

**PANDUAN PRAKTIS  
MIKROKONTROLER KELUARGA AVR  
MENGUNAKAN  
DT-COMBO AVR-51 STARTER KIT  
DAN DT-COMBO AVR EXERCISE KIT**

**PANDUAN PRAKTIS  
MIKROKONTROLER KELUARGA AVR  
MENGUNAKAN  
DT-COMBO AVR-51 STARTER KIT DAN DT-COMBO AVR EXERCISE KIT**

Didik Wiyono, S.T.

©2007, Innovative Electronics

Hak Cipta dilindungi undang-undang

Sampul & Tata Letak : Gersom Sutedjo, S.T.

Diterbitkan pertama kali oleh :

Innovative Electronics, Surabaya 2007

*Website* : [www.innovativeelectronics.com](http://www.innovativeelectronics.com)

Semua pertanyaan tentang produk ini dapat dikirimkan melalui e-mail ke:  
**[support@innovativeelectronics.com](mailto:support@innovativeelectronics.com)**

Dilarang mengutip, memfotokopi, atau memperbanyak sebagian atau seluruh isi buku ini tanpa izin tertulis dari penerbit.

AVR adalah merk dagang terdaftar dari Atmel Corporation.

CodeVisionAVR adalah hak cipta dari Pavel Haiduc, IIP InfoTech s.r.l.

Windows adalah merk dagang terdaftar dari Microsoft Corporation.

MCS-51 adalah merk dagang terdaftar dari Intel Corporation.

## KATA PENGANTAR

Puji Syukur kepada Tuhan Yang Maha Kuasa atas berkat yang telah diberikanNya dalam penyusunan Panduan Praktis Mikrokontroler Keluarga AVR ini. Terima kasih juga kepada semua pihak yang telah membantu baik secara langsung maupun tidak.

Panduan Praktis Mikrokontroler Keluarga AVR ini digunakan berdampingan dengan DT-COMBO AVR-51 Starter Kit dan DT-COMBO AVR Exercise Kit. Panduan ini mengandung beberapa bagian bahasan mengenai fitur-fitur yang dimiliki mikrokontroler AVR<sup>®</sup>. Masing-masing bagian akan dijelaskan secara singkat dan disertai dengan contoh program dan soal latihan yang menggunakan CodeVisionAVR<sup>®</sup> yang berjalan pada sistem operasi Windows<sup>®</sup>.

Besar harapan penyusun agar para Pengguna, baik Praktikan maupun Pembimbing, dapat memberikan kritik dan saran mengenai isi ataupun penyusunan Panduan Praktis Mikrokontroler Keluarga AVR ini.

Akhir kata, penyusun mengucapkan terima kasih kepada para Pengguna. Selamat berlatih!

Surabaya, Maret 2007

Tim Penyusun

## DAFTAR ISI

HALAMAN JUDUL .....	i
KATA PENGANTAR .....	iii
DAFTAR ISI .....	iv
DAFTAR GAMBAR .....	vii
DAFTAR TABEL .....	xi
1. PENDAHULUAN .....	1
1.1. TUJUAN .....	1
1.2. RUANG LINGKUP .....	1
1.3. SISTEMATIKA PANDUAN PRAKTIS MIKROKONTROLER KELUARGA AVR .....	1
2. PENJELASAN PERANGKAT KERAS .....	3
3. PENJELASAN PERANGKAT LUNAK .....	9
3.1. INSTALASI CODEVISIONAVR C COMPILER .....	9
3.2. CARA MEMBUAT PROYEK BARU .....	13
1) Tanpa CodeWizardAVR .....	13
2) Dengan CodeWizardAVR .....	18
3.3. PEMROGRAMAN MIKROKONTROLER AVR DENGAN DT-HIQ AVR IN SYSTEM PROGRAMMER MELALUI CODEVISIONAVR .....	24
3.4. CARA MENGGUNAKAN PROGRAM TERMINAL PADA CODEVISIONAVR .....	28
4. BASIC INPUT/OUTPUT .....	30
4.1. TUJUAN .....	30
4.2. DASAR TEORI .....	30
4.3. SKEMATIK DAN CONTOH PROGRAM .....	31
1) Pin I/O sebagai Output untuk Mengendalikan LED .....	31
2) Pin I/O sebagai Input untuk Membaca Penekanan Saklar .....	35
3) Pin I/O sebagai Output untuk Mengendalikan Relay .....	38
4.4. SOAL LATIHAN .....	42

5.	SCANNING (7 SEGMENT, DOT MATRIX, dan KEYPAD) .....	44
5.1.	TUJUAN .....	44
5.2.	DASAR TEORI .....	44
5.3.	SKEMATIK DAN CONTOH PROGRAM .....	45
	1) <i>Scanning 7 Segment</i> .....	45
	2) <i>Scanning Dot Matrix</i> .....	60
	3) <i>Scanning Keypad</i> .....	72
5.4.	SOAL LATIHAN .....	79
6.	PARALEL (LCD & EXTERNAL DATA MEMORY) .....	81
6.1.	TUJUAN .....	81
6.2.	DASAR TEORI .....	81
6.3.	SKEMATIK DAN CONTOH PROGRAM .....	82
	1) Display LCD .....	82
	2) Memori RAM .....	91
6.4.	SOAL LATIHAN .....	98
7.	USART .....	99
7.1.	TUJUAN .....	99
7.2.	DASAR TEORI .....	99
7.3.	SKEMATIK DAN CONTOH PROGRAM .....	101
7.4.	SOAL LATIHAN .....	110
8.	ADC (Analog-to-Digital Converter) .....	111
8.1.	TUJUAN .....	111
8.2.	DASAR TEORI .....	111
8.3.	SKEMATIK DAN CONTOH PROGRAM .....	113
	1) Pengukuran Suhu dengan Sensor Suhu LM35DZ .....	114
	2) Pengukuran Intensitas Cahaya dengan Sensor Cahaya LDR ( <i>Light Dependent Resistor</i> ) .....	120
8.4.	SOAL LATIHAN .....	125
9.	SERIAL (SPI dan TWI) .....	126
9.1.	TUJUAN .....	126
9.2.	DASAR TEORI .....	126
9.3.	SKEMATIK & CONTOH PROGRAM .....	133
	1) Penggunaan Komunikasi SPI Mikrokontroler AVR ATMEGA8535 untuk Membaca Tanda Pengenal ( <i>Read Signature</i> ) Mikrokontroler AVR ATTINY2313 dan Menulis ( <i>Write</i> ) Suatu Data ke Memori Program ( <i>Flash</i> ) .....	133

2)	Penggunaan TWI Mikrokontroler AVR ATMEGA8535 untuk Menulis atau Membaca Data pada Memori EEPROM 24C01A .....	143
9.4.	SOAL LATIHAN .....	150
10.	MOTOR (DC, <i>STEPPER</i> , dan <i>SERVO</i> ) .....	151
10.1.	TUJUAN .....	151
10.2.	DASAR TEORI .....	151
10.3.	SKEMATIK DAN CONTOH PROGRAM .....	155
1)	Penggunaan <i>16-bit Timer/Counter1</i> sebagai PWM pada Mikrokontroler AVR ATMEGA8535 untuk Mengendalikan Kecepatan Putar Suatu Motor DC .....	155
2)	Penggunaan <i>Pin I/O</i> Mikrokontroler untuk Menggerakkan Motor <i>Stepper</i> serta Menggunakan <i>8-bit Timer/Counter0</i> untuk Mengatur Kecepatan Putar Motor <i>Stepper</i> .....	162
3)	Penggunaan <i>Pin I/O</i> Mikrokontroler untuk Menggerakkan Motor <i>Servo</i> serta <i>8-bit Timer/Counter2</i> sebagai Waktu Tunda .....	170
10.4.	SOAL LATIHAN .....	174
11.	DAC ( Digital-to-Analog Converter ) .....	175
11.1.	TUJUAN .....	175
11.2.	DASAR TEORI .....	175
11.3.	SKEMATIK DAN CONTOH PROGRAM .....	175
11.4.	SOAL LATIHAN .....	184
	PENUTUP .....	185
	DAFTAR PUSTAKA .....	186

# 1. PENDAHULUAN

## 1.1. TUJUAN

Tujuan dari keseluruhan panduan ini adalah agar Pengguna mampu memprogram mikrokontroler keluarga AVR menggunakan fitur-fitur yang dimiliki oleh DT-COMBO AVR-51 *Starter Kit* dalam program-program berbahasa C dengan bantuan DT-COMBO AVR *Exercise Kit*.

## 1.2. RUANG LINGKUP

Panduan ini mencakup percobaan menggunakan DT-COMBO AVR-51 *Starter Kit*, DT-COMBO AVR *Exercise Kit*, dengan bantuan *Personal Computer* (PC) dan *software CodeVisionAVR*<sup>®</sup>.

## 1.3. SISTEMATIKA PANDUAN PRAKTIS MIKROKONTROLER KELUARGA AVR

Panduan Praktis Mikrokontroler Keluarga AVR ini terbagi dalam 11 bab. Adapun garis besar masing-masing bab adalah sebagai berikut:

- a. Bab I PENDAHULUAN:  
Membahas mengenai tujuan, ruang lingkup, serta sistematika Panduan Praktis Mikrokontroler Keluarga AVR.
- b. Bab II PENJELASAN PERANGKAT KERAS:  
Menjelaskan mengenai fitur-fitur DT-COMBO AVR-51 *Starter Kit* dan cara menghubungkan perangkat keras.
- c. Bab III PENJELASAN PERANGKAT LUNAK:  
Menjelaskan mengenai cara penggunaan CodeVisionAVR<sup>®</sup> dan fiturnya.
- d. Bab IV *BASIC INPUT/OUTPUT*:  
Membahas teori dan latihan mengenai penggunaan *Port* sebagai jalur *input* dan *output* secara per *bit* dan per *byte*.
- e. Bab V *SCANNING (7 SEGMENT, DOT MATRIX, dan KEYPAD)*:  
Membahas teori dan latihan mengenai teknik *scanning* menggunakan *delay* dan *timer* untuk membaca penekanan tombol *keypad*, menyalakan *7 segment* dan menyalakan *dot matrix*.
- f. Bab VI PARALEL (*LCD & EXTERNAL DATA MEMORY*):  
Membahas teori dan latihan mengenai penggunaan *pin I/O AVR* untuk mengakses LCD (*Liquid Crystal Display*) dan memori eksternal.
- g. Bab VII USART:  
Membahas teori dan latihan mengenai penggunaan komunikasi serial USART untuk berkomunikasi dengan komputer.
- h. Bab VIII ADC (*Analog-to-Digital Converter*):  
Membahas teori dan latihan mengenai penggunaan ADC untuk mengkonversi *input* tegangan analog menjadi nilai digital.

- i. Bab IX SERIAL (SPI dan TWI):  
Membahas teori dan latihan mengenai penggunaan SPI (*Serial Peripheral Interface*) dan TWI (*Two-Wire serial Interface*) untuk mengakses *device* lain.
- j. Bab X MOTOR (DC, *STEPPER*, dan *SERVO*) :  
Membahas teori dan latihan mengenai pengendalian motor DC, motor *stepper*, dan motor *servo* dengan bantuan *timer*.
- k. Bab XI DAC ( Digital-to-Analog Converter ) :  
Membahas teori dan latihan mengenai pengendalian DAC untuk menghasilkan tegangan analog.

## 11. DAC ( Digital-to-Analog Converter )

### 11.1. TUJUAN

- Mampu membuat program untuk mengakses DAC menggunakan *pin* I/O AVR.
- Mampu membuat program untuk menghasilkan suatu sinyal segitiga dengan menggunakan DAC.

### 11.2. DASAR TEORI

- DAC berfungsi untuk menghasilkan tegangan analog dari nilai digital, merupakan kebalikan dari ADC yang mengkonversi tegangan analog menjadi nilai digital.
- Seperti pada ADC yang membutuhkan tegangan referensi untuk konversi tegangan analog menjadi nilai digital maka DAC juga membutuhkan tegangan referensi untuk konversi dari nilai digital menjadi tegangan analog. Pada DAC, tegangan referensi digunakan untuk resolusi perubahan tegangan *output* dan sebagai batas tegangan *output* maksimum DAC. Namun *output* DAC dapat diperbesar dengan menggunakan rangkaian *amplifier* (penguat) seperti rangkaian *op-amp inverting amplifier*, *op-amp non-inverting amplifier*, dan sebagainya. Dengan rangkaian tambahan, tegangan maksimum DAC dapat diatur dari rangkaian *amplifier*, tetapi resolusi perubahan tegangan *output* akan berubah sesuai dengan nilai penguatan pada rangkaian *amplifier*.

### 11.3. SKEMATIK DAN CONTOH PROGRAM

- Contoh program menggunakan ADC pada ATMEGA8535. Agar contoh program dapat berjalan dengan baik dan benar (hasil konversi ADC benar) maka pastikan bahwa pengaturan jumper tegangan referensi ADC pada DT-COMBO AVR-51 STARTER KIT terhubung dengan tegangan 5 volt (AVCC), yaitu posisi jumper J12 (ADC Ref Voltage) pada *pin* 1-2.
- Skematik untuk rangkaian DAC dengan mikrokontroler ATMEGA8535 terdapat pada gambar 11.1. Tipe DAC yang digunakan yaitu DAC0832 dengan resolusi sebesar 8-bit. Keterangan yang lebih lengkap terdapat pada *datasheet* DAC0832 tentang *Analog Considerations (Using the DAC0830 in a Voltage Switching Configuration)*. Tegangan *output* DAC ( $V_{OUT}$ ) pada gambar 11.1 dapat dihitung dengan menggunakan persamaan:

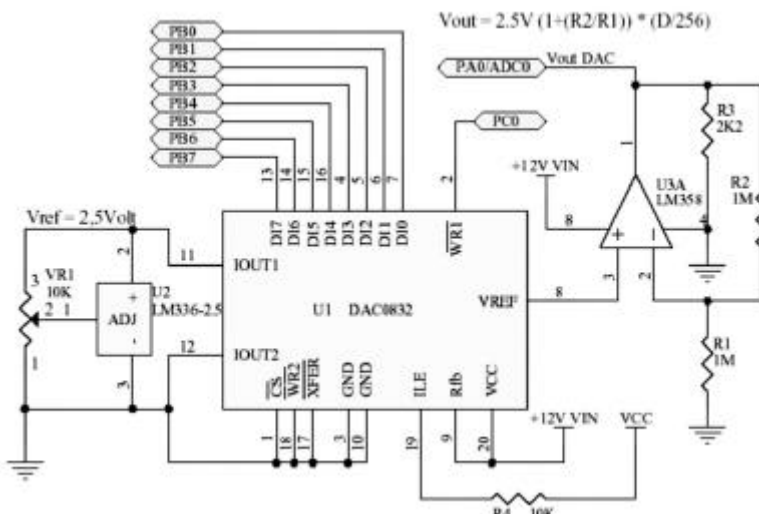
$$V_{OUT} = +2.5V_{DC} \left( 1 + \frac{R_2}{R_1} \right) \left( \frac{D}{256} \right)$$

$$V_{OUT} = +2.5V_{DC} \left( 1 + \frac{1M\Omega}{1M\Omega} \right) \left( \frac{D}{256} \right)$$

$$V_{OUT} = 1.5V_{DC} \left( \frac{D}{256} \right)$$

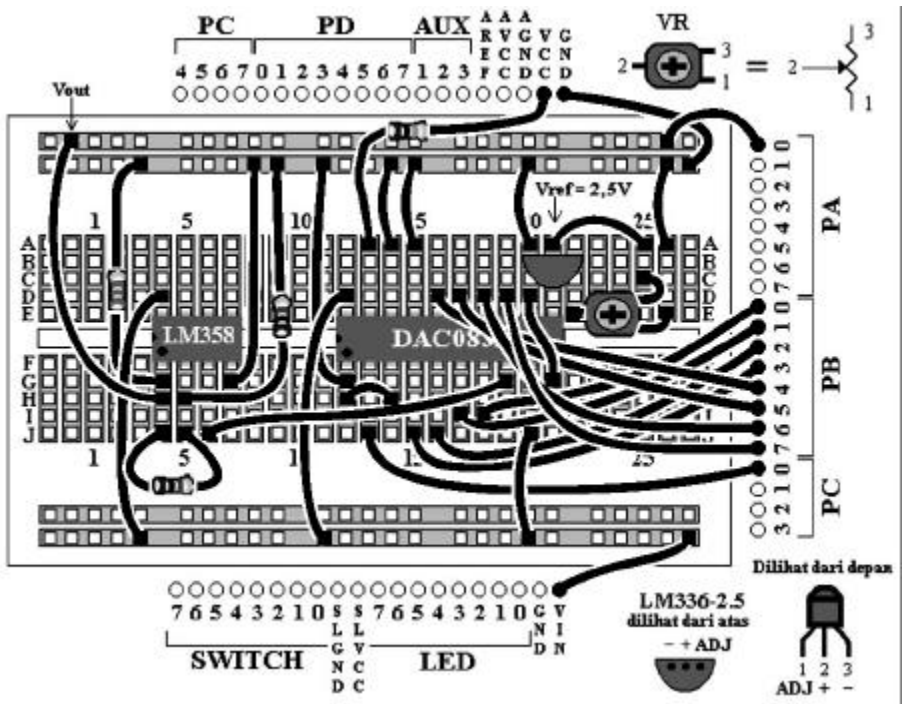
Nilai D merupakan nilai digital *input* untuk DAC yaitu bernilai 0 sampai 255. Agar *output* DAC sesuai dengan persamaan di atas, tegangan referensi harus diatur sebesar 2,5 volt pada *pin* 11 DAC0832. Untuk mendapatkan tegangan referensi sebesar 2,5 volt yaitu dengan mengatur (memutar) resistor variabel VR1, lihat pada gambar 11.1.

- Pada gambar 11.1, *op-amp* LM358 serta resistor R1 dan R2 dirangkai sedemikian rupa sehingga membentuk rangkaian *non-inverting amplifier* dengan penguatan sebesar 2x jika nilai R1 dan R2 sama. Pemilihan resistor R1 dan R2 sebesar 1 M $\Omega$  agar didapatkan impedansi *output* rangkaian *op-amp* yang besar yaitu sebesar 1 M $\Omega$ . Resistor R3 digunakan untuk mengurangi *output* tegangan saturasi dari *op-amp* pada saat *input op-amp* 0 volt.
- Untuk mengubah nilai digital dari mikrokontroler pada DAC yaitu dengan mengirimkan nilai digital terlebih dahulu (*pin* DI0-DI7) kemudian memberikan sinyal *write* ke DAC (*pin*  $\overline{WR1}$  diberi logika '0'). Setelah beberapa waktu (minimal sinyal *write* ( $t_{DS}$ ) 320 ns, dengan  $V_{CC}$  DAC sebesar 12 volt), *pin*  $\overline{WR1}$  diberi logika '1'. Tegangan *output* DAC dapat diukur pada *pin* 1 pada LM358 ( $V_{OUT}$ ), lihat pada gambar 11.3.
- Komponen yang dibutuhkan:
  - Resistor 1 M $\Omega$  (warna: coklat, hitam, hijau, emas) 2 buah
  - Resistor 10 k $\Omega$  (warna: coklat, hitam, jingga, emas) 1 buah
  - Resistor 2,2 k $\Omega$  (warna: merah, merah, merah, emas) 1 buah
  - VR 10 k $\Omega$  1 buah
  - LM336 – 2,5 volt 1 buah
  - *Op Amp* LM358 1 buah
  - DAC0832 1 buah



Gambar 11.1. Rangkaian DAC0832 dengan Mikrokontroler ATMEGA8535

- Gambar rangkaian DAC0832 dengan mikrokontroler ATMEGA8535 yang dirangkai pada papan proyek (*project board*) ditunjukkan pada gambar 11.2.



Gambar 11.2. Tata Letak Komponen pada *Project Board* untuk DAC0832

- Program 1 adalah contoh program untuk mengirimkan nilai digital mulai dari 0 sampai 255 ke DAC0832. Setiap nilai digital yang dikirimkan ke DAC0832 dibaca kembali oleh ADC mikrokontroler ATMEGA8535. Hasil pembacaan ADC dikirimkan ke komputer melalui USART. Untuk melihat hasil pengiriman hasil ADC mikrokontroler pada komputer menggunakan program *terminal*.

Program 1. Program untuk mengirimkan nilai digital ke DAC.

```
#include <mega8535.h>
#include <delay.h>
#include <string.h>
#include <stdio.h>

#define Wx          PORTC.0
#define PortDAC    PORTB
```

```

unsigned long int hasil;
//Fungsi untuk mengubah input biner 16 bit menjadi
//bertuk BCD 5 digit
unsigned long int bin2BCD (unsigned int input)
{
    hasil=((input/10000)*65536)+
        (((input%10000)/1000)*4096)+
        (((((input%10000)%1000)/100)*256)+
        ((((((input%10000)%1000)%100)/10)*16)+
        (((((input%10000)%1000)%100)%10));
    return(hasil);
}

//Tegangan referensi : pin AREF
//Hasil konversi ADC : right adjusted
#define ADC_VREF_TYPE 0x00
//Fungsi untuk membaca tegangan input ADC (konversi
//ADC)
unsigned int read_adc (unsigned char adc_input)
{
    unsigned char temp;
    unsigned int data_adc;
    ADMUX=adc_input|ADC_VREF_TYPE;
    //Start konversi ADC
    ADCSRA|=0x40;          //ADCSRA di-or-kan dengan 40H
                        //((ADCSRA=ADCSRA|0x40)
    //Tunggu sampai konversi ADC selesai (flag ADIF)
    while ((ADCSRA & 0x10)==0);
    ADCSRA|=0x10;        //Clear flag ADIF
    data_adc=0x0000;
    temp=ADCL;          //Baca ADC Data Register LOW
    data_adc=ADCH;      //Baca ADC Data Register HIGH
    data_adc<<=8;       //Geser data_adc HIGH ke kiri
                        //8 kali
    data_adc|=temp;     //Hasil konversi ADC= 0000
                        //00xx xxxx xxxx
    return data_adc;    //Right adjusted
}

flash unsigned char string[]= {"Hasil ADC0 : "};

```

```

flash unsigned char string1[] = {
"Input DAC : %d ; ";
}
void main(void)
{
unsigned char temp,adc_input,dac;
DDRA=0x00;
DDRB=0xFF;
DDRC=0x01;           //PC0 sebagai output
PORTB.0=1;
//Tegangan referensi ADC : pin AREF
ADMUX=ADC_VREF_TYPE;
//Inisialisai ADC (Enable ADC)
//Frekuensi clock ADC      : 31.250 kHz (4MHz/128)
ADCSRA=0x87;
SFIOR&=0xEF;        //Sumber trigger : Free
                        //Running mode

// Inisialisasi USART 8-N-1 dengan baud rate
// 9600 bps (4MHz)
UCSRA=0x00;
UCSRB=0x18;
UCSRC=0x86;
UBRRH=0x00;
UBRRL=0x19;

dac=0x00;           //Nilai awal data DAC

while (1)
{
printf(string1,dac);
PortDAC=dac;       //Kirim nilai digital ke
                        //DAC0832
asm("nop");        //Data setup time (tDS)
asm("nop");
WR=0;              //Write DAC
asm("nop");        //Write time (tW)
asm("nop");
WR=1;
dac++;
printf(string);

adc_input=0x00;
}
}

```

```

    hasil=read_adc(adc_input);
        //Baca tegangan pada pin ADC0
    hasil=hasil*49;    //Hasil pembacaan
                        //dikalikan dengan 1 LSB
    hasil=bin2BCD(hasil);
        //Konversi hasil ADC ke format BCD
    temp=(hasil>>16 & 0x0F)+0x30;
        //Ubah format BCD ke ASCII
    putchar(temp);    //Kirim ASCII ke USART
    putchar(' ');
    temp=(hasil>>12 & 0x0F)+0x30;
    putchar(temp);
    temp=(hasil>>8 & 0x0F)+0x30;
    putchar(temp);
    temp=(hasil>>4 & 0x0F)+0x30;
    putchar(temp);
    temp=(hasil & 0x0F)+0x30;
    putchar(temp);
    putchar(' ');
    putchar('\n');
    putchar(0x0D);    //Kirim ENTER pada terminal
                        //untuk baris baru

    putchar(0x0A);
    delay_ms(200);
}
}

```

- Program 2 merupakan contoh program untuk membuat sinyal segitiga sama kaki menggunakan DAC. Nilai digital mulai dari 0 sampai 255 dikirimkan ke DAC, setelah sampai ke nilai 255, nilai digital mulai dari 254 sampai 0 dikirimkan ke DAC, kemudian kembali ke nilai 0 sampai 255, demikian seterusnya. Jadi nilai 0 sampai 255 kemudian 254 sampai 0 dikirimkan berulang-ulang sehingga tegangan *output* DAC dapat membentuk segitiga sama kaki. Apabila tombol yang terhubung dengan *pin* PC6 ditekan maka bentuk segitiga akan semakin landai (pengiriman nilai digital ke DAC diperlambat dengan menambah *delay*), sebaliknya apabila tombol yang terhubung dengan *pin* PC7 ditekan maka bentuk segitiga akan semakin lancip (pengiriman nilai digital ke DAC dipercepat dengan mengurangi *delay*).

Catatan:

- Contoh program ini menggunakan tombol *push-button* yang terhubung dengan *pin* PC6 dan PC7 untuk membuat sinyal segitiga semakin landai

atau lancip. Agar tombol *push-button* dapat berfungsi maka hubungkan *header* “PORTC” dengan *header* “SWITCH” menggunakan kabel 10-*pin* atau kabel pita/*flat ribbon cable*.

- Voltmeter dapat digunakan untuk memeriksa tegangan keluaran DAC. Jika naik/turunnya tegangan makin cepat, berarti segitiga makin lancip. Jika naik/turunnya tegangan makin pelan, berarti segitiga makin landai.

Program 2. Program untuk mengirimkan sebuah kata ke komputer.

```
#include <mega8535.h>
#include <delay.h>
#include <string.h>
#include <stdio.h>

#define WR          PORTC.0
#define PortDAC    PORTB

unsigned long int hasil;
//Fungsi untuk mengubah input biner 16 bit menjadi
//bentuk BCD 5 digit
unsigned long int bin2BCD (unsigned int input)
{
    hasil=((input/10000)*65536)+
           (((input%10000)/1000)*4096)+
           (((((input%10000)%1000)/100)*256)+
           ((((((input%10000)%1000)%100)/10)*16)+
           (((((input%10000)%1000)%100)%10);
    return(hasil);
}

//Tegangan referensi : pin AREF
//Hasil konversi ADC : right adjusted
#define ADC_VREF_TYPE 0x00
//Fungsi untuk membaca tegangan input ADC
//(konversi ADC)
unsigned int read_adc (unsigned char adc_input)
{
    unsigned char temp;
    unsigned int data_adc;
    ADMUX=adc_input|ADC_VREF_TYPE;
    //Start konversi ADC
    ADCSRA|=0x40;          //ADCSRA di-or-kan dengan 40H
                          // (ADCSRA=ADCSRA|0x40)
    //Tunggu sampai konversi ADC selesai (flag ADIF)
    while ((ADCSRA & 0x10)==0);
    ADCSRA|=0x10;         //Clear flag ADIF
    data_adc=0x0000;
    temp=ADCR;           //Baca ADC Data Register LOW
    data_adc=ADCH;       //Baca ADC Data Register HIGH
    data_adc<<=8;        //Geser data_adc HIGH
```

```

        //ke kiri 8 kali
data_adc|=temp; //Hasil konversi ADC= 0000
                //00xx xxxx xxxx
return data_adc; //Right adjusted
}

flash unsigned char string[]= {"Hasil ADC0 : "};
flash unsigned char string1[]= {
"Input DAC : %d ; "};
bit i;
void main(void)
{
    unsigned char temp,dly,adc_input,dac;
    DDRA=0x0C;
    DDRB=0xFF;
    DDRC=0x01; //PC0 sebagai output,
                //PC1-PC7 sebagai input
    PORTB.C=1;
    //Tegangan referensi ADC : pin AREF
    ADMUX=ADC_VREF_TYPE;
    //Inisialisasi ADC (Enable ADC)
    //Trekkuensi clock ADC : 31.250 kHz (4MHz/128)
    ADCSRA=0x87;
    SFIOR&=0xEF; //Sumber trigger : Free
                //Running mode
    // Inisialisasi USART 8-N-1 dengan baud rate
    //9600 bps (4MHz)
    UCSRA=0x00;
    UCSRB=0x18;
    UCSRC=0x86;
    UBRRH=0x00;
    UBRRL=0x19;

    dac=0x00; //Nilai awal data DAC
    i=0; //Variabel dac --> increment
    dly=50; //Nilai awal variabel delay
    while (1)
    {
        printf(string1,dac);
        PortDAC=dac;//Kirim nilai digital ke
                //DAC0832
        #asm("nop");//Data setup time (tDS)
        #asm("nop");
        WR=0; //Write DAC
        #asm("nop");//Write time (tW)
        #asm("nop");
        WR=1;
        if (i==0)
        {
            dac++;
            if (dac==255) i=1;
        }
    }
}

```

```

else
{
    dac--;
    if (dac==0) i=0;
}
printf(string);
adc_input=0x00;
hasil=read_adc(adc_input);
    //Baca tegangan pada pin ADC0
hasil=hasil*49;
    //Hasil pembacaan dikalikan dengan 1 LSB
hasil=bin2BCD(hasil);
    //Konversi hasil ADC ke format BCD
temp=(hasil>>16 & 0x0F)+0x30;
    //Ubah format BCD ke ASCII
putchar(temp);    //Kirim ASCII ke USART
putchar(',');
temp=(hasil>>12 & 0x0F)+0x30;
putchar(temp);
temp=(hasil>>8 & 0x0F)+0x30;
putchar(temp);
temp=(hasil>>4 & 0x0F)+0x30;
putchar(temp);
temp=(hasil & 0x0F)+0x30;
putchar(temp);
putchar(' ');
putchar('V');

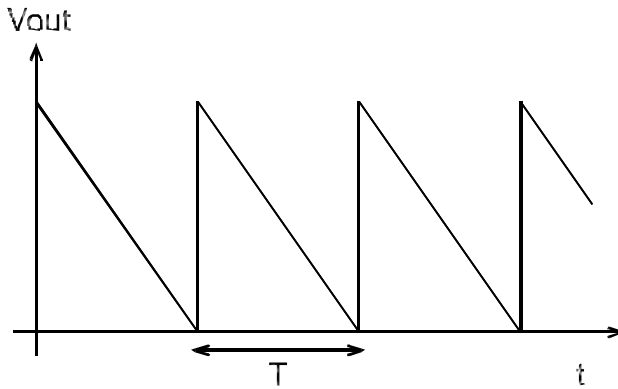
putchar(0x0D);    //Kirim ENTER pada terminal
                  //untuk baris baru
putchar(0x0A);

if (PINC.6==0)    //Jika tombol yang
                  //terhubung dengan PC6
    {
        //ditekan maka sinyal
        //segitiga semakin landai
        dly++;
        if (dly==0x00) dly=0xFF;
    }
if (PINC.7==0)    //Jika tombol yang
                  //terhubung dengan PC7
    {
        // ditekan maka sinyal
        //segitiga semakin lancip
        dly--;
        if (dly==0) dly=1;
    }
delay_ms(dly);
}
}

```

#### 11.4. SOAL LATIHAN

1. Buatlah program untuk membuat sinyal segitiga seperti ditunjukkan pada gambar 11.3 dengan lebar periode ( $T$ ) dapat diatur. Jadi mikrokontroler memberikan nilai digital sebesar 255 kemudian turun sampai 0, kemudian kembali lagi ke nilai 255 sampai 0 demikian seterusnya. Pemberian nilai digital ini merupakan kebalikan dari contoh program 1. Pengaturan lebar periode yaitu dengan menunda pemberian nilai digital ke DAC seperti pada contoh program 2, gunakan tombol PC6 untuk memperlebar periode (*delay* diperbesar sehingga sinyal segitiga semakin landai) dan tombol PC7 untuk memperkecil periode (*delay* diperkecil sehingga sinyal segitiga semakin lancip).



Gambar 11.3. Bentuk Sinyal Segitiga untuk Soal 1.