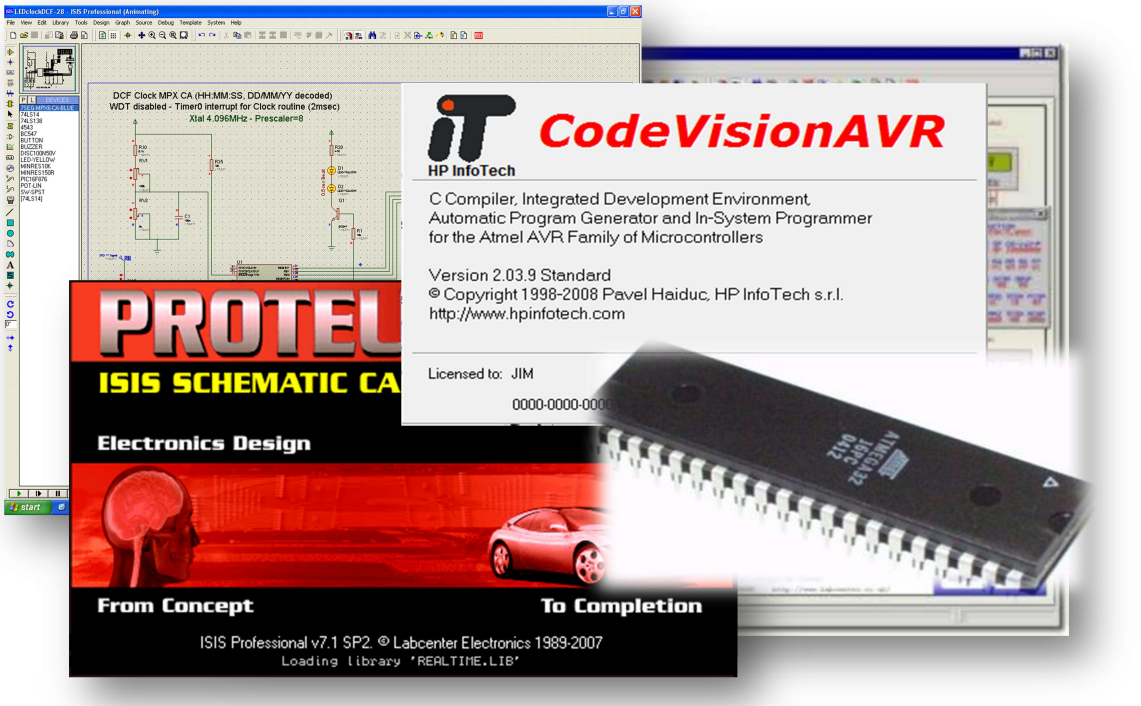
MODUL

PROTEUS PROFESIONAL
UNTUK SIMULASI RANGKAIAN DIGITAL DAN MIKROKONTROLER
(*Materi Lanjutan Mikrokontroler*)



PROGRAM PENGABDIAN MASYARAKAT (PPM)

31 Agustus, 7 & 14 September 2013

TIM

Muhamad Ali, M.T.

Ariadie Chandra N., M.T

Andik Asmara, S.Pd.

PENDIDIKAN TEKNIK ELEKTRO

FAKULTAS TEKNIK UNIVERSITAS NEGERI YOGYAKARTA

Praktik Mikrokontroler

Topik: Pengenalan Bahasa C dan Software CVAVR

A. Struktur Bahasa C pada CVAVR

Penggunaan mikrokontroler yang diterapkan di berbagai alat rumah tangga, otomotif, sampai dengan kendali, membuat mikrokontroler mulai masuk didunia pendidikan. Banyak varian dan type dari mikrokontroler yang dipelajari dan digunakan di dunia pendidikan. Salah satu varian yang banyak dipelajari dan digunakan adalah produk dari ATMEL dengan type keluarga AVR. Banyak software yang dapat digunakan untuk memprogram mikrokontroler keluarga AVR, dengan bahasa pemrograman masing-masing.

Salah satu bahasa pemrograman yang dikembangkan atau digunakan dunia pendidikan adalah bahasa C dengan struktur dan kemudahan yang dimilikinya. Perkembangan bahasa pemrograman yang dimulai dari bahasa tingkat rendah (bahasa assembly/bahasa mesin) sampai dengan bahasa tingkat tinggi (salah satunya bahasa C). Bagi mikrokontroler bahasa assembly merupakan bahasa yang mudah untuk diterjemahkan bagi prosesornya, sehingga dikatakan sebagai bahasa tingkat rendah. Sedangkan bahasa tinggkat tinggi merupakan bahasa yang sulit diterjemahkan oleh prosesor yang ada di didalam mikrokontroler. Pemilihan bahasa C sebagai bahasa pemrograman untuk mikrokontroler dikarenakan mudah dipahami dan diterjemahkan bagi user atau programmer.

Bahasa C memiliki struktur pemrograman yang khusus, selain itu bahasa C memiliki sifat ***case sensitive***. Artinya tersebut adalah bahwa penulisan kata/word program sangat sensitif dengan mendeteksi perbedaan kapital tidaknya huruf yang digunakan. Satu huruf yang berbeda pada satu kata yang diulang, menyebabkan software tidak akan bisa meng-compile seluruh program yang dibuat.

Setiap bahasa pemrograman memiliki type data masing-masing. Type data merupakan jangkauan suatu data yang mampu/dapat dikerjakan/diolah oleh mikroprosesor dalam program yang dibuat. Penggunaan type data ini juga harus sesuai kebutuhan dan disesuaikan dengan fungsi setiap data. Pemilihan penggunaan type data dapat mempengaruhi besarnya memory file yang dibuat. Berikut daftar type data yang dapat digunakan dalam pemrograman bahasa C;

Type	Size (Bits)	Range (jangkauan)
bit	1	0 , 1
Bool, _Bool	8	0 , 1
char	8	-128 sampai 127
unsigned char	8	0 sampai 255
signed char	8	-128 sampai 127
Int	16	-32768 sampai 32767
short int	16	-32768 sampai 32767
unsigned int	16	0 sampai 65535
signed int	16	-32768 sampai 32767
long int	32	-2147483648 sampai 2147483648
unsigned long int	32	0 sampai 4294967295
signed long int	32	-2147483648 sampai 2147483648
Float	32	±1.175e-38 sampai 3.402e38
Double	32	±1.175e-38 sampai 3.402e38

Penggunaan type data bersamaan dengan variable data yang akan digunakan. Penulisan type data sesuai struktur dapat dilihat sebagai berikut:

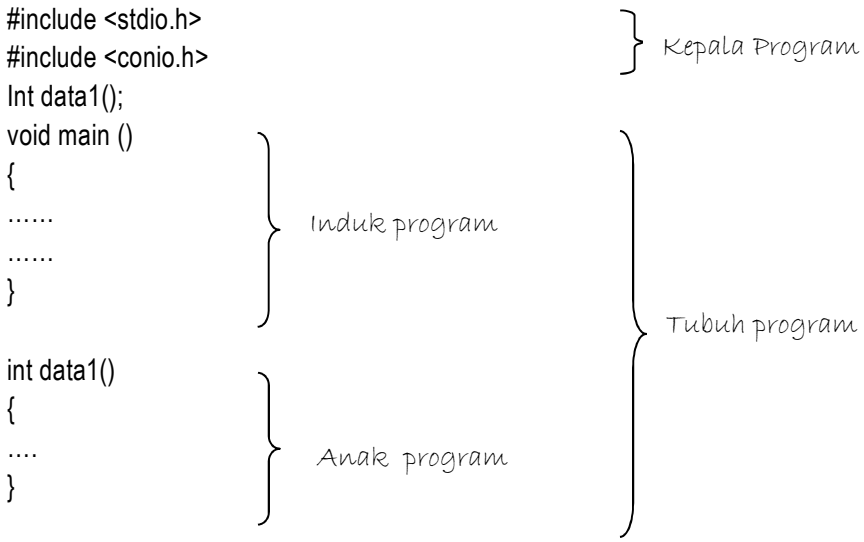
bit data_1; → terdapat variable dengan nama **data_1** dengan type data **bit**

int data_2; → terdapat variable dengan nama **data_2** dengan type data **integer**

Selain tipe data, bahasa C memiliki struktur penulisan akan simbol-simbol operasi aritmatik. Setiap penggunaan simbol-simbol aritmatik memiliki fungsi masing-masing. Berikut table simbol-simbol aritmatik yang digunakan dalam bahasa C;

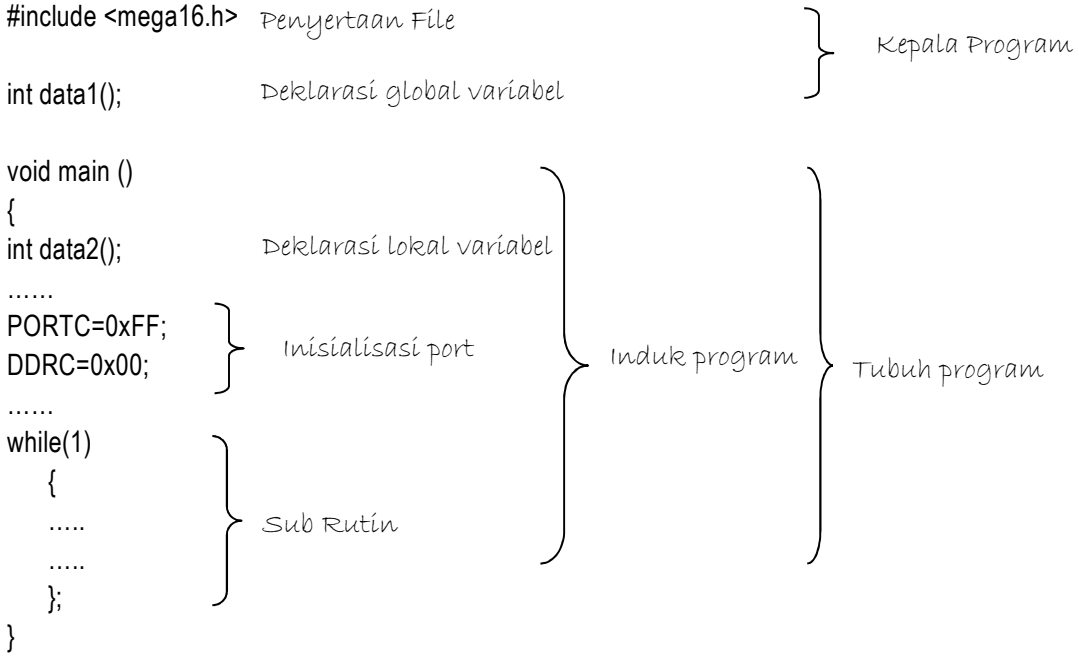
Operator	Keterangan	Operator	Keterangan
+	Penjumlahan	-	Pengurangan
*	Perkalian	/	Pembagian
%	Modulus	++	Penjumalahan berkelanjutan
--	Pengurangan berkelanjutan	=	Sama dengan/memberikan nilai
==	Nilainya sama dengan	~	NOT
!		!=	Hasil tidak sama dengan
<	Lebih kecil	>	Lebih besar
<=	Hasil lebih kecil sama dengan	>=	Hasil lebih besar samadengan
&	Dan/AND	&&	AND (dua kondisi)
|	OR	||	OR (dua kondisi)
^	Faktor pangkat	?:	
<<	Geser bit kekiri	>>	Geser bit kekanan
-=	Hasil pengurangan sama dengan	+=	Hasil penjumlahan sama dengan
/=	Hasil bagi sama dengan	%=	Hasil modulus sama dengan
&=	Hasil peng-AND-an sama dengan	*=	Hasil perkalian sama dengan
^=	Hasil pangkat sama dengan	|=	Hasil peng-OR-an sam dengan
>>=	Hasil penggeseran bit kekanan sama dengan	<<=	Hasil penggeseran bit kekiri sama dengan

Instruksi-instruksi bahasa pemrograman yang ada pada bahasa C tidak semuanya digunakan dalam pemrograman mikrokontroler. Struktur dan urutan penulisan program hampir sama untuk keduanya. Struktur bahasa C memiliki kepala program, dan tubuh program, sedangkan tubuh program bisa terdiri dari induk program dan anak program. Berikut struktur sederhana dari pemrograman bahasa C.



Penggunaan struktur penulisan bahasa pemrograman bahasa C dapat terusun dari sebuah tubuh program yang dapat terdiri dari sebuah induk program dan satu atau lebih anak program. Anak program memiliki fungsi untuk mengerjakan satu blok program yang sering digunakan secara berulang-ulang. Anak program akan diakses oleh induk program sesuai dengan kebutuhan akan sub bagian program tersebut. Sedangkan kepala program berfungsi untuk menyertakan file acuan/library guna mengolah (Compile/Build) program yang telah dibuat.

Penulisan struktur bahasa C didalam CVAVR dapat dilihat seperti dibawah ini:



Deklarasi sebuah variable dapat digolongkan menjadi dua, yaitu local variable dan global variable. *Local variable* dipakai dan hanya dapat diakses pada sub program tempat mendeklarasikannya, sedangkan *global variable* dipakai dan dapat diakses seluruh bagian program. *Inisialisasi PORT* digunakan untuk memfungsikan PORT yang dituju sebagai masukan/keluaran serta nilai defaultnya. Sedangkan bagian *sub rutin* adalah blok program yang akan selalu dikerjakan terus-menerus oleh mikroprosesor selama mikrokontroler hidup.

Beberapa Instruksi-instruksi dalam bahasa C yang sering digunakan dapat ditulis sebagai berikut:

No	Fungsi	Bahasa Pemrograman
1	Syarat	***if (kondisi)*** *{* *…….(aksi yang dikerjakan)* *};*
2.	Percabangan	***if (kodisi)*** *{* *……(aksi yang dikerjakan)* *}* ***else if** (kondisi)* *{* *…..(aksi yang dikerjakan)* *}* *….*

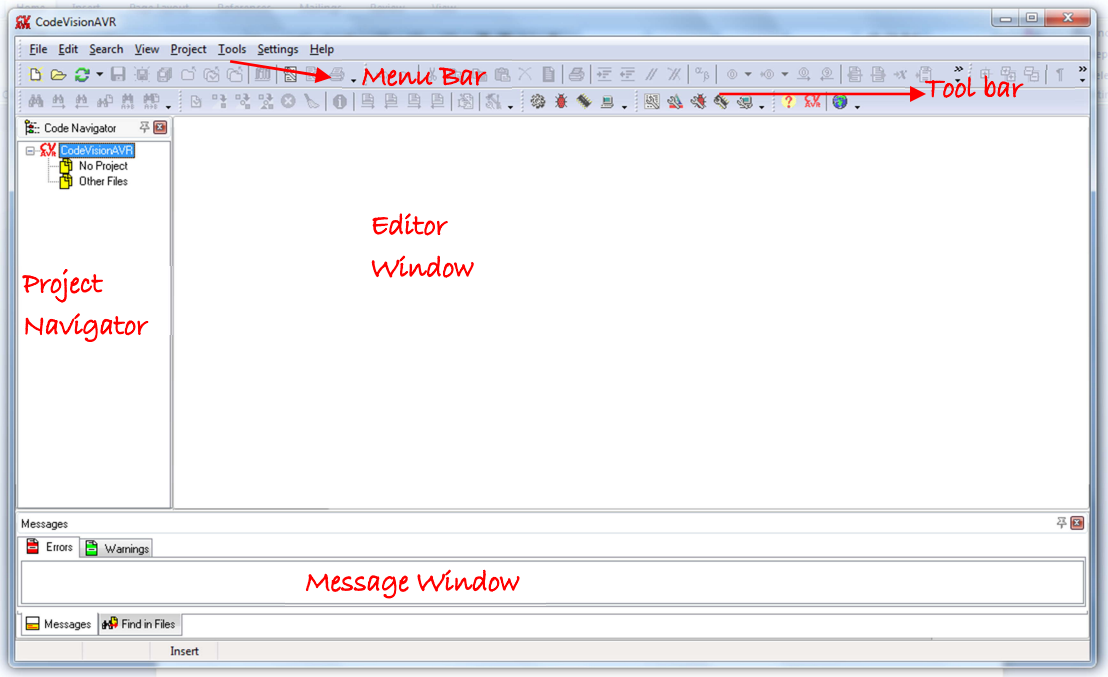
		..... **else** { .....(aksi yang dikerjakan) };
3.	Percabangan	**switch (**variable**)** { **case** nilai_variabel_ke-1**:** { ..... (aksi yang dikerjakan) } **case** nilai_variabel_ke-2**:** { ..... (aksi yang dikerjakan) } .... **default:** { .... (aksi yang dikerjakan) } }
4.	Melompat	**goto** alamat_tujuan; .... alamat_tujuan: ....
5.	Melompat keluar dari perulangan	**break;**
6.	Perulangan	**while (**kondisi**)** { .....(aksi yang dikerjakan) }
7.	Perulangan	**do** { .....(aksi yang dikerjakan) } **while (**syarat**);**
8.	Perulangan	**for (**nilai_awal,syarat,operasi++/--**)** { .....(aksi yang dikerjakan) **};**

dan masih banyak instruksi program lainnya *(ref: Pemrograman Bahasa C, Abdul Kadir).*

B. Pengenalan Software Code Vision AVR (CVAVR)

CodeVisionAVR merupakan sebuah cross-compiler C, Integrated Development Environtment (IDE), dan Automatic Program Generator yang didesain untuk mikrokontroler buatan Atmel seri AVR. Cross-compiler C mampu menerjemahkan hampir semua perintah dari bahasa ANSI C, sejauh yang diijinkan oleh arsitektur dari AVR, dengan tambahan beberapa fitur untuk mengambil kelebihan khusus dari arsitektur AVR dan kebutuhan pada sistem embedded.

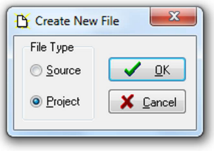
CodeVisionAVR juga mempunyai Automatic Program Generator bernama CodeWizardAVR, yang mengizinkan Anda untuk menulis, dalam hitungan menit, semua instruksi yang diperlukan untuk membuat beberapa fungsi-fungsi tertentu. Dengan fasilitas ini mempermudah para programmer pemula untuk belajar pemrograman mikrokontroler menggunakan CVAVR. Secara garis besar bagian-bagian CVAVR dapat diuraikan seperti gambar berikut ini;



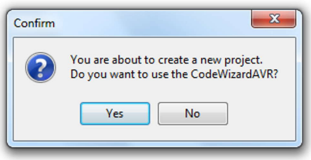
Gb.1 Tampilan window utama CVAVR

Untuk memulai menulis program didalam software CVAVR terlebih dahulu melakukan langkah-langkah sebagai berikut:

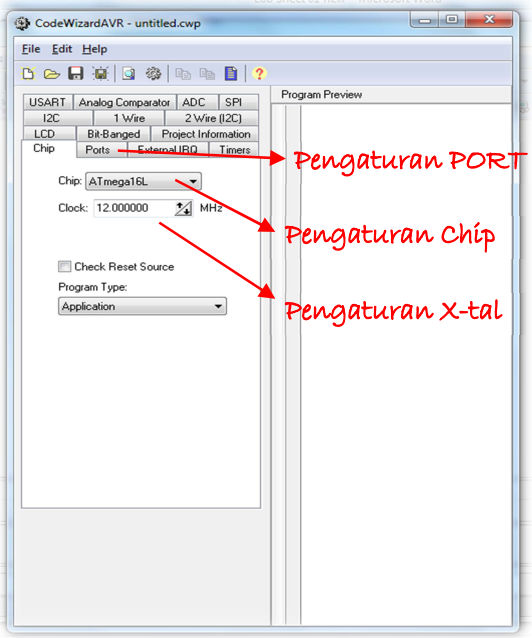
1. File → New → Pilih Project



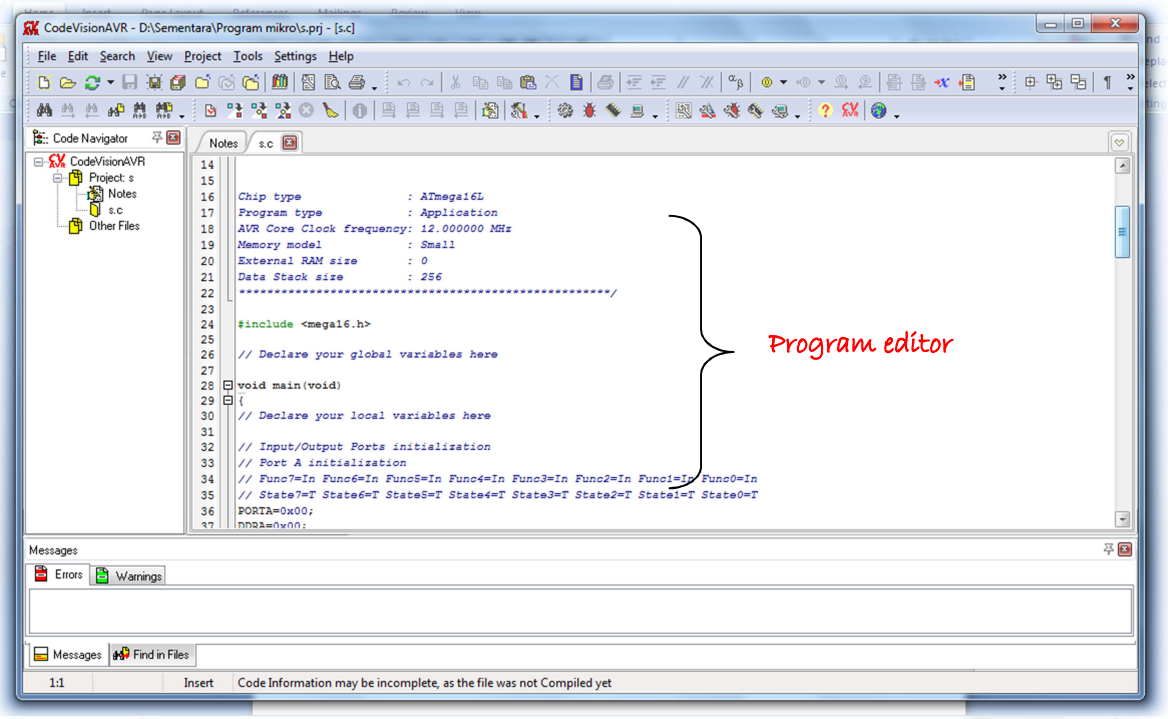
2. Selanjutnya akan muncul window konfirmasi menggunakan AGP CodeWizardAVR → Yes



3. Window CodeWizardAVR digunakan untuk pengaturan PORT dan fasilitas sesuai dengan fungsi yang diinginkan



4. Setelah selesai dengan pengaturan pada CodeWizardAVR pilih File → Generate, Save and exit *(catatan: pemeberian nama file sebanyak 3x; dengan nama file yang sama; hindari kalimat yang panjang, capital dan spasi)*
5. Selesai pemberian nama file, akan muncul window utama editor program seperti berikut



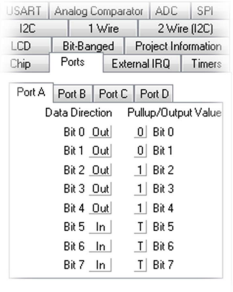
Beberapa langkah instruksi diatas akan sering dilakukan apabila setiap ingin membuat project baru. Untuk itu, langkah-langkah pembuatannya harus dihafal dan dipahami.

Praktik Mikrokontroler

Topik: Output

A. Kajian Teori

Pheripheral mikrokontroler keluarga AVR (ATMega16/8535) memungkinkan untuk diset sebagai keluaran dan masukan. Pengaturan tersebut dapat dilakukan dengan bantuan Code Wizard AVR pada salah satu port yang diinginkan. Penggunaan program secara langsung juga dapat dilakukan untuk megatur fungsi dari pada setiap port pada mikrokontroler. Berikut gambaran secara umum;



Penulisan secara program:
PORTA=0xFF;
DDRA=0xFF;

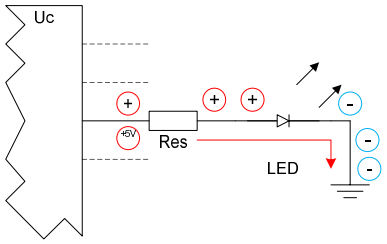
Gb.1a. Pengaturan Port Mikrokontroler CodeWizard 1b. Pengaturan PORT Secara Program

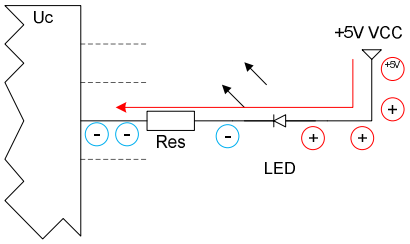
Sebagai contoh pengaturan port-A pada gambar diatas (1a), menunjukkan pada data direction sebagai Output memiliki nilai keluaran dua buah, yaitu 0 (low=0) dan 1 (high=+5V). Nilai keluaran pengaturan port mikro menentukan nilai default awal dari keluarannya. Sedangkan pengaturan port secara program (1b) seperti penulisan diatas, memiliki fungsi pada setiap instruksi sebagai berikur;

PORTA=0xXX; → pengaturan terhadap nilai keluaran Port –A
- *0xFF* → nilai keluaran Port-A pada setiap Bit = Tinggi (*0b11111111*)
- *0x00* → nilai keluaran Port-A pada setiap Bit = Rendah(*0b0000000*)
- *0x0F* → nilai keluaran Port-A pada 4 bit LSB = Tinggi dan 4 bit MSB = Rendah (*00001111*)

DDRA=0xXX; → pengaturan terhadap fungsi port-A
- *0xFF* → Nilai pengaturan port-A pada semua bit sebagai keluaran/output (*0b11111111*)
- *0x00* → Nilai pengaturan port-A pada semua bit sebagai masukan/input (*0b00000000*)
- *0x0F* → Nilai pengaturan port-A pada 4 bil LSB sebagai keluaran dan 4 bit MSB sebagai masukan (*0b00001111*)

Kembali sebagai fungsi keluaran, dapat mempengaruhi kerja dari pada hardware atau rangkaian yang nantinya akan diakses. Ada beberapa tipe kerja rangkaian untuk mengaksesnya, yaitu Aktif LOW dan Aktif High. Aktif LOW merupakan kerja rangkaian yang dapat dioperasikan/di –ON –kan dengan diberi logika rendah (“0”/0). Sedangkan Aktif HIGH merupakan kerja rangkaian yang dapat dioperasikan/di-ON-kan dengan diberi logika tinggi (“1”/+5V). Berdasarkan skematik dari kerja rangkaian diatas dapat digambarkan sebagai berikut;





Gb. 2a. Rangkaian dengan kerja Aktif High 2b. Rangkaian dengan kerja Aktif Low

Pengaturan nilai keluaran setiap port disesuaikan dengan prinsip kerja rangkaian yang akan dioperasikan. Secara logika untuk pengaturan nilai keluaran pada setiap port harus berkebalikan dengan logika untuk menghidupkan/mengoperasikan rangkaian tersebut. Misalkan, rangkaian LED aktif low, maka nilai keluaran pada CodeWizard harus diatur dengan nilai 1/Tinggi. Sedangkan sebaliknya, untuk rangkaian LED aktif high, maka nilai keluaran diatur dengan nilai 0/Rendah. Modul Led yang digunakan dalam praktik memiliki kerja aktif low, sehingga nilai keluaran port-A harus diatur menjadi Tinggi. Pengaturan tersebut dengan tujuan untuk mematikan rangkaian saat pertama kali dihidupkan, atau bisa dikatakan tidak langsung bekerja.

Instruksi yang digunakan dalam CVAVR untuk meng-akses atau mengeluarkan data (output) ke salah satu Port sudah baku. Ada dua macam peng-akses-an port, yaitu secara bersamaan dan secara satu-persatu pin/bit. Sebagai contohnya adalah berikut ini (Akses ke -PORTA);

Instruksi CVAVR Secara bersamaan:

PORTA=0x0F; → pada 8 bit data PORTA akan mengeluarkan data 00001111

atau

PORTA=0b00001111; → pada 8 bit data PORTA akan mengeluarkan data 00001111

Instruksi CVAVR Secara per-bit:

PORTA.0=0; → Pada bit ke-0 PORTA akan mengeluarkan data 0 (low/0)

⋮

PORTA.3=0; → Pada bit ke-3 PORTA akan mengeluarkan data 0 (low/0)

PORTA.4=1; → Pada bit ke-4 PORTA akan mengeluarkan data 1 (high/+5V)

⋮

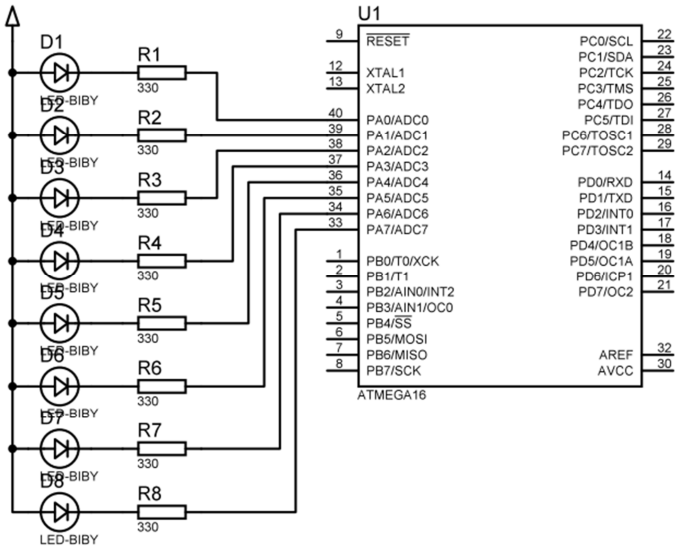
PORTA.7=1; → Pada bit ke-4 PORTA akan mengeluarkan data 1 (high/+5V)

dst.

Instruksi diatas dapat di ilustrasikan sebagai berikut:

hexa	*0x*	*0*				*F*			
biner	*0b*	*0*	*0*	*0*	*0*	*1*	*1*	*1*	*1*
logika		low	low	low	low	high	high	high	high
V out		0	0	0	0	5V	5V	5V	5V
		Bit ke-7	Bit ke-6	Bit ke-5	Bit ke-4	Bit ke-3	Bit ke-2	Bit ke-1	Bit ke-0
		MSB bit							LSB bit

B. Gambar Rangkaian Simulasi



Gb. 3. Skematik Simulasi Modul LED

Tabel 1. Keyword Komponen

Nama Komponen	Keyword ISIS
Mikrokontroler ATMega16	ATMEGA16
Resistor	RES
LED	LED-BIBY

C. Contoh program

C.1. Program LED menyala semua secara bersama

```
#include <mega16.h>

void main(void)
{
……..
……..
while (1)
    {
    PORTA=0x00;
    };
}
```

C.2. Program LED Led-1 On, Led-2 Off, Led-3 On, Led-4 Off, Led-5 On, Led-6 Off, Led-7 On, Led-8 Off

```
#include <mega16.h>

void main(void)
{
……..
……..
```

```
while (1)
    {
    PORTA.0=0;
        PORTA.1=1;
        PORTA.2=0;
        PORTA.3=1;
        PORTA.4=0;
        PORTA.5=1;
        PORTA.6=0;
        PORTA.7=1;
    };
}
```

C.3. Program LED berkedip bersamaan

```
#include <mega16.h>
#include <delay.h>

void main(void)
{
……..
……..
while (1)
    {
    PORTA=0xFF;
    delay_ms(1000);
    PORTA=0x00;
    delay_ms(1000);
    };
}
```

C.4. Program LED geser bergantian ke-kanan

```
#include <mega16.h>
#include <delay.h>

void main(void)
{
……..
……..
while (1)
    {
    PORTA=0b11111111;
    delay_ms(1000);
    PORTA=0b11111110;
    delay_ms(1000);
    PORTA=0b11111101;
    delay_ms(1000);
    PORTA=0b11111011;
```

```
        delay_ms(1000);
        PORTA=0b11110111;
        delay_ms(1000);
        PORTA=0b11101111;
        delay_ms(1000);
        PORTA=0b11011111;
        delay_ms(1000);
        PORTA=0b10111111;
        delay_ms(1000);
              PORTA=0b01111111;
        delay_ms(1000);
        };
}
```

D. Latihan Mandiri

D.1. Buatlah program untuk menyalakan LED geser bergantian kekiri!

D.2. Buatlah program untuk menyalakan LED geser bergantian kekanan-kekiri!

D.3. Aplikatif:

Buatlah program untuk menjalankan 3 buah motor nyala berurutan, dengan jeda waktu penyalaan antara motor satu dengan yang lain mendekati 5 detik (setelah ketiga motor nyala, tetap dipertahankan posisi tersebut/tidak berulang)!

D.4. Aplikatif:

Buatlah program untuk menjalankan 3 buah motor nyala bergantian, dengan jeda waktu penyalaan antara motor satu dengan yang lain mendekati 5 detik (setelah motor ketiga nyala, kembali penyalaan pada motor kesatu)!

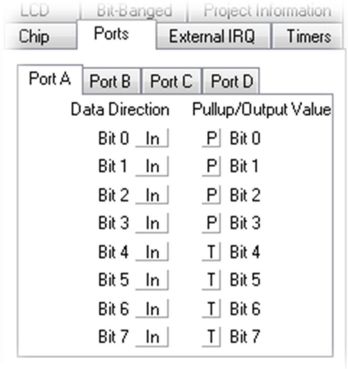
Praktik Mikrokontroler

Topik: INPUT - OUTPUT

A. Kajian Teori

Pada topic sebelumnya sudah dikenalkan cara meng-akses port sebagai keluaran, sehingga selanjutnya pada topic ini akan digabung dengan masukan atau input. Masukan untuk mikrokontroler bisa dari saklar, sinyal logika, atau rangkaian lain yang memiliki keluaran. Sebagai dasar mempelajari masukan pada mikrokontroler, pada topic ini akan digunakan saklar/button sebagai masukannya.

Pengaturan inisialisasi port pada mikrokontroler dapat dilakukan dengan dua cara, secara menggunakan CodeWizardAVR, atau secara penulisan program. Sedangkan sebagai kondisi port sebagai masukan terdapat dua karakter yaitu 'P' dan 'T'. 'P' merupakan kependekan dari Pull Up, sedangkan 'T' merupakan kependekan dari Toggle. Berikut contoh pengaturan port mikro secara CodeWizardAVR atau tertulis;



Secara penulisan Program adalah sebagai berikut:

PORTA=0xF0;
DDRA=0x00;

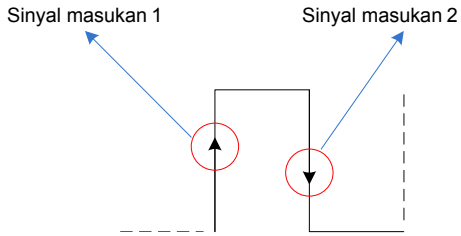
Gb.1. Pengaturan Port sebagai masukan secara CodeWizardAVR

Seperti yang telah dijelaskan pada topic sebelumnya dalam pengaturan secara tertulis inisialisasi port masukan memiliki fungsi sebagai derikut;

PORTA =*0x00;* → Kondisi 8 bit pada PORTA semuanya Toggle ('T')
=*0xFF;* → Kondisi 8 bit pada PORTA semuanya Pull up ('P')
=*0xF0;* → Kondisi 4 bit LSB PORTA berfungsi sebagai Toggle ('T'), sedangkan 4 bit MSB PORTA berfungsi sebagai Pull up ('P').

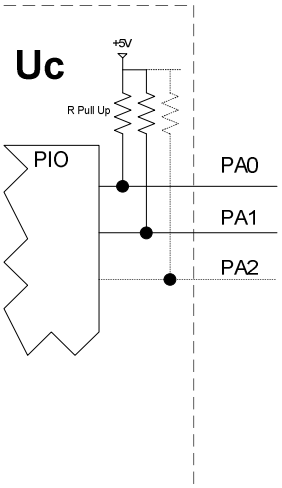
DDRA =*0x00;* → Semua 8 bit pada PORTA berfungsi sebagai masukan.

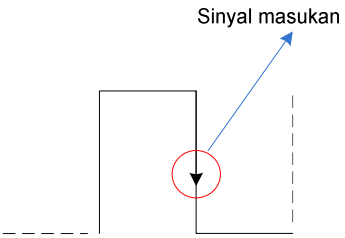
Fungsi pada kondisi Toggle masukan mikrokontroler akan membaca sinyal setiap ada perubahan logika. Perubahan itu bisa dari logika tinggi (1) menuju rendah (0) dikatakan sebagai kondisi *falling edge*, atau sebaliknya dari logika rendah (0) ke tinggi (1) dikatakan sebagai kondisi *rising edge*. Prinsip tersebut mengakibatkan dalam pembacaan satu gelombang sinyal terdapat dua kali sinyal masukan ke mikrokontroler. Berikut secara ilustrasi pembacaannya;



Gb.2. Pembacaan sinyal masukan pada fungsi Toggle

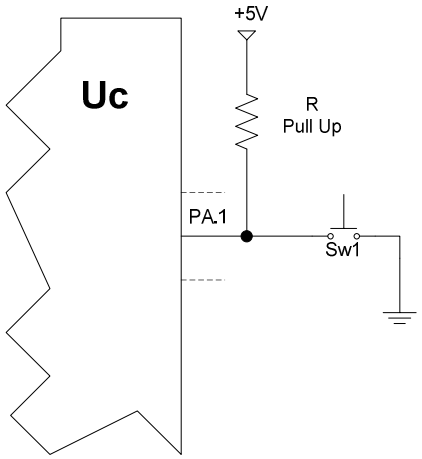
Kondisi pengaturan port masukan pada Pull up ('P') mendeteksi/membaca masukan hanya satu kali dalam satu gelombang masukan. Pembacaan tersebut pada saat gelombang pada kondisi dari logika tinggi (1) ke logika rendah (0) dikatakan sebagai kondidi *falling edge*. Selain itu bahwa pada pengaturan kondisi pull up mengeset pin masukan didalam mikro terhubung dengan VCC (5V) melalui resistor. Resistor yang memiliki prinsip seperti tersebut dinamakan sebagai *resistor pull up*. Resistor ini menjaga agar pada pin masukan yang telah diatur berlogika tinggi, dan menunggu sinyal masukan dengan logika rendah untuk meng-aktif-kannya. Berikut secara ilustrasi prinsip kerja masukan pada kondisi pull up;





Gb.3a. Kondisi didalam mikrokontroler pada posisi Pull Up
Gb.3b. Pembacaan sinyal pada kondisi Pull Up

Kebanyakan rangkaian masukan ke mikrokontroler mengambil prinsip *falling edge* sebagai sinyal tanda aktif, atau bisa dikatakan memiliki logika aktif jika sinyal masukannya rendah (low). Apabila terhubung dengan sebuah masukan dari saklar/button, maka saklar saat tertutup terhubung dengan ground (Gnd). Sebaliknya, apabila saklar dalam kondisi terbuka akan mempertahankan logika tinggi (high) pada masukan, dikarenakan terdapat resistor pull up yang menjaga jalur data masukan dalam kondisi tinggi. Walaupun dalam pengaturan kondisi masukan sudah di pull up, akan tetapi untuk mengamankan kondisi datanya , maka akan dipasang resistor pull up lagi di luar pada system minimum. Berikut ilustrasi skematiknya;



Gb.4 Pemasangan Resistor Pull Up External

Pengambilan data atau mendeteksi sinyal masukan dari luar dilakukan mikroprosesor dengan instruksi program yang telah ditentukan. Instruksi pemrograman dalam bahasa C pada Code Vision AVR yaitu "*PINx*". Berikut penjabaran penulisan program untuk membaca sinyal data dari luar;

PINA==0b11111101; → pada PORTA bit 1 berlogika rendah (terdapat sinyal masukan), bit 0 dan bit 2-7 berlogika 1 (tidak terdapat sinyal masukan)

Atau,

PINA.1==0; → Pada PORTA bit 0 berlogika rendah yang menunjukkan terdapat sinyal masukan (saklar tertutup)

Instruksi program masukan PIN biasanya digunakan bersamaan dengan dengan intruksi syarat pada bahasa C. Salah satunya yaitu penggunaanya bersama instruksi "IF", berikut contohnya;

```
if(PINA.1==0)
    {
    …….. (aksi yang dilakukan)
    };
```

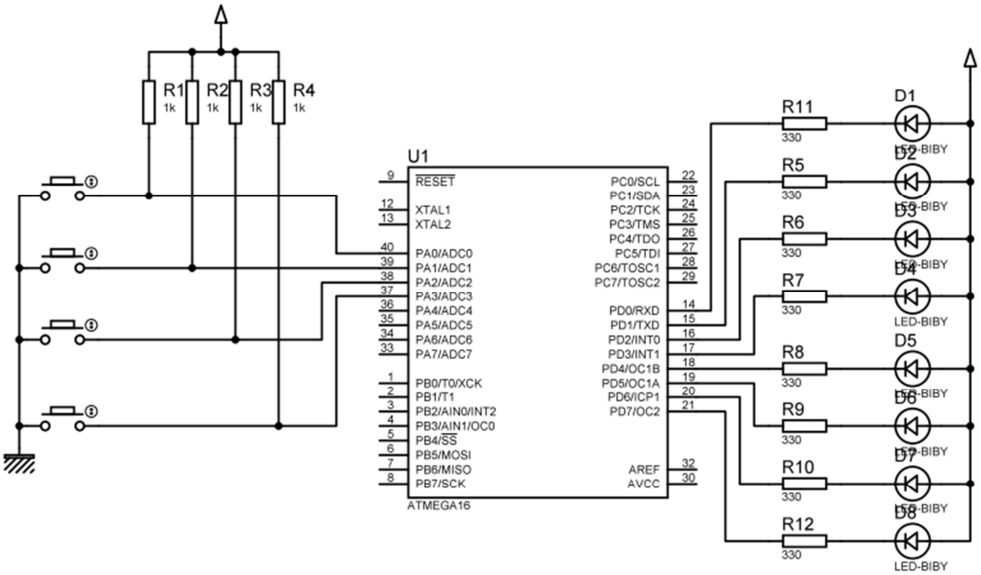
Atau pada perulangan "while";

```
while(PINA.1==0)
    {
    …….. (aksi yang dilakukan berulang-ulang)
    };
```

Penggunaan symbol "==" (sama dengan dua kali), mempunyai fungsi sebagai pertanyaan kondisi pada PIN yang dituju. Apakah kondisi PIN masukan dalam kondisi rendah atau pada kondisi tinggi. Sedangkan untuk mengetahui hasil dari pembacaan masukan program masukan (INPUT) digabung dengan program keluaran (OUTPUT).

B. Gambar Rangkaian Simulasi

Pada labsheet kali ini akan menggunakan modul tambahan (selaian system minimum) sebagai berikut;



Gb. 5. Skematik Simulasi Modul LED & Switch

Tabel 1. Keyword Komponen

Nama Komponen	Keyword ISIS
Mikrokontroler ATMega16	ATMEGA16
Resistor	RES
LED	LED-BIBY
Push Button	BUTTON

C. Contoh program

C.1. Jika saklar SW0 ditekan (tertutup) LED0 akan menyala, dan sebaliknya.

```
.......
......
while(1)
        {
        if(PINA.0==0)
            {
            PORTD.0=0;              //LED bit 0 ON
            }
        else
            {
            PORTD.0=1;              //LED bit 0 OFF
            };
        };
```

C.2. Saklar Swo untuk menghidupkan LED, Saklar Sw1 untuk mematikan LED

```
.....
.....
while(1)
    {
        if(PINA.0==0)
            {
            PORTD.0=0;              //LED 0 ON
            }
        if(PINA.1==0)
            {
            PORTD.0=1;              //LED 0 OFF
            };
    }
```

D. Latihan Mandiri

D.1. Buatlah program apabila ditekan Sw1 ditekan nyala LED berjalan bergantian ke kiri, apabila ditekan Sw2 nyala LED berjalan bergantian ke kanan.

D.2. Aplikatif

Buatlah program untuk menjalankan dua buah motor nyala bergantian terus menerus dengan jeda pindah mendekati 2 detik.

Sw1 = sebagai START (mulai menjalankan motor bergantian)

Sw2 = sebagai STOP (menghentikan bergantinya jalan motor/berhenti pada salah satu motor)

Sw3 = sebagai RESET (menghentikan dan mematikan semua motor)

D.3. Aplikatif

Buatlah program apabila Sw1 ditekan pertama LED menyala dan akan tetap menyala saat saklar dilepas, pada penekanan Sw1 yang ke dua akan mematikan LED, dan seterusnya.

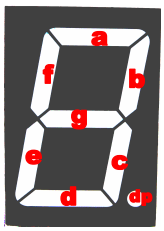
Praktik Mikrokontroler

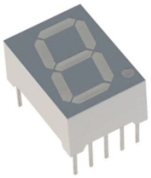
Topik: 7-Segment

A. Kajian Teori

Seven Segment (7-Seg) tidak asing lagi, yang sering dijumpai pada kehidupan sehari-sehari, seperti pada jam tangan, jam dinding, mesin cuci, serta alat-alat elektronik lainnya. Walaupun bisa dikatakan bahwa 7-Seg merupakan tampilan yang sudah lama ada, akan tetapi trend pengguanaanya tidak bisa digantikan dengan tampilan lain. Hal inilah yang menjadikan 7-seg tetap masih dipertahankan sebagai salah satu tampilan pada segala jenis alat-alat elektronik.

Pada dasarnya 7-seg terdiri dari 7 buah LED, yang dirangkai menjadi satu sehingga dapat membentuk angka-angka 0-9. Pada perkembangannya 7-seg ditambahkan satu bagian lagi sebagai tanda titik (dot point). Berdasarkan standart penamaan setiap bagian pada 7-seg dapat dituliskan dengan ilustrasi gambar sebagai berikut;

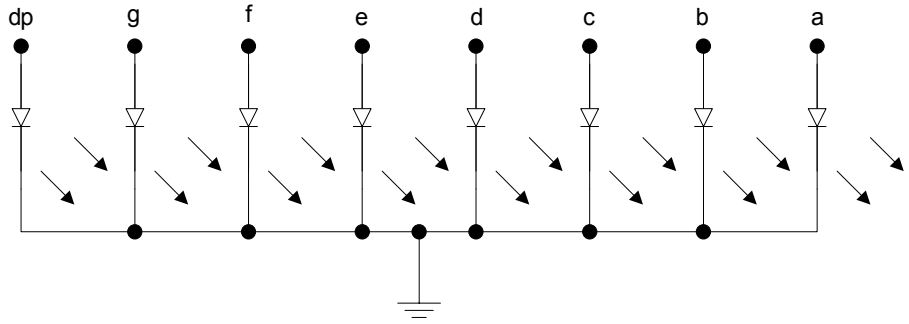




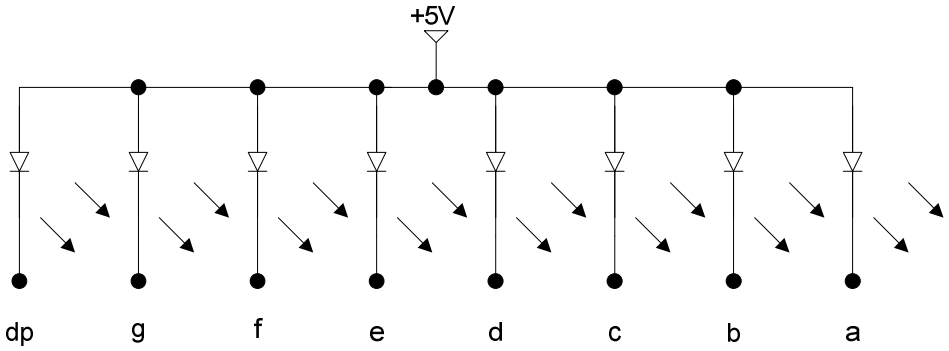
Gb.1. Konfigurasi penamaan masing-masing segment

Gb.2. Bentuk fisik 7-Segment

7-seg terdiri dari 2 jenis atau type yang beredar dipasaran, yaitu Common Anode dan Common Cathode. Common memiliki terjemahan "bersama", artinya salah satu kutup pada 7-seg dijadikan menjadi satu, atau dapat dikatakan satu kaki 7-seg dipakai bersama dengan jenis kutup yang sejenis. Pengetahuan akan common pada setiap penggunaan 7-seg sangatlah penting, dikarena berkaitan dengan cara untuk menghidupkannya apakah active high atau active low. Secara skematik dua jenis tersebut dapat digambarkan sebagai berikut;



Gb.3. Skematik 7-seg Common Cathode



Gb.4. Skematik 7-seg Common Anode

Secara program untuk menghidupkan 7-seg seperti halnya menghidupkan 8 buah LED. Pengaturan Port sebagai keluaran dengan nilai keluaran sesuai dengan common 7-seg yang dipakai. Berikut table daftar data kelauran untuk menghidupkan 7-seg;

Common Anode (active Low)			Common Cathode (active High)		
Angka	**Hexa**	**Biner**	**Angka**	**Hexa**	**Biner**
0	0xC0	0b11000000	0	0x3F	0b00111111
1	0xF9	0b11111001	1	0x06	0b00000110
2	0xA4	0b10100100	2	0x5B	0b01011011
3	0xB0	0b10110000	3	0x4F	0b01001111
4	0x99	0b10011001	4	0x66	0b01100110
5	0x92	0b10010010	5	0x6D	0b01101101
6	0x83	0b10000011	6	0x7D	0b01111101
7	0xF8	0b11111000	7	0x07	0b00000111
8	0x80	0b10000000	8	0x7F	0b01111111
9	0x98	0b10011000	9	0x6F	0b01101111

Peng-aksesan 7-seg dapat dilakukan dengan data hexa atau biner seperti pada table diatas. Untuk menghidupkan 7-seg common anode maka dibutuhkan sinyal keluaran rendah (active low), sedangkan untuk menghidupkan common cathode dibutuhkan sinyal keluaran tinggi (active high). Pada table diatas segment dp (dot point) tidak diaktifkan. Segment ini dipakai untuk fungsi bilangan-bilangan tertentu, seperti penada ribuan,pecahan, decimal dan masih banyak lainnya.

Pemasangan 7-seg untuk menampilkan suatu informasi data biasanya dirangkai lebih dari satu. Seperti untuk menampilakan bilangan puluhan, ratusan, ribuan dan seterusnya. Pada prinsip pengiriman data hampir sama dengan yang satu 7-seg, akan tetapi untuk menghidupkan dua tau lebih 7-seg dengan karakter yang berbeda maka dibutuhkan teknik penyalaan yang bergantian. Pemilihan nyala 7-seg diikuti dengan data yang ingin ditampilkan secara serentak, hal ini dapat dilihat pada ilustrasi berikut ini;

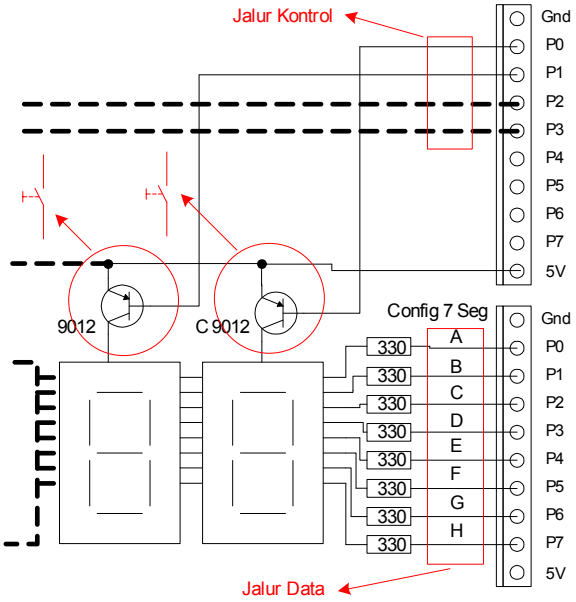
Contoh: Algoritma menampilkan 2 digit angka yaitu 26 dengan 7-seg common Anode

Label ulang:
Hidupkan 7-seg satuan dan matikan 7-seg puluhan
Kirim data biner angka 6 (0b10010010)
Tunda 1 mili detik
Hidupkan 7-seg puluhan dan matikan 7-seg satuan
Kirim data biner angka 2 (0b10100100)
Tunda 1 mili detik

Kembali ke label ulang

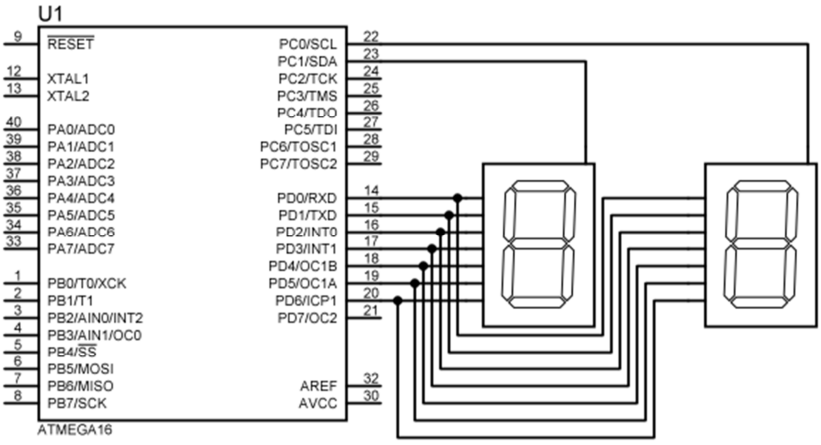
Sebenarnya untuk menampilkan pada 2 7-seg atau lebih dengan tampilan data yang berbeda antara 7-seg satu dengan yang lain, dihidupkan secara bergantian dan bersamaan data yang akan ditampilkan. Penampilan data dilakukan dengan kecepatan tinggi dalam orde mili detik, sehingga mata akan terkelabuhi yang terlihat bahwa tampilan 7-seg 2 digit atau lebih nyala bersamaan. Pada teknik penyalaan 7-seg dua digit atau lebih dikenal dua istilah, yaitu jalur data (PORT data) dan jalur control (PORT control). Jalur data merupakan jalur dimana data-data biner/hexa dikirim untuk menampilkan karakter pada 7-seg. Sedangkan jalur control merupakan kendali untuk memilih 7-seg mana yang akan dinyalakan sesuai dengan data yang ingin ditampilkan.

Pada jalur control bisa terhubung langsung dari mikrokontroller ke 7-seg atau melalui transistor, keduanya berfungsi seperti saklar yang digunakan untuk memilih 7-seg mana yang akan dihidupkan. Berikut secara ilustrasi daripada jalur control dan jalur data;



Gb.5. Jalur data dan jalur control Multi 7-seg

B. Gambar Rangkaian Simulasi



Gb. Skematik Simulasi 7-Segment

Tabel 1. Keyword Komponen

Nama Komponen	Keyword ISIS
Mikrokontroler ATMega16	ATMEGA16
7-Segment Com Anode	7SEG-COM-AN-BLUE

PART1: 7-SEG Tunggal

C. Contoh program

C.1. Program menampilkan angka 5 pada 1 buah 7 segment (common Anode)

```
#include <mega16.h>
void main()
{
……
while(1)
        {
        PORTD=0b10010010;
        };
}
```

C.2. Program menampilkan angka 0 sampai dengan 9 pada 1 buah 7 segment (common Anode)

```
#include <mega16.h>
#include <delay.h>
void main()
{
…..
while(1)
        {
        PORTD=0b11111111;
        PORTD=0b11000000;
        delay_ms(500);
        PORTD=0b11111001;
        delay_ms(500);
        PORTD=0b10100100;
        delay_ms(500);
        PORTD=0b10110000;
        delay_ms(500);
        PORTD=0b10011001;
        delay_ms(500);
        PORTD=0b10010010;
        delay_ms(500);
        PORTD=0b10000011;
        delay_ms(500);
        PORTD=0b11111000;
        delay_ms(500);
        PORTD=0b10000000;
        delay_ms(500);
        PORTD=0b10011000;
        delay_ms(500);
        };
}
```

D. Latihan Mandiri

D.1. Buatlah program untuk menampilkan Counter down dengan satu buah 7-seg common anode!

D.2 Buatlah program untuk menampilkan Couter up 0-9, kemudian menjadi counter down 9-0 secara otomatis dengan menggunakan 1 buah 7-seg common anode!

D.3 Aplikatif

Terdapat dua buah saklar yang berfungsi sebagai berikut:

Sw1 ditekan maka 7-seg menampilkan couter up 0-9

Sw2 ditekan maka 7-seg menampilkan couter down 9-0

Buatlah programnya menggunakan 7-seg common anode!

PART2: 7-SEG dua buah atau lebih

E. Contoh program

E.1. Program menampilkan angka 25 pada 2 buah 7 segment (common Anode)

```
#include <mega16.h>
#include <delay.h>
void main()
{
....
while(1)
        {
        PORTD=0b10010010;
        PORTC.0=1;PORTC.1=0;
        delay_ms(1);
        PORTD=0b10100100;
        PORTC.0=0;PORTC.1=1;
        delay_ms(1)
        };
}
```

E.2. Program Counter up 0-99 dengan library (array dan perulangan) – (common Anode)

```
#include <mega16.h>
#include <delay.h>
unsigned char satuan,puluhan,ulang;
unsigned char data[10]={0xC0,0xF9,0xA4,0xB0,0x99,0x92,0x83,0xF8,0x80,0x98};

void main ()
{
……
while(1)
        {
        for(puluhan=0;puluhan<=9;puluhan++)
                {
                for(satuan=0;satuan<=9;satuan++)
                        {
                        for(ulang=0;ulang<=250;ulang++)
                                {
                                PORTD=data[satuan];
                                PORTC.0=1;PORTC.1=0;
                                delay_ms(1);
                                PORTD=data[puluhan];
```

```
                                    PORTC.0=0;PORTC.1=1;
                                    delay_ms(1);
                                    }
                                }
                            }
                        };
    }
```

F. Latihan Mandiri

F.1. Buatlah program counter down 99-0 dengan dua buah 7-seg common anode!

F.2. Buatlah program couter up 0-99, kemudian otomatis menjadi couter down 99-0 dengan 2 buah 7-seg common anode!

F.3. Aplikatif

Terdapat sebuah mesin dengan 1 buah motor dan 2buah 7-seg sebagai indikatornya, serta 2 buah saklar push button.

- Apabila sw1 ditekan maka motor akan on, bersamaan dengan itu 7-seg akan couter up sampai 25 dan berhenti. Pada waktu counter up berhenti maka motor juga ikut mati.
- Apabila sw2 ditekan maka motor akan on, bersamaan dengan itu 7-seg akan couter down dari 50-0 dan berhenti. Pada waktu counter down selesai maka motor juga ikut mati.

Buatlah programnya dengan jeda waktu penambahan/pengurangan counter mendekati 1detik, gunakan 2 buah 7-seg common anode!

Praktik Mikrokontroler

Topik: LCD Mikrokontroler (16x2) Dan ADC

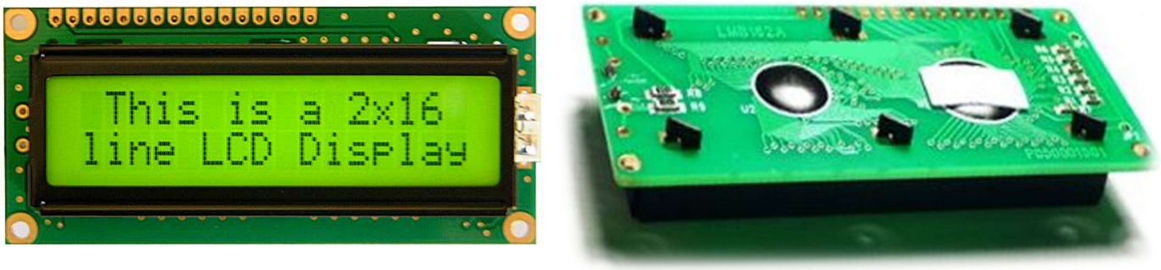
A. Kajian Teori

LCD 16 x2 (Liquid Cristal Display)

LCD *(Liquid Cristal Display)* merupakan teknologi yang digunakan untuk menampilkan suatu poin (titik/dot) dalam jumlah lebih dari satu sehingga membentuk suatu karakter. Teknologi ini tergolong baru, dengan menggantikan CRT *(Cathode Ray Tube)* sebagai pendahulu untuk menampilkan data/informasi. Penggunaan LCD saat ini telah berkembang cepat, dikarenakan banyak faktor keuntungan yang didapatkan. Antara lain penggunaan LCD yang utama sebagai hemat energy tau arus listrik untuk mengoperasikan cukup kecil. Selain itu bentuk fisiknya tipis, kecil, serta dengan berat yang ringan.

Perkembangan teknologi LCD tidak berhenti, saat ini telah muncul LED sebagai penyusun tampilan poin (titik/dot) yang lebih hemat, dan dari segi tampilan lebih tajam. LCD yang digunakan untuk menampilkan data dari mikrokontroler menggunakan jenis 16x2 (16 kolom, 2 baris), sedangkan teknologi LED diberi nama dengan OLED 16 x2. Pada dasarnya prinsip untuk akses menampilkan data LCD dan OLED sama, karena instruksi-instruksi sudah ada pada library CVAVR.

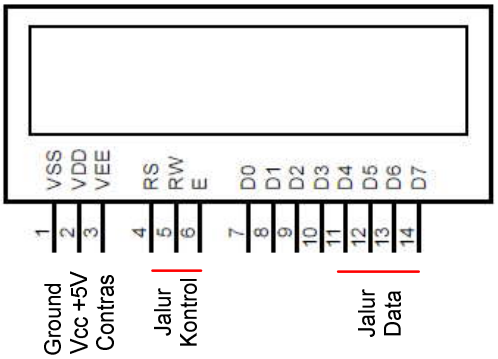
Berikut bentuk secara fisik LCD dan OLED;



Gb.1 LCD 16x2 Character

Gb.2 OLED 16x2 Character

Tegangan kerja dari LCD dan OLED +5VDC, dengan konsumsi arus yang kecil. Tegangan pada LCD terdapat dua bagian, 1 bagian untuk kerja Rangkaian LCD (Pin 1 dan 2) dan 1 bagian lainnya untuk tegangan cahaya latar *(back Light)* (PIN 15 dan 16). Jalur data untuk meng-akses karakter terdiri dari 4bit MSB yang berada pada kaki 11-14, sedangkan 4bit LSB (kaki 7-10) tidak dihubungkan dengan mikrokontroler. Berikut ilustrasi konfigurasi fungsi masing-masing pin;



Gb. 3 Fungsi setiap PIN pada LCD 16x2

Penulisan program untuk mengirim atau menampilkan data pada LCD telah di-library-kan pada CVAVR, sehingga cukup menggunakan instruksi-instruksi yang sudah disediakan data dapat ditampilkan. Berikut beberapa instruksi yang disediakan oleh CVAVR;

lcd_clear(); → Menghapus LCD

lcd_gotoxy(x,y); → Meletakkan posisi dalam memulai menampilkan karakter
x → sebagai posisi kolom (integer)
y → sebagai posisi baris (integer)

lcd_putchar(x); → Menampilkan sebuah karakter dengan meng-akses library karakter pada LCD
x→ bilangan decimal/Hexadecimal

lcd_putsf(x); → Menampilkan char/string yang tersimpan pada flash
x → data char/string (kata/kalimat) contoh: lcd_putsf("Hallo Word");

lcd_puts(x); → Menampilkan char/string yang tersimpan pada RAM
x → data char/string (kata/kalimat) contoh: lcd_pustf(data);

Selain beberapa instruksi diatas, terdapat instruksi khusus untuk penyimpanan sementara pada RAM data yang akan ditampilkan LCD. Instruksi ini harus diikuti dengan penyertaan file #include <stdio.h>, serta untuk menampilkan memakai lcd_puts(…);. Berikut instruksinya;

sprintf(array_penyimpanan,"operator",data_asli);

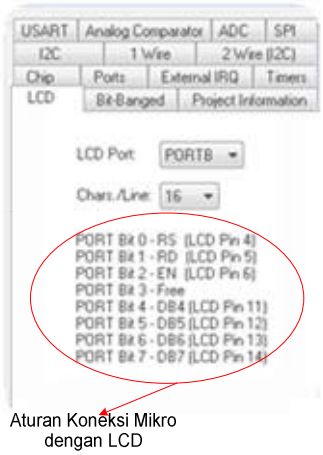
contoh:

```
……
#include <stdio.h>
…..
unsigned char lcd_buffer[30]; //menyiapkan variable penyimpan data pada RAM
…..
…..
sprintf(lcd_buffer,"%d",data);
lcd_puts(lcd_buffer);
……
```

LCD dari segi rangkaian elektronik juga memiliki IC controller yang digunakan untuk meng-akses langsung pada LCDnya. Jadi pada dasarnya komunikasi mikrokontroler dengan LCD hanya sekedar meng-akses karakter-karakter yang tersimpan didalam library hard ware LCD. Berikut secara pengalamatan karakter yang ada pada LCD;

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50
							!	"	#	$	%	&		(	)	*	+	,	-	.	/	0	1	2
51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75
3	4	5	6	7	8	9	:	;	<	=	>	?	@	A	B	C	D	E	F	G	H	I	J	K
76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100
L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	[	¥	]	^	_	`	a	b	c	d
101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120	121	122	123	124	125
e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	{	|	}
126	127	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	144	145	146	147	148	149	150
→	←																							
151	152	153	154	155	156	157	158	159	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175
										｡	｢	｣												
176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191	192	193	194	195	196	197	198	199	200
201	202	203	204	205	206	207	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223	224	225
226	227	228	229	230	231	232	233	234	235	236	237	238	239	240	241	242	243	244	245	246	247	248	249	250
251	252	253	254	255																				

Guna memulai memprogram LCD menggunakan CVAVR, pengaturan PORT yang terhubung ke LCD dapat diatur pada CodeWizardAVR pada seluruh PORT. Pengaturan dapat dilakukan dengan secara visual (CodeWizardAVR) serta secara penulisan program. Berikut ilustrasi pengaturan PORT yang terhubung dengan LCD;




```
#asm
  .equ __lcd_port=0x18 ;PORTB
#endasm

#include <lcd.h>
```

Gb.4 Pengaturan PORT mikrokontroler yang terhubung dengan LCD

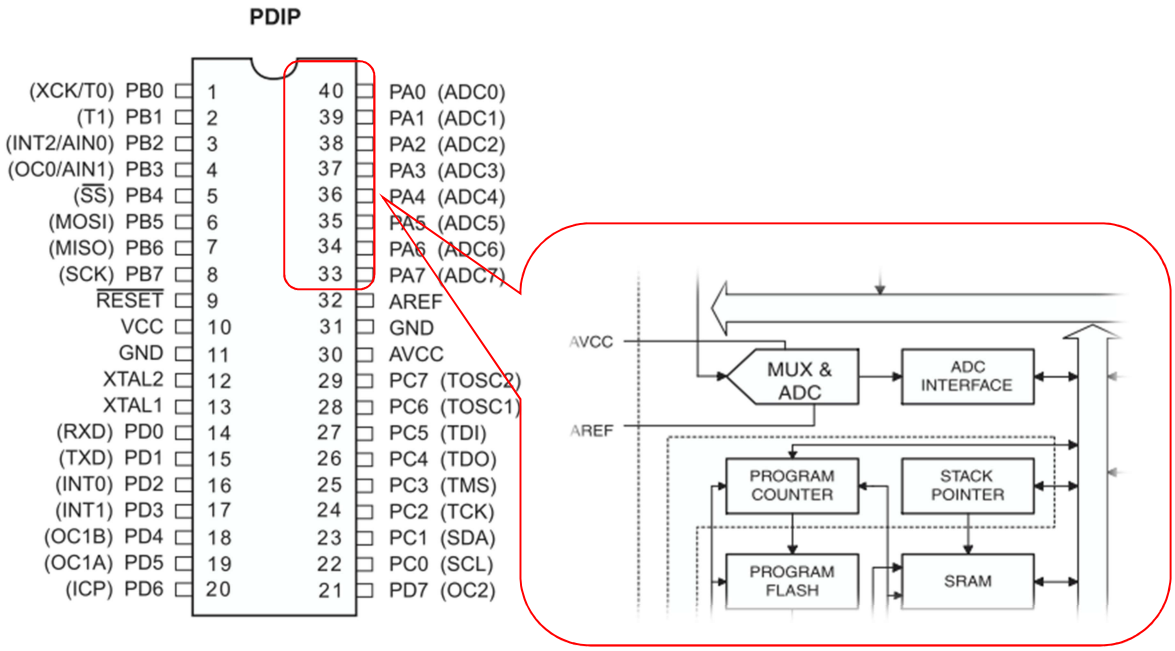
a. Secara CodeWizard
b. Secara tertulis

Penghubungan PORT mikro ke LCD sudah memiliki aturan seperti yang telah di cantumkan dalam pengaturan menggunakan CodeWizard. Pemasangan ini telah disesuaikan dengan aturan komunikasi yang dibuat pada library CVAVR. Penghubungan dapat dilihat pada sub. B Gambar rangkaian Hardware selanjutnya.

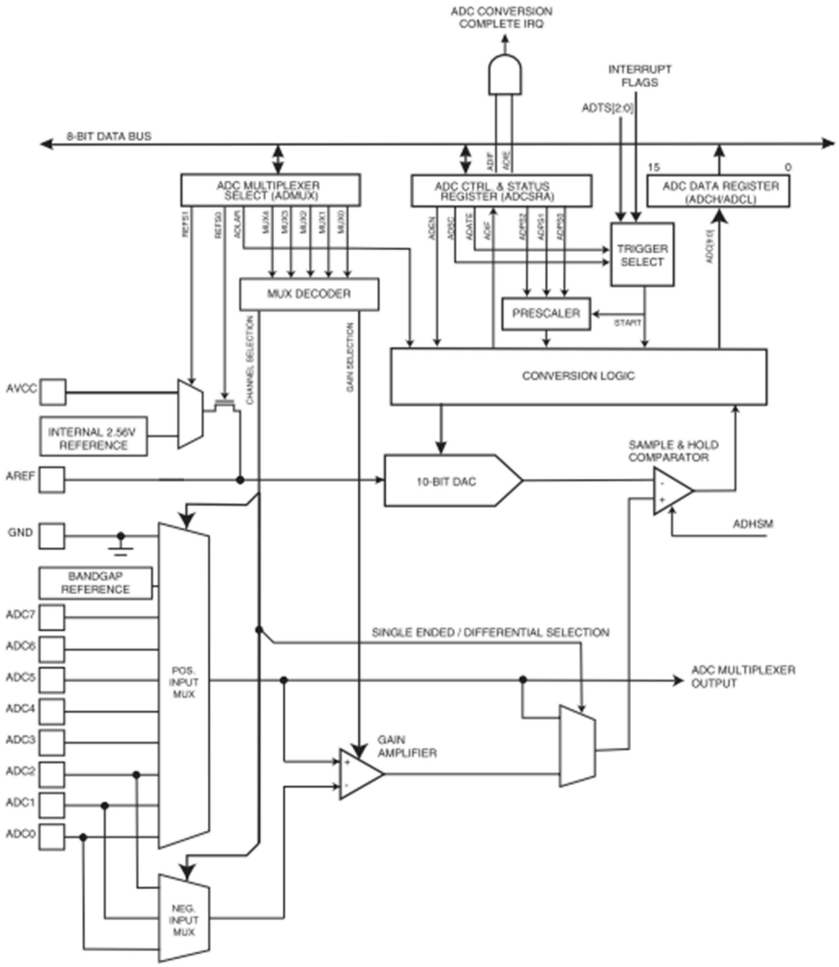
ADC (Analog to Digital Converter)

ADC (Analog to Digital Converter) merupakan sebuah system yang berupa rangkaian elektronik dengan fungsi untuk mengubah sinyal/tegangan analog menjadi sinyal atau data-data digital. Pengubahan ini bertujuan untuk mendapatkan data-data digital berupa hexa atau biner, sehingga mikroprosesor dapat mengolah data tersebut. Data data digital hasil pengubahan ADC merupakan representasi dari masukan yang berupa data tegangan analog.

ADC dalam pembahasan kali ini focus pada ADC yang dimiliki mikrokontroller keluarga AVR. ADC mikrokontroller keluarga AVR yang dimiliki merupakan ADC 8bit dan 10bit. Dengan tegangan referensi yang dapat diatur oleh keinginan programmer. Setiap tipe mikrokontroller AVR dengan seri ATMega xxxx memiliki fasilitas ADC yang dapat programmer digunakan. Setiap tipe memiliki jumlah ADC yang berbeda (lihat pada data sheet), akan tetapi memiliki resolusi yang sama yaitu 8bit. Berikut ilustrasi dari ADC mikro yang ada didalam IC menjadi satu dengan system;

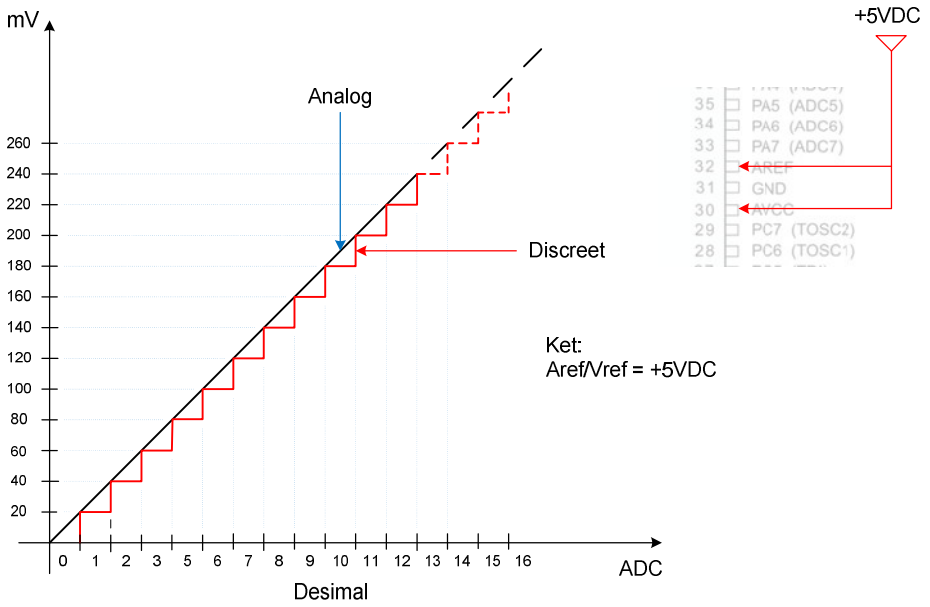


Gb.5 Fasilitas konfigurasi ADC pada mikrokontroler ATMega16



Gb.6 Blok Konfigurasi internal ADC mikrokontroler

Berdasarkan gambar diatas terdapat terminal yang penting dan harus diperhatikan dalam menggunakan fasilitas ADC. Terminal/PIN/Kaki IC tersebut diberikan nama AVCC dan ARef. AVCC merupakan tegangan yang digunakan untuk kerja rangkaian ADC yang ada didalam mikrokontroler. Pin tersebut agar dapat bekerja secara makasimal diberikan tegangan +5VDC. Sedangkan pin ARef merupakan tegangan referensi yang digunakan sebagi tegangan pembanding dan acuan ADC mikro dalam mengkonversi tegangan analog menjadi digital. Tegangan Aref dapat disesuikan dengan kebutuhan akan kerapatan data dalam pengkonversiannya. Semakin kecil tegangan referensi maka resolusi pembacaan ADC semakin rapat. Berikut ilustrasi pengubahan Analog ke Digital;



Gb.7 Grafik ilustrasi pengubahan tegangan analog menjadi data digital

Penjelasan gambar diatas terdiri dari dua bentuk sinyal, yaitu sinyal analog dengan sinyal discreet yang nanti dijadikan data hasil konversi digital. Pengubahan tegangan masukan analog menjadi digital memperhatikan beberapa variabel, seperti V/Aref yang menetukan kerapatan resolusi (tinggi step discreet) dan tipe kemampuan pengolahan ADC yang digunakan. Tipe ADC pada keluarga AVR mikrokontroler seperti diatas telah dijelaskan yaitu ADC 8bit. Guna menghitung kerapatan pengubahan sinyal discrit perlu memperhatikan dua factor tersebut, misalkan terdapat beberapa contoh sebagai berikut:

- ADC 8bit ATMega 16 menggunakan tegangan referensi (ARef) sebesar +5VDC, berapa resolusinya?

  Jawaban dari pertanyaan seperti itu adalah:

$$= \frac{}{(2\ )\quad 1}$$
$$= \frac{5}{255}$$
$$= 20$$

  Sehingga besarnya step discreet terjadi perubahan pembacaan data digital setiap kelipatan 20mV, seperti pada gambar diatas. Setiap kenaikan tegangan masukan 20mV akan mengubah data digital satu tingkat lebih tinggi.

- Apabila terdapat tegangan masukan ADC sebesar 3,4V maka berapa pembacaan data ADC mikrokontroler?

  Jawaban dari pertanyaannya adalah

$$= \frac{}{}$$
$$= \frac{3{,}4}{20}$$
$$= 170$$

  Data ADC sebesar 170 merupakan pembacaan mikrokontroler terhadap masukan tegangan analog sebesar 3,4V.

Tegangan referensi (ARef) sebesar +5V DC digunakan apabila untuk pembacaan tegangan masukan ADC dengan range dan jangkauan yang tinggi. Sedangkan apabila dibutuhkan resolusi pembacaan ADC yang lebih kecil, dapat dilakukan perubahan tegangan referensi yang lebih kecil. Sebagai contoh berikut ini:

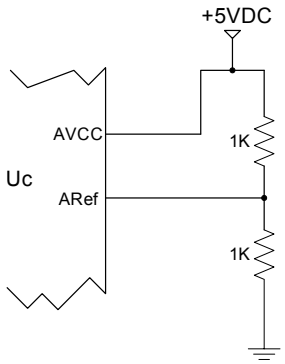
- Jika terdapat sensor suhu LM35 dengan perubahan tegangan 10mV/°C, maka agar suhu dapat dibaca akurat perlu diberikan tegangan referensi sebagai berikut;

  = ((2) 1)

  = 10 255

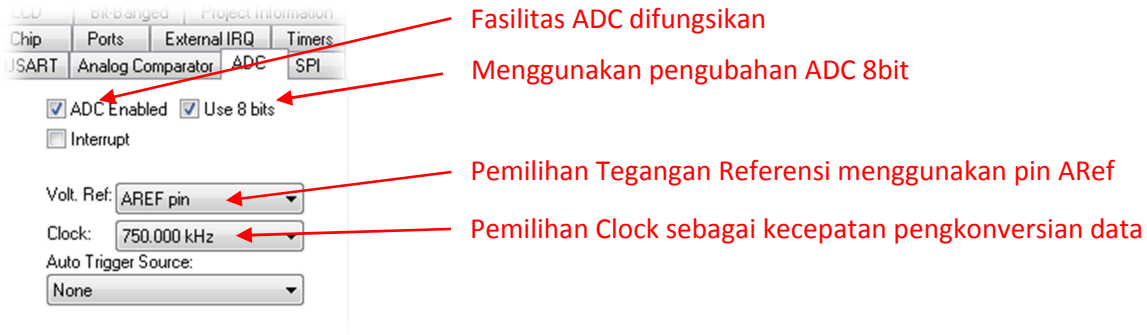
  = 2,5

  Sehingga agar dapat membaca perubahan suhu /°C maka perlu dipasang tegangan referensi sebesar 2,5V. Berikut skematik yang dapat diterapkan agar mendapatkan tegangan referensi 2,5V:



Gb.8 Skematik Aref dengan tegangan 2,5V

Secara rangkaian elektronik dan pengubahan ADC telah dijelaskan diatas, sedangkan dalam pemrograman perlu dilakukan pengaturan pada CodeWizard AVR sebagai berikut;



Gb.9 Konfigurasi CodeWizard AVR dalam mengaktifkan penggunaan Fasilitas ADC

Setelah pengaturan diatas selesai dan masuk pada area program (Generate, Save and exit) maka akan muncul sebuah blok fungsi yang sudah dibuatkan CVAVR sebagai fungsi pembacaan/pengkonversian ADC. Berikut listing program yang dapat dituliskan;

```
.....
#define ADC_VREF_TYPE 0x20
// Read the 8 most significant bits
// of the AD conversion result
unsigned char read_adc(unsigned char adc_input)
{
```

```
ADMUX=adc_input | (ADC_VREF_TYPE & 0xff);
// Delay needed for the stabilization of the ADC input voltage
delay_us(10);
// Start the AD conversion
ADCSRA|=0x40;
// Wait for the AD conversion to complete
while ((ADCSRA & 0x10)==0);
ADCSRA|=0x10;
return ADCH;
}
……..
```

Sedangkan pada CVAVR sendiri sudah terdapat instruksi/statement yang dapat digunakan untuk meng-akses pembacaan tegangan analog, yaitu;

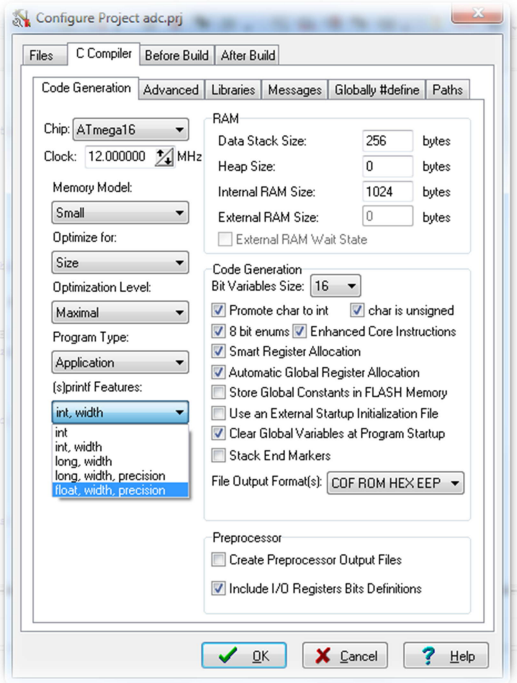
read_adc(no ADC)

contoh: (missal terdapat tegangan masukan yang akan dibaca pada ADC0)

adc=read_adc(0);

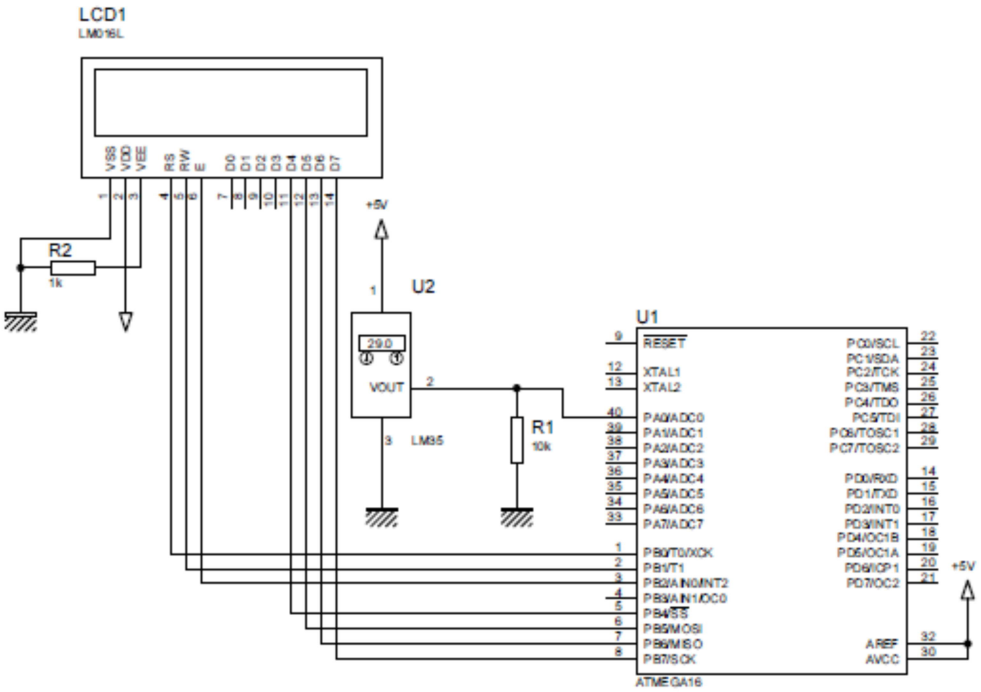
Keterangan program diatas adalah "baca tegangan pada ADC0 dan hasilnya dimasukan dalam variable adc".

Penggunaan fasilitas ADC pada mikrokontroler harus dibarengi dengan penggunaan suatu tampilan yang dipakai sebagai penampil data. Tampilan data tersebut bisa berupa LED, 7Seg, atau sebuah LCD. Khusus untuk penggunaan LCD yang digunakan menampilkan data pecahan dari hasil pengolahan suhu, serta menggunakan instruksi program sprintf(…), perlu ada pengaturan yang khusus sebelum program itu di build. Pengaturan itu dapat dilakuka dengan cara: Menu Project → Configure →C Compiler →kemudian pada sprintf Feature dipilih float width, precision.



Gb.10 Pengaturan sprintf feature

B. Gambar Rangkaian Simulasi



Gb. Connection LCD 16x2 dan Fungsi ADC

Tabel 1. Keyword Komponen

Nama Komponen	Keyword ISIS
Mikrokontroler ATMega16	ATMEGA16
Resistor	RES
LCD	LM016L
Sensor Suhu LM35	LM35

C. Contoh program

C1. Program menampilkan Kalimat "Suhu=28°C"

```
#include <mega16.h>
…….
#include <lcd.h>
…….
void main (void)
{
…..
while(1)
    {
    lcd_gotoxy(5,0);
    lcd_putsf("Suhu=28");
    lcd_putchar(223);
    lcd_gotoxy(14,0);
    lcd_putsf("C");
```

```
    };
}
```

C2. Program menampilkan Kalimat "Hallo Word" berkkedip

```
#include <mega16.h>
#include <delay.h>
…….
#include <lcd.h>
…….
void main (void)
{
…..
while(1)
    {
    lcd_gotoxy(5,0);
    lcd_putsf("Hallo Word");
    delay_ms(500);
    lcd_clear();
    delay_ms(100);
    };
}
```

C3. Program Menampilkan Kalimat "Hallo Word" berganti posisi

```
#include <mega16.h>
#include <delay.h>
…….
#include <lcd.h>
…….
void main (void)
{
…..
while(1)
    {
    lcd_clear();
    lcd_gotoxy(3,0);
    lcd_putsf("Hallo Word");
    delay_ms(500);
    lcd_clear();
    lcd_gotoxy(0,1);
    lcd_putsf("Hallo Word");
    delay_ms(500);
    lcd_clear();
    lcd_gotoxy(6,0);
    lcd_putsf("Hallo Word");
    delay_ms(500);
    };
}
```

C4. Program menampilkan Counter Up 0-100 pada LCD

```
#include <mega16.h>
#include <delay.h>
#include <stdio.h>
…….
#include <lcd.h>
…….
unsigned char lcd_buffer[30];
unsigned char x;

void main (void)
{
…..
while(1)
    {
    for(x=0;x<=100;x++)
            {
            lcd_clear();
            lcd_gotoxy(3,0);
            lcd_putsf("Counter Up");
            sprintf(lcd_buffer,"%d",x);
            lcd_gotoxy(8,1);
            lcd_puts(lcd_buffer);
            delay_ms(500);
            }
    };
}
```

C5. Program menampilkan data pengubahan data analog menjadi data discreet menggunakan sensor lm35 sebagai masukan dan LCD sebagai tampilan keluaran.

```
#include <mega16.h>
#include <delay.h>
#include <stdio.h>
……

unsigned char read_adc(unsigned char adc_input)
{
……..
……..
}
……..
unsigned char data_discreet;
unsigned char lcd_buffer[30];
void main (void)
{
……
……
```

```
while(1)
        {
        data_discreet=read_adc(0);
        sprintf(lcd_buffer,"Discreet = %u",data_discreet);
        lcd_clear();
        lcd_gotoxy(0,0);
        lcd_puts(lcd_buffer);
        delay_ms(500);
        };
}
```

C6. Program menampilkan suhu ruangan dengan menggunakan LM35 sebagai sensornya, dan LCD sebagai keluaran data. *(catatan: keluaran suhu kelipatan 2)*

```
#include <mega16.h>
#include <delay.h>
#include <stdio.h>
.....

unsigned char read_adc(unsigned char adc_input)
{
.....
.....
}
.....
unsigned int data_suhu;
unsigned char lcd_buffer[30];
void main (void)
{
.....
.....
while(1)
        {
        data_suhu=(read_adc(0)*2);
        sprintf(lcd_buffer,"Suhu= %d",data_suhu);
        lcd_clear();
        lcd_gotoxy(0,0);
        lcd_puts(lcd_buffer);
        lcd_gotoxy(9,0);
        lcd_putchar(223);
        lcd_putsf("C");
        delay_ms(500);
        };
}
```

D. Latihan Mandiri

D1. Buatlah program menampilkan counter down berupa angka dari 200-0 pada LCD!

D2. Buatlah program counter up (0-100), kemudian secara otomatis menjadi counter down (100-0) dengan menggunakan LCD!

D3. Buatlah tampilan Text Berjalan menggunakan LCD!
D4. Aplikatif
Terdapat sebuah mesin dengan menggunakan tampilan LCD 16x2 untuk menunjukan kondisi proses yang sedang berlangsung, berikut kerjanya;

- Saat start tampilan waktu mendekati 60 detik berjalan dengan counter down, setelah sampai 0 tampilan LCD akan menampilkan tulisan "Motor 1 ON" selama mendekati 10detik
- Setelah tampilan "Motor 1 ON" selama mendekati 10 detik, tampilan akan menampilkan waktu mendekati 30 detik dengan couter down, setelah sampai 0 tampilan LCD menampilkan ttulisan "Motor 2 ON" selama mendekati 10 detik
- Setelah tampilan "Motor 2 ON" selama mendekati 10 detik, tampilan akan menampilkan waktu mendekati 90 detik dengan couter up, setelah sampai 90 tampilan LCD menampilkan ttulisan "Motor 1&2 OFF" setelah itu berhenti.

Buatlah programnya dengan alur kerja seperti diatas!

D5. Buatlah program untuk menampilkan suhu dari sensor LM 35 dengan kelipatan suhu 1°C dengan menggunakan LCD sebagai tampilannya!
D6. Buatlah program tampilan suhu LM35 dengan perubahan satuan suhu pada satu tampilan LCD yaitu °C, °F, °K dan R!
D7. Aplikatif
Terdapat sensor LM35 pada sebuah mesin penetas telur sebagai elemen kendali suhu ruangan penetasan. Kerja alat kendali sebagai berikut:

- Apabila suhu menunjukan pada lcd dibawah 23°C, maka pada lcd juga akan menampilkan "suhu rendah"
- Apabila suhu menunjukkan pada lcd 23-38°C, maka pada lcd juga akan menampilkan "suhu normal")
- Apabila suhu menunjukan pada lcd diatas 38°C, maka pada lcd juga akan menampilkan "suhu tinggi"

Buatlah programnya!

Praktik Mikrokontroler

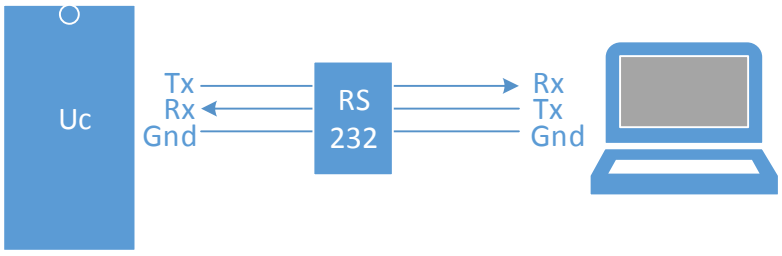
Topik: Komunikasi Serial

C. Kajian Teori

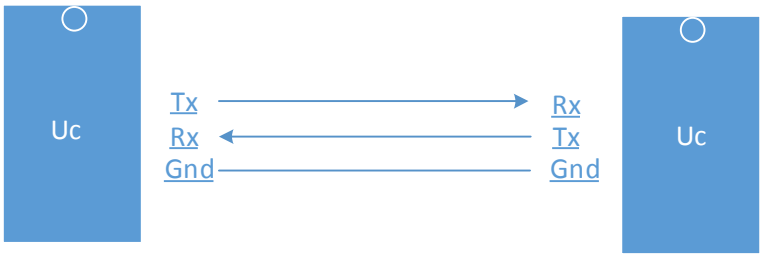
Perkembangan komunikasi di masa sekarang sangat cepat. Dimulai dari teknik komunikasi secara parallel samapai pengembangan teknik komunikasi serial yang dilakukan pengembangan sangat cepat. Awalnya kita mengenal port parallel (DB25) sebagai piranti komunikasi computer dengan printer dan berbagai alat lainnya. Akan tetapi dilihat dari pengembangan kedepannya, teknik ini mengalami kesulitan dan kendala, hal ini dipandang dari sudut ekonomisnya. Dengan jumlah jalur komunikasi yang banyak, menjadikan komunikasi parallel ini mulai ditinggalkan dan beralih ke teknik komunikasi serial.

Komunikasi serial yang dimulai dari port serial (DB9) sampai dengan saat ini dikenal dengan teknologi USB, SATA dan Wi-fi menjadikan komunikasi serial sebagai teknik komunikasi yang mengalami perkembangan sangat cepat. Dipandang cukup ekonomis hanya membutuhkan dua jalur komunikasi yaitu transmit (Tx) dan receive (Rx), dan jika menggunakan piranti kabel maka cukup membutuhkan minimal 3 kabel yaitu transmit, receive dan ground. Berikut ilustrasi dari komunikasi serial menggunakan piranti kabel sebagai jalur komunikasinya;

Gambar 1. Komunikasi serial Tak-Sinkron PC dengan Laptop



Gambar 2. Komunikasi serial Tak-Sinkron Mikrokontroler dengan Laptop melalui antar muka RS232



Gambar 3. Komunikasi serial Mikrokontroler dengan Mikrokontroler (*TTL to TTL*)

Komunikasi serial antar computer (PC/Laptop) yang memiliki fasilitas serial port (DB9) secara langsung dapat dikomunikasikan, tanpa menggunakan antarmuka (interface). Hal ini dikarenakan pada sebuah PC/Laptop yang memiliki port serial sudah dilengkapi RS232 pada masing-masing hardwarenya. Sedangkan agar terjadinya komunikasi serial antara mikrokontroler dengan PC/laptop, maka dibutuhkan rangkaian tambahan sebagai antarmuka (interface) yang diberi nama rangkaian RS232. Untuk komunikasi antar IC kita dapat menggunakan komunikasi serial, seperti gambar 3. Komunikasi serial antar IC dapat dilakukan secara langsung dengan jenis bahan penyusun IC yang sama (*TTL to TTL*).

Prinsip komunikasi serial adalah pengiriman data secara serial dengan menggunakan karakter-karakter didalam ASCII. Karakter ini yang nantinya akan dirubah menjadi signal digital oleh hardware transmitter (Tx), dan akan dterjemahkan lagi menjadi data karakter oleh hardware receiver (Rx).

Komunikasi serial mikrokontroler sangatlah sederhana, dikarenakan sudah memiliki instruksi-instruksi pemrograman yang standart. Berikut instruksi yang sering digunakan dalam komunikasi serial mikrokontroler;

printf("……");

- ➔ digunakan untuk mencetak atau mengeluarkan data string (kata/kalimat) ke jalur komunikasi serial mikrokontroler

putchar('…'); atau *putchar(no_char);*

- ➔ digunakan untuk mencetak atau mengeluarkan data char (karakter) ke jaluar komunikasi serial mikrokontroler

scanf(&varibel_penyimpan);

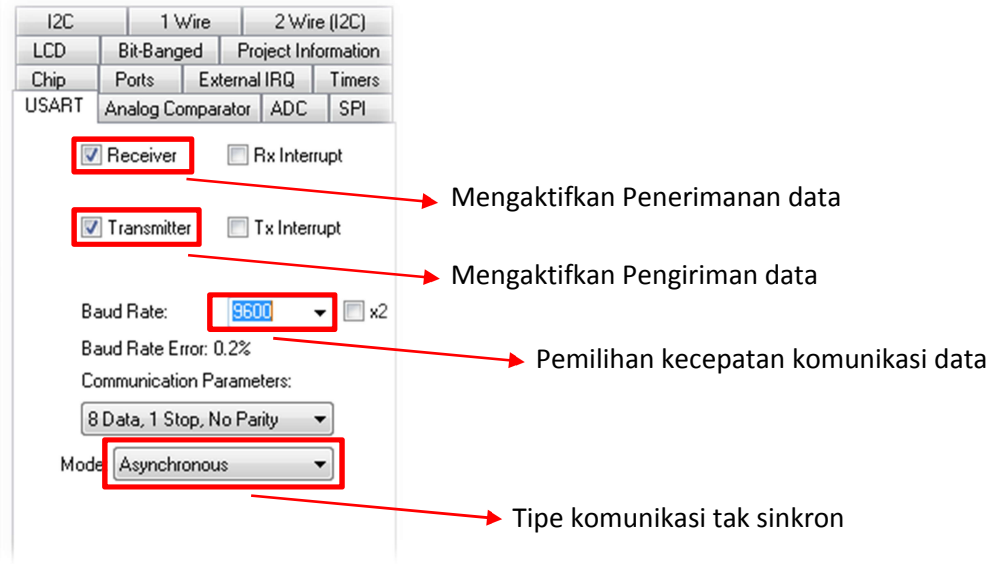
- ➔ digunakan untuk membaca/menerima data string/char dari jalur komunikasi serial mikrokontroler

getchar();

- ➔ digunakan untuk membaca/menerima data char (karakter) dari jalur komunikasi serial mikrokontroler

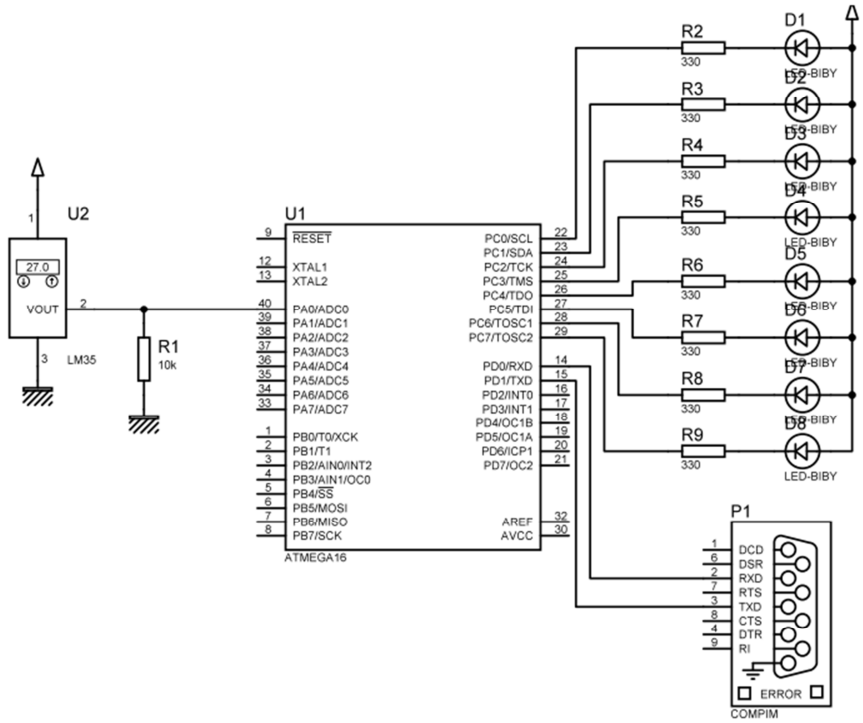
Untuk membantu membaca karakter yang dikirim, biasanya digunakan fasilitas software HyperTerminal atau fasilitas terminal pada software CVAVR. Software tersebut dapat menerima dan mengirim karakter/string dari mikrokontroler.

Pengaturan mikrokontroler untuk memfungsikan komunikasi serial dapat dilihat seperti gambar berikut;



Gambar 4. USART (kom. Serial) setting

D. Gambar Rangkaian Simulasi



Gambar. Rangkaian simulasi Komunikasi serial

Tabel 1. Keyword Komponen

Nama Komponen	Keyword ISIS
Mikrokontroler ATMega16	ATMEGA16
Resistor	RES
LCD	LM016L

Sensor Suhu LM35	LM35
Led	LED-BIBY
Port Serial DB9	COMPIM

E. Contoh program

C.1 Program menampilkan/mengirim kalimat "Hallo Word" pada HyperTerminal

```
#include <mega16.h>
#include <stdio.h>
#include <delay.h>
…….
void main()
{
……
while(1)
        {
        printf("Hallo Word");
        delay_ms(1000);
        };
}
```

C.2 Program Menampilkan dan menerima Char (karakter) yang ditekan pada keybord PC/Laptop pada HyperTerminal

```
#include <mega16.h>
#include <stdio.h>
#include <delay.h>
…..
void main()
{
…….
while(1)
        {
        putchar(getchar());
        delay_ms(10);
        };
}
```

C.3 Program Menyalakan LED dengan bantuan kendali angka pada keyboard

```
#include <mega16.h>
#include <stdio.h>
#include <delay.h>
…..
char data;
void main()
{
…….
while(1)
        {
```

```
        data=getchar();
        if(data=='0') PORTC.0=0;          // LED ke-0 ON
        if(data=='1') PORTC.1=0;
        if(data=='2') PORTC.2=0;
        if(data=='3') PORTC.3=0;
        if(data=='4') PORTC.4=0;
        if(data=='5') PORTC.5=0;
        if(data=='6') PORTC.6=0;
        if(data=='7') PORTC.7=0;          // LED ke-7 ON
        if(data=='o') PORTC=0xFF;         //lampu Led Mati Semua
        };
}
```

F. Latihan Mandiri

D.1 Buatlah program komunikasi serial untuk menampilkan suhu dari sensor LM35 ke computer pada software HyperTerminal/Terminal.