Guida all' IDE di Bascom-AVR

Integrated Development Enviroment per uC ATMEL serie AVR (AT90, ATtiny, Atmega, ATxmega)

Giovanni De Luca

www.delucagiovanni.com
deluca@lns.infn.it

Alcuni Compilatori e sistemi di sviluppo presenti sul mercato per i Microcontrollori AVR

- *AVRStudio4 della ATMEL*
- *AVR-GCC Free GNU*
- *CodeVision AVR*
- *ICC-AVR della ImageCraft*
- *FastAVR*
- *mikroBasic for AVR*
- ***BASCOM-AVR della MCSELEC***

Dove scaricare la documentazione e le Demo

www.delucagiovanni.com > Docs > Archimede

Il sito ufficiale Bascom:

www.mcselec.com

Il sito ufficiale Atmel:

www.atmel.com

Download DEMO e Utility

- Download Demo
- Manuale 2.0.7.3 in Inglese
- Help
- AVR Calculator
- PWM Calculator
- AVR Designer

Perché il BASCOM ?

- *Linguaggio ad alto livello, vicino al PLM-51 – è possibile scrivere anche in "asm" per creare librerie personalizzate compatte distribuibili e criptate*
- *Di facile intuizione – Sintassi dei comandi tra il "C standard" e il "BASIC"*
- *Linguaggio ad esecuzione sequenziale e strutturato con subroutine e funzioni*
- *Velocità nella produzione di firmware robusto ed affidabile (basso Time to Market)*
- *Grandissima varietà di micro AVR supportati dal compilatore 8/16/32 bit*
- *Presente sul mercato da oltre 15 anni*
- *Costo non elevato del pacchetto professionale*
- *Librerie disponibili per la gestione di periferiche: GLCD, ADC, DAC, IO ext, MEM, TCP-IP, USB*
- *Ottimizzazione del codice nativo grazie alla compilazione a passaggi multipli*
- *Possibilità di implementare un FileSystem compatibile DOS-FAT16/32*
- *Gestione diretta di HARDISK IDE, SD-CARD, e altri dispositivi simili*
- *Versioni del Software in continuo aggiornamento e implementazione nuovi comandi*
- *Possibile emulazione del firmware prodotto, con AVRStudio o PROTEUS-ISIS*
- *Moltissimi lavori professionali prodotti con questo pacchetto software*
- *Molta documentazione in rete e vari esempi esaustivi*
- *Forum attivo sul sito www.MSCELEC.com (registrazione richiesta)*
- *Supporto on-line professionale 24H24 (a pagamento)*

Perché AVR con Bascom

da una fonte autorevole

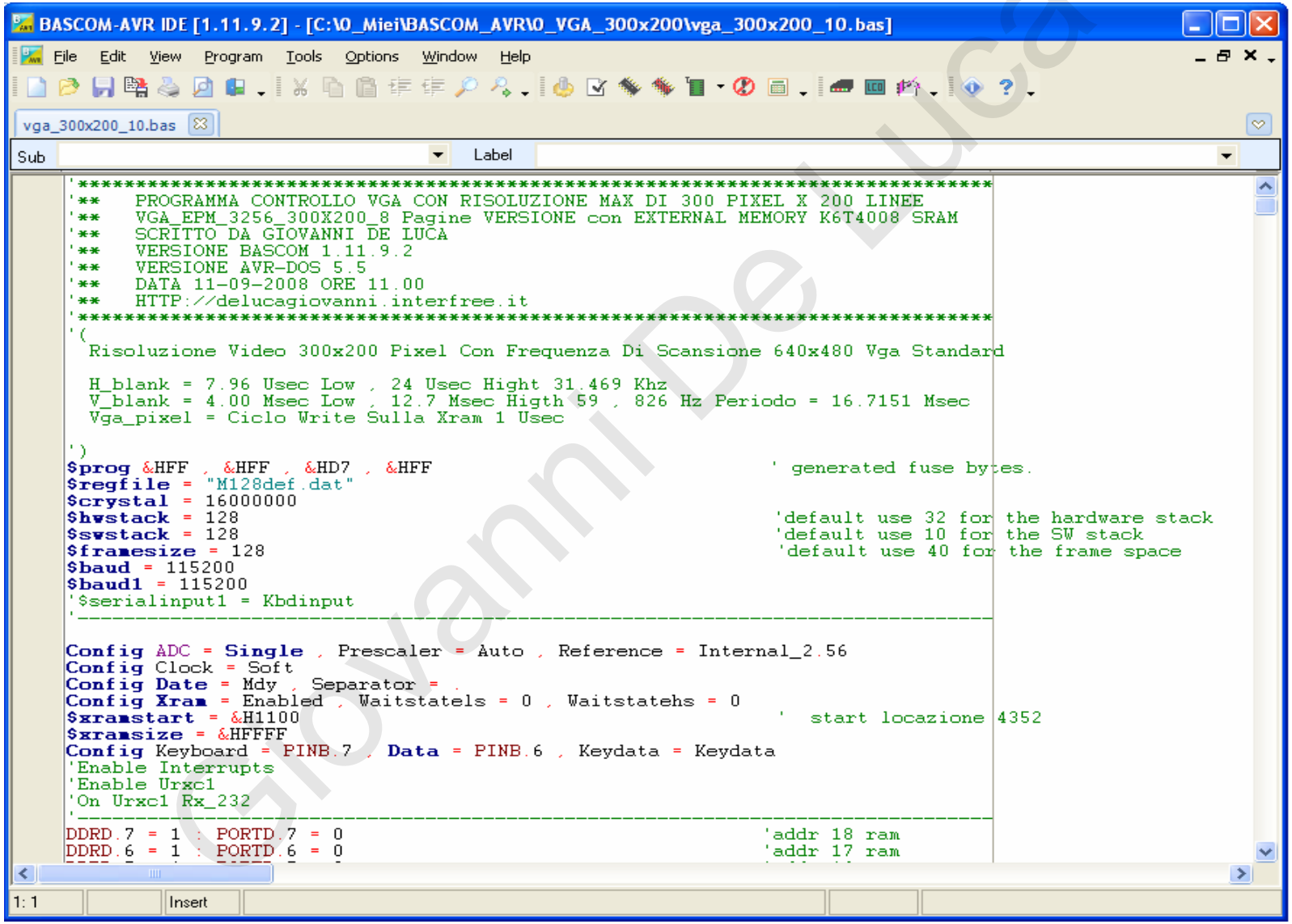
- *AVR is a family of 8-bit microcontrollers with a large range of variants differing in:*
  - size of program-memory (flash)
  - size of eeprom memory
  - number of I/O pins
  - number of on-chip features such as uart and adc
  - package forms

- *The smallest microcontroller is the ATTiny11 with 1k flash and 6 I/O pins. The largest is the ATxMEGA256x with 256k flash, 54 I/O pins and lots of on-chip features. All AVR controllers have the same RISC-like instruction set, enabling fairly easy porting of Bascom programs between microcontroller types.*
- ***(Tutti i controllori AVR hanno lo stesso set di istruzioni RISC-like, che consente il porting abbastanza facile dei programmi di Bascom tra i tipi di microcontroller)***

- *They execute one instruction per clock-cycle making them appreciably faster than comparable 8-bit 4 clock-cycles-per-instruction Microchip PIC controllers .*

  (Essi eseguono una sola istruzione per ciclo di clock li rende sensibilmente più veloce rispetto ai tradizionali 8-bit 4 cicli di clock-per-istruzione Microchip PIC controller)

L’ambiente di sviluppo IDE



“C” vs Bascom ‘Blink’

```
int ledPin = 13;
void setup()
    {
      pinMode(ledPin, OUTPUT);
    }

void loop()
    {
      digitalWrite(ledPin, HIGH);
      delay(1000);
      digitalWrite(ledPin, LOW);
      delay(1000);
    }
```

'1112 bytes

```
$Include "DuinoLib.inc"
Config Pin13 = Output

Do
  Set Out13
  WaitmS 1000
  Reset Out13
  WaitmS 1000
Loop
```

'230 bytes

“C” vs bascom 'Smoothing'

```
#define NUMREADINGS 10
int readings[NUMREADINGS];
int index = 0;
int total = 0;
int average = 0;
int inputPin = 0;
void setup()
      {
        Serial.begin(9600);
        for (int i = 0; i < NUMREADINGS;i++)
         readings[i] = 0;
      }

void loop()
      {
        total - = readings[index];
        readings[index] =
      analogRead(inputPin);
        total + = readings[index];
        index =(index + 1);
        if(index >= NUMREADINGS)
          index = 0;
           average = total / NUMREADINGS;
        Serial.println(average);
      }
```

'2,838 bytes

```
$Include "DuinoLib.inc"
$External AnaRead
Dim Index as Byte , Value As Word
Dim Total As Word , Average as Word
Dim Readings(10) As Word

Const NumReadings = 10
Index = 1

Do
   Total = Total - Readings(Index)
   AnaRead 5 , Value
   Readings(Index) = Value
   Total = Total + Value
   Incr Index
   If Index > NumReadings Then Index = 1
   Average = Total / NumReadings
   Print Average
Loop
```

'870 bytes

Quante istruzioni dobbiamo conoscere per scrivere un Programma FW ?

- *133 - Istruzioni Assembly AVR*
- *405 - Istruzioni, nel Bascom AVR nella versione 2.0.5.0*

Primi passi con il BAS-AVR

Esempio di configurazione base del dispositivo ATMEGA128

```
$prog &HFF , &HFF , &HD7 , &HFF
$regfile = "M128def.dat"
$crystal = 16000000
$hwstack = 128
$swstack = 128
$framesize = 128
$baud = 115200
$baud1 = 115200
```

La potenza del compilatore BAS-AVR

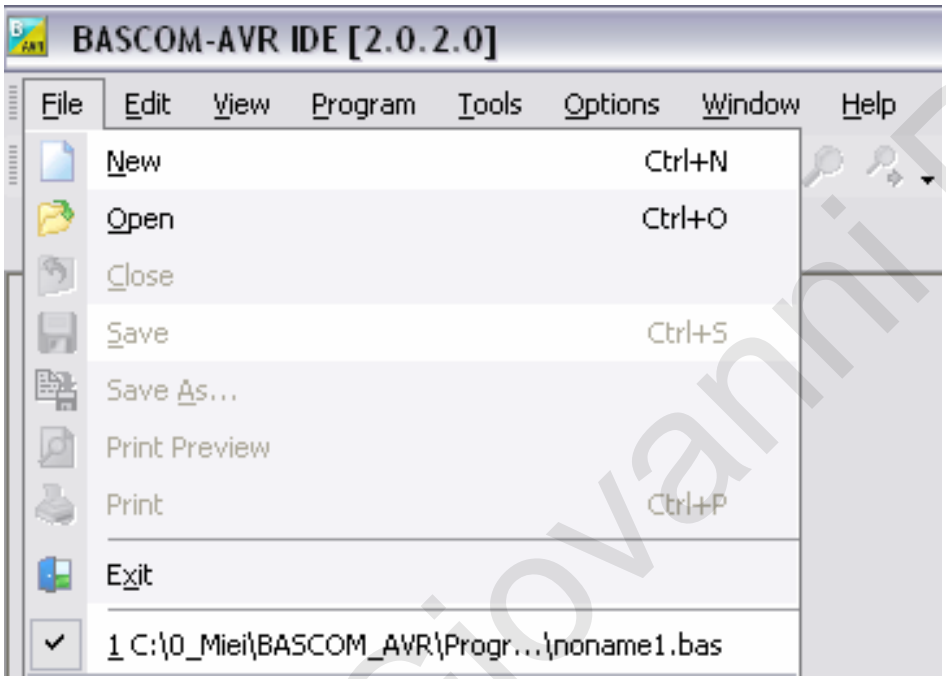
```
DIM X as byte
    Print X
End
```

Linguaggio Assembly

```
        .DSEG
.ORG    256
x:                      .Byte    1

        .CSEG
.ORG    0

_Reset:
        ldi     yl,Low(RAMEND)
        out     SPL,yl
        ldi     yh,high(RAMEND)
        out     SPL+1,yh
        sbiw    yl,32
        ldi     zl,0x18
        out     UCSR0B,zl
        ldi     zh,high(8)
        ldi     zl,Low(8)
        out     UBRR0L,zl
        sts     UBRR0H,zh


;****** USERS BASIC CODE **********************

;-Line--0013----Print x--
        lds     zl,x
        call    _B2Str
        call    _PrBW
        call    _PCL

;-Line--0014----End--
L0000:
        jmp L0000

;****** End OF USER BASIC CODE *****************
```

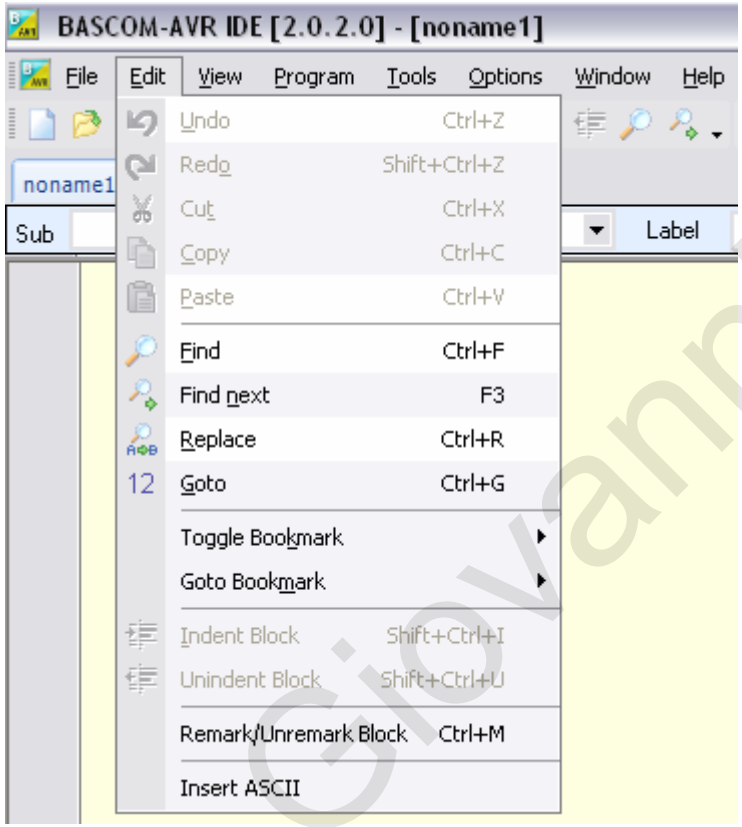
```
;////// Print Byte & Word /////////////////
_PrBW:  ld      r24,Z+
        tst     r24
        breq    _PBW1
        rcall   _Pch
        rjmp    _PrBW
_PBW1:  ret

;////// Print Cr, Lf & any char//////////////
_PCL:   ldi     r24,0x0d
        rcall   _Pch
        ldi     r24,0x0a
_Pch:   sbis    UCSR0A,5
        rjmp    _Pch
        out     UDR0,r24
        ret

;////// ByteToStr /////////////////////////
_B2str: clr     zh
        clt
        push    yl
        push    yh
        st      -Y,zh
_N2str: ldi     r21,0x08
        Sub     r22,r22
_N2st1: lsr     zh
        rol     zl
        rol     r22
        rol     zh
        cpi     r22,0x0a
        brcs    _N2st2
        sbci    r22,0x0a
        inc     zl
_N2st2: dec     r21
        brne    _N2st1
        subi    r22,-0x30
        st      -Y,r22
        tst     zl
        brne    _N2str
_N2st3: mov     zl,yl
        mov     zh,yh
        pop     yh
        pop     yl
        ret

;System Global Variables: 0  bytes
;User   Global Variables: 1  bytes
```

Menù File



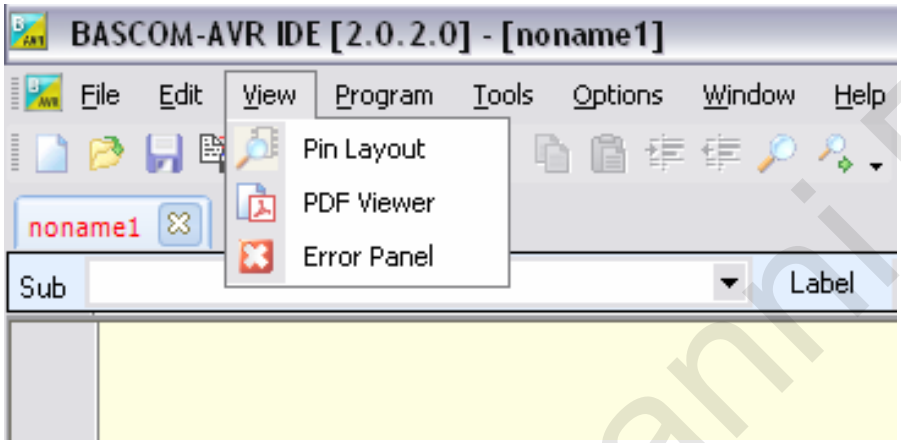
- Classico menù '**File**' di Windows.
- Abbiamo la possibilità di creare un nuovo file, di aprirne uno esistente o di salvarlo.
- Menù di stampa con preview e uscita dal programma**.**

Menù Edit



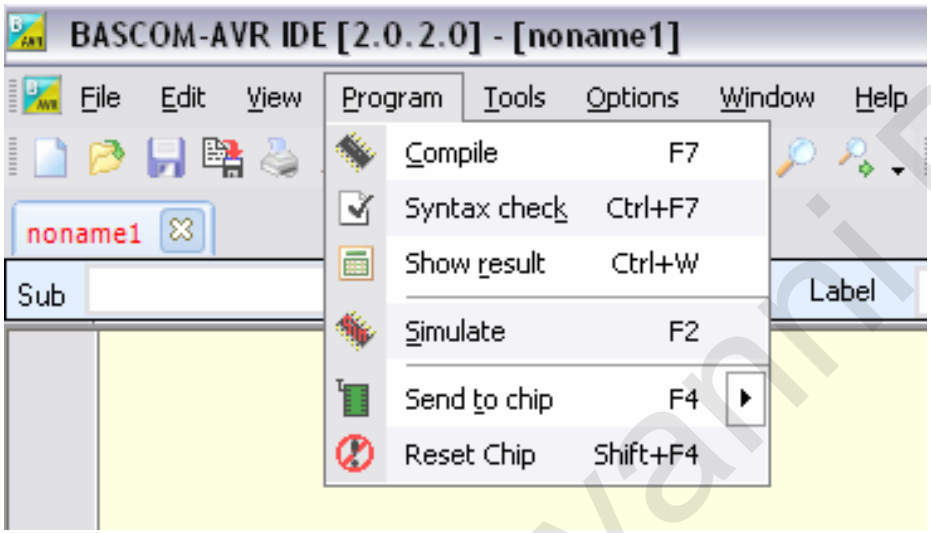
- Classico '**Edit**' di Windows.
- Copy and Paste
- Find and replace
- Bookmark

Menù View



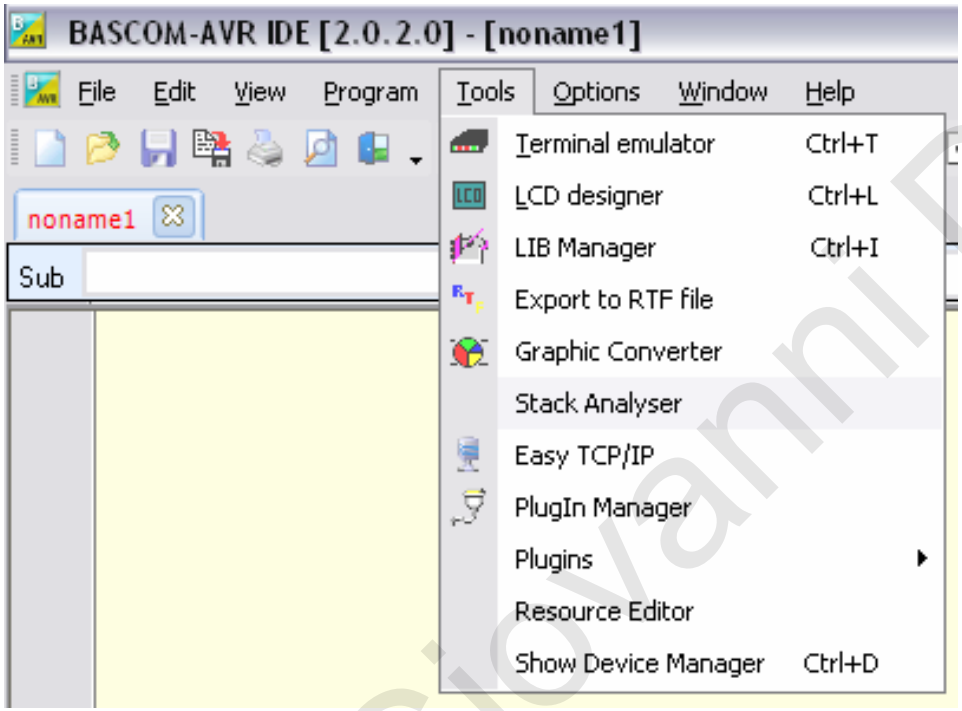
- Con '**View**' è possibile abilitare la visualizzazione dei pin del chip selezionato.
- E' possibile fare un link al pdf del chip usato nel progetto

Menù Program



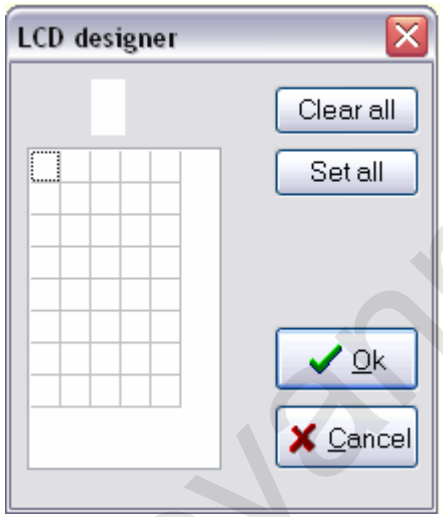
- Da questo menù si accede alla compilazione, al syntax check, alla simulazione e alla programmazione del File.bin sulla flash del uC.

Menù Tools



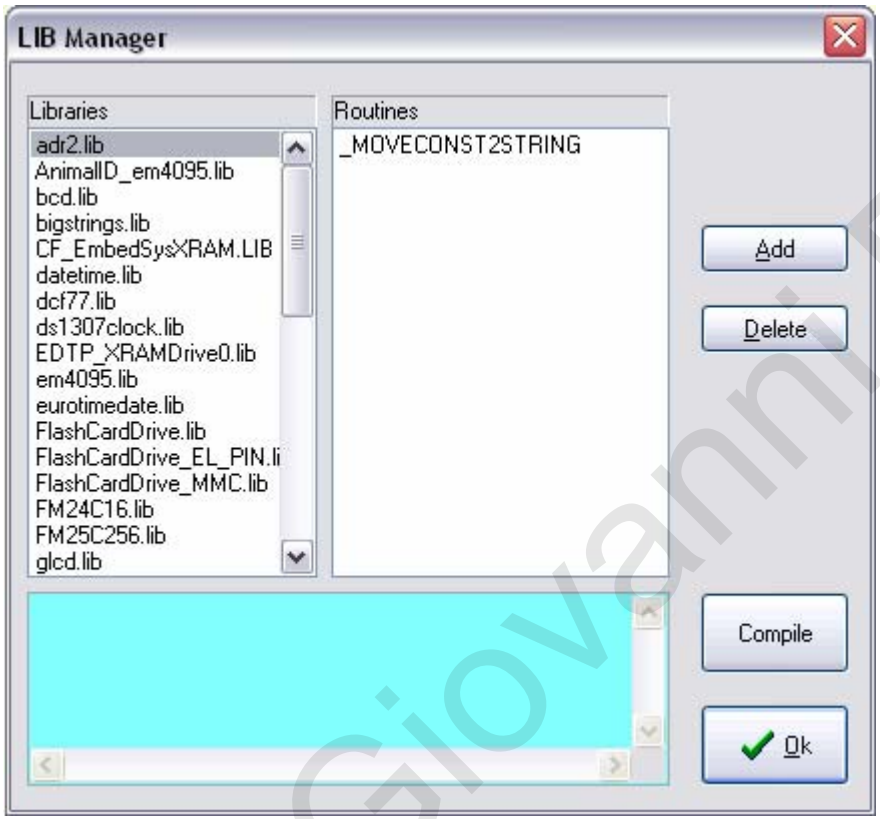
- Da qui è possibile accedere al terminale RS232, al LCD designer, e ad altri importanti tools che il Bascom mette a disposizione.

Tools -> LCD designer



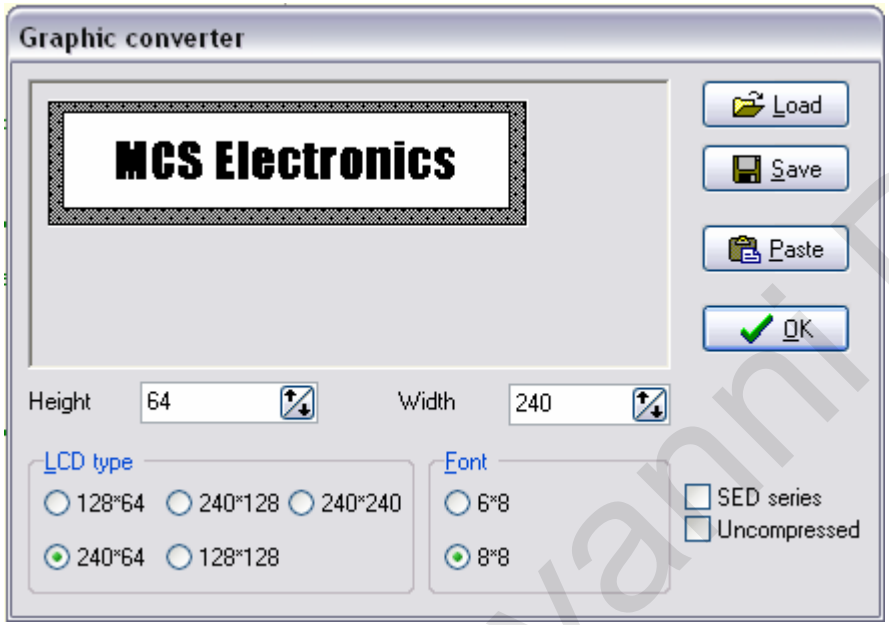
- Questo tools permette di creare caratteri da usare con LCD es. 16x2

Tools -> LIB manager



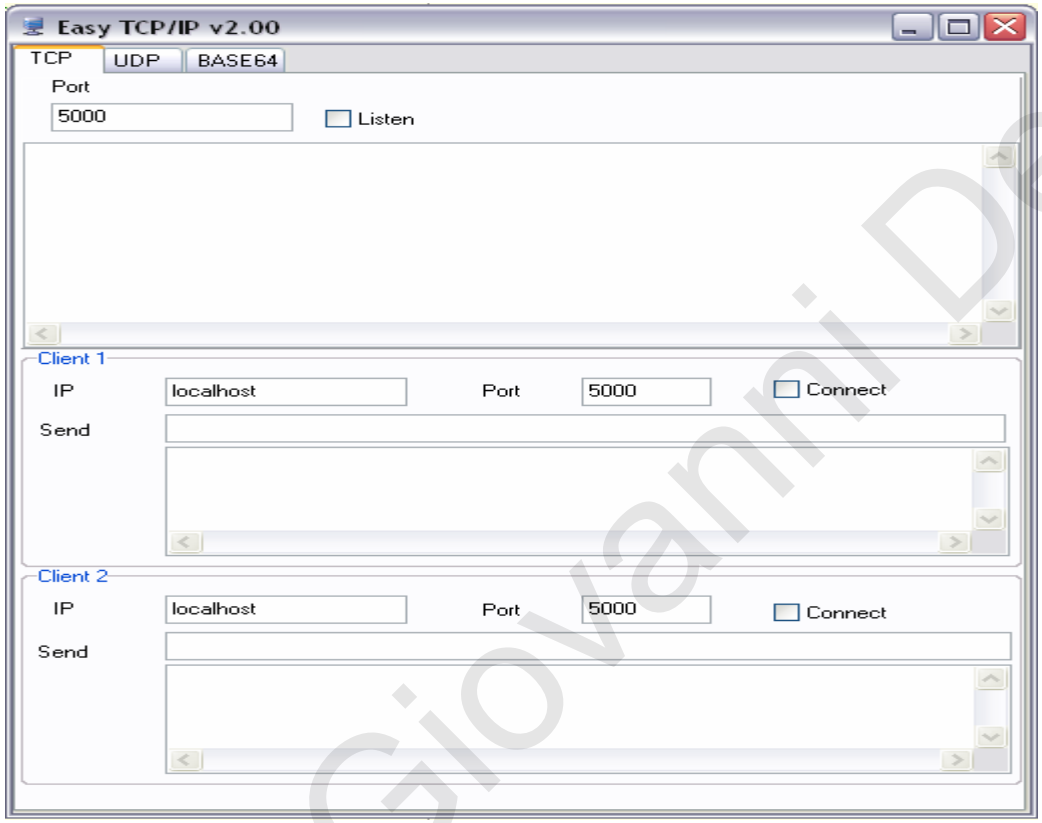
- Questo tool permette di compilare librerie scritte in assembly e criptarle come obj per distribuirle insieme al progetto sorgente.

Tools -> Graphic converter



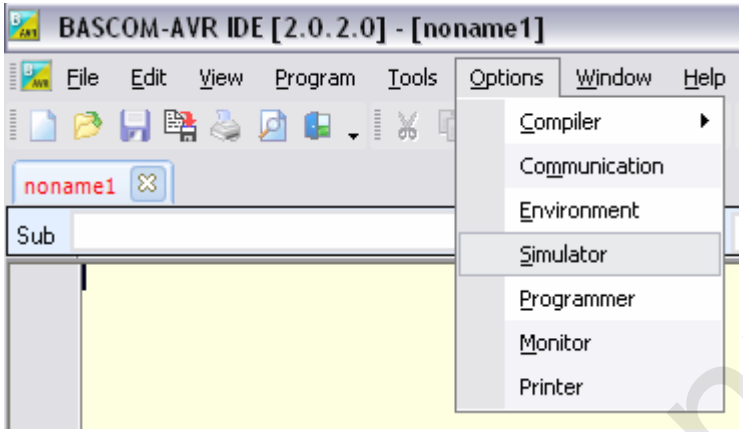
- Se si usa un LCD grafico sarà necessario convertire immagini bitmap nel formato BGF con questo tool.
- E' possibile scegliere tra vari formati, e se creare un file compresso o no.

Tools -> Easy TCP/IP



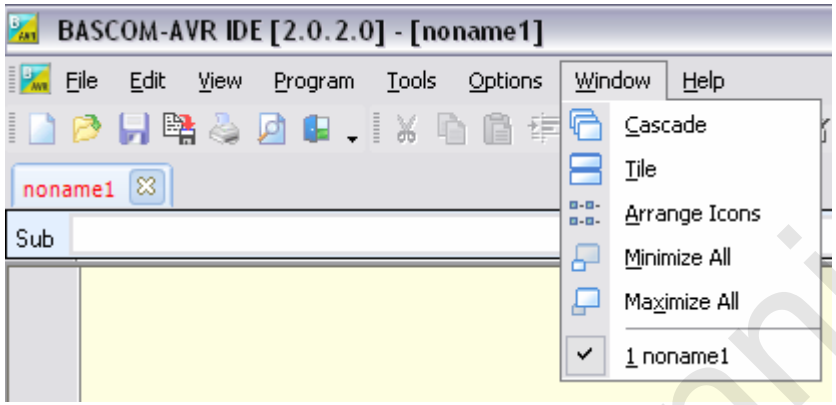
- Con questa utility è possibile comunicare con applicazioni che usano interfacce di comunicazione ethernet.
- E' possibile stabilire connessioni TCP o UDP.

Menù Option



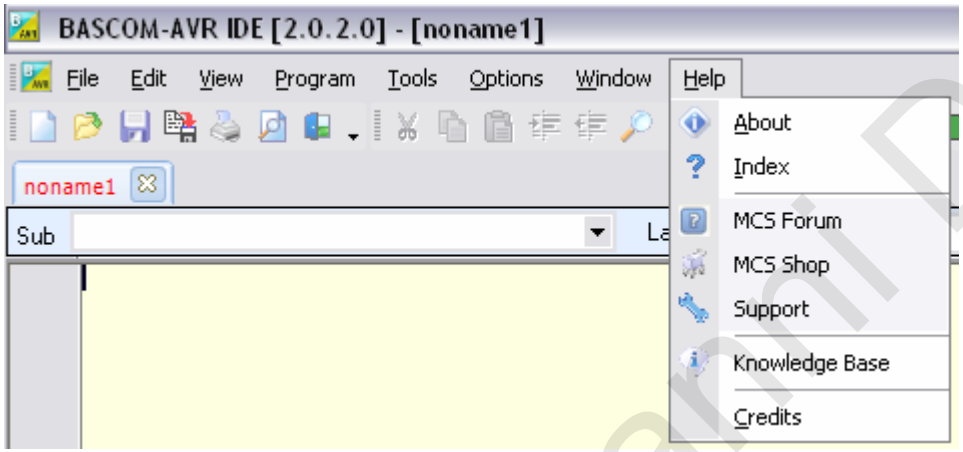
- Menù per la configurazione del chip, delle comunicazioni e del programmatore.

Menù Window



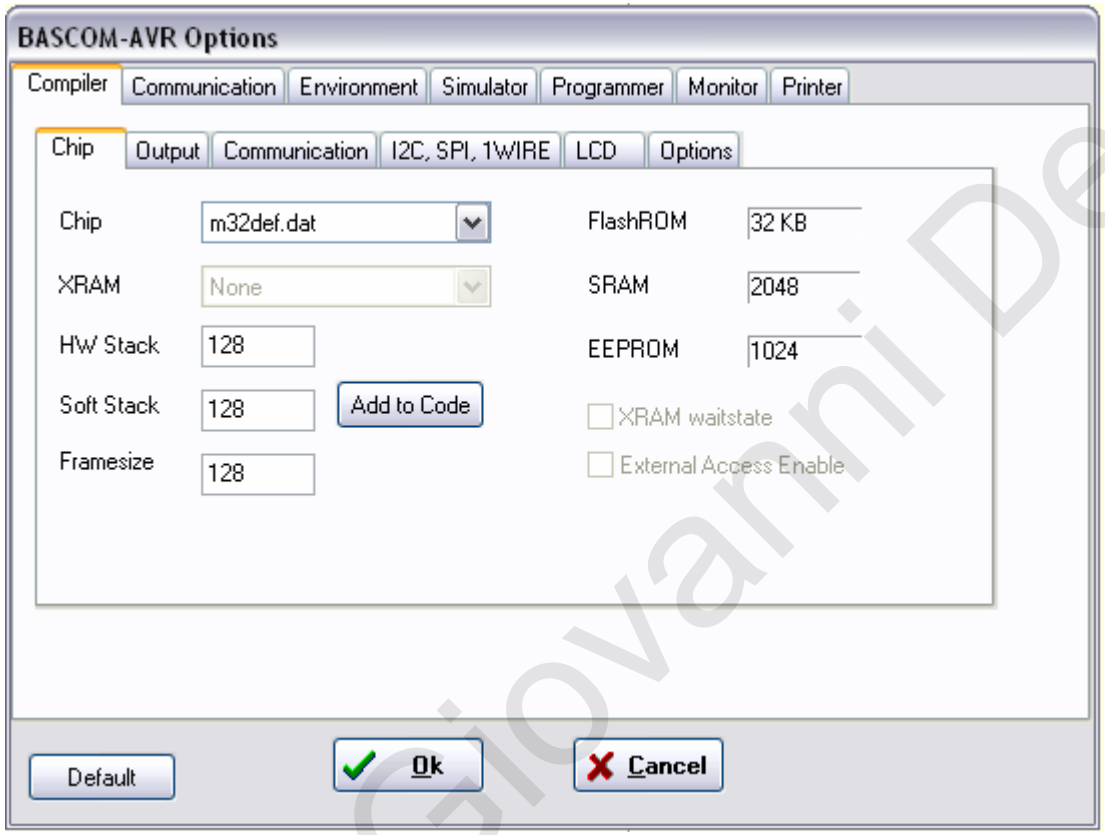
- Menù per la modalità della visualizzazione delle finestre dell'applicazione.

Menù Help



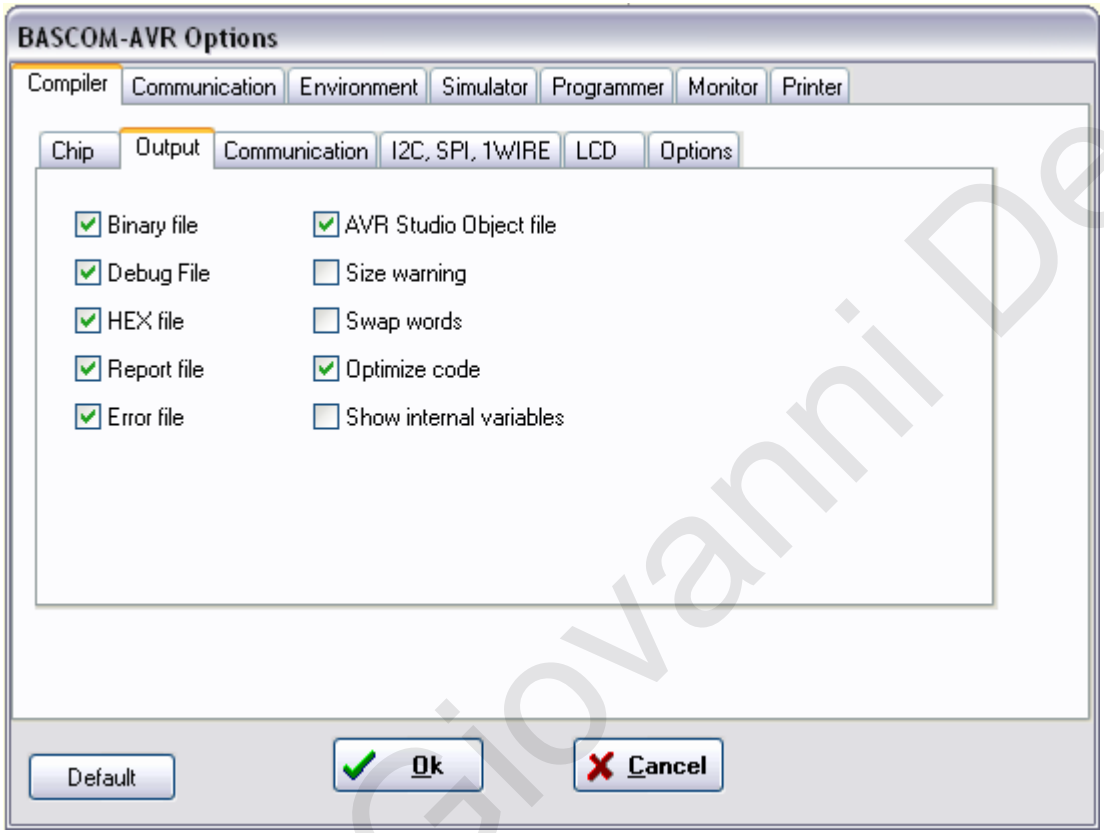
- Menù attraverso il quale si accede ai file di Help, ai forum e al support on-line

Option -> Compiler -> Chip



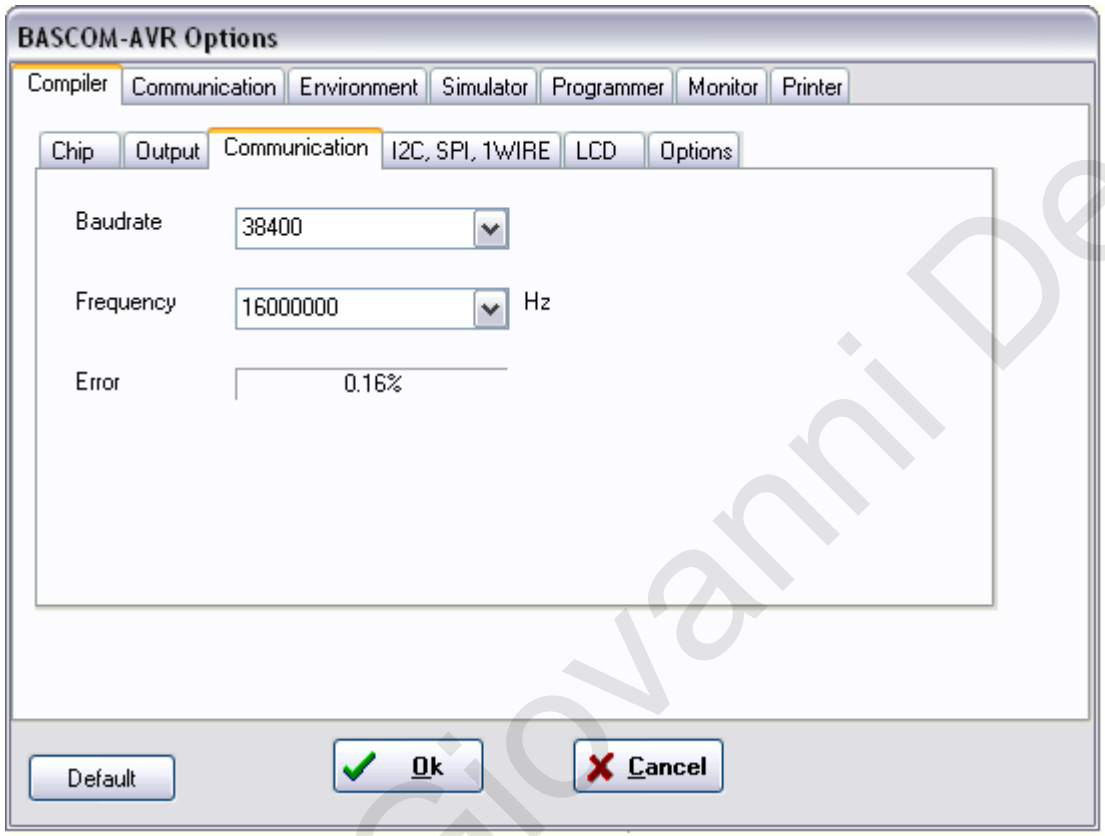
- Da questa finestra è possibile selezionare il chip, abilitare o disabilitare l'accesso alla memoria esterna, settare l'ammontare di memoria da riservare allo stack.

Option -> Compiler -> Output



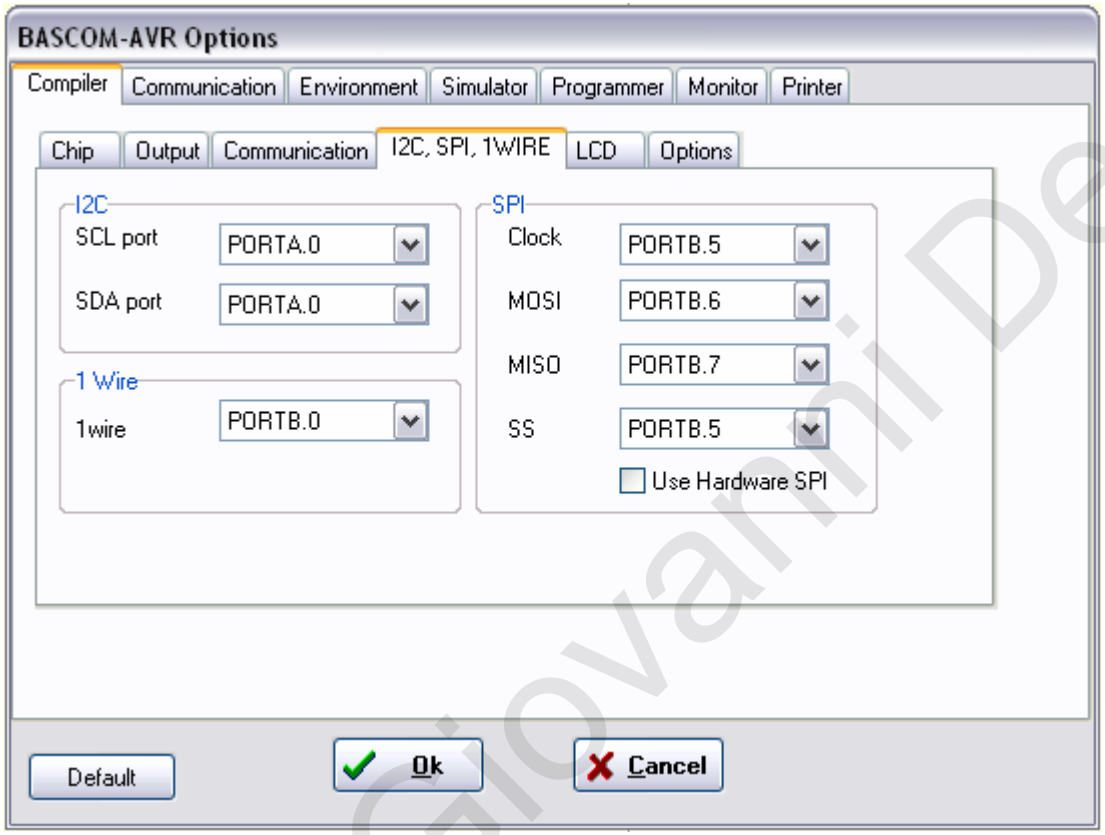
- Qui si decide quale tipo di file deve essere generato durante la compilazione del sorgente, se si vuole abilitare l'opzione di 'optimize code'

Option -> Compiler -> Comm



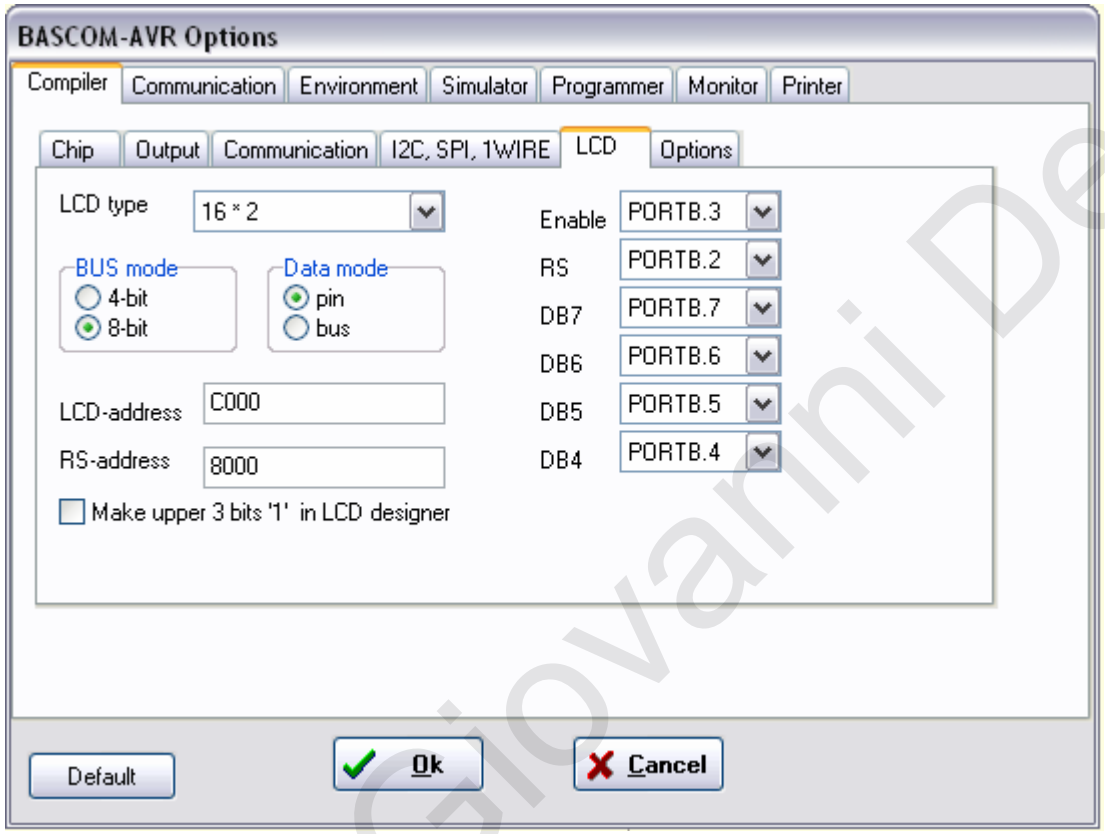
- Da questa finestra è possibile settare la frequenza di clock e il valore di baud rate relativo alla porta Com RS232.
- Un text box ci mostra se nei calcoli del baud rate verrà introdotto un errore espresso in %.

Option -> Compiler -> I2C etc



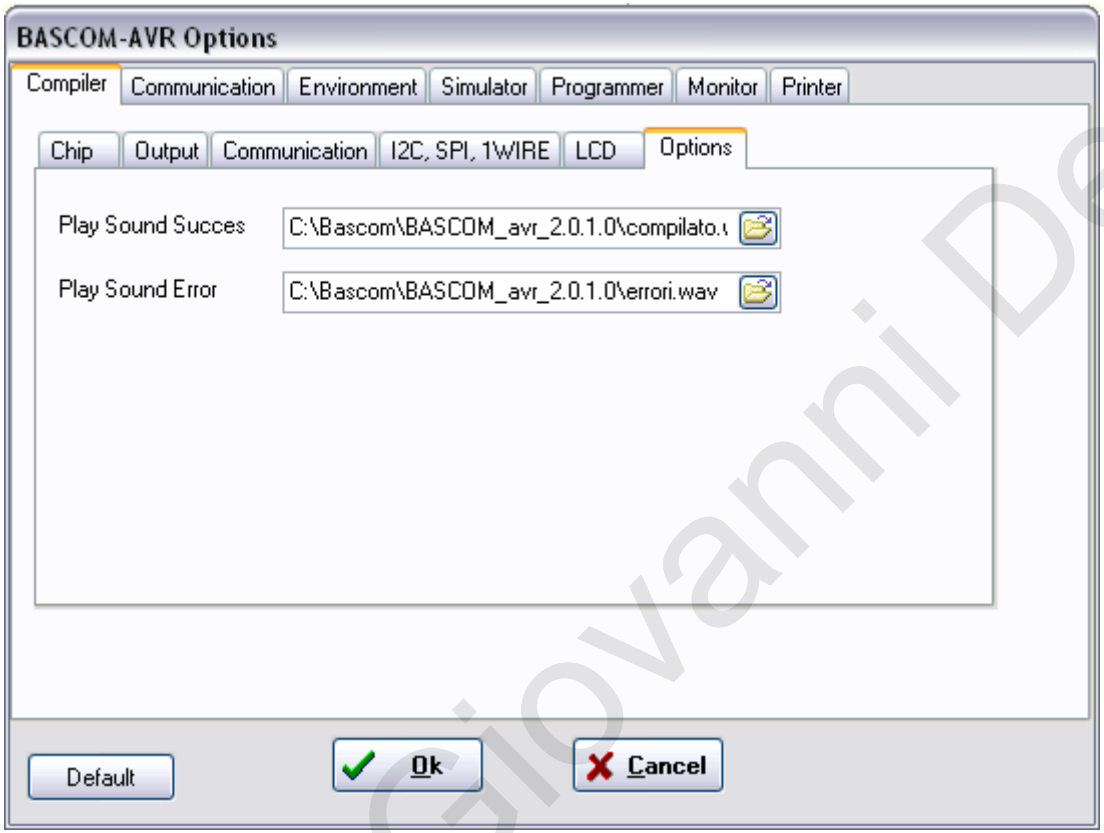
- Possiamo scegliere e configurare i pin da assegnare alle interfacce integrate : I2C, 1Wire 2 SPI.

Option -> Compiler -> LCD



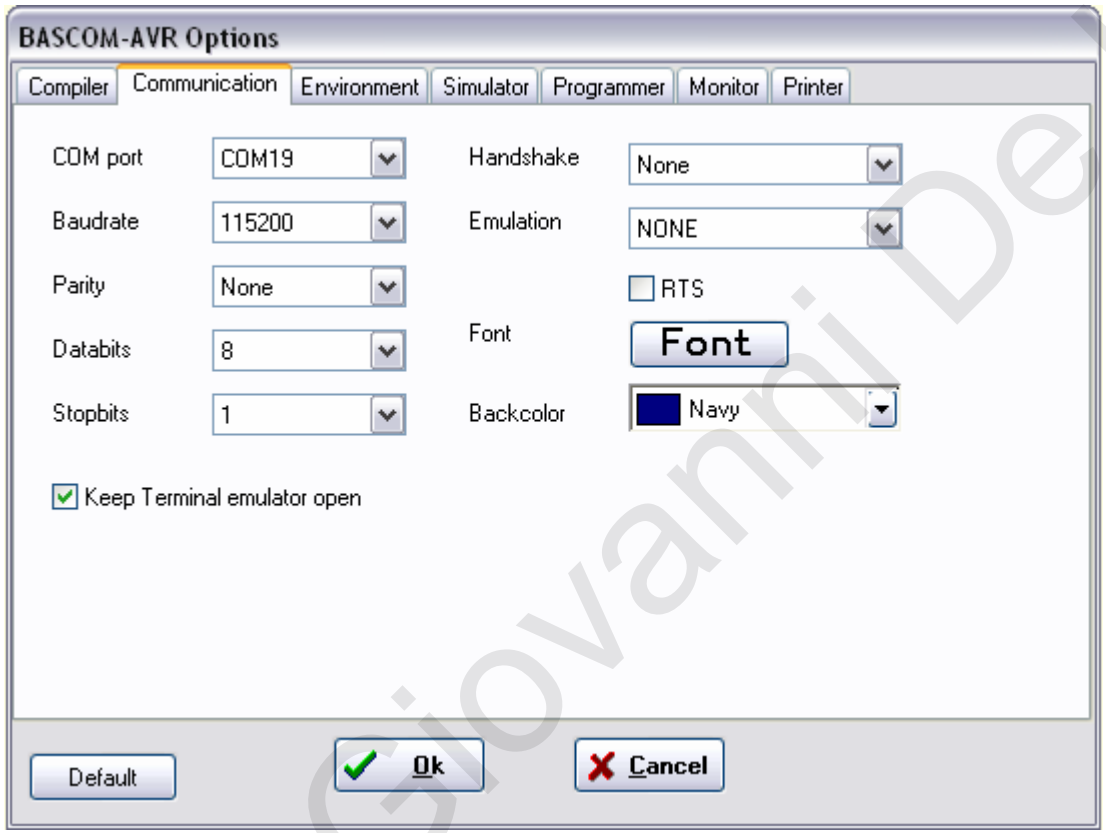
- Qui possiamo configurare il tipo di LCD da collegare al nostro uC.
- Inoltre possiamo scegliere di mappare l'LCD in memoria oppure utilizzare i singoli pin del display.

Option -> Compiler -> Option



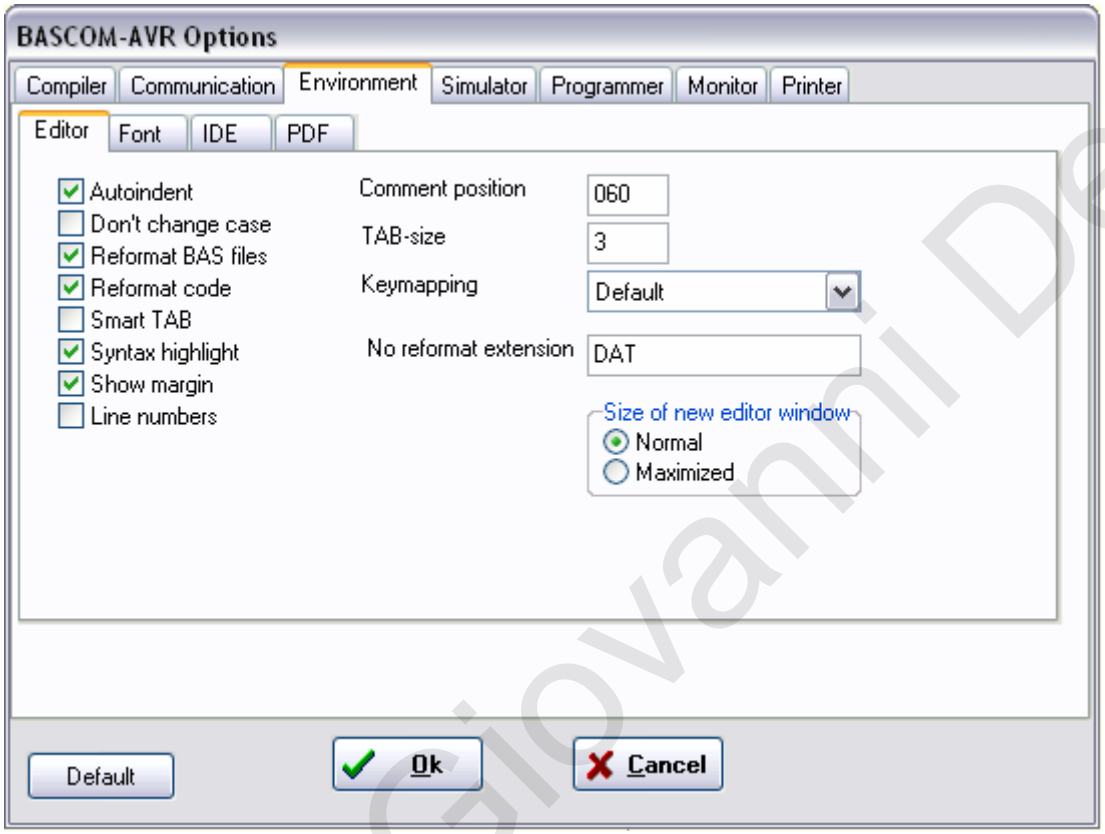
- Possiamo assegnare dei suoni ad eventi particolari.
- Per esempio alla fine della compilazione il sistema dice a voce “Programma compilato con successo”.
- Se il compilatore rileva errori il sistema dice “Errori trovati”.

Option -> Comm -> Port



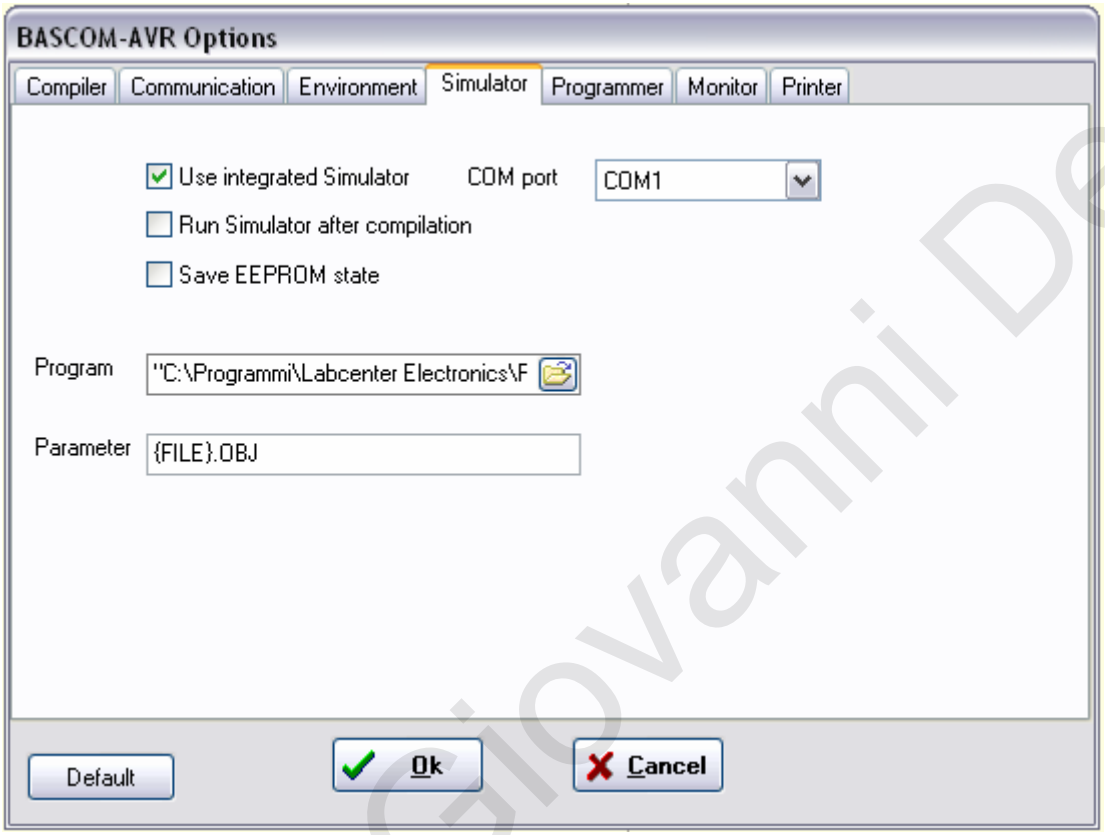
- Qui configuriamo i parametri del terminale: numero della Com, Baudrate, Parity, etc.

Option -> Enviroment



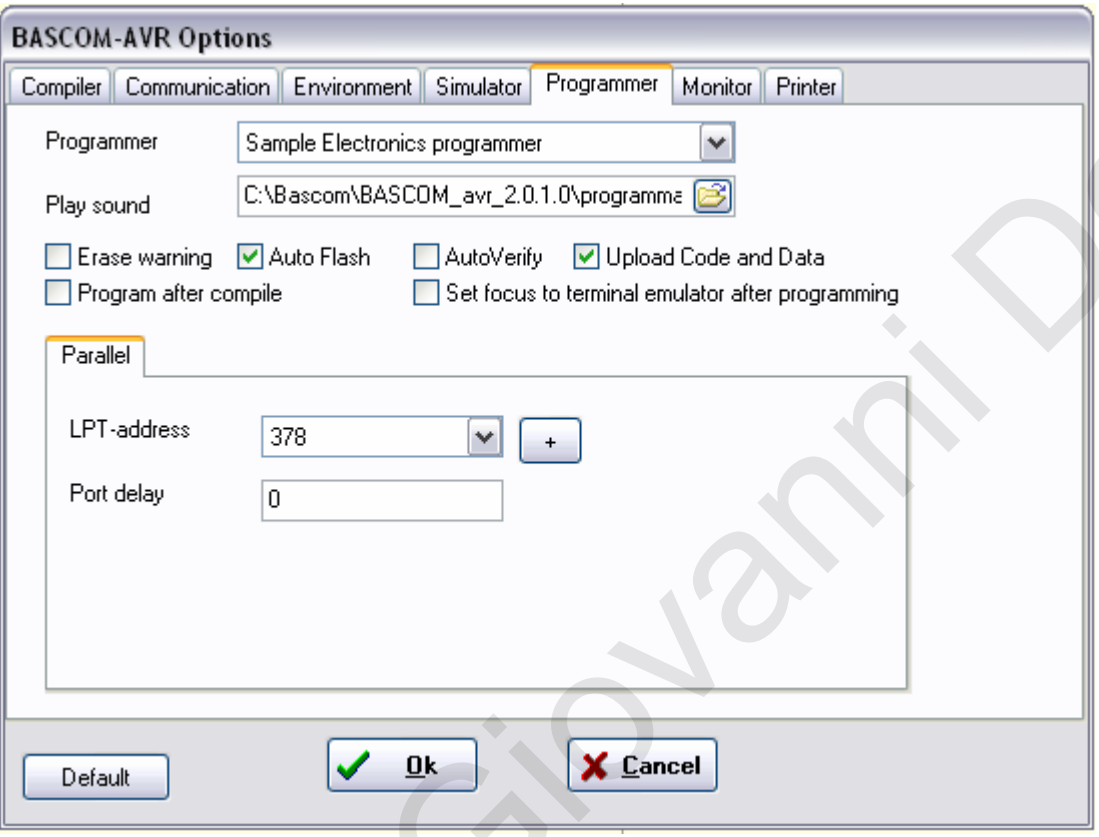
- Si possono definire alcuni parametri relativi alla modalità di visualizzazione dell' editor.

Option -> Simulator



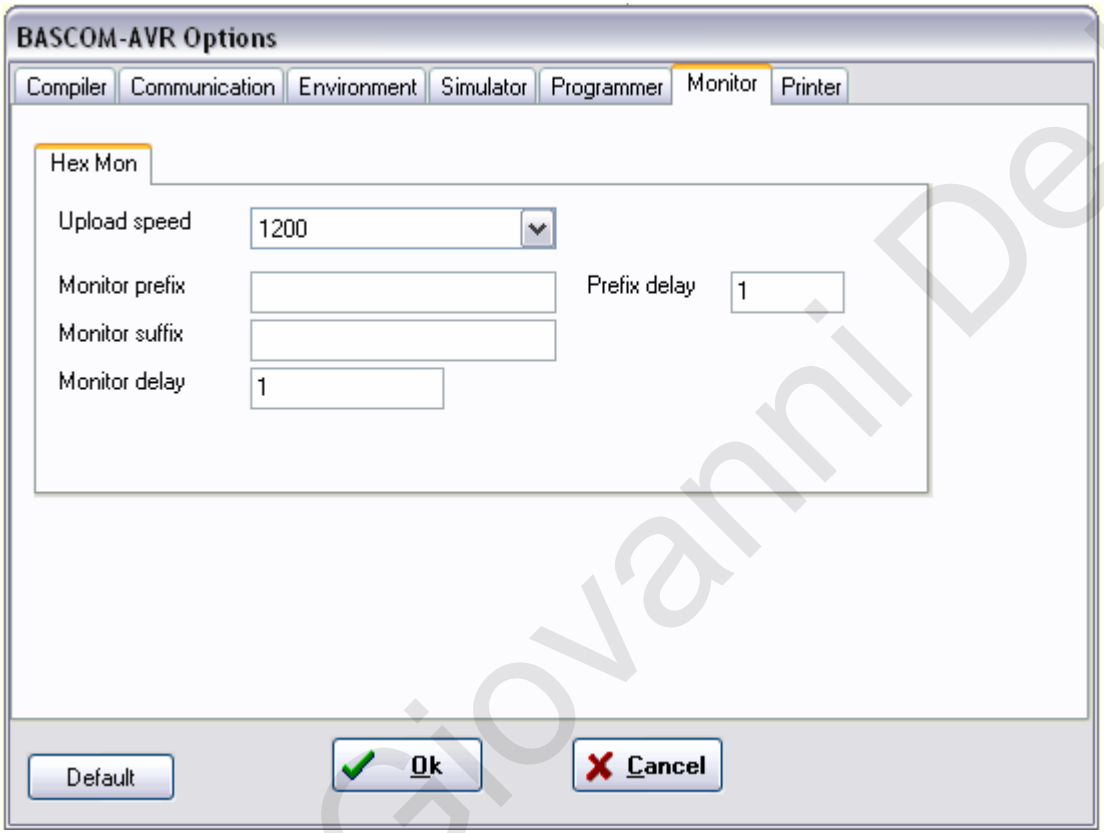
- Uso del simulatore: Interno o esterno.

Option -> Programmer



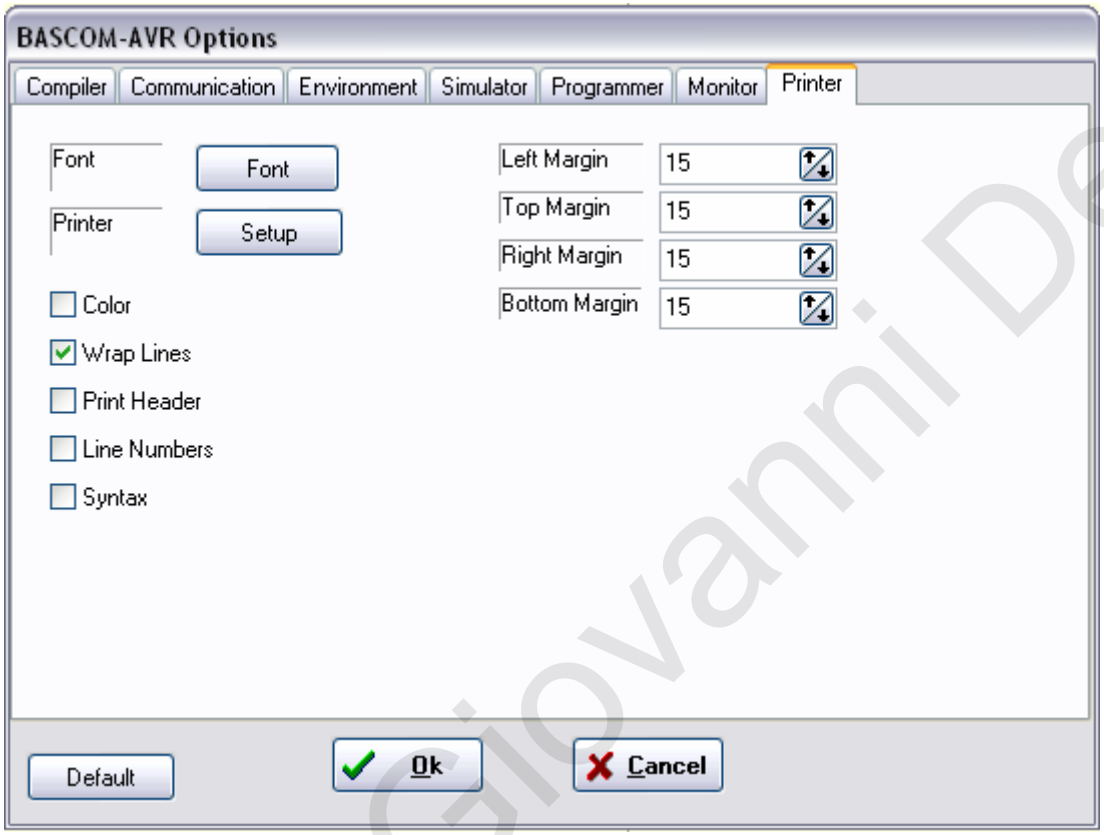
- Da qui è possibile scegliere il tipo di programmatore; MCS permette di utilizzare moltissimi tipi di programmatori.
- Alcuni possono essere auto-costruiti e collegati facilmente alla porta parallela di qualsiasi PC.
- Altri possono essere collegati alle porte seriali, altri alle porte USB.

Option -> Monitor



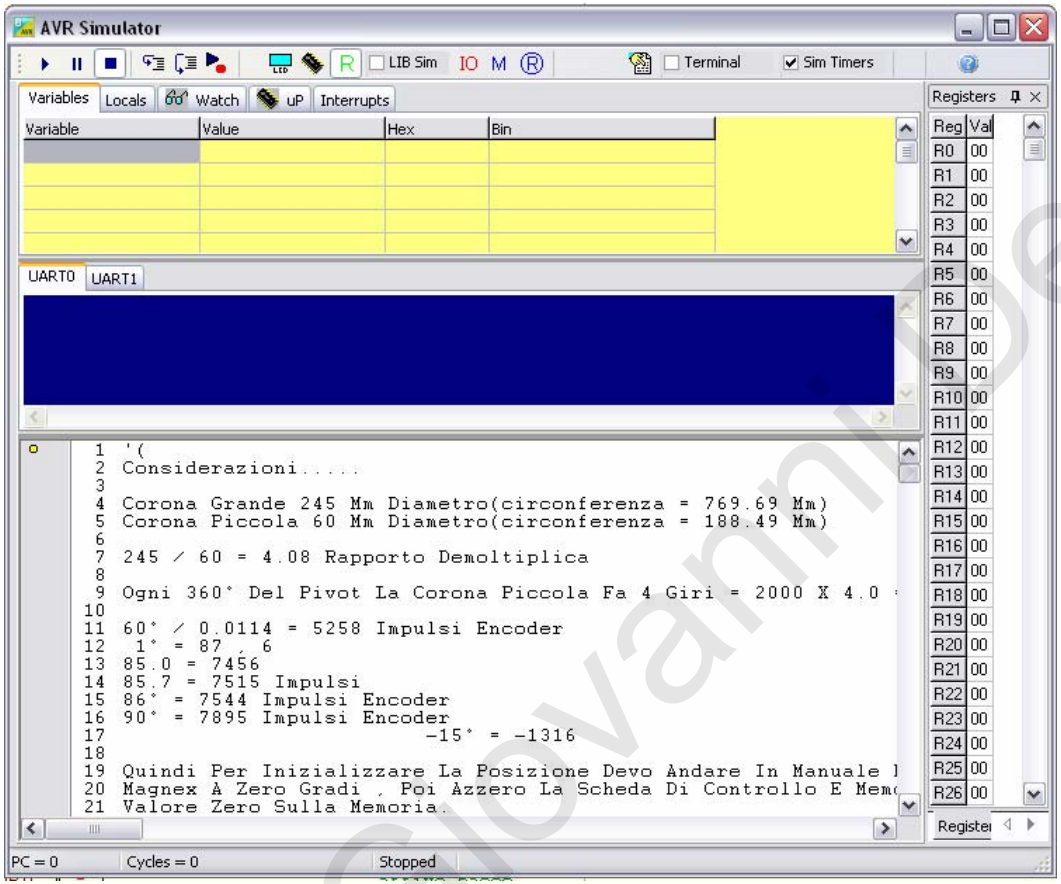
- E' possibile configurare il programma per il monitoraggio del download da BootLoader.

Option -> Printer



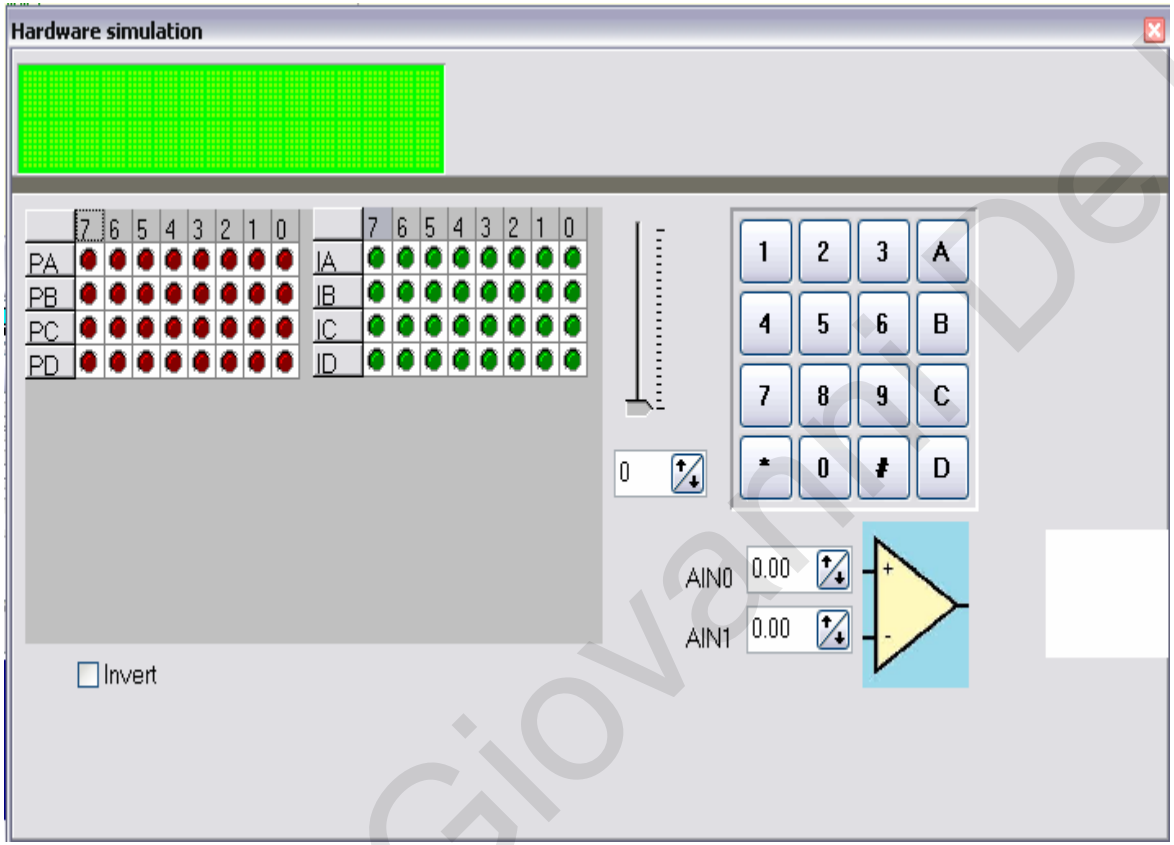
- Configurazione della stampante, set dei colori e modalità di impaginazione

AVR Simulator



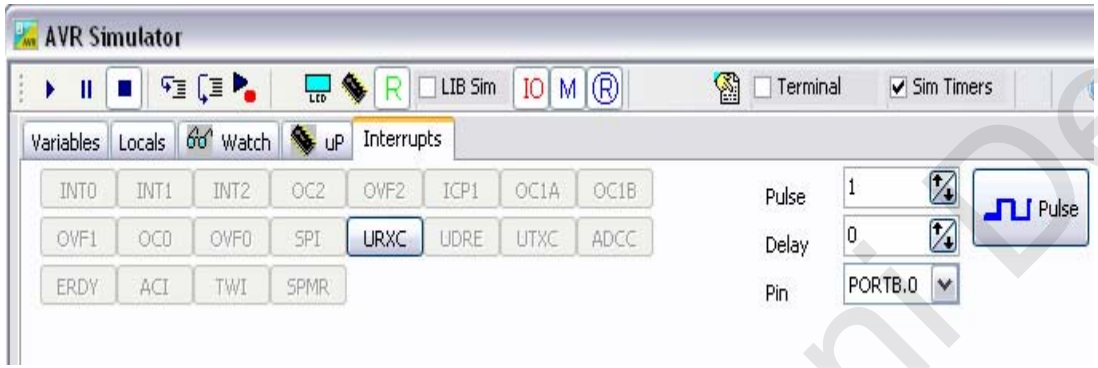
- Una delle finestre più importanti dell'IDE di Bascom è il Simulatore).
- Si possono gestire i segnali di stimulus, gli interrupts, e le periferiche analogiche presenti sul chip.

I/O Hardware simulation



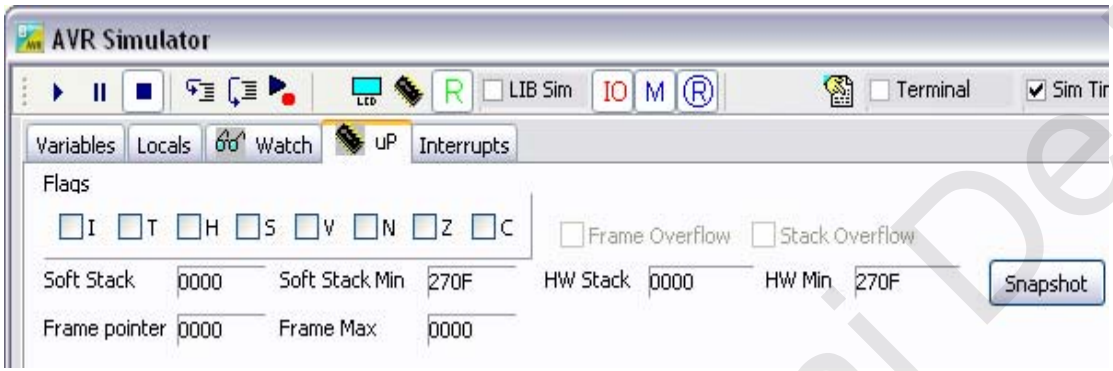
- Da questa finestra è possibile simulare ed applicare segnali digitali ai singoli pin, agli ingressi analogici, usare una keyboard matrix 4x4 e visualizzare i dati su un LCD.

Interrupts stimulus



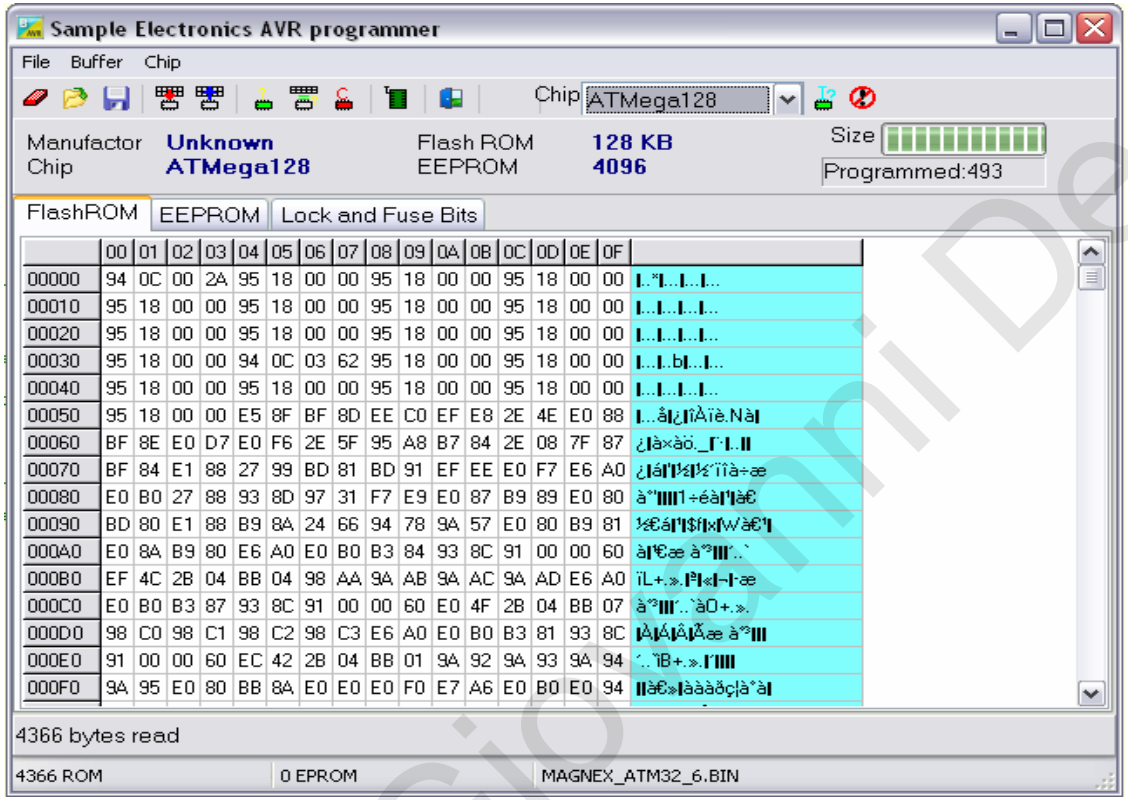
- Cliccando sui pulsanti si possono generare impulsi di stimolo e interrupts vari:
- Int. non mascherabili, provenienti dalla fine conversione dell'ADC o dalla seriale e altro.

uC state



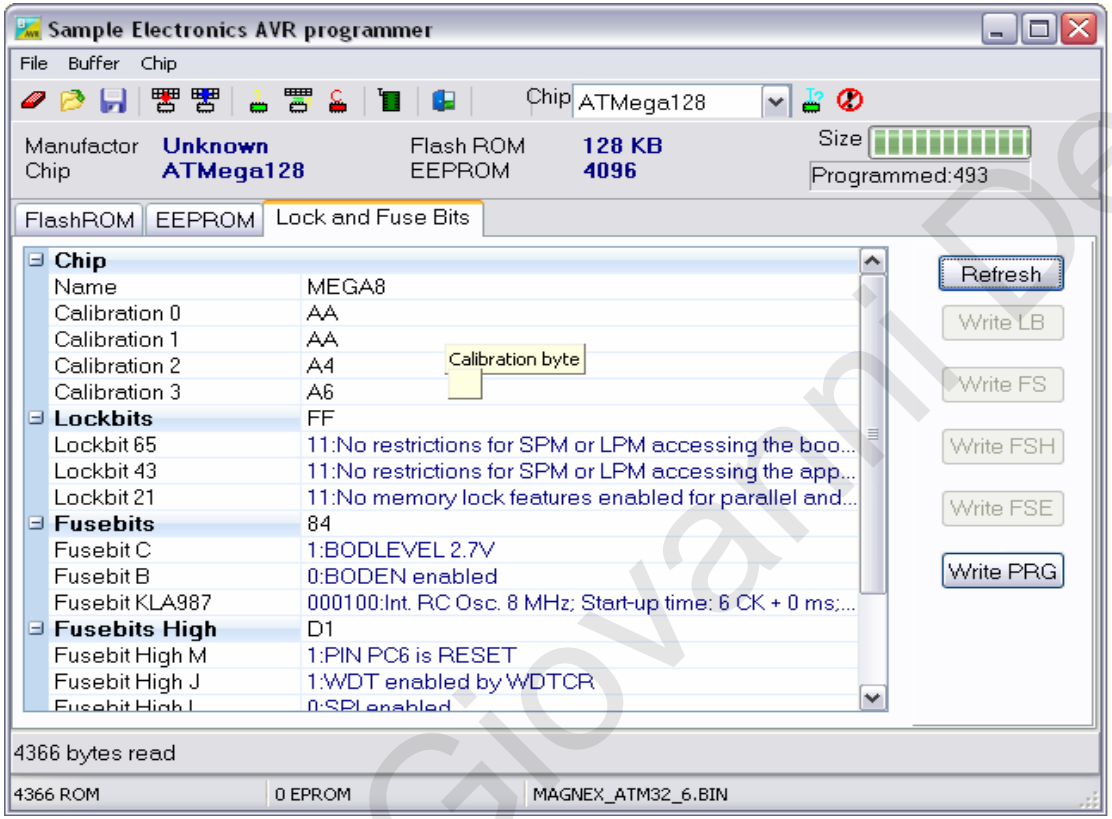
- Vengono mostrati in questa finestra: lo stato dei flags, degli stacks.

Sample programmer



- Questa è l'interfaccia di uno dei programmatori disponibili; Sample Programmer)
- Lo schema elettrico è disponibile sull'Help.
- Abbiamo la possibilità di programmare la flash, la eeprom o i fuse bits del micro.

Lock and fuse bits



- E' possibile configurare i fuse bits a seconda delle esigenze.
- E' possibile scegliere il generatore di clock, abilitare o disabilitare la protezione alla lettura, abilitare il watchdog hardware e altro.

Introduzione al Bascom-AVR

- Scelta del microcontrollore
- Utilizzo dei file di definizione (Def.dat)
- Configurazione della porta Com1
- Configurazione del display LCD
- Configurazione delle porte di I/O
- Configurazione dell'ADC interno
- Dimensionamento delle variabili
- Tipi di variabili
- Struttura del MAIN
- Esempio: Blink Led
- Uso di Locate, LCD, Cls, Cursor
- Uso della UART, Print e Input
- Interrupt seriale URXC

Scelta del microcontrollore

- Prima di procedere con la stesura di qualsiasi programma è necessario stabilire il tipo di uC da utilizzare, o almeno fissarne il package (DIP, TQFP)
- Bisogna fare i conti con il numero dei pin disponibili e con le periferiche hardware che ci necessitano (come stabilito sullo schema elettrico).
- N.B. Non tutti i uC della stessa famiglia sono pin to pin compatibili.
- Una volta scritto un programma sarà comunque possibile, facendo piccole modifiche, ricompilarlo per altri chip della stessa famiglia.

Primo passo : file Def.dat

$regfile = "m128def.dat"

$crystal = 14745600

$baud = 115200

$hwstack = 128

$swstack = 128

$framesize = 128

$HWSTACK, $FRAMESIZE

- The Hardware stack is room in RAM that is needed by your program. When you use GOSUB label, the microprocessor pushes the return address on the hardware stack and will use 2 bytes for that. When you use RETURN, the HW stack is popped back and the program can continue at the proper address. When you nest GOSUB, CALL or functions, you will use more stack space. Most statements use HW stack because a machine language routine is called.

- You need a minimum frame size of 24 bytes. This space is used by a number of routines. For example string<>numeric conversion routines. If you use Print numVar, then the numeric variable "numvar" is converted into a string representation of the binary number. The framespace buffer is used for that. While the framespace server as dynamic memory, a fixed address is used. For this reason the buffer has a fixed size of 24 bytes.

Configurazione della Com1

CONFIG COM1 = baud , synchrone=0|1,parity=none|disabled|even|odd,stopbits=1|2,databits=4|6|7|8|9,clockpol=0|1

baud	Baud rate to use. Use 'dummy' to leave the baud rate at the $baud value.
synchrone	0 for asynchrone operation (default) and 1 for synchrone operation.
Parity	None, disabled, even or odd
Stopbits	The number of stop bits : 1 or 2
Databits	The number of data bits : 4,5,7,8 or 9.
Clockpol	Clock polarity. 0 or 1.

Config Com1 = 115200 , Synchrone = 0 , Parity = None , Stopbits = 1 , Databits = 8 , Clockpol = 0

Configurazione dell' LCD

CONFIG LCD = LCDtype , CHIPSET=KS077 | Dogm163v5 | DOG163V3 | DOG162V5 | DOG162V3 [,CONTRAST=value]

LCDtype	The type of LCD display used. This can be : 40 * 4,16 * 1, 16 * 2, 16 * 4, 16 * 4, 20 * 2 or 20 * 4 or 16 * 1a or 20*4A. Default 16 * 2 is assumed.
Chipset KS077	Most text based LCD displays use the same chip from Hitachi. But some use the KS077 which is highly compatible but needs an additional function register to be set. This parameter will cause that this register is set when you initialize the display.
CHIPSET DOGM	The DOGM chip set uses a special function register that need to be set. The 16 x 2 LCD displays need DOG162V3 for 3V operation or DOG162V5 for 5V operation. The 16 x 3 LCD displays need DOG163V3 for 3V operation or Dogm163v5 for 5V operation
CONTRAST	The optional contrast parameter is only supported for the EADOG displays. By default a value from the manufacture is used. But you might want to override this value with a custom setting.

Config Lcd = 16 * 2

Configurazioni delle porte I/O

CONFIG PORTx = state
CONFIG PINx.y = state

state	A numeric constant that can be INPUT or OUTPUT. INPUT will set the data direction register to input for port X. OUTPUT will set the data direction to output for port X. You can also use a number for state. **&B**00001111, will set the upper nibble to input and the lower nibble to output. You can also set a single port pin with the CONFIG PIN = state, statement. Again, you can use INPUT, OUTPUT or a number. In this case the number can be only zero or one.

Config Portd = **Input**

Config Porta = **Output**

Configurazioni delle porte I/O

Altra modalità usando i registri di configurazione:

```
DDRA=&B_1111_1111  'configura tutti i pin della porta A come output
DDRA=&HFF          'configura tutti i pin della porta A come output
DDRB=&B_0000_1111  'configura i bit 3..0 come output, 7..4 come input
DDRB.3=1           'configura il bit 3 come output
```

Uso delle resistenza di pull-up:

```
DDRA.0=0           'configura il bit 0 della PORT(A) come input
PORTA.1=1          'abilitiamo la resistenza di pull-up forzando a 1 il pin
```

Uso di 'ALIAS':

Pulsante ALIAS PINA.0 'al pin d'ingresso PINA.0 diamo il nome 'Pulsante'

Configurazione ADC

```
Config Adc = Single , Prescaler = Auto , Reference = Avcc
Start Adc
Dim W As Word , Channel As Byte

Channel = 0
Do

  W = Getadc(channel)
  Print "Channel " ; Channel ; " value " ; W
  Incr Channel
  If Channel > 7 Then Channel = 0

Loop
End
```

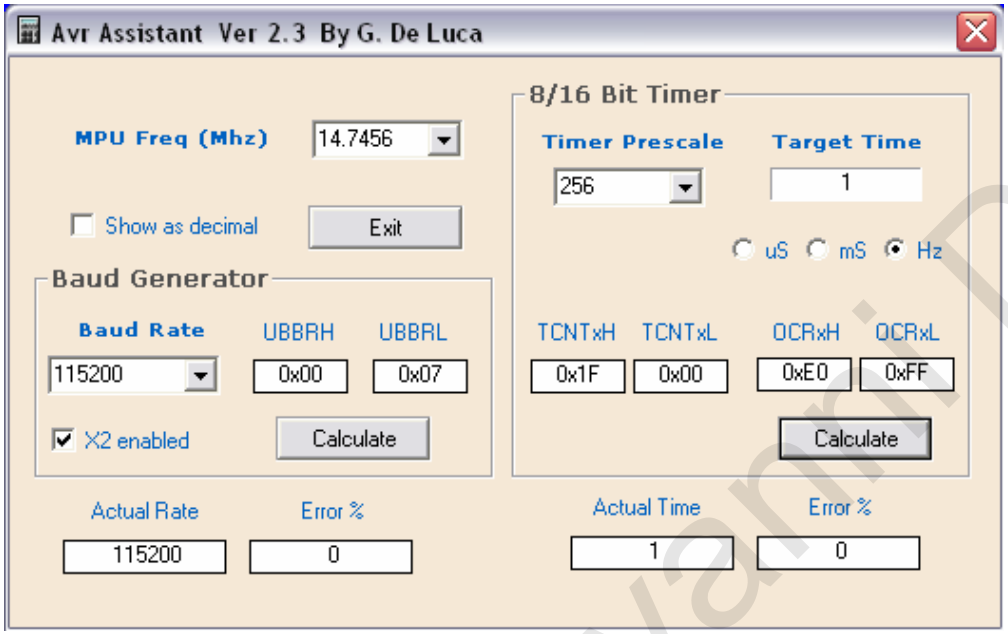
Lettura ADC in free mode

```
Config Adc = Free , Prescaler = Auto , Reference =Internal
On Adc Adc_isr Nosave
Enable Adc
Enable Interrupts
Dim W As Word , Channel As Byte

Channel = 0
Do
  Channel = 0
  Start Adc
  Idle
  Stop Adc
  Print "Channel " ; Channel ; " value " ; W
Loop
End
```

```
Adc_isr:
      push r26
      push r27
      push r24
      in r24,sreg
      push r24
      push r25
      W = Getadc(channel)

      pop r25
      pop r24
      !out sreg,r24
      pop r24
      pop r27
      pop r26
Return
```

Configurazione del Timer 0

Uso di AVR Assistant

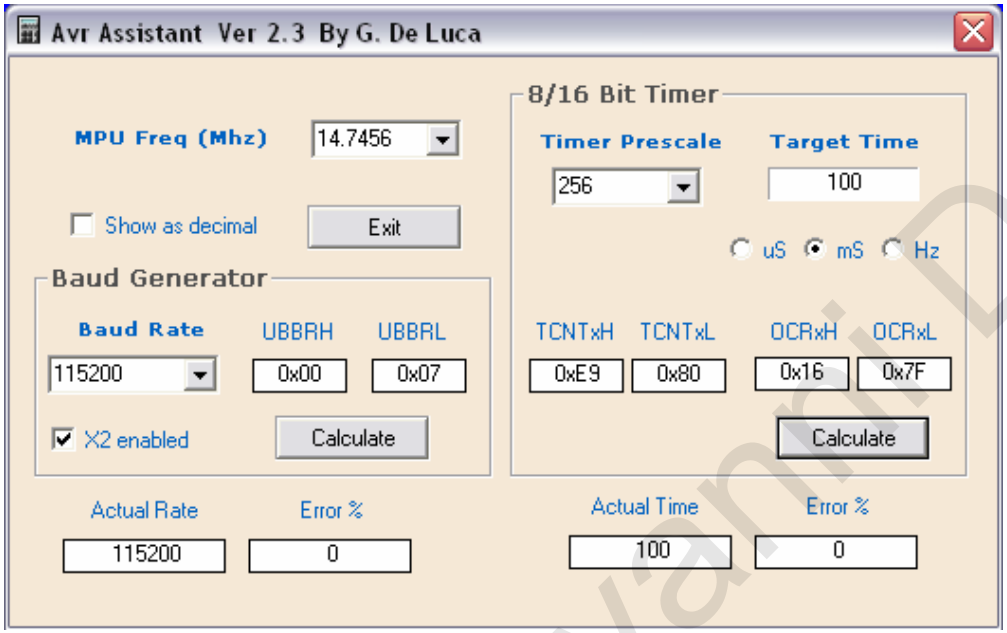


In questo esempio con un cristallo di 14.7456 Mhz, impostando opportunamente i registri è possibile ottenere un Baudrate di 115200 e una frequenza di intervento del Timer pari a 1Hz

Per calcolare i valori da assegnare ai registri dei timer per impostare il BaudRate o il tempo di intervento del Timer 0,1,2, è possibile utilizzare questa utility scaricabile dal sito : www.delucagiovanni.com

Configurazione del Timer 0

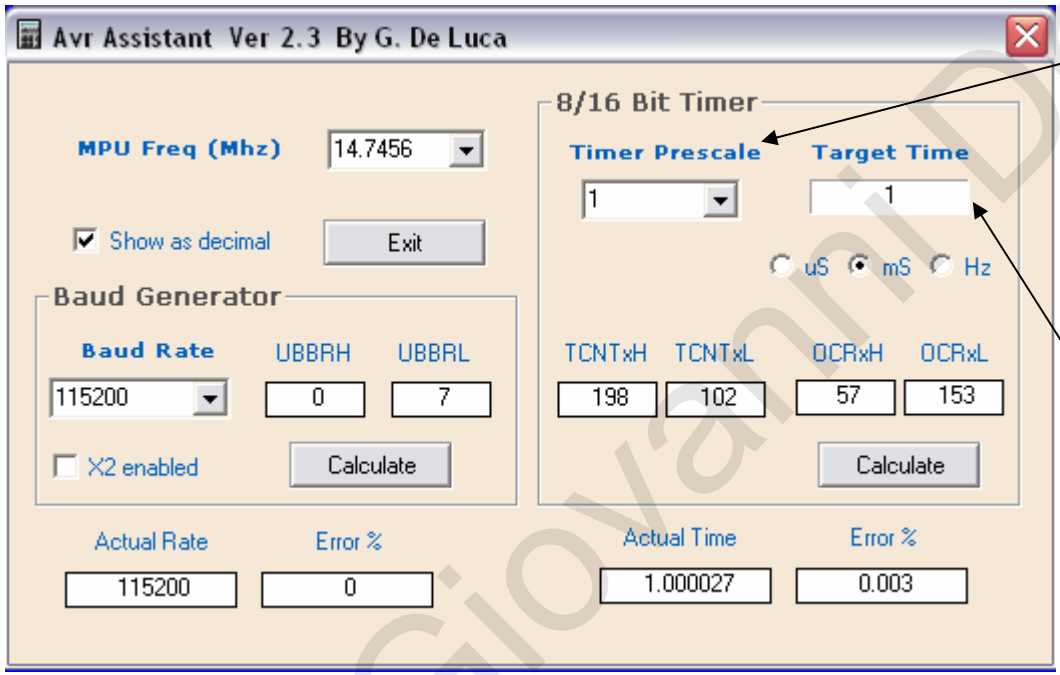
Uso di AVR Assistant



Con questa configurazione otteniamo un periodo di intervento di 100 mSec. Abilitando l'interrupt corrispondente e indicando l'indirizzo di gestione, il programma ogni 100 mSec salterà all' Interrupt Handler ed eseguirà le istruzioni contenute nella subroutine.

Per calcolare i valori da assegnare ai registri dei timer per impostare il BaudRate o il tempo di intervento del Timer 0,1,2, è possibile utilizzare questa utility scaricabile dal sito : www.delucagiovanni.com

Configurazione del Timer 1



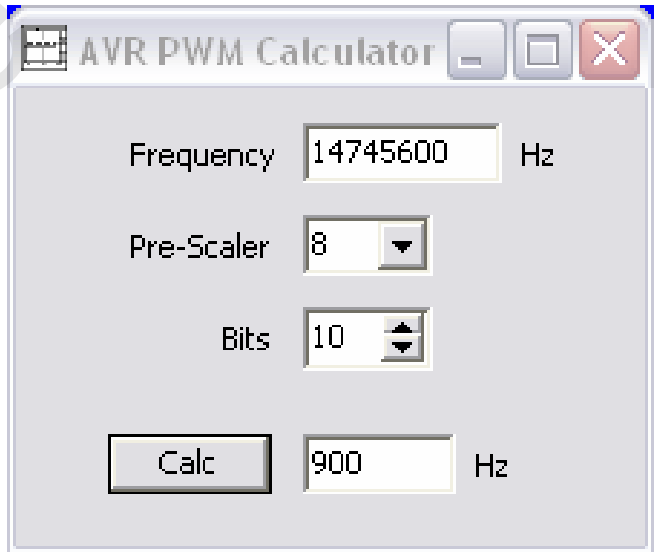

```
$regfile = "m32def.dat"
$crystal = 14745600
Config Timer1 = Timer , Prescale = 1
Ddra.0 = 1
Enable Interrupts
Enable Timer1
On Ovf1 Timer_1
Start Timer1

Do
    nop
Loop
End
'-- entra ogni 1 msec --
Timer_1:
  Tcnt1h = 198
  Tcnt1l = 102
  Ocr1ah = 57
  Ocr1al = 153
  Toggle Porta.0
Return
```

Configurazione del PWM

PWM risoluzione = 8bit
14745600 / 256 = 57600
57600 / 8 (prescale) = 7200 se 8bit
abbiamo 2 uscite OC1A e OC1B
il valore della freq va diviso x 2
abbiamo così 3600 Hz per canale.

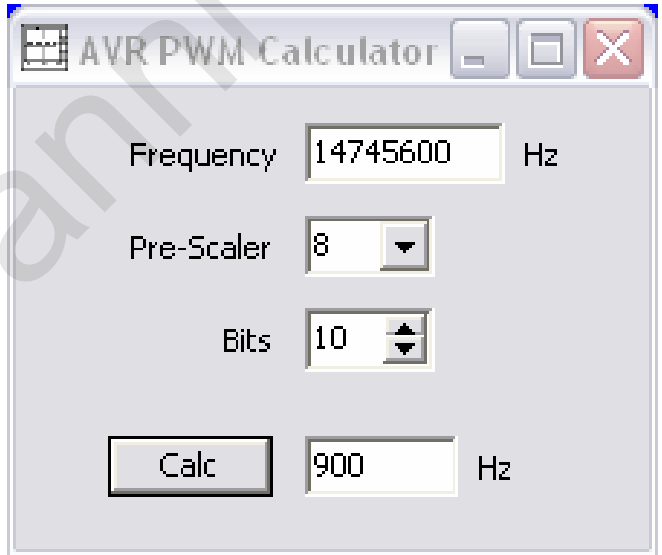
PWM risoluzione = 9bit
14745600 / 256 = 57600
57600 / 16 (prescale) = 3600 se 9bit
abbiamo 2 uscite OC1A e OC1B
il valore della freq va diviso x 2
abbiamo così 1800 Hz per canale.

PWM risoluzione = 10bit
14745600 / 256 = 57600
57600 / 32 (prescale) = 1800 se 10bit
abbiamo 2 uscite OC1A e OC1B
il valore della freq va diviso x 2
abbiamo così 900 Hz per canale.



Configurazione del PWM

```
$crystal = 14745600
Config Timer1 = Pwm , Pwm = 10 , Compare A Pwm = Clear Down , Prescale = 8
            ‘14745600 / 256 = 57600 / 32 (10bit) = 1800 Hz
            ‘abbiamo 2 uscite la freq viene divisa x 2 : abbiamo 900Hz per canale

Pwm1a = 100
Do
   nop
Loop
End
```



Dimensionamento Variabili

DIM var AS [XRAM/SRAM/ERAM]type [AT location/variable] [OVERLAY]

Var	Any valid variable name such as b1, i or longname. var can also be an array : ar(10) for example.
Type	Bit, Byte, Word, Integer, Long, Single, Double or String
XRAM	Specify XRAM to store variable into external memory
SRAM	Specify SRAM to store variable into internal memory (default)
ERAM	Specify ERAM to store the variable into EEPROM
OVERLAY	Specify that the variable is overlaid in memory.
location	The address of name of the variable when OVERLAY is used.

Tipi di variabili

Bit (1/8 byte). A bit can hold only the value 0 or 1. A group of 8 bits is called a byte.

Byte (1 byte). Bytes are stores as unsigned 8-bit binary numbers ranging in value from 0 to 255.

Integer (2 bytes). Integers are stored as signed sixteen-bit binary numbers ranging in value from -32,768 to +32,767.

Word (2 bytes). Words are stored as unsigned sixteen-bit binary numbers ranging in value from 0 to 65535.

Long (4 bytes). Longs are stored as signed 32-bit binary numbers ranging in value from -2147483648 to 2147483647.

Single. Singles are stored as signed 32 bit binary numbers. Ranging in value from 1.5 x 10^–45 to 3.4 x 10^38

Double. Doubles are stored as signed 64 bit binary numbers. Ranging in value from 5.0 x 10^–324 to 1.7 x 10^308

String (up to 254 bytes). Strings are stored as bytes and are terminated with a 0-byte. A string dimensioned with a length of 10 bytes will occupy 11 bytes.

Struttura del Main

```
$sim
$regfile = "m128def.dat"
$crystal = 14745600
$baud = 115200
$hwstack = 128
$swstack = 128
$framesize = 128

Main:
Do
   nop
Loop
End
```

Esempio 1:Blink Led

```
$sim                          ‘uso del simulatore
$regfile = "m128def.dat"
$crystal = 14745600
$baud = 115200
$hwstack = 128
$swstack = 128
$framesize = 128

Ddra.0 = 1                    ‘configurazione output
Led Alias Porta.0             ‘uso di alias
Main:                         ‘main programm
Do                            ‘ciclo do-loop
   Toggle Led                 ‘uso di toggle
   Waitms 1000                ‘aspetta 1000 mSec=1Sec
Loop                          ‘end loop
End                           ‘end programm
```

Uso di Locate & LCD

```
(configurazione uC)

Config Lcd = 16 * 2

Dim J As Byte

Cls
Cursor Off

Do
  Locate 1 , 1 : Lcd J
  Incr J
  Waitms 100
Loop
End
```

Possiamo fare alcune prove cambiando il dimensionamento della variabile J; word, integer, single, long, double,
e il tempo relativo a WAITMS.
E' possibile sostituire il valore 100 o altro valore con una costante o con una variabile:

```
CONST Tempo=100
Waitms Tempo
```

```
Dim Tempo1 as byte
Tempo1=100
Waitms Tempo1
```

Uso della UART -> Print

```
Dim J As Byte
Do
  Print J
  Incr J
  Waitms 100
Loop
End
```

Uso della UART -> Input

```
Dim J As Byte
Do
  Input "Number:" , J
  Print J
Loop
End
```

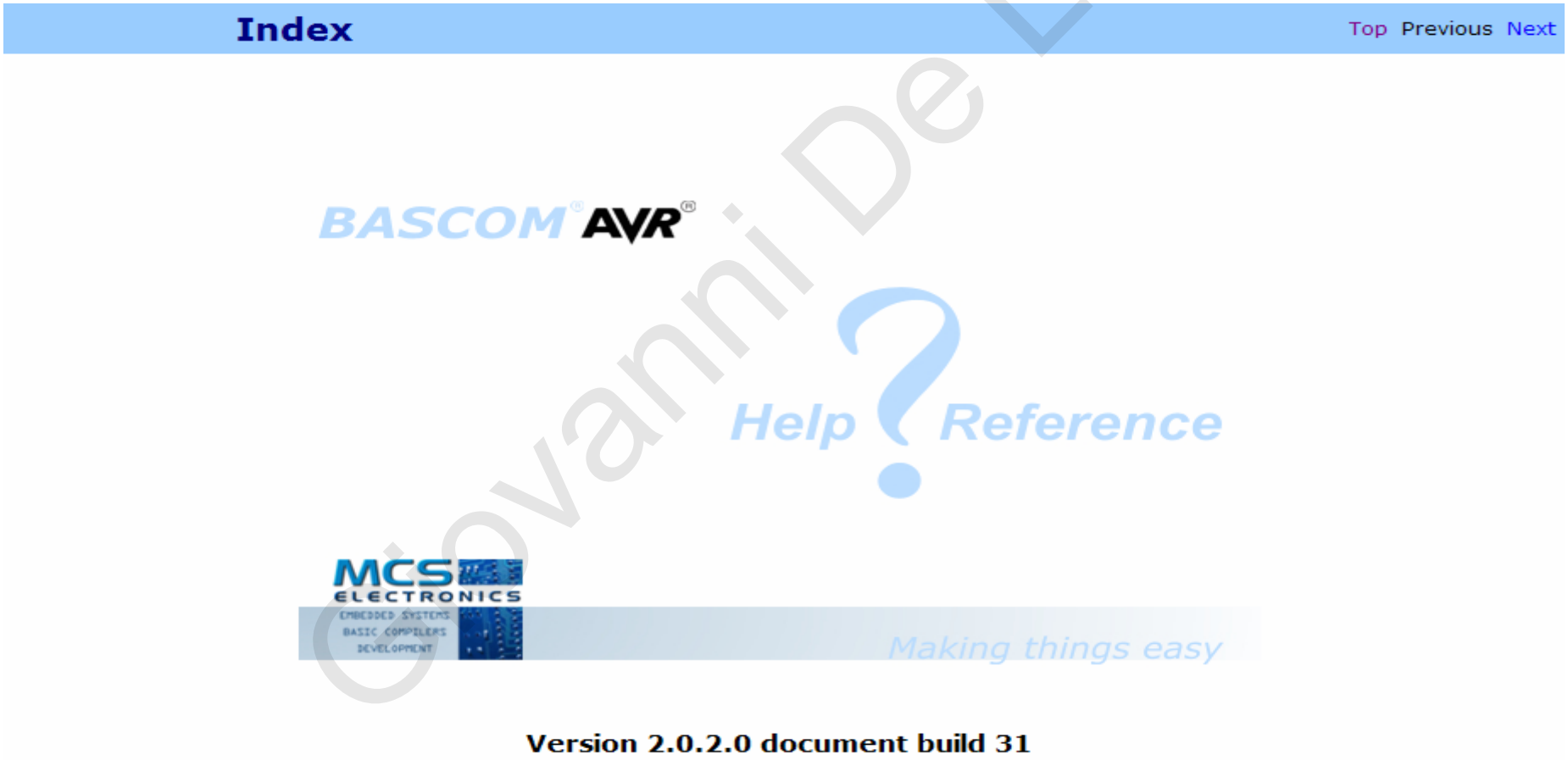
Interrupts -> URXC

Il più semplice & il più usato

```
Enable Interrupts
Enable Urxc
On Urxc Rs232

Dim J As Byte
Dim Rxok As Bit
Rxok=0
Do
  If Rxok = 1 Then
     Rxok = 0
     Print J
  End If
Loop
End

Rs232:
  Input J
  Rxok = 1
Return
```

KEYWORD REFERNCE

in ordine alfabetico



KEYWORD REFERENCE

1Wire routines allow you to communicate with Dallas 1wire chips.
1WRESET , 1WREAD , 1WWRITE , 1WSEARCHFIRST , 1WSEARCHNEXT ,1WVERIFY , 1WIRECOUNT

Conditions execute a part of the program depending on a condition being True or False
IF-THEN-ELSE-END IF , WHILE-WEND , ELSE , DO-LOOP , SELECT CASE - END SELECT , FOR-NEXT

Configuration commands initialize the hardware to the desired state.
CONFIG , CONFIG ACI , CONFIG ADC , CONFIG ADCx , CONFIG BCCARD , CONFIG CLOCK , CONFIG COM1 , CONFIG COM2 , CONFIG DAC , CONFIG DATE , CONFIG DMXSLAVE, CONFIG EEPROM ,CONFIG EXTENDED_PORT , CONFIG PS2EMU , CONFIG ATEMU , CONFIG I2CSLAVE , CONFIG INPUT, CONFIG GRAPHLCD , CONFIG KEYBOARD , CONFIG TIMER0 , CONFIG TIMER1 , CONFIG LCDBUS , CONFIG LCDMODE , CONFIG 1WIRE , CONFIG LCD , CONFIG OSC, CONFIG SERIALOUT , CONFIG SERIALIN , CONFIG SPI , CONFIG SPIx, CONFIG SYSCLOCK , CONFIG LCDPIN , CONFIG PRIORITY , CONFIG SDA , CONFIG SCL , CONFIG DEBOUNCE , CONFIG WATCHDOG , CONFIG PORT , COUNTER0 AND COUNTER1 , CONFIG TCPIP , CONFIG TWISLAVE , CONFIG SINGLE , CONFIG X10 , CONFIG XRAM , CONFIG USB , CONFIG DP , CONFIG TCXX

KEYWORD REFERENCE

A conversion routine is a function that converts a number or string from one form to another.
BCD , GRAY2BIN , BIN2GRAY , BIN , MAKEBCD , MAKEDEC , MAKEINT , FORMAT , FUSING , BINVAL , CRC8 , CRC16 , CRC16UNI , CRC32 , HIGH , HIGHW , LOW , AESENCRYPT , AESDECRYPT

Date Time routines can be used to calculate with date and/or times.
DATE , TIME , DATE$, TIME$, DAYOFWEEK , DAYOFYEAR , SECOFDAY , SECELAPSED , SYSDAY , SYSSEC , SYSSECELAPSED

Delay routines delay the program for the specified time.
WAIT , WAITMS , WAITUS , DELAY

KEYWORD REFERENCE

Directives are special instructions for the compiler. They can override a setting from the IDE.

$ASM , $BAUD , $BAUD1 , $BIGSTRINGS , $BGF , $BOOT , $CRYSTAL , $DATA , $DBG , $DEFAULT , $EEPLEAVE , $EEPROM , $EEPROMHEX , $EEPROMSIZE, $EXTERNAL , $HWSTACK , $INC , $INCLUDE , $INITMICRO , $LCD , $LCDRS , $LCDPUTCTRL , $LCDPUTDATA , $LCDVFO , $LIB , $LOADER , $LOADERSIZE , $MAP , $NOCOMPILE , $NOINIT , $NORAMCLEAR , $NORAMPZ , $PROJECTTIME, $PROG , $PROGRAMMER , $REGFILE , $RESOURCE , $ROMSTART $SERIALINPUT, $SERIALINPUT1 , $SERIALINPUT2LCD , $SERIALOUTPUT , $SERIALOUTPUT1 , $SIM , $SWSTACK , $TIMEOUT , $TINY , $WAITSTATE , $XRAMSIZE , $XRAMSTART , $XA

KEYWORD REFERENCE

File commands can be used with AVR-DOS, the Disk Operating System for AVR.
BSAVE , BLOAD , GET , VER , DISKFREE , DIR , DriveReset , DriveInit , LINE INPUT , INITFILESYSTEM , EOF , WRITE , FLUSH , FREEFILE , FILEATTR , FILEDATE , FILETIME , FILEDATETIME , FILELEN , SEEK , KILL , DriveGetIdentity , DriveWriteSector , DriveReadSector , LOC , LOF , PUT , OPEN , CLOSE

Graphical LCD commands extend the normal text LCD commands.
GLCDCMD , GLCDDATA , SETFONT , LINE , PSET , SHOWPIC , SHOWPICE , CIRCLE , BOX

KEYWORD REFERENCE

I2C commands allow you to communicate with I2C chips with the TWI hardware or with emulated I2C hardware.
I2CINIT , I2CRECEIVE , I2CSEND , I2CSTART,I2CSTOP,I2CRBYTE,I2CWBYTE

I/O commands are related to the I/O pins and ports of the processor chip.
ALIAS , BITWAIT , TOGGLE , RESET , SET , SHIFTIN , SHIFTOUT , DEBOUNCE , PULSEIN , PULSEOUT

Micro statements are specific to the micro processor chip.
IDLE , POWER mode , POWERDOWN , POWERSAVE , ON INTERRUPT , ENABLE , DISABLE , START , END , VERSION , CLOCKDIVISION , CRYSTAL , STOP

KEYWORD REFERENCE

Memory functions set or read RAM , EEPROM or flash memory.
ADR , ADR2 , WRITEEEPROM , CPEEK , CPEEKH , PEEK , POKE , OUT , READEEPROM , DATA , INP , READ , RESTORE , LOOKDOWN , LOOKUP , LOOKUPSTR , CPEEKH , LOAD , LOADADR , LOADLABEL , LOADWORDADR , MEMCOPY

Remote control statements send or receive IR commands for remote control.
RC5SEND , RC6SEND , GETRC5 , SONYSEND

RS-232 are serial routines that use the UART or emulate a UART.
BAUD , BAUD1, BUFSPACE , CLEAR, ECHO , WAITKEY , ISCHARWAITING , INKEY , INPUTBIN , INPUTHEX , INPUT , PRINT , PRINTBIN , SERIN , SEROUT , SPC , MAKEMODBUS

KEYWORD REFERENCE

SPI routines communicate according to the SPI protocol with either hardware SPI or software emulated SPI.
SPIIN , SPIINIT , SPIMOVE , SPIOUT

String routines are used to manipulate strings.
ASC , CHARPOS, UCASE , LCASE , TRIM , SPLIT , LTRIM , INSTR , SPACE , STRING , RTRIM , LEFT , LEN , MID , RIGHT , VAL , STR , CHR , CHECKSUM , HEX , HEXVAL , QUOTE , REPLACECHARS

TCP/IP routines can be used with the W3100/IIM7000/IIM7010 modules.
BASE64DEC , BASE64ENC , IP2STR , UDPREAD , UDPWRITE , UDPWRITESTR , TCPWRITE , TCPWRITESTR , TCPREAD , GETDSTIP , GETDSTPORT , SOCKETSTAT , SOCKETCONNECT , SOCKETLISTEN , GETSOCKET , CLOSESOCKET , SETTCP , GETTCPREGS , SETTCPREGS , SETIPPROTOCOL , TCPCHECKSUM

KEYWORD REFERENCE

Text LCD routines work with normal text based LCD displays.
HOME , CURSOR , UPPERLINE , THIRDLINE , INITLCD , LOWERLINE , LCD , LCDAT , FOURTHLINE , DISPLAY , LCDCONTRAST , LOCATE , SHIFTCURSOR , DEFLCDCHAR , SHIFTLCD , CLS , LCDAUTODIM

Trig and Math routines work with numeric variables.
ACOS , ASIN , ATN , ATN2 , EXP , RAD2DEG , FRAC , TAN , TANH , COS , COSH , LOG , LOG10 , ROUND , ABS , INT , MAX , MIN , SQR , SGN , POWER , SIN , SINH , FIX , INCR , DECR , DEG2RAD , CHECKFLOAT

KEYWORD REFERENCE

Various

This section contains all statements that were hard to put into another group

CONST , DBG , DECLARE FUNCTION , DEBUG, DECLARE SUB , DEFXXX , DIM , DTMFOUT , EXIT , ENCODER , GETADC , GETKBD , GETATKBD , GETRC , GOSUB , GOTO , LOCAL ,ON VALUE , POPALL , PS2MOUSEXY , PUSHALL , RETURN , RND , ROTATE , SENDSCAN , SENDSCANKBD , SHIFT , SOUND , STCHECK , SUB , SWAP , VARPTR , X10DETECT , X10SEND , READMAGCARD , REM , BITS , BYVAL , CALL , #IF , #ELSE , #ENDIF , READHITAG

XMEGA

READSIG

CONFIG

DIRECTIVE	RE-USABLE	XMEGA ONLY
CONFIG 1WIRE	NO	
CONFIG ACXX	YES	X
CONFIG ACI	YES	
CONFIG ADC	NO	
CONFIG ADCx	YES	X
CONFIG ATEMU	NO	
CONFIG BASE	NO	
CONFIG BCCARD	NO	
CONFIG CLOCK	NO	
CONFIG CLOCKDIV	YES	
CONFIG COM1	YES	
CONFIG COM2 also COM3 - COM8	YES	
CONFIG DAC	YES	X
CONFIG DATE	NO	
CONFIG DCF77	NO	
CONFIG DEBOUNCE	NO	
CONFIG DMXSLAVE	NO	
CONFIG DP	NO	
CONFIG EEPROM	NO	X
CONFIG EXTENDED_PORT	NO	
CONFIG GRAPHLCD	NO	
CONFIG HITAG	NO	
CONFIG I2CDELAY	NO	
CONFIG I2CSLAVE	NO	
CONFIG INPUT	NO	
CONFIG INTx	YES	
CONFIG KBD	NO	
CONFIG KEYBOARD	NO	
CONFIG LCD	NO	
CONFIG LCDBUS	NO	
CONFIG LCDMODE	NO	
CONFIG LCDPIN	NO	
CONFIG OSC	YES	X
CONFIG RC5	NO	
CONFIG PORT	YES	
CONFIG PRIORITY	YES	X
CONFIG PRINT	NO	
CONFIG PRINTBIN	NO	
CONFIG PS2EMU	NO	
CONFIG SERIALIN	NO	
CONFIG SERIALIN1	NO	
CONFIG SERIALIN2	NO	
CONFIG SERIALIN3	NO	
CONFIG SERIALOUT	NO	
CONFIG SERIALOUT1	NO	
CONFIG SERIALOUT2	NO	
CONFIG SERIALOUT3	NO	
CONFIG SERVOS	NO	
CONFIG SHIFTIN	NO	
CONFIG SINGLE	NO	
CONFIG SDA	NO	
CONFIG SCL	NO	
CONFIG SPI	NO	
CONFIG SPIx	YES	X
CONFIG SYSCLOCK	YES	X
CONFIG TCXX	YES	X
CONFIG TCPIP	NO	
CONFIG TWI	YES	
CONFIG TWISLAVE	NO	
CONFIG TIMER0	YES	
CONFIG TIMER1	YES	
CONFIG TIMER2 and 3	YES	
CONFIG USB	NO	
CONFIG WATCHDOG	YES	
CONFIG WAITSUART	NO	
CONFIG X10	NO	
CONFIG XRAM	YES	

SAMPLE CABLE PROGRAMMER

Sample Electronics cable programmer

Top Previous

Sample Electronics submitted the simple cable programmer.

They produce professional programmers too. This simple programmer you can make yourself within 10 minutes.
What you need is a DB25 centronics male connector, a flat cable and a connector that can be connected to the target MCU board.

The connections to make are as following:

DB25 pin	Target MCU pin (AT90S8535)	Target MCU M103/M128	Target MCU pin 8515	DT104
2, D0	MOSI, pin 6	PE.0, 2	MOSI, 6	J5, pin 4
4, D2	RESET, pin 9	RESET, 20	RESET, 9	J5, pin 8
5, D3	CLOCK, pin 8	PB.1,11	CLOCK, 8	J5, pin 6
11, BUSY	MISO, pin 7	PE.1, 3	MISO, 7	J5, pin 5
18-25,GND	GROUND	GROUND	GND,20	J5, pin 1

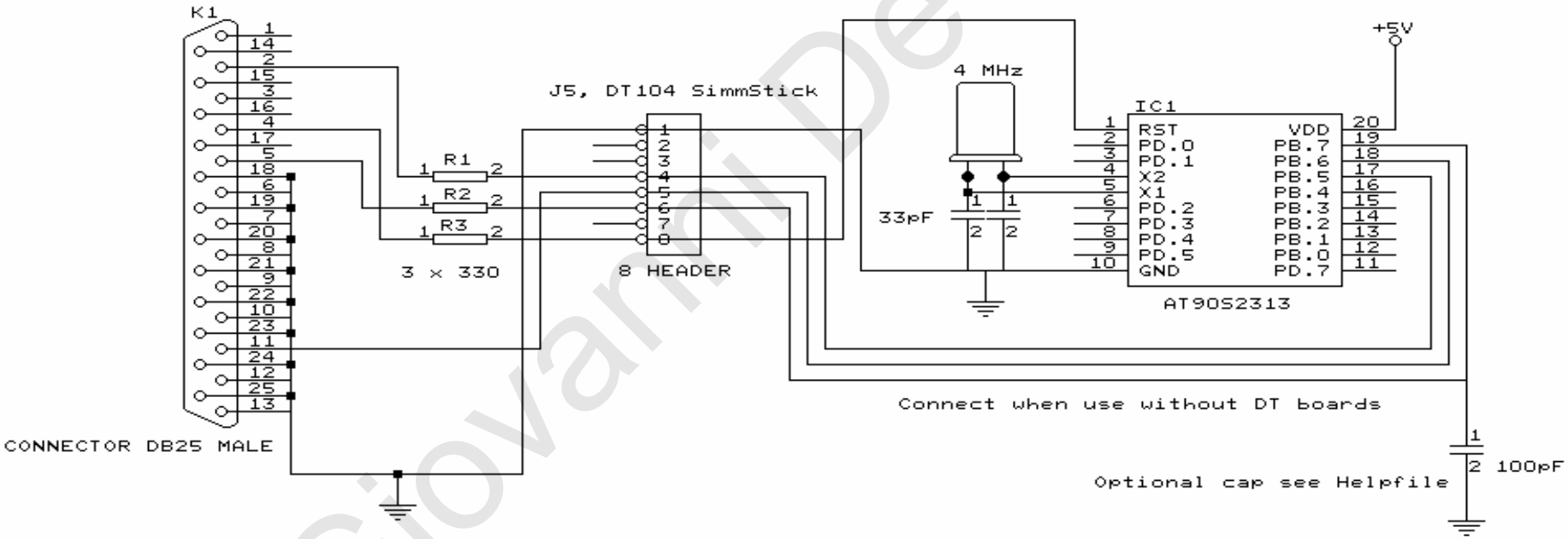
The MCU pin numbers are shown for an 8535! And 8515
Note that 18-25 means pins 18,19,20,21,22,23,24 and 25

You can use a small resistor of 100-220 ohm in series with the D0, D2 and D3 line in order not to short circuit your LPT port in the event the MCU pins are high.
It was tested without these resistors and no problems occurred.

Tip : when testing programmers etc. on the LPT it is best to buy an I/O card for your PC that has a LPT port. This way you don't destroy your LPT port that is on the motherboard in the event you make a mistake!

SAMPLE PROGRAMMER

The following picture shows the connections to make. Both a setup for the DT104 and stand-alone PCB are shown.



Remember

- Definizione File.dat
- Definizione degli Stack
- Definizione del Clock di sistema
- Definizione del Baud-rate
- Configurazione delle Porte I/O
- Configurazione Interrupts
- Configurazione LCD, LCDpin
- Configurazione UART
- Configurazione ADC
- Configurazione TIMER come Timer
- Calcolo Valori Timer con Utility
- Gestione Interrupts

Remember

- Dichiarazione delle Variabili
- Dichiarazione delle Costanti
- Dichiarazione degli Alias

- Configurazione TIMER come PWM
- Configurazione I2C e uso della periferica TWI
- Configurazione della SPI per uso SD-Card DOS