Introduction to BASCOM AVR Compiler

Written by Cholis Safrudin (YD1CHS), mid of April 2011
#1 – kumpulan materi belajar bareng AVR

BASCOM AVR?

Tulisan berikut ditulis bukan sebagai text book, karena isinya dituangkan berdasarkan pengalaman penulis selama belajar otodidak tentang mikrokontroler. Hanya cocok untuk diaplikasikan untuk keperluan non profesional, eksperimen, dan hobby.

Banyak sekali kompiler berbahasa level tinggi (C, BASIC, PASCAL, dll) yang bertebaran di Internet dan sebagian ditawarkan secara gratis. Salah satunya adalah kompiler menggunakan bahasa BASIC untuk mikrokontroler produk AVR, yaitu BASCOM. Hingga saat tulisan ini ditulis, MCS Electronics pemroduksi BASCOM hanya dikembangkan untuk mikrokontroler produk ATMEL, yaitu 8051 dan AVR. Namun disini akan dibahas AVR saja. MSC memberikan kesempatan kepada kita untuk mendownload versi demo dengan keterbatasan besarnya file code hanya sampai dengan 4KB saja, dan bagi yang berminat silakan mengikuti link berikut:
http://www.mcselec.com/index.php?option=com_docman&task=cat_view&gid=99&Itemid=54

Mungkin ada yang bertanya 4KB code itu besar atau kecil. Saya ambil contoh saja AVR ATMEGA-8 memiliki program memory (memori tempat menyimpan kode program) sebesar 8KB, artinya adalah 50% dari total kapasitanya, menurut saya … untuk keperluan belajar dan hobby sudah sangat besar. Mari kita ambil contoh:

- Sebuah HF Frequency Counter dengan tampilan LCD, dilengkapi dengan fitur Autoranging Display, IF Shifted Display dan manipulasi EEPROM yang telah saya buat hanya membutuhkan 40% dari total program memory ATMEGA-8 atau hanya 3.2KB.
- Sebuah FLL (Frequency Locked Loop), yang didalamnya berisi Frequency Counter, Autoranging Display, IF Shifted display, EEPROM Manipulation dan Locking Mechanism dengan PWM hanya membutuhkan 48% dari total program memory ATMEGA-8 atau hanya 3.84KB.

Mudah-mudahan ketiga contoh projects diatas bisa memberikan gambaran bahwa 4KB di dalam pemrograman mikrokontroler adalah sangat-sangat berarti.

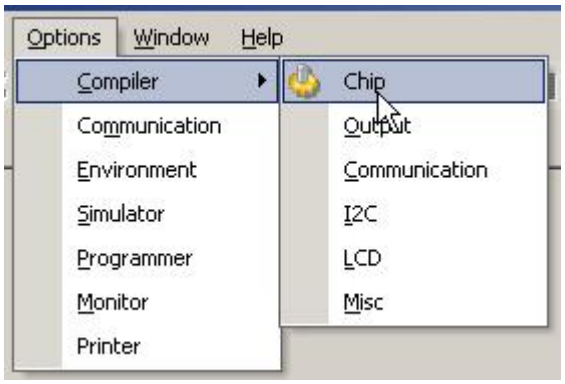
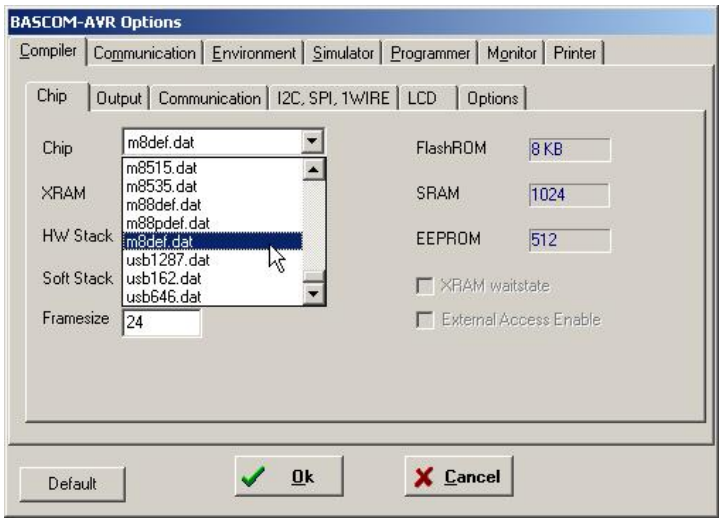
MENGGUNAKAN BASCOM

Seperti yang telah saya sebutkan di atas, bahwa apa yang saya sampaikan disini bisa jadi bukanlah prosedur paling tepat sesuai rekomendasi BASCOM, namun hanya praktik yang biasa saya lakukan. Jadi hanya cocok untuk eksperimen saja dan segala resiko … mohon maaf, silakan ditanggung sendiri. Hehehe.

Saya tidak mau bertele-tele untuk menjelaskan berbagai hal yang sudah mulai menyentuh teori atau hal-hal terkait text book, karena tidak cocok dengan metode belajar kita, yang menuntuk instan serta konsentrasi hal-hal yang biasa kita perlukan saja untuk membangun project-project hobby.

SETTING ENVIRONTMENT BASCOM

Saya hanya asal comot saja penyebutan prosedur ini, silakan anda namakan apa saja yang jelas prosedur ini memastikan bahwa environtment dari kompiler BASCOM sesuai dengan mikrokontroler yang sedang kita buat firmware-nya. Setting ini hanya melekat kepada satu file firmware saja, jadi setiap membuat file firmware harus dilakukan prosedur ini. Berikut langkah utamanya.

Ilustrasi Grafis	Ilustrasi Deskriptif
	1. Langkah pertama, buka aplikasi BASCOM. 2. Buat file firmware yang baru, beri nama sesuai dengan keinginan anda (baca prosedur pembuatan firmware di bawah).
	3. Pada Menu Bar (Atas), click Option >> Compiler >> Chip 4. Pada dialog box Option, pada menu bar click Chip >> Chip: sesuai dengan mikrokontroler yang sedang dikerjakan, contoh disini adalah ATMEGA-8, maka pilih m8def.dat 5. Simpan file tersebut

Sebenarnya kenapa kita harus melakukan setting dasar ini?

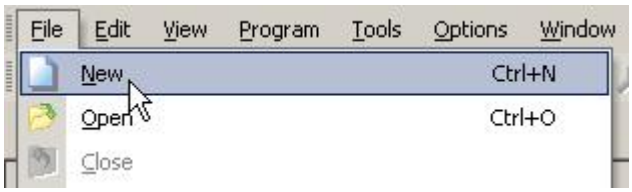
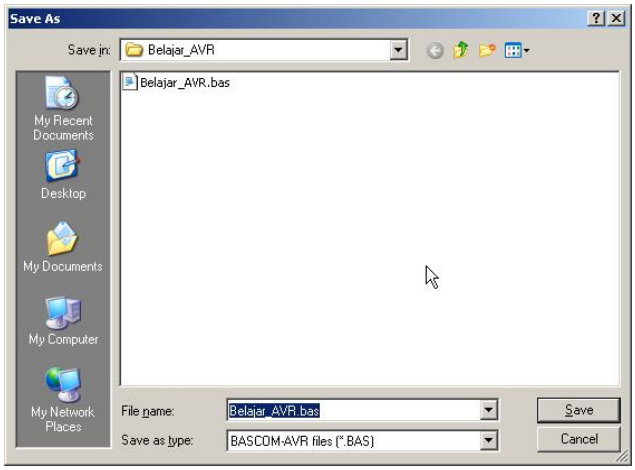
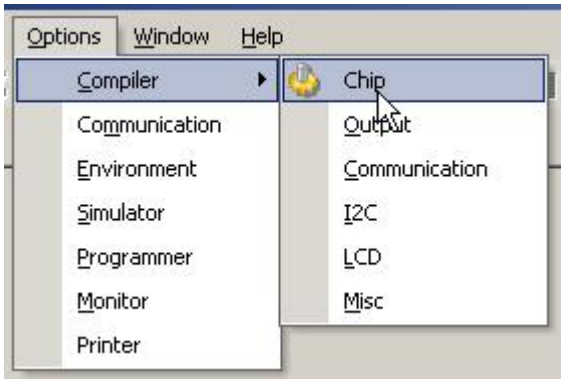
Karena seluruh command atau library yang terkait dengan mikrokontroler tersebut disimpan pada file m8def.dat (untuk ATMEGA-8). Isi file ini diantaranya adalah deklarasi variable-variable default dari mikrokontroler.

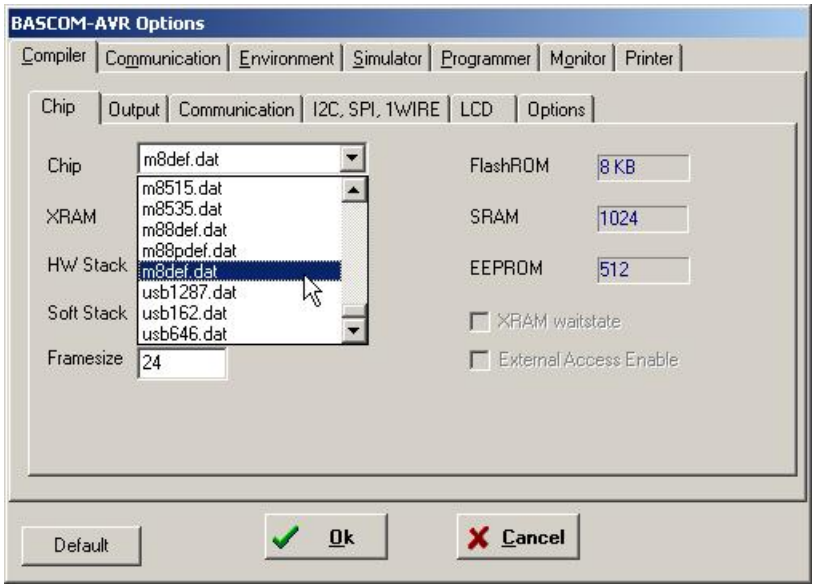
Prosedur ini sebenarnya perlu kita lakukan untuk memaksa kompiler untuk melakukan aktivitasnya bagi mikrokontroler tertentu. Prosedur serupa, namun metodenya lain bisa kita lakukan dengan melakukannya di dalam firmware. Biasanya kita tuliskan/ lakukan pada baris teratas firmware kita, dengan contoh *syntax*:

```
$regfile = "m8def.dat"  → untuk memaksa environtment compiler ATMEGA-8
$crystal = 8000000      → memberitahu compiler bahwa Xtal yang dipakai adalah 8MHz
```

Nah, kebiasaan saya adalah melakukan kedua-duanya. Silakan anda yang memilih sesuai kesukaan anda.

MEMBUAT FIRMWARE BARU

Ilustrasi Grafis	Ilustrasi Deskriptif
	1. Buka BASCOM 2. Pada Menu Bar (Atas), click File >> New
	3. Pada dialog box Save As, beri nama file firmware, saya anjurkan untuk menyimpan 1 project/ firmware dalam 1 direktori berbeda, karena setelah dikompile nanti akan muncul file-file tambahan lainnya, sehingga lebih mudah mengorganisasikannya 4. Click Save
	5. Pada Menu Bar (Atas), click Option >> Compiler >> Chip

Ilustrasi Grafis	Ilustrasi Deskriptif
	6. Pada dialog box Option, pada menu bar click Chip >> Chip: sesuai dengan mikrokontroler yang sedang dikerjakan, contoh disini adalah ATMEGA-8, maka pilih m8def.dat 7. Simpan kembali file tersebut 8. Sekarang file firmware dan environment BASCOM telah sesuai dengan ATMEGA-8

MENULIS FIRMWARE

Gaya cara menulis sebuah program atau firmware sangat bervariasi dan unik untuk masing-masing individu, namun sekali lagi saya akan sampaikan disini kebiasaan saya menuliskan, silakan memilih sendiri gaya penulisan yang sesuai.

```
'-----------------------------------------------------
' 1. Judul Project, deskripsi dan sebagainya
'-----------------------------------------------------
' 2. Environtment untuk kompiler (compiler directives)
'-----------------------------------------------------
' 3. Setup-setup PORT IO, dll
'-----------------------------------------------------
' 4. Deklarasi Variables
'-----------------------------------------------------
' 5. Program Loop Utama
'-----------------------------------------------------
' 6. Kumpulan Function/ Subroutine
'-----------------------------------------------------
```

Berikut contoh gaya penulisan project Digital Voltmeter using AVR yang telah saya buat untuk ATMEGA-16 dengan external Crystal sebesar 8MHz, sekali lagi gaya penulisan tergantung dengan masing-masing orang, silakan menyesuaikan.

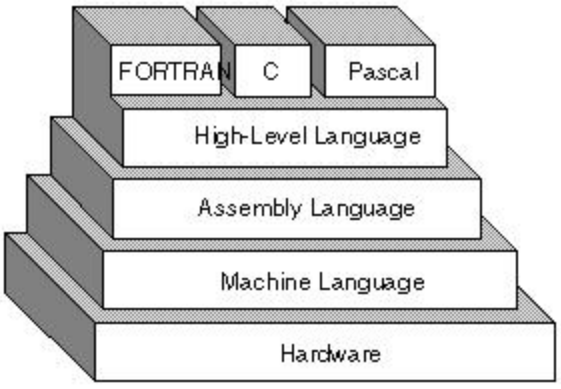
```
'-------------------------------------------------------------------------------
'1. Project title and Description
'   Project: Digital Voltmeter
'   Designed by Cholis Safrudin YD1CHS
'   ----------------------------------
'   Using ADC to sense the voltage variation from the divider circuit
'   Displaying into LCD
'-------------------------------------------------------------------------------
```

```
'2. Compiler Directives
'-----------------------------------------------------------------------
$regfile = "m16def.dat"                                    'Atmega-16
$crystal = 8000000                                         'Xtal 8MHz

'3. Setup Port IO dan Hardware
'-----------------------------------------------------------------------
Config Lcdpin = Pin , Db4 = Portc.4 , Db5 = Portc.5 , _
Db6 = Portc.6 , Db7 = Portc.7 , E = Portc.2 , Rs = Portc.0  '4 bits mode, Port-C
Config Lcd = 16 * 2                                        '16x4 Characters LCD
Cls                                                        'Clear Screen
Cursor Off                                                 'Cursor is Off
Waitms 500                                                 'Wait for 500ms

Config Adc = Single , Prescaler = Auto                     'ADC Configuration
Start Adc                                                  'Start ADC, not necessary since
                                                           'it started automatically
'4. Variable Declaration
'-----------------------------------------------------------------------
Dim Bat_pow As Word                                        'Variable to Store Battery Power
                                                           'Word = 6 Bytes, value from 0 to
65535
Dim Bat_fact As Single                                         'Variable to Store Correction
Factor
                                                           'Single = 3 Bytes, value from 1.5
x 10^-45 to 3.4 x 10^38
Dim Bat_volt As Single                                     'Variable to Store Battery Voltage
                                                           'Single = 3 Bytes, value from 1.5
x 10^-45 to 3.4 x 10^38
Bat_fact = 12 / 508                                        'Set Battery Scale

'5. Print Welcome Messages & Wait until VCO is Steady state
'-----------------------------------------------------------------------
Lcd "AVRLCD Voltmeter"                                     'Print 1st Message
Waitms 100                                                 'wait
Lowerline                                                  'go to line 2
Lcd "by Cholis YD1CHS"                                     'Print 2nd Message
Waitms 2000                                                'wait for 2000ms
Cls                                                        'Clear Screen

'6. Main Program Loop - Battery's Power Testing dan Display Into LCD
'-----------------------------------------------------------------------
' Battery is 12V, using external resistors as scaler 18K//4.7K
' Battery is 12V, Vmax = (4.7K // 18K)*12 = 2.484581
' 5V input is indicates by 1024, so that 2.484581 is indicates by 508
' So that, 1 = 12/508 = This is a correction factor
```

```
Do
   Bat_pow = Getadc(0)                                  'Get Batt Power at ADC0
   Bat_volt = Bat_pow * Bat_fact                        'Get a riil voltage value
   If Bat_pow < 430 Then                                'Battery is low, warn to user
      Lcd "Batt. Power: BAD"                            'Print "Batt. Power: BAD"
      Lowerline                                         'go to line 2
      Lcd Fusing(bat_volt , "##.##") ; " Volts"         'Print Battery Voltage Value
      Waitms 1000                                       'wait
   Else                                                 'Battery is Good
      Lcd "Batt. Power: OK"                             'Print "Batt. Power: OK"
      Lowerline                                         'go to line 2
      Lcd Fusing(bat_volt , "##.##") ; " Volts"         'Print Battery Voltage Value
      Waitms 1000                                       'wait
   End If
   Cls                                                  'Clear Screen
Loop                                                    'Repeated again

End                                                     'end program
```

MENGKOMPILE FIRMWARE

Saya yakin semua telah mengetahui proses apa ini, paling gampangnya ngomong adalah mengubah firmware dari bahasa tingkat tinggi, melingkingkan dengan berbagai hal, kemudian merubahnya ke bahasa mesin, sehingga siap untuk didownload ke uP. Berikut ilustrasi menarik, hirarki dari tingkatan-tingkatan bahasa pemrograman.



Ilustrasi disampaing saya ambil dari http://www.webopedia.com/TERM/H/high_level_language.html

Menempati puncak piramida atau yang biasa disebut dengan bahasa tingkat tinggi (maksudnya bahasa yang paling dekat dengan bahasa manusia). Makin ke bawah posisinya di dalam piramida ini, maka bahasa tersebut makin sulit dimengerti oleh manusia. Makanya di satu tingkat sebelum dasar piramida biasa disebut dengan bahasa mesin (paling dasar sendiri adalah hardware, dalam hal ini misalnya adalah mikrokontroler AVR, PIC, dll).

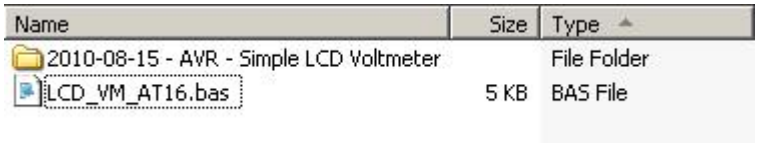
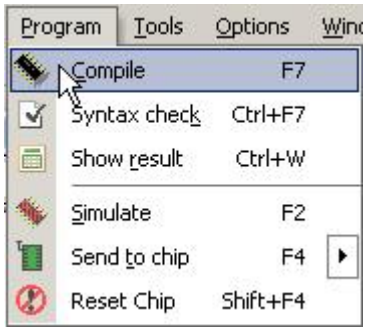
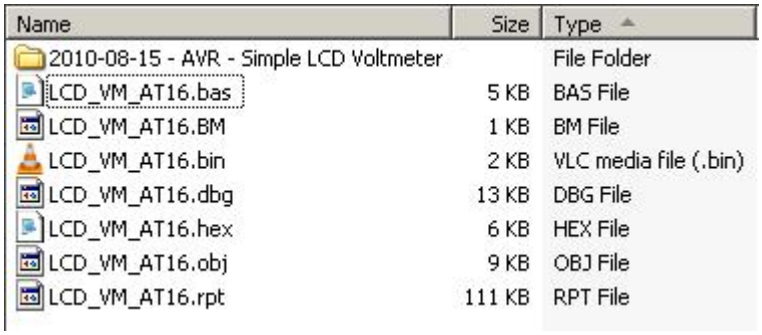
OK, kembali ke tangga tertinggi, yaitu bahasa tingkat tinggi, ilustrasi tersebut mencontohkan beberapa diantaranya: Fortran, C, Pascal … ihik-ihik BASIC nggak disebut karena terlalu gampang dan biasa disebut bahasa anak-anak. Kasihan dech … hehehe. Dalam topik belajar bareng ini, bahasa tingkat tinggi akan diwakili oleh file dengan ekstensi *.bas (bas singkatan dari BASIC).

Pada saat BASCOM melakukan Compile, maka akan diproduksi beberapa file langsung yang mewakili masing-masing tingkatan bahasa di atas. File dengan ekstensi *.asm (asm singkatan dari Assembler) adalah mewakili tingkatan Assembly Language. Dan satu lagi file dengan ekstensi *.hex (hex

singkatan hexadecimal sebab isinya file ini hanyalah kode-kode hexadecimal dan hanya dikenali oleh mesin atau mikrokontroler saja) yang mewakili tingkatan Machine Language. Beberapa file lainnya yang diproduksi oleh BASCOM adalah file-file selama proses linking dan sebagainya, namun kita paling banyak akan berkepentingan hanya pada dua macam file saja *.bas dan *.hex.

Dengan satu kalimat perjalanan firmware tersebut adalah *.bas → *.asm → *.hex, proses ini yang disebut compiling.

Nah cara melakukan compiling di BASCOM ditunjukkan dengan prosedur berikut:

Ilustrasi Grafis	Ilustrasi Deskriptif
	Gambar disamping memperlihatkan isi folder project LCD Voltmeter, tampak sepelum dilakukan compiling hanya ada 1 file *.bas saja
 	1. Untuk mengcompile, lakukan save dulu file *.bas, lalu pada top bar menu Program >> Compile. Bisa juga dilakukan dengan menekan tombol F7, atau bisa juga dengan click icon bergambar  IC. 2. Tunggu sebentar, maka akan muncul dialog BOX status compiling, Flash Used: 11% memperlihatkan jumlah memory yang dipakai. Karena project ini memakai ATMEGA-16 dengan Flash Memory 16kB, maka 11%*16kB = 1.76kB 3. Bila tidak ada pesan error, maka firmware tersebut telah selesai dicompile
	Gambar disamping memperlihatkan pertambahan jumlah file setelah dilakukan compiling. File yang terpenting untuk proses berikutnya adalah *.bas dan *.hex File dengan nama LCD_VM_AT16.hex yang akan kita burning ke mikrokontroler

KESIMPULAN

Tulisan sederhana ini berusaha memberikan petunjuk sederhana dalam mengoperasikan aplikasi BASCOM.

BASCOM juga menyediakan HELP file yang akan memberikan panduan kepada kita dalam upaya lebih mendalami aplikasi ini.

Semoga materi sederhana ini berguna … de yd1chs