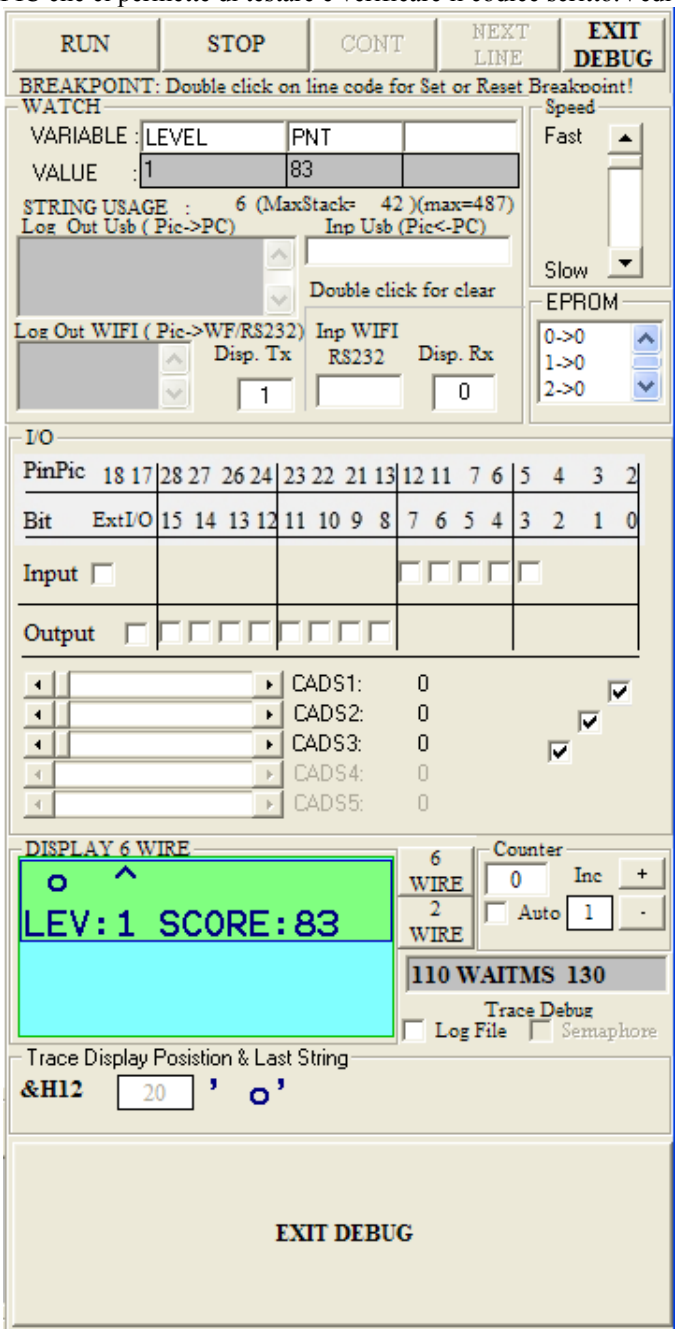
DATA NOT SEQUENTIAL : Le istruzioni DATA debbono essere consecutive

SENDORx ADC DISABLE: Il convertitore Analogico Digitale non è abilitato o presente

WARNING FLOAT PRECISION: Ecceduto sul numero di cifre decimali (MAX 11)

DEBUG INTEGRATO NEL COMPILATORE

Dopo aver salvato e compilato un codice scritto in EABasic possiamo provarlo nel debugger nel PC. Il compilatore ha all'interno un emulatore del PIC che ci permette di testare e verificare il codice scritto.Vedi DEBUGGER DEL PPTEA.



ESPANSIONE DI MEMORIA PER IL CODICE

(Eprom Esterna con protocollo I2C)

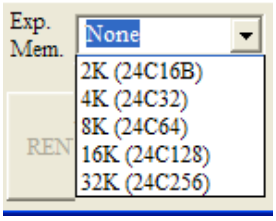
Il PPTEA ha la possibilità di gestire una EEPROM esterna da 16kbit (24C16B) fino ad una da 256Kbit mediante un colloquio a due fili (BUS I2C) totalmente trasparente all'utente. Con l'aggiunta di questa memoria vengono persi due bit di I/O:il bit 9 e bit 10. Questa memoria, come anche quella presente all'interno del PIC, può essere utilizzata per vari scopi:

1.Contenere dati (rimangono presenti anche in assenza di alimentazione)

2.Contenere un programma eabasic (molto più grande di quello permesso nel PIC :si passa da 254 Token fino ad un massimo di 32787 Token)

3.Contenere programma e dati.

Il programma viene inserito nella memoria esterna se viene settato uno di questi flag inerenti la memoria che viene utilizzata.



Nella barra dello Status & ChipUsage si nota che lo spazio a disposizione aumentando la dimensione della memoria esterna aumenta in modo sensibile e sono consentiti programmi molto più lunghi. Utilizzando un programma esterno la velocità di esecuzione è circa dimezzata (Vedi Test Velocità EABASIC PPTEA). Nelle programma riportato viene testata la eeprom esterna scrivendo e leggendo un valore su tutta la memoria. Il programma, che impiega diversi secondi, deve essere caricato nella memoria interna (None sulla EXp. Mem.). Se è presente verrà visualizzata sulla finestra di out del PC la scritta "INIZIO!..... .....FINE!".Se verrà visualizzata la scritta "INIZIO!.ERR=0" la eemprom esterna non è presente o non viene vista correttamente.

Esempio:

```
10 REM SCRITTURA SU 24C16B ESTERNA
12 USBOUT="INIZIO!"
15 EEXTERNAL=1
20 A=0
25 CELL=A
30 V=a/2 AND 255
35 WEEPROM=V
40 B=REEPROM
47 USBOUT="."
50 IF b <> V THEN 85
55 WAITMS 1
60 A++
70 GOTO  25
75 USBOUT="FINE!"
80 GOTO  80
85 IF A=2048 THEN 75
86 USBOUT="ERR=" & A
90 GOTO  90
```

ATTENZIONE:Eliminando la EEPROM ESTERNA con un programma caricato al suo interno :il PPTEA va in BLOCCO. Anche se non vengono inserite le due resistenze di pull-up presenti sulla eeprom esterna il PPTEA va in BLOCCO.

TEST VELOCITA' EABASIC PPTEA

Il PPTEA esegue le istruzioni mediante un interprete interno al Firmware e le istruzioni hanno una velocità di circa 500 microsecondi. L'esempio riportato viene eseguito in circa 5 secondi.

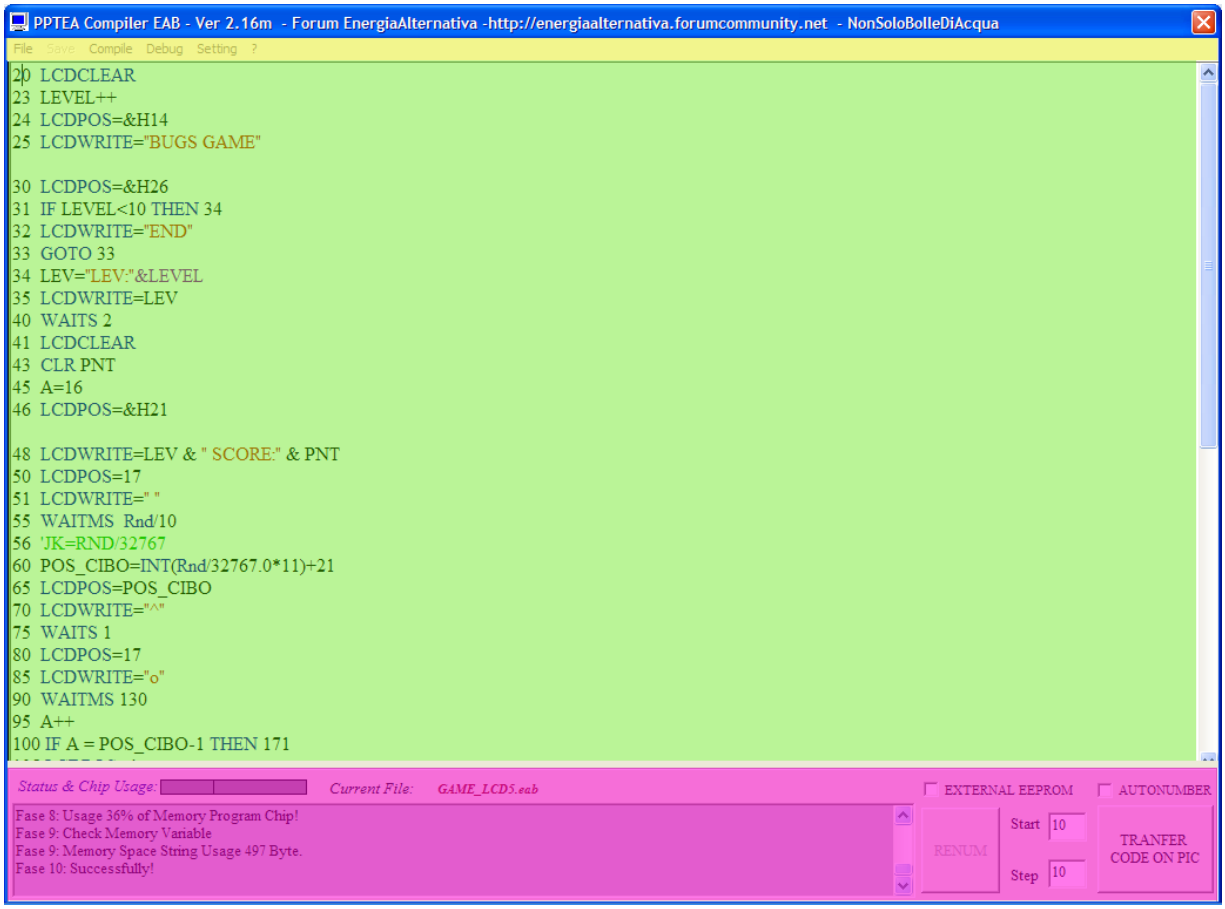
Esempio:

```
10 'TEST VELOCITA' EABASIC
15 USBOUT="INIZIO!"
20 B++
25 IF B <5000 THEN 20
35 USBOUT="FINE!"
45 GOTO 45
```

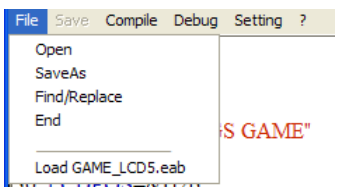
Output PC: *INIZIO!FINE!*

COMPILATORE PPTEA

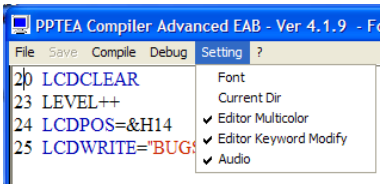
Il compilatore del PPTEA è composto da 3 aree:



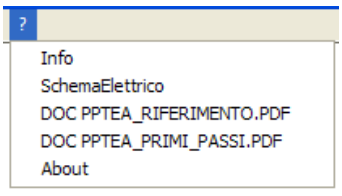
1.Area Menù (colore giallo)



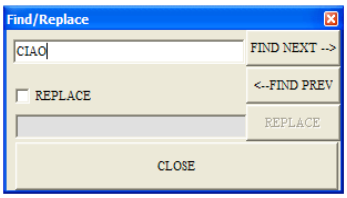
(Sottomenù della voce File)



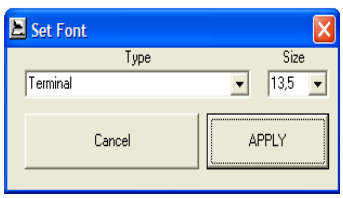
(Sottomenù della voce Setting)



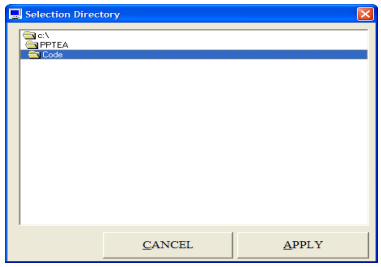
(Sottomenù della voce ?)



(Finestra Find/Replace)



(Finestra SetFont)



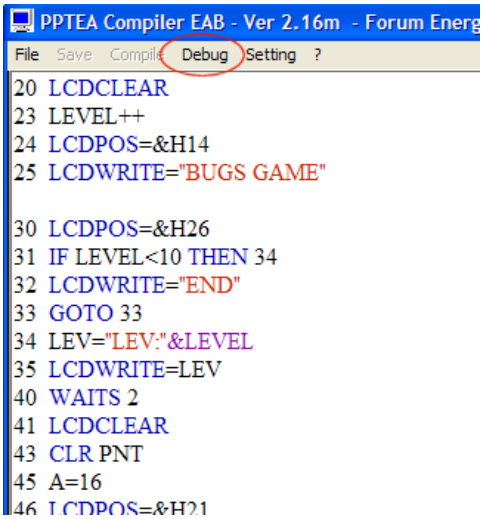
(Finestra Directory)

2.Area Testo (colore verde)
Il questa area si scrive o si inserisce il codice scritto in linguaggio EABASIC per essere compilato, per effettuare le operazioni di debug o per essere trasferito all'interno del CHIP PPTEA. Nell'area di testo premendo contemporaneamente i tasti CTRL-F si apre la finestra di Find/Replace. Inserendo il flag su AUTONUMBER ogni volta che si va a capo viene calcolato,in modo automatico, un nuovo numero di linea. Il pulsante CTRL-P lancia l'esecuzione dell'algoritmo di colorazione del programma eabasic e il pulsante CTRL-K lancia l'esecuzione della riformattazione del codice.

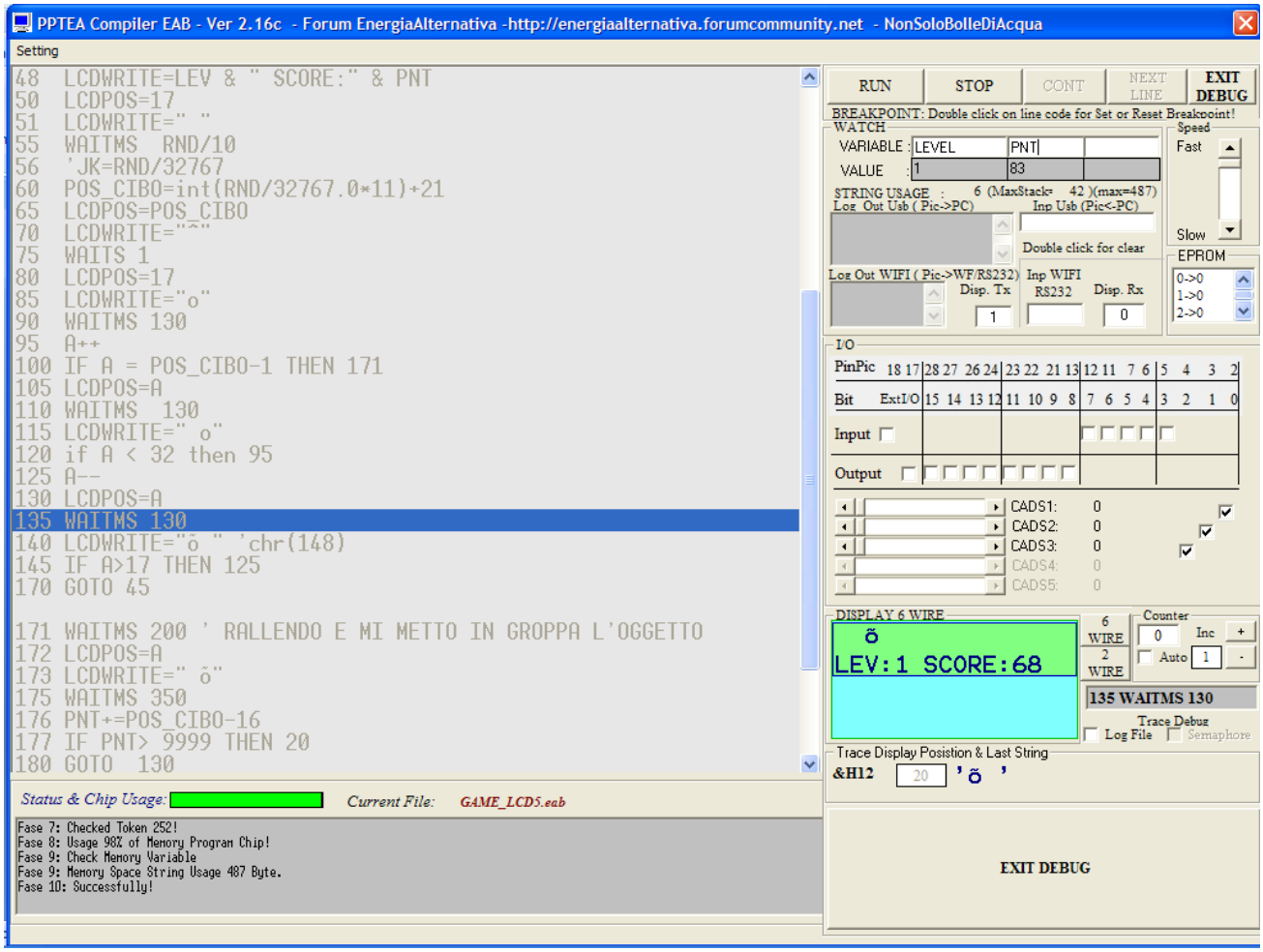
3.Area Comandi&Segnalazioni (colore viola)
Il questa area si ha la visione del file su cui si lavora, la sua occupazione di memoria e si tengono sotto controllo le segnalazioni del compilatore. Si può effettuare una re-numerazione delle linee del programma e/o effettuare una numerazione automatica. Una volta terminata la compilazione di può passare alla modalità trasferimento codice mediante il pulsante " TRANFER CODE ON PIC"

DEBUGGER DEL PPTEA

Il compilatore del PPTEA ha la possibilità di effettuare una emulazione del codice scritto mediante l'opzione di *Debug* (presente nelle opzioni del menù visibile). L'opzione diventa disponibile sole se la compilazione del codice è avvenuta con successo.



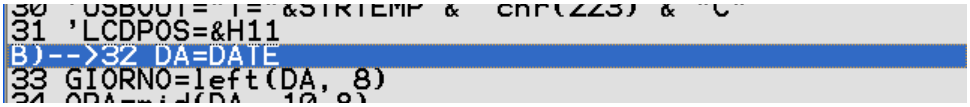
Quando si entra nel Debug l'area di testo contenente il codice diventa di colore grigio e sulla destra dello schermo appare una finestra che permette la gestione della simulazione.



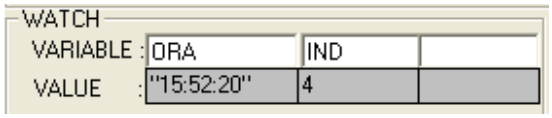
L'area di comandi è composta da 5 pulsanti che permettono la gestione della simulazione:

Il pulsante *RUN* avvia l'esecuzione del programma e la barra di colore Blu nell'area di codice identifica la linea in esecuzione.L'esecuzione viene fermata se si preme il pulsante STOP o se viene incontrato un Breakpoint (blocco di esecuzione).
Il pulsante *STOP* arresta l'esecuzione del programma.
Il pulsante *CONT* permette la ripresa dell'esecuzione.
Il pulsante *NEXT LINE* permette l'esecuzione di una singola linea di codice alla volta.
Il pulsante ***EXIT DEBUG*** permette l'uscita dall'ambiente di *DEBUG*.

Un BreakPoint viene inserito effettuando un doppio click sull'area di codice e il BreakPoint viene evidenziato con una "B)-→".

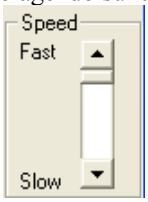


Effettuando un doppio click sul BreakPoint questo viene disattivato. Si possono inserire tanti break a piacimento.
Quando il Debugger incontra un Break dallo stato di Run passa allo stato di STOP.
L'area WATCH permette di visualizzare i valori delle variabili utilizzate nel programma:

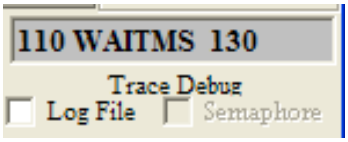


Nell'esempio vengono visualizzate le variabili ORA e IND e il loro contenuto è visualizzato nella riga sottostante le variabili.
Si possono visualizzare contemporaneamente al massimo tre variabili ed il loro contenuto è aggiornato in modo automatico.
E' possibile modificare il contenuto della variabile (nello stato di STOP) modificando il valore e inviando un <RETURN>. La modifica del contenuto della variabile verrà avvisato con un beep.

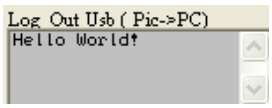
La velocità di esecuzione del Debug può essere deciso agendo sulla scroll bar verticale:



L'ultima istruzione di codice eseguita viene mostrata nell'area di Trace:

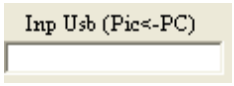


L'area di Log OUT USB visualizza i dati che il PPTEA invia al PC sulla porta USB. I comandi che inviano i dati sulla porta usb sono: USBOUT, USBOUTB:



Nell'esempio sopra viene riportato il log del codice Hello World.eab

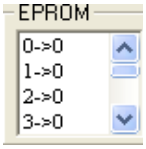
Per simulare i dati ricevuti dal PPTEA provenienti dalla porta USB occorre inserirli nell'area INP USB:



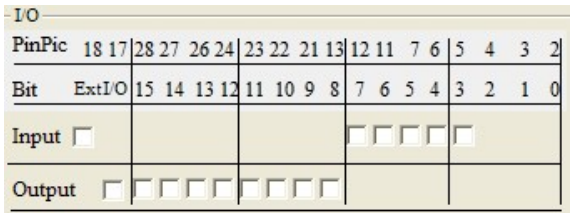
L'area di Log OUT WIFI visualizza i dati che il PPTEA invia in radiofrequenza :

Per simulare i dati ricevuti dal PPTEA provenienti dal WIFI occorre inserirli nell'area INP WIFI:

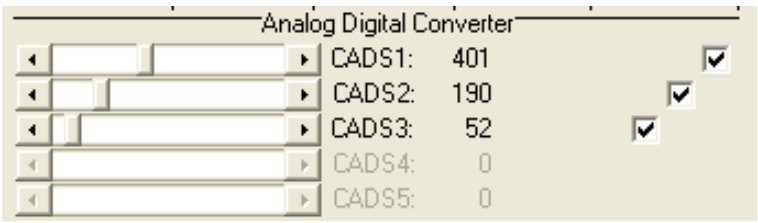
L'area che emula l'EEPROM interna o esterna è :



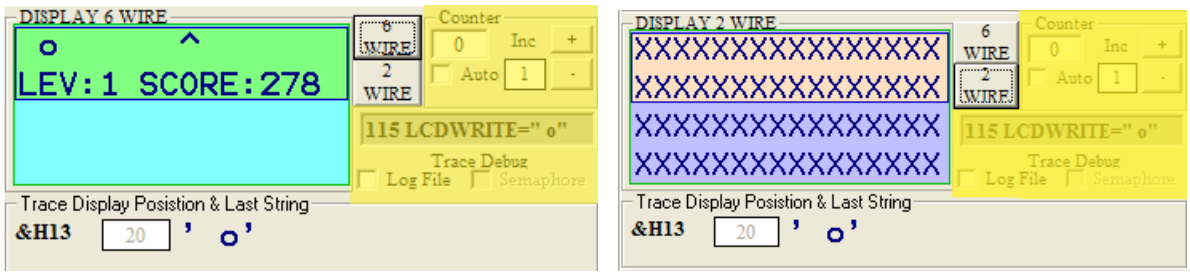
L'area relativa all'I/O premette la visualizzazione dello stato e la modifica dei bit. Le righe *PinPic* e *Bit* identificano rispettivamente i pin del 18f2550 e i bit dell' I/O interni al PPTEA. Un bit è un ingresso se è posto nella riga degli *Input* ed è una uscita se è posto nella riga degli *Output*. I bit Output è di valore1 se è presente il flag altrimenti vale 0. Un bit può essere ingresso o uscita e questo dipende dal valore che impostiamo con il comando SETIO. Il valore di un uscita dipende dal comando OUT. La funzione INP torna i lo stato dei bit di input. Dalla versione 2.16c sono implementati due estensione di bit (uno di ingresso pin 18 ed uno di uscita pin 17).



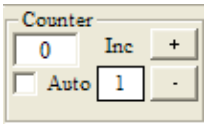
L'area Analog Digital Converter permette la modifica dei valori (da 0 a 1023) dei convertitori presenti.
CADSN seleziona il numero di convertitori utilizzati, CADSn e CadsInd(n) permette la lettura del sensore analogico digitale n (dove n assume un valore da 1 a 5).



L'area DISPLAY permette la visualizzazione del display attivo 2x16 e 4x16. L'area Trace Display Position & Last String permette la visualizzazione della posizione dell'ultimo carattere visualizzato e l'ultima stringa mandata in presentazione. Nel Debug sono presenti i display a 6 fili e il display a 2 fili. Il pulsante "6WIRE" manda in presentazione il display a 6 fili, mentre il pulsante "2WIRE" quello a due fili. In presentazione viene mandato l'ultimo display che ha ricevuto un comando. I display sono riconoscibili sia dal titolo che dai colori differenti.



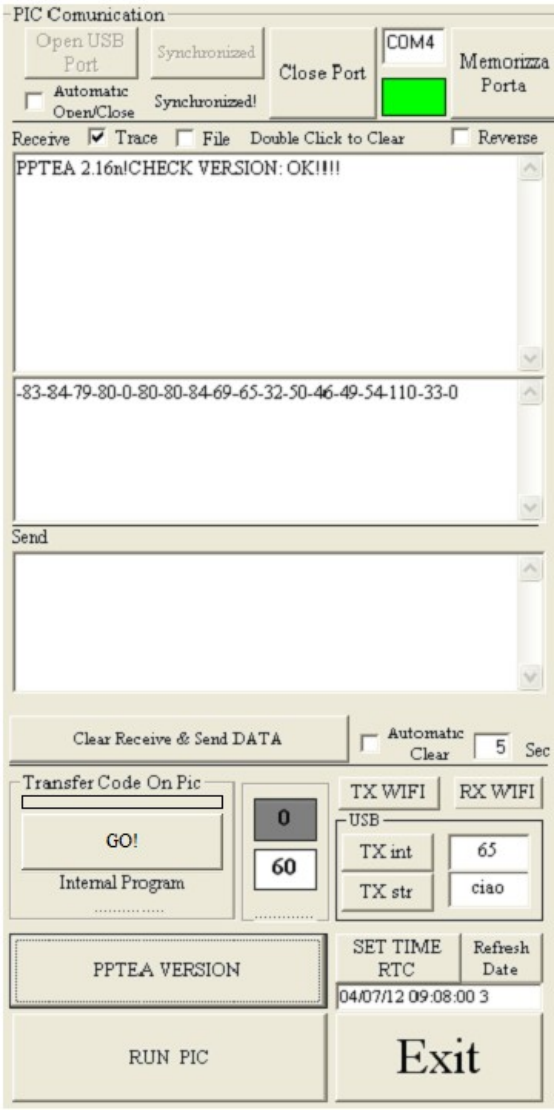
Il debugger del PPTEA può gestire un contatore esterno. Il valore è rappresentato nell'area testo sotto la label Counter presente in basso a destra nell'area di debug. I pulsanti + e – incrementano o decrementano il valore del COUNTER. E' possibile configurare una modalità automatica di auto incremento mettendo il flag su *Auto* e il valore viene incrementato di Inc.



Si esce dalla modalità di Debug premendo il pulsante *EXIT DEBUG*.

COMUNICAZIONE PPTEA PC

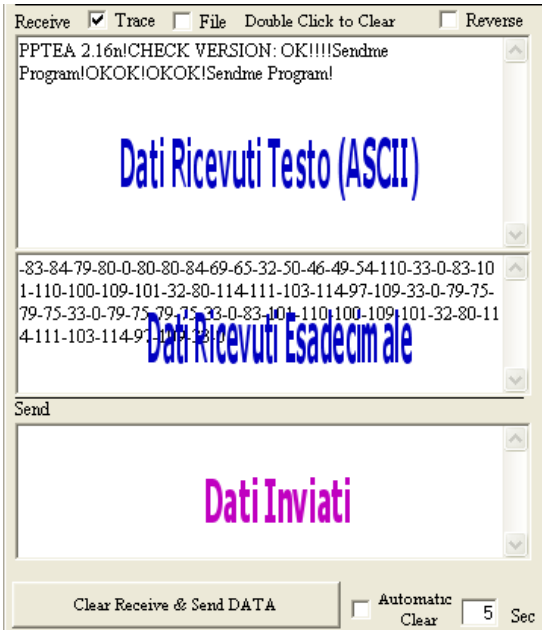
Premendo il pulsante “TRANSFER CODE ON PIC” presente in basso a destra nella pagina principale, si apre, sul fianco destro, la ***Finestra di Comunicazione PPTEA-PC:***



La parte in alto gestisce la porta seriale, la parte centrale visualizza i dati ricevuti ed inviati in formato testo ed esadecimale, la parte in basso il trasferimento del codice, la versione , esecuzione del programma , interazioni usb, il setting dell’orologio.

- Gestione Porta Seriale
  Il pulsante “Open USB Port” apre la porta seriale
  Il pulsante Syncronized tenta ripetutamente di aprire la porta, il campo testo sotto il pulsante riporta lo stato delle operazioni
  Il pulsante Close Port chiude l porta
  Il campo testo (dove è presente COM4) identifica la porta seriale associata alla USB che permette la comunicazione con il PPTEA.
  Il colore (nell’immagine verde) identifica lo stato della porta
  Il pulsante “Memorizza Porta” permette di memorizzare in modo permanente la porta inserita nel campo testo

- Scambio Dati
  Il flag *Trace* se abilitato visualizza il traffico di dati in ricezione (PPTEA->PC) e trasmissione (PC->PPTEA)
  Il flag *File* se abilitato scrive accodando i dati sul file "PPTEA_LOG.TXT"
  Il flag *Reverse* presenta i dati in arrivo in ordine inverso …l'ultimo dato viene scritto in testa all'area di testo.
  Il pulsante *Clear Receive & Send Data* cancella le aree di testo di ricezione e trasmissione dati.
  Il flag *Automatic Clear* effettua una cancellazione automatica con i secondi specificati nel campo *Sec.*



- Trasferimento programma, controllo versione, esecuzione programma
  Il pulsante *"GO!"* ,presente nell'area *Transfer code on pi,c* permette l'invio del codice nel PPTEA.
  Questo pulsante è abilitato se la connessione è effettuata con successo e se il codice è stato compilato con successo. Quando avviene il trasferimento vengono segnalati ,nell'area testo, lo stato dei pacchetti trasferiti.
  Sul fianco destro del pulsante *"GO!"* sono presneti due aree di testo numeriche che identificano il numero di byte da trasmettere (in basso) e quelli che sono trasmessi (in alto). Il colore verde identifica che il trasferimento è terminato positivamente.
  Il pulsante *PPTEA VERSION* effettua la verifica delle versioni tra PPTEA e Compilatore.
  Il pulsante *RUN PIC* lancia il programma caricato nel processore se il PPTEA è nello stato di *Apprendimento*.

- Interazione WIFI, USB e setting orologio (RTC)
  Il pulsante *TX WIFI* invia un dato in wireless, il pulsante RX WIFI permette la ricezione di un dato in wireless
  I due pulsanti *TX int* e *TX str,* presenti nell'area USB, inviano i dati al PPTEA mediante la porta USB.
  Il pulsante "SET TIME RTC" invia la data al Real Tiem Clock del PPTEA
  Il pulsante "Refresh Date" permette di aggiornare il campo Date prendendo l'orario corrente del PC
  L'area Date contiene il formato DATE da inviare al PPTEA: Giorno/Mese/Anno Ora:Minuti:Secondi GiornoDellaSettiama.

MACRO DEL COMPILATORE

Sono utilizzate per rendere il codice più leggibile, il compilatore sostituisce la ,acro con il relativo comando:

MACRO	*COMANDO*
RESET_TIMER (*)	PWMDC1=0
STOP_PWM1	PWMDC1=0
STOP_PWM2	PWMDC2=0
STOP_PWM3	PWMDC3=0
BEEP	FRQ 220,150
BEEP_OK	FRQ 70,150
BEEP_NOK	FRQ 880,150
PRINT ….	USBOUT= …- & " "
PRINTLCD …..	LCDWRITE= ….
;	&

(*) RESET_TIMER :Attenzione alla chiamata i bit associati vengono portati a zero, se si vuole utilizzare il timer occorre resettare l'io del PWM1 oppure non utilizzare il bit 16 (default)

OPZIONI DEL COMPILATORE/DEBUGGER

Sono comandi ed opzioni di utilizzo esclusivo del compilatore e del debugger.

_ONLY_DEBUG *Istruzione*

L'istruzione viene eseguita solo in fase di debug. Quando si compila non verranno incrementati i token e l'istruzione non sarà eseguita dal PPTEA.

Esempio:
```
10 _ONLY_DEBUG A=20
```

_ONLY_PPTEA *Istruzione*

L'istruzione viene eseguita solo dal PPTEA e non viene eseguita in fase di debug. Quando si compila verranno incrementati i token e l'istruzione sarà eseguita solo in fase di esecuzione dal PPTEA.

Esempio:
```
10 _ONLY_DEBUG B=30
```

Esempio di programma che viene configurato a seconda di dove viene eseguito senza che venga alterato il codice:

Esempio:
```
10 _ONLY_DEBUG A=1
20 _ONLY_PPTEA  A=200
25 USBOUT="A="& A
30 GOTO 30
```

Output Debug :	*Output PPTEA:*
A=1	A=200

MODALITA' DEL PPTEA

Il PPTEA ha tre modalità di funzionamento:
1.Avvio
2.Apprendimento
3.Esecuzione

Modalità Avvio:
Il PPTEA entra in modalità avvio all'accensione del PIC e rimane in questo stato per circa 5 secondi per passare automaticamente alla modalità esecuzione. L'attesa di 5 secondi può essere annullata se a livello hardware viene inserita la resistenza di pull-up di IMMEDIATE START (vedi IMMEDIATE START), in questo caso il PPTEA all'accensione passerà immediatamente nella modalità esecuzione.

Modalità Apprendimento:
Nella modalità di apprendimento il PPTEA è in grado di effettuare diverse operazioni:
a.Ricevere un nuovo programma scritto in EABASIC e precedentemente compilato (mediante il pulsante "GO!").
b.Rispondere sulla versione del firmware (mediante il pulsante "PPTEA VERSION").
c.Lanciare l'esecuzione del programma eabasic (mediante il pulsante "RUN PIC").
d.Mandare un byte sulla porta usb (mediante il pulsante "SEND USB").
e.Inserire ora e data nell'orologio esterno (mediante il pulsante "SET RTC")
e così via.
Il PPTEA se è nella modalità esecuzione passa nella modalità apprendimento mediante il pulsante STOP (segnalata sulla porta USB).
Se il PPTEA è già nella modalità di apprendimento la pressione del pulsante STOP (vedi schema elettrico) non ha effetto.

Modalità Esecuzione:
Nella modalità di esecuzione il PPTEA esegue il programma. E' possibile fermare il programma mediante il pulsante STOP (vedi schema elettrico) e passare nella modalità Apprendimento.

IMMEDIATE START

Il PPTEA si avvia immediatamente (saltando l'attesa iniziale) se è presente una resistenza di 470 Ohm sul pin 26 (bit 13) messa a positivo (vedi schema elettrico).

PULSANTE DI STOP

Il pulsante di Stop se presente, permette di fermare l'esecuzione del programma eabasic. Quando viene premuto viene resettato l'I/O e le uscite vengono portate a massa compresa l'uscita associata al PWM. Per la connessione vedere lo schema elettrico.

COMPILATORE

Il compilatore (eseguibile per Windows) permette la stesura di programmi scritti in eabasic. I programmi andranno compilati e solo dopo essere stati compilati con successo potranno essere trasferiti nel PPTEA mediante la procedura di trasferimento del programma EABasic.

CONNESSIONE PC USB/PPTEA Syncronized (agganciare/sganciare la porta USB)

Questo pulsante permette il sincronismo tra PPTEA e PC mediante la portaUSB. Il compilatore tenta in modo automatico la connessione con il PPTEA. La connessione effettuata viene segnalata con il colore verde.

ARCHITETTURA DEL PPTEA

Il PPTEA gestisce una piccolissima memoria (512 byte) che viene condivisa dalle variabili, dai vettori e dalle stringhe. Ogni variabile occupa 5 byte e i vettori (dichiarati con l'istruzione DIM) occupano 2 byte ad elemento .Le variabili utilizzate possono essere massimo 26 (130 byte) . Se i vettori non vengono utilizzati tale spazio è libero. Questo controllo viene effettuato dal compilatore, mentre la gestione delle stringhe è lasciata all'utente non conoscendo a priori cosa realizza il software.

AREA PPTEA di 512 Byte

| VARIABILI | (Spazio occupato: variabili * 5 byte)

| VETTORI | (Se non presenti non occupano spazio , altrimenti si prendono lo spazio rimasto)

| STRINGHE |

La somma dell'area non deve eccedere i 512 byte , un primo controllo viene effettuato dal compilatore (variabili utilizzate e vettori) ma a priori non si può sapere come il software possa gestire la memoria delle stringhe, quindi tutto è in mano al programma in eabasic.

Facciamo un esempio di un **programma errato** dove la stringa contenuta nella variabile A diventa sempre più grande fino a finire lo spazio della memoria:

```
10  A = "ciao-"
20  A=A&A
30 USBOUT =A
40 GOTO 20
```

Output pc:

ciao-ciao-ciao-ciao-ciao-ciao-ciao-ciao-ciao-ciao-ciao-ciao-ciao-ciao-ciao-ciao-ciao-ciao-ciao-ciao-ciao-ciao-ciao-ciao-ciao-ciao-ciao-ciao--ciao-ciao-ciaociao-ciao-ciao-ciao-ciao-ciao-ciao-ciao-ciao-ciao-ciao-ciao-ciao-ciao-ciao-ciao-ciao-ciao-ciao-ciao-ciao-ciao-ciao-ciao-ciao-ciao-ciao-ciao-ciao-ciao-ciao-ciao-ciao-ciao-ciao-ciao-ciao-ciao-ciao-ciao-

Facciamo un esempio di un programma che non utilizza variabili:

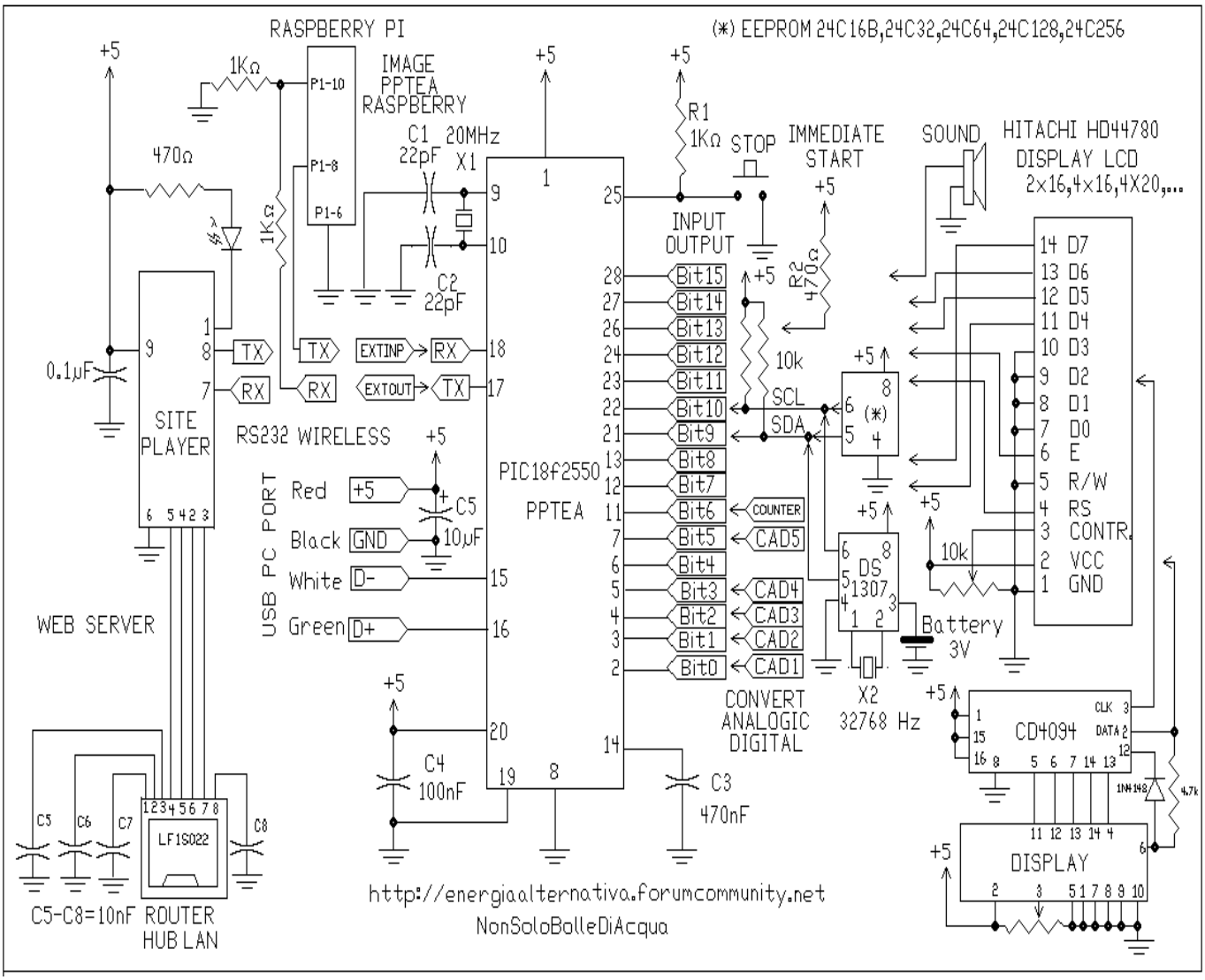
```
10 USBOUT="NON USO VARIABILI!-“
20 WAITMS 250
30 GOTO 10
```

Output pc:
NON USO VARIABILI!-NON USO VARIABILI!-NON USO VARIABILI!-NON USO VARIABILI!-NON USO VARIABILI!-NON USO VARIABILI!-NON USO VARIABILI!-NON USO VARIABILI!-…….

CONSUMI ED ASSORBIMENTO DEL PPTEA

L'assorbimento del PPTEA è molto basso: con display , RTC, sensore di temperatura arriva ad assorbire sui 40 mA… cioè **200 mW**…**cioè 1/5 di Watt…molto poco**, quindi un anno acceso consuma meno di 2kWh…cioè meno di 50 centesimi di Euro.

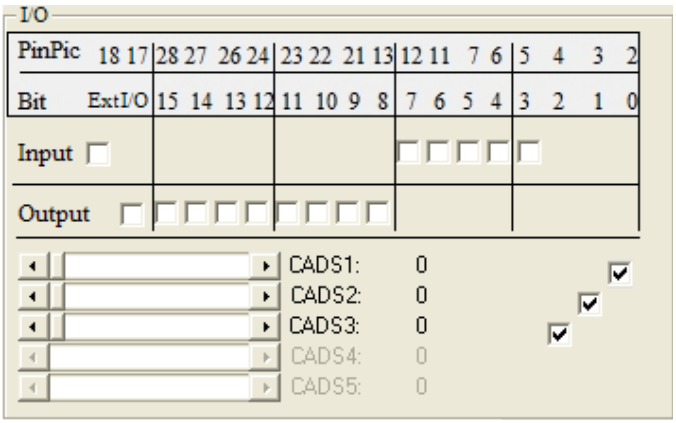
PPTEA - SCHEMA ELETTRICO & PIN/BIT



I/O del PPTEA

Il PPTEA dispone di 18 ingressi/uscite di cui 16 configurabili mediante il comando ***SETIO***. Nelle 18 porte sono presenti cinque Convertitori Analogico Digitali (CAD) configurabili mediante il comando ***CADS.*** Lo stato degli ingressi torna mediante il comando ***INP*** (o ***INPBIT***) e le uscite sono impostabili mediante il comando ***OUT*** (o ***OUTBIT***). I due bit di estensione dell'I/O non sono configurabili e si accede ad essi mediante i comandi ***EXTINP***, ***EXTOUT***.

Allo StartUp il PPTEA presenta il seguente I/O visibile nella finestra di *Debug* del *Compiler*:



La modalità di **default** è visibile in grassetto:

BIT	PIN 18F2550	MODALITA'
0	2	Ingresso/Uscita/**CAD1**
1	3	Ingresso/Uscita/**CAD2**
2	4	Ingresso/Uscita/**CAD3**
3	5	**Ingresso**/Uscita/CAD4
4	6	**Ingresso**/Uscita
5	7	**Ingresso**/Uscita/CAD5
6	11	**Ingresso**/Uscita
7	12	**Ingresso**/Uscita
8	13	Ingresso/**Uscita**
9	21	Ingresso PullUp/**Uscita**
10	22	Ingresso PullUp /**Uscita**
11	23	Ingresso PullUp/**Uscita**
12	24	Ingresso PullUp /**Uscita**
13	26	Ingresso PullUp /**Uscita**
14	27	Ingresso PullUp /**Uscita**
15	28	Ingresso PullUp /**Uscita**
EXT OUTPUT	17	**Uscita**
EXT INPUT	18	**Ingresso**

I sette bit da 9 a 15 (9,10,11,12,13,14,15 cioè i pin 21, 22,23,24,26,27 e 28) sono connessi a delle resistenze di pull-up.

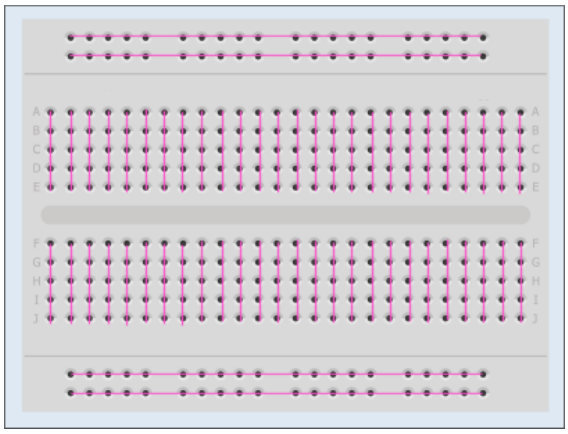
Il PPTEA può modificare in modo automatico la modalità o lo stato di alcuni BIT. Vediamo tutti i casi:

- ***Presenza di Espansione di Memoria***:
  Se disponiamo di una EEPROM Esterna il BIT9 ed il BIT10 vengono utilizzati dal PPTEA per colloquiare e di fatto questi due BIT non possono essere utilizzati.

- ***Presenza di Orologio Esterno(Real Time Clock)***:
  Quando viene utilizzata una qualsiasi chiamata alle funzioni dell'Orologio il PPTEA colloquiando con il dispositivo, modifica la modalità e lo stato dei BIT9 e del BIT10.

- ***Presenza di Display:***
  a)Display a 6 fili:
  Quando viene utilizzata una qualsiasi chiamata alle funzioni del Display il PPTEA riconfigura i BIT 14-13-12-11-8-7. Questi bit non possono essere usati o se vengono usati occorre settarli nella giusta modalità per non compromettere le operazioni del Display.
  a)Display a 2 fili:
  Quando viene utilizzata una qualsiasi chiamata alle funzioni del Display il PPTEA riconfigura i BIT4 e il BIT11. Questi bit non possono essere usati o se vengono usati occorre settarli nella giusta modalità per non compromettere le operazioni del Display.

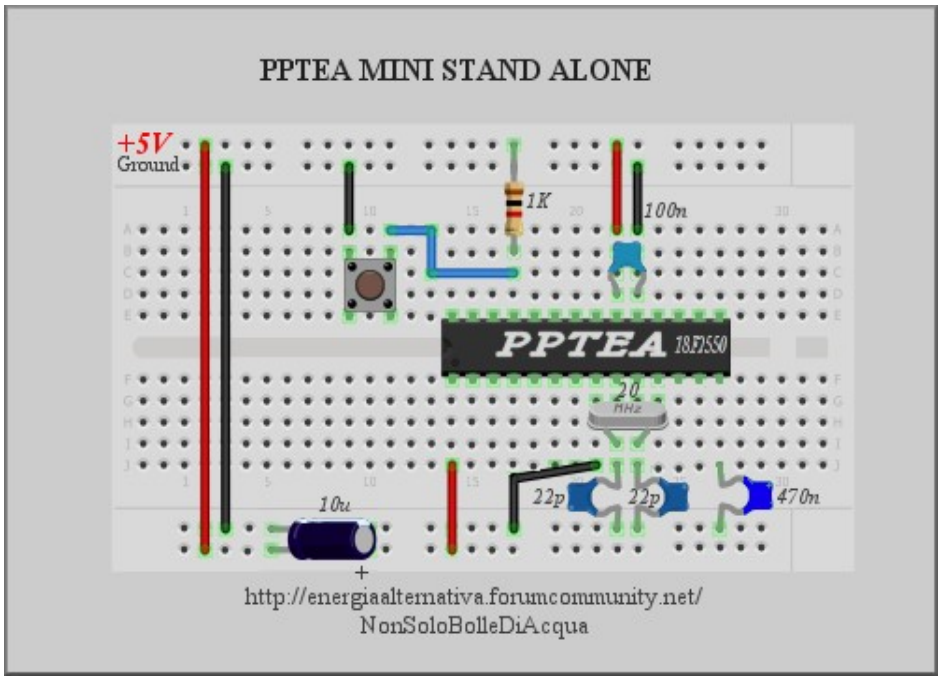
CONNESSIONI DELLA BREADBOARD

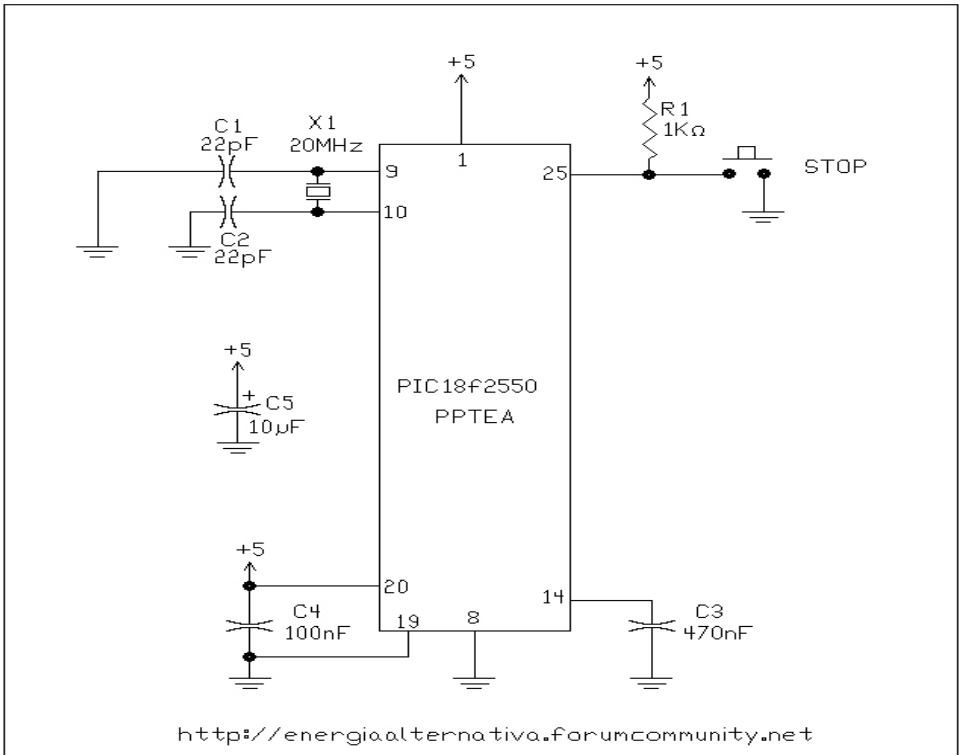


CONFIGURAZIONI HARDWARE DEL PPTEA

Il PPTEA ha **tre configurazioni di base**:

1. **PPTEA MINI STAND ALONE**
   Questa configurazione è la minima che permette al PPTEA di funzionare in modalità stand-alone, cioè senza essere connesso al PC. Necessita di alimentazione stabilizzata di +5V.



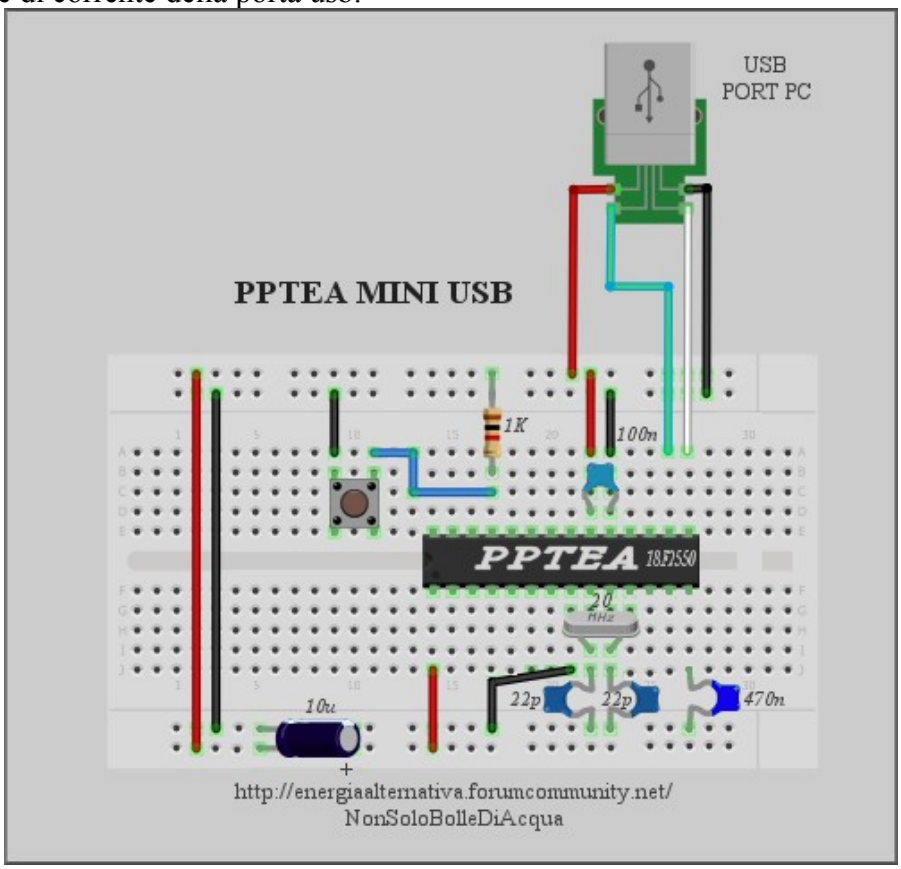


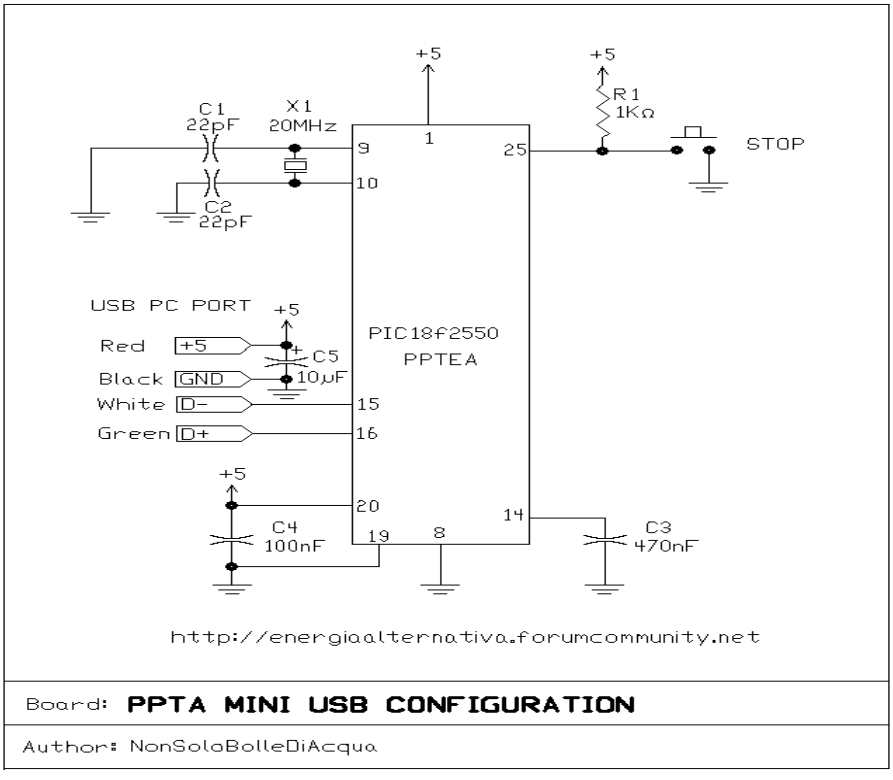
Board: **PPTA MINI STAND ALONE CONFIGURATION**

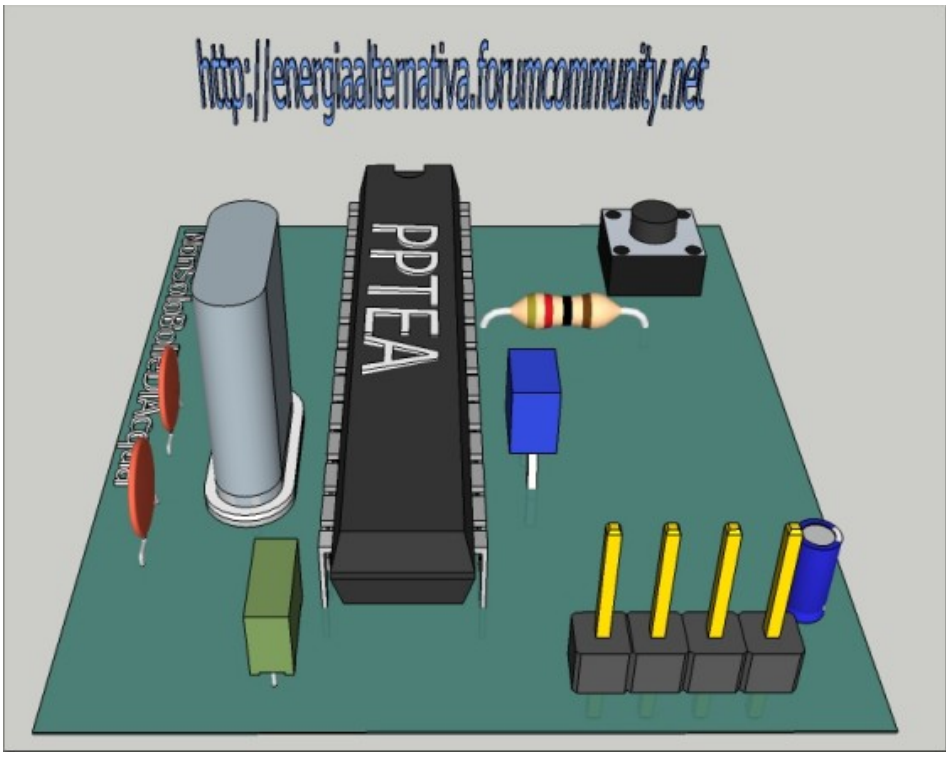
Author: NonSoloBolleDiAcqua

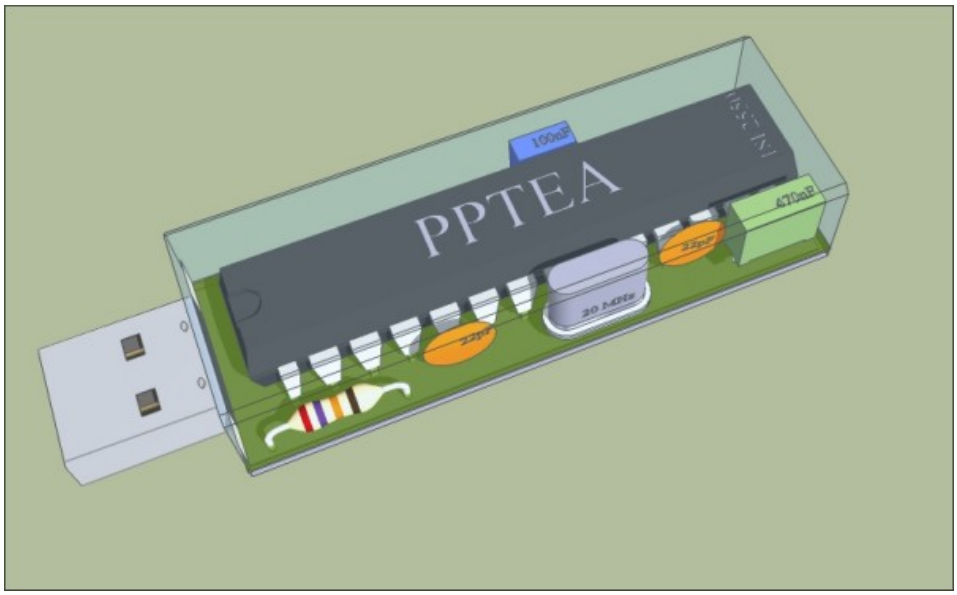
2. PPTEA MINI USB

Questa configurazione è la minima che permette al PPTEA di funzionare quando è collegato alla porta usb del pc. Viene alimentato dal pc stesso e non necessita di nessuna alimentazione esterna, sempre se non vengono utilizzati componenti che superano l'erogazione di corrente della porta usb.





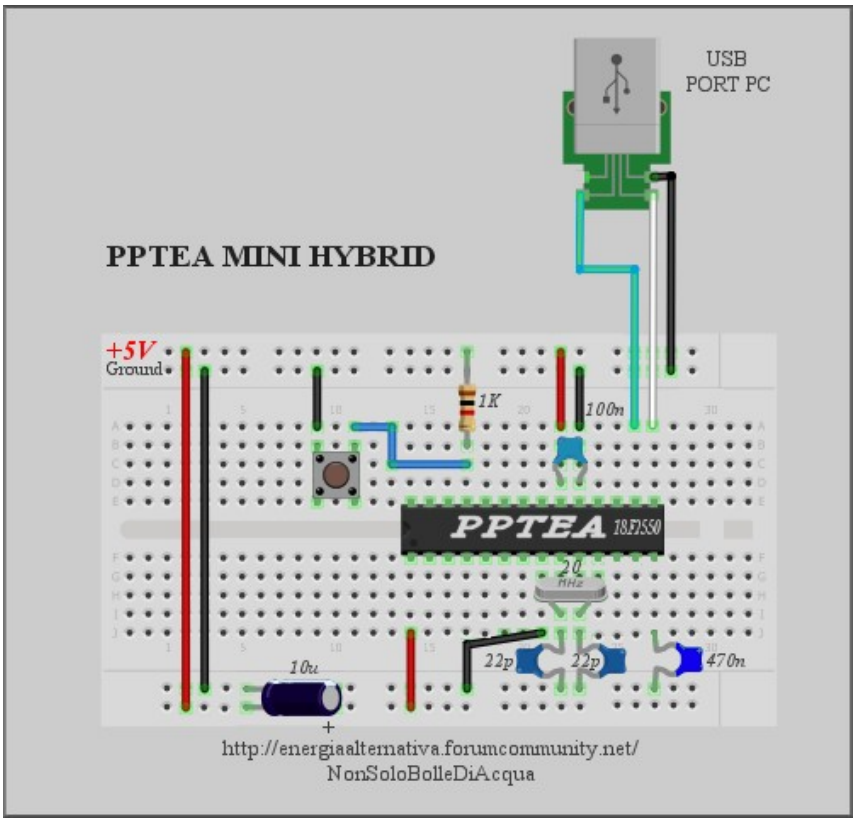

http://energiaalternativa.forumcommunity.net
PPTEA

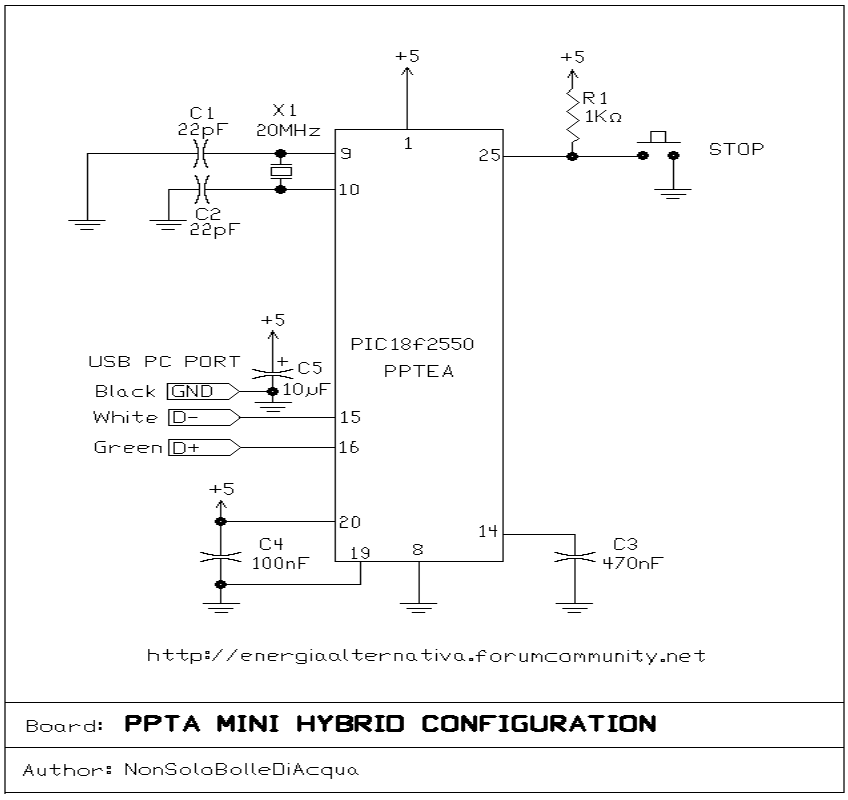

PPTEA
100nF
470nF
22pF
20 MHz

3. PPTEA MINI HYBRID

Questa è la configurazione ibrida minima che permette al PPTEA di funzionare collegato sia ad una alimentazione stabilizzata esterna di +5V che alla porta usb del pc.

Fare attenzione al positivo della porta USB che non è collegato.





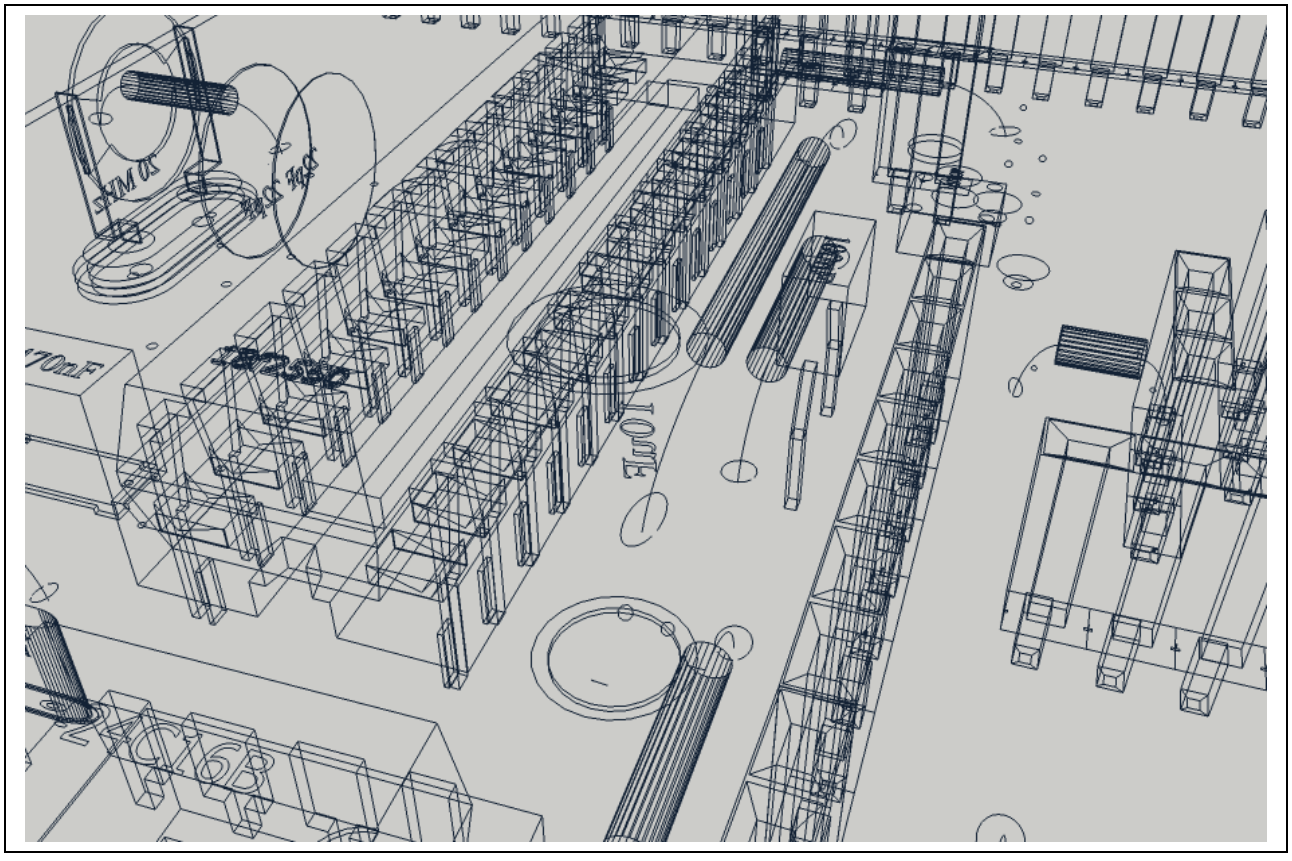
DISPOSITIVI SULLE CONFIGURAZIONI BASE

Ad una delle tre possibili configurazioni (1.STAND ALONE, 2.USB, 3.HYBRID) si possono aggiungere uno o più dei seguenti Dispositivi:

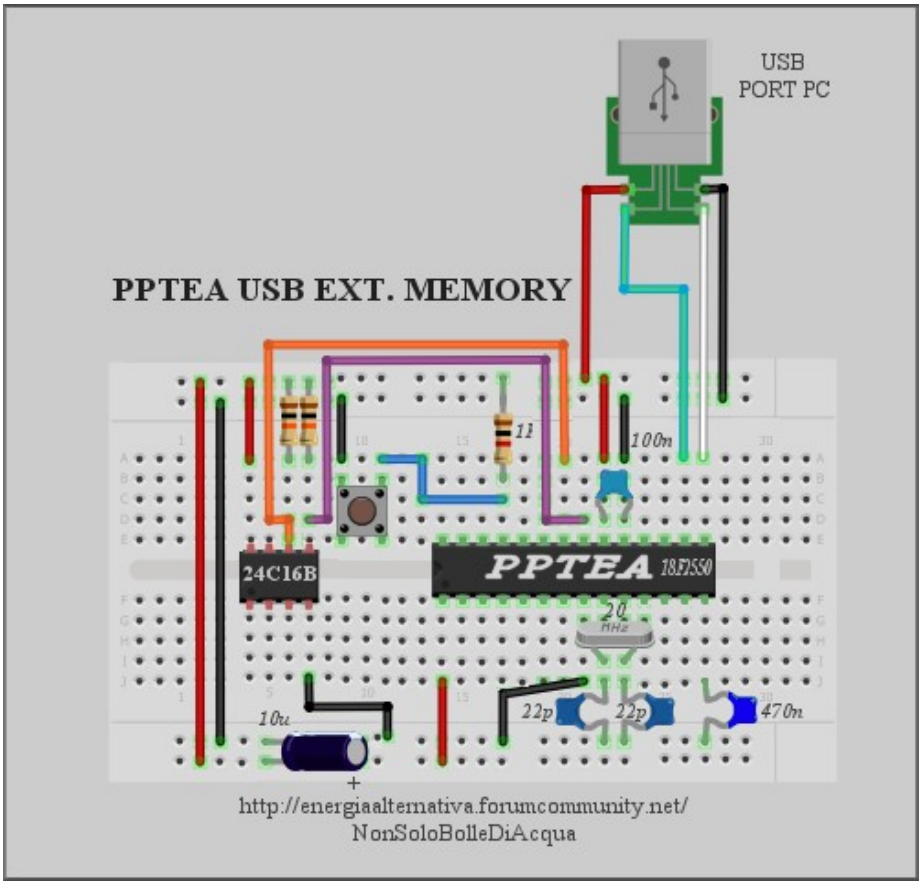
A. Real Time Clock (RTC)
B. Extended Memory(EM)
C. Display (LDC), Display 2 WIRE (LCD2W)
D. WIFI (WF)
E. AUDIO (SOUND)
F. RS232 (RS)
G. Generic Device/Sensor (GDS)
H. WEB SERVER
I. RASPBERRY

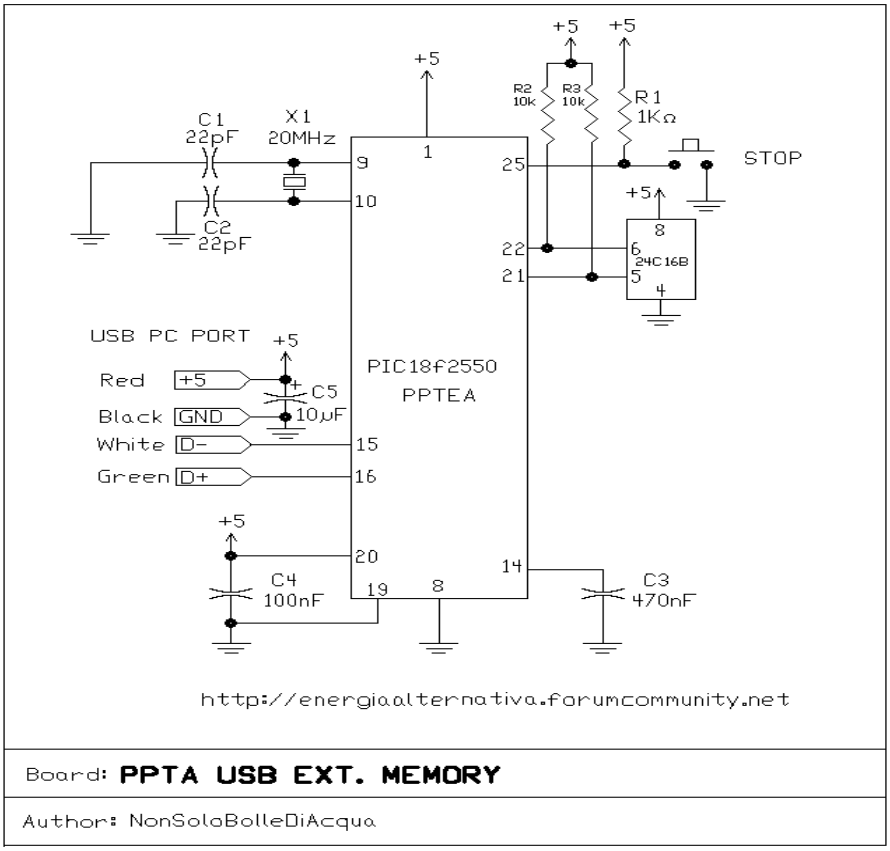
Le possibili implementazioni sono tantissime e rendono il PPTEA molto versatile.
Facciamo degli esempi sulle possibili implementazioni:

1A, 1B, 1C, 1D, 1E, 1AB, 1AC, 1AD, 1AE, 1BC, 1BD, 1BE, 1CD,1CE, 1DE, 1ABC, 1ABD, 1ABE, 1ABCDE...
2A, 2B, 2C, 2D, 2E, 2AB, 2AC, 2AD, 2AE, 2BC, 2BD, 2BE, 2CD,2CE, 2DE, 2ABC, 2ABD, 2ABE, 2ABCDE...
3A, 3B, 3C, 3D, 3E, 3AB, 3AC, 3AD, 3AE, 3BC, 3BD, 3BE, 3CD,3CE, 3DE, 3ABC, 3ABD, 3ABE, 3ABCDE...

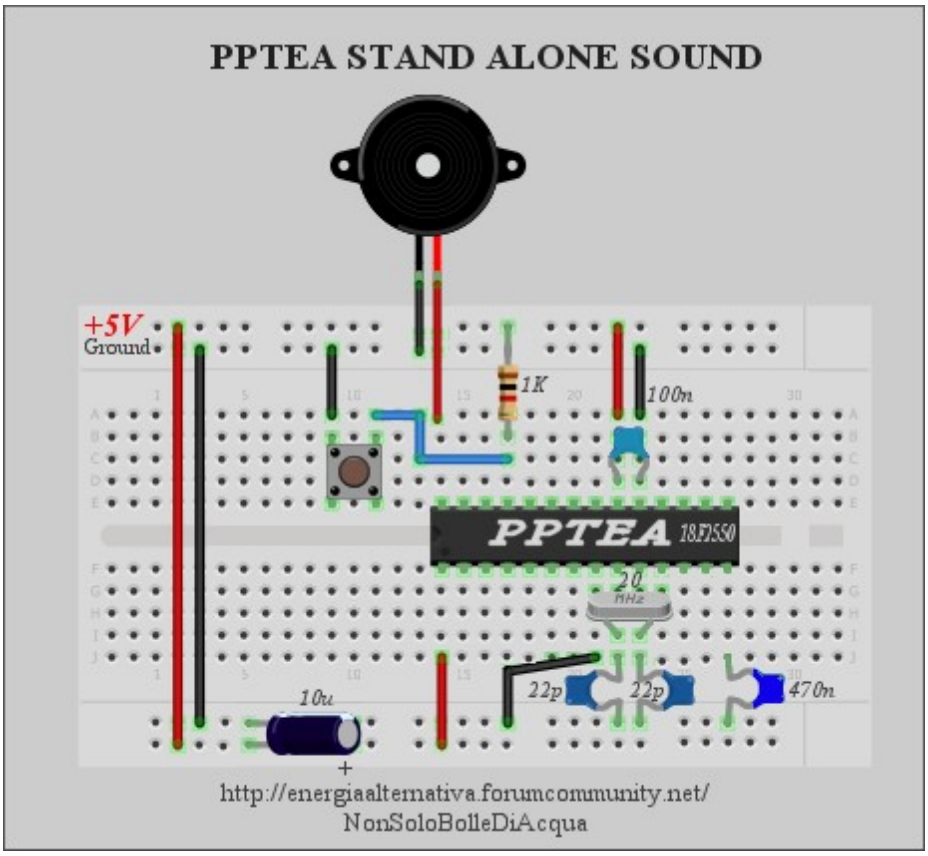


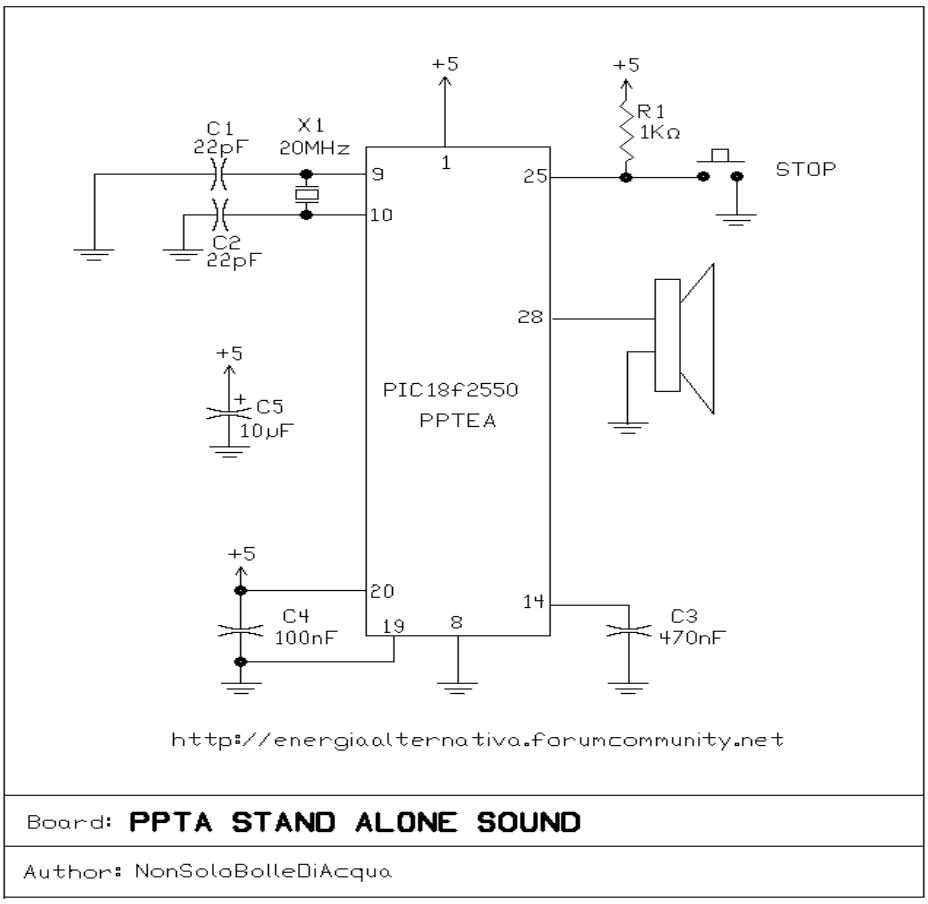
Vediamo l'implementazione ***2B PPTEA USB EM***, cioè la configurazione ***2. PPTEA USB con il dispositivo per l'Espansione di Memoria (B)***.



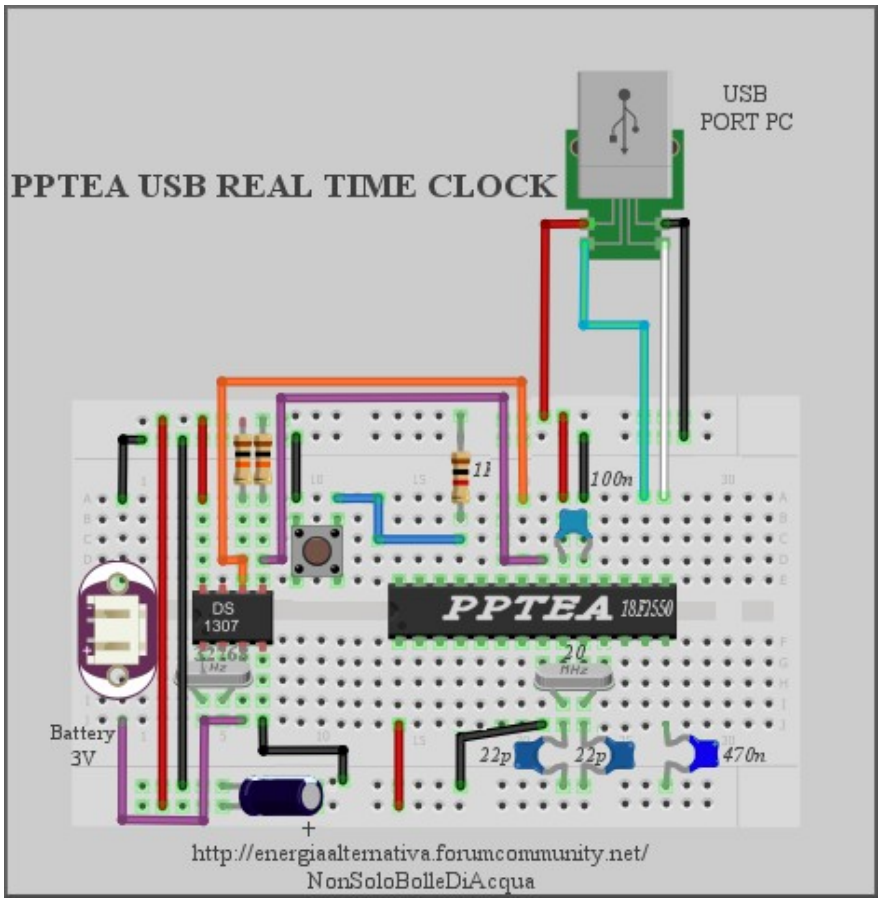


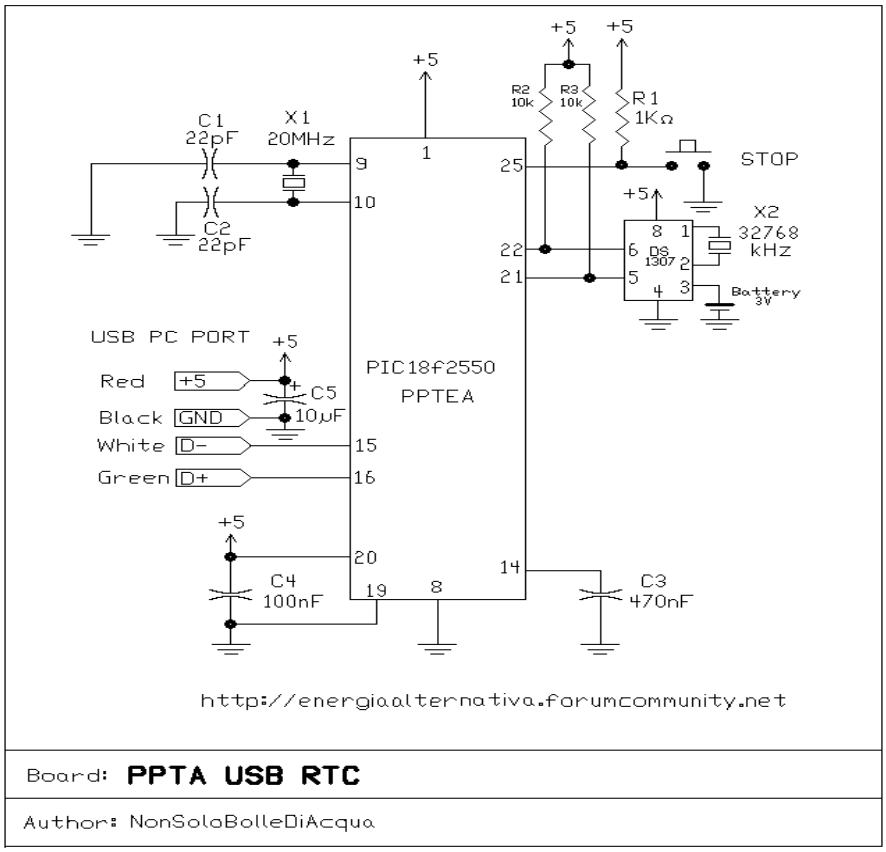
Vediamo l'implementazione ***1E PPTEA STAND ALONE SOUND***, cioè la configurazione ***1. PPTEA STAND ALONE con l'altoparlante piezoelettrico per il suono***.



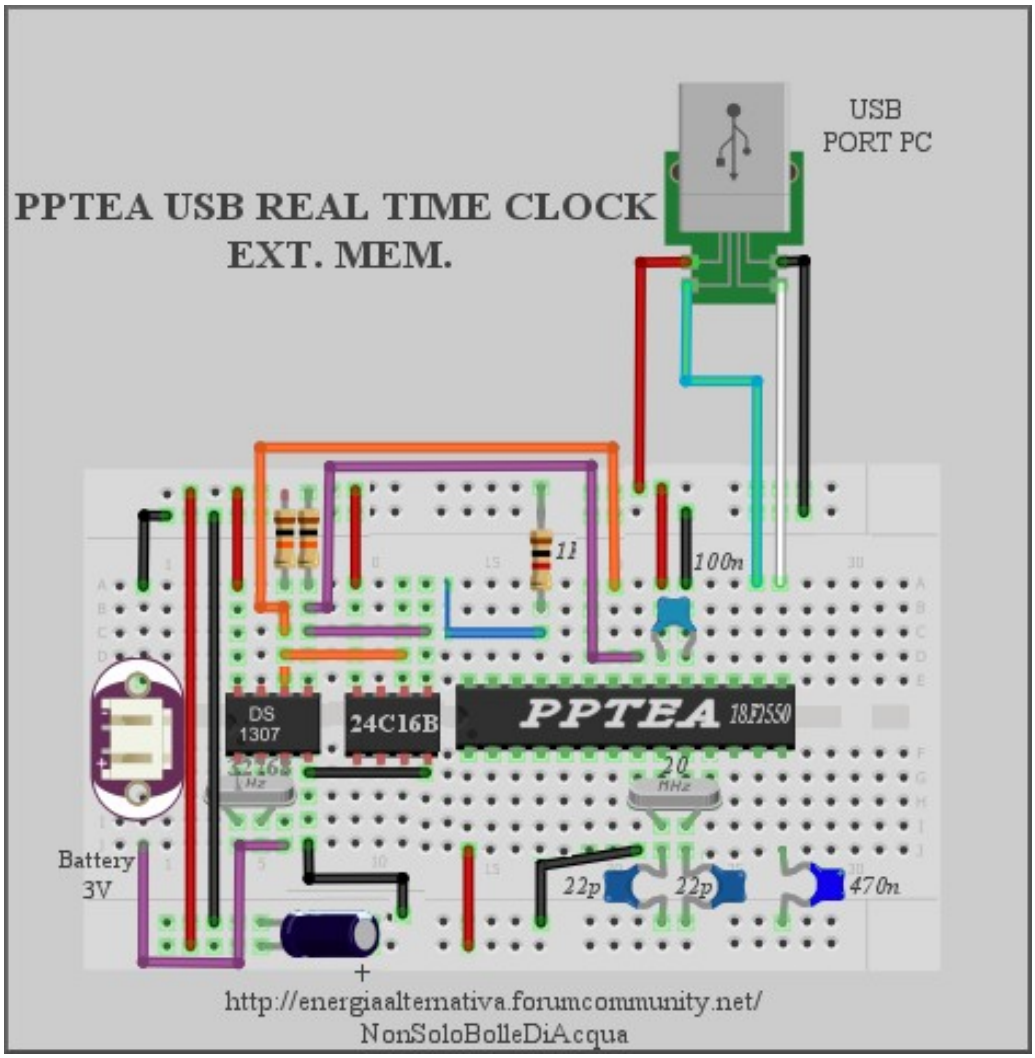


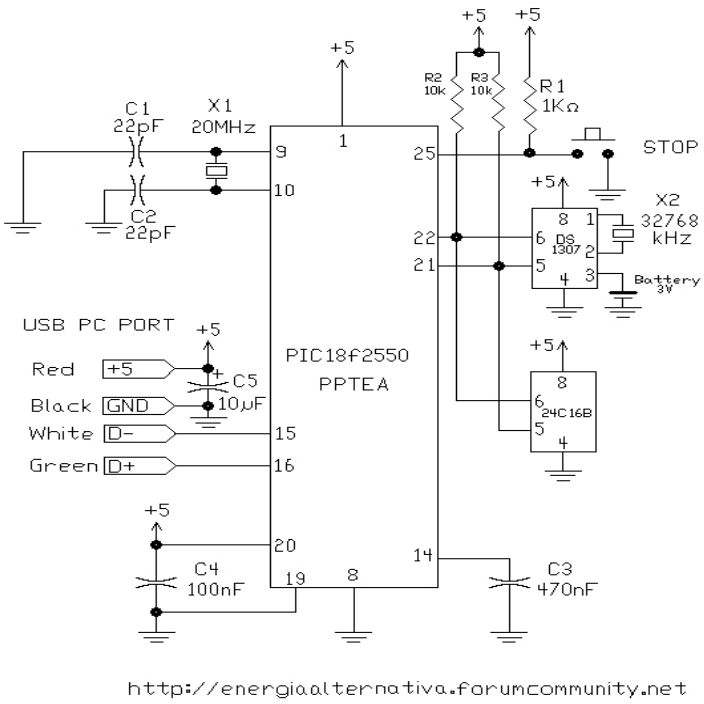
Vediamo l'implementazione **2A PPTEA USB REAL TIME CLOCK**, cioè la configurazione ***2. PPTEA USB con il dispositivo per l'orologio (A)***.





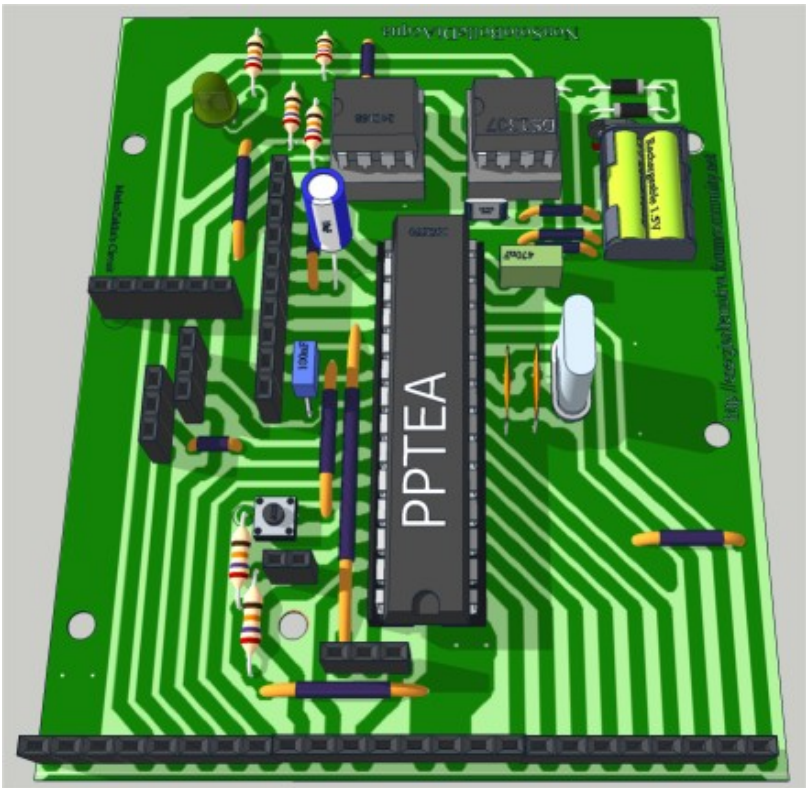
Vediamo l'implementazione **2AB PPTEA USB REAL TIME CLOCK ed ESPANSIONE MEMORIA**, cioè la configurazione ***2. PPTEA USB con il dispositivo per l'orologio (A) e sullo stesso bus I2C la memoria (B)***.



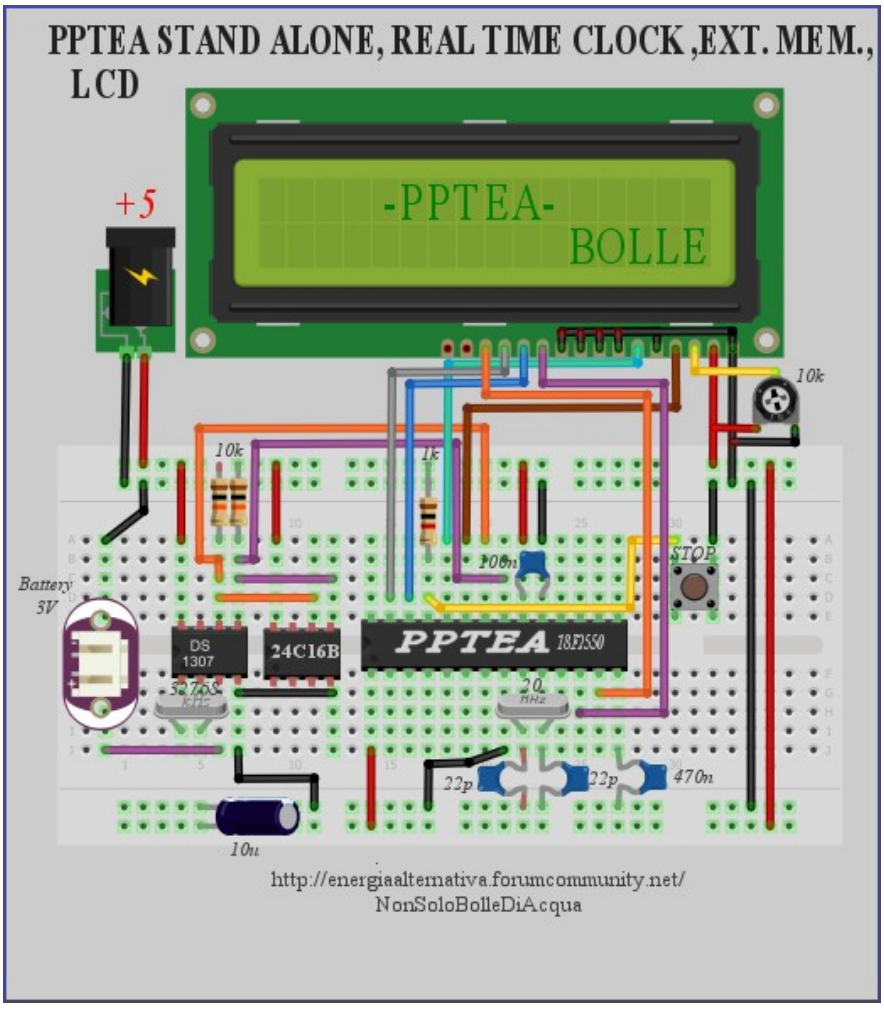


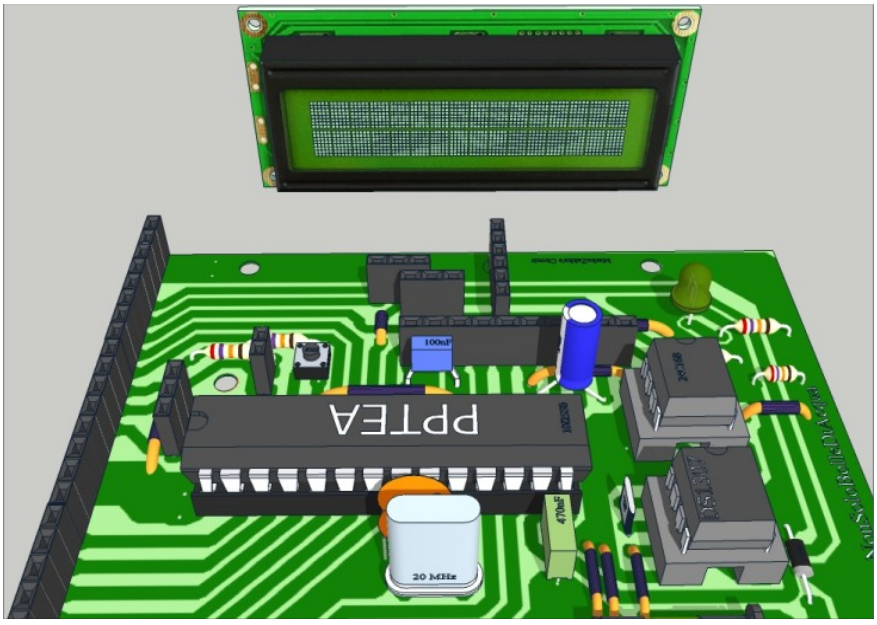
Board: **PPTA USB RTC EXP. MEM**

Author: NonSolaBolleDiAcqua

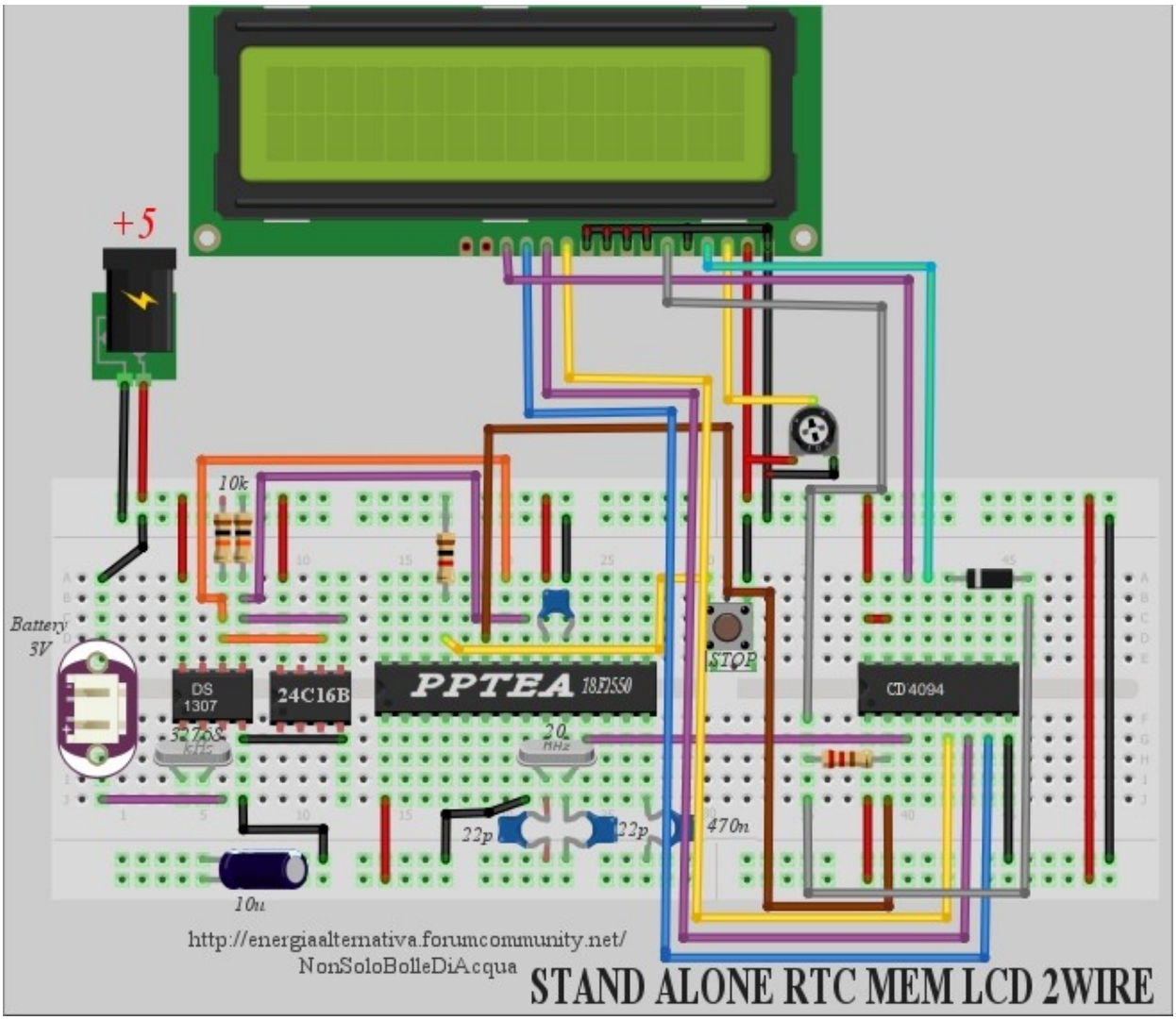


Vediamo l'implementazione **1ABC PPTEA STAND ALONE, REAL TIME CLOCK, ESPANSIONE MEMORIA** e il **DISPLAY**, cioè la configurazione **1. PPTEA** che funge con alimentazione stabilizzata a +5V, con il dispositivo per l'orologio (A), la memoria (B) e il DISPLAY LCD (C).





Vediamo l'implementazione **1ABC PPTEA STAND ALONE, REAL TIME CLOCK, ESPANSIONE MEMORIA** e il **DISPLAY 2 FILI**, cioè la configurazione **1. PPTEA** che funge con alimentazione stabilizzata a +5V, con il dispositivo per l'orologio (A), la memoria (B) e il DISPLAY LCD2W (C).



Vediamo l'implementazione **2ABC PPTEA USB, REAL TIME CLOCK, ESPANSIONE MEMORIA** e il **DISPLAY**, cioè la configurazione **1. PPTEA** che funge con alimentazione della USB con il dispositivo per l'orologio (A), la memoria (B) e il DISPLAY LCD (C).

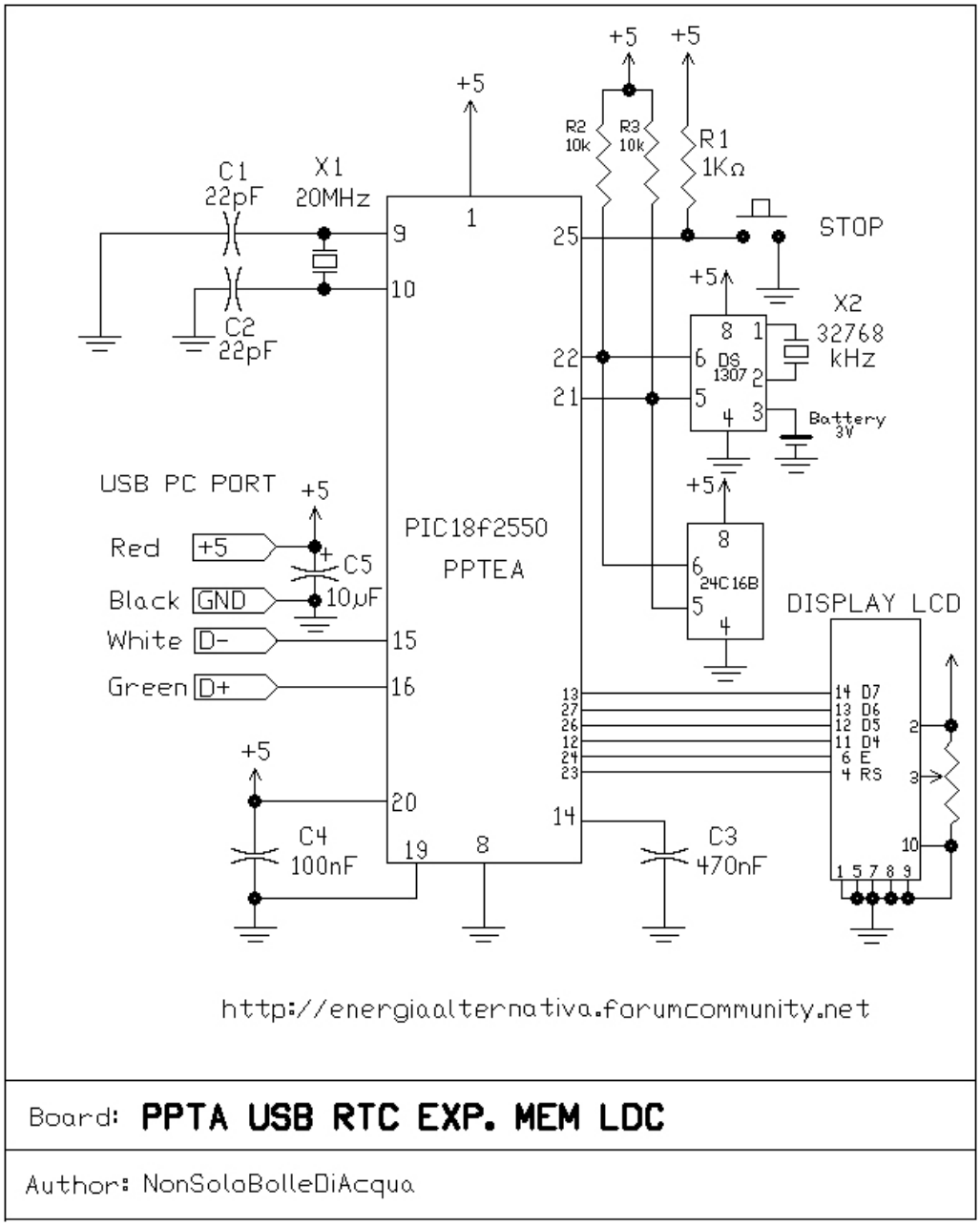


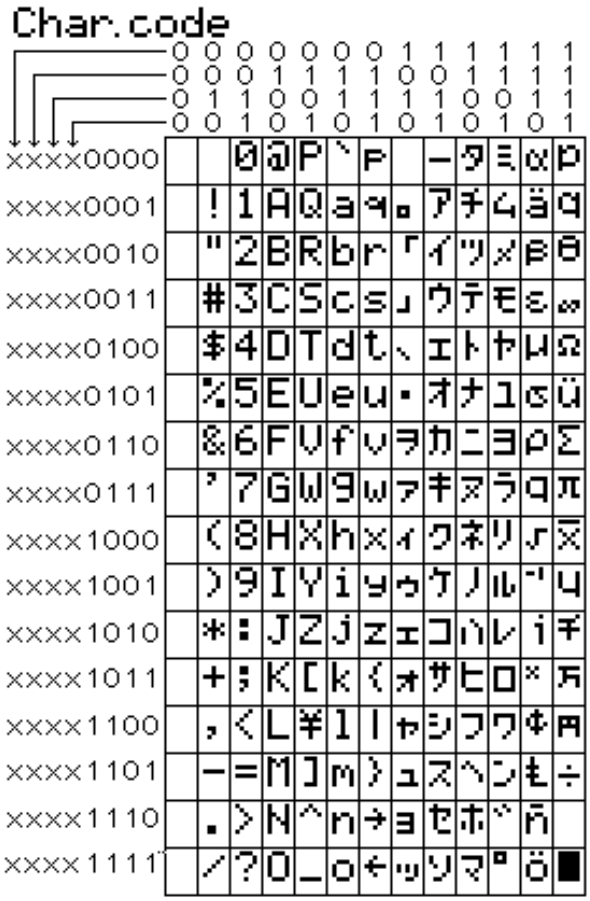
Table ASCII (0 - 127)

Décimal	Octal	Hex	Binaire	Caractère	
000	000	00	00000000	NUL	(Null char.)
001	001	01	00000001	SOH	(Start of Header)
002	002	02	00000010	STX	(Start of Text)
003	003	03	00000011	ETX	(End of Text)
004	004	04	00000100	EOT	(End of Transmission)
005	005	05	00000101	ENQ	(Enquiry)
006	006	06	00000110	ACK	(Acknowledgment)
007	007	07	00000111	BEL	(Bell)
008	010	08	00001000	BS	(Backspace)
009	011	09	00001001	HT	(Horizontal Tab)
010	012	0A	00001010	LF	(Line Feed)
011	013	0B	00001011	VT	(Vertical Tab)
012	014	0C	00001100	FF	(Form Feed)
013	015	0D	00001101	CR	(Carriage Return)
014	016	0E	00001110	SO	(Shift Out)
015	017	0F	00001111	SI	(Shift In)
016	020	10	00010000	DLE	(Data Link Escape)
017	021	11	00010001	DC1	(XON)(Device Control 1)
018	022	12	00010010	DC2	(Device Control 2)
019	023	13	00010011	DC3	(XOFF)(Device Control 3)
020	024	14	00010100	DC4	(Device Control 4)
021	025	15	00010101	NAK	(Negative Acknowledgement)
022	026	16	00010110	SYN	(Synchronous Idle)
023	027	17	00010111	ETB	(End of Trans. Block)
024	030	18	00011000	CAN	(Cancel)
025	031	19	00011001	EM	(End of Medium)
026	032	1A	00011010	SUB	(Substitute)
027	033	1B	00011011	ESC	(Escape)
028	034	1C	00011100	FS	(File Separator)
029	035	1D	00011101	GS	(Group Separator)
030	036	1E	00011110	RS	(Request to Send)
031	037	1F	00011111	US	(Unit Separator)
032	040	20	00100000	SP	(Space)
033	041	21	00100001	!	(exclamation mark)
034	042	22	00100010	"	(double quote)
035	043	23	00100011	#	(number sign)
036	044	24	00100100	$	(dollar sign)
037	045	25	00100101	%	(percent)
038	046	26	00100110	&	(ampersand)
039	047	27	00100111	'	(single quote)
040	050	28	00101000	(	(left opening parenthesis)
041	051	29	00101001	)	(right closing parenthesis)
042	052	2A	00101010	*	(asterisk)
043	053	2B	00101011	+	(plus)
044	054	2C	00101100	,	(comma)
045	055	2D	00101101	-	(minus or dash)
046	056	2E	00101110	.	(dot)
047	057	2F	00101111	/	(forward slash)
048	060	30	00110000	0	
049	061	31	00110001	1	
050	062	32	00110010	2	
051	063	33	00110011	3	
052	064	34	00110100	4	

053	065	35	00110101	5	
054	066	36	00110110	6	
055	067	37	00110111	7	
056	070	38	00111000	8	
057	071	39	00111001	9	
058	072	3A	00111010	:	(colon)
059	073	3B	00111011	;	(semi-colon)
060	074	3C	00111100	<	(less than sign)
061	075	3D	00111101	=	(equal sign)
062	076	3E	00111110	>	(greater than sign)
063	077	3F	00111111	?	(question mark)
064	100	40	01000000	@	(AT symbol)
065	101	41	01000001	A	
066	102	42	01000010	B	
067	103	43	01000011	C	
068	104	44	01000100	D	
069	105	45	01000101	E	
070	106	46	01000110	F	
071	107	47	01000111	G	
072	110	48	01001000	H	
073	111	49	01001001	I	
074	112	4A	01001010	J	
075	113	4B	01001011	K	
076	114	4C	01001100	L	
077	115	4D	01001101	M	
078	116	4E	01001110	N	
079	117	4F	01001111	O	
080	120	50	01010000	P	
081	121	51	01010001	Q	
082	122	52	01010010	R	
083	123	53	01010011	S	
084	124	54	01010100	T	
085	125	55	01010101	U	
086	126	56	01010110	V	
087	127	57	01010111	W	
088	130	58	01011000	X	
089	131	59	01011001	Y	
090	132	5A	01011010	Z	
091	133	5B	01011011	[	(left opening bracket)
092	134	5C	01011100	\	(back slash)
093	135	5D	01011101	]	(right closing bracket)
094	136	5E	01011110	^	(caret cirumflex)
095	137	5F	01011111	_	(underscore)
096	140	60	01100000	`	
097	141	61	01100001	a	
098	142	62	01100010	b	
099	143	63	01100011	c	
100	144	64	01100100	d	
101	145	65	01100101	e	
102	146	66	01100110	f	
103	147	67	01100111	g	
104	150	68	01101000	h	
105	151	69	01101001	i	
106	152	6A	01101010	j	
107	153	6B	01101011	k	
108	154	6C	01101100	l	
109	155	6D	01101101	m	
110	156	6E	01101110	n	
111	157	6F	01101111	o	
112	160	70	01110000	p	
113	161	71	01110001	q	
114	162	72	01110010	r	

115	163	73	01110011	s	
116	164	74	01110100	t	
117	165	75	01110101	u	
118	166	76	01110110	v	
119	167	77	01110111	w	
120	170	78	01111000	x	
121	171	79	01111001	y	
122	172	7A	01111010	z	
123	173	7B	01111011	{	(left opening brace)
124	174	7C	01111100	|	(vertical bar)
125	175	7D	01111101	}	(right closing brace)
126	176	7E	01111110	~	(tilde)
127	177	7F	01111111	DEL	(delete)

Table ASCII Display HITACHI



COSTANTI PREDEFINITE

Costante	*Valore*	*Significato*
CR_LF	***chr(13) & chr(10)***	*Carriage Return & Line Feed (a capo automatico)*
CR_LF_RASP	***chr(10)***	*Carriage Return (a capo automatico)*
CAD_TO_VOLT	***.004887***	Cad To Volt (5/1023)
CAD_TO_TEMP	***.4887***	Cad To Temp Lm35 (5/1023*100)
MAX_INT	***2147483647***	Valore intero massimo
MIN_INT	***-2147483648***	Valore intero minimo
BLANK	***" "***	Spazio
BLANK16	***" "***	16 Spazi
ZERO	***0***	zero
ONE	***1***	Uno
TRUE	***1***	Vero
FALSE	***0***	Falso
INPUT	***1***	Valore bit setio
OUTPUT	***0***	Valore bit setio
LOW	***0***	Valore bit livello basso
HIGH	***1***	Valore bit livello alto
BIT_0	***0***	Bit 0
BIT_1	***1***	Bit 1
BIT_2	***1***	Bit 2
BIT_3	***1***	Bit 3
BIT_4	***1***	Bit 4
BIT_5	***1***	Bit 5
BIT_6	***1***	Bit 6
BIT_7	***1***	Bit 7
BIT_8	***1***	Bit 8
BIT_9	***1***	Bit 9
BIT_10	***10***	Bit 10
BIT_11	***11***	Bit 11
BIT_12	***12***	Bit 12
BIT_13	***13***	Bit 13
BIT_14	***14***	Bit 14
BIT_15	***15***	Bit 15
BIT_0_VALUE	***1***	Valore Bit 0
BIT_1_VALUE	***2***	Valore Bit 1
BIT_2_VALUE	***4***	Valore Bit 2
BIT_3_VALUE	***8***	Valore Bit 3
BIT_4_VALUE	**16**	Valore Bit 4
BIT_5_VALUE	***32***	Valore Bit 5
BIT_6_VALUE	***64***	Valore Bit 6
BIT_7_VALUE	***128***	Valore Bit 7
BIT_8_VALUE	***256***	Valore Bit 8
BIT_9_VALUE	***512***	Valore Bit 9
BIT_10_VALUE	***1024***	Valore Bit 10
BIT_11_VALUE	***2048***	Valore Bit 11
BIT_12_VALUE	***4096***	Valore Bit 12
BIT_13_VALUE	***8192***	Valore Bit 13
BIT_14_VALUE	***16384***	Valore Bit 14
BIT_15_VALUE	***32768***	Valore Bit 15
LCD4X20_ROW_1	***1***	PRIMA RIGA DISPLAY 20X4
LCD4X20_ROW_2	***65***	SECONDA RIGA DISPLAY 20X4

LCD4X20_ROW_3		***21***	TERZA RIGA DISPLAY 20X4
LCD4X20_ROW_4		***85***	QUARTA RIGA DISPLAY 20X4
PAUSE	***0***		Attesa commando sonoro
_DO	***956***		Periodo quarti del Do
_DO#	***902***		Periodo quarti del Do#
_RE	***851***		Periodo quarti del Re
_MIb	***804***		Periodo quarti del MIb
……..	……		……..
_SI	***506***		Periodo quarti del SI
	Prima ottava		
_DO0	***3822***		Periodo quarti del Do della prima ottava
_DO#0	***3608***		Periodo quarti del Do# della prima ottava
_RE0	***3405***		Periodo quarti del Re della prima ottava
_MIb0	***3214***		Periodo quarti del MIb della prima ottava
……..	……		……..
_SI0	***2024***		Periodo quarti del SI della prima ottava
	Seconda ottava		
_DO1	***1911***		Periodo quarti del Do della seconda ottava
_DO#1	***1804***		Periodo quarti del Do# della seconda ottava
_RE1	***1703***		Periodo quarti del Re della seconda ottava
_MIb1	***1607***		Periodo quarti del MIb della seconda ottava
……..	……		……..
_SI1	***1013***		Periodo quarti del SI della seconda ottava
……..	……		……..
……..	……		……..
……..	……		……..
……..	……		……..
……..	……		……..
	Settima ottava		
_DO6	***60***		Periodo quarti del Do della settima ottava
_DO#6	***56***		Periodo quarti del Do# della settima ottava
_RE6	***53***		Periodo quarti del Re della settima ottava
_MIb6	***50***		Periodo quarti del MIb della settima ottava
……..	……		……..
_SI6	***32***		Periodo quarti del SI della settima ottava

VELOCITA' SERIALE RS232

SPEED_57600	**12**	Imposta la velocità della seriale al baud-rate specificato
SPEED_38400	**19**	**“ ”**
SPEED_19200	**40**	**“ ”**
SPEED_9600	**80**	**“ ”**
SPEED_4800	**160**	**“ ”**

COSTANTI REAL TIME CLOCK (FDATE)

DATE_AND_TIME	**0**	gg/mm/aa hh:mm:ss
TIME_ONLY	**1**	hh:mm:ss
DATE_ONLY	**2**	gg/mm/aa
DATE_TIME_DAY	**3**	gg/mm/aa gds hh:mm:ss
SECONDS_DAY	**4**	**ss** (Secondi trascorsi dalla mezzanotte)
SECONDS_WEEK	**5**	**ss** (Secondi trascorsi dall’inizio della settimana)
SECONDS_MONTH	**6**	**ss** (Secondi trascorsi dall’inizio del mese)

VOLUME ALTOPARLANTE

VOLUME_0	**0**	(Volume minino)
VOLUME_1	**1**	
VOLUME_2	**2**	
VOLUME_3	**4**	
VOLUME_4	**8**	
VOLUME_5	**16**	
VOLUME_6	**32**	
VOLUME_7	**60**	
VOLUME_8	**80**	
VOLUME_9	**100**	(Volume massimo)

COSTANTI CON BASE DIVERSA DAL DECIMALE

E’ possibile scrivere costanti in formato Binario ed Esadecimale fino a 4 BYTE. La costante esadecimale (ammette valori numerici con le lettere che vanno da A ad F) viene scritta con & seguita da una H. La costante binaria (ammette solo valori 0 o 1) viene scritta con & seguita da una B.

Esempio costante esadecimale:
&HF0 -> Valore decimale 240

Esempio costante binaria:
&B11110001 -> Valore decimale 241

OPERATORI MATEMATICI/STRINGHE

op1 + op2	*Somma i due operandi.*
op1 - op2	*Sottrae dal primo il secondo operando.*
op1 * op2	*Moltiplica i due operandi.*
op1 / op2	*Divide il primo operando per il secondo.*
op1 AND op2	*AND Binario tra primo operando e il secondo.*
op1 OR op2	*OR Binario tra il primo operando e il secondo.*
op1 XOR op2	*XOR Binario tra il primo operando e il secondo.*
NOT op1	*NOT Binario*
op1 MOD op2	*Modulo tra il primo e secondo operando*
var++	*Incrementa di 1 la variabile var*
var--	*Decrementa di 1 la variabile var*
var+=op1	*Somma op1 alla variabile var*
var-=op1	*Sottrae op1 alla variabile var*
var/=op1	*Divide var per op1*
var*=op1	*Moltiplica Var per op1*
op1 = op2	*Uguale*
op1 >= op2	*Maggiore uguale*
op1 <= op2	*Minore uguale*
op1 <> op2	*Diverso*
(	*Parentesi Aperta*
)	*Parentesi Chiusa*
op1 AND op2	*And logico*
op1 OR op2	*OR logico*
+	*Operatore Unario*
-	*Operatore Unario*
!	*Not logico*
&	*Somma due stringhe*

PRIORITA' OPERATORI MATEMATICI

Ogni operatore ha una priorità, viene eseguito prima quello a priorità più alta.

Operatore	Priorità
OR	5
AND	6
=	7
<>	7
>=	8
<=	8
>	8
<	8
+	9
-	9
&	9
*	10
/	10
MOD	10
XOR	10
-(unario)	11
+(unario)	11
NOT (unario)	11

Esempi:

A=50-3+6+9	*‘ A vale 62*	A= 1+5*5-1<=33-3*3+1	*‘ A vale 1 (o -1)*
A=3+5*15	*‘ A vale 78*	A= 1+5*5-1<=33-3*3-1	*‘ A vale 0*
A=10-3-7+20-5-5	*‘ A vale 10*	A=1257*-9	*‘ A vale -11313*
A=5+3*2*7-3*5	*‘ A vale 32*	A=float(1000+5*4+4)	*‘ A vale 1024.0*
A=25+37/1024.0*5 -10 +3*2	*‘ A vale 21,181*	A=37/float(1014+3*2+4)*5-10 +3*2	*‘ A vale -3,81933*
A= 5 or 7	*‘ A vale 7*	A=5*(3+2)	*‘ A vale 25*
A= 1+5*5-1=33-3*3+1	*‘ A vale 1 (o -1)*	A= ((155 mod 20)*1+1)+3	*‘ A vale 19*
A= 1+5*5-1=33-3*3	*‘ A vale 0*	A="22.5" & chr(223) &"C"	*‘ A vale “22.5°C”*
A= 5*2 mod 3 +1	*‘ A vale 2*	A=left(“25.4343”,4)	*‘ A vale “25.4”*