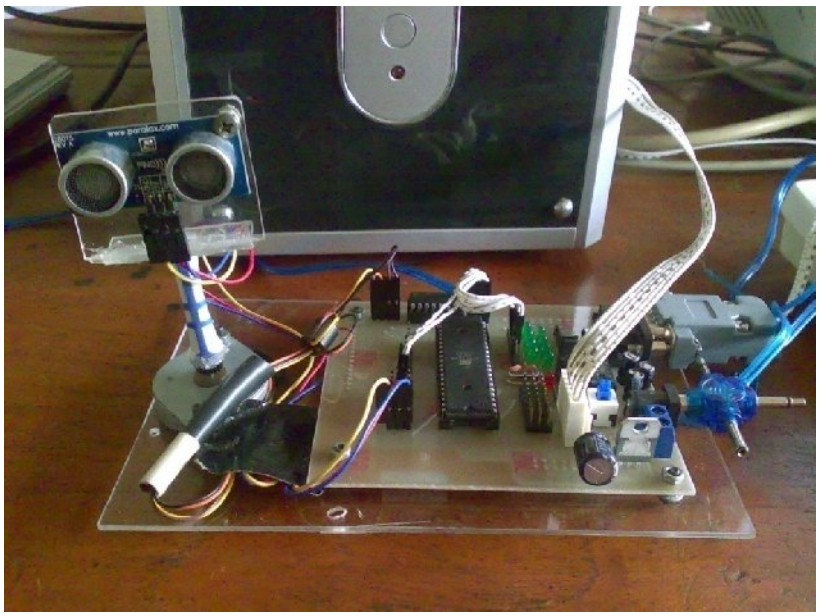
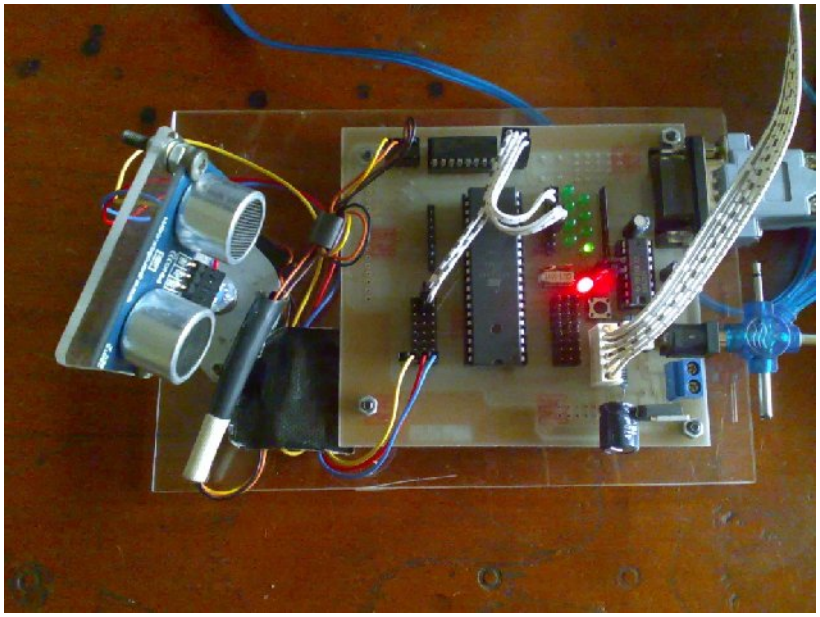
LAMPIRAN A

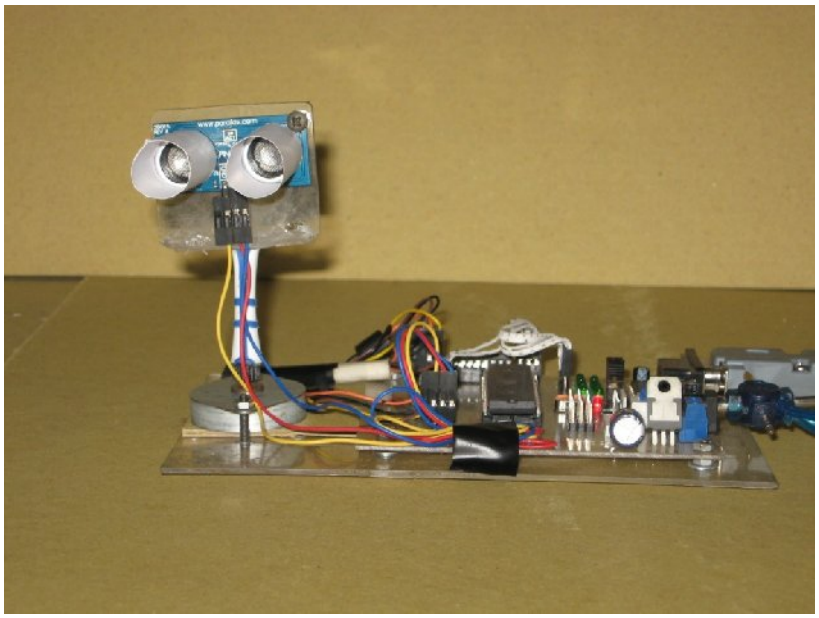
FOTO ALAT



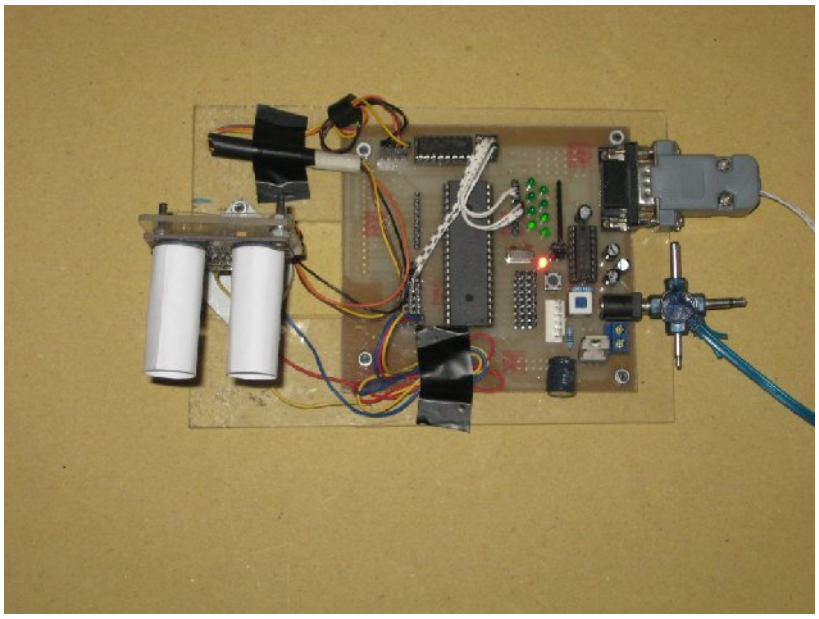
Gambar A.1 Foto Alat Tampak Samping Depan



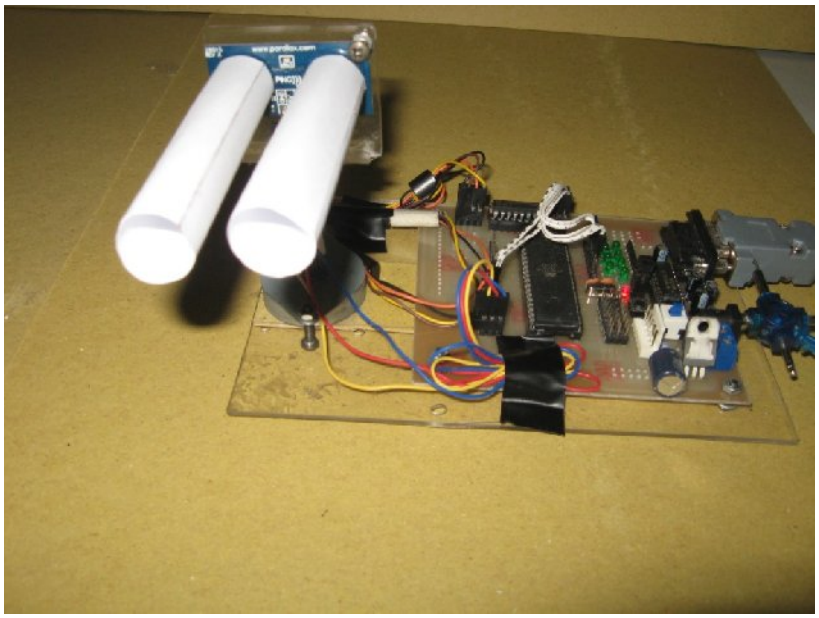
Gambar A.2 Foto Alat Tampak Atas



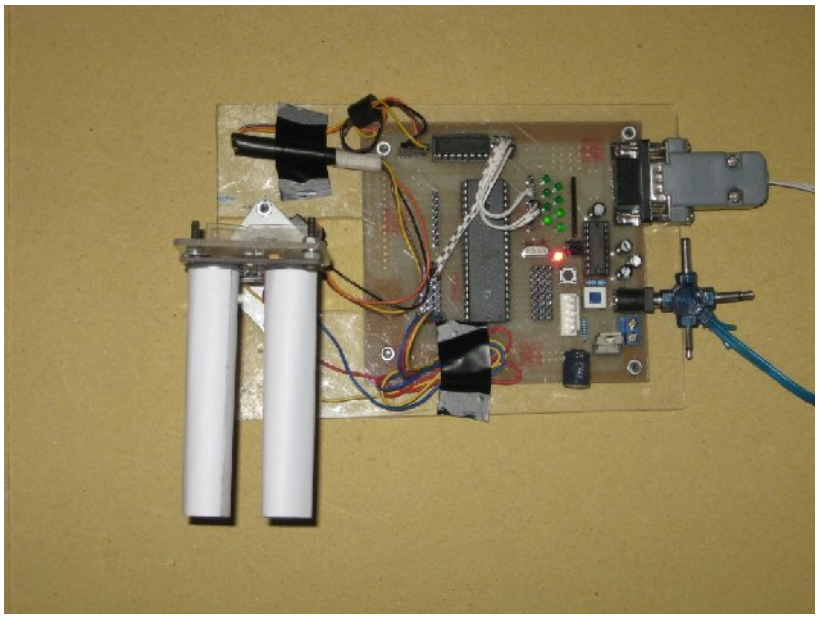
Gambar A.3 Foto Alat Tampak Samping Depan dengan Cerobong Pengarah 3 cm



Gambar A.4 Foto Alat Tampak Atas dengan Cerobong Pengarah 3 cm



Gambar A.5 Foto Alat Tampak Samping Depan dengan Cerobong Pengarah 8 cm



Gambar A.6 Foto Alat Tampak Atas dengan Cerobong Pengarah 8 cm

LAMPIRAN B

LISTING PROGRAM

Program pada Mikrokontroler

```
/*****************************************************
This program was produced by the
CodeWizardAVR V1.25.3 Professional
Automatic Program Generator
© Copyright 1998-2007 Pavel Haiduc, HP InfoTech s.r.l.
http://www.hpinfotech.com

Project :
Version :
Date    : 8/16/2008
Author  : Nurjani
Company : Studio25
Comments:


Chip type           : ATmega16
Program type        : Application
Clock frequency     : 11.059200 MHz
Memory model        : Small
External SRAM size  : 0
Data Stack size     : 256
*****************************************************/

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

volatile char i,akhir;
volatile int data[96];
volatile unsigned int j;
```

```
volatile bit start=0,stop=0;

#define RXB8 1
#define TXB8 0
#define UPE 2
#define OVR 3
#define FE 4
#define UDRE 5
#define RXC 7

#define FRAMING_ERROR (1<<FE)
#define PARITY_ERROR (1<<UPE)
#define DATA_OVERRUN (1<<OVR)
#define DATA_REGISTER_EMPTY (1<<UDRE)
#define RX_COMPLETE (1<<RXC)

// USART Receiver buffer
#define RX_BUFFER_SIZE 8
char rx_buffer[RX_BUFFER_SIZE];

#if RX_BUFFER_SIZE<256
unsigned char rx_wr_index,rx_rd_index,rx_counter;
#else
unsigned int rx_wr_index,rx_rd_index,rx_counter;
#endif

// This flag is set on USART Receiver buffer overflow
bit rx_buffer_overflow;

// USART Receiver interrupt service routine
interrupt [USART_RXC] void usart_rx_isr(void)
{
```

```
char status,data;
status=UCSRA;
data=UDR;
if ((status & (FRAMING_ERROR | PARITY_ERROR |
DATA_OVERRUN))==0)
  {
  rx_buffer[rx_wr_index]=data;
  if (++rx_wr_index == RX_BUFFER_SIZE) rx_wr_index=0;
  if (++rx_counter == RX_BUFFER_SIZE)
    {
    rx_counter=0;
    rx_buffer_overflow=1;
    };
  };
  data=getchar();
  if(data=='x')
  start=1;
  if(data=='z')
  stop=1;
}

#ifndef _DEBUG_TERMINAL_IO_
// Get a character from the USART Receiver buffer
#define _ALTERNATE_GETCHAR_
#pragma used+
char getchar(void)
{
char data;
while (rx_counter==0);
data=rx_buffer[rx_rd_index];
if (++rx_rd_index == RX_BUFFER_SIZE) rx_rd_index=0;
#asm("cli")
```

```
--rx_counter;
#asm("sei")
return data;
}
#pragma used-
#endif

// USART Transmitter buffer
#define TX_BUFFER_SIZE 8
char tx_buffer[TX_BUFFER_SIZE];

#if TX_BUFFER_SIZE<256
unsigned char tx_wr_index,tx_rd_index,tx_counter;
#else
unsigned int tx_wr_index,tx_rd_index,tx_counter;
#endif

// USART Transmitter interrupt service routine
interrupt [USART_TXC] void usart_tx_isr(void)
{
if (tx_counter)
   {
   --tx_counter;
   UDR=tx_buffer[tx_rd_index];
   if (++tx_rd_index == TX_BUFFER_SIZE) tx_rd_index=0;
   };
}

#ifndef _DEBUG_TERMINAL_IO_
// Write a character to the USART Transmitter buffer
#define _ALTERNATE_PUTCHAR_
#pragma used+
```

```
void putchar(char c)
{
while (tx_counter == TX_BUFFER_SIZE);
#asm("cli")
if (tx_counter || ((UCSRA & DATA_REGISTER_EMPTY)==0))
   {
   tx_buffer[tx_wr_index]=c;
   if (++tx_wr_index == TX_BUFFER_SIZE) tx_wr_index=0;
   ++tx_counter;
   }
else
   UDR=c;
#asm("sei")
}
#pragma used-
#endif

// Standard Input/Output functions
#include <stdio.h>

// Declare your global variables here

void main(void)
{
// Declare your local variables here

// Input/Output Ports initialization
// Port A initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In
Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTA=0x00;
```

```
DDRA=0x00;

// Port B initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In
Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTB=0x00;
DDRB=0xff;

// Port C initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In
Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTC=0x00;
DDRC=0x00;

// Port D initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In
Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTD=0x00;
DDRD=0xfc;

// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: Timer 0 Stopped
// Mode: Normal top=FFh
// OC0 output: Disconnected
TCCR0=0x00;
TCNT0=0x00;
OCR0=0x00;
```

```
// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: Timer 1 Stopped
// Mode: Normal top=FFFFh
// OC1A output: Discon.
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer 1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
TCCR1A=0x00;
TCCR1B=0x00;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: Timer 2 Stopped
// Mode: Normal top=FFh
// OC2 output: Disconnected
ASSR=0x00;
TCCR2=0x00;
TCNT2=0x00;
OCR2=0x00;
```

```
// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
// INT2: Off
MCUCR=0x00;
MCUCSR=0x00;

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x00;

// USART initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART Receiver: On
// USART Transmitter: On
// USART Mode: Asynchronous
// USART Baud rate: 9600
UCSRA=0x00;
UCSRB=0xD8;
UCSRC=0x86;
UBRRH=0x00;
UBRRL=0x47;

// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
ACSR=0x80;
SFIOR=0x00;

// Global enable interrupts
#asm("sei")
delay_ms(2000);
```

```
PORTD=4;

while (1)
    {
    mulai:
    if(start==1)
    {
    goto eksekusi;
    }
    goto mulai;

//================================================
eksekusi:

//hapus data sebelumnya=================================
    for(i=0;i<96;i++)
    data[i]=0;

//muter sambil ngambil data=============================
    for(i=0;i<96;i++)
    {
    if(stop==1)
    goto balik;
    awal:
    DDRA.0=1;                    // sensor ultrasonik bekerja
    PORTA.0=1;
    delay_us(20);
    PORTA.0=0;
    DDRA.0=0;
    PORTA.0=1;

    for(j=0;j<1000;j++)
```

```
    {
    if(PINA.0==1)
    goto lanjut;
    delay_us(1);
    }

    goto awal;

    lanjut:
    while(PINA.0==1)
    {data[i]=data[i]+1;delay_us(1);}

    delay_ms(100);
    PORTD=PORTD*2;              //  motor stepper berputar berlawan arah jarum
jam
    if(PORTD==64)
    PORTD=4;
    }

    balik:
    stop=0;

    delay_ms(500);                  // delay sebelum berputar balik
    akhir=i;


    for(i=0;i<akhir;i++)            // kirim data ke komputer
    printf("#%3u",data[i]);
    printf("#");

    for(i=0;i<akhir;i++)            // motor stepper berputar kembali ke posisi awal
    {                               // searah jarum jam
```

```
    PORTD=PORTD/2;
    if(PORTD==2)
    PORTD=32;
    delay_ms(100);
    }

    putchar(13);
    delay_ms(1000);                 //delay akhir

    start=0;
    };
}
```

Program pada Komputer

```
Dim skala As Integer
Dim InputMikro As Variant
Dim Proses As Variant
Dim masukan As Variant
Dim x, y As Single
Dim r, theta As Single
Dim Panjang(360) As Single
Dim X_Aksen(360) As Single
Dim Y_Aksen(360) As Single
Dim i As Single
-----------------------------------------------------------------------------------------------
Sub drawing()

Picture1.Cls

Picture1.DrawWidth = 1
For grid_x = 0 To Picture1.Height Step Picture1.Height / 10
Picture1.Line (0, grid_x)-(Picture1.Width, grid_x)
Next grid_x

For grid_y = 0 To Picture1.Width Step Picture1.Width / 10
Picture1.Line (grid_y, 0)-(grid_y, Picture1.Height)
Next grid_y

Picture1.DrawWidth = 8
Picture1.PSet (3000, 3000), vbRed

   InputMikro = Text1.Text
   Proses = Split(InputMikro, "#")
```

```
    Picture1.DrawWidth = 5
    q = 1

    For theta = 0 To 360 Step (360 / 96)

        Panjang(theta) = (Val(Proses(q)) * 1.08 * skala)

        q = q + 1

        x = Panjang(theta) * Sin((theta * 3.14159) / 180)
        y = Panjang(theta) * Cos((theta * 3.14159) / 180)

        X_Aksen(theta) = Val(Picture1.Width / 2 - x)
        Y_Aksen(theta) = Val(Picture1.Height / 2 - y)

        If theta < 360 Then
            Picture1.PSet (X_Aksen(theta), Y_Aksen(theta)), vbGreen

        End If

    Next theta

    Picture1.DrawWidth = 1

    For i = 0 To 360 Step (360 / 96)

        If i < (360 - (360 / 96)) Then
        Picture1.Line (X_Aksen(i), Y_Aksen(i))-(X_Aksen(i + (360 / 96)),
Y_Aksen(i + (360 / 96))), vbBlue

        ElseIf i = (360 - (360 / 96)) Then
```

```
        Picture1.Line (X_Aksen(i), Y_Aksen(i))-(X_Aksen(0), Y_Aksen(0)),
vbBlue

      Else

      End If
   Next i

End Sub
-----------------------------------------------------------------------------------------------
Private Sub Command1_Click()

MSComm1.Output = "x"
Timer1 = True
Timer1.Interval = 11000

End Sub
-----------------------------------------------------------------------------------------------
Private Sub Command2_Click()

MSComm1.Output = "z"

End Sub
-----------------------------------------------------------------------------------------------
Private Sub Command3_Click()

Select Case skala

Case 1
skala = 2
zoom
Label1.Caption = "1500mm"
```

Label2.Caption = "1200mm"
Label3.Caption = "900mm"
Label4.Caption = "600mm"
Label5.Caption = "300mm"
Label6.Caption = "0mm"
Label7.Caption = "300mm"
Label8.Caption = "600mm"
Label9.Caption = "900mm"
Label10.Caption = "1200mm"
Label11.Caption = "1500mm"

Case 2
skala = 3
zoom
Label1.Caption = "1000mm"
Label2.Caption = "800mm"
Label3.Caption = "600mm"
Label4.Caption = "400mm"
Label5.Caption = "200mm"
Label6.Caption = "0mm"
Label7.Caption = "200mm"
Label8.Caption = "400mm"
Label9.Caption = "600mm"
Label10.Caption = "800mm"
Label11.Caption = "1000mm"

Case 3
skala = 4
zoom
Label1.Caption = "750mm"
Label2.Caption = "600mm"
Label3.Caption = "4500mm"

Label4.Caption = "300mm"

Label5.Caption = "150mm"

Label6.Caption = "0mm"

Label7.Caption = "150mm"

Label8.Caption = "300mm"

Label9.Caption = "450mm"

Label10.Caption = "600mm"

Label11.Caption = "750mm"

Case 4

skala = 5

zoom

Label1.Caption = "600mm"

Label2.Caption = "480mm"

Label3.Caption = "360mm"

Label4.Caption = "240mm"

Label5.Caption = "120mm"

Label6.Caption = "0mm"

Label7.Caption = "120mm"

Label8.Caption = "240mm"

Label9.Caption = "360mm"

Label10.Caption = "480mm"

Label11.Caption = "600mm"

Case 5

skala = 6

zoom

Label1.Caption = "500mm"

Label2.Caption = "400mm"

Label3.Caption = "300mm"

Label4.Caption = "200mm"

Label5.Caption = "100mm"

Label6.Caption = "0mm"
Label7.Caption = "100mm"
Label8.Caption = "200mm"
Label9.Caption = "300mm"
Label10.Caption = "400mm"
Label11.Caption = "500mm"

Case 6
skala = 8
zoom
Label1.Caption = "375mm"
Label2.Caption = "300mm"
Label3.Caption = "225mm"
Label4.Caption = "150mm"
Label5.Caption = "75mm"
Label6.Caption = "0mm"
Label7.Caption = "75mm"
Label8.Caption = "150mm"
Label9.Caption = "225mm"
Label10.Caption = "300mm"
Label11.Caption = "375mm"

Case 8
skala = 10
zoom
Label1.Caption = "300mm"
Label2.Caption = "240mm"
Label3.Caption = "180mm"
Label4.Caption = "120mm"
Label5.Caption = "60mm"
Label6.Caption = "0mm"
Label7.Caption = "60mm"

Label8.Caption = "120mm"
Label9.Caption = "180mm"
Label10.Caption = "240mm"
Label11.Caption = "300mm"

Case 10
skala = 15
zoom
Label1.Caption = "200mm"
Label2.Caption = "160mm"
Label3.Caption = "120mm"
Label4.Caption = "80mm"
Label5.Caption = "40mm"
Label6.Caption = "0mm"
Label7.Caption = "40mm"
Label8.Caption = "80mm"
Label9.Caption = "120mm"
Label10.Caption = "160mm"
Label11.Caption = "200mm"

Case 15
skala = 20
zoom
Label1.Caption = "150mm"
Label2.Caption = "120mm"
Label3.Caption = "90mm"
Label4.Caption = "60mm"
Label5.Caption = "30mm"
Label6.Caption = "0mm"
Label7.Caption = "30mm"
Label8.Caption = "60mm"
Label9.Caption = "90mm"

```
Label10.Caption = "120mm"
Label11.Caption = "150mm"

Case 20
skala = 30
zoom
Label1.Caption = "100mm"
Label2.Caption = "80mm"
Label3.Caption = "60mm"
Label4.Caption = "40mm"
Label5.Caption = "20mm"
Label6.Caption = "0mm"
Label7.Caption = "20mm"
Label8.Caption = "40mm"
Label9.Caption = "60mm"
Label10.Caption = "80mm"
Label11.Caption = "100mm"

End Select

End Sub
---------------------------------------------------------------------------------------------
Private Sub Command4_Click()

Select Case skala
Case 30
skala = 20
zoom
Label1.Caption = "150mm"
Label2.Caption = "120mm"
Label3.Caption = "90mm"
Label4.Caption = "60mm"
```

```
Label5.Caption = "30mm"
Label6.Caption = "0mm"
Label7.Caption = "30mm"
Label8.Caption = "60mm"
Label9.Caption = "90mm"
Label10.Caption = "120mm"
Label11.Caption = "150mm"

Case 20
skala = 15
zoom
Label1.Caption = "200mm"
Label2.Caption = "160mm"
Label3.Caption = "120mm"
Label4.Caption = "80mm"
Label5.Caption = "40mm"
Label6.Caption = "0mm"
Label7.Caption = "40mm"
Label8.Caption = "80mm"
Label9.Caption = "120mm"
Label10.Caption = "160mm"
Label11.Caption = "200mm"

Case 15
skala = 10
zoom
Label1.Caption = "300mm"
Label2.Caption = "240mm"
Label3.Caption = "180mm"
Label4.Caption = "120mm"
Label5.Caption = "60mm"
Label6.Caption = "0mm"
```

```
Label7.Caption = "60mm"
Label8.Caption = "120mm"
Label9.Caption = "180mm"
Label10.Caption = "240mm"
Label11.Caption = "300mm"

Case 10
skala = 8
zoom
Label1.Caption = "375mm"
Label2.Caption = "300mm"
Label3.Caption = "225mm"
Label4.Caption = "150mm"
Label5.Caption = "75mm"
Label6.Caption = "0mm"
Label7.Caption = "75mm"
Label8.Caption = "150mm"
Label9.Caption = "225mm"
Label10.Caption = "300mm"
Label11.Caption = "375mm"

Case 8
skala = 6
zoom
Label1.Caption = "500mm"
Label2.Caption = "400mm"
Label3.Caption = "300mm"
Label4.Caption = "200mm"
Label5.Caption = "100mm"
Label6.Caption = "0mm"
Label7.Caption = "100mm"
Label8.Caption = "200mm"
```

Label9.Caption = "300mm"
Label10.Caption = "400mm"
Label11.Caption = "500mm"

Case 6
skala = 5
zoom
Label1.Caption = "600mm"
Label2.Caption = "480mm"
Label3.Caption = "360mm"
Label4.Caption = "240mm"
Label5.Caption = "120mm"
Label6.Caption = "0mm"
Label7.Caption = "120mm"
Label8.Caption = "240mm"
Label9.Caption = "360mm"
Label10.Caption = "480mm"
Label11.Caption = "600mm"

Case 5
skala = 4
zoom
Label1.Caption = "750mm"
Label2.Caption = "600mm"
Label3.Caption = "4500mm"
Label4.Caption = "300mm"
Label5.Caption = "150mm"
Label6.Caption = "0mm"
Label7.Caption = "150mm"
Label8.Caption = "300mm"
Label9.Caption = "450mm"
Label10.Caption = "600mm"

Label11.Caption = "750mm"

Case 4

skala = 3

zoom

Label1.Caption = "1000mm"

Label2.Caption = "800mm"

Label3.Caption = "600mm"

Label4.Caption = "400mm"

Label5.Caption = "200mm"

Label6.Caption = "0mm"

Label7.Caption = "200mm"

Label8.Caption = "400mm"

Label9.Caption = "600mm"

Label10.Caption = "800mm"

Label11.Caption = "1000mm"

Case 3

skala = 2

zoom

Label1.Caption = "1500mm"

Label2.Caption = "1200mm"

Label3.Caption = "900mm"

Label4.Caption = "600mm"

Label5.Caption = "300mm"

Label6.Caption = "0mm"

Label7.Caption = "300mm"

Label8.Caption = "600mm"

Label9.Caption = "900mm"

Label10.Caption = "1200mm"

Label11.Caption = "1500mm"

```
Case 2
skala = 1
zoom
Label1.Caption = "3000mm"
Label2.Caption = "2400mm"
Label3.Caption = "1800mm"
Label4.Caption = "1200mm"
Label5.Caption = "600mm"
Label6.Caption = "0mm"
Label7.Caption = "600mm"
Label8.Caption = "1200mm"
Label9.Caption = "1800mm"
Label10.Caption = "2400mm"
Label11.Caption = "3000mm"
End Select

End Sub
---------------------------------------------------------------------------------------------------
Private Sub Command5_Click()

Picture1.Cls
Picture1.DrawWidth = 1
For grid_x = 0 To Picture1.Height Step Picture1.Height / 10
Picture1.Line (0, grid_x)-(Picture1.Width, grid_x)
Next grid_x

For grid_y = 0 To Picture1.Width Step Picture1.Width / 10
Picture1.Line (grid_y, 0)-(grid_y, Picture1.Height)
Next grid_y

Label1.Caption = "3000mm"
Label2.Caption = "2400mm"
```

```
Label3.Caption = "1800mm"
Label4.Caption = "1200mm"
Label5.Caption = "600mm"
Label6.Caption = "0mm"
Label7.Caption = "600mm"
Label8.Caption = "1200mm"
Label9.Caption = "1800mm"
Label10.Caption = "2400mm"
Label11.Caption = "3000mm"

Picture1.DrawWidth = 8
Picture1.PSet (3000, 3000), vbRed

Text1.Text = ""
skala = 1

End Sub
---------------------------------------------------------------------------------------------------
Private Sub Command6_Click()

Unload Me

End Sub
---------------------------------------------------------------------------------------------------
Private Sub form_activate()

Picture1.DrawWidth = 1
For grid_x = 0 To Picture1.Height Step Picture1.Height / 10
Picture1.Line (0, grid_x)-(Picture1.Width, grid_x)
Next grid_x

Picture1.DrawWidth = 1
```

```
For grid_y = 0 To Picture1.Width Step Picture1.Width / 10
Picture1.Line (grid_y, 0)-(grid_y, Picture1.Height)
Next grid_y

Picture1.DrawWidth = 8
Picture1.PSet (3000, 3000), vbRed

End Sub
-----------------------------------------------------------------------------------------------
Private Sub zoom()

Picture1.Cls

Picture1.DrawWidth = 1
For grid_x = 0 To Picture1.Height Step Picture1.Height / 10
Picture1.Line (0, grid_x)-(Picture1.Width, grid_x)
Next grid_x

For grid_y = 0 To Picture1.Width Step Picture1.Width / 10
Picture1.Line (grid_y, 0)-(grid_y, Picture1.Height)
Next grid_y

Picture1.DrawWidth = 8
Picture1.PSet (3000, 3000), vbRed

   InputMikro = Text1.Text
   Proses = Split(InputMikro, "#")

   Picture1.DrawWidth = 5
   q = 1

   For theta = 0 To 360 Step (360 / 96)
```

```
        Panjang(theta) = (Val(Proses(q)) * 1.08 * skala)

        q = q + 1

        x = Panjang(theta) * Sin((theta * 3.14159) / 180)
        y = Panjang(theta) * Cos((theta * 3.14159) / 180)

        X_Aksen(theta) = Val(Picture1.Width / 2 - x)
        Y_Aksen(theta) = Val(Picture1.Height / 2 - y)

        If theta < 360 Then
           Picture1.PSet (X_Aksen(theta), Y_Aksen(theta)), vbGreen

        End If

    Next theta

    Picture1.DrawWidth = 1

    For i = 0 To 360 Step (360 / 96)

        If i < (360 - (360 / 96)) Then
           Picture1.Line (X_Aksen(i), Y_Aksen(i))-(X_Aksen(i + (360 / 96)),
Y_Aksen(i + (360 / 96))), vbBlue

        ElseIf i = (360 - (360 / 96)) Then
           Picture1.Line (X_Aksen(i), Y_Aksen(i))-(X_Aksen(0), Y_Aksen(0)),
vbBlue

        Else
```

```
        End If

    Next i

End Sub
---------------------------------------------------------------------------------------------------
Private Sub Form_Load()

MSComm1.PortOpen = True
skala = 1

End Sub
---------------------------------------------------------------------------------------------------
Private Sub Timer1_Timer()

Text1.Text  = MSComm1.Input

Timer1 = False
Call drawing

End Sub
---------------------------------------------------------------------------------------------------
```

LAMPIRAN C

DATA KOMPONEN

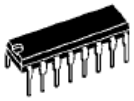
C.1 IC L293D

SGS-THOMSON MICROELECTRONICS

L293D
L293DD

PUSH-PULL FOUR CHANNEL DRIVER WITH DIODES

- 600mA OUTPUT CURRENT CAPABILITY PER CHANNEL
- 1.2A PEAK OUTPUT CURRENT (non repetitive) PER CHANNEL
- ENABLE FACILITY
- OVERTEMPERATURE PROTECTION
- LOGICAL "0" INPUT VOLTAGE UP TO 1.5 V (HIGH NOISE IMMUNITY)
- INTERNAL CLAMP DIODES



SO(12+4+4) Powerdip (12+2+2)

ORDERING NUMBERS:

L293DD L293D

DESCRIPTION

The Device is a monolithic integrated high voltage, high current four channel driver designed to accept standard DTL or TTL logic levels and drive inductive loads (such as relays solenoides, DC and stepping motors) and switching power transistors.

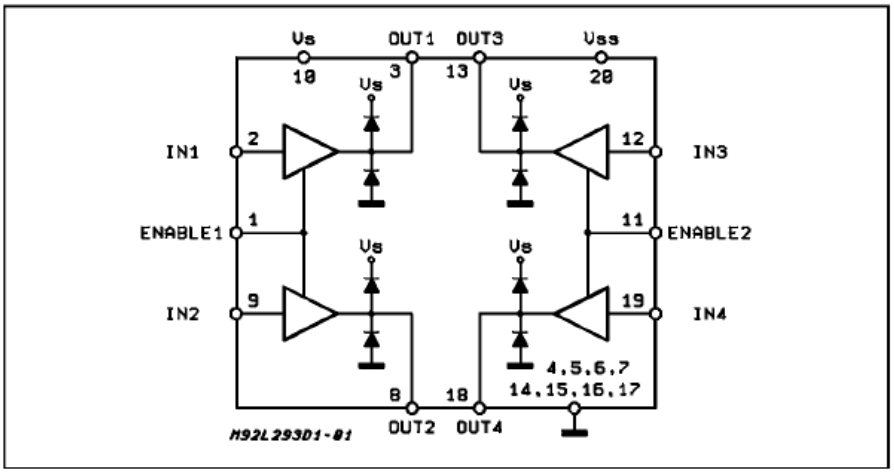
To simplify use as two bridges each pair of channels is equipped with an enable input. A separate supply input is provided for the logic, allowing operation at a lower voltage and internal clamp diodes are included.

This device is suitable for use in switching applications at frequencies up to 5 kHz.

The L293D is assembled in a 16 lead plastic packaage which has 4 center pins connected together and used for heatsinking

The L293DD is assembled in a 20 lead surface mount which has 8 center pins connected together and used for heatsinking.

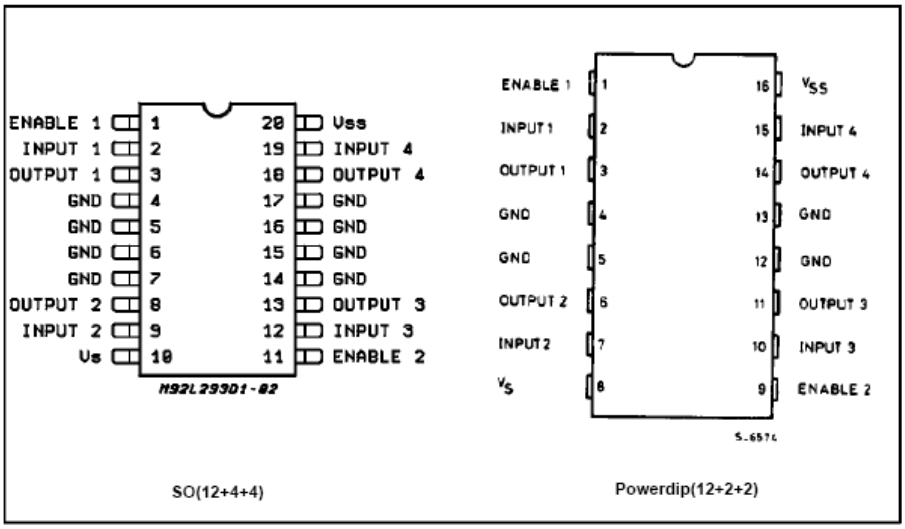
BLOCK DIAGRAM



ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
V_S	Supply Voltage	36	V
V_{SS}	Logic Supply Voltage	36	V
V_I	Input Voltage	7	V
V_{en}	Enable Voltage	7	V
I_o	Peak Output Current (100 µs non repetitive)	1.2	A
P_{tot}	Total Power Dissipation at T_{pins} = 90 °C	4	W
T_{stg}, T_j	Storage and Junction Temperature	– 40 to 150	°C

PIN CONNECTIONS (Top view)



SO(12+4+4) Powerdip(12+2+2)

THERMAL DATA

Symbol	Decription		DIP	SO	Unit
$R_{th\ j\text{-}pins}$	Thermal Resistance Junction-pins	max.	–	14	°C/W
$R_{th\ j\text{-}amb}$	Thermal Resistance junction-ambient	max.	80	50 (*)	°C/W
$R_{th\ j\text{-}case}$	Thermal Resistance Junction-case	max.	14	–	

(*) With 6sq. cm on board heatsink.

C.2 ATMEGA 16

Features

- High-performance, Low-power AVR® 8-bit Microcontroller
- Advanced RISC Architecture
  - 131 Powerful Instructions – Most Single-clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 16 MIPS Throughput at 16 MHz
  - On-chip 2-cycle Multiplier
- Nonvolatile Program and Data Memories
  - 16K Bytes of In-System Self-Programmable Flash
    Endurance: 10,000 Write/Erase Cycles
  - Optional Boot Code Section with Independent Lock Bits
    In-System Programming by On-chip Boot Program
    True Read-While-Write Operation
  - 512 Bytes EEPROM
    Endurance: 100,000 Write/Erase Cycles
  - 1K Byte Internal SRAM
  - Programming Lock for Software Security
- JTAG (IEEE std. 1149.1 Compliant) Interface
  - Boundary-scan Capabilities According to the JTAG Standard
  - Extensive On-chip Debug Support
  - Programming of Flash, EEPROM, Fuses, and Lock Bits through the JTAG Interface
- Peripheral Features
  - Two 8-bit Timer/Counters with Separate Prescalers and Compare Modes
  - One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
  - Real Time Counter with Separate Oscillator
  - Four PWM Channels
  - 8-channel, 10-bit ADC
    8 Single-ended Channels
    7 Differential Channels in TQFP Package Only
    2 Differential Channels with Programmable Gain at 1x, 10x, or 200x
  - Byte-oriented Two-wire Serial Interface
  - Programmable Serial USART
  - Master/Slave SPI Serial Interface
  - Programmable Watchdog Timer with Separate On-chip Oscillator
  - On-chip Analog Comparator
- Special Microcontroller Features
  - Power-on Reset and Programmable Brown-out Detection
  - Internal Calibrated RC Oscillator
  - External and Internal Interrupt Sources
  - Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby and Extended Standby
- I/O and Packages
  - 32 Programmable I/O Lines
  - 40-pin PDIP, 44-lead TQFP, and 44-pad MLF
- Operating Voltages
  - 2.7 - 5.5V for ATmega16L
  - 4.5 - 5.5V for ATmega16
- Speed Grades
  - 0 - 8 MHz for ATmega16L
  - 0 - 16 MHz for ATmega16

8-bit AVR® Microcontroller with 16K Bytes In-System Programmable Flash

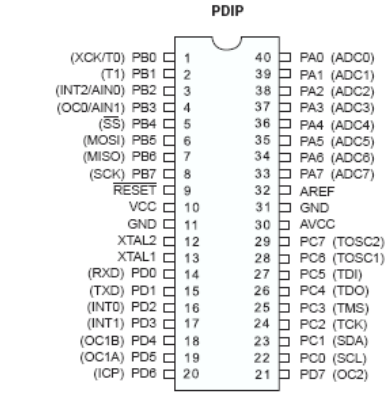
ATmega16
ATmega16L

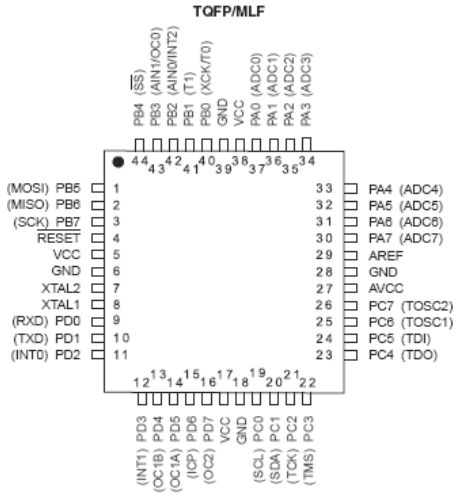
Preliminary

Rev. 2466E–AVR–10/02

Pin Configurations

Figure 1. Pinouts ATmega16





Disclaimer

Typical values contained in this data sheet are based on simulations and characterization of other AVR microcontrollers manufactured on the same process technology. Min and Max values will be available after the device is characterized.