LAMPIRAN A
Listing Program

Program pada Mikrokontroler A-1
Program pada Microsoft Visual Basic 6.0 A-10
**Listing Program pada Mikrokontroler**

---

Chip type: ATmega8535  
Clock frequency: 11.059200 MHz

---

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

int hijau1;
int hijau2;
int hijau3;
int hijau4;
char pilihan;

#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;
    }
    if(data=='Z')
    {
      pilihan = 1;
      putsf("mode 1");
    }
    if(data=='z')
    {
      pilihan = 0;
      putsf("mode 0");
    }
  }
  switch (data)
  {
    case 'A':
      hijau1 = 3000;
      putsf("HIJAU1 3 DETIK");
      break;
    case 'B':
      hijau1 = 6000;
      putsf("HIJAU1 6 DETIK");
      break;
    case 'C':
      hijau1 = 9000;
      putsf("HIJAU1 9 DETIK");
      break;
  }
}
case 'D':
    hijau2=3000;
    putsf("HIJAU2 3 DETIK");
    break;

    case 'E':
    hijau2=6000;
    putsf("HIJAU2 6 DETIK");
    break;

    case 'F':
    hijau2=9000;
    putsf("HIJAU2 9 DETIK");
    break;

    case 'G':
    hijau3=3000;
    putsf("HIJAU3 3 DETIK");
    break;

    case 'H':
    hijau3=6000;
    putsf("HIJAU3 6 DETIK");
    break;

    case 'I':
    hijau3=9000;
    putsf("HIJAU3 9 DETIK");
    break;

    case 'J':
    hijau4=3000;
    putsf("HIJAU4 3 DETIK");
    break;
case 'K':
hijau4=6000;
putsf("HIJAU4 6 DETIK");
break;

case 'L':
hijau4=9000;
putsf("HIJAU4 9 DETIK");
break;

case 'O':
hijau4=9000;
putsf("MENYALA SEMUA");
PORTA = 0xFF;
PORTB = 0xFF;
break;

case 'P':
hijau4=9000;
putsf("PADAM SEMUA");
PORTA = 0x00;
PORTB = 0x00;
break;

}
// 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=Out Func4=Out Func3=Out Func2=Out Func1=Out Func0=Out
    // State7=T State6=T State5=0 State4=0 State3=0 State2=0 State1=0 State0=0
    PORTA=0x00;
    DDRA=0xFF;

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

    // 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;
Lampiran

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")

while (1)
{
  while(pilihan == 0)
  {
    PORTA = 0x0C;
    PORTB = 0x09 ;
    delay_ms(6000);
    PORTA= 0x0A;
    delay_ms(1000);
    PORTA= 0x09;
    delay_ms(4000);
    PORTA=0x21;
    delay_ms(6000);
    PORTA=0x11;
    delay_ms(1000);
    PORTA=0x09;
    delay_ms(4000);
    PORTB=0x0C;
    delay_ms(6000);
    PORTB=0x0A;
    delay_ms(1000);
    PORTB=0x09;
    delay_ms(4000);
    PORTB=0x21;
    delay_ms(6000);
    PORTB=0x11;
    delay_ms(1000);
    PORTB=0x09;
    delay_ms(4000);
  }
}
while (pilihan == 1) {
    PORTA = 0x0C;
    PORTB = 0x09;
    delay_ms(hijau1);
    PORTA = 0x0A;
    delay_ms(1000);
    PORTA = 0x09;
    delay_ms(4000);
    PORTA = 0x21;
    delay_ms(hijau2);
    PORTA = 0x11;
    delay_ms(1000);
    PORTA = 0x09;
    delay_ms(4000);
    PORTB = 0x0C;
    delay_ms(hijau3);
    PORTB = 0x0A;
    delay_ms(1000);
    PORTB = 0x09;
    delay_ms(4000);
    PORTB = 0x21;
    delay_ms(hijau4);
    PORTB = 0x11;
    delay_ms(1000);
    PORTB = 0x09;
    delay_ms(4000);
}
Listing Program pada Microsoft Visual Basic 6.0

Option Explicit

Dim P1, P2, P3, P4 As Integer
Dim i, j, warna, r, G, b, X, luas, lebar As Integer
Dim buffx, buffy As Integer
Dim ZZ As Long
Dim x1, x2, x3, x4, x5, x6, x7, x8, x9, x10, x11, x12, x13, x14, x15, x16, y1, y2,
y3, y4, y5, y6, y7, y8, y9, y10, y11, y12, y13, y14, y15, y16 As Integer

Private Sub cmdjlr1_Click()
If cmdjlr1.Tag = "0" Then
    cmdjlr1.Tag = "1"
    cmdjlr1.Caption = "RST Jalur1"
Else
    cmdjlr1.Tag = "0"
    cmdjlr1.Caption = "Set Jalur1"
End If
End Sub

Private Sub cmdjlr2_Click()
If cmdjlr2.Tag = "0" Then
    cmdjlr2.Tag = "1"
    cmdjlr2.Caption = "RST Jalur2"
Else
    cmdjlr2.Tag = "0"
    cmdjlr2.Caption = "Set Jalur2"
End If
End Sub

Private Sub cmdjlr3_Click()
If cmdjlr3.Tag = "0" Then
    cmdjlr3.Tag = "1"
    cmdjlr3.Caption = "RST Jalur3"
Else
    cmdjlr3.Tag = "0"
    cmdjlr3.Caption = "Set Jalur3"
End If
End Sub

Private Sub cmdjlr4_Click()
If cmdjlr4.Tag = "0" Then
    cmdjlr4.Tag = "1"
    cmdjlr4.Caption = "RST Jalur4"
Else
    cmdjlr4.Tag = "0"
    cmdjlr4.Caption = "Set Jalur4"
End If
End Sub

Private Sub Command1_Click()

'================================== JALUR 1 ==================='
Dot(1).X = Text1.Text
Dot(1).Y = Text2.Text
Dot(2).X = Text3.Text
Dot(2).Y = Text4.Text

Dim panjang As Integer
Dim lebar As Integer
Dim hitam, putih As Integer

hitam = 0
putih = 0
Text17.Text = ""
panjang = Abs(Dot(1).X - Dot(2).X)
lebar = Abs(Dot(1).Y - Dot(2).Y)

For i = Dot(1).X To Dot(2).X
    For j = Dot(1).Y To Dot(2).Y
        warna = Picture1.Point(i, j)
        r = warna And RGB(255, 0, 0)
        G = Int((warna And RGB(0, 255, 0)) / 256)
        b = Int(Int((warna And RGB(0, 0, 255)) / 256) / 256)
        X = (r + G + b) / 3

        If X < 128 Then
            X = 0
            hitam = hitam + 1
        End If

        If X >= 128 Then
            X = 255
            putih = putih + 1
        End If

        Picture1.PSet (i, j), RGB(X, X, X)
    Next j
Next i
Lampiran

Text17.Text = (putih / (hitam + putih)) * 100
If Val(Text17.Text) < 30 Then
Text22.Text = "3 Detik"
ElseIf Val(Text17.Text) >= 30 And Val(Text17.Text) < 60 Then
Text22.Text = "6 detik"
ElseIf Val(Text17.Text) >= 60 Then
Text22.Text = "9 detik"
End If

'=================== JALUR 2 ===================
Dot(1).X = Text5.Text
Dot(1).Y = Text6.Text
Dot(2).X = Text7.Text
Dot(2).Y = Text8.Text

hitam = 0
putih = 0
Text18.Text = ""
panjang = Abs(Dot(1).X - Dot(2).X)
lebar = Abs(Dot(1).Y - Dot(2).Y)

For i = Dot(1).X To Dot(2).X
    For j = Dot(1).Y To Dot(2).Y
        warna = Picture1.Point(i, j)
        r = warna And RGB(255, 0, 0)
        G = Int((warna And RGB(0, 255, 0)) / 256)
        b = Int(Int((warna And RGB(0, 0, 255)) / 256) / 256)
        X = (r + G + b) / 3
        If X < 128 Then
            X = 0
            hitam = hitam + 1
        End If
        If X >= 128 Then
            X = 255
            putih = putih + 1
        End If
        Picture1.PSet (i, j), RGB(X, X, X)
    Next j
Next i

Text18.Text = (putih / (hitam + putih)) * 100
If Val(Text18.Text) < 30 Then
Lampiran

Text23.Text = "3 Detik"
ElseIf Val(Text18.Text) >= 30 And Val(Text18.Text) < 60 Then
Text23.Text = "6 detik"
ElseIf Val(Text18.Text) >= 60 Then
Text23.Text = "9 detik"
End If

'============== JALUR 3 ==================
Dot(1).X = Text9.Text
Dot(1).Y = Text10.Text
Dot(2).X = Text11.Text
Dot(2).Y = Text12.Text

hitam = 0
putih = 0
Text19.Text = ""
pnjang = Abs(Dot(1).X - Dot(2).X)
lebar = Abs(Dot(1).Y - Dot(2).Y)

For i = Dot(1).X To Dot(2).X
    For j = Dot(1).Y To Dot(2).Y
        warna = Picture1.Point(i, j)
        r = warna And RGB(255, 0, 0)
        G = Int((warna And RGB(0, 255, 0)) / 256)
        b = Int(Int((warna And RGB(0, 0, 255)) / 256) / 256)
        X = (r + G + b) / 3

        If X < 128 Then
            X = 0
            hitam = hitam + 1
        End If

        If X >= 128 Then
            X = 255
            putih = putih + 1
        End If

        Picture1.PSet (i, j), RGB(X, X, X)
    Next j
Next i

Text19.Text = (putih / (hitam + putih)) * 100
If Val(Text19.Text) < 30 Then
Text24.Text = "3 Detik"
ElseIf Val(Text19.Text) >= 30 And Val(Text19.Text) < 60 Then
Text24.Text = "6 detik"
ElseIf Val(Text19.Text) >= 60 Then
Text24.Text = "9 detik"
End If

'=================== JALUR 4 ===================
Dot(1).X = Text13.Text
Dot(1).Y = Text14.Text
Dot(2).X = Text15.Text
Dot(2).Y = Text16.Text

hitam = 0
putih = 0
Text20.Text = ""
panjang = Abs(Dot(1).X - Dot(2).X)
lebar = Abs(Dot(1).Y - Dot(2).Y)

For i = Dot(1).X To Dot(2).X
    For j = Dot(1).Y To Dot(2).Y
        warna = Picture1.Point(i, j)
        r = warna And RGB(255, 0, 0)
        G = Int((warna And RGB(0, 255, 0)) / 256)
        b = Int(Int((warna And RGB(0, 0, 255)) / 256) / 256)
        X = (r + G + b) / 3

        If X < 128 Then
            X = 0
            hitam = hitam + 1
        End If

        If X >= 128 Then
            X = 255
            putih = putih + 1
        End If

        Picture1.PSet (i, j), RGB(X, X, X)
    Next j
Next i

Text20.Text = (putih / (hitam + putih)) * 100
If Val(Text20.Text) < 30 Then
    Text25.Text = "3 Detik"
ElseIf Val(Text20.Text) >= 30 And Val(Text20.Text) < 60 Then
    Text25.Text = "6 detik"
ElseIf Val(Text20.Text) >= 60 Then

Universitas Kristen Maranatha
Text25.Text = "9 detik"
End If
End Sub

Private Sub Command2_Click()
Call TWAIN_LogFile(1)
Call TWAIN_SetHideUI(1)
Call TWAIN_SetIndicators(0)
If TWAIN_OpenSource("USB PC Camera (SN9C102P)") <> 0 Then
    Call TWAIN_SetPixelType(2)
    Call TWAIN_SetXferCount(1)
    Call TWAIN_SetAutoScan(0)
    ' If you can't use Me.hwnd, pass 0:
    Call TWAIN_AcquireToFilename(Me.hwnd, "C:\image.bmp")
End If
If TWAIN_LastErrorCode() <> 0 Then
    Call TWAIN_ReportLastError("Unable to scan.")
End If
Image1.Picture = LoadPicture("c:\image.bmp")
Picture1.Picture = Image1.Picture
End Sub

Private Sub Command5_Click()
'JALUR 1
P1 = Val(Text17.Text)

If P1 < 30 Then
    MSComm1.Output = "Z" 'AKTIFKAN PILIHAN MODE 1
    While MSComm1.InBufferCount = 0
        Text21.Text = " "
    Wend
    Text21.Text = MSComm1.Input
    For ZZ = 1 To 1000000: Next ZZ
    MSComm1.Output = "A"
    While MSComm1.InBufferCount = 0
        Text21.Text = " "
    Wend
    Text21.Text = MSComm1.Input
    For ZZ = 1 To 1000000: Next ZZ
ElseIf (P1 > 30) And (P1 < 60) Then
    MSComm1.Output = "B"
    While MSComm1.InBufferCount = 0
        Text21.Text = " "
    Wend
    Text21.Text = MSComm1.Input
    For ZZ = 1 To 1000000: Next ZZ
End If
ElseIf P1 > 60 Then
    MSComm1.Output = "C"
    While MSComm1.InBufferCount = 0
        Text21.Text = " "
    Wend
    Text21.Text = MSComm1.Input
    For ZZ = 1 To 1000000: Next ZZ
End If

'JALUR 2
P2 = Val(Text18.Text)

If P2 < 30 Then
    MSComm1.Output = "Z" 'AKTIFKAN PILIHAN MODE 1
    While MSComm1.InBufferCount = 0
        Text21.Text = " "
    Wend
    Text21.Text = MSComm1.Input
    For ZZ = 1 To 1000000: Next ZZ
    MSComm1.Output = "D"
    While MSComm1.InBufferCount = 0
        Text21.Text = " "
    Wend
    Text21.Text = MSComm1.Input
    For ZZ = 1 To 1000000: Next ZZ
    MSComm1.Output = "F"
    While MSComm1.InBufferCount = 0
        Text21.Text = " "
    Wend
    Text21.Text = MSComm1.Input
    For ZZ = 1 To 1000000: Next ZZ
ElseIf (P2 > 30) And (P2 < 60) Then
    MSComm1.Output = "E"
    While MSComm1.InBufferCount = 0
        Text21.Text = " "
    Wend
    Text21.Text = MSComm1.Input
    For ZZ = 1 To 1000000: Next ZZ
ElseIf P2 > 60 Then
    MSComm1.Output = "F"
    While MSComm1.InBufferCount = 0
        Text21.Text = " "
    Wend
    Text21.Text = MSComm1.Input
    For ZZ = 1 To 1000000: Next ZZ
End If

'JALUR 3
P3 = Val(Text19.Text)
If P3 < 30 Then
Lampiran

MSComm1.Output = "Z" 'AKTIFKAN PILIHAN MODE 1
While MSComm1.InBufferCount = 0
  Text21.Text = " "
  Wend
Text21.Text = MSComm1.Input
For ZZ = 1 To 1000000: Next ZZ
MSComm1.Output = "G"
While MSComm1.InBufferCount = 0
  Text21.Text = " "
  Wend
Text21.Text = MSComm1.Input
For ZZ = 1 To 1000000: Next ZZ

ElseIf (P3 > 30) And (P3 < 60) Then
  MSComm1.Output = "H"
  While MSComm1.InBufferCount = 0
    Text21.Text = " "
    Wend
  Text21.Text = MSComm1.Input
  For ZZ = 1 To 1000000: Next ZZ
End If

'JALUR 4
P4 = Val(Text20.Text)

If P4 < 30 Then
  MSComm1.Output = "J"
  While MSComm1.InBufferCount = 0
    Text21.Text = " "
    Wend
  Text21.Text = MSComm1.Input
  For ZZ = 1 To 1000000: Next ZZ
End If

ElseIf (P4 > 30) And (P4 < 60) Then
  MSComm1.Output = "Z" 'AKTIFKAN PILIHAN MODE 1
  While MSComm1.InBufferCount = 0
    Text21.Text = " "
    Wend
  Text21.Text = MSComm1.Input

A-17
For ZZ = 1 To 1000000: Next ZZ

MSComm1.Output = "K"
While MSComm1.InBufferCount = 0
    Text21.Text = " "
Wend
Text21.Text = MSComm1.Input
For ZZ = 1 To 1000000: Next ZZ

ElseIf P4 > 60 Then
    MSComm1.Output = "Z"  'AKTIFKAN PILIHAN MODE 1
    While MSComm1.InBufferCount = 0
        Text21.Text = " "
    Wend
    Text21.Text = MSComm1.Input
    For ZZ = 1 To 1000000: Next ZZ
    MSComm1.Output = "L"
    While MSComm1.InBufferCount = 0
        Text21.Text = " "
    Wend
    Text21.Text = MSComm1.Input
    For ZZ = 1 To 1000000: Next ZZ
End If
End Sub

Private Sub Form_Load()
    MSComm1.CommPort = 1
    MSComm1.Settings = "9600,N,8,1"
    MSComm1.InputLen = 0
    MSComm1.PortOpen = True
End Sub

Private Sub Picture1_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
    Call MouseDown
    buffx = X
    buffy = Y
    Picture1.CurrentX = buffx
    Picture1.CurrentY = buffy

    'jalur satu
    If cmdjlr1.Tag = "2" Then
        x2 = buffx
        y2 = buffy
        cmdjlr1.Tag = "0"
        Text3.Text = x2
    End If
End Sub
Text4.Text = y2
cmdjlr1.Caption = "Set Jalur1"
x3 = x2
y3 = y1
x4 = x1
y4 = y2
End If
If cmdjlr1.Tag = "1" Then
  x1 = buffx
  y1 = buffy
  cmdjlr1.Tag = "2"
  Text1.Text = x1
  Text2.Text = y1
End If
'jalur dua
If cmdjlr2.Tag = "2" Then
  x6 = buffx
  y6 = buffy
  cmdjlr2.Tag = "0"
  Text7.Text = x6
  Text8.Text = y6
  cmdjlr2.Caption = "Set Jalur2"
  x7 = x6
  y7 = y5
  x8 = x5
  y8 = y6
End If
If cmdjlr2.Tag = "1" Then
  x5 = buffx
  y5 = buffy
  cmdjlr2.Tag = "2"
  Text5.Text = x5
  Text6.Text = y5
End If
'jalur tiga
If cmdjlr3.Tag = "2" Then
  x10 = buffx
  y10 = buffy
  cmdjlr3.Tag = "0"
  Text11.Text = x10
  Text12.Text = y10
  cmdjlr3.Caption = "Set Jalur3"
  x11 = x10
y11 = y9
x12 = x9
y12 = y10
End If

If cmdjlr3.Tag = "1" Then
    x9 = buffx
    y9 = buffy
    cmdjlr3.Tag = "2"
    Text9.Text = x9
    Text10.Text = y9
End If

'jalur empat
If cmdjlr4.Tag = "2" Then
    x14 = buffx
    y14 = buffy
    cmdjlr4.Tag = "0"
    Text15.Text = x14
    Text16.Text = y14
    cmdjlr4.Caption = "Set Jalur4"
    x15 = x14
    y15 = y13
    x16 = x13
    y16 = y14
End If

If cmdjlr4.Tag = "1" Then
    x13 = buffx
    y13 = buffy
    cmdjlr4.Tag = "2"
    Text13.Text = x13
    Text14.Text = y13
End If

End Sub

Private Sub Picture1_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)

Dim buffx As Integer, buffy As Integer

If writelet = True Then
    buffx = X
    buffy = Y
End If
Lampiran

    Label1.Caption = buffx
    Label2.Caption = buffy

End If

End Sub

Private Sub Picture1_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
    writelet = False
    End Sub

Private Sub Command3_Click()
    End
    End Sub

Private Sub Timer1_Timer()
    Call Command2_Click
    End Sub

Private Sub Timer2_Timer()
    Call Command1_Click
    End Sub

Private Sub Timer3_Timer()
    Call Command5_Click
    End Sub
Modules:

1. **EZTwain.bas**
2. **Module.bas**

Public Type Node
   X As Integer
   Y As Integer
End Type

Public Dot(1 To 2) As Node

Public writelet As Boolean
Public Sub MouseDown()

   writelet = True
   HoldX = X
   HoldY = Y
End Sub
LAMPIRAN B

FOTO ALAT
Gambar B.1 Maket Tampak Atas

Gambar B.2 Maket Tampak Samping
Gambar B.3 Gambar Keseluruhan
LAMPIRAN C

Datasheet ATMega8535 C-1
Datasheet MAX232 C-8
Features
- High-performance, Low-power AVR<sup>®</sup> 8-bit Microcontroller
- Advanced RISC Architecture
  - 130 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
  - 8K Bytes 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
  - 512 Bytes Internal SRAM
- Programming Lock for Software Security
- Peripheral Features
  - Two 8-bit Timers/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 A/D
  - 8 Single-ended Channels
  - 7 Differential Channels for TSFP Package Only
  - 2 Differential Channels with Programmable Gain at 1x, 10x, or 200x for TSFP Package Only
  - 8-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, A/D Noise Reduction, Power-save, Power-down, Standby and Extended Standby
- I/O and Packages
  - 32 Programmable I/O Lines
  - 40-pin PDIP, 44-pin TSFP, 44-pin PLCC, and 44-pad QFNMLF
- Operating Voltages
  - 2.7 – 5.5V for ATmega835L
  - 4.5 – 5.5V for ATmega835
- Speed Grades
  - 0 – 8 MHz for ATmega835L
  - 0 – 16 MHz for ATmega835

Summary

ATmega8535
ATmega8535L

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.
The ATmega8535 is a low-power CMOS 8-bit microcontroller based on the AVR enhanced RISC architecture. By executing instructions in a single clock cycle, the ATmega8535 achieves throughputs approaching 1 MIPS per MHz, allowing the system designer to optimize power consumption versus processing speed.

The AVR core combines a rich instruction set with 32 general purpose working registers. All 32 registers are directly connected to the Arithmetic Logic Unit (ALU), allowing two independent registers to be accessed in one single instruction executed in one clock cycle. The resulting architecture is more code efficient while achieving throughputs up to ten times faster than conventional CISC microcontrollers.

The ATmega8535 provides the following features: 8K bytes of In-System Programmable Flash with Read-While-Write capabilities, 512 bytes EEPROM, 512 bytes SRAM, 32 general purpose I/O lines, 32 general purpose working registers, three flexible Timer/Counters with compare modes, internal and external interrupts, a serial programmable USART, a byte oriented Two-wire Serial Interface, an 8-channel, 10-bit ADC with optional differential input stage with programmable gain in TOPP package, a programmable Watchdog Timer with Internal Oscillator, an SPI serial port, and six software selectable power saving modes. The Idle mode stops the CPU while allowing the SRAM, Timer/Counters, SPI port, and interrupt system to continue functioning. The Power-down mode saves the register contents but freezes the Oscillator, disabling all other chip functions until the next interrupt or Hardware Reset. In Power-save mode, the asynchronous timer continues to run, allowing the user to maintain a timer base while the rest of the device is sleeping. The ADC Noise Reduction mode stops the CPU and all I/O modules except asynchronous timer and ADC, to minimize switching noise during ADC conversions. In Standby mode, the crystal/resonator Oscillator is running while the rest of the device is sleeping. This allows very fast start-up combined with low-power consumption. In Extended Standby mode, both the main Oscillator and the asynchronous timer continue to run.

The device is manufactured using Atmel's high density nonvolatile memory technology. The On-chip ISP Flash allows the program memory to be reprogrammed In-System through an SPI serial interface, by a conventional nonvolatile memory programmer, or by an On-chip Boot program running on the AVR core. The boot program can use any interface to download the program application in the Application Flash memory. Software in the Boot Flash section will continue to run while the Application Flash section is updated, providing true Read-While-Write operation. By combining an 8-bit RISC CPU with In-System Self-Programmable Flash on a monolithic chip, the Atmel ATmega8535 is a powerful microcontroller that provides a highly flexible and cost effective solution to many embedded control applications.

The ATmega8535 AVR is supported with a full suite of program and system development tools including: C compilers, macro assemblers, program debugger/simulators, In-Circuit Emulators, and evaluation kits.

AT90S8535 Compatibility

The ATmega8535 provides all the features of the AT90S8535. In addition, several new features are added. The ATmega8535 is backward compatible with AT90S8535 in most cases. However, some incompatibilities between the two microcontrollers exist. To solve this problem, an AT90S8535 compatibility mode can be selected by programming the S8535C fuse. ATmega8535 is pin compatible with AT90S8535, and can replace the AT90S8535 on current Printed Circuit Boards. However, the location of fuse bits and the electrical characteristics differ between the two devices.

Programming the S8535C fuse will change the following functionally:
- The timed sequence for changing the Watchdog Timer-out period is disabled. See "Timed Sequences for Changing the Configuration of the Watchdog Timer" on page 45 for details.
- The double buffering of the USART Receive Register is disabled. See "AVR USART vs. AVR UART – Compatibility" on page 146 for details.
Pin Descriptions

\( \text{Vcc} \)
Digital supply voltage.

\( \text{GND} \)
Ground.

Port A (PA0..PA7)
Port A serves as the analog inputs to the A/D Converter.
Port A also serves as an 8-bit bi-directional I/O port, if the A/D Converter is not used. Port A can provide internal pull-up resistors (selected for each bit). The Port A output buffers have symmetrical drive characteristics with both high sink and source capability. When pins PA0 to PA7 are used as inputs and are externally pulled low, they will source current if the internal pull-up resistors are activated. The Port A pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port B (PB0..PB7)
Port B is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port B output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port B pins that are externally pulled low will source current if the pull-up resistors are activated. The Port B pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port C (PC0..PC7)
Port C is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port C output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port C pins that are externally pulled low will source current if the pull-up resistors are activated. The Port C pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port D (PD0..PD7)
Port D is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port D output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port D pins that are externally pulled low will source current if the pull-up resistors are activated. The Port D pins are tri-stated when a reset condition becomes active, even if the clock is not running.

RESET
Reset Input. A low level on this pin for longer than the minimum pulse length will generate a reset, even if the clock is not running. The minimum pulse length is given in Table 15 on page 37. Shorter pulses are not guaranteed to generate a reset.

XTAL1
Input to the inverting Oscillator amplifier and input to the internal clock operating circuit.

XTAL2
Output from the inverting Oscillator amplifier.

AVCC
AVCC is the supply voltage pin for Port A and the A/D Converter. It should be externally connected to \( \text{Vcc} \), even if the ADC is not used. If the ADC is used, it should be connected to \( \text{Vcc} \) through a low-pass filter.

AREF
AREF is the analog reference pin for the A/D Converter.
done, writing a byte to the SPI Data Register starts the SPI Clock Generator, and the hardware shifts the eight bits into the Slave. After shifting one byte, the SPI clock generator stops, setting the end of Transmission Flag (SPIF). If the SPI Interrupt Enable bit (SPIE) in the SPCR Register is set, an interrupt is requested. The Master may continue to shift the next byte by writing it into SPDR, or signal the end of packet by pulling high the Slave Select, SS line. The last incoming byte will be kept in the buffer register for later use.

When configured as a Slave, the SPI interface will remain sleeping with MISO tri-stated as long as the SS pin is driven high. In this state, software may update the contents of the SPI Data Register, SPDR, but the data will not be shifted out by incoming clock pulses on the SCK pin until the SS pin is driven low. As one byte has been completely shifted, the end of Transmission Flag, SPIF is set. If the SPI Interrupt Enable bit, SPIE, in the SPCR Register is set, an interrupt is requested. The Slave may continue to place new data to be sent into SPDR before reading the incoming data. The last incoming byte will be kept in the buffer register for later use.

**Figure 66. SPI Master-Slave Interconnection**

The system is single buffered in the transmit direction and double buffered in the receive direction. This means that bytes to be transmitted cannot be written to the SPI Data Register before the entire shift cycle is completed. When receiving data, however, a received character must be read from the SPI Data Register before the next character has been completely shifted in. Otherwise, the first byte is lost.

In SPI Slave mode, the control logic will sample the incoming signal of the SCK pin. To ensure correct sampling of the clock signal, the minimum low and high periods should be:

- Low period: Longer than 2 CPU clock cycles.
- High period: Longer than 2 CPU clock cycles.

When the SPI is enabled, the data direction of the MOSI, MISO, SCK, and SS pins is overridden according to Table 56 on page 138. For more details on automatic port overrides, refer to “Alternate Port Functions” on page 57.

**Table 56. SPI Pin Overrides**

<table>
<thead>
<tr>
<th>Pin</th>
<th>Direction, Master SPI</th>
<th>Direction, Slave SPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOSI</td>
<td>User Defined</td>
<td>Input</td>
</tr>
<tr>
<td>MISO</td>
<td>Input</td>
<td>User Defined</td>
</tr>
<tr>
<td>SCK</td>
<td>User Defined</td>
<td>Input</td>
</tr>
<tr>
<td>SS</td>
<td>User Defined</td>
<td>Input</td>
</tr>
</tbody>
</table>

Note: 1. See “Alternate Functions Of Port B” on page 66 for a detailed description of how to define the direction of the user-defined SPI pins.

The following code examples show how to initialize the SPI as a Master and how to perform a simple transmission. DDR_SPI in the examples must be replaced by the actual Data Direction Register controlling the SPI pins. DD_MOSI, DD_MISO and DD_SCK must be replaced by the actual data direction bits for these pins. For example, if MOSI is placed on pin PB6, replace DD_MOSI with DDB6, and DDR_SPI with DDRB.

**Assembly Code Example**

---

Universitas Kristen Maranatha
The following code examples show how to initialize the SPI as a Slave and how to perform a simple reception.

**Assembly Code Example**

```assembly
; Set MISO output, all others input
li r17, 0x0000001F
out DDR_SPI, r17

; Enable SPI, Master, set clock rate 19.66 MHz
li r17, 0x00000020
or 0x80, r17
out SPIR, r17

; Start transmission of data [16]
out SPDR, 16

; Wait for transmission complete
sbis SPISR, SPIF
jr sp

; Read received data and return
in r16, SPDR
jr sp
```

**C Code Example**

```c
void SPI_MasterInit(void) {
    // Set MOSI and SCK output, all others input
    DDR_SPI = 0x00000001 | 0x00000005;
    // Enable SPI, Master, set clock rate 19.66 MHz
    SPIR = 0x01 | 0x00000008;
}

void SPI_MasterTransmit(char cData) {
    // Start transmission
    SPDR = cData;
    // Wait for transmission complete
    while((SPISR & 0x00000001) == 0)
}
```

**Assembly Code Example**

```assembly
; Set MISO output, all others input
li r17, 0x0000001F
out DDR_SPI, r17

; Enable SPI
li r17, 0x00000008
out SPIR, r17

; Read received data and return
in r16, SPDR
jr sp
```

**C Code Example**

```c
void SPI_SlaveInit(void) {
    // Set MISO output, all others input
    DDR_SPI = 0x00000001 | 0x00000005;
    // Enable SPI
    SPIR = 0x08;
}

void SPI_SlaveTransmit(char cData) {
    // Start transmission
    SPDR = cData;
    // Wait for reception complete
    while((SPISR & 0x00000001) == 0)
    // Return Data Register
    return SPDR;
}
```

SS Pin Functionality

Slave Mode
When the SPI is configured as a Slave, the Slave Select (SS) pin is always input. When SS is held low, the SPI is activated, and MOSI becomes an output if configured as an output. All other pins are inputs. When SS is driven high, all pins are inputs, and the SPI is passive, which means that it will not receive incoming data. Note that the SPI logic will be reset once the SS pin is driven high.

The SS pin is useful for packet-style synchronization to keep the Slave bit counter synchronous with the Master clock generator. When the SS pin is driven high, the SPI Slave will immediately reset the send and receive logic, and drop any previously received data in the Shift Register.

Master Mode
When the SPI is configured as a Master (MSTR in SPCR is set), the user can determine the direction of the SS pin.

If SS is configured as an output, the pin is a general output pin which does not affect the SPI system. Typically, the pin will be driving the SS pin of the SPI Slave.

If SS is configured as an input, it must be held high to ensure Master SPI operation. If the SS pin is driven low by peripheral circuitry when the SPI is configured as a Master with the SS pin defined as an input, the SPI system interprets this as another Master selecting the SPI as a Slave and starting to send data to it. To avoid bus contention, the SPI system takes the following actions:
1. The MSTR bit in SPCR is cleared and the SPI system becomes a Slave. As a result of the SPI becoming a Slave, the MOSI and SCK pins become inputs.
2. The SPIF Flag in SPSR is set, and if the SPI interrupt is enabled, and the I-bit in SREG is set, the interrupt routine will be executed.

Thus, when interrupt-driven SPI transmission is used in Master mode, there exists a possibility that SS is driven low, the interrupt should always check that the MSTR bit is still set. If the MSTR bit has been cleared by a Slave Select, it must be set by the user to re-enable SPI Master mode.

SPI Control Register – SPCR

<table>
<thead>
<tr>
<th>Bit</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>FILE</td>
<td>SPE</td>
<td>SPE</td>
<td>DORD</td>
<td>MSTR</td>
<td>CRP</td>
<td>CPOL</td>
<td>CPHA</td>
<td>SPSR</td>
<td>SPCR</td>
</tr>
<tr>
<td>Value</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
</tr>
</tbody>
</table>

- **Bit 7 – SPIE: SPI Interrupt Enable**
  This bit causes the SPI interrupt to be executed if SPIF bit in the SPSR Register is set and if the Global Interrupt Enable bit in SREG is set.

- **Bit 6 – SPE: SPI Enable**
  When the SPE bit is written to one, the SPI is enabled. This bit must be set to enable any SPI operations.

- **Bit 5 – DORD: Data Order**
  When the DORD bit is written to one, the LSB of the data word is transmitted first.

- **Bit 4 – MSTR: Master/Slave Select**
  This bit selects Master SPI mode when written to one, and Slave SPI mode when written to zero. If SS is configured as an input and is driven low while MSTR is set, MSTR will

be cleared, and SPIF in SPSR will become set. The user will then have to set MSTR to re-enable SPI Master mode.

- **Bit 3 – CPOL: Clock Polarity**
  When this bit is written to one, SCK is high when idle. When COPOL is written to zero, SCK is low when idle. Refer to Figure 67 and Figure 68 for an example. The CPOL functionality is summarized below:

<table>
<thead>
<tr>
<th>CPOL</th>
<th>Leading Edge</th>
<th>Trailing Edge</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Rising</td>
<td>Falling</td>
</tr>
<tr>
<td>1</td>
<td>Falling</td>
<td>Rising</td>
</tr>
</tbody>
</table>

- **Bit 2 – CPHA: Clock Phase**
  The settings of the Clock Phase bit (CPHA) determine if data is sampled on the leading (first) or trailing (last) edge of SCK. Refer to Figure 67 and Figure 68 for an example. The CPHA functionality is summarized below:

<table>
<thead>
<tr>
<th>CPHA</th>
<th>Leading Edge</th>
<th>Trailing Edge</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Sample</td>
<td>Setup</td>
</tr>
<tr>
<td>1</td>
<td>Setup</td>
<td>Sample</td>
</tr>
</tbody>
</table>

- **Bits 1, 0 – SPR1, SPR0: SPI Clock Rate Select 1 and 0**
  These two bits control the SCK rate of the device configured as a Master. SPR1 and SPR0 have no effect on the Slave. The relationship between SCK and the Oscillator Clock Frequency \( f_{OSC} \) is shown in the following table:

<table>
<thead>
<tr>
<th>SPRX</th>
<th>SPR1</th>
<th>SPR0</th>
<th>SCK Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>( f_{OSC}/4 )</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>( f_{OSC}/16 )</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>( f_{OSC}/64 )</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>( f_{OSC}/128 )</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>( f_{OSC}/2 )</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>( f_{OSC}/8 )</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>( f_{OSC}/32 )</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>( f_{OSC}/64 )</td>
</tr>
</tbody>
</table>
Data Modes

There are four combinations of SCK phase and polarity with respect to serial data, which are determined by control bits CPHA and CPOL. The SPI data transfer formats are shown in Figure 67 and Figure 68. Data bits are shifted out and latched in on opposite edges of the SCK signal, ensuring sufficient time for data signals to stabilize. This is clearly seen by summarizing Table 57 and Table 58, as done below:

Table 60. CPOL Functionality

<table>
<thead>
<tr>
<th>CPOL, CPHA</th>
<th>Leading Edge</th>
<th>Trailing Edge</th>
<th>SPI Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPOL=0, CPHA=0</td>
<td>Sample (Rising)</td>
<td>Setup (Falling)</td>
<td>0</td>
</tr>
<tr>
<td>CPOL=0, CPHA=1</td>
<td>Setup (Rising)</td>
<td>Sample (Falling)</td>
<td>1</td>
</tr>
<tr>
<td>CPOL=1, CPHA=0</td>
<td>Sample (Falling)</td>
<td>Setup (Rising)</td>
<td>2</td>
</tr>
<tr>
<td>CPOL=1, CPHA=1</td>
<td>Setup (Falling)</td>
<td>Sample (Rising)</td>
<td>3</td>
</tr>
</tbody>
</table>

Figure 67. SPI Transfer Format with CPHA = 0

Figure 68. SPI Transfer Format with CPHA = 1


**General Description**

The MAX2023E-MAX203E, MAX221E-MAX241E line drivers/receivers are designed for RS-232 and V.28 communications in harsh environments. Each transmitter output and receiver input is protected against a ±15 kV electrostatic discharge (ESD) shock, without latchup. The various combinations of features are outlined in the Selector Guide. The drivers and receivers for all ten devices meet all EIA/TIA-232-D and CCITT V.28 specifications at data rates up to 130 kbps, when loaded in accordance with the EIA/TIA-232-E specification.

The MAX21E/MAX23E/MAX241E are available in 20-pin SO packages, as well as a 28-pin SSOP that uses 60% less board space. The MAX203E/MAX204E come in 16-pin TSSOP, narrow SO, wide SO, and DIP packages. The MAX203E comes in a 20-pin SSOP package, and needs no external charge-pump capacitors. The MAX203E comes in a 24-pin SSOP package, and also eliminates external charge-pump capacitors. The MAX203E/MAX204E come in 24-pin SO, SSOP, and narrow DIP packages. The MAX21E/MAX241E, operate with four 10 pF capacitors, while the MAX203E/MAX204E/MAX205E/MAX233E/MAX234E, operate with four 10 pF capacitors, further reducing cost and board space.

**Applications**

- Notebook/Notebook and Palmtop Computers
- Battery-Powered Equipment
- Hand-Held Equipment

**Pin Configurations**

Pin Configurations continued at end of data sheet.

**Selector Guide**

<table>
<thead>
<tr>
<th>PART</th>
<th>No. of RS-232 DRIVERS</th>
<th>No. of RS-232 RECEIVERS</th>
<th>RECEIVERS ACTIVE IN SHUTDOWN</th>
<th>LOW-POWER SHUTDOWN</th>
<th>TTL THREE-STATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAX21E</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>4 (10 pF)</td>
<td>No</td>
</tr>
<tr>
<td>MAX23E</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>4 (10 pF)</td>
<td>No</td>
</tr>
<tr>
<td>MAX241E</td>
<td>5</td>
<td>5</td>
<td>0</td>
<td>4 (10 pF)</td>
<td>Yes</td>
</tr>
<tr>
<td>MAX233E</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>4 (10 pF)</td>
<td>Yes</td>
</tr>
<tr>
<td>MAX234E</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>4 (10 pF)</td>
<td>Yes</td>
</tr>
<tr>
<td>MAX221E</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>4 (10 pF)</td>
<td>Yes</td>
</tr>
<tr>
<td>MAX2241E</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>4 (10 pF)</td>
<td>Yes</td>
</tr>
</tbody>
</table>

LapLink is a registered trademark of Telebit Software, Inc.

**ABSOLUTE MAXIMUM RATINGS**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>SYMBOL</th>
<th>CONDITION</th>
<th>MIN</th>
<th>TYP</th>
<th>MAX</th>
<th>UNITS</th>
</tr>
</thead>
<tbody>
<tr>
<td>DC CHARACTERISTICS</td>
<td>VCC Supply Current</td>
<td>Icc</td>
<td>No load, Tj = +25°C</td>
<td>MAX2023E</td>
<td>8</td>
<td>mA</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>MAX203E</td>
<td>11</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>MAX211E</td>
<td>14</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>MAX232E</td>
<td>5</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>MAX231E</td>
<td>7</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>MAX221E</td>
<td>1</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>MAX2241E</td>
<td>1</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>MAX221E</td>
<td>15</td>
<td>50</td>
<td></td>
</tr>
</tbody>
</table>

**LOGIC**

- Input Pull-Up Current | T_IN | 6 V (MAX2023E: 3.3 ±0.1 V) | 15 | 200 | μA |
- Input Leakage Current | T_IN | 0 V to VCC (MAX2023E: 3.3 ±0.1 V) | 10 | | μA |
- Input Threshold Low | VIL | 0 V to VCC (MAX2023E: 3.3 ±0.1 V) | 0.8 | | V |
- Input Threshold High | VIH | EN, SHDN (MAX2023E: 3.3 ±0.1 V) | 2.0 | | V |
- Output Voltage Low | VOL | R.OUT | 0.3 V (MAX2023E: 2.0 ±0.1 V) | 0.4 | | V |
- Output Voltage High | VOH | R.OUT | 0.3 V (MAX2023E: 2.0 ±0.1 V) | 3.5 | | μA |
- Output Leakage Current | EN = VCC | 3 V to 5.5 V, R.OUT ≤ 5 V | 40.0 | 10 | μA |

For pricing, delivery, and ordering information, please contact Maxim/Dallas Direct! at 1-888-629-4642, or visit Maxim’s website at www.maxim-ic.com.
\[ ±15kV \text{ ESD-Protected, } +5V \text{ RS-232 Transceivers} \]

**Applications Information**

**Capacitor Selection**

The capacitor type used for C1-C4 is not critical for proper operation. The MAX202E, MAX206-MAX206E, MAX213E, and MAX213E require 0.1\(\mu F\) capacitors and the MAX223E and MAX241E require 1\(\mu F\) capacitors, although in all cases capacitors up to 10\(\mu F\) can be used without harm. Ceramic, aluminum-electrolytic, or tantalum capacitors are suggested for the 0.1\(\mu F\) capacitors, and ceramic disc capacitors are suggested for the 1\(\mu F\) capacitors. When using the minimum recommended capacitor values, make sure the capacitance value does not degrade excessively as the operating temperature varies. If in doubt, use capacitors with a larger (e.g., 2X) nominal value. The capacitors' effective series resistance (ESR), which usually rises at low temperatures, influences the amount of ripple on +V and -V.

**Driving Multiple Receivers**

Each transmitter is designed to drive a single receiver. Transmitters can be paralleled to drive multiple receivers.

**Driver Outputs When Exiting Shutdown**

The driver outputs display no ringing or undesirable transients as they come out of shutdown.

**High Data Rates**

These transceivers maintain the RS-232 ±5.0V minimum driver output voltages at data rates of over 120kbs. For data rates above 120kbs, refer to the Transmitter Output Voltage vs. Load Capacitance graphs in the typical Operating Characteristics. Communication at these high rates is easier if the capacitive loads on the transmitters are small; i.e., short cables are best.

**Table 2. Summary of EIA/TIA-232E, V.28 Specifications**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>CONDITIONS</th>
<th>EIA/TIA-232E, V.28 SPECIFICATIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Driver Output Voltage</td>
<td>0 Level</td>
<td>3kΩ to 7kΩ load</td>
</tr>
<tr>
<td></td>
<td>1 Level</td>
<td>3kΩ to 7kΩ load</td>
</tr>
<tr>
<td>Driver Output Level, Max</td>
<td>No load</td>
<td>±25V</td>
</tr>
<tr>
<td>Data Rate</td>
<td></td>
<td>Up to 20kbaud</td>
</tr>
<tr>
<td>Receiver Input Voltage</td>
<td>0 Level</td>
<td>3kΩ to RL ≤ 7kΩ, CL ≤ 2500pF</td>
</tr>
<tr>
<td></td>
<td>1 Level</td>
<td>3kΩ to RL ≤ 7kΩ, CL ≤ 2500pF</td>
</tr>
<tr>
<td>Receiver Input Level</td>
<td></td>
<td>±25V</td>
</tr>
<tr>
<td>Instantaneous Short-Circuit Rate, Max</td>
<td>3kΩ to RL ≤ 7kΩ, CL ≤ 2500pF</td>
<td>300mA</td>
</tr>
<tr>
<td>Driver Output Short-Circuit Current, Max</td>
<td></td>
<td>100mA</td>
</tr>
<tr>
<td>Transition Rate on Driver Output</td>
<td></td>
<td>±2V or ±10V or ±2V</td>
</tr>
<tr>
<td>Driver Output Resistance</td>
<td></td>
<td>30kΩ</td>
</tr>
</tbody>
</table>

**Notes:**
- MAX213E, ... tested with VCC = +5V ±5%.
- ESD Protection Voltage:
  - Human Body Model: ±15 kV
  - IEC 61000-4-2, Contact Discharge: ±8 kV
  - IEC 61000-4-2, Air Gap Discharge: ±15 kV
±15kV ESD-Protected, +5V RS-232 Transceivers

Table 3. DB9 Cable Connections Commonly Used for EIA/TIAE-232E and V.24 Asynchronous Interfaces

<table>
<thead>
<tr>
<th>PIN</th>
<th>CONNECTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Received Line Signal Detector (sometimes called Carrier Detect DC) Handshake from DCE</td>
</tr>
<tr>
<td>2</td>
<td>Receive Data (RX) Data from DCE</td>
</tr>
<tr>
<td>3</td>
<td>Transmit Data (TX) Data from DTE</td>
</tr>
<tr>
<td>4</td>
<td>Data Terminal Ready Handshake from DTE</td>
</tr>
<tr>
<td>5</td>
<td>Signal Ground Reference point for signals</td>
</tr>
<tr>
<td>6</td>
<td>Data Set Ready (DSR) Handshake from DCE</td>
</tr>
<tr>
<td>7</td>
<td>Request to Send (RTS) Handshake from DTE</td>
</tr>
<tr>
<td>8</td>
<td>Clear to Send (CTS) Handshake from DCE</td>
</tr>
<tr>
<td>9</td>
<td>Ring Indicator Handshake from DCE</td>
</tr>
</tbody>
</table>

*Pin Configurations and Typical Operating Circuits (continued)*
LAMPIRAN D

DOKUMENTASI

D-1
PENGUJIAN 18

PENGUJIAN 19

PENGUJIAN 20

PENGUJIAN 21
PENGUJIAN 22

PENGUJIAN 23

PENGUJIAN 24

PENGUJIAN 25

Universitas Kristen Maranatha
LAMPIRAN E

EZTwain Pro User Guide
EZTwain Pro
User Guide

A guide to the EZTwain library for developers.

Version 3.09

By Spike McLarty for Dosadi. Revised 1/27/2006 12:44 PM
Copyright © 2003-2006 by Dosadi. All rights reserved.
EZTwain, EZTwain Pro and Dosadi are trademarks of Dosadi. Microsoft and Windows are registered trademarks of Microsoft. Other trademarks are the property of their respective owners.
TWAIN_LogFile
void TWAIN_LogFile(int fLog);
EZTwain can write a quite detailed log of its activity, including every TWAIN call it makes and the result. Log output goes to \texttt{c:}\textbackslash\texttt{eztwain.log}. You can use TWAIN_SetLogFolder to direct the log file to another directory.
TWAIN_LogFile(0) close log file and turn off logging
TWAIN_LogFile(1) open log file (if not already) and start logging.
If logging is already turned on, TWAIN_LogFile(1) flushes the logfile to disk so prior output won't be lost in a subsequent crash.

TWAIN_SetHideUI / TWAIN_GetHideUI
void TWAIN_SetHideUI(int fHide);
int TWAIN_GetHideUI(void);
These functions control the 'hide source user interface' flag. This flag is initially FALSE(0), but if you set it non-zero, then when a source is enabled it will be asked to hide its user interface. Note this is a request - some sources will ignore it.
See: How To: Hide the Datasource User Interface.
If the user interface is hidden, you will probably want to set at least some of the basic acquisition parameters yourself – see Negotiating Scanning Parameters. See also: HasControllableUI

TWAIN_SetIndicators
int TWAIN_SetIndicators(BOOL bVisible)
Tell the source to show (hide) progress indicators during acquisition.

TWAIN_OpenSource
int TWAIN_OpenSource(LPCSTR pzName);
Opens the Source with the given name.
If that source is already open, does nothing and returns TRUE. If another source is open, closes it and attempts to open the specified source. Will load and open the Source Manager if needed.
If this call returns TRUE, TWAIN is in State 4 (TWAIN_SOURCE_OPEN)

TWAIN_SetPixelType
int TWAIN_SetPixelType(int nPixType);
Try to set the current pixel type for acquisition.
The source may select this pixel type, but don't assume it will.
This function should be used in place of the older TWAIN_SetCurrentPixelType.

Pixel Type Codes (TWPT_*)

<table>
<thead>
<tr>
<th>Code</th>
<th>TWAIN Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>TWPT_BW</td>
<td>1-bit per pixel, black and white</td>
</tr>
<tr>
<td>1</td>
<td>TWPT_GRAY</td>
<td>grayscale, 8 or 4-bit</td>
</tr>
<tr>
<td>2</td>
<td>TWPT_RGB</td>
<td>RGB color, 24-bit (rarely, 15,16,32-bit)</td>
</tr>
<tr>
<td>3</td>
<td>TWPT_PALETTE</td>
<td>indexed color (image has a color table) 8 or 4-bit.</td>
</tr>
<tr>
<td>4</td>
<td>TWPT_CMY</td>
<td>CMY color, 24-bit</td>
</tr>
<tr>
<td>5</td>
<td>TWPT_CMYK</td>
<td>CMYK color, 32-bit</td>
</tr>
</tbody>
</table>

Universitas Kristen Maranatha
**TWAIN_SetXferCount**

```c
int TWAIN_SetXferCount(int nXfers);
```

Tell the Source the number of images the application will accept.  

- `nXfers = -1` means any number (the default, when a device is opened.)  
- Returns: 1 for success, 0 for failure.

**TWAIN_SetAutoScan**

```c
int TWAIN_SetAutoScan(int fYes);
```

(Try to) turn on/off scan-ahead (CAP_AUTOSCAN). Returns TRUE(1) if successful, FALSE(0) otherwise.  

- This is an optional feature supported by some ADF scanners. When enabled, the scanner will scan pages before they are requested, buffering them in the scanner or host PC. When disabled, the scanner will not feed and scan a page until the application asks for it. Used to achieve maximum throughput on ADF scanners.  
  **Note:** A few high-speed scanners (e.g. Kodak i200) have this capability permanently on – such scanners always scan all pages in the feeder once they start.

**TWAIN_AcquireToFilename**

```c
int TWAIN_AcquireToFilename(HWND hwndApp, LPCSTR pszFile);
```

Acquire an image and save it to a file. If the filename contains a standard extension (.bmp, .jpg, .jpeg, .tif, .tiff, .png, .pdf, .gif, .dcx) then the file is saved in the implied format. Otherwise the file is saved in the default save format – see TWAIN_SetSaveFormat.  

- If `pszFile` is NULL or an empty string, the user is prompted for the file name and format with a standard Save File dialog. Only available and appropriate formats are presented in the Save File dialog. If you use this feature, you can call TWAIN_LastOutputFile to obtain the filename.  
- See also TWAIN_Acquire below.  

Return values:  

- 0 success.  
- -1 the Acquire failed.  
- -2 file open error (invalid path or name, or access denied)  
- -3 invalid DIB, or image incompatible with file format, or...  
- -4 writing failed, possibly output device is full.  
- -10 user cancelled File Save dialog  

The minimal use of EZTwain is to call this function with null arguments:  

```
ErrCode = TWAIN_AcquireToFilename(0, "");
```

**TWAIN_LastErrorCode**

```c
int TWAIN_LastErrorCode(void);
```

Return the most recent EZTwain error code, one of the EZTEC_ codes – See the EZTwain declaration file for your programming language, or refer to eztwain.h.

**TWAIN_ReportLastError**

```c
void TWAIN_ReportLastError(LPCSTR pzMsg);
```

Like TWAIN_ErrorBox, but if some details are available from TWAIN about the last failure, they are included in the message box. This function uses TWAIN_LastErrorText to find out about the last error – see below.