MODUL

MODUL AVR ATMEGA 8535

alexandernugroho@gmail.com
SMG
MODUL

ALEXANDER NUGROHO
Jalan Griya Prasetya Selatan V/168, Semarang 50167
Telp. 024-6704602 (085866331024)

PENGENALAN BAHASA C

1

1. PENDAHULUAN

Bahasa C pertama kali digunakan di komputer Digital Equipment Corporation PDP-11 yang menggunakan sistem opersi UNIX C adalah bahasa yang standar, artinya suatu program yang ditulis dengan bahasa C tertentu akan dapat dikonversi dengan bahasa C yang lain dengan sedikit modifikasi. Standar bahasa C yang asli adalah standar dari UNIX. Patokan dari standar UNIX ini diambil dari buku yang ditulis oleh *Brian Kerningan dan Dennis Ritchie* berjudul "The C Programming Language", diterbitkan oleh Prentice-Hall tahun 1978. Deskripsi C dari Kerninghan dan Ritchie ini kemudian kemudian dikenal secara umum sebagai "K dan R C"

2. PENULISAN PROGRAM BAHASA C

Program Bahasa C tidak mengenal aturan penulisan di kolom tertentu, jadi bisa dimulai dari kolom manapun. Namun demikian, untuk mempermudah pembacaan program dan untuk keperluan dokumentasi, sebaiknya penulisan bahasa C diatur sedemikian rupa sehingga mudah dan enak dibaca. Berikut contoh penulisan Program Bahasa C:

```
#include <mega8535.h>
#include <delay.h>
main ()
{
……
……
}
```

Program dalam bahasa C selalu berbentuk fungsi seperti ditunjukkan dalam **main( ).** Program yang dijalankan berada di dalam tubuh program yang dimulai dengan tanda kurung buka **{** dan diakhiri dengan tanda kurung tutup **}**. Semua yang tertulis di dalam tubuh program ini disebut dengan blok. Tanda () digunakan untuk mengapit **argumen** suatu fungsi. Argumen adalah suatu nilai yang akan digunakan dalam fungsi tersebut. Dalam fungsi **main** diatas tidak ada argumen, sehingga tak ada data dalam (). Dalam tubuh fungsi antara tanda **{** dan anda **}** ada sejumlah pernyataan yang merupakan perintah yang harus dikerjakan oleh prosesor. Setiap pernyataan diakhiri dengan tanda titik koma **;** Baris pertama **#include** <**…**> bukanlah pernyataan, sehingga tak diakhiri dengan tanda titik koma (;). Baris tersebut meminta kompiler untuk menyertakan file yang namanya ada di antara tanda <**…**> dalam proses kompilasi. File-file ini (berekstensi.h) berisi deklarasi fungsi ataupun variable.

program sumber dan seandainya tidak ditemukan pencarian akan dilanjutkan File ini disebut **header**. File ini digunakan semacam perpustakaan bagi pernyataan yang ada di tubuh program. *#include* merupakan salah satu jenis pengarah praprosesor *(preprocessor directive)*. Pengarah praprosesor ini dipakai untuk membaca file yang di antaranya berisi deklarasi fungsi dan definisi konstanta. Beberapa file judul disediakan dalam C. File-file ini mempunyai ciri yaitu namanya diakhiri dengan ekstensi **.h.** Misalnya pada program #include <stdio.h> menyatakan pada kompiler agar membaca file bernama *stdio.h* saat pelaksanaan kompilasi.Bentuk umum #include: **#include "namafile"** Bentuk pertama (#include <namafile>) mengisyaratkan bahwa pencarian file dilakukan pada direktori khusus, yaitu direktori file *include*. Sedangkan bentuk kedua (#include "namafile") menyatakan bahwa pencarian file dilakukan pertama kali pada direktori aktif tempat pada direktori lainnya yang sesuai dengan perintah pada sistem operasi.

3. TIPE DATA

Tipe data merupakan bagian program yang paling penting karena tipe data mempengaruhi setiap instruksi yang akan dilaksanakan oleh computer. Misalnya saja 5 dibagi 2 bisa saja menghasilkan hasil yang berbeda tergantung tipe datanya. Jika 5 dan bertipe integer maka akan menghasilkan nilai 2, namun jika keduanya bertipe float maka akan menghasilkan nilai 2.5000000. Pemilihan tipe data yang tepat akan membuat proses operasi data menjadi lebih efisien dan efektif.

Bentuk Tipe data:

No	Tipe Data	Ukuran Range (Jangkauan)
1	Char	1 byte -128 s/d 127
2	Int	2 byte -32768 s/d 32767
3	Unsigned int	2 byte 0 s/d 65535
4	Long Int	4 byte -2147483648 s/d 2147483648
5	Unsigned Long	int 4 byte 0 s/d 4294967296
6	Float	4 byte -3.4E-38 s/d 3.4E+38
7	Double	8 byte 1.7E-308 s/d 1.7E+308
8	Long Double	10 byte 3.4E-4932 s/d 1.1E+4932
9	Char	1 byte -128 s/d 127
10	Unsigned char	1 byte 0 s/d 255

4. KONSTANTA

Konstanta merupakan suatu nilai yang tidak dapat diubah selama proses program berlangsung. Konstanta nilainya selalu tetap. Konstanta harus didefinisikan terlebih dahulu di awal program. Konstanta dapat bernilai integer, pecahan, karakter dan string.
Contoh konstanta : 50; 13; 3.14; 4.50005; ‘A’; ‘Bahasa C’.

5. VARIABLE

Variabel adalah suatu pengenal (identifier) yang digunakan untuk mewakili suatu nilai tertentu di dalam proses program. Berbeda dengan konstanta yang nilainya selalu tetap, nilai dari suatu variable bisa diubah-ubah sesuai kebutuhan. Nama dari suatu variable dapat ditentukan sendiri oleh pemrogram dengan aturan sebagai berikut :

• Terdiri dari gabungan huruf dan angka dengan karakter pertama harus berupa huruf. Bahasa C bersifat case-sensitive artinya huruf besar dan kecil dianggap berbeda.
• Tidak boleh mengandung spasi.
• Tidak boleh mengandung symbol-simbol khusus, kecuali garis bawah (underscore). Yang termasuk symbol khusus yang tidak diperbolehkan antara lain : $, ?, %, #, !, &, *, (,), -, +, = dsb
• Panjangnya bebas, tetapi hanya 32 karakter pertama yang terpakai.

6. DEKLARASI

Deklarasi diperlukan bila kita akan menggunakan pengenal (identifier) dalam program. Identifier dapat berupa variable, konstanta dan fungsi.

6.1. DEKLARASI VARIABEL

Bentuk umum pendeklarasian suatu variable adalah : Nama_tipe nama_variabel;
Contoh : int x; // *Deklarasi x bertipe integer*

\6.2. DEKLARASI KONSTANTA

Dalam bahasa C konstanta dideklarasikan menggunakan preprocessor #define.
Contohnya : #define PHI 3.14 #define phone “0246704602”

6.3. DEKLARASI FUNGSI

Fungsi merupakan bagian yang terpisah dari program dan dapat diaktifkan atau dipanggil di manapun di dalam program. Fungsi dalam bahasa C ada yang sudah disediakan sebagai fungsi pustaka seperti printf(), scanf(), getch() dan untuk menggunakannya tidak perlu dideklarasikan. Fungsi yang perlu dideklarasikan terlebih dahulu adalah fungsi yang dibuat oleh programmer. Bentuk umum deklarasi sebuah fungsi adalah : Tipe_fungsi nama_fungsi(parameter_fungsi);
Contohnya :
float luas_lingkaran(int jari); void tampil(); int tambah(int x, int y);

7. OPERATOR

7.1. OPERATOR PENUGASAN

Operator penugasan (*Assignment operator*) dalam bahasa C berupa tanda sama dengan (“=”).

7.2. OPERATOR ARITMATIKA

Bahasa C menyediakan lima operator aritmatika, yaitu :

• .* : untuk perkalian

• / : untuk pembagian

• % : untuk sisa pembagian (modulus)

• + : untuk penjumlahan

• - : untuk pengurangan

7.2.1. PERKALIAN

```
#include <mega8535.h>
#include <delay.h>
void main()
{
int bil1;
int bil2;
DDRB=0xFF;
PORTB=0xFF;
bil1=4;
bil2=2;
PORTB=bil1*bil2;
}
```

7.2.2. PEMBAGIAN

```
#include <mega8535.h>
#include <delay.h>
void main()
{
int bil1;
int bil2;
DDRB=0xFF;
PORTB=0xFF;
bil1=10;
bil2=2;
PORTB=bil1/bil2;
}
```

7.2.3. MODULUS

```
#include <mega8535.h>
#include <delay.h>
void main()
{
int bil1;
int bil2;
DDRB=0xFF;
PORTB=0xFF;
bil1=13;
bil2=2;
PORTB=bil1%bil2;
}
```

7.2.4. PENJUMLAHAN

```
#include <mega8535.h>
#include <delay.h>
void main()
{
int bil1;
int bil2;
DDRB=0xFF;
PORTB=0xFF;
bil1=0x30;
bil2=0x20;
PORTB=bil1+bil2;
}
```

7.2.5. PENGURANGAN

```
#include <mega8535.h>
#include <delay.h>
void main()
{
int bil1;
int bil2;
DDRB=0xFF;
PORTB=0xFF;
bil1=0x30;
bil2=0x20;
PORTB=bil1-bil2;
}
```

7.3. OPERATOR HUBUNGAN (PERBANDINGAN)

Operator hubungan digunakan untuk membandingkan hubungan antara dua buah operand /sebuah nilai atau variable. Operasi majemuk seperti pada tabel dibawah ini:

Operator Hubungan

Operator Arti Contoh

< Kurang dari X<Y Apakah X kurang dari Y

<= Kurang dari sama dengan X<=Y Apakah X Kurang dari sama dengan Y

> Lebih dari X>Y Apakah X Lebih dari Y

>= Lebih dari sama dengan X==Y Apakah X Lebih dari sama dengan Y

== Sama dengan X==Y Apakah X Sama dengan Y

!= Tidak sama dengan X!= Y Apakah X Tidak sama dengan Y

7.4. OPERATOR LOGIKA

Jika operator hubungan membandingkan hubungan antara dua buah operand, maka operator logika digunakan untuk membandingkan logika hasil dari operator.

operator hubungan.

Operator logika ada tiga macam, yaitu :

• && : Logika AND (DAN)

• || : Logika OR (ATAU)

• ! : Logika NOT (INGKARAN)

Operasi AND akan bernilai benar jika dua ekspresi bernilai benar. Operasi OR akan bernilai benar jika dan hanya jika salah satu ekspresinya bernilai benar. Sedangkan operasi NOT menghasilkan nilai benar jika ekspresinya bernilai salah, dan akan bernilai salah jika ekspresinya bernilai benar.

```
#include <mega8535.h>
#include <delay.h>
void main()
{
char in1;
char in2;
DDRB=0xFF;
PORTB=0xFF;
in1=0xf0;
in2=0x40;
if((in1==0xf0) && (in2==0x40))
{PORTB = 0x2A;}
}
```

7.5. OPERATOR BITWISE (MANIPULASI PER BIT)

Operator bitwise digunakan untuk memanipulasi bit-bit dari nilai data yang ada di memori. Operator bitwise dalam bahasa C di SDCC adalah sebagai berikut :

- **<<** : Pergeseran bit ke kiri
- **>>** : Pergeseran bit ke kanan
- & : Bitwise AND
- ^ : Bitwise XOR (exclusive OR)
- | : Bitwise OR
- ~ : Bitwise NOT
- Pertukaran Nibble dan Byte
- Mengambil Bit yang paling Berbobot

7.5.1. OPERASI GESER KIRI (<<)

Operasi geser kiri merupakan operasi yang akan menggeser bit-bit kekiri sehingga bit 0 akan berpindah ke bit 1 kemudian bit 1 akan berpindah ke bit 2 dan seterusnya. Operasi geser kiri membutuhkan dua buah operan disebelah kiri tanda **<<** merupakan nilai yang akan digeser sedangkan disebelah kanannya merupakan jumlah bit penggerseran.

Contohnya :

Datanya = 0x03 << 2 ; // 0x03 digeser kekiri 2 bit hasilnya ditampung di datanya

a << = 1 // Isi variabel A digeser ke kiri 1 bit hasilnya

// kembali disimpan di A

```
#include <mega8535.h>
#include <delay.h>
void main()
{
char a, led;
DDRB=0xFF;
PORTB=0xFF;
led=0x01;
for (a=0;a<8;a++)
{
PORTB=led;
led=led <<1;
}
}
```

7.5.2. OPERASI GESER KANAN(>>)

Operasi geser kiri merupakan operasi yang akan menggeser bit-bit kekanan sehingga bit 7 akan berpindah ke bit 6 kemudian bit 6 akan berpindah ke bit 5 dan seterusnya. Operasi geser kanan membutuhkan dua buah operan disebelah kiri tanda **<<** merupakan nilai yang akan digeser sedangkan disebelah kanannya merupakan jumlah bit penggerseran.

Contohnya :

Datanya = 0x03 **>>** 2 ; // 0x03 digeser kekiri 2 bit hasilnya ditampung di datanya

a **>>** = 1 // Isi variabel A digeser ke kiri 1 bit hasilnya

// kembali disimpan di A

```
#include <mega8535.h>
#include <delay.h>
void main()
{
char a, led;
DDRB=0xFF;
PORTB=0xFF;
led=0x01;
for (a=0;a<8;a++)
{
PORTB=led;
led=led <<1;
}
}
```

7.5.3. OPERASI BITWISE AND (&)

Operasi bitwise AND akan melakukan operasi AND pada masing-masing bit, sehingga bit 0 akan dioperasikan dengan bit 0 dan bit 1 dan seterusnya.

Contohnya :

Hasil = 0x03 & 0x31; Operasinya 0x03 = 00000011

0x31 = 00110001

Hasil 0x01 = 00000001

```
#include <mega8535.h>
#include <delay.h>
void main()
```

```
{ char a=0x03;
char b=0x31;
DDRB=0xFF;
PORTB=0xFF;
PORTB= a & b ;}
```

7.5.4. OPERASI BITWISE OR (|)

Operasi bitwise OR akan melakukan operasi OR pada masing-masing bit, sehingga bit0 akan dioperasikan dengan bit 0 dan bit 1 dan seterusnya.

Contohnya :

Hasil = 0x05 | 0x31; Operasinya 0x01 = 00000001

0x31 = 00110001

Hasil 0x01 = 00110001

```
#include <mega8535.h>
#include <delay.h>
void main()
{ char a=0x03;
char b=0x31;
DDRB=0xFF;
PORTB=0xFF;
PORTB= a | b ; }
```

7.5.5. OPERASI BITWISE XOR(^)

Operasi bitwise XOR akan melakukan operasi XOR pada masing-masing bit, sehingga bit 0 akan dioperasikan dengan bit 0 dan bit 1 dan seterusnya.

Contohnya : Hasil = 0x02 ^ 0xFA; Operasinya 0x02 = 00000010

0xFA = 11111010

Hasil 0x01 = 11111000

```
#include <mega8535.h>
#include <delay.h>
void main()
{ char a=0x02;
char b=0xFA;
DDRB=0xFF;
PORTB=0xFF;
PORTB=a ^ b ;}
```

7.5.6. OPERASI BITWISE NOT(~)

Operasi bitwise XOR akan melakukan operasi XOR pada masing-masing bit, sehingga bit 0 akan dioperasikan dengan bit 0 dan bit 1 dan seterusnya.

Contohnya : Hasil = ~ 0x31; 0x31 = 00110001

Hasil ~0x31 = 11001110

```
#include <mega8535.h>
#include <delay.h>
void main()
{ char a= 0x31;
DDRB=0xFF;
PORTB=0xFF;
PORTB= ~a; }
```

7.5.7. PERTUKARAN NIBBLE DAN BYTE

Pertukaran nibble dalam bahasa C dikenali SDCC dengan bentuk pernyataan sebagai berikut ini:

```
volatile unsigned char i:
i = (( i << 4) | ( i >> 4)); //pertukaran nibble
```

Dan pernyataan sebagai berikut ini sebagai pertukaran byte:

```
volatile unsigned char j:
j = (( j << 8) | ( j >> 8)); //pertukaran byte
#include <mega8535.h>
#include <delay.h>
union kint
{
unsigned char a[2];
unsigned int b;
};
void main()
{
union kint tmp;
volatile unsigned char i=0x37;
volatile unsigned int j=0x9973;
DDRA=0xFF;
DDRB=0xFF;
DDRD=0xFF;
PORTA=i;
tmp.b=j;
PORTD= tmp.a[1];
PORTB=tmp.a[0];
i= ((i<<4) | (i>>4)); //pertukaran nibble
j= ((j<<8) | (j>>8)); //pertukaran byte
PORTA=i; //I dikeluarkan ke Port 1
tmp.b=j;
PORTB=tmp.a[0];
PORTD=tmp.a[1];
}
```

7.5.8. MENGAMBIL BIT YANG PALING BERBOBOT

Untuk mendapatkan bit yang paling berbobot (MSB) untuk tipe long, short, int, dan char maka dapat dilakukan dengan pertanyaan berikut:

```
Volatile unsigned char gint;
Unsigned char hop;
Hop = (gint >> 7) & 1 // mengambil MSB
#include <mega8535.h>
#include <delay.h>
void main()
{
volatile unsigned char gint=0xaa;
volatile unsigned char hob;
unsigned a;
DDRB=0xFF;
for(a=0;a<8;a++)
{
hob=(gint>>7) &1;
PORTB=hob;
gint=((gint<<1)|(gint>>7));
}
}
```

7.6. OPERATOR UNARY

Operator Unary merupakan operator yang hanya membutuhkan satu operand saja.

Dalam bahasa C terdapat beberapa operator unary, yaitu :

Tabel 2.3 Operasi Unary

Contohnya :

n = 0

Jum = 2 * ++n;

Jum = 2 * n++;

```
#include <mega8535.h>
#include <delay.h>
void main()
{
int a;
DDRB=0xFF;
for(a=0;a<20;a++)
PORTB=a;
}
```

7.7. OPERATOR MAJEMUK

Operator majemuk terdiri dari dua operator yang digunakan untuk menyingkat penulisan. Operasi majemuk seperti pada tabel dibawah ini

Tabel Operasi majemuk

Operator	Contoh	Kependekan dari
+=	Counter +=1;	Counter = counter + 1
-=	Counter -=1	Counter = counter - 1
*=	Counter *=1	Counter = counter * 1
/=	Counter /=1	Counter = counter / 1
%=	Counter %=1	Counter = counter % 1
<<=	Counter <<=1	Counter = counter << 1
>>=	Counter >>=1	Counter = counter >> 1
&=	Counter &=1	Counter = counter & 1
|=	Counter |=1	Counter = counter | 1
^=	Counter ^=1	Counter = counter ^ 1
~=	Counter ~=1	Counter = counter ~ 1

8. KOMENTAR PROGRAM

Komentar program hanya diperlukan untuk memudahkan pembacaan dan pemahaman suatu program (untuk keperluan dokumentasi program). Dengan kata lain, komentar program hanya merupakan keterangan atau penjelasan program. Untuk memberikan komentar atau penjelasan dalam bahasa C digunakan pembatas /* dan */ atau menggunakan tanda // untuk komentar yang hanya terdiri dari satu baris. Komentar program tidak akan ikut diproses dalam program (akan diabaikan).

Contoh pertama :

// program ini dibuat oleh ….

Dibelakang tanda // tak akan diproses dalam kompilasi. Tanda ini hanya untuk satu baris kalimat.

Contoh kedua :

/* program untuk memutar motor DC atau
motor stepper */

Bentuk ini berguna kalau pernyataannya berupa kalimat yang panjang sampai beberapa baris.

9. PENYELEKSIAN KONDISI

Penyeleksian kondisi digunakan untuk mengarahkan perjalanan suatu proses. Penyeleksian kondisi dapat diibaratkan sebagai katup atau kran yang mengatur jalannya air. Bila katup terbuka maka air akan mengalir dan sebaliknya bila katup tertutup air tidak akan mengalir atau akan mengalir melalui tempat lain. Fungsi penyeleksian kondisi penting artinya dalam penyusunan bahasa C, terutama untuk program yang kompleks.

9.1. STRUKTUR KONDISI "IF…."

Struktur if dibentuk dari pernyataan if dan sering digunakan untuk menyeleksi suatu kondisi tunggal. Bila proses yang diseleksi terpenuhi atau bernilai benar, maka pernyataan yang ada di dalam blok if akan diproses dan dikerjakan. Bentuk umum struktur kondisi if adalah :

if(kondisi)

pernyataan;

```
#include <mega8535.h>
#include <delay.h>
void main()
{
char inp1;
DDRA=0xFF;
DDRB=0xFF;
inp1=PORTB;
if(inp1==0x40)
{PORTA = 0x20;}
}
```

9.2. STRUKTUR KONDISI "IF.....ELSE…."

Dalam struktur kondisi if.....else minimal terdapat dua pernyataan. Jika kondisi yang diperiksa bernilai benar atau terpenuhi maka pernyataan pertama yang dilaksanakan dan jika kondisi yang diperiksa bernilai salah maka pernyataan yang kedua yang dilaksanakan. Bentuk umumnya adalah sebagai berikut :

if(kondisi)

pernyataan-1

else

pernyataan-2

Contoh

IF

```
if (angka = fo) /* bila angka sama dengan fo */
{ /*kerjakan berikut ini */
for (k = 0; k<4 ; k++)
{
i=tabel1(k);
PORTA = i; // pernyataan dalam blok ini bisa kosong
tunda50(100); // berarti tidak ada yang dikerjakan
}
}
else //bila tidak sama kerjakan berikut ini
{
for (k = 0; k<4 ; k++)
{
i=tabel2(k); // pernyataan dalam blok ini bisa kosong

PORTA = i; // berarti tidak ada yang dikerjakan
tunda50(100);
}
}
```

```
#include <mega8535.h>
#include <delay.h>
void main()
{
char inp1;
DDRA=0xFF;
DDRB=0xFF;
inp1=PORTB;
if(inp1==0x01)
{PORTA = 0x20;}
else
{PORTA=0x80;}
}
```

9.3. STRUKTUR KONDISI “SWITCH...CASE... DEFAULT…”

Struktur kondisi switch....case....default digunakan untuk penyeleksian kondisi dengan kemungkinan yang terjadi cukup banyak. Struktur ini akan melaksanakan salah satu dari beberapa pernyataan ‘case’ tergantung nilai kondisi yang ada di dalam switch. Selanjutnya proses diteruskan hingga ditemukan pernyataan ‘break’. Jika tidak ada nilai pada case yang sesuai dengan nilai kondisi, maka proses akan diteruskan kepada pernyataan yang ada di bawah ‘default’. Bentuk umum dari struktur kondisi ini adalah :

```
switch(kondisi)
{
case 1 : pernyataan-1;
break;
case 2 : pernyataan-2;
break;
.....
case n : pernyataan-n;
break;
default : pernyataan-m
}
```

contoh

SWITCH …. CASE …

```
switch(fo)
{
case 1:
for (k = 0; k<4 ; k++)
{
i=tabel1(k);
PORTA = i;
tunda(100);
}
break;
case 2:
for (k = 0; k<4 ; k++)
{
i=tabel2(k);
PORTA = i;
tunda(100);
}
break;
#include <mega8535.h>
#include <delay.h>
void main()
{
char a;
DDRA=0xFF;
DDRB=0xFF;
a=PORTA;
switch(a)
{
case 0: PORTB=5;break;
case 1: PORTB=10;break;
case 2: PORTB=15;break;
case 3: PORTB=20;break;
case 4: PORTB=40;break;
case 5: PORTB=60;break;
```

```
default: PORTB=0;break;
}
}
```

10. PERULANGAN

Dalam bahasa C tersedia suatu fasilitas yang digunakan untuk melakukan proses yang berulangulang sebanyak keinginan kita. Misalnya saja, bila kita ingin menginput dan mencetak bilangan dari 1 sampai 100 bahkan 1000, tentunya kita akan merasa kesulitan. Namun dengan struktur perulangan proses, kita tidak perlu menuliskan perintah sampai 100 atau 1000 kali, cukup dengan beberapa perintah saja. Struktur perulangan dalam bahasa C mempunyai bentuk yang bermacam-macam.

10.1. STRUKTUR PERULANGAN " WHILE"

Perulangan WHILE banyak digunakan pada program yang terstruktur. Perulangan ini banyak digunakan bila jumlah perulangannya belum diketahui. Proses perulangan akan terus berlanjut selama kondisinya bernilai benar (true) dan akan berhenti bila kondisinya bernilai salah. Bentuk umum dari struktur kondisi ini adalah:

```
While (ekspresi)
{
Pernyataan_1
Pernyataan_2
}
```

Contoh Program 1 :

```
while (!TF0);
{
TF0 = 0;
TR0 = 0;
}
#include <mega8535.h>
#include <delay.h>
void main()
{
char a=10;
DDRA=0xFF;
while(a>=0)
{
PORTA=a;
a--;
}
}
```

10.2. STRUKTUR PERULANGAN "DO.....WHILE…"

Pada dasarnya struktur perulangan do....while sama saja dengan struktur while hanya saja pada proses perulangan dengan while, seleksi berada di while yang letaknya di atas sementara pada perulangan do....while, seleksi while berada di bawah batas perulangan. Jadi dengan menggunakan struktur do…while sekurang-kurangnya akan terjadi satu kali perulangan. Bentuk umum dari struktur kondisi ini adalah:

```
Do
{
Pernyataan_1
Pernyataan_2
}
While (ekspresi)
#include <mega8535.h>
#include <delay.h>
void main()
{
char a=10;
DDRA=0xFF;
do
{
PORTA=a;
a--;
} while(a>=0);
}
```

10.3. STRUKTUR PERULANGAN “FOR”

Struktur perulangan for biasa digunakan untuk mengulang suatu proses yang telah diketahui jumlah perulangannya. Dari segi penulisannya, struktur perulangan for tampaknya lebih efisien karena susunannya lebih simpel dan sederhana. Bentuk umum perulangan for adalah sebagai berikut :

```
for(inisialisasi; syarat; penambahan)
pernyataan;
```

Keterangan:

Inisialisasi : pernyataan untuk menyatakan keadaan awal dari variabel kontrol.

syarat : ekspresi relasi yang menyatakan kondisi untuk keluar dari perulangan.

penambahan : pengatur perubahan nilai variabel kontrol.

Contoh

```
#include <mega8535.h>
#include <delay.h>
void main()
for (k = 0; k<4 ; k++)
{
i=tabel1(k);
PORTA = i;
tunda50(100);
}
{
char a;
DDRA=0xFF;
for(a=10;a>=0;a--)
PORTA=a;
}
```

11. ARAY (LARIK)

Array merupakan kumpulan dari nilai-nilai data yang bertipe sama dalam urutan tertentu yang menggunakan nama yang sama. Letak atau posisi dari elemen array ditunjukkan oleh suatu index. Dilihat dari dimensinya array dapat dibagi menjadi Array dimensi satu, array dimensi dua dan array multi-dimensi.

11.1. ARRAY DIMENSI SATU

Setiap elemen array dapat diakses melalui indeks. Indeks array secara default dimulai dari 0. Deklarasi Array Bentuk umum :

Deklarasi array dimensi satu:

[Tipe_array][nama_array][elemen1];

11.2. ARRAY DIMENSI DUA

Array dua dimensi merupakan array yang terdiri dari m buah baris dan n buah kolom. Bentuknya dapat berupa matriks atau tabel.

Deklarasi array dimensi dua :

[Tipe_array][nama_array][elemen1][elemen2];

11.3. ARRAY MULTI-DIMENSI

Array multi-dimensi merupakan array yang mempunyai ukuran lebih dari dua. Bentuk pendeklarasian array sama saja dengan array dimensi satu maupun array dimensi dua. Bentuk umumnya yaitu :

[tipe_array][nama_array][elemen1][elemen2]…[elemenN];

12. FUNGSI

12.1. PENGERTIAN FUNGSI

Fungsi merupakan suatu bagian dari program yang dimaksudkan untuk mengerjakan suatu tugas tertentu dan letaknya terpisah dari program yang memanggilnya. Fungsi merupakan elemen utama dalam bahasa C karena bahasa C sendiri terbentuk dari kumpulan fungsi-fungsi. Dalam setiap program bahasa C, minimal terdapat satu fungsi yaitu fungsi main(). Fungsi banyak diterapkan dalam program-program C yang terstruktur. Keuntungan penggunaan fungsi dalam program yaitu program akan memiliki struktur yang jelas (mempunyai readability yang tinggi) dan juga akan menghindari penulisan bagian program yang sama.

12.2. PENDEFISIAN FUNGSI

Sebelum digunakan fungsi harus didefinisikan terlebih dahulu. Bentuk definisi fungsi adalah:

```
Tipe_Nilai_Balik nama_fungsi(argumen1, argumen2)
{
Pernyataan1;
Pernyataan1;
return(ekspresi);
}
```

Contoh:

```
int jumlah(int bil1,int bil2) //definisi fungsi jumlah
{
int hasil;
hasil = bil1 + bil2
return(hasil);
}
```

```
int jumlah(int bil1,int bil2)
```

1 2 3 4

Keterangan:

1. tipe data nilai balik fungsi
2. merupakan nama fungsi
3. tipe argumen
4. nama argumen

```
#include <mega8535.h>
#include <delay.h>
int jumlah(int bil1,int bil2)
{
return(bil1+bil2);
}
void main()
{
DDRA=0xFF;
PORTA=jumlah(20,50);
}
```

12.3. PROTOTYPE FUNGSI

Ketentuan pendefinisian fungsi yang mendahului fungsi pemanggil dapat merepotkan untuk program yang komplek atau besar. Untuk mengatasi hal tersebut maka fungsi dapat dideklarasikan sebelum digunakan, terletak sebelum fungsi main. Deklarasi fungsi dikenal dengan prototype fungsi. Cara mendeklarasikan fungsi sama dengan header fungsi dan diakhiri tanda titik koma (;)

```
#include <mega8535.h>
#include <delay.h>
int jumlah(int bil1,int bil2);
void main()
{
DDRA=0xFF;
```

```
PORTA=jumlah(20,50);
}
int jumlah(int bil1,int bil2)
{
return(bil1+bil2);
}
```

12.4. VARIABEL LOKAL DAN GLOBAL

Variabel lokal adalah variabel yang dideklarasikan di dalam suatu fungsi, variabel ini hanya dikenal fungsi tersebut. Setelah keluar dari fungsi ini maka variabel ini akan hilang. Variabel global adalah variabel yang dideklarasikan di luar fungsi, sehingga semua fungsi dapat memakainya.

```
#include <mega8535.h>
#include <delay.h>
int jumlah(int bil1,int bil2);
int data1;
void main()
{
int data1;
DDRA=0xFF;
data1=jumlah(20,50);
PORTA = data1;
}
int jumlah(int bil1,int bil2)
{
return(bil1+bil2);
}
```

12.5. KATA KUNCI *EXTERN* DAN *STATIC*

Kata kunci *extern* dan *static* digunakan untuk menyatakan sifat dari variabel atau fungsi. Suatu variabel atau fungsi yang didepannya ditambah dengan kata kunci extern maka artinya variabel atau fungsi tersebut didefinisikan di luar file tersebut. Variabel global atau fungsi yang didepannya ditambah kata kunci static mempunyai arti bahwa variabel global atau fungsi tersebut bersifat pivate bagi file tersebut, sehingga tidak dapat diakses dari file yang lain. Kata kunci *static* yang ditambahkan didepan variabel lokal (variabel di dalam suatu fungsi) artinya variabel tersebut dialokasikan pada memori statik. Nilai yang tersimpan dalam variabel statik tidak hilang walaupun sudah keluar dari fungsi.

12.6. FUNGSI TANPA NILAI BALIK

Fungsi yang tidak mempunyai nilai balik menggunakan kata kunci void sedangkan fungsi yang tidak mempunyai argumen, setelah nama fungsi dalam kurung
dapat kosong atau dengan menggunakan kata kunci void.

Contoh:

```
void tunda(void)
{
for(i = 0; i < 10 ; i++);
}
```

atau

```
void tunda()
{
for(i=0;i<10;i++);
{}
}
/* fungsi tunda_panjang */
void tunda_panjang(int n)
{
int i;
for (i=0; i<n;i++)
tunda();
}
```

12.7. FUNGSI DENGAN NILAI BALIK (*RETURN VALUE*)

Nilai balik dinyatakan delam pernyataan return. Tipe nilai balik dapat berupa char, int, short, long, atau float

Contoh:

```
int jumlah(int bil1,int bil2)
{
return(bil1+bil2);
}
```

13. MENYISIPKAN INSTRUKSI ASSEMBLI

CVAVR juga mendukung penyisipan instruksi dalam bahasa asembli. Instruksi asembli dituliskan diantara kata kunci #asm dan #endasm seperti berikut ini:

```
Void tunda()
{
#asm
mov r0, #20
00001$: djnz r0, 00001$
#endasm;
}
#include <mega8535.h>
#include <delay.h>
void tunda()
{
#asm
mov r0, #0x0f5
01$: mov r1, #0x0ff
02$: mov r2, #0
djnz r1, 02$
djnz r0, 01$
#endasm;
}
void main()
{
char a;
char k; DDRA=0xFF;
DDRB=0xFF
```

13.1. PENGGUNAAN LABEL PADA INSTRUKSI ASSEMBLI

Label pada instruksi assembli berupa anggka nnnnn$ dengan nnnnn berupa angka di bawah 100. label pada instruksi assembli hanya dikenal oleh instruksi assembli, bahasa C tidak mengenal label pada penyisipan assembli dan juga sebaliknya.

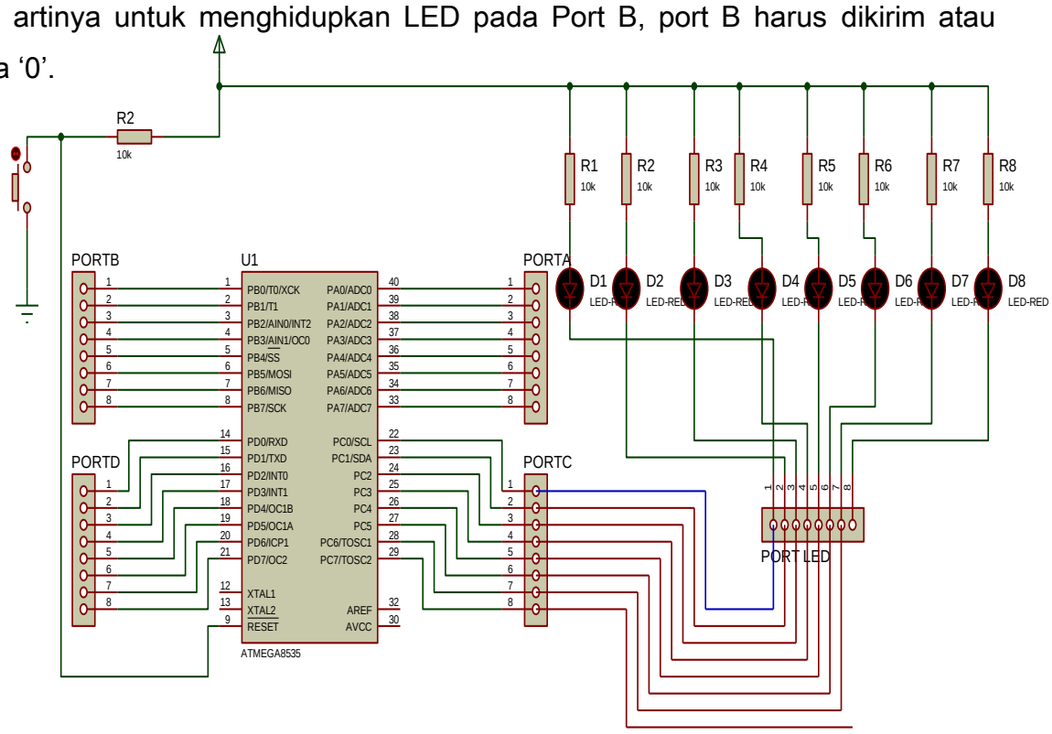
Contoh:

```
Void conto()
{
/*Pernyataan C*/
#asm
; beberapa instruksi asembli
ljmp 00003$
#endasm;
/*Pernyataan C*/
clabel: /*instruksi assembli tidak mengenal*/
#asm
00003$: ; hanya dapat dikenal oleh assembli
#endasm;
}
```

BAB I
APLIKASI OUTPUT

1.1. RANGKAIAN LAMPU LED

Rangkaian minimum untuk menghidupkan 8 LED melalui Port B ditunjukan pada Gambar 3.1. yang perlu diperhatikan adalah konfigurasi rangkaian LED yaitu *Common Anode* (CA) artinya untuk menghidupkan LED pada Port B, port B harus dikirim atau diberi logika '0'.



Gambar 1.1. Gambar rangkaian lampu led

1.2. PEMROGRAMAN MENYALAKAN LED

1.2.a Alat dan Bahan

- Trainer Atmega 8535
- Kabel conector molek 8 pin
- Kabel USB
- Laptop/notebook
- Software Prog ISP168
- Software Code vision AVR
- Multimeter

1.2.b Langkah kerja

1. Hubungkan modul Output LED (Port Led) dengan Sismin AVR Atmega8535, pilih PORTC mikrokontroller,

2. Tulis program bahasa C pada codevisioan AVR untuk menghidupkan LED , buat new project atur port I/O kemudian beri nama file Led1,

Ketik program sebagai berikut :

```
//-----------------------------------------------------
//Program LED Menyala
//-----------------------------------------------------
#include <mega8535.h>
#include <delay.h>
void main(void)
{
char a; a=0x000;
DDRC=0xFF;
while(1)
{
PORTC = a;
}
}
```

3. Jika sudah benar penulisan program, lakukan compile / build all file, sehinga menghasilkan output file. Hex

4. Buka software ProgISP 168, hubungkan kabel USB dari Laptop ke USB downloader

5. Lakukan download program ke Trainer Atmega 8535

6. Pastikan kabel ISP sudah terhubung

7. Lihat hasilnya.

8. Perhatian Sumber daya (power supply +5V) didapat dari kabel USB, jika menggunakan sumber dari luar lepas kabel USB. Jika menggunakan Power USB trainer dapat diprogram dan langsung dijalankan !

9. Lakukan langkah 3 – 6 untuk percobaan berikutnya.

Cara kerja program:

Pada Program LED Menyala, di perlukan deklarasi register dan delay untuk mikrokontroller jenis ATMEGA8535. Setelah mendeklarasi register, maka program akan masuk ke dalam program utama. Di dalam program utama, terdapat variabel karakter yang berfungsi untuk menyimpan data angka 0x000. Data 0x00 digunakan untuk menyalakan LED karena LED di pasang common anoda Data tersebut akan di keluarkan oleh mikrokontroller dengan menggunakan PORTC. Data tersebut di simpan dalam variabel a yang dideklarasikan sebagai char. Data tersebut dikeluarkan dengan menggunakan PORTC sehingga harus dideklarasikan PORTC sebagai output dengan DDRC=0xFF. Instruksi while merupakan instruksi perulangan, sehingga mikrokontroller akan mengeluarkan data yang di simpan oleh variabel karakter secara terus menerus.

1.3. PEMROGRAMAN LED BERKEDIP

Setelah membuat dan menjalankan program menyalakan lampu LED, maka sekarang saatnya anda membuat program kedua yang digunakan untuk menghidupkan LED berkedip.

Program sebagai berikut :

```
//-------------------------------------------------------
//Program Bab 3.2. LED Berkedip
//-------------------------------------------------------
#include <mega8535.h>
#include <delay.h>
void main(void)
{
char a; char b;
a=0x000; b=0x0FF;
DDRB=0xFF;
while(1)
{
PORTB= a;
delay_ms(500);
PORTB= b;
delay_ms(500);
}
}
```

Cara kerja program:

Pada program Program LED Berkedip, terlihat menggunakan mikrokontroller ATMEGA8535, sehingga di perlukan deklarasi register untuk mikrokontroller jenis ATMEGA8535. Di dalam program utama, terdapat variabel karakter yang berfungsi untuk menyimpan data 00 dan FF. Data tersebut akan di keluarkan oleh mikrokontroller dengan menggunakan port 0. Instruksi while merupakan instruksi perulangan.

1.4. PEMROGRAMAN LED FLIP FLOP

Setelah membuat dan menjalankan program menyalakan lampu LED berkedip, maka sekarang saatnya Anda membuat program ketiga yang digunakan untuk menghidupkan LED flip-flop 1.

Ketik program sebagai berikut :

```
//-------------------------------------------------------
//Program Bab 3.3. LED Flip-Flop
//-------------------------------------------------------
#include <mega8535.h>
#include <delay.h>
void main(void)
{
char a; char b;
a=0x00f; b=0x0f0;
DDRB=0xFF;
while(1)
{
PORTB= a;
```

```
delay_ms(500);
PORTB= b;
delay_ms(500);
}
}
```

Cara kerja program:

Pada program LED Flip-Flop di perlukan deklarasi register untuk mikrokontroller jenis ATMEGA8535. Setelah mendeklarasi register, maka program akan mendeklarasikan waktu 1 sekon. Waktu tersebut berfungsi untuk waktu tunda. Kemudian mikrokontroller akan mengeksekusi program utama. Di dalam program utama, terdapat variabel karakter yang berfungsi untuk menyimpan data 0x00F dan 0x0F0. Data tersebut akan di keluarkan oleh mikrokontroller dengan menggunakan port B.

1.5. PEMROGRAMAN LED BERJALAN KEKANAN

Setelah membuat dan menjalankan program menyalakan lampu LED flip-flop, maka sekarang saatnya Anda membuat program yang digunakan untuk menghidupkan LED berjalan kanan. Program LED berjalan kekanan ini dijalankan pada hardware nyala led berlogika tinggi atau logika 1. jika menggunakan logika rendah maka LED bukan menyala tetapi akan mati. Program LED berjalan kekanan menggunakan operasi geser kiri . Operasi geser kiri akan menggeser bit-bit kekanan sehingga bit 0 akan berpindah ke bit 1 dan bit 1 akan berpindah ke bit 2 dan seterusnya.

Ketik program sebagai berikut :

```
//-----------------------------------------------------
//Program Bab 3.4. LED Berjalan Kekanan
//-----------------------------------------------------
#include <mega8535.h>
#include <delay.h>
void main(void)
{
volatile unsigned char a=0x01;
DDRB=0xFF;
while(1)
{
a=((a>>7) | (a<<1));
delay_ms(1000);
PORTB=a;
}
}
```

Cara kerja program:

Pada program Program LED berjalan Kekanan di perlukan deklarasi register untuk mikrokontroller jenis ATMEGA8535. Setelah mendeklarasi register, maka program akan mendeklarasikan waktu kurang lebih 1 sekon. Kemudian mikrokontroller akan mengeksekusi program utama. Di dalam program utama, terdapat variabel karakter yang berfungsi untuk menyimpan data 0x01. Data tersebut akan di keluarkan oleh mikrokontroller dengan menggunakan port 0. kemudian mikrokontroller menjalankan

operasi geser kiri yang menggeser bit – bit kekanan sehingga bit 0 akan berpindah ke bit 1 bit 1 akan berpindah ke bit 2 demikian seterusnya. Diantara operasi program LED bergeser kanan mikrokontrol mengeluarkan data di PORTB terdapat waktu tunda kurang lebih 1 sekon. Didalam program utama terdapat pernyataan while(1). Pernyataan itu berfungsi untuk melakukan *Looping* secara terus menerus.

1.6. PEMROGRAMAN LED BERJALAN KEKIRI

Setelah membuat dan menjalankan program menyalakan lampu LED berjalan kekanan, maka sekarang saatnya Anda membuat program yang digunakan untuk menghidupkan LED berjalan kekiri. Program LED berjalan kekanan menggunakan operasi geser kiri. Operasi geser kiri akan menggeser bit-bit kekanan sehingga bit 7 akan berpindah ke bit 6 dan bit 6 akan berpindah ke bit 5 dan seterusnya.

Program sebagai berikut :

```
//-----------------------------------------------------
//Program Bab 3.5. LED berjalan ke kiri
//-----------------------------------------------------
#include <mega8535.h>
#include <delay.h>
void main(void)
{
volatile unsigned char a=0x01;
DDRB=0xFF;
while(1)
{
a=((a<<7) | (a>>1));
delay_ms(500);
PORTB=a;
}
}
```

Cara kerja program:

Pada program Program LED berjalan ke kiri di perlukan deklarasi register untuk mikrokontroller jenis ATMEGA8535. Setelah mendeklarasi register, maka program akan mendeklarasikan waktu kurang lebih 1 sekon. Kemudian mikrokontroller akan mengeksekusi program utama. Di dalam program utama, terdapat variabel karakter yang berfungsi untuk menyimpan data 0x01. Data tersebut akan di keluarkan oleh mikrokontroller dengan menggunakan port 0. kemudian mikrokontroller menjalankan operasi geser kekiri. Diantara operasi geser kiri dan mengeluarkan data di PORTB tersebut terdapat waktu tunda kurang lebih 1 sekon. Didalam program utama terdapat pernyataan while(1). Pernyataan itu berfungsi untuk melakukan *Looping* secara terus menerus.

1.7. PEMROGRAMAN LED BERJALAN BOLAK-BALIK

Setelah membuat dan menjalankan program menyalakan lampu LED berjalan menyala kekiri, maka sekarang saatnya Anda membuat program ketuga yang digunakan untuk menghidupkan LED bolak balik. Program LED bolak balik menggunakan operasi pernyataan geser kanan dan geser kiri.

Ketik program sebagai berikut :

```
//-----------------------------------------------------
//Program Bab 3.6. ROLING LED
//-----------------------------------------------------
#include <mega8535.h>
#include <delay.h>
void jalankiri(unsigned int n)
{
unsigned char i=0, a=0x01;
DDRC=0xFF;
PORTC = 0xff;
while(n)
{
for(i=0;i<7;i++)
{
a=((a>>7) | (a<<1));
delay_ms(100);
PORTC=a;
}
n--;
}
}
void jalankanan(unsigned int n)
{
unsigned char i=0, a=0x80;
DDRC=0xFF;
PORTC = 0xFF;
while(n)
{
for(i=0;i<7;i++)
{
a=((a<<7) | (a>>1));
delay_ms(100);
PORTC=a;
}
n--;
}
}
void main(void)
{
while(1)
{
DaryantoKentus:
jalankiri(1);
jalankanan(1);
goto DaryantoKentus;
```

```
}
}
```

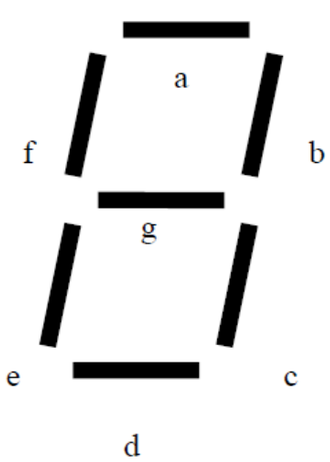
Cara kerja program:

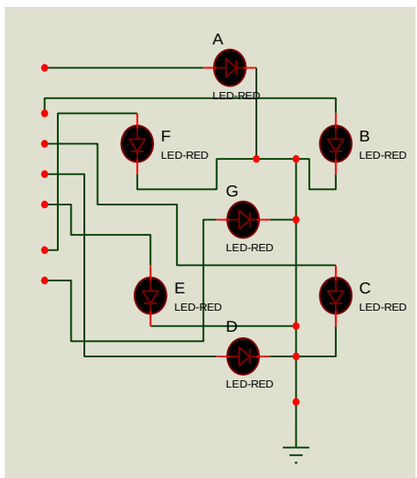
Pada program menyalakan LED dari kiri ke kanan di perlukan deklarasi register untuk mikrokontroller jenis ATMEGA8535. Setelah mendeklarasi register, maka program akan mendeklarasikan waktu kurang lebih 1 sekon. Waktu tersebut berfungsi untuk waktu tunda. Waktu tunda itu tidak tidak akurat. Kemudian mikrokontroller akan mengeksekusi program utama. Di dalam program utama, terdapat variabel karakter yang berfungsi untuk menyimpan data 0x01. Data tersebut akan di keluarkan oleh mikrokontroller dengan menggunakan port 0. kemudian mikrokontroller menjalankan operasi geser kanan. Diantara operasi geser kiri dan mengeluarkan data di port 0 tersebut terdapat waktu tunda kurang lebih 1 sekon. Didalam program utama terdapat pernyataan while(1). Pernyataan itu berfungsi untuk melakukan *Looping* secara terus menerus.

APLIKASI SEVEN SEGMEN

1.8. PENDAHULUAN

Peralatan keluaran yang sering digunakan dalam menampilkan bilangan adalah penampil seven segmen yang ditunjukkan pada gambar 4.1 (a). tujuh segmen tersebut dilabelkan dengan huruf a sampai g





Gambar 1.2. (a) Tampilan Fisik LED, (b) Skema dalam LED

Peraga seven segmen dapat dibuat dalam berbagai cara. Tiap tujuh segmen tersebut dapat berupa filamen tipis yang berpijar. Jenis peraga ini disebut peraga pijar (*meandescent display*), dan sama dengan bola lampu biasa. Peraga jenis lain adalah LCD

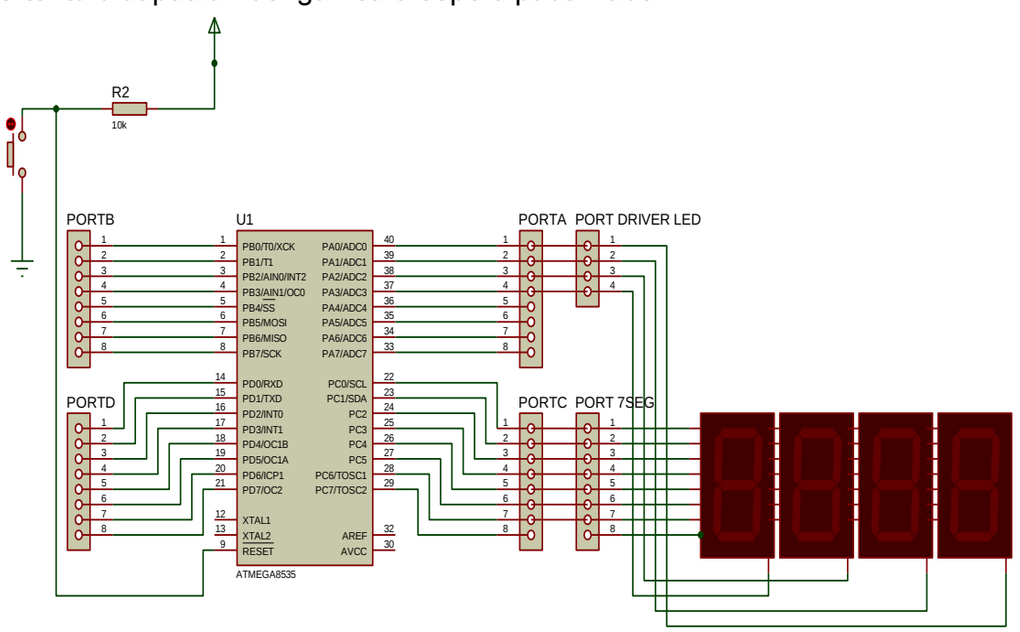
(*liquid crystal display*), peraga cairan, yang ,menghasilkan angka – angka berwarna kelabu atau puth perak. Dioda pemancar cahaya (LED, *Light Emiting Dioda*) menghasilkan cahaya kemerah – merahan. Pada peraga LED, LED membutuhkan arus khusus sebesar 20 mA. Karena berupa dioda, LED sensitif terhadap polaritas. Katoda (K) harus dihubung ke negatif (GND) dari catu daya dan Anoda (A) dihubung ke positif dari catu daya. Seven segmen ini mempunyai 2 tipe yaitu *common anoda* dan *common katoda*. Gambar 4.1(b) memperlihatkan catu daya yang dihubungkan ke seven segmen *common katode*.

1.9. RANGKAIAN SEVEN SEGMENT TUNGGAL 1

1.9.a Alat dan Bahan

- Trainer Atmega 8535
- Kabel conector molek 8 pin
- Kabel USB
- Laptop/notebook
- Software Prog ISP168
- Software Code vision AVR
- Multimeter

Rangkaian seven segment tunggal adalah rangkaian untuk menggerakkan penampil 7 segment secara langsung dari port keluaran mikrokontroller. Penampil seven segment yang digunakan common anoda. Data yang digunakan untuk menghasilkan angka atau huruf tertentu didapatkan dengan cara seperti pada Tabel 1.1



Gambar 1.3. Rangkaian aplikasi penggerak seven segmen *common catode*

char	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	HEX
	DOT	G	F	E	D	C	B	A	
0	0	0	1	1	1	1	1	1	0X3F
1	0	0	0	0	0	1	1	0	0X06
2	0	1	0	1	1	0	1	1	0X5B
3	0	1	0	0	1	1	1	1	0X4F
4	0	1	1	0	0	1	1	0	0X66
5	0	1	1	0	1	1	0	1	0X6D
6	0	1	1	1	1	1	0	1	0X7D
7	0	0	0	0	0	1	1	1	0X07
8	0	1	1	1	1	1	1	1	0X7F
9	0	1	1	0	1	1	1	1	0X6F

Tabel 1.1 Data Karakter Angka Pada 7 Segment *common catode*

1.9.1. PEMROGRAMAN SEVENT SEGMENT TUNGGAL 1

1.9.1.b Langkah Kerja

1. Hubungkan modul output port 7 segment dengan modul sismin Atmega 8535 pilih salah satu port.
2. Buka software codeVision AVR kemudian tulis program seperti berikut:
3. Program ini digunakan untuk menampilkan data angka dari 0 sampai 9 dan kembali ke awal secara terus menerus.
4. Setelah selesai lakukan compile program dan pastikan tidak ada error
5. Buka software ISPprog 168 untuk proses downloader
6. Hubungkan kabel USB downloader dengan USB notebook
7. lakukan proses downloader (pastikan kabel ISP terhubung)!
8. Lihat hasil akhir
9. Lakukan langkah kerja tersebut untuk percobaan berikutnya

Ketik program sebagai berikut ini:

```
//-----------------------------------------------------
// Program Sevent Segmen Tunggal
//-----------------------------------------------------
#include <mega8535.h>
#include <delay.h>
#include <stdio.h>
```

```
unsigned char rr=0;
unsigned char data1;
void bin7seg()
{
switch(data1)
{
case 0 :
PORTA = 0x3F;
break;
case 1 :
PORTA = 0x06;
break;
case 2 :
PORTA = 0x5B;
break;
case 3 :
PORTA = 0x4F;
break;
case 4 :
PORTA = 0x66;
break;
case 5 :
PORTA = 0x6D;
break;
case 6 :
PORTA = 0x7D;
break;
case 7 :
PORTA = 0x07;
break;
case 8 :
PORTA = 0x7F;
break;
case 9 :
PORTA= 0X6F;
break;


}
}
void display(unsigned int x)
{
int digit1;
digit1=(x/1);
data1=digit1;
bin7seg();
}
void main(void)
{
DDRA=0xFF;
DDRC=0xFF;
while(1)
{
awal:
(rr=0);
ulang3:
PORTC=0xFE;
display(rr);delay_ms(1000);
rr++;
```

```
if (rr<10) goto ulang3;
else
goto awal;
     }
}
```

Cara kerja program:

Pada program Sevent Segmen Tunggal, di perlukan deklarasi register untuk mikrokontroller jenis ATMEGA 8535. Setelah mendeklarasi register, maka program akan masuk ke dalam program utama. Di dalam program utama, mikrokontroller akan mengeluarkan data angka 0. Data tesebut di konversi BCD ke karakter 7-segment dan akan di keluarkan oleh mikrokontroller dengan menggunakan PORT A. Kemudian penyalaan led dikendalikan oleh driver PORTC memanggil tunda 1000 mili sekon dan memangil data angka 2 Instruksi while merupakan instruksi perulangan, sehingga mikrokontroller akan mengeluarkan data angka 0 sampai 9 secara terus menerus.

1.9.2. APLIKASI SEVEN SEGMENT TERMULTIPLEKS

Rangkaian seven segment termultipleks Seven Segment adalah rangkaian untuk menggerakkan 4 buah penampil 7 segment secara scaning data langsung dari port keluaran mikrokontroller dengan data input Seven Segment. 7-segment ini dikendalikan oleh PORT pada mikro Atmega 8535 secara langsung.

1.9.3. PEMROGRAMAN SEVENT SEGMENT TERMULTIPLEKS

Lakukan langkah kerja seperti program 7 segment tunggal. Setelah rangkaian sevent segment dibuat dan dihubungkan dengan port pararel mikrokontroller, maka sekarang saatnya Anda membuat program yang digunakan untuk menampilkan data menit dan detik.

Ketik program sebagai berikut ini

```
//-------------------------------------------------------
//Program Menampilkan menit dan detik
//-------------------------------------------------------
#include <mega8535.h>
#include <delay.h>
#include <stdio.h>
unsigned char rr=0;
unsigned char i=0;
unsigned char a=0;
unsigned char b=0;
unsigned char c=0;
unsigned char d=0;
unsigned char data1;
void bin7seg()
{
switch(data1)
```

```
{
case 0 :
PORTD = 0x3F;
break;
case 1 :
PORTD = 0x06;
break;
case 2 :
PORTD = 0x5B;
break;
case 3 :
PORTD = 0x4F;
break;
case 4 :
PORTD = 0x66;
break;
case 5 :
PORTD = 0x6D;
break;
case 6 :
PORTD = 0x7D;
break;
case 7 :
PORTD = 0x07;
break;
case 8 :
PORTD = 0x7F;
break;
case 9 :
PORTD= 0X6F;
break;
}
}
void display(unsigned int x)
{
int digit1;
int digit2;
digit1=(x/1);
data1=digit1;
bin7seg();
digit2=(digit1*10);
data1=digit2;
bin7seg();
}
void main(void)
{
DDRD=0xFF;
DDRC=0xFF;
while(1)
{
awal:
(rr=0);
(i=0);
(a=0);
(b=0);
(c=0);
(d=0);
ulang3:
++rr;
```

```
PORTC=0XFB;
PORTD=0x48; delay_ms(2);
PORTC=0xF7;
display(d); delay_ms(2);
PORTC=0xFD;
display(b); delay_ms(2);
PORTC=0xFE;
display(i);delay_ms(2);
if (rr<125) goto ulang3;
else
(rr=0);
++i;
++a;
if (a<10) goto ulang3;
 else
(rr=0);
(a=0);
(i=0);
++b;
++c;
if (c<6) goto ulang3;
   else
(rr=0);
(a=0);
(i=0);
(b=0);
(c=0);
++d ;
if (d<10) goto ulang3;
else
goto awal;
}
}
```

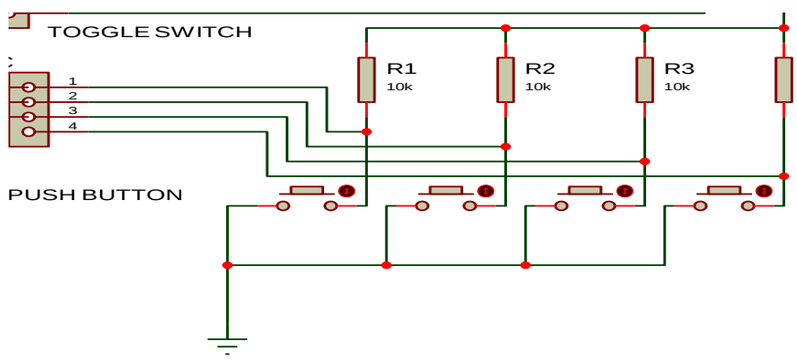
Cara kerja program:

Pada program Program 7-segment 2. Sevent Segmen Termultiplek, di perlukan deklarasi register untuk mikrokontroller jenis AT89x51. Setelah mendeklarasi register, maka program akan masuk ke dalam program utama. Di dalam program utama, pada awal tampilan mikrokontroller akan mengeluarkan data angka 0 : 00 kemudian mulai menampilkan angka 0 sampai 9 pada digit 1 dengan delay 1 detik, setelah sampai angka 9 data digit 2 akan menampilkan data dari 0 sampai 6 dengan delay 10 detik tiap perubahan nilai . Berikutnya pada digit 3 menampilkan tanda : dan pada digit 4 menampilkan angka menit yang menghitung dari 0 sampai 9 dengan delay 60 detik tiap perubahan nilai . Data tersebut akan masuk kedalam prosedure display untuk mengeluarakan data pada portD dan pada penyalaan digit 1,2,3 dan 4 led dikendalikan oleh PORTC mikrokontroller. Data berulang sampai 10 menit kembali ke awal secara terus menerus.

BAB II
APLIKASI INPUT

2.1. PENDAHULUAN

Agar tombol tersebut dapat memberi input pada mikrokontroller, maka terlebih dahulu tombol ini harus disusun dalam sebuah rangkaian di mana terdapat perbedaan kondisi pada pin-pinnya antara kondisi tidak ada penekanan tombol, penekanan tombol 1, 2, 3 dan seterusnya. Kondisi tidak adanya penekanan tombol diatur dengan adanya kondisi logika high. Pada saat tombol tidak ditekan, maka arus akan mengalir dari VCC melalui resistor menuju ke port seperti tampak pada gambar berikut.

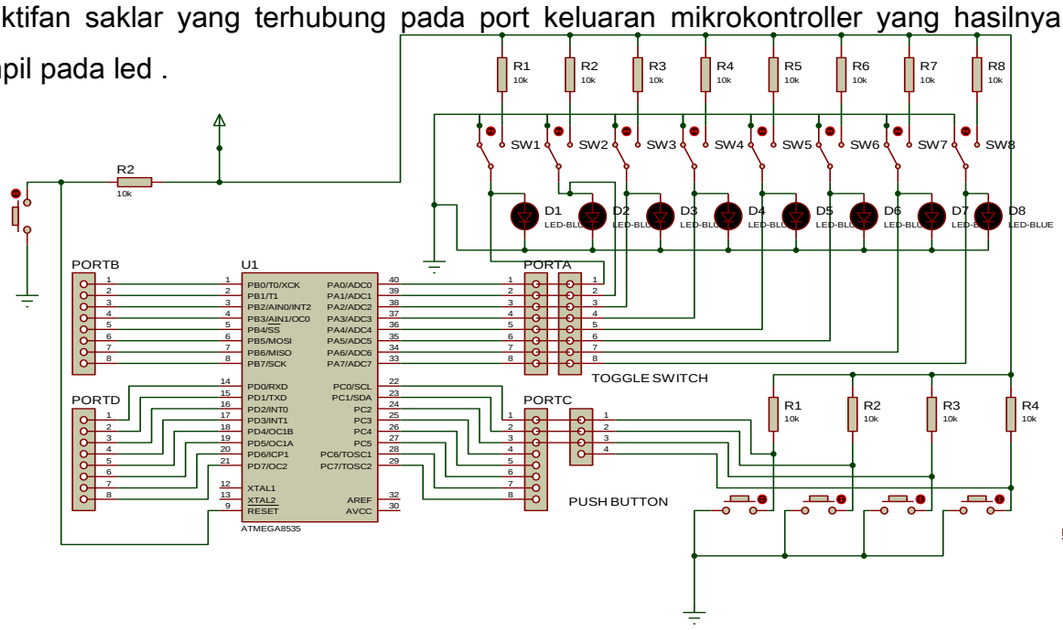


Gambar 2.1. Rangkaian saklar

Sedangkan saat tombol ditekan, maka baris dan kolom akan terhubung ke ground sehingga kondisi pada baris dan kolom tersebut akan menjadi low.

2.2. RANGKAIAN PEMBACAAN 8 TOGGLE SWITCH

Rangkaian pembacaan 8 buah saklar toggle adalah rangkaian untuk membaca pengaktifan saklar yang terhubung pada port keluaran mikrokontroller yang hasilnya tertampil pada led .



Gambar 2.2. Rangkaian aplikasi pembacaan 8 toggle switch dan push button

2.3. PEMROGRAMAN PEMBACAAN 8 BUAH SAKLAR TOGGLE

Setelah rangkaian tombol dibuat dan dihubungkan dengan port pararel mikrokontroller, maka sekarang saatnya Anda membuat program pembacaan tombol.

Program sebagai berikut ini

```
//-------------------------------------------------------
//Program Program pembacaan 8 buah toggle switch
//-------------------------------------------------------
#include <mega8535.h>
#include <delay.h>
void main(void)
{
DDRC=0x00;
DDRA=0xFF;
while(1)
{
PORTA = PINC;
}
}
```

Cara kerja program:

Pada program pembacaan 8 buah toggle switch, di perlukan deklarasi register untuk mikrokontroller jenis ATMEGA8535. Setelah mendeklarasi register, maka program akan masuk ke dalam program utama. Di dalam program utama, mikrokontroller akan membaca PORT C. Data dari PORT C akan dimasukan ke dalam variabel, Kemudian data yang ada di variabel tersebut akan dikeluarakan pada PORT A oleh mikrokontroller. Didalam program utama terdapat pernyataan while(1). Pernyataan itu berfungsi untuk melakukan *Looping* secara terus menerus.

2.4. PEMROGRAMAN PEMBACAAN TOMBOL

Setelah rangkaian tombol dibuat dan dihubungkan dengan port pararel mikrokontroller, maka sekarang saatnya Anda membuat program Program pembacaan tombol tunggal.

Program sebagai berikut ini

```
//-------------------------------------------------------
//Program membaca 1 tombol
//-------------------------------------------------------
#include <mega8535.h>
#include <delay.h>
void jalankiri()
{
char i;
volatile unsigned char dataLED=0x80;
DDRB=0xFF;
PORTB = 0;
```

```
for(i=0; i<8;i++)
{
dataLED= ((dataLED<<1) | (dataLED >>7));
PORTB=dataLED;
delay_ms(100);
}
}
void jalankanan()
{
char i;
volatile unsigned char dataLED=0x01;
DDRB=0xFF;
PORTB = 0;
for(i=0; i<8;i++)
{
dataLED= ((dataLED<<7) | (dataLED >>1));
PORTB=dataLED;
delay_ms(100);
}
}
void main(void)
{
DDRC=0x00;
while(1)
{
if (PINC.0==1)
{
jalankanan();
}
else
{
jalankiri();
}
}
}
```

Cara kerja program:

Pada program satu tombol, di perlukan deklarasi register untuk mikrokontroller jenis ATMEGA8535. Setelah mendeklarasi register, maka program akan masuk ke dalam program utama. Di dalam program utama, mikrokontroller akan membaca PORT C.0. Kemudian data tersebut akan dibandingkan untuk mengeluarakan data pada PORT B oleh mikrokontroller. Jika PORT C.0 berlogika rendah maka led pada PORT B akan bergeser ke kiri, jika port PORT C.0 berlogika rendah maka led pada PORT B akan bergeser ke kanan. Kemudian memanggil tunda 1 sekon Didalam program utama terdapat pernyataan while(1). Pernyataan itu berfungsi untuk melakukan *Looping* secara terus menerus.

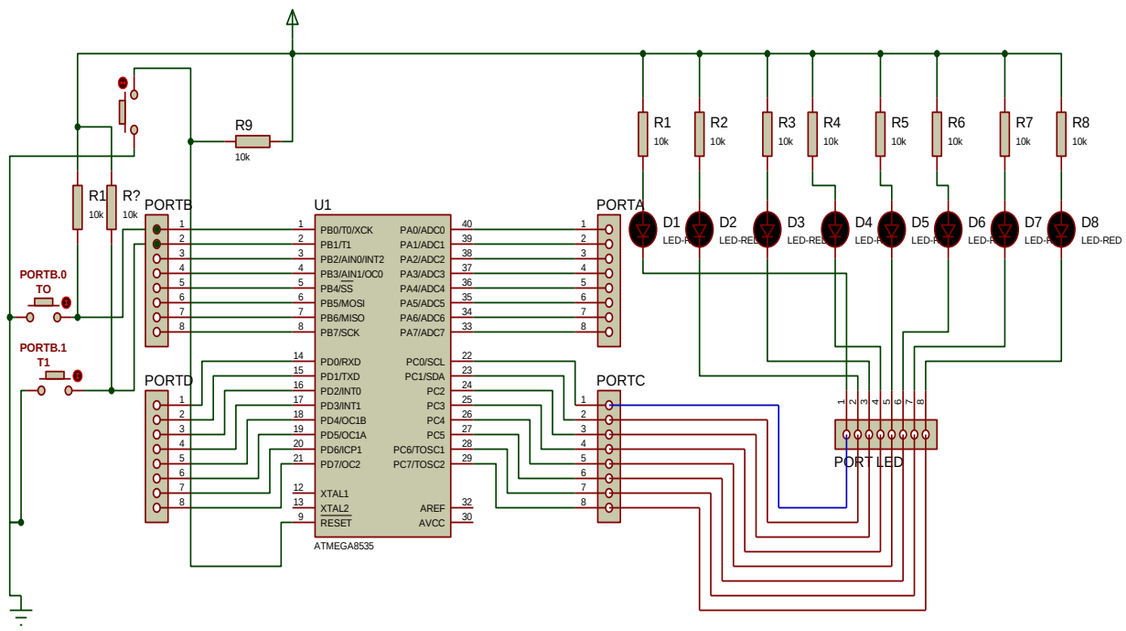
BAB III
TIMER DAN COUNTER

3.1. PENDAHULUAN

Timer dan Counter merupakan sarana input yang kurang dapat perhatian pemakai mikrokontroler, dengan sarana input ini mikrokontroler dengan mudah bisa dipakai untuk mengukur lebar pulsa, membangkitkan pulsa dengan lebar yang pasti. AVR ATMEGA8535 memiliki tiga buah *timer*, yaitu *Timer/Counter0* (8 bit), *Timer/Counter1* (16 bit), dan *Timer/Counter3* (16 bit).

3.2. RANGKAIAN MENCACAH COUNTER TIMER T0

Rangkaian minimum untuk counter melalui Port B.0 ditunjukan pada Gambar 3.1. Rangkaian tersebut menggunakan penampil led. Konfigurasi rangkaian LED yaitu *Common Anode* (CA) artinya untuk menghidupkan LED pada Port C, port C harus dikirim atau diberi logika '0'.



Gambar 3.1. Hasil pemasangan komponen-komponen mencacah counter T0 dan T1

3.3. PEMROGRAMAN MENCACAH COUNTER T0

Setelah rangkaian dibuat dan dihubungkan dengan port mikrokontroller, maka sekarang saatnya anda membuat program yang digunakan untuk mencacah. Program cacah menggunakan port B.0 pada mikrokontroller.

Program sebagai berikut ini

```
//-----------------------------------------------------
// Program MENCACAH COUNTER TIMER 0
//-----------------------------------------------------
#include <mega8535.h>
#include <delay.h>
#include <stdio.h>
unsigned char led,a;
void InisialisasiTIMER ();
void main (void)
{
DDRB=0x00;
DDRA= 0xff;
led=0x00;
InisialisasiTIMER();
while(1)
{
a = TCNT0;
if (a == 0x06)
{
led = PINA;
PORTA=~led;
TCNT0=0x00;
}
}
}
void InisialisasiTIMER ()
{
TCNT0=0x00;
TCCR0=0x07;
}
```

Cara kerja program:

Program mencacah counter T0 merupakan program untuk menghidupkan dan mematikan led dengan menekan satu tombol sebanyak 6x. Program ini, di perlukan deklarasi register untuk mikrokontroller jenis ATMEGA8535. Setelah mendeklarasi register, maka program akan mendeklasrasikan timer sebagai counter. Untuk mendeklarasikan timer sebagai counter maka register TCCR0 diisi dengan nilai 0x07. Tcnt0 = 0.Untuk menghapus isi dari register timer 0 maka register TCNT0 di beri nilai 0x00 Di dalam program utama, mikrokontroller akan membaca cacahan melalui PORTB.0. Cacahan tersebut akan di masukan kedalam register TCNT0, kemudian di masukan kedalam variabel. Nilai cacahan yang terdapat di dalam variabel tersebut akandibandingkan, pada saat nilai cacahan = 6 maka led akan menyala dan jika tombol ditekan lagi sebanyak 6x maka led akan mati. Didalam program utama terdapat pernyataan while(1). Pernyataan itu berfungsi untuk melakukan *Looping* secara terus menerus.

3.4. PEMROGRAMAN MENCACAH TIMER T0

Setelah rangkaian dibuat dan dihubungkan dengan port mikrokontroller, maka sekarang saatnya Anda membuat program yang digunakan untuk mencacah. Program cacah menggunakan timer pada mikrokontroller.

Program sebagai berikut ini

```
//-------------------------------------------------------
// Program MENCACAH TIMER T0
//-------------------------------------------------------
#include <mega8535.h>
#include <delay.h>
#include <stdio.h>
unsigned char led=0;
char a;
void InisialisasiTIMER ();
void main (void)
{
DDRB=0x00;
DDRD=0xFF;
PORTD=led;
InisialisasiTIMER();
led = 0x01;
while(1)
{
if (led == 0x80)
{
led = 0x01;
}
a = TCNT0;
if (a == 0xFE)
{
PORTD=led;
TCNT0=0x00;
led=led <<1;
}
}
}
void InisialisasiTIMER ()
{
TCNT0=0x00;
TCCR0=0x05;
}
```

Cara kerja program:

Pada Program mencacah Timer T0, di perlukan deklarasi register untuk mikrokontroller jenis ATMEGA8535. Setelah mendeklarasi register, maka program akan mendeklasrasikan timer sebagai counter. Program utama ini digunakan untuk menghitung banyaknya cacahan timer. Nilai dari cacahan tersebut akan di simpan di register TCNT0. Saat TCNT0 sama dengan 0xFE maka led yang di pasang pada PORTD akan bergeser satu digit. Dan sampai pada digit ke 8 maka data led akan dikembalikan ke posisi awal.

3.5. RANGKAIAN MENCACAH COUNTER TIMER T1

Rangkaian minimum untuk counter melalui Port B.1 ditunjukan pada Gambar 3.1. Rangkaian tersebut menggunakan penampil led. Konfigurasi rangkaian LED yaitu *Common Anode* (CA) artinya untuk menghidupkan LED pada Port D, port D harus dikirim atau diberi logika '0'.

3.6. PEMROGRAMAN MENCACAH COUNTER T1

Setelah rangkaian dibuat dan dihubungkan dengan port mikrokontroller, maka sekarang saatnya Anda membuat program yang digunakan untuk mencacah. Program cacah menggunakan port B.1 pada mikrokontroller.

Program sebagai berikut ini

```
//-----------------------------------------------------
// Program MENCACAH COUNTER TIMER 1
//-----------------------------------------------------
#include <mega8535.h>
#include <delay.h>
#include <stdio.h>
unsigned char led,a;
void InisialisasiTIMER ();
void main (void)
{
DDRD = 0xff;
led=0x00;
InisialisasiTIMER();
while(1)
{
a = TCNT1L + TCNT1H;
if (a == 0x06)
{
led = PIND;
PORTD=~led;
TCNT1L=0x00;
TCNT1H=0x00;
}
}
}
void InisialisasiTIMER ()
{
TCNT1L=0x00;
TCNT1H=0x00;
TCCR1A=0x00;
TCCR1B=0x07;
}
```

Cara kerja program:

Program mencacah counter T1 merupakan program untuk menghidupkan dan mematikan led dengan menekan satu tombol sebanyak 6x. Program ini, di perlukan deklarasi register untuk mikrokontroller jenis ATMEGA8535. Setelah mendeklarasi register, maka program akan mendeklasrasikan timer sebagai counter. Untuk mendeklarasikan timer sebagai counter maka register TCCR1 diisi dengan nilai 0x07.

Tcnt1 = 0.Untuk menghapus isi dari register timer 0 maka register TCNT1 di beri nilai 0x00 Di dalam program utama, mikrokontroller akan membaca cacahan melalui PORTB.1. Cacahan tersebut akan di masukan kedalam register TCNT1, kemudian di masukan kedalam variabel. Nilai cacahan yang terdapat di dalam variabel tersebut akan dibandingkan, pada saat nilai cacahan = 6 maka led akan menyala dan jika tombol di tekan lagi sebanyak 6x maka led akan mati. Didalam program utama terdapat pernyataan while(1). Pernyataan itu berfungsi untuk melakukan *Looping* secara terus menerus.

3.7. PEMROGRAMAN MENCACAH TIMER T1

Setelah rangkaian dibuat dan dihubungkan dengan port mikrokontroller, maka sekarang saatnya Anda membuat program yang digunakan untuk mencacah. Program cacah menggunakan timer pada mikrokontroller.

Program sebagai berikut ini

```
//-------------------------------------------------------
// Program MENCACAH TIMER T0
//-------------------------------------------------------
#include <mega8535.h>
#include <delay.h>
#include <stdio.h>
unsigned char led=0;
char a;
void InisialisasiTIMER ();
void main (void)
{
DDRB=0x00;
DDRD=0xFF;
PORTD=led;
InisialisasiTIMER();
led = 0x01;
while(1)
{
if (led == 0x80)
{
led = 0x01;
}
a = TCNT1L + TCNT1H;
if (a == 0xFE)
{
PORTD=led;
TCNT1L=0x00;
TCNT1H=0x00;
led=led <<1;
}
}
}
void InisialisasiTIMER ()
{
TCNT1L=0x00;
```

```
TCNT1H=0x00;
TCCR1A=0x00;
TCCR1B=0x05;
}
```

Cara kerja program:

Pada Program mencacah Timer T1, di perlukan deklarasi register untuk mikrokontroller jenis ATMEGA8535. Setelah mendeklarasi register, maka program akan mendeklasrasikan timer sebagai counter. Program utama ini digunakan untuk menghitung banyaknya cacahan timer. Nilai dari cacahan tersebut akan di simpan di register TCNT1L dan TCNT1H.. Saat TCNT1L + TCNT1H sama dengan 0xFE maka led yang di pasang pada PORT D akan bergeser satu digit. Dan sampai pada digit ke 8 maka data led akan dikembalikan ke posisi awal.

BAB IV
INTERUPSI MIKROKONTROLLER

4.1. PENDAHULUAN

Interupsi adalah suatu kejadian atau peristiwa yang menyebabkan mikrokontroler berhenti sejenak untuk melayani interupsi tersebut. Program yang dijalankan pada saat melayani interupsi disebut **Interrupt Service Routine**. Pada sistem mikrokontroler yang sedang menjalankan programnya, saat terjadi interupsi , program akan berhenti sesaat, melayani interupsi tersebut dengan menjalankan program yang berada pada alamat yang ditunjuk oleh vektor dari interupsi yang terjadi hingga selesai dan kembali meneruskan program yang terhenti oleh interupsi tadi. Pengetahuan mengenai interupsi tidak cukup hanya dibahas secara teori saja, diperlukan contoh program yang konkrit untuk memahami. ATMEGA8535 memiliki 21 buah sumber interupsi. Interupsi tersebut bekerja jika bit I pada Register status atau *Status Register* (SREG) dan bit pada masing-masing register bernilai 1.

4.2. RANGKAIAN INTERUPSI EKTERNAL

Rangkaian berikut digunakan untuk interupsi ekternal mikrokontroller. Rangkaian tersebut menggunakan interupsi eksternal 0, 1, dan 2 yang menggunakan tampilan LED yang dihubungkan pada Port A.

4.2.1. PEMROGRAMAN INTERUPSI EKTERNAL INT0

Setelah membuat rangkaian interupsi ekternal untuk menghidupkan LED, maka sekarang saatnya Anda membuat program yang digunakan untuk menghidupkan LED dengan menggunakan interupsi external 0.

Program sebagai berikut ini

```
//-----------------------------------------------------
//Program rutin interupsi eksternal 0
//-----------------------------------------------------
#include <mega8535.h>
#include <delay.h>
#include <stdio.h>
unsigned char dt=0x01;
void InisialisasiINT0();
void main (void)
{
DDRA=0xff; InisialisasiINT0();
#asm ("sei");
while(1)
```

```
{
PORTA=dt;
delay_ms(100);
dt=dt<<1;
if (dt==0) {dt=0x01;}
}
}
interrupt [EXT_INT0] void ext_int0_isr(void)
{
unsigned char rr=0;
while (rr<5)
{
PORTA=0x0f;
delay_ms(5);
PORTA=0xf0;
delay_ms(5);
++rr;
}
}
void InisialisasiINT0 ()
{
GICR|=0x80;
MCUCR=0x0C;
MCUCSR=0x00;
GIFR=0x80;
}
```

Cara kerja program:

Pada program rutin interupsi eksternal 0, di perlukan deklarasi register untuk mikrokontroller jenis ATMEGA8535. Setelah mendeklarasi register, maka program akan masuk ke dalam program utama. Program ini akan menginisialisasi interupsi ekternal 0 dan akan mengaktifkan interupsi ekternal 0. Sebelum terjadi interupsi eksternal mikrokontroller mengeluarkan data 0x01 pada port A. Kemudian data tersebut di geser ke kiri, sehingga led akan bergeser ke kanan. Saat terjadi interupsi maka mikrokontroller akan mengeluarkan data flip-flop pada port A

4.2.2. PEMROGRAMAN INTERUPSI EKTERNAL INT1

Setelah membuat rangkaian interupsi ekternal int 1, maka sekarang saatnya Anda membuat program yang digunakan untuk menghidupkan LED dengan menggunakan interupsi external int1

Program sebagai berikut ini

```
//-------------------------------------------------------
//Program rutin interupsi eksternal 1
//-------------------------------------------------------
#include <mega8535.h>
#include <delay.h>
#include <stdio.h>
unsigned char dt=0x01;
void InisialisasiINT1();
void main (void)
{
DDRA=0xff; InisialisasiINT1();
#asm ("sei");
```

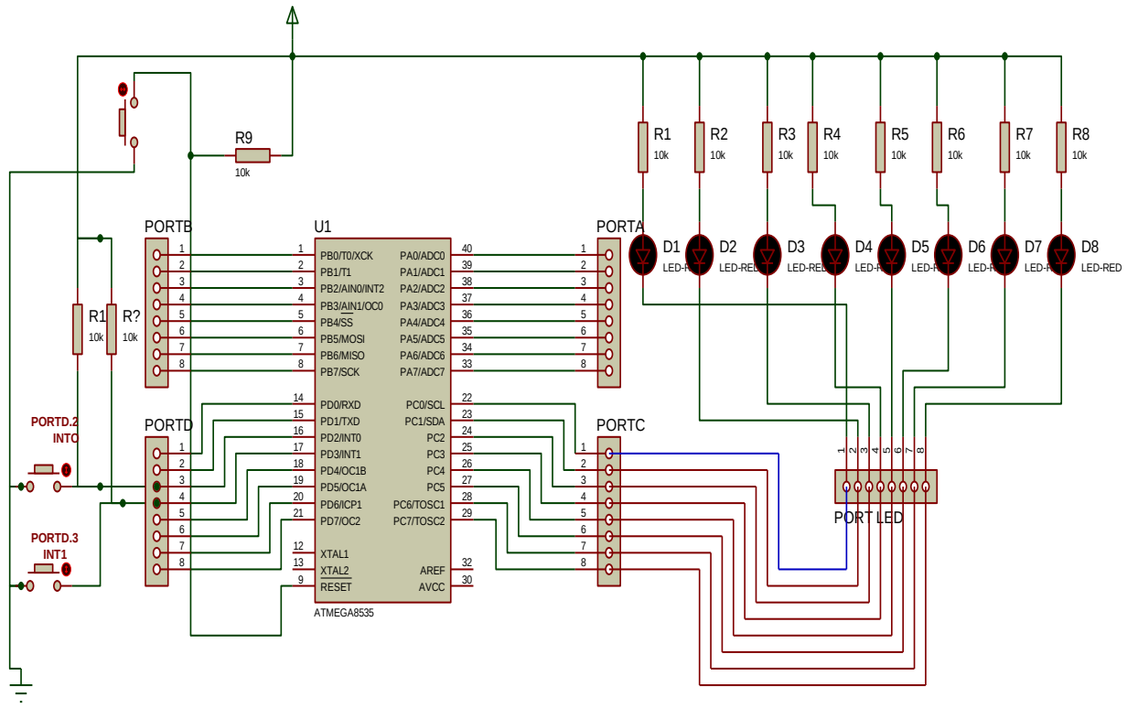
```
while(1)
{
PORTA=dt; delay_ms(100);
dt=dt<<1;
if (dt==0) {dt=0x01;}
};
}
interrupt [EXT_INT1] void ext_int1_isr(void)
{
unsigned char rr=0;
while (rr<5) {
PORTA=0x0f;
delay_ms(5);
PORTA=0xf0;
delay_ms(5);
++rr;
}
}
void InisialisasiINT1()
{
GICR|=0x80;
MCUCR=0x0C;
MCUCSR=0x00;
GIFR=0x80;
}
```

Cara kerja program:

Pada program rutin interupsi eksternal 1, di perlukan deklarasi register untuk mikrokontroller jenis ATMEGA8535. Setelah mendeklarasi register, maka program akan masuk ke dalam program utama. Program ini akan menginisialisasi interupsi ekternal 1 dan akan mengaktifkan interupsi ekternal 1. Sebelum terjadi interupsi eksternal mikrokontroller mengeluarkan data 0x01 pada port A. Kemudian data tersebut di geser ke kiri, sehingga led akan bergeser ke kanan. Saat terjadi interupsi maka mikrokontroller akan mengeluarkan data flip-flop pada port A

4.3. RANGKAIAN INTERUPSI TIMER MIKROKONTROLLER

Rangkaian berikut digunakan untuk interupsi ekternal mikrokontroller. Rangkaian tersebut menggunakan interupsi timer 0 dan 1 yang menggunakan tampilan LED yang dihubungkan pada Port D.



Gambar 4.1. Rangkaian interupsi timer mikrokontroller

4.3.1. PEMROGRAMAN INTERUPSI TIMER 0

Setelah membuat rangkaian interupsi timer untuk menghidupkan LED, maka sekarang saatnya Anda membuat program yang digunakan untuk menghidupkan LED dengan menggunakan interupsi timer 0.

Program sebagai berikut ini

```
//-----------------------------------------------------
// Program INTERUPSI TIMER 0
//-----------------------------------------------------
#include <mega8535.h>
#include <delay.h>
#include <stdio.h>
unsigned char led=0xfe;
void InisialisasiTIMER0();
void main (void)
{
DDRD=0xff; InisialisasiTIMER0();
#asm ("sei"); while(1);
}
interrupt [TIM0_OVF] void timer0_overflow(void)
{
TCNT0=0x00; led<<=1; led|=1;
```

```
if (led==0xff) led=0xfe; PORTD=led;
}
void InisialisasiTIMER0 ( )
{
TCNT0=0x00; TCCR0=0x05;
TIMSK=0x01; TIFR=0x01;
}
```

Cara kerja program:

Pada program rutin interupsi timer 0, di perlukan deklarasi register untuk mikrokontroller jenis ATMEGA8535. Setelah mendeklarasi register, maka program akan masuk ke dalam program utama. Program ini akan menginisialisasi interupsi timer 0 dan akan mengaktifkan interupsi timer 0. sebelum interupsi mikrokontroller akan menyalakan led, setelah interupsi led geser kanan.

4.3.2. PEMROGRAMAN INTERUPSI TIMER 1

Setelah membuat rangkaian interupsi timer untuk menghidupkan LED, maka sekarang saatnya Anda membuat program yang digunakan untuk menghidupkan LED dengan menggunakan interupsi timer 1.

Program sebagai berikut ini

```
//-----------------------------------------------------
// Program INTERUPSI TIMER 1
//-----------------------------------------------------
#include <mega8535.h>
#include <delay.h>
#include <stdio.h>
unsigned char led=0xfe;
void InisialisasiTIMER1();
void main (void)
{
DDRD=0xff; InisialisasiTIMER1();
#asm ("sei"); while(1);
}
interrupt [TIM1_OVF] void timer1_ovf_isr(void)
{
TCNT1L=0x00; TCNT1H=0x00;
led<<=1;
led|=1; delay_ms(100);
if (led==0xff) led=0xfe;
PORTD=led;
}
void InisialisasiTIMER1()
{
TCNT1L=0x00; TCNT1H=0x00; TCCR1A=0x00;
TCCR1B=0x01; TIMSK=0x04; TIFR=0x04;
}
```

Cara kerja program:

Pada program rutin interupsi timer 1, di perlukan deklarasi register untuk mikrokontroller jenis ATMEGA8535. Setelah mendeklarasi register, maka program akan masuk ke dalam program utama. Program ini akan menginisialisasi interupsi timer 1 dan akan

mengaktifkan interupsi timer 1. sebelum interupsi mikrokontroller akan menyalakan led, setelah interupsi led geser kanan.

4.3.3. PEMROGRAMAN INTERUPSI TIMER 2

Setelah membuat rangkaian interupsi timer untuk menghidupkan LED, maka sekarang saatnya Anda membuat program yang digunakan untuk menghidupkan LED dengan menggunakan interupsi timer 2.

Program sebagai berikut ini

```
//-----------------------------------------------------
// Program INTERUPSI TIMER 2
//-----------------------------------------------------
#include <mega8535.h>
#include <delay.h>
#include <stdio.h>
unsigned char led=0xfe;
void InisialisasiTIMER2();
void main (void)
{
DDRD=0xff; InisialisasiTIMER2();
#asm ("sei"); while(1);
}
interrupt [TIM2_OVF] void timer2_ovf_isr(void)
{
TCNT2=0x00; led<<=1; led|=1; delay_ms(100);
if (led==0xff) led=0xfe; PORTD=led;
}
void InisialisasiTIMER2()
{
TCCR2=0x05; TCNT2=0x00; TIMSK=0x40; TIFR=0x40;
}
```

Cara kerja program:

Pada program rutin interupsi timer 2, di perlukan deklarasi register untuk mikrokontroller jenis ATMEGA8535. Setelah mendeklarasi register, maka program akan masuk ke dalam program utama. Program ini akan menginisialisasi interupsi timer 2 dan akan mengaktifkan interupsi timer 2. sebelum interupsi mikrokontroller akan menyalakan led, setelah interupsi led geser kanan.

BAB V
LCD

5.1. PENDAHULUAN

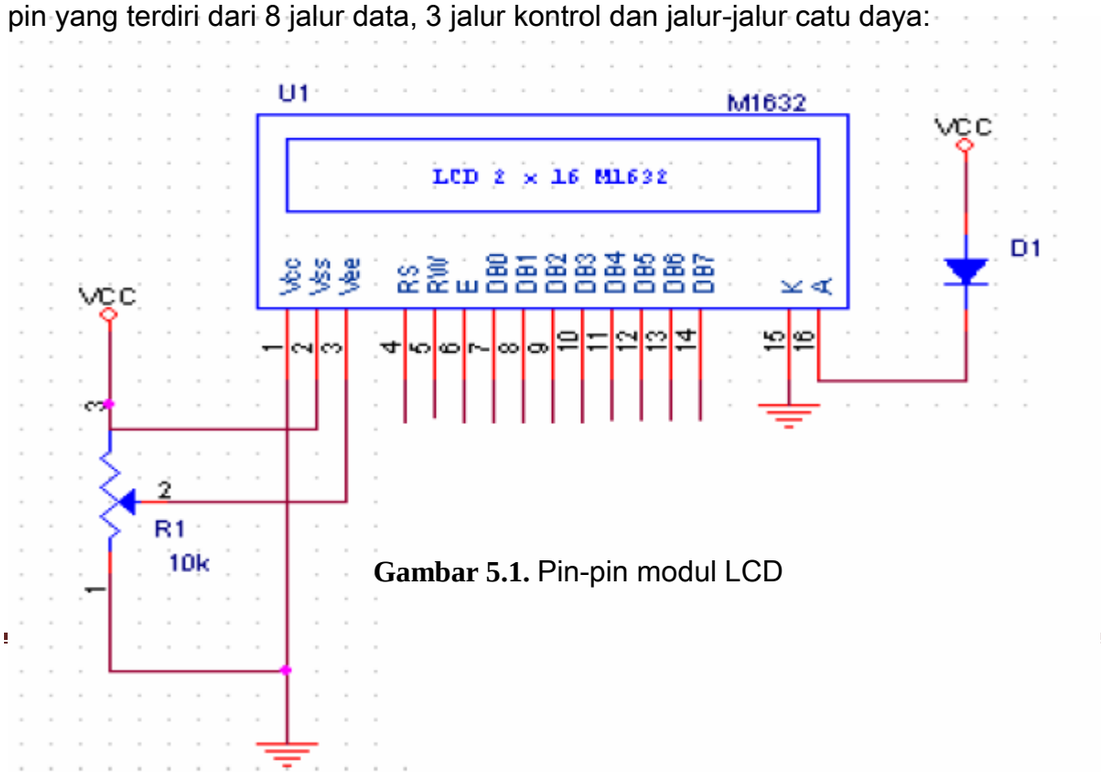
Kemampuan dari LCD untuk menampilkan tidak hanya angka-angka, tetapi juga huruf-huruf, kata-kata dan semua sarana simbol, lebih bagus dan serbaguna daripada penampil-penampil menggunakan 7-segment LED (*Light Emiting Diode*) yang sudah umum. Modul LCD mempunyai basic *interface* yang cukup baik, yang mana sesuai dengan minimum *system* 8031. Sesuai juga dengan keluarga mikrokontroler yang lain. Bentuk dan ukuran modul-modul berbasis karakter banyak ragamnya, salah satu variasi bentuk dan ukuran yang tersedia dan dipergunakan pada peralatan ini adalah 16x 2 karakter (panjang 16, baris 2, karakter 32) dan 16 pin.

5.2. M1632 MODULE LCD 16 X 2 BARIS (M1632)

M1632 adalah merupakan modul LCD dengan tampilan 16 x 2 baris dengan konsumsi daya yang rendah. Modul ini dilengkapi dengan mikrokontroler yang didisain khusus untuk mengendalikan LCD. Mikrokontroler HD44780 buatan Hitachi yang berfungsi sebagai pengendali LCD ini mempunyai CGROM (Character Generator Read Only Memory), CGRAM (Character Generator Random Access Memory) dan DDRAM (Display Data Random Access Memory).

5.3. FUNGSI PIN-PIN MODUL LCD

Modul LCD berukuran 16 karakter x 2 baris dengan fasilitas back lighting memiliki 16 pin yang terdiri dari 8 jalur data, 3 jalur kontrol dan jalur-jalur catu daya:



Gambar 5.1. Pin-pin modul LCD

1. Pin 1 dan 2

Merupakan sambungan catu daya, Vss, dan Vdd. Pin Vdd dihubungkan dengan tegangan positif catu daya, dan Vss pada 0 volt atau ground.

2. Pin 3

Merupakan pin kontrol Vcc yang digunakan untuk mengatur kontras display.

3. Pin 4

Merupakan *register select* (RS), masukan yang pertama dari tiga command control input. Dengan membuat RS menjadi *high*, data karakter dapat ditransfer dari dan menuju modulnya.

4. Pin 5

Read/Write (R/W). Untuk memfungsikan sebagai perintah *Write* maka R/W low atau menulis karakter ke modul.

5. Pin 6

Enable (E), input ini digunakan untuk transfer aktual dari perintahperintah atau karakter antara modul dengan hubungan data.

6. Pin 7 sampai 14

Pin 7 sampai 14 adalah delapan jalur data (D0 – D7) dimana data dapat ditransfer ke dan dari *display*.

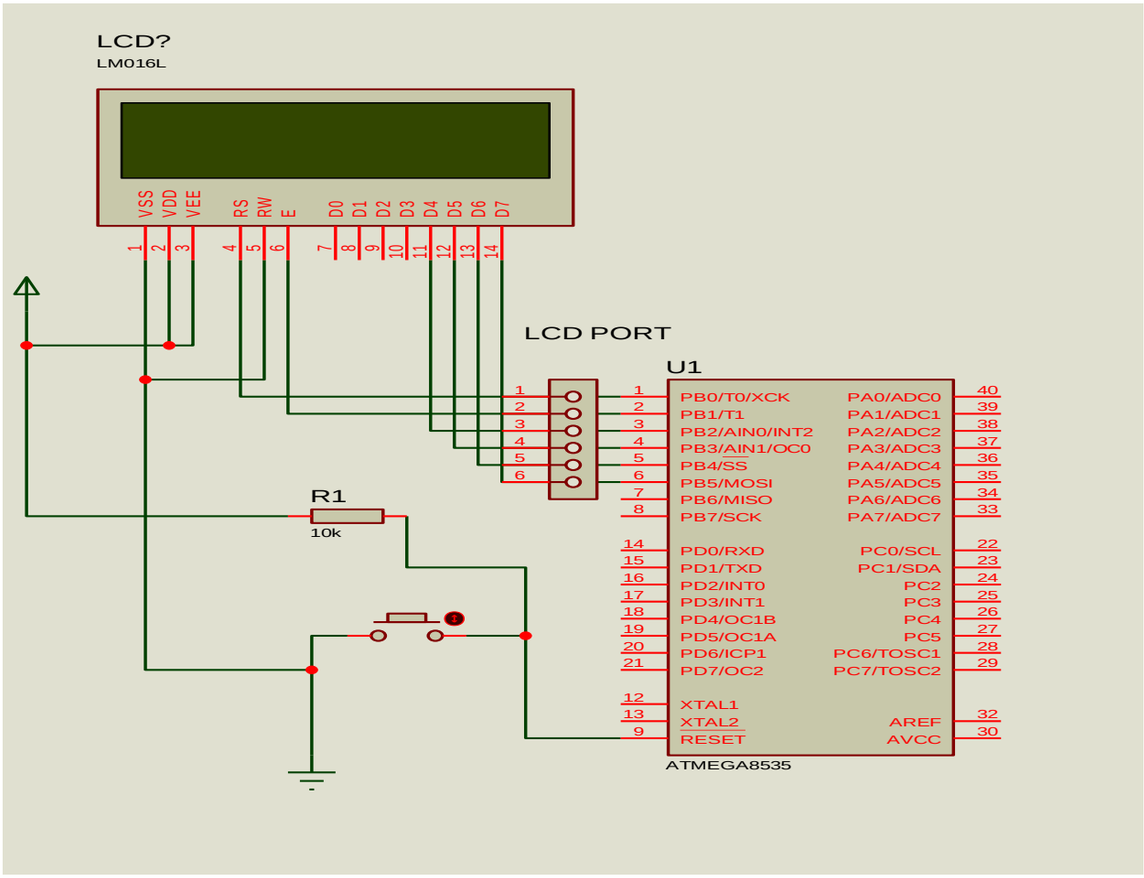
7. Pin 15 dan 16

Pin 15 atau A (+) mempunyai level DC +5 V berfungsi sebagai LED *backlight* + sedangkan pin 16 yaitu K (-) memiliki level 0 V

5.4. RANGKAIAN LCD

Rangkaian LCD adalah rangkaian untuk menghubungkan LCD secara langsung dari port keluaran mikrokontroller dengan input LCD.

Rangkaian LCD pada modul Trainer ATMEGA 8535 konfigurasi rangkaiannya hanya diprogram menggunakan BASCOM tidak sesuai dengan program codevision AVR, jadi pada praktikum / percobaan berikutnya yang berhubungan dengan display LCD menggunakan bahasa BASCOM



Gambar 5.2. Rangkaian LCD mikrokontroller

5.4.1. PEMROGRAMAN LCD

Setelah membuat rangkaian LCD, maka sekarang saatnya Anda membuat program LCD. Buka / jalankan software BOSCOM AVR, kemudian tuliskan Program sebagai berikut / copy paste :(jika sudah , lakukan compile)

‘Program LCD dengan BASCOM AVR’

```
$regfile = "8535def.dat"def
$crystal = 8000000
Config Lcdpin = Pin , Db4 = Portd.2 , Db5 = Portd.3 , Db6 = Portd.4 , Db7 = Portd.5 ,
 E = Portd.1 , Rs = Portd.0
Dim A As Byte
Config Lcd = 16 * 2
Cursor Off
Cls
Waitms 500
Do
For A = 1 To 70
Shiftlcd Left
Locate 1 , 1 : Lcd "SUDARYANTO KENTUS"
Waitms 500
Next
Cls
Locate 2 , 1 : Lcd "SMK COKROAMINOTO"
Waitms 500
```

Cara kerja program:

Pada program LCD, di perlukan deklarasi register mikrokontroller jenis ATMEGA8535. Setelah mendeklarasi register, maka program akan masuk ke dalam program utama. Program ini akan menginisialisasi LCD /mengkonfigurasi pin LCD serta pemilihan LCD 16 *2 dan akan menampilkan karakter dan tulisan di LCD. Tulisan pertama adalah Daryanto kentus yang akan ditampilkan pada baris pertama, dan akan ditampilkan di baris kedua berupa tulisan SMK COKROAMINOTO.

Tulis program LCD 2 ke BASCOM AVR sebagai berikut:

```
$regfile = "8535def.dat"def
$crystal = 8000000

Config Lcdpin = Pin , Db4 = Portd.2 , Db5 = Portd.3 , Db6 = Portd.4 , Db7 = Portd.5 , E
= Portd.1 , Rs = Portd.0
Dim A As Byte
Config Lcd = 16 * 2
Cursor Off
Cls
Waitms 500
Do
For A = 1 To 70
Shiftlcd Left
Locate 1 , 1 : Lcd "SUDARYANTO KENTUS"
Waitms 500
Next
Cls
Locate 2 , 1 : Lcd "SMK COKROAMINOTO"
Waitms 500
For A = 1 To 70
Shiftlcd Right
Locate 1 , 1 : Lcd " TEKNIK ELEKTRO"
Waitms 500
Next
Cls
Locate 2 , 1 : Lcd " MEMANG OK"
Waitms 500
Loop
End
```

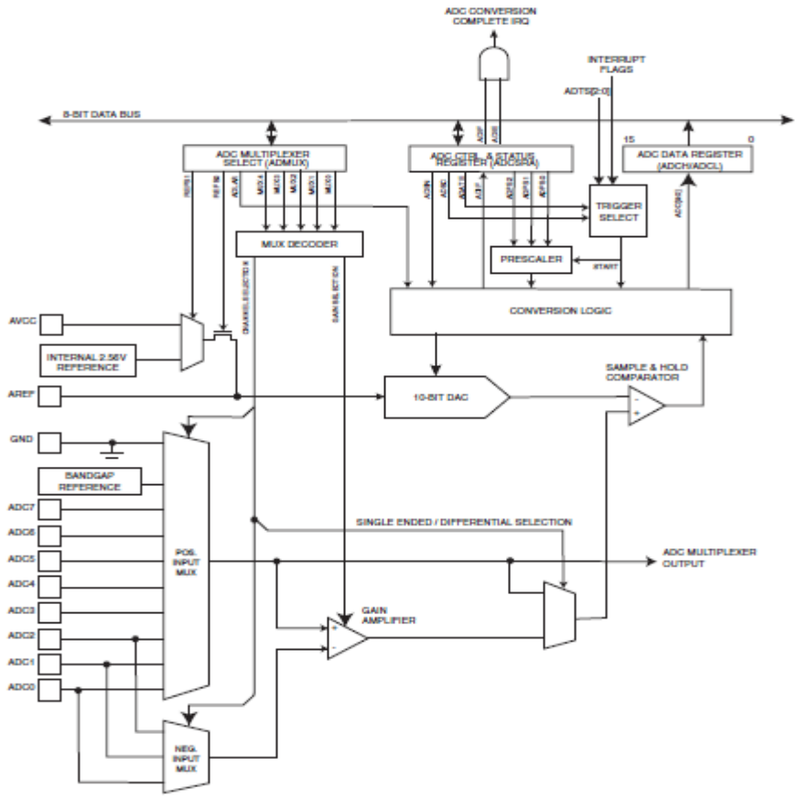
BAB VI
ANALOG TO DIGITAL CONVERTER MIKROKONTROLLER

6.1. PENDAHULUAN

Dalam dunia komputer, semua nilai tegangan dijadikan dalam bentuk digital, dan menggunakan sistem bilangan biner. Untuk itu dalam sistem ini, karena output dari sensor suhu berupa tegangan analog, maka diperlukan pengubah tegangan analog ke digital. ADC (*Analog to Digital Converter*) adalah suatu piranti yang digunakan untuk mengubah isyarat analog ke isyarat digital, rangkaian ini digunakan untuk mengubah isyarat analog dari sensor ke bentuk digital yang nantinya masuk ke komputer.

6.2. ADC ATMEGA8535

ATMEGA8535 merupakan tipe AVR yang dilengkapi dengan 8 saluran ADC internal dengan fidelitas 10 bit. Dalam mode operasinya, ADC ATMEGA8535 dapat dikonfigurasi, baik sebagai *single ended input* maupun pewaktuan, tegangan referensi, mode operasi, dan kemampuan filter derau yang amat fleksibel sehingga dapat dengan mudah disesuaikan dengan kebutuhan dari ADC itu sendiri.



Gambar 6.1.

Diagram Blok *ADC*

Proses inisialisasi ADC meliputi proses penentuan clock, tegangan referensi, format output data, dan mode pembacaan. Register yang perlu diset nilainya adalah ADMUX *(ADC Multiplexer Selection Register)*, ADCSRA *(ADC Control and Status Register A)*, dan SFIOR *(special Function IO Register)*. ADMUX merupakan register 8 bit yang berfungsi menentukan tegangan referensi ADC, format data output, dan saluran ADC yang digunakan. Konfigurasi register ADMUX pada Gambar 6.2.

REF1	REF0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0

Gambar 6.2. Register ADMUX

Bit penyusunnya sebagai berikut:

a. REF[1..0] merupakan bit pengatur tegangan referensi ADC ATMega8535. Memeiliki Nilai Awal 00 sehingga referensi tegangan berasal dari pin AREF. Detail nilai yang lain dapat dilihat pada tabel 6.1.

Tabel 6.1. Pemilihan Mode Tegangan Referensi *ADC*

REF1	REF0	Mode Tegangan Referensi
0	0	Berasal dari pin AREF
0	1	Berasal dari pin AVCC
1	0	Tidak dipergunakan
1	1	Berasal dari tegangan referensi internal 2,56 V

b. ADLAR merupakan bit pemilih mode data keluaran ADC. Bernilai awal 0, sehingga 2 bit tertinggi data hasil konversinya berada di register ADCH dan 8 bit sisanya berada di register ADCL, seperti dalam tabel 6.3. Apabila bernilai 1, maka hasilnya pada tabel

–	–	–	–	–	–	ADC9	ADC8	ADCH
ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0	ADCL

Gambar 6.3. Format Data *ADC* dengan ADLAR=0

ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADCH
ADC1	ADC0	-	-	-	-	-	-	ADCL

Gambar 6.4. Format Data *ADC* dengan ADLAR=1

c. MUX[4..0] merupakan bit pemilih saluran pembacaan *ADC*. Bernilai awal 00000. Untuk *mode single ended input*, MUX[4..0] bernilai dari 00000 hingga 00111

MUX4..0	Single Ended Input	Pos Differential Input	Neg Differential Input	Gain
00000	ADC0			
00001	ADC1			
00010	ADC2			
00011	ADC3			
00100	ADC4	N/A		
00101	ADC5			
00110	ADC6			
00111	ADC7			
01000		ADC0	ADC0	10x
01001		ADC1	ADC0	10x
01010		ADC0	ADC0	200x
01011		ADC1	ADC0	200x
01100		ADC2	ADC2	10x
01101		ADC3	ADC2	10x
01110		ADC2	ADC2	200x
01111		ADC3	ADC2	200x
10000		ADC0	ADC1	1x
10001		ADC1	ADC1	1x
10010	N/A	ADC2	ADC1	1x
10011		ADC3	ADC1	1x
10100		ADC4	ADC1	1x
10101		ADC5	ADC1	1x
10110		ADC6	ADC1	1x
10111		ADC7	ADC1	1x
11000		ADC0	ADC2	1x
11001		ADC1	ADC2	1x
11010		ADC2	ADC2	1x
11011		ADC3	ADC2	1x
11100		ADC4	ADC2	1x

ADCSRA merupakan register 8 bit yang berfungsi melakukan manajemen sinyal kontrol dan status dari *ADC*. Memiliki susunan dalam tabel 6.5.

ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0

Gambar 6.5. Register ADCSRA

Bit penyusunnya sebagai berikut:

a. ADEN merupakan bit pengatur aktivasi *ADC*. Bernilai awal 0. Jika bernilai 1, maka ADC aktif.

b. ADSC merupakan bit penanda mulainya konversi *ADC*. Bernilai awal 0 selama konversi *ADC* akan bernilai 1, sedangkan jika konversi selesai, akan bernilai 0.

c. ADATE merupakan bit pengatur aktivasi picu otomatis operasi *ADC*. Bernilai awal 0, jika bernilai1 maka konversi ADC akan dimulai pada saat transisi positif dari sinyal picu yang diplih. Pemiliha sinyal picu menggunakan bit ADTS pada register SFIOR.

d. ADIF merupakan bit penanda akhir suatu konversi *ADC*. Bernilai awal 0. Jika bernilai 1, maka konversi *ADC* pada saluran telah selesai dan data siap diakses.

e. ADIE merupakan bit pengatur aktivasi interupsi yang berhubungan dengan akhir konversi *ADC*. Bernilai awal 0. Jika berniali 1 dan jika konversi *ADC* telah selesai, sebuah interupsi akan dieksekusi.

f. ADPS[2..0] merupakan bit pengatur clock *ADC*. Bernilai awal 000. Detail nilai bit dalam tabel 6.6.

Tabel 6.6. Konfigurasi *Prescaler ADC*

ADPS2	ADPS1	ADPS0	Faktor Pembagi
0	0	0	2
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

SFIOR merupakan register 8 bit pengatur sumber picu konversi *ADC*, apakah dari picu eksternal atau dari picu internal. Susunannya dalam tabel 6.7.

ADTS2	ADTS1	ADTS0	–	ACME	PUD	PSR2	PSR10	SFIOR

Gambar 6.7. Register SFIOR

ADTS[2..0] merupakan bit pengatur picu eksternal operasi *ADC*. Hanya berfungsi jika bit ADATE pada register ADCSRA bernilai 1. Bernilai awal 000 sehingga *ADC* bekerja pada mode *free running* dan tidak ada interupsi yang akan dihasilkan. Detail nilai ADTS[2..0] dapat dilihat pada tabel 6.8 Untuk Operasi ADC, bit ACME, PUD, PSR2, dan PSR10 tidak diaktifkan.

Tabel 6.8. Pemilihan Sumber Picu *ADC*

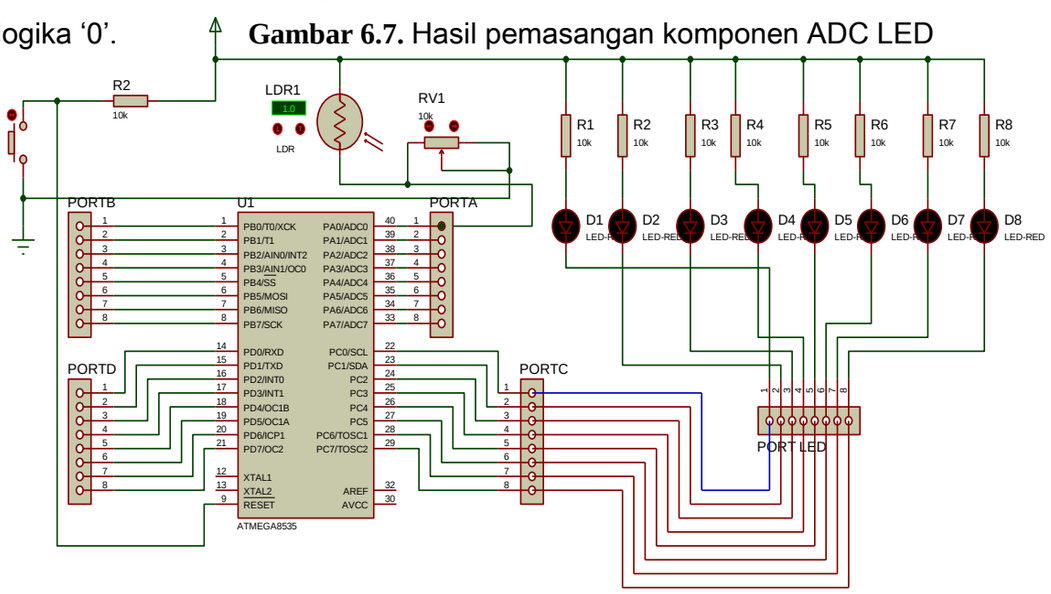
ADTS2	ADTS1	ADTS0	Sumber Picu
0	0	0	Mode *Free Running*
0	0	1	Komparator *Analog*
0	1	0	Interrupt External *Request 0*
0	1	1	*Timer/Counter0 Compare Match*
1	0	0	*Timer/Counter0 Overflow*
1	0	1	*Timer/Counter1 Compare Match B*
1	1	0	*Timer/Counter1 Overflow*
1	1	1	*Timer/Counter1 Capture Event*

Dalam proses pembacaan hasil konversi ADC, dilakukan pengecekan terhadap bit ADIF *(ADC Interupt Flag)* pada register ADCSRA. ADIF akan benilai satu jika konversi sebuah saluran *ADC* telah selesai dilakukan dan data hasil konversi siap untuk diambil, dan demikian sebaliknya. Data disimpan dalam dua buah register, yaitu ADCH dan ADCL.

6.3. RANGKAIAN ADC ATMEGA DENGAN LED

Rangkaian minimum untuk membaca ADC dengan tempilan LED ditunjukan pada Gambar 6.7 yang perlu diperhatikan adalah konfigurasi rangkaian LED yaitu *Common Anode* (CA) artinya untuk menghidupkan LED pada Port D, port D harus dikirim atau diberi logika ‘0’.

Gambar 6.7. Hasil pemasangan komponen ADC LED



6.4. PEMROGRAMAN ADC ATMEGA8535

Setelah rangkaian adc mikrokontroller ATMEGA8535 dibuat, maka sekarang saatnya Anda membuat program yang digunakan untuk membaca ADC dari sensor LDR dan menampilkan data ADC sensor LDR menggunakan LED yang terhubung pada PORTD yang konfigurasi rangkaian LED yaitu *Common Anode* (CA).

Ketikan Program Bahasa C sebagai berikut :

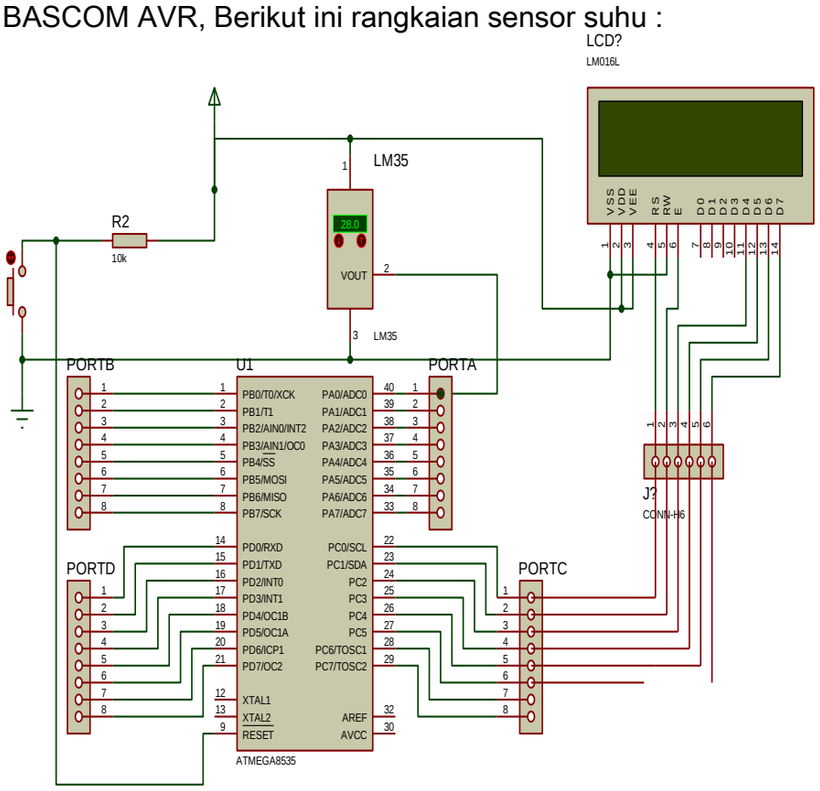
```
//-----------------------------------------------------
//Program ADC LED
//-----------------------------------------------------
#include <mega8535.h>
#include <stdio.h>
#include <delay.h>
unsigned int data_adc;
int sinar;
#define ADC_VREF_TYPE 0x60
unsigned char read_adc(unsigned char adc_input)
{
ADMUX=adc_input|ADC_VREF_TYPE;
ADCSRA|=0x40;
while ((ADCSRA & 0x10)==0);
ADCSRA|=0x10;
return ADCH;
}
void main(void)
{
DDRD = 0xFF;
ADMUX=ADC_VREF_TYPE;
ADCSRA=0x87;
SFIOR&=0xEF;
while (1)
{
data_adc=read_adc(0);
sinar=~data_adc;
PORTD = sinar;
}
}
```

Cara kerja program:

Pada program ADC LED, di perlukan deklarasi register untuk mikrokontroller jenis ATMEGA8535. Setelah mendeklarasi register, maka program akan masuk ke dalam program utama. Program ini akan membaca adc 0 dari data tegangan output sensor LDR kemudian data adc akan ditampilkan dengan LED yang konfigurasi rangkaian LED yaitu *Common Anode* (CA). Sintac DDRD = 0xFF merupakan ungkapan untuk mendeklarasikan PORT D sebagai output. Sintac ini ADCSRA=0x87; SFIOR&=0xEF berfungsi untuk mengisi register ADCSRA dan register SFIOR. data_adc=read_adc(0)merupakan ungkapan untuk mendapatkan nilai adc 0. Sintac sinar=~data_adc merupakan ungkapan untuk membalik data adc, karena adc akan dikeluarkan melalui LED yang konfigurasinya rangkaian LED yaitu *Common Anode* (CA), sehingga data yang ditampilkan akan sama dengan nyalanya LED.

6.5. RANGKAIAN ADC ATMEGA DENGAN LCD

Rangkaian minimum untuk membaca ADC dengan tempilan LCD, untuk tampilan LCD sudah dibahas sebelumnya bahwa konfigurasi LCD hanya bisa dipakai menggunakan bahasa BASCOM jadi program pembacaan adc dengan tampilan LCD akan ditulis dalam BASCOM AVR, Berikut ini rangkaian sensor suhu :



Gambar 6.8. Hasil pemasangan komponen ADC LCD SENSOR SUHU

6.6. PEMROGRAMAN ADC ATMEGA8535 DENGAN LCD

Setelah rangkaian adc mikrokontroller ATMEGA8535 dibuat dan dihubungkan dengan LCD, maka sekarang saatnya Anda membuat program yang digunakan untuk membaca ADC SENSOR SUHU ATMEGA8535 dan ditampilkan menggunakan LCD. Buka program BASCOM AVR dan tulis program berikut ini;

```
'-----------------------------------------
'Thermometer Digital
'-----------------------------------------
$regfile = "m8535.dat"
'Jika menggunakan ATMega8535 maka diganti dengan "m8535.dat"
$crystal = 12000000
'=========================
Config Lcdpin = Pin , E = Portc.1 , Rs = Portc.0 ,
Config Lcdpin = Pin , Db4 = Portc.2 , Db5 = Portc.3 , Db6 = Portc.4 , Db7 = Portc.5
Config Lcd = 16 * 2
Config Adc = Single , Prescaler = Auto , Reference = Avcc
Start Adc
'-------------------------
Dim Suhu_ref As Word
Dim Suhu As Word
Dim A As Byte
```

```
'-------------------------
Deflcdchar 0 , 12 , 18 , 18 , 12 , 32 , 32 , 32 , 32
'-------------------------
Cls
Cursor Off
'-------------------------
Do
Suhu_ref = Getadc(0)
Suhu = Suhu_ref * 5
Suhu = Suhu / 10
For A = 1 To 70
Shiftlcd Right
Locate 1 , 1 : Lcd "THERMOMETER DIGITAL"
Waitms 250
Next
For A = 1 To 80
Shiftlcd Right
Locate 2 , 1 : Lcd "BY YOSEF"
Waitms 250
Next
Cls
Locate 1 , 1
Lcd "SUHU TERDETEKSI "
Locate 2 , 1
Lcd "Suhu="
Locate 2 , 6
Lcd "   "
Locate 2 , 6
Lcd Suhu
Locate 2 , 9
Lcd Chr(0)
Locate 2 , 10
Lcd "C"
Locate 2 , 11
Lcd "      "
Wait 5
Loop
'--------------- 'end
```

Cara kerja program:

Pada program ADC LCD, di perlukan deklarasi register untuk mikrokontroller jenis ATMEGA8535. Setelah mendeklarasi register, maka program akan masuk ke dalam program utama. Program ini akan membaca data adc 0 darai sensor LM35 datanya akan ditampilkan dengan LCD. Configurasi ADC Single , prescale auto dan referensi = AVCC be rfungsi untuk mengisi register pemilihan adc tungal dan referensi tegangan dari AVCC. suhu_ref=getadc(0)merupakan ungkapan untuk mendapatkan nilai adc 0. Sintac suhu=suhu_ref*5 merupakan ungkapan untuk menyimpan data referensi adc. Kemudian data tersebut akan ditampilkan melalui LCD.

BAB VII
PWM ATMEGA8535

7.1. PENDAHULUAN

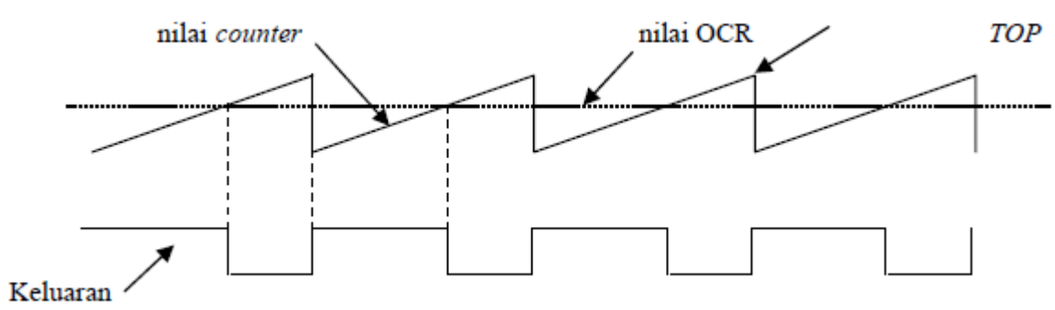
PWM (*Pulse Width Modulation*) dapat digunakan untuk mengatur kecepatan motor, yaitu dengan cara mengatur lebar pulsa (waktu ON) dari tegangan sumbernya (tegangan DC). Perbandingan antara waktu ON dan waktu OFF disebut *duty cycle* (siklus kerja). Semakin besar siklus kerjanya, akan semakin besar pula keluaran yang dihasilkan, sehingga kecepatan motor akan semakin besar. Pembangkitan sinyal PWM dengan mikrokontroler memiliki beberapa keuntungan, seperti teknik pemrograman yang sederhana, dan rangkaian listrik menjadi sederhana. Mikrokontroler AVR ATMEGA8535 dapat digunakan sebagai pembangkit gelombang PWM. Mikrokontroler AVR ATMEGA8535 mempunyai PWM yang telah terintegrasi dalam *chip*. Keluaran dari PWM tersebut terdapat pada pin 15 (OC1). Untuk menjalankan program PWM, diperlukan 3 unit register *timer*, yaitu:

a. *Timer/Counter Control Register* (TCCR), untuk menentukan mode PWM.

b. *Timer/Counter Register* (TCNT), digunakan untuk menentukan modulasi frekuensinya.

c. *Output Compare Register* (OCR), untuk menentukan nilai siklus kerjanya.

Dalam mikrokontroler ATMEGA8535, terdapat beberapa mode PWM. Mode PWM yang akan dibahas adalah mode *Fast PWM*, karena dalam perancangan sistem robot ini menggunakan mode *Fast PWM*. Pada mode *Fast PWM*, semakin besar nilai OCR, maka akan semakin besar pula siklus kerja yang dihasilkan. Keluaran PWM akan berlogika tinggi setelah nilai *TOP* tercapai sampai nilai OCR tercapai dan kemudian akan berlogika rendah sampai nilai *TOP* tercapai kembali. Prinsip kerja dari *Fast* PWM dapat dilihat pada Gambar7.1.



Gambar 7.1. Prinsip Kerja Mode *Fast PWM*

Untuk menghitung siklus kerja digunakan rumus:

$$D = \frac{OCR}{1+TOP} x100\% \quad \text{.....}$$

Untuk menentukan frekuensi PWM dihitung dengan rumus:

$$f_{PWM} = \frac{f_{clock}}{N(1+TOP)} \quad \text{.....}$$

Sedangkan untuk menentukan resolusi PWM digunakan rumus:

$$R_{PWM} = \frac{\log(TOP+1)}{\log 2} \quad \text{.....}$$

keterangan:

N adalah faktor *prescaler* (1, 8, 64, 256, atau 1024), dan

TOP adalah nilai tertinggi dari pengaturan *counter*.

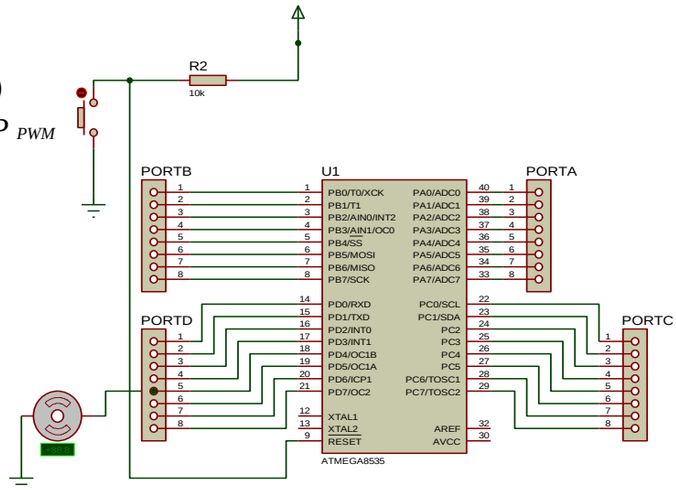
7.2. RANGKAIAN PWM MIKROKONTROLLER

Rangkaian minimum untuk pwm melalui Port D.4 dan Port D.5 ditunjukan pada Gambar 7.2. Rangkaian tersebut menggunakan diver motor dc yaitu transistor. Rangkaian driver tersebut akan di hubungkan dengan pin D.4 dan pin D.5.

log 2

log(+1)

$R = TOP_{PWM}$



Gambar 7.2. Hasil pemasangan komponen rangkaian minimum untuk pwm

7.3. PEMROGRAMAN PWM MIKROKONTROLLER

Setelah rangkaian dibuat dan dihubungkan dengan port mikrokontroller, maka sekarang saatnya Anda membuat program yang digunakan untuk mengatur putaran motor dc.

Program sebagai berikut ini

```
//-------------------------------------------------------
//Program Bab 10.1. PWM
//-------------------------------------------------------
#include <stdio.h>
#include <mega8535.h>
#include <delay.h>
void InisialisasiPWM();
int data1;
int data2;
void main (void)
{
InisialisasiPWM();
while(1)
{
data1 = 50;
data2 = 1024;
OCR1A=data1;
OCR1B=data2;
TIFR=0;
}
}
void InisialisasiPWM()
{
DDRD=0xff;
TCCR1A=0xa3;
TCCR1B=0x0b;
TCNT1=0x0000;
}
```

Cara kerja program:

Pada program ini perlukan deklarasi register untuk mikrokontroller jenisATMEGA8535. Setelah mendeklarasi register, maka program akan masuk ke dalam program utama. Program utama ini digunakan untuk mengendalikan putaran dua buah motor dengan dua PWM. Dengan PWM 50 maka putaran motor tidak terlalu cepat dan dengan PWM 1024 maka putaran motor akan sepat. Jadi untuk mendapatkan putaran motor yang sangat cepat maka PWM yang digunakan sangat tinggi dan untuk mendapatkan putaran sangat pelan maka PWM yang digunakan sangat rendah

7.3. PEMROGRAMAN PWM DAN TAMPILAN LCD

Hubungkan rangkaian motor dc dengan port PWM out mikrokontroller hubungkan port LCD dengan Port mikro sesuai program, kemudian buat program yang digunakan untuk mengatur putaran motor dc yang ditampilkan LCD menggunakan bahasa basic pada BASCOM AVR.

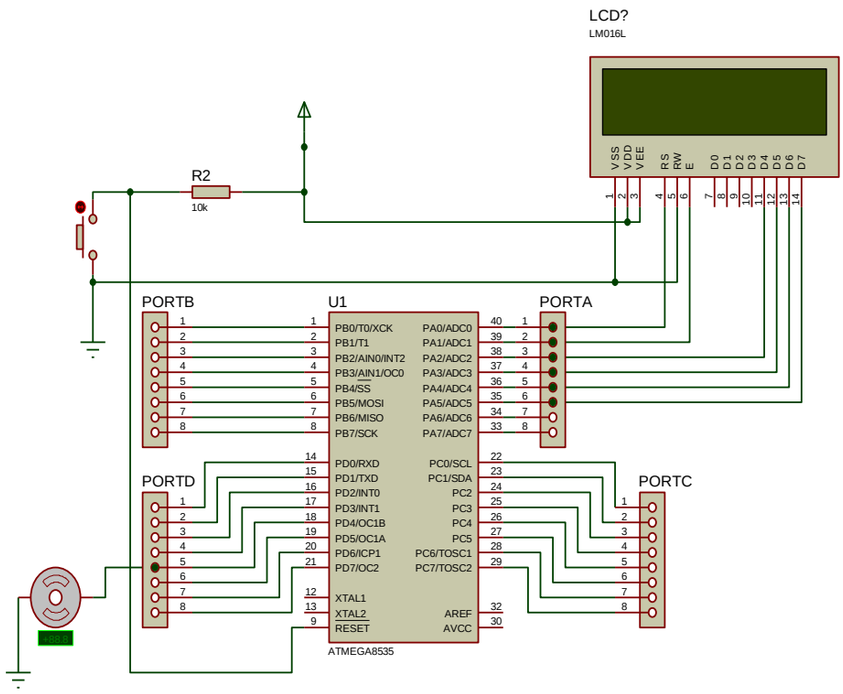
Ketik program sebagai berikut ini:

```
'-------------------------
'Pengatur Kecepatan Motor DC
'-------------------------
$regfile = "m8535.dat"
$crystal = 12000000
'-------------------------SET_PENGATURAN PUTARAN
Config Timer1 = Pwm , Pwm = 8 , Compare A Pwm = Clear Down , Prescale = 8
Config Lcdpin = Pin , E = Portc.1 , Rs = Portc.0 ,
Config Lcdpin = Pin , Db4 = Portc.2 , Db5 = Portc.3 , Db6 = Portc.4 , Db7 = Portc.5
Config Lcd = 16 * 2
'-------------------------SET_TOMBOL
Pwm1a = 20
'----------
Ddrb.0 = 0
Portb.0 = 1
Ddrb.1 = 0
Portb.1 = 1
'-------------------------
Dim Putar As Integer
Dim Pwm_ref As Word
Dim Pwm As Word

'-------------------------
Cls
Cursor Off
```

```
'=========================================================
'                    """""MULAI"""""
'=========================================================
Putar = 1
Do
Pwm = Pwm1a
'---------------------------
Locate 1 , 1
Lcd "KECEPATAN MOTOR"
Locate 2 , 1
Lcd "pwm="
Locate 2 , 6
Lcd "   "
Locate 2 , 6
Lcd Pwm
If Pinb.0 = 0 Then
Waitms 200
Putar = Putar + 1
End If
If Pinb.1 = 0 Then
Waitms 200
Putar = Putar - 1
End If
If Putar > 7 Then
Putar = 7
End If
If Putar < 1 Then
Putar = 1
End If
If Putar = 7 Then
Pwm1a = 225
End If
If Putar = 6 Then
Pwm1a = 200
End If
If Putar = 5 Then
Pwm1a = 110
End If
```

```
If Putar = 4 Then
Pwm1a = 90
End If
If Putar = 3 Then
Pwm1a = 70
End If
If Putar = 2 Then
Pwm1a = 50
End If
If Putar = 1 Then
Pwm1a = 0
End If
Waitms 300
Loop
'---------------------------- end
```



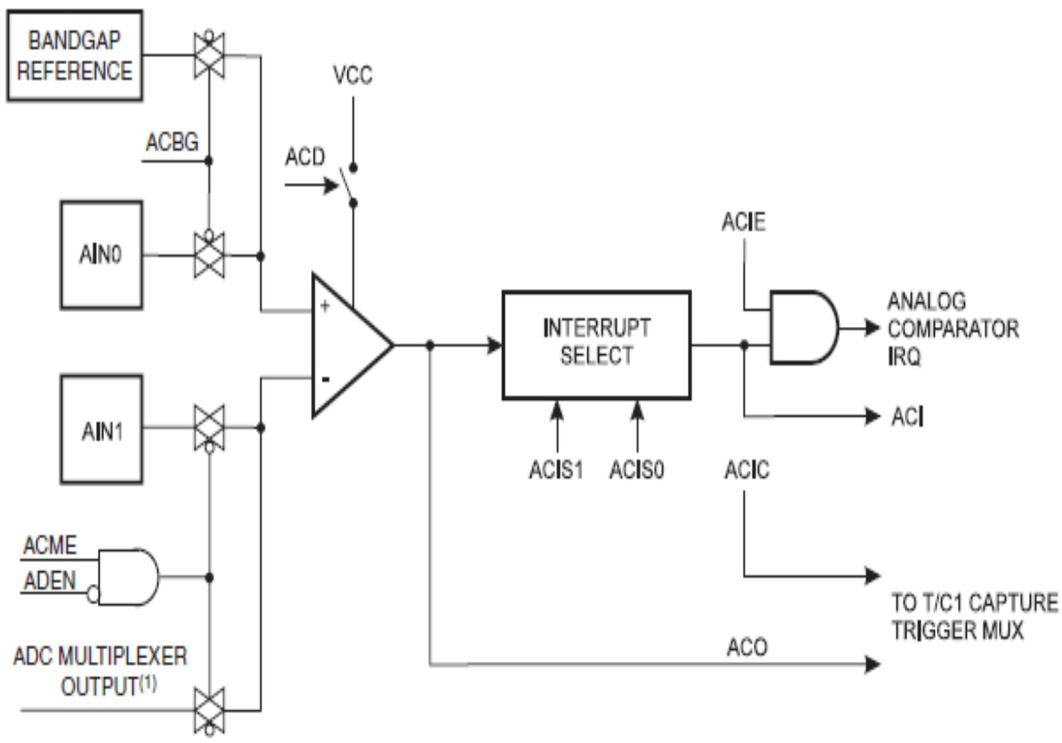
BAB VIII
KOMPARATOR ATMEGA8535

8.1. PENDAHULUAN

Komparator analog merupakan salah satu fitur pada ATMEGA8535. Fitur ini langsung membandingkan 2 input analog. Karena input analog adalah fungsi altenatif dari PORT B (PORTB.2 dan PORTB.3) maka PORTB.2 dan PORTB.3 harus kita set sebagai input dengan menonaktifkan R-*pullup* internal.

Komparator analog memiliki dua tahap yaitu:

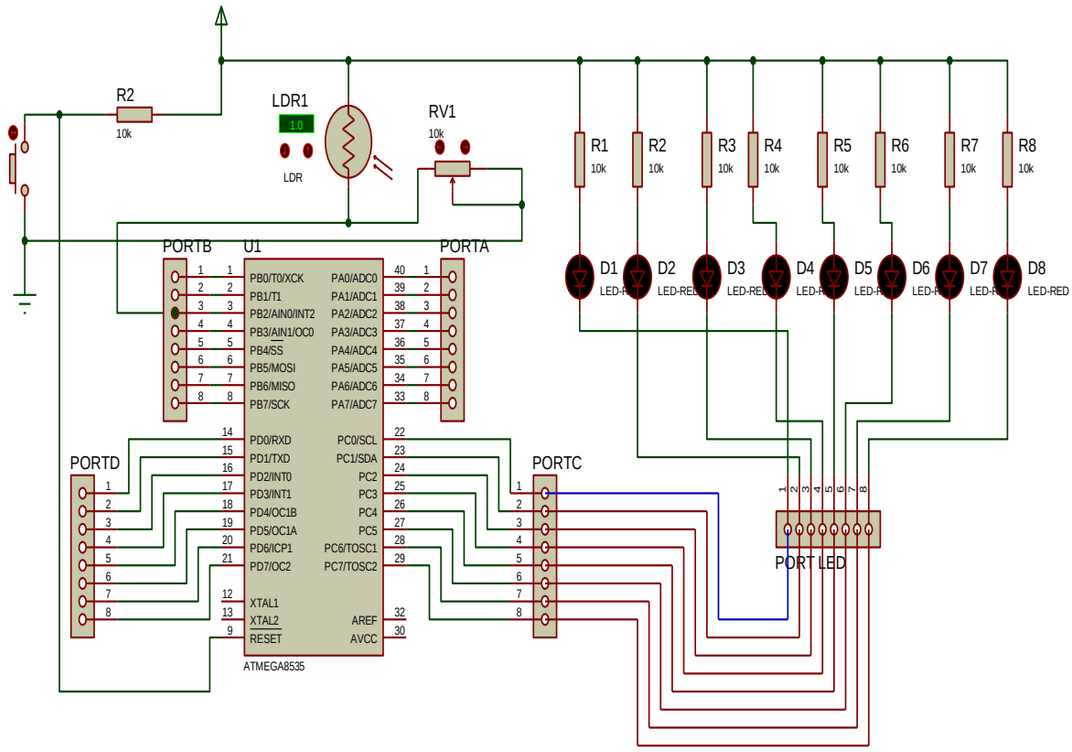
- Tahap pertama adalah komparator membandingkan input analog 0(AIN0) dan input analog 1 (ANI1)
- Tahap kedua adalahdari output komparator analog tersebut menuju ke logika flag interupsi (ACL)



Gambar 8.1. Blok diagram komparator analog

8.2. RANGKAIAN KOMPARATOR

Rangkaian komparator adalah rangkaian untuk membandingkan tegangan input analog. Yang hasil pembadingan akan di keluarkan melalui LED. Pada rangkaian berikut yang dibandingkan adalah tegangan keluaran dari sensor LDR dengan tegangan patokan / referensi bisa diambil dari tegangan dari luar. Adapun rangkaiannya diperlihatkan gambar berikut:



Gambar 8.2. Rangkaian komparator analog mikrokontroller

8.3. PEMROGRAMAN KOMPARATOR ANALOG

Setelah membuat rangkaian komparator analog mikrokontroller, maka sekarang saatnya Anda membuat program komparator analog mikrokontroller.

Program sebagai berikut ini

```
//-----------------------------------------------------
//Program KOMPARATOR
//-----------------------------------------------------
#include <stdio.h>
#include <mega8535.h>
#include <delay.h>
void InisialisasiCOMPARATOR ();
void main()
{
DDRD=0xFF;
InisialisasiCOMPARATOR();
#asm("sei")
while(1)
{
if (ACSR.5==0) {PORTD=0;}
else {PORTD=0xff;}
}
}
void InisialisasiCOMPARATOR ()
{
ACSR=0x20;
SFIOR=0x00;
}
```

Cara kerja program:

Pada program Bab 10.1. komparator analog, di perlukan deklarasi register untuk mikrokontroller jenis AT90S2313. Setelah mendeklarasi register, maka program akan mendeklarasi port D sebagai output dan PORTB.0 dan PORTB.1 sebagai komparator. Kemudian program masuk ke dalam program utama. Di dalam Program ini akan membandingkan antara komparator analog 1 dan komparator analog 2. Jika komparator analog 0 lebih besar dari pada komparator 1 maka LED mati dan sebaliknya jika komparator analog 1 lebih besar dari pada analog 0 maka LED menyala.