Lampiranl–Listing Program

```
unit Vapture;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  ExtCtrls, Videocap, StdCtrls, Menus, TeEngine, TeeFunc:, Series,
  TeeProcs, Chart,

type
  TForm1 = class(TForm)
    VideoCap1: TVideoCap;
    Image1: Tlrnage;
    Button1: TButton;
    Button2: TButton;
    CheckBox1: TCheckBox;
    MainMenu1: TMainMenu;
    File1 : TMenuItem;
    Exit1 : TMenuItem;
    About 1: TMenuItem;
    Image2: Tlmage;
    Image3: Tlmage;
    Image4: TImage;
    atas: TButton;
    Kiri: TButton;
    Kanan: TButton;
    bawah: TButton;
    Edit4: TEdit.
    Label4: TLabel;
    GroupBox4: TGroupBox;
    Button4: TButton;
    RadioGroup1: TRadioGroup;
    RadioGroup2: TRadioGroup;
    Chart 1: TChart;
    Series1: TLineSeries;
    TeeFunction1: TAddTeeFunction;
    RadioGroup3: TRadioGroup;
    procedure Button 1Click(Sender: TObject);
    procedure CheckBoxlClick(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure Exit 1Click(Sender: TObject);
    procedure About 1Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure atasclick( Sender: TObject);
    procedure bawahClick( Sender: TObject);
    procedure KiriClick(Sender: TObject);
    procedure KananClick(Sender: TObject);
    procedure Button4Click(Sender: TObject);

  private
    { Private declarations }
  public
    { Public declarations }
  end.
```

```
var
  Form1  TFornil,
  i  integer,
  a,b  integer,
  capterus,tekancap  boolean,

implementation
uses DlgTreiber, Unit1, NTPort, Unit2;

{$R *.DFM)

Const
   Maxlnput = 5,
   MaxLabel = 8,
   MaxOutput = 3,
   MaxRules = 1024,
   Full_Ux = 1,

type
  Detailsln = record
                In–Name : string[30];
                In  Unit : string[30];
                In_Min, In–Max : real;
                NumlnMF : byte.
                InMF_Name  : array[0..MaxLabel-1]  of string[30];
                InMF_Point  : array[0..MaxLabel-1,  1..4] of real;
              end;

  DetailsOut = record
                 OutName  string[30],
                 Out_Unit  string[30];
                 Out_Min,  Out–Max  real,
                 NumOutMF  byte.
                 OutMF_Name  array[0..MaxLabel-1] of string[30],
                 OutMF–Point  array[0..Maxlabel-1] of real,
               end,

  DetailsRule = record
                  Num_Ant,  Num_Con  word,
                  Antecedent  array[0  Maxlnput-1]  of byte,
                  Consequent  array[0..MaxOutput-1] of byte,
                  RuleGrade  real.
                end.

var
   Cnt, Cnt1,  Cnt2 : word,

   Datalnput : array[0  .Maxlnput-1]of detailsln,
   Dataoutput : array[0  .MaxOutput-1]of detailsout,
   DataRules : array[0.  MaxRules-1]  of DetailsRule.
   Crisplnput : array[0  .MaxInput of real,
   Fuzzylnput : array[0..Maxlnput-1,  O..MaxLabel-1]  of real.
   FuzzyOutput : array[0..MaxOutput-1,  0..MaxLabel-1]of real;
   CrispOutput  : array(0.Masoutput-I]  of real,
   Numlnput, NumOutput  : byte.
   NumRules : word;
```

```
Procedure DataMF–Rules;
Begin
  NumInput := 1;
  NumOutput := 2;
  NumRules := 10;

//---- Membership Input ----
  DataInput[0] .In_Name := 'Posisi';
  DataInput[0].In_Unit := 'Pixels';
  DataInput[0].In_Min := 0;
  DataInput[0].In_Max := 320;
  DataInput[0] .NumInMF := 5;
  DataInput[0].InMF_Name[0] := 'KR2';
  DataInput[0].InMF_Name[1] := 'KR1';
  DataInput[0] .InMF_Name[2] := 'T';
  DataInput[0].InMF_Name[3] := 'KA1';
  DataInput[0] .InMF_Name[4] := 'KA2';

  DataInput[0] InMF_Point[0,1] = 0,
  DataInput[0] InMF_Point[0,2] = 0,
  DataInput[0] InMF_Point[0,3] = 40,
  DataInput[0] InMF_Point[0,4] = 100,

  DataInput[0].InMF_Point[1,1] := 40;
  DataInput[0].InMF_Point[1,2] := 100;
  DataInput[0].InMF_Point[1,3] := 100;
  DataInput[0].InMF_Point[1,4] := 160;

  DataInput[0].InMF_Point[2,1] := 100;
  DataInput[0] .InMF_Point[2,2] := 160;
  DataInput[0].InMF_Point[2,3] := 160,
  DataInput[0]. InMF_Point[2,4] := 200;

  DataInput[0].InMF-Point[3,1] := 160;
  DataInput[0]. InMF_Point[3,2] := 200;
  DataInput[0] .InMF_Point[3,3] := 200;
  DataInput[0].InMF_Point[3,4] := 280;

  DataInput[0].InMF_Point[4,1] := 200;
  DataInput[0].InMF_Point[4,2] := 280;
  DataInput[0].InMF_Point[4,3] := 320;
  DataInput[0] InMF_Point[4,4] := 320;

//---- Membership Output–1 ----
  DataOutput[0]. Out–Name := 'Arah';
  DataOutput[0] .Out_unit := '_'
  DataOutput[0].Out-Min := -1;
  DataOutput[0].Out–Max := 1;
  DataOutput[0].NumOutMF := 3;
  DataOutput[0].OutMF_Name[0] := 'KR';
  DataOutput[0].OutMF_Name[1] := 'L';
  DataOutput[0].OutMF_Name[2] := 'KA';
  DataOutput[0].OutMF_Point[0] := -1;
  DataOutput[0].OutMF_Point[1] := 0;
  DataOutput[0] .OutMF–Point123 := 1;
```

```
//---- Membership Output-2 ----
   Dataoutput[I] OutName = 'Delay',
   DataOutput[1] Out_Unit = 'Out 1';
   Dataoutput[1]Out_Min = 0,
   DataOutput[1] Out-Max = 100,
   DataOutput[1] NumOutMF = 3,
   Dataoutput[1] OutMF-Name[0] = 'S',
   DataOutput[1] OutMF_Name[1] = 'L';
   DataOutput[1]OutMF_Name[2] = 'M';
   DataOutput[1] OutMF-Point[0] = 40,
   DataOutput[1] OutMF-Point[1] = 60,
   DataOutput1]OutMF_Point[2] = 50,

//---- Rules ----
   DataRules[0] Num_Ant = I,
   DataRules[0] Num_Con = 1;
   DataRules[0] Antecedent[0] = $0,
   DataRules[0] Consequent[0] = $80,

   DataRules[1]Num_Ant = 1,
   DataRules[1]Num_Con = 1;
   DataRules[1]Antecedent[0] = $01;
   DataRules[1]Consequent[0] = $80,

   DataRules[2].Num_Ant := 1;
   DataRules[2].Num_Con := 1;
   DataRules[2],Antecedent[0] := $02,
   DataRules[2].Consequent[0] := $81,

   DataRules[3] Num_Ant = 1,
   DataRules[3] Num_Con = I,
   DataRules[3] Antecedent0[] = $03,
   DataRules[3] Consequent[0] = $82;

   DataRules[4].Num_Ant := 1:
   DataRules[4].Num_Con := I;
   DataRules[4]. Antecedent[0] := 504;
   DataRules[4].Consequent[0] := $82;

   DataRules[5] Num_Ant = 1,
   DataRules[5] Num_Con = 1,
   DataRules[5] Antecedent[0] = SO,
   DataRules[5] Consequent[0] = $8A;

   DataRules[6] Num_Ant = I,
   DataRules[6] Num_Con = 1;
   DataRules[6] Antecedent[0] = $01,
   DataRules[6] Consequent[0] = $88,

   DataRules[7] Num_Ant = 1,
   DataRules[7] Num_Con = I,
   DataRules[7] Antecedent[0] = $02,
   DataRules[7] Consequent[0] = $89,

   DataRules[8] Num_Ant = 1.
   DataKules[8] Num_Con = 1,
   DataRules[8] Antecedent[0] = $03;
   DataRules[8] Consequent[0] = $88;
```

```
  DataRules[9] Nuin–Ant  = 1,
  DataRules[B] Num_Con   = 1,
  DataRules[9] Antecedent[0]   = $04,
  DataRules[9] Consequent[0]   = $84,
End.

function GetNoln(By    byte)  byte,
begin
   GetNoIn =(By and $38) div 8,
end.

function GetNoLabel(By : byte): byte;
begin
   GetNoLabel := By and $7;
end;

procedure Fuzzification,
var
  P1, P2, P3, P4, MP   Real,
begin
   for cnt I  = 0 to Maxlnput-l do
      for cnt2  = 0 to MaxLabel-1 do
         FuzzyInput[cnt l, cnt2]  = 0,

   for cntl  = 0 to NumInput-1 do
   begin
      MP  = CnspInput[cnt1],
      for cnt2  = 0 to Datalnput[cnt1]  NumInMF-1 do
      begin
         P1 =DataInput[cnt 1]InMF_Point[cnt2,  1];
         P2 =DataInput[cnt 1]InMF_Point[cnt2,  2];
         P3 =DataInput[cnt 1]InMF_Point[cnt2, 3];
         P4 =DataInput[cnt1]  lnMF_Point[cnt2, 41,

         if (Crisplnput[cnt1]  >= P1) and (Crisplnput[cnt1]  <= P4)
         then
         begin
            if (CrispInput[cnt1]  >= P2) and (CrispInput[cnt 1] <= P3)
            then Fuzzylnput[cnt 1, cnt2]  := Full_Ux;

            if CrispInput[cnt1]  < P2 then
               Fuzzylnput[cnt1.  cnt2] .= Full_Usl(P2-P1)*
               (MP-P1),

            if Crisplnput[cnt 1]> P3 then
               Fuzzylnput[cnt 1, cnt2]  = Full_Ux(P4-P3)*
               (P4-MP),
         end,
      end,
   end,
end,
```

```
procedure EvaluateRules,
var
  TempVal   real,
  NoIn, NoLabel   byte,
begin
    for cnt1  = 0 to MaxOutput-1 do
      for cnt2  = 0 to MaxLabel-1 do
        FuzzyOutput[cnt1,cnt2]  = 0,

    for cnt  = 0 to NumRules-1 do
    begin
      TempVal  = Full_Ux,
      for cntl  = 0 to DataRules[cnt] Num_Ant-1  do
      begin
        Noh  = GetNoIn(DataRules[cnt]  Antecedent[cnt 1]),
        NoLabel  = GetNoLabel(DataRules[cnt]  Antecedent[cnt 1]);
        if FuzzyInput[NoIn,  NoLabel] < TempVal then
          TempVal  = FuzzyInput[NoIn,  NoLabel],
        if TempVal = 0 then break,
      end,

      DataRules[cnt]  RuleGrade  = TempVal,
      if TempVal > 0 then
        for cnt1  = 0 to DataRules[cnt] Num_Con- 1 do
        begin
          NoIn  = GetNoln(DataRules[cnt]  Consequent[cnt 1]),
          NoLabel  = GetNoLabel(DataRules[cnt] Consequent[cnt 1]),
          if FuzzyOutput[Noln, NoLabel] < TempVal then
            FuzzyOutput[NoIn, NoLabel]  = TempVal,
        end,
    end,
end,

procedure Defuzzification(Metode  : string);
var
  TempVal],  TempVal2 : real;
begin
    if Metode = 'COG' then
    begin
      for cnt :=0 to NumOutput-1  do
      begin
        TempVal1 :=0;
        TempVal2 := 0;
        for cntl  :=0 to DataOutput[cnt].NumOutMF-1 do
        begin
          TempVal1 := TempVal1 + FuzzyOutput[cnt,  cntl];
          TempVal2 := TempVal2 + FuzzyOutput[cnt,  cntl] *
          DataOutput[cnt].OutMF_Point[cnt 13;
        end;
        if TempVal1 = 0 then
          CrispOutput[cnt] .= 0
        else
          CrispOutput[cnt] :=  TempVal2/TempVal1 ;
      end;
    end;
end;
```

```
Procedure Sobel(im1:TBitmap; var im2:TBitmap);
   TYPE
    TRGBTripleArray =  ARRAY[WORD] OF TRGBTriple;
    pRGBTripleArray =  ^TRGBTripleArray;

   VAR
    1      INTEGER,
    J      INTREGER;
    rowA   pRGBTripleArrap.
    rowB   pKGBTripleArray,
    rowC   pRGBTripleArray,
    rowD   pRGBTripleArray,
    temp   integer,

begin
   iml .PixelFormat := pf24bit;
   im2.PixelFormat := pf24bit;
   For J:=0 To iml .Height-1 Do
    Begin
     rowA := Iml .Scanline[i];
     rowB := lm2,Scanline[j];
      For I:=0 To inil.Width-1 Do
      Begin
      if((i=0) or (j=0) or Q=(iml.Height-1))  or (i=(imI.Width-1)))  then
       temp:=0
      else
       begin
        rowC := Iml .Scanline[j-1];
        rowD := Iml .Scanline[j+l];
        Temp:=Abs(rowC[I+ 1].rgbtRed
             +2*rowA[l+ l] rgbtRed
             +rowD[l+ l]. rgbtRed
             -rowC[l-I] .rgbtRed
             -2*rowA[I-1].rgbtRed
             -rowD[l-l] .rgbtRed)+
             Abs(rowC[I-1].rgbtRed
             +2*rowC[l].rgbtRed
             +rowC[I+ 11.rgbtRed
             -rowD[I-1].rgbtRed
             -2*rowD[I] .rgbtRed
             -iowD[I+ 11.rgbtRed);
       end;

      If (Temp255) Then
       begin
        rowB[i] rgbtRed    =255,
        rowB[i] rgbtGreen  2255,
        rowB[i] rgbtBlue   -255,
       end
      Else
       begin
        rowB[i] rgbtRed    =temp,
        rowB[i] rgbtGreen  =temp,
        rowB[i] rgbtBlue   =temp,
       end
      End,
    end,
end,
```

```
Procedure Laplacian(im1 TBitmap, var im2 TBitmap),
   TYPE
    TRGBTripleArray =  ARRAY[WORD] OF TRGBTriple,
    pRGBTripleArray =  ^TRGRTripleArray,

   VAR
    i   : INTEGER;
    j   : INTEGER;
    rowA:  pRGBTripleArray,
    rowB:  pRGBTripleArray;
    rowC:  pRGBTripleArray;
    rowD:  pRGBTripleArray;
    temp: integer;

begin
   iml .PixelFormat := pf24bit,
   im2.PixelFormat := pf24bit;

   For J:=0 To iml.Height-1 Do
    Begin
     rowA  := Iml .Scanline[j];
     rowB  := Im2.Scanline[j];
      For 1:=0 To im1.Width-1 Do
      Begin
      if ((i=0) or [j=0) or Q=(iml.Height-1))  or (i=(imI .Width-1))) then
       temp:=255
      else
       begin
        rowC  := Iml .Scanline[j-11];
        rowD  := Iml .Scanline[j+1l];
        Temp:=Abs(-rowC[1+ 1].rgbtRed
             -rowA[I+ 1].rgbtRed
             -rowD[I+ 1].rgbtRed
             -rowC[1-11.rgbtRed
             -rowA[1-1].rgbtRed
             -rowD[I-1].rgbtRed
             -rowC[I].rgbtRed
             -8*rowA[I] .rgbtRed
             -rowD[I].rgbtRed),
       end;

      If (Temp>255) Then
       begin
        rowB[i] rgbtRed    =255,
        rowB[i] rgbtGreen  =2S5,
        rowB[i] rgbtBlue   =255,
       end
      Else
       begin
        rowB[i] rgbtRed    =temp,
        rowB[i] rgbtGreen  =temp
        rowB[i] rgbtBlue   =temp,
       end
      End,
    end,
end,
```

```
Procedure GrayMean(im1:TBitmap; var im2:TBitmap);
   TYPE
    TRGBTripleArray = ARRAY[WORD] OF TRGBTriple,
    pRGBTripleArray = ^TRGBTripleArray;

   VAR
    1     INTEGER,
    j     INTEGER,
    rowA  pRGBTnpleArray,
    rowB  pRGBTripleArray,
    pmean integer,

begin
   iml .PixelFormat := pf24bit;
   im2.PixelFormat := pf24bit;

   For J:=0 To iml.Height-1 Do
    Begin
    rowA := Iml.Scanline[j];
    rowB := Im2.Scanline[j];
      For I:=0 To iml.Width-1 Do
      Begin
    pmean:=round((rowA[i] .rgbtRed +rowA[i] .rgbtCreen+rowA[i].rgbtBlue)/3);
      rowB[i] rgbtRed   :=pmean;
      rowB[i].rgbtGreen :=pmean;
      rowB[i] .rgbtBlue :=pmean;
    END,
      End;
end.

Procedure Cangradien(im 1:TBitmap;var xa,xb:integer);
   VAR
    i  . INTEGER;
    j  . INTEGER;
    grad: real;
    kiriatas,kiribawah:TPoint;
    kananatas,kananbawah:TPoint;
    gradatas,gradbawah: TPoint;
    ketemu:boolean;
begin
   iml .PixelFormat := pf24bit;

   //---------kiriatas----------
   ketemu:=false;
   For J.=0 To iml.Height-1 Do
   begin
     For I =0 To iml .Width-1 Do
       if iml1.Canvas.Pixels[i,j]=clBlack then
         begin
           kiriatas x.=i,
           kiriatas y =j,
           ketemu .=true,
           if ketemu then break;
         end.
     if ketemu then break;
   end,
```

```
//---------kiribawah----------
    ketemu:=false;
    For J:=im1.Height-1 downTo 0 Do
//    For J:=iml.Height-21 downTo 0 Do
    begin
      For 1:=0 To iml.Width-1 Do
        if im1.Canvas.Pixels[i,j]=clBlackhen
          begin
            kiribawah.x:=i,
            kiribawah.y.=j,
            ketemu:=true;
            if ketemu then break;
          end;
      if ketemu then break.
    end;
    iml.Canvas.Pen.Color:=ClRed,
    im1.Canvas.MoveTo(kinatas.x,kiriatas.y);
    im1.Canvas.LineTo(kiribawah.x,kiribawah.y);
    im1.CanvasPen.Color.=CIBlack:

    //---------kananatas----------
    ketemu:=false;
    For J:=0 To iml.Height-1 Do
    begin
      For I:=iml.Width-1 DownTo 0 Do
        if imI.Canvas.Pixels[i,j]=clBlackthen
          begin
            kananatas.x:=i;
            kananatas.y.=j;
            ketemu:=true;
            if ketemu then break,
          end;
      if ketemu then break:
    end;

    //---------kananbawah----------
    ketemu:=false;
    For J:=iml.Height-1 dowvnTo 0 Do
//    For J:=iml .Height-21 downTo 0 Do
    begin
      For I:=iml.Width-1 DownTo 0 Do
        if iml .Canvas.Pixels[i.j]=clBlack then
          begin
            kananbawah.x:=i.
            kananbawah.y.=.i.
            ketemu:=true,
            if ketemu then break.
          end;
      if ketemu then break.
    end;
     ■  Canvas.Pen Color =CIBlue,
    iml.Canvas MoveTo(kananatas.x,kananatas.y);
    iml .CanvasLineTo(kananbawah.x,kananbawah.y);
    iml .Canvas.Pen.Color.=ClBlack,
    gradatas.x.=round((kiriatas.x+kananatas.x)/2);
    gradbawah.x:=round((kiribawah x+kananbawah.x)/2),
    gradatas.y:=0;
    gradbawah y:=iml.Height,
```

```
    im1.Canvas.Pen.Color:=CiGreen;
    iml .Canvas.MoveTo(gradatas.x,gradatas y);
//    iml .Canvas.LineTo(gradbawah.x,gradbawah.y);
    im1.Canvas.LineTo(gradbawah.x,gradbawah.y-2
    iml .Canvas.Pen.Color.=ClBlack;
//    Forml .Edit4.Text:=lntToStr(kananatas.x-kiriatas.x);
    xa:=gradatas.x;
    xb:=gradbawah x;
end;

Procedure Graylllum(iml  TBitmap, var im2 TBitmap),
    TYPE
      TRGBTripleArray =  ARRAY[WORD] OF TRGBTriple,
      pRGBTripleArray =  ^TRGBTripleArrav,

    VAR
      i      INTEGER,
      j      INTEGER,
      rowA   pRGBTripleArray.
      rowB   pRGBTripleArray,
      illum  integer,

begin
    iml .PixelFormat := pf24bit,
    im2.PixelFormat  := pf24bit;

    For J:=0 To iml.Height-1 Do
     Begin
      rowA  := Iml .Scanline[j].
      rowB  := Im2.Scanline[j];
        For 1.=0 To iml.Width-1 Do
        Begin
         illum:=round(0.229*rowA[i].rgbtRed
              +0.587*rowA[i].rgbtGreen
              +0.114*rowA[i].rgbtBlue);
        rowB[i].rgbtRed    :=illum;
        rowB[i].rgbtGreen  :=illum;
        rowB[i].rgbtBlue   :=illum;
       End;
     End;
end;

Procedure ThresholdSingle(iml  TBitmap,T integer, var im2 TBitmap),
    TYPE
      TRGBTripleArray =  ARRAY[WORD] OF TRGBTriple,
      pRGBTripleArray =  ''TRGBTripleArray,

    VAR
      i      INTEGER,
      J      INTEGER,
      rowA   pRGBTripIeArray.
      rowB   pRGBTripleArray,
      temp   integer,

begin
    iml  PixelFormat  = pf24bit,
    im2 PixelFormat   = pf24bit,
```

```
    For J:=0 To iml.Height-1 Do
     Begin
      rowA := Iml .Scanlinef[i];
      rowB := Im2.Scanline[j];
        For I:=0 To iml.Width-1  Do
        Begin
      temp:=rowA[i].rgbtRed;
      If temp > T then
         begin
         rowB[i]..gbtRed   :=0;
         rowB[i].rgbtGreen :=0;
         rowB[i].rgbtBlue  :=0;
         end
      else
        begin
        rowB[i].rgbtRed   :=255;
        rowB[i] .rgbtGreen :=255;
        rowB[i].rgbtBlue  :=255;
        end;
      END,
        End;
end;

Procedure ThresholdDual(im 1 :TBitmap;TL,TH:integer; var im2:TBitmap);
    TYPE
      TRGBTripleArray =  ARRAY[WORD] OF TRGBTriple;
      pRGBTripleArray =  "TRGBTripleArray;

    VAR
      1      INTEGER,
      J      INTEGER,
      rowA   pRGBTripleArray,
      rowB   pRGBTripleArray,
      temp  integer,

begin
    iml .PixelFormat := pf24bit;
    im2.PixelFormat := pf24bit;

    For J:=0 To iml .Height-1 Do
     Begin
      rowA := Iml .Scanline[j];
      rowB := Im2.Scanline[j];
        For I:=0 To iml.Width-1 Do
        Begin
      temp:=rowA[i].rgbtRed;
      If (temp>TL) and (temp<TH) then
         begin
         rowB[i].rgbtRed   :=0,
         rowB[i].rgbtGreen :=0;
         rowB[i].rgbtBlue  :=0;
         end
      else
        begin
        rowB[i].rgbtRed   :=255,
        rowB[i].rgbtGreen .=255;
        rowB[i].rgbtBlue  :=255,
        end,
```

```
      END;
       End;
end:

procedure TForm1 FormCreate( Sender  TObject),
begin
  Capterus  =false,
  Tekancap =false,
  Licenselnfo('Agus Widian'. 1 1 13840),
end.

procedure TForm1.Button1Click(Sender. TObject);
Var DrvList:TStrings,
begin
DlgEinstell:=TDlgEinstell.Create(elf),
 drvList:= GetDriverList ,
 dlgEinstell.Combobox1 .Items := drvlist,
 VideoCap1.DriverOpen := false;
 dlgEinstell.ComboBox1.Itemindex := VideoCap1 .DriverIndex;
 if DlgEinstell ShowModal = mrOK then
    begin
     videocap1.Driverlndex:= dlgEinstell.combobox1.Itemlndex;
    end;
 VideoCap1.DriverOpen:=true;
 DlgEinstell.Free,
 drvList.Clear;
 drvList.Free;
 Button2.Enabled:=True;
 Button4.Enabled:=True,
 CheckBox1.Enabled:=True,
 atas.Enabled.=True.
 kiri.Enabled:=True;
 kanan.Enabled.=True,
 bawah.Enabled:=True;
end:

procedure TForm1.CheckBoxlClick(Sender:TObject);
begin
videocap1.VideoPreview:=checkbox1.Checked;
end;

procedure TForm1.Button?Click(Sender  TObject);
var
   bitmap 1,bmp2,bmp3:tbitmap;
   asa1,tujuan:trect;
   hasilpos:integer;
   hxa,hxb,t :integer;
   hasilgrad:real;
begin
 If not tekancap then
begin
 capterus:=false;
 Button2,Caption:='Stop';
 tekancap:=true;
 t:=l;
 series1.Clear;
 DataMF_Rules;
repeat
```

```
//---- Dari Driver Kamera ke Image -----
   VideoCap1.SingleImageFile.='lmage001.Bmp';
   videocap1.GrabFrameNoStop;
   VideoCap1,SaveAsDIB;
   Bitmap1:=TBitmap.Create;
   bmp2:=TBitmap.Create,
   bmp3:=TBitmap.Create;

//---- Ambil Image -----
   Bitmap1.LoadFromFile('lmage00  1.bmp');

//---- Crop ----
   asal:=Rect(0,Bitmapl  .Height-21,Bitmap1.Width-1,Bitmap1.Height-1);
   tuJuan:=Rect(0,0,Bitmap  1.Width-1,20),
   Imagel .Canvas.CopyRect(tujuan,Bitmap1.Canvas,asal);
   Imagel .Refresh;
   Imagel .Show;
   Image1.Picture.SaveToFile('lmagetemp.bmp');

   Bmp2.LoadFromFile('lmagetemp  .bmp');
   Bmp3.LoadFromFile('lmagetemp1.bmp'),
   Bitmap1.LoadFromFile('Imagetemp1.bmp');

//---- Mengubah nilai RGB Ke Gray Scale ----
   case radiogroup2  ltemlndex of
     0 Graymean(bmp2,bmp3),
     1 Grayillum(bmp2,bmp3),
   end,
   Image2 Picture Bitmap =bmp3,
   Image2 Refresh,
   Image2 Show,
   Image2 Picture SaveToFile('l3 bmp'),

//---- Pemilihan warna garis -----
  case RadioGroup1 .ItemIndex of

//---- Warna Abu - abu ----
  0: Begin
      ThresholdDual(bmp3, StrTolnt(form2,edit1.text),StrTolnt(form2.edit2.text),bmp2);
      Laplacian(bmp2,Bitmap1);
      Sobel(Bitmap 1,bmp2);
      ThresholdSingle(bmp2,105,bitmap 1);
      carigradien(bitmap1,hxa,hxb);
      hasilpos:= round(abs(hxa+hxb)/2);

//---- Plot koordinat garis (Pv) ----
     if t >= 100 then
     begin
       chartl .bottomaxis.minimum:= chart1 .bottomaxis.minimum + 1;
       chart 1.bottomaxis.maximum:=chart 1.bottomasis.maximum + 1;
     end;
     series1.Add(hasilpos,lntToStr(t),clred),
     t:=t+l;

//---- Proses Control ----
     case radiogroup3.ltemlndex of

//---- On/Off Control ----
```

```
    0 : case hasilpos of
         0..110      : kiri.Click, //-- belok kiri
         111...210 : atas.click; //-- Lurus
         21 1..320 : kanan.Click; //-- Belok kanan
        end;

//---- Fuzzy Control ----
    I : Begin
         Crisplnput[0] =HasilPos;
         Fuzzification,
         EvaluateRules;
         DeFuzzification('C0G);

        If CrispOutput[0] < -0.25 Then
          Begin
           outport($378,$06);
           sleep(Round(CrispOutput[1]));
           outport($378,$00);
          End
         Else
          If CrispOutput[0] < 0.25 Then
           Begin
            outport($378,$05);
            sleep(Round(CrispOutput[I]));
            outport($378,$00);
          End
          Else
            Begin
             outport($378,$09);
             sleep(Round(CrispOutput[1]));
             outport($378,$00).
            End;
       end,
    end;
    Image3.Picture.Bitmap:=bitmapl;
    Image3.Refresh;
    Image3.Show;
   end;

//---- Warna Hijau ----
  I :begin
//---- Ambil koordinat garis ----
    ThresholdDual(bmp3,StrTolnt(form2.edit3.text),StrToInt(form2.edit4.text),bmp2);
    Laplacian(bmp2,Bitmap 1);
    Sobel(Bitmap 1,bmp2);
    ThresholdSingle(bmp2,105,bitmap1);
    carigradien(bitmap 1,hxa,hxb);
    hasilpos.= round(abs(hsa+hxb)/2).

//---- Plot koordinat garis (Pv) ----
    if t >= 100 then
    begin
      chart I .bottomasis.minimum:=chartl .bottomaxis.minimum + I ;
      chartl .bottomaxis.maximum:= chartl .bottomaxis.maximum + 1;
    end,
    series1 Add(hasilpos,lntToStr(t),clred);
    t:=t+ I,
```

```
//---- Proses Control ----
    case radiogroup3.ItemIndex Of

//---- On/Off Control ----
    0   case hasilpos of
          0  110      kiri Click,  //-- belok kiri
          111  210    atas click,  //-- Lurus
          211  320    kanan Click, //-- Belok Kanan
        end,

//---- Fuzzy Control ----
    1 : Begin
          CrispInput[0] := HasilPos;
          Fuzzification;
          EvaluateRules;
          Defuzzification('COG');

          If CrispOutput[0] < -0 25 Then
            Begin
              outport($378,$06),
              sleep(Round(CrispOutput[ 1])),
              outport($378,$00),
            End
          Else
            If CrispOutput[0] < 0 25 Then
              Begin
                outport($3 78,$05),
                sleep(Round(CrispOutput[ 1]));
              outport($378,$00),
            End
            Else
              Begin
                outport($378,$09),
                sleep(Round(CnspOutput[ 1])),
                outport($378,$00),
              End,
        End,

    end;
    Image4.Picture.Bitmap:=bitmap1;
    Image4.Refresh;
    Image4.Show;
    end;
   end,
  Bitmap1.Free;
  Bmp2.Free;
  Bmp3.Free;
  Sleep(300);
  Application.ProcessMessages;
until capterus
end
else
 begin
  Button2.Caption:='Start Automatic',
  tekancap:=false;
  Capterus:=true;
 end;
end,
```

```
procedure TForm1.Button3CGck(Sender: TObject);
begin
  videocap1.DlgVFormat;
end,

procedure TForm1.About1Click(Sender: TObject);
begin
  aboutboxshow;
end;

procedure TForm1atasclick(Sender TObject),
begin
  outport($378,$05),
  sleep(StrTolnt(Edit4 Text)),
  outport($378,$00),
end,

procedure TForm1 bawahClick(Sender. TObject);
begin
  outport($378,$0A),
  sleep(StrToInt(Edit4 Text)),
  outport($378,$00),
end,

procedure TForm1.KiriClick(Sender: TObject);
begin
  outport($378,$06);
  sleep(StrToInt(Edit4,Text));
  outport($378,$00);
end;

procedure TForml KananClick(Sender TObject),
begin
  outport($378,$09),
  sleep(StrToInt(Edit4 Text)),
  outport(S378,$00),
end,

procedure TForm1.Exit1Click(Sender: TObject);
begin
   Close;
end;

procedure TForm1.Button4Click(Sender: TObject);
begin
  Form2.Show;
end;

End
```

**Lampiran2_Listing Program**

```
unit Unit1;

interface

uses Windows, SysUtils, Classes, Graphics, Forms, Controls, StdCtrls,
  Buttons, ExtCtrls, Db, DBTables;

type
  TAboutBox = class(TForm)
    Panel1: TPanel;
    ProgramIcon: TImage;
    ProductName: TLabel;
    Version: TLabel;
    Copyright: TLabel;
    OKButton: TButton;
    Label1: TLabel;
    Label2: TLabel;
    procedure OKButtonClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  AboutBox: TAboutBox;

implementation

{$R *.DFM}

procedure TAboutBox.OKButtonClick(Sender: TObject);
begin
Close;
end;

end
```

**Lampiran3_Listing Program**

```
unit Unit2:

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, videocap;

type
  TForm2 = class(TForm)
    Button3: TButton;
    Edit1: TEdit;
    Edit2: TEdit;
    Edit3: TEdit;
    Edit4: TEdit;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    Button1: TButton;
    procedure Button3Click(Sender: TObject);
    procedure ButtonlClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form2: TForm2;

implementation

uses Vapture;

{$R *.DFM}

procedure TForrn2,Button3Click(Sender: TObject);
begin
   Form1.videocap1.DlgVForrnat,
end,

procedure TForm2 Button1Click(Sender. TObject),
begin
  close,
end,

end.
```

INTEGRATE0 CIRCUITS
UNITRODE

**L293**
**L293D**

# Push-Pull Four Channel Driver

## FEATURES

- Output Current **1A** Per Channel (600mA for L293D)
- Peak Output Current 2A Per Channel **(1.2A** for L293D)
- Inhibit Facility
- High Noise Immunity
- Separate Logic Supply
- Over-Temperature Protection

## DESCRIPTION

The *L293* and L293D are quad push-pull drivers capable of delivering output currents to 1A or 600mA per channel respectively. Each channel is controlled by a ITL-compatible logic input and each pair of drivers (a full bridge) is equipped with an inhibit input which turns off all four transistors. **A** separate supply input is provided for the logic **so** that it may be run off a lower voltage to reduce dissipation.

Additionally the L293D includes the output clamping diodes within the IC for complete interfacing with inductive loads.

Both devices are available in 16-pin Batwing DIP packages. They are also available in Power S0lC and Hermetic DIL packages.

## TRUTH TABLE

| $V_I$ (each channel) | $V_{INH}$* | $V_O$ |
|---|---|---|
| H | H | H |
| L | H | L |
| H | L | X** |
| L | L | X** |

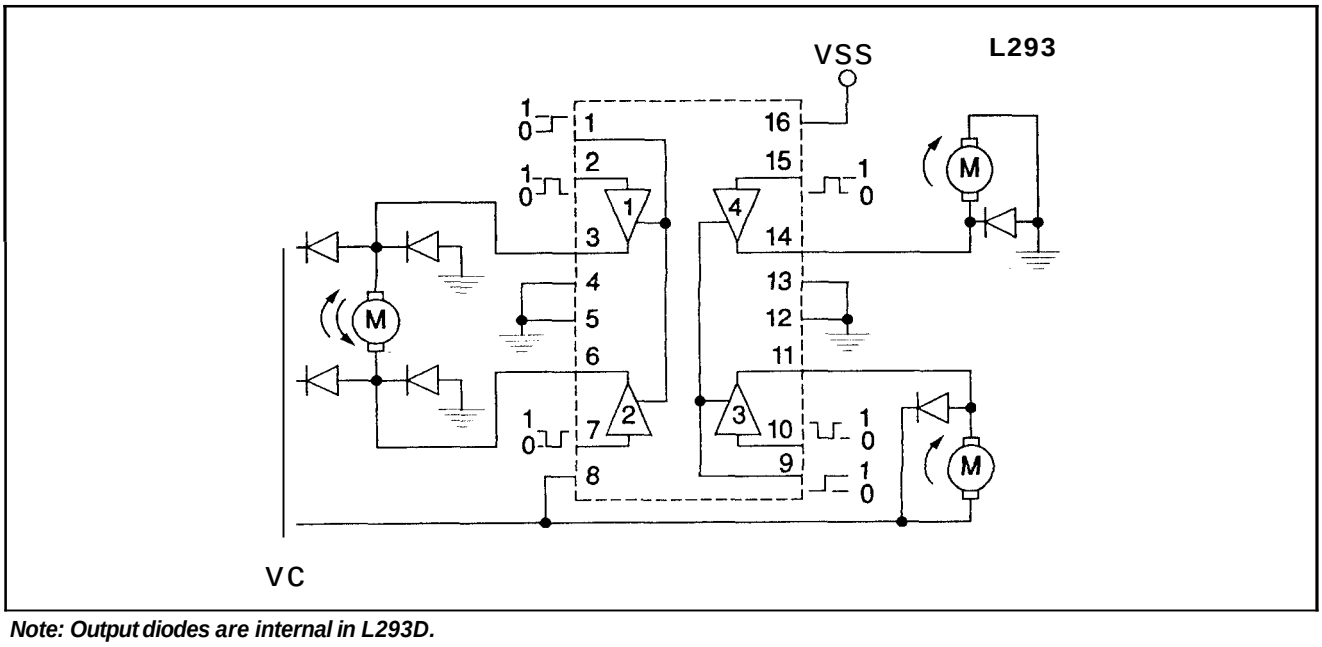
***High** output **impedence***

## ABSOLUTE MAXIMUM RATINGS

Collector Supply Voltage, Vc . . . . . 36V
Logic Supply Voltage, Vss. . . . . 36V
Input Voltage, $V_I$. . . . . ***7V***
Inhibit Voltage, $V_{INH}$ . . . . . ***7V***
Peak Output Current (Non-Repetitive), $I_{OUT}$ (L293) . . . . . ***2A***
$I_{OUT}$(L293D) . . . . . 1.2A
Total Power Dissipation
at Tground-pins = **80°C,** N Batwing pkg, (Note) . . . . . **5W**
Storage and Junction Temperature, Tstg, $T_J$ . . . . . **-40 to +150°C**

***Note: Consult packaging section of Databook for thermal limitations and considerations of packages.***

## BLOCK DIAGRAM

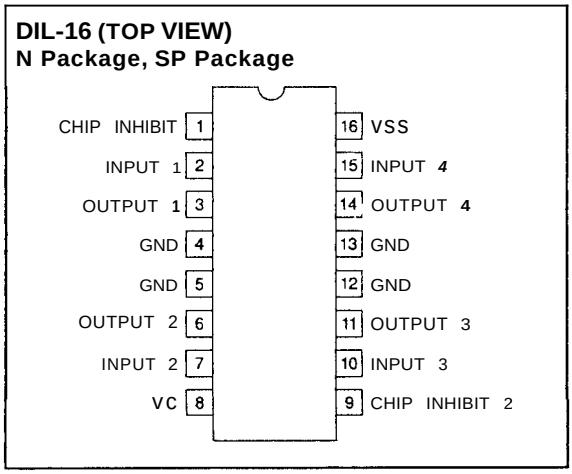



***Note: Output diodes are internal in L293D.***

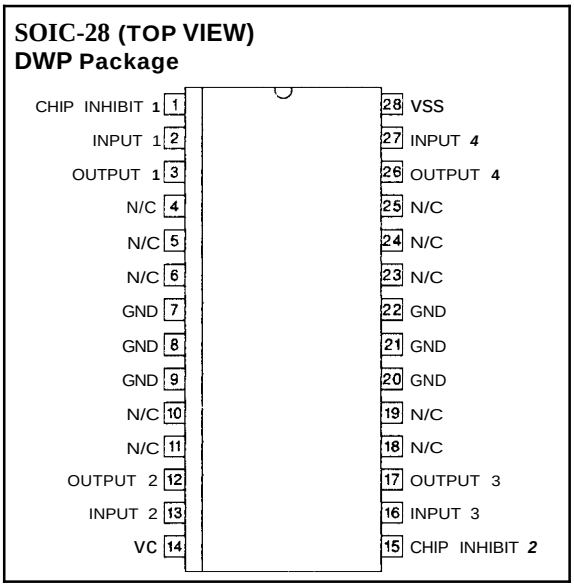
***8/94***

## CONNECTION DIAGRAMS

**DIL-16 (TOP VIEW)**
**N Package, SP Package**



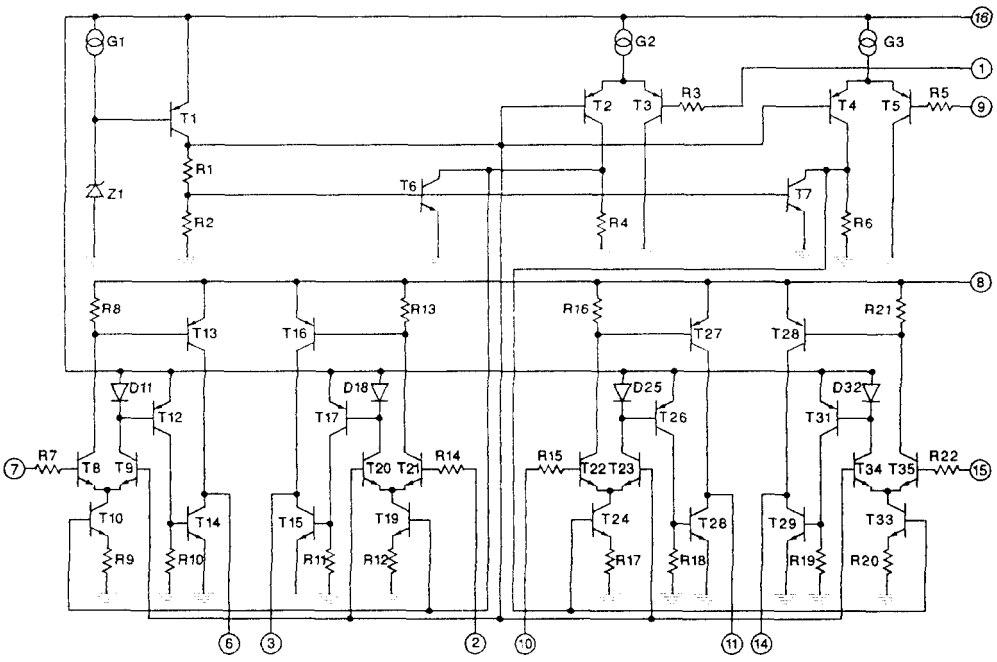

**SOIC-28 (TOP VIEW)**
**DWP Package**




**ELECTRICAL CHARACTISTICS:** (For each channel, $V_C$ = 24V, $V_{SS}$ = 5V, $T_{AMB}$ = 25°C, unless otherwise specified; $T_A = T_J$)

| PARAMETER | SYMBOL | TEST CONDITION | MIN. | TYP. | MAX | UNITS |
|---|---|---|---|---|---|---|
| Collector Supply Voltage | $V_C$ | | | | 36 | V |
| Logic Supply Voltage | $V_{SS}$ | | 4.5 | | 36 | V |
| Collector Supply Current | $I_C$ | $V_I$ = L, $I_O$ = 0, $V_{INH}$ = H | | 2 | 6 | mA |
| | | $V_I$ = H, $I_O$ = 0, $V_{INH}$ = H | | 16 | 24 | mA |
| | | $V_{INH}$ = L | | | 4 | mA |
| Total Quiescent Logic Supply Current | $I_{SS}$ | $V_I$ = L, $I_O$ = 0, $V_{INH}$ = H | | 44 | 60 | mA |
| | | $V_I$ = H, $I_O$ = 0, $V_{INH}$ = H | | 16 | 22 | mA |
| | | $V_{INH}$ = L | | 16 | 24 | mA |
| Input Low Voltage | $V_{IL}$ | | -0.3 | | 1.5 | V |
| Input High Voltage | $V_{IH}$ | $V_{SS} \leq 7V$ | 2.3 | | $V_{SS}$ | V |
| | | $V_{SS} \geq 7V$ | 2.3 | | 7 | V |
| Low Voltage Input Current | $I_{IL}$ | $V_I$ = 0V | | | -10 | µA |
| High Voltage Input Current | $I_{IH}$ | $V_I$ = 4.5V | | 30 | 100 | µA |
| Inhibit Low Voltage | $V_{INH,L}$ | | -0.3 | | 1.5 | V |
| Inhibit High Voltage | $V_{INH,H}$ | $V_{SS} \leq 7V$ | 2.3 | | $V_{SS}$ | V |
| | | $V_{SS}$ >7V | 2.3 | | 7 | V |
| Low Voltage Inhibit Current | $V_{INH,L}$ | | | -30 | -100 | µA |
| High Voltage Inhibit Current | $V_{INH,H}$ | | | | 10 | µA |
| Source Output Saturation Voltage | $V_{CEsatH}$ | $I_O$ = -1A (-0.6A for L293D) | | 1.4 | 1.8 | V |
| Sink Output Saturation Voltage | $V_{CEsatL}$ | $I_O$ = 1A (0.6A for L293D) | | 1.2 | 1.8 | V |
| Clamp Diode Forward Voltage (L293D only) | $V_F$ | $I_F$ = 0.6A | | 1.3 | | V |
| Rise Time | $T_R$ | 0.1 to 0.9 $V_O$ (See Figure 1) | | 100 | | ns |
| Fall Time | $T_F$ | 0.9 to 0.1 $V_O$ (See Figure 1) | | 350 | | ns |
| Turn-on Delay | $T_{ON}$ | 0. 5 $V_I$ to 0.5 $V_O$ (See Figure 1) | | 750 | | ns |
| Turn-off Delay | $T_{OFF}$ | 0. 5 $V_I$ to 0.5 $V_O$ (See Figure 1) | | 200 | | ns |

## SCHEMATICDIAGRAM




## APPLICATION INFORMATION

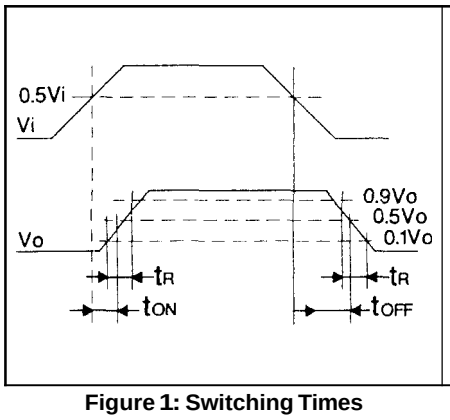



Figure 1: Switching Times

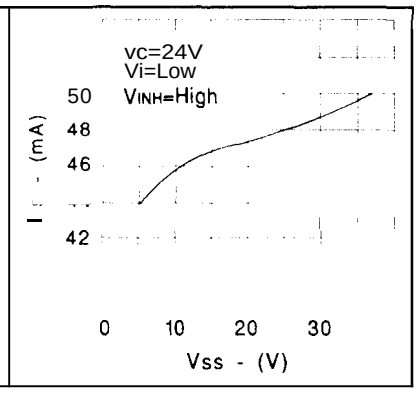



Figure 2: Quiescent Logic Supply Current vs Logic Supply Voltage

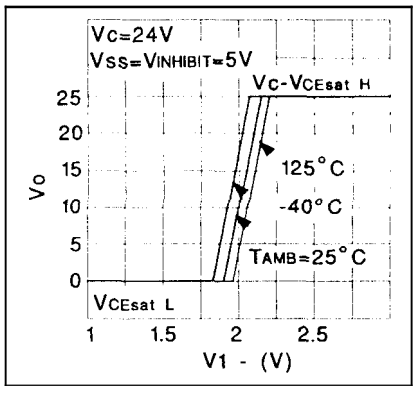



Figure 3: Output Voltage vs Input Voltage

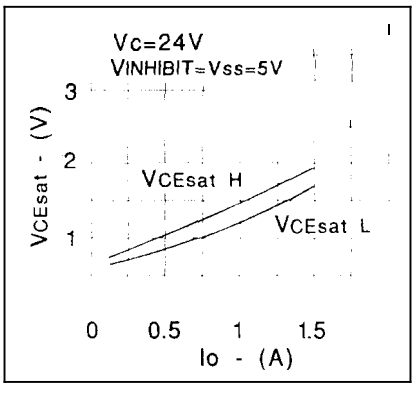
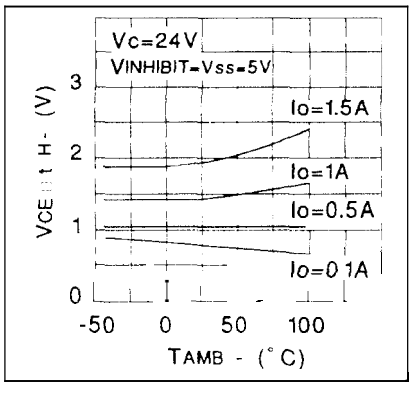



Figure 4: L293 Saturation vs Output Currrent



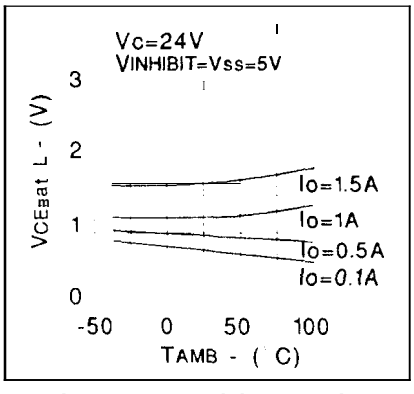




Figure 6: L293 Sink Saturation Voltage vs Ambient Temperature

*NOTE: For L293D curves, multiply output current by 0.6*

## APPLICATION INFORMATION (Cont.)

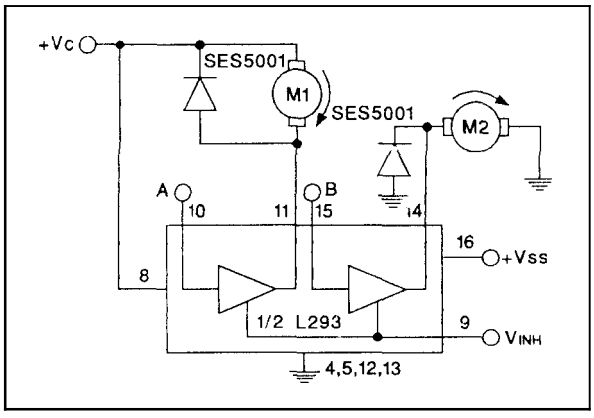



**Figure 7: DC Motor Controls (with Connection to Ground and to Supply Voltage)**

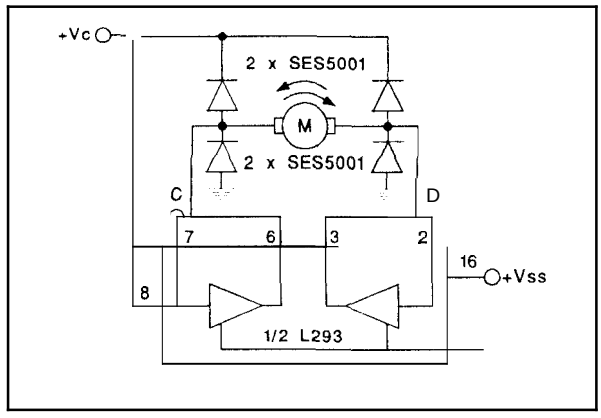



**Figure 8: Bidirectional DC Motor Control**

| VINH | A | M1 | B | M2 |
|---|---|---|---|---|
| H | H | Fast Motor Stop | H | Run |
| H | L | Run | L | Fast Motor Stop |
| | | Free Running Motor Stop | X | Free Running Motor Stop |

| INPUTS | | FUNCTION |
|---|---|---|
| VINH = H | C = H; D = L | Turn Right |
| | C = L; D = H | Turn Left |
| | C = D | Fast Motor Stop |
| | | Free Running Motor |

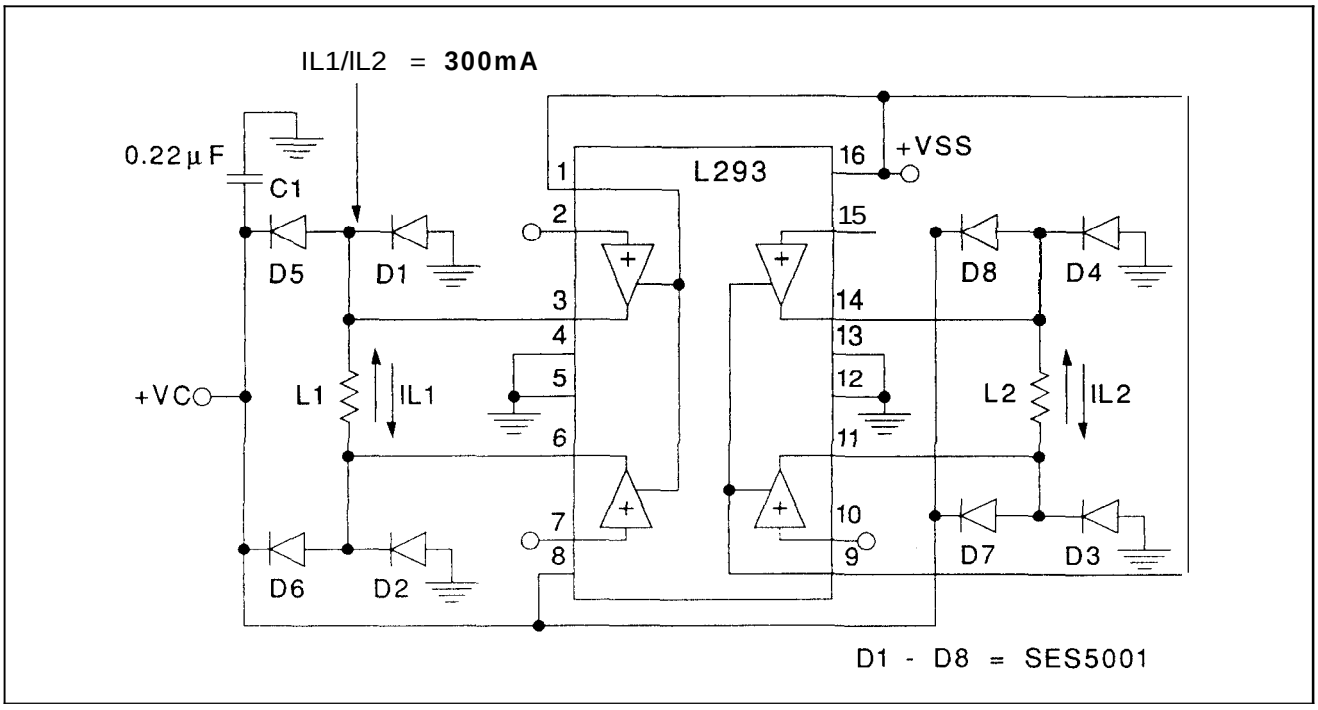



**Figure 9: Bipolar Stepping Motor Control**

## MOUNTING INSTRUCTIONS

The Rthj-amp of the L293 can be reduced by soldering the GND pins to a suitable copper area of the printed circuit board or to an external heatsink.

The diagram of Figure 13 shows the maximum package power Ptot and the $\theta_{JA}$ as a function of the side "l" of two equal square copper areas having a thickness of 35µ (see Figure 10). In addition, it is possible to use an external heatsink (see Figure 11).

During soldering the pins' temperature must not exceed 260°C and the soldering time must not be longer than 12 seconds.

The external heatsink or printed circuit copper area must be connected to electrical ground.

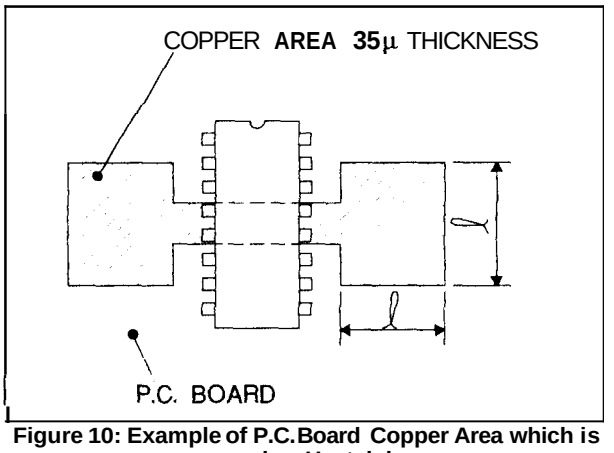



**Figure 10: Example of P.C.Board Copper Area which is used as Heatsink**

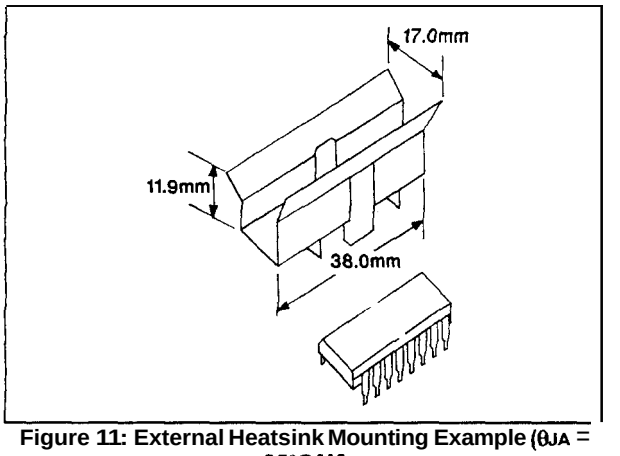



**Figure 11: External Heatsink Mounting Example ($\theta_{JA}$ = 25°C/W)**

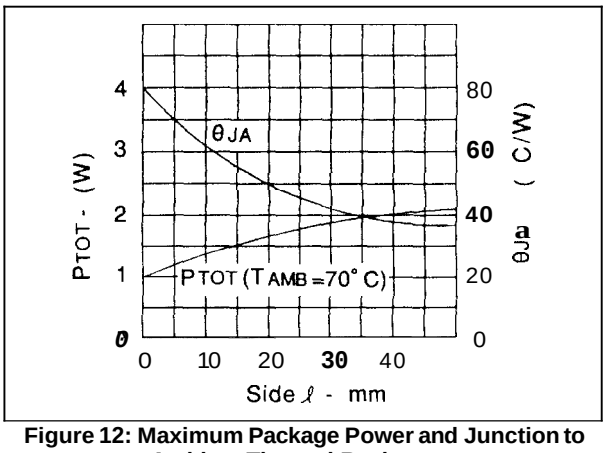



**Figure 12: Maximum Package Power and Junction to Ambient Thermal Resistance**

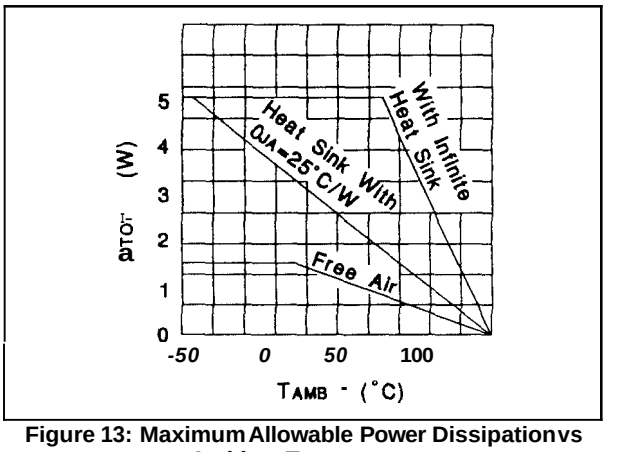



**Figure 13: Maximum Allowable Power Dissipation vs Ambient Temperature**

UNITRODE INTEGRATED CIRCUITS
7 CONTINENTAL BLVD. • MERRIMACK, NH 03054
TEL: (603) 424-2410 • FAX (603) 424-3460