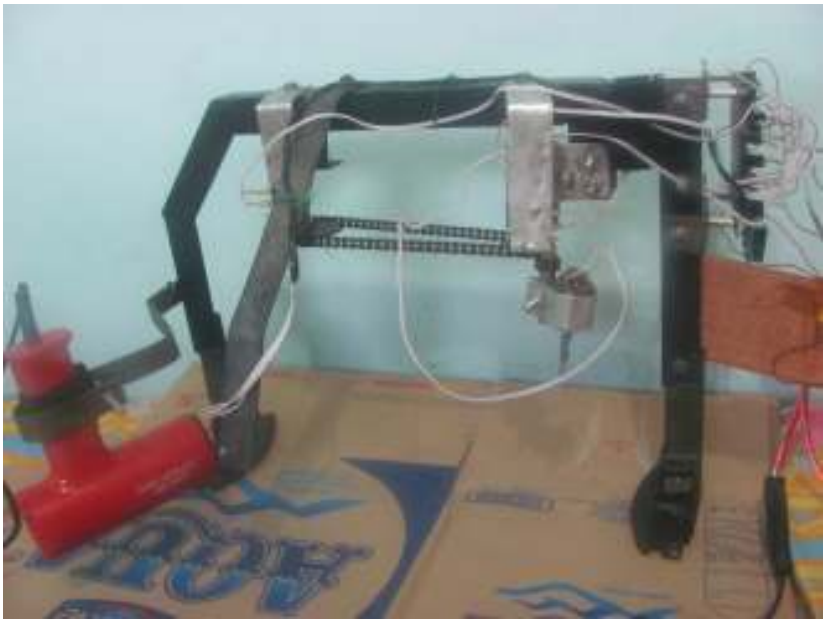
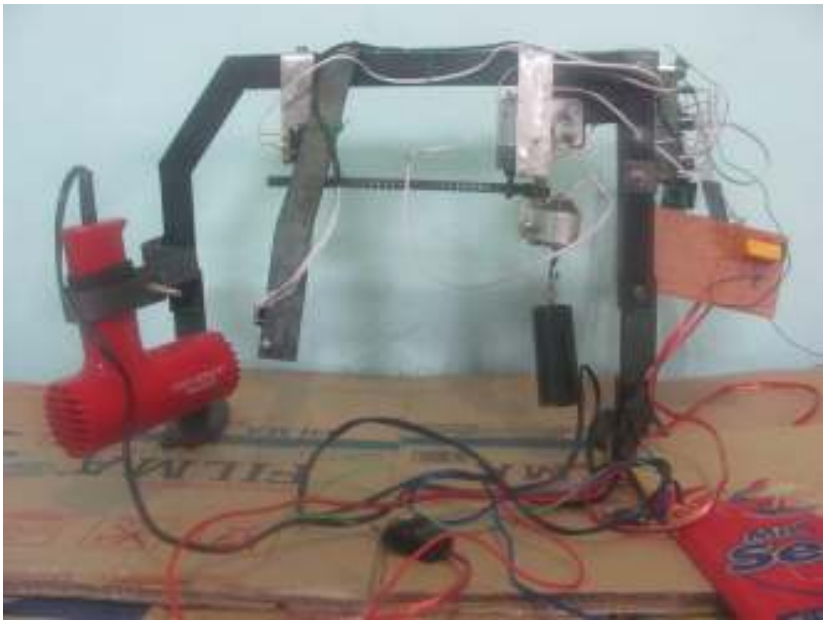
LAMPIRAN A
FOTO REALISASI ALAT

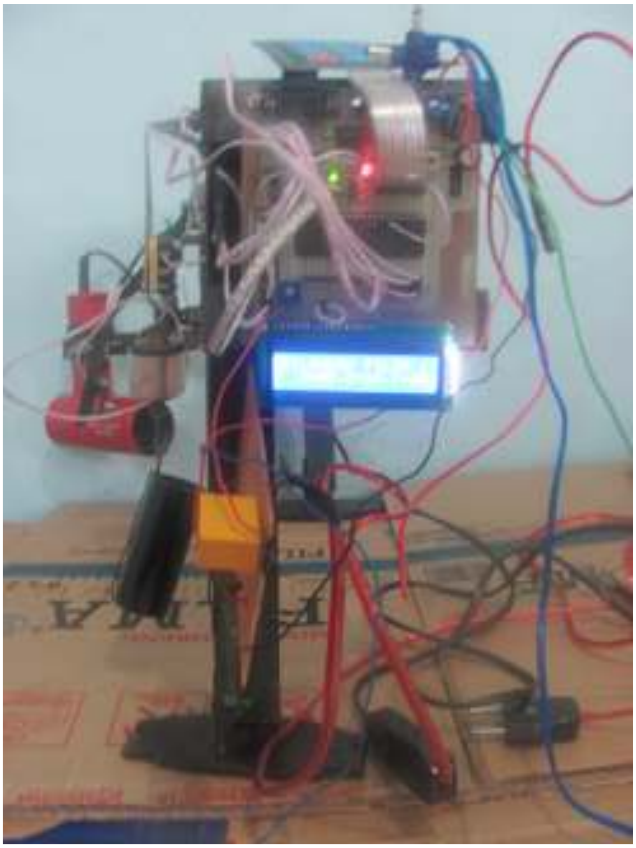
TAMPAK DEPAN



TAMPAK BELAKANG

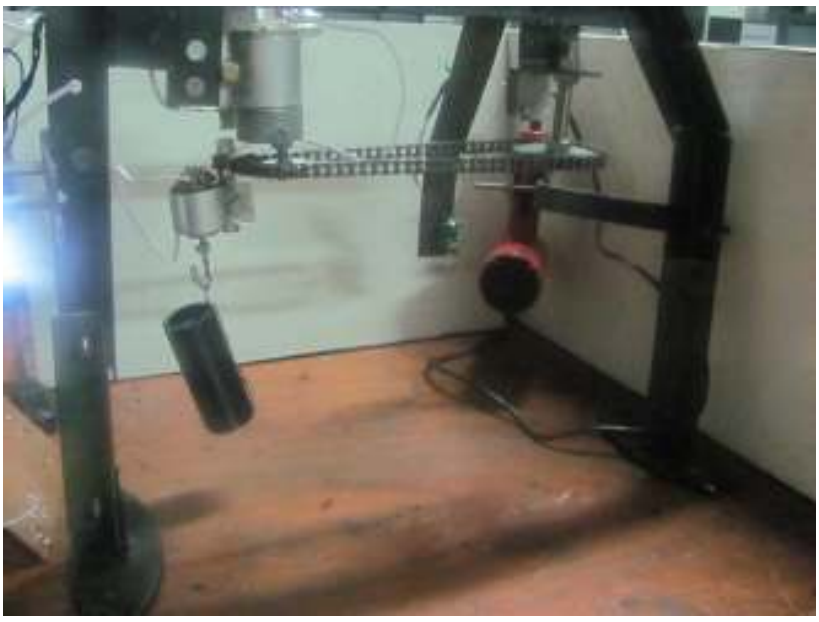


TAMPAK SAMPING



PEMBACAAN LCD

PROSES PENGERINGAN



PERBANDINGAN PEMBACAAN SENSOR TPA 81 DENGAN DIGITAL THERMOMETER CONSTANT 20T

LAMPIRAN B
PROGRAM PADA PENGONTROL MIKRO
AT MEGA 16

```
/*****************************************************
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    : 3/18/2011
Author  : F4CG
Company : F4CG
Comments:


Chip type           : ATmega16
Program type        : Application
Clock frequency     : 11.059300 MHz
Memory model        : Small
External SRAM size  : 0
Data Stack size     : 256
*****************************************************/

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


unsigned int a,b,c,d,time,n,m,g;
unsigned char text[32];
unsigned char temp,temp1,temp2,temp3,temp4,temp5,temp6;
eeprom  unsigned char
tmp[50],tmp1[50],tmp2[50],tmp3[50],tmp4[50],tmp5[50],tmp6[50];
eeprom  unsigned int t;
// I2C Bus functions
#asm
   .equ __i2c_port=0x1B ;PORTA
   .equ __sda_bit=6
   .equ __scl_bit=7
#endasm
#include <i2c.h>

// Alphanumeric LCD Module functions
#asm
   .equ __lcd_port=0x15 ;PORTC
#endasm
#include <lcd.h>
```

```
// 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=Out Func1=In
Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=0 State1=P State0=P
PORTA=0x03;
DDRA=0x04;

// Port B initialization
// Func7=In Func6=In Func5=In Func4=In Func3=Out Func2=Out Func1=Out
Func0=Out
// State7=P State6=P State5=P State4=P State3=1 State2=1 State1=1 State0=0
PORTB=0xFE;
DDRB=0x0F;

// 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=Out Func6=Out Func5=Out Func4=Out Func3=Out Func2=Out
Func1=In Func0=In
// State7=0 State6=0 State5=0 State4=0 State3=0 State2=0 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: Off
// USART Transmitter: On
// USART Mode: Asynchronous
// USART Baud rate: 9600
UCSRA=0x00;
UCSRB=0x08;
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;

// I2C Bus initialization
i2c_init();

// LCD module initialization
lcd_init(16);
t=0;
while (1)
      {
      // Place your code here

        ulang:
        a=0;

        if(PINB.4==0)
        {
        lcd_clear();
        sprintf(text,"keluarkan data");
        lcd_puts(text);
        delay_ms(1000);
        t=0;
           for(t=0;t<=50;t++)
           {
           lcd_clear();
            printf("%d %d %d %d %d %d %d
t=%d\n\r",tmp[t],tmp1[t],tmp2[t],tmp3[t],tmp4[t],tmp5[t],tmp6[t],t);
           sprintf(text,"%d %d %d %d %d %d %d
t=%d",tmp[t],tmp1[t],tmp2[t],tmp3[t],tmp4[t],tmp5[t],tmp6[t],t);
           lcd_puts(text);
           delay_ms(700);
```

```
   }
goto next;
}
else if(PINB.5==0)
{
lcd_clear();
sprintf(text,"C");
lcd_puts(text);
delay_ms(1000);
d=60;
goto next;
}
else if(PINB.6==0)
{
lcd_clear();
sprintf(text,"B");
lcd_puts(text);
delay_ms(1000);
d=58;//suhu
goto next;
}
else if(PINB.7==0)
{
lcd_clear();
sprintf(text,"A");
lcd_puts(text);
delay_ms(1000);
d=55;//suhu
goto next;
}

lcd_clear();
sprintf(text," SILAHKAN PILIH: A=55,B=58,C=60");
lcd_puts(text);
delay_ms(1000);
goto ulang;



next:
lcd_clear();
sprintf(text,"PILIHAN ANDA :  suhu %d",d);
lcd_puts(text);
delay_ms(1000);
```

```
        while(PINA.0==1)    //maju sampai switch ditekan
        {
        PORTD.6=1;//pwm1
        PORTD.7=0;//dir11
        PORTD.2=1;//dir12
        }
        PORTD.6=0;//pwm1
        PORTD.3=0;//pwm2

        PORTD.3=1;//pwm2
           PORTD.5=1;//dir22
           PORTD.4=0;//dir21
           delay_ms(1);
           PORTD.3=0;//pwm2
           delay_ms(2);


        //tunggu kering
        while(a<=1)
        {
        PORTA.2=1;
        PORTD.0=0;//pwm1
//           for(b=0;b<1000;b++)
//           {
//           PORTD.3=1;//pwm2
//           PORTD.5=1;//dir22
//           PORTD.4=0;//dir21
//           delay_ms(1);
//           PORTD.3=0;//pwm2
//           delay_ms(1);   //speed motor putar
//           }
           for(c=0;c<400;c++)
           {
           PORTD.3=1;//pwm2
           PORTD.4=1;//dir21
           PORTD.5=0;//dir22
           delay_ms(1);
           PORTD.3=0;//pwm2
           delay_us(1000); //speed motor putar
           }

        PORTD.6=0;//pwm1
        PORTD.3=0;//pwm2
        delay_ms(100);
        i2c_start();
        i2c_write(0xD0);
```

```
        i2c_write(0x01);
        i2c_start();
        i2c_write(0xD1);
        temp=i2c_read(1);
        temp1=i2c_read(1);
        temp2=i2c_read(1);
        temp3=i2c_read(1);
        temp4=i2c_read(1);
        temp5=i2c_read(1);
        temp6=i2c_read(0);
        i2c_stop();
        lcd_clear();
        sprintf(text,"%d %d %d %d %d %d %d a=%d
t=%d",temp,temp1,temp2,temp3,temp4,temp5,temp6,a,time);
        lcd_puts(text);
        tmp[t]=temp;
        tmp1[t]=temp1;
        tmp2[t]=temp2;
        tmp3[t]=temp3;
        tmp4[t]=temp4;
        tmp5[t]=temp5;
        tmp6[t]=temp6;
        t++;
        time++;
        if(temp1>d)
        {

//            if(time<180 )
//            {
             if(d==55 && time<180 )
             {
                for(m=0;m<10;m++) //berhenti 10 detik  untuk 55
                {
                PORTA.2=0;
                delay_ms(1000);
                time++;
                }
             }
             else if(d==58 && time<210)
             {
                for(n=0;n<10;n++) //berhenti 10 detik  untuk 58
                {
                PORTA.2=0;
                delay_ms(1000);
                time++;
                }
```

```
            }
            else if( d==60 && time<240)
            {
               for(g=0;g<10;g++) //berhenti 10 detik untuk 60
               {
               PORTA.2=0;
               delay_ms(1000);
               time++;
               }
            }
//          }
         else if((time>=180 && d==55) || (time>=210 && d==58) ||(time>=240
&& d==60))
         {
         a++;
         }
      }
      if((time>=180 && d==55) || (time>=210 && d==58) ||(time>=240 &&
d==60)) a++;
      }
      PORTA.2=0;
      time=0;




      a=0;
      lcd_clear();
      sprintf(text,"KERING");
      lcd_puts(text);
      delay_ms(1000);
      PORTD.6=0;//pwm1
      PORTD.3=0;//pwm2
      while(PINA.1==1)    //m sampai switch ditekan
      {
      PORTD.6=1;//pwm1
      PORTD.7=1;//dir11
      PORTD.2=0;//dir12
      }
      PORTD.6=0;//pwm1
      PORTD.3=0;//pwm2

      lcd_clear();
      sprintf(text,"FINISH");
      lcd_puts(text);
      delay_ms(1000);
```

```
        goto ulang;

    };
}
```

LAMPIRAN C
DATASHEET

TPA81 Thermopile Array

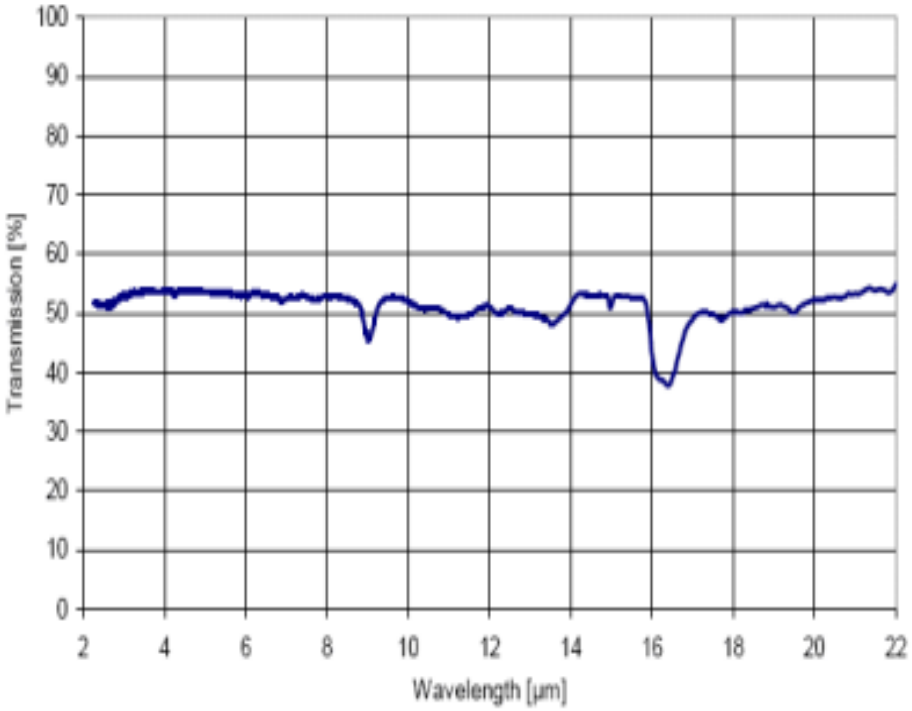
Technical Specification

Introduction

The TPA81 is a thermopile array detecting infra-red in the 2um-22um range. This is the wavelength of radiant heat. The Pyro-electric sensors that are used commonly in burglar alarms and to switch on outside lights, detect infra-red in the same waveband. These Pyro-electric sensors can only detect a change in heat levels though - hence they are movement detectors. Although useful in robotics, their applications are limited as they are unable to detect and measure the temperature of a static heat source. Another type of sensor is the thermopile array. These are used in non-contact infra-red thermometers. They have a very wide detection angle or field of view (FOV) of around 100° and need either shrouding or a lens or commonly both to get a more useful FOV of around 12°. Some have a built in lens. More recently sensors with an array of thermopiles, built in electronics and a silicon lens have become available. This is the type used in the TPA81. It has a array of eight thermopiles arranged in a row. The TPA81 can measure the temperature of 8 adjacent points simultaneously. The TPA81 can also control a servo to pan the module and build up a thermal image. The TPA81 can detect a candle flame at a range 2 metres (6ft) and is unaffected by ambient light!

Spectral Response

The response of the TPA81 is typically 2µm to 22µm and is shown below:



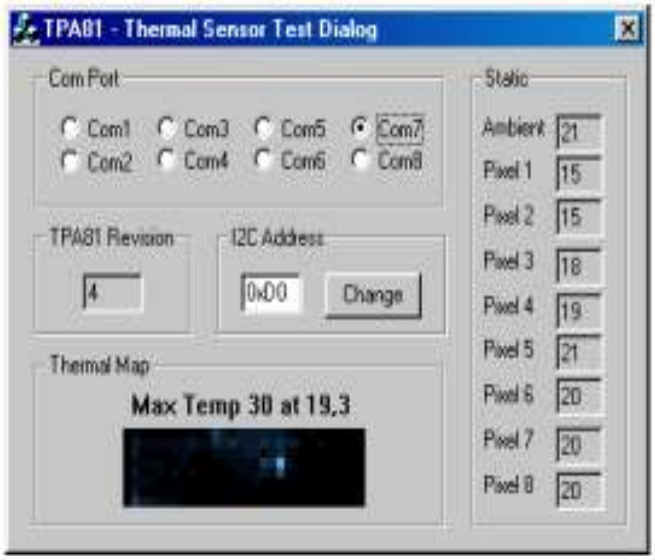
The typical field of view of the TPA81 is 41° by 6° making each of the eight pixels 5.12° by 6°. The array of eight pixels is orientated along the length of the PCB - that's from top to bottom in the diagram below. Pixel number one is nearest the tab on the sensor - or at the bottom in the diagram below.

Sensitivity

Here's some numbers from one of our test modules:
For a candle, the numbers for each of the eight pixels at a range of 1 meter in a cool room at 12°C are:
11 10 11 12 12 29 15 13 (All °C)
You can see the candle showing up as the 29°C reading. At a range of 2 meters this reduces to 20°C - still around 8° C above ambient and easily
detectable. At 0.6 meter (2ft) its around 64°C. At 0.3 meter (1ft) its 100°C+.

In a warmer room at 18°C, the candle measures 27°C at 2 meters. This is because the candle only occupies a small part of the sensors field of view and the candles point heat source is
added to the back ground ambient - not swamped by it. A human at 2 meters will show up as around 29°C with a background 20°C ambient.

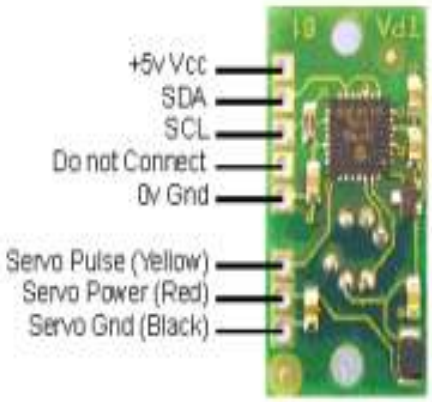
The following is a snapshot of our test program. It displays a 32x8 bitmap produced by using a servo to pan the sensor. If you want a copy of this windows based program, its here, but you will need an RF04/CM02 to connect the TPA81 to your PC. Here you can see a candle flame about a meter away showing up as the bright spot.



Connections

All communication with the TPA81 is via the I2C bus. If you are unfamiliar with the I2C bus, there is a tutorial which will help. The TPA81 uses our standard I2C 5 pin connection layout. The "Do Not Connect" pin should be left unconnected. It is actually the CPU MCLR line and is used once only in our workshop to program the PIC16F88 on-board after assembly, and has an internal pull-up resistor. The SCL and SDA lines should each have a pull-up resistor to +5v somewhere on the I2C bus. You only need one pair of resistors, not a pair for every module. They are normally located with the bus master rather than the slaves. The TPA81 is always a slave - never a bus master. If you need them, I recommend 1.8k resistors. Some modules such as the OOPic already have pull-up

resistors and you do not need to add any more. A servo port will connect directly to a standard RC servo and is powered from the modules 5v supply. We use an HS311. Commands can be sent to the TPA81 to position the servo, the servo pulses are generated by the TPA81 module.



Registers

The TPA81 appears as a set of 10 registers.

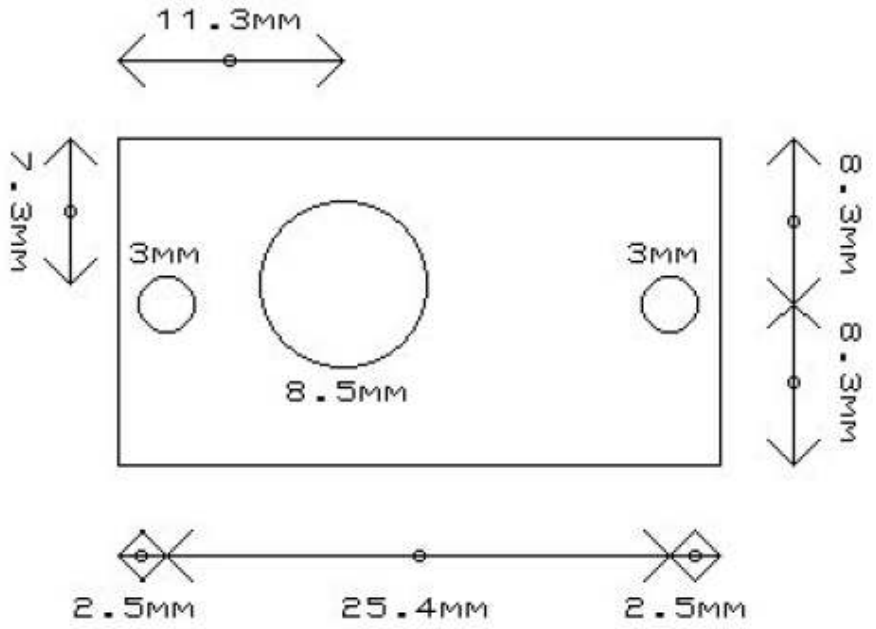
Register	Read	Write
0	Software Revision	Command Register
1	Ambient Temperature °C	Servo Range (V6 or higher only)
2	Pixel 1 Temperature °C	N/A
3	Pixel 2	N/A
4	Pixel 3	N/A
5	Pixel 4	N/A
6	Pixel 5	N/A
7	Pixel 6	N/A
8	Pixel 7	N/A
9	Pixel 8	N/A

Only registers 0, and 1 can be written to. Register 0 is the command register and is used to set the servo position and also when changing the TPA81's I2C address. It cannot be read. Reading from register 0 returns the TPA81 software revision. Writing to register 1 sets the servo range - see below. It cannot be read back, reading register 1 reads the ambient temperature.

There are 9 temperature readings available, all in degrees centigrade (°C). Register 1 is the ambient temperature as measured within the sensor. Registers 2-9 are the 8 pixel temperatures. Temperature acquisition is continuously performed and the readings will be correct approx 40mS after the sensor points to a new position.

Servo Position

Commands 0 to 31 set the servo position. There are 32 steps (0-31) which typically represent 180° rotation on a Hitec HS311 servo. The calculation is SERVO_POS*60+540uS. So the range of the servo pulse is 0.54mS to


11.3MM
7.3MM
3MM
3MM
8.5MM
8.3MM
8.3MM
2.5MM
25.4MM
2.5MM

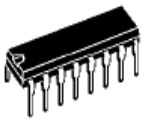
SGS-THOMSON MICROELECTRONICS

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.

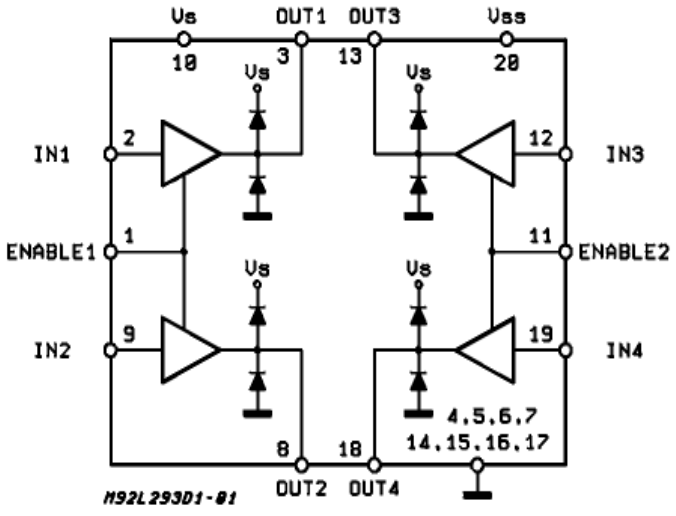
To simplify use as two bridges each pair of channels is equipped with an enable input. A separate supply input is provided for the logic, allowing operation at a lower voltage and internal clamp diodes are included.

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.

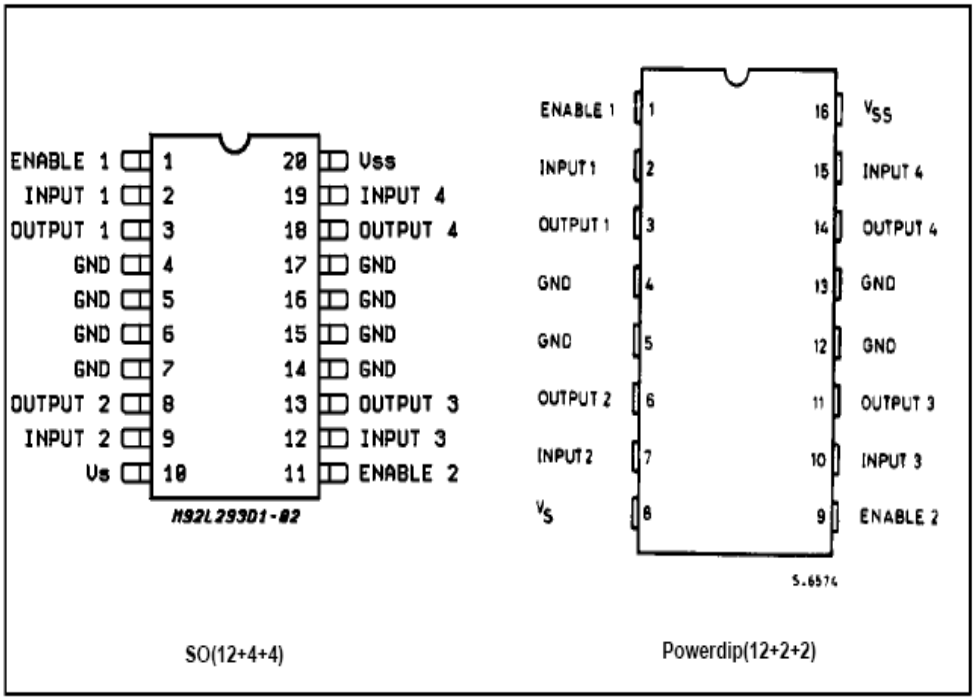
BLOCK DIAGRAM



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.

ELECTRICAL CHARACTERISTICS (for each channel, V_S = 24 V, V_{SS} = 5 V, T_{amb} = 25 °C, unless otherwise specified)

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
V_S	Supply Voltage (pin 10)		V_{SS}		36	V
V_{SS}	Logic Supply Voltage (pin 20)		4.5		36	V
I_S	Total Quiescent Supply Current (pin 10)	V_i = L ; I_O = 0 ; V_{en} = H		2	6	mA
		V_i = H ; I_O = 0 ; V_{en} = H		16	24	mA
		V_{en} = L			4	mA
I_{SS}	Total Quiescent Logic Supply Current (pin 20)	V_i = L ; I_O = 0 ; V_{en} = H		44	60	mA
		V_i = H ; I_O = 0 ; V_{en} = H		16	22	mA
		V_{en} = L		16	24	mA
V_{IL}	Input Low Voltage (pin 2, 9, 12, 19)		– 0.3		1.5	V
V_{IH}	Input High Voltage (pin 2, 9, 12, 19)	$V_{SS} \leq 7$ V	2.3		V_{SS}	V
		$V_{SS} > 7$ V	2.3		7	V
I_{IL}	Low Voltage Input Current (pin 2, 9, 12, 19)	V_{IL} = 1.5 V			– 10	µA
I_{IH}	High Voltage Input Current (pin 2, 9, 12, 19)	2.3 V ≤ V_{IH} ≤ V_{SS} – 0.6 V		30	100	µA
$V_{en\,L}$	Enable Low Voltage (pin 1, 11)		– 0.3		1.5	V
$V_{en\,H}$	Enable High Voltage (pin 1, 11)	$V_{SS} \leq 7$ V	2.3		V_{SS}	V
		$V_{SS} > 7$ V	2.3		7	V
$I_{en\,L}$	Low Voltage Enable Current (pin 1, 11)	$V_{en\,L}$ = 1.5 V		– 30	– 100	µA
$I_{en\,H}$	High Voltage Enable Current (pin 1, 11)	2.3 V ≤ $V_{en\,H}$ ≤ V_{SS} – 0.6 V			± 10	µA
$V_{CE(sat)H}$	Source Output Saturation Voltage (pins 3, 8, 13, 18)	I_O = – 0.6 A		1.4	1.8	V
$V_{CE(sat)L}$	Sink Output Saturation Voltage (pins 3, 8, 13, 18)	I_O = + 0.6 A		1.2	1.8	V
V_F	Clamp Diode Forward Voltage	I_O = 600nA		1.3		V
t_r	Rise Time (*)	0.1 to 0.9 V_O		250		ns
t_f	Fall Time (*)	0.9 to 0.1 V_O		250		ns
t_{on}	Turn-on Delay (*)	0.5 V_i to 0.5 V_O		750		ns
t_{off}	Turn-off Delay (*)	0.5 V_i to 0.5 V_O		200		ns

(*) See fig. 1.

TRUTH TABLE (one channel)

Input	Enable (*)	Output
H	H	H
L	H	L
H	L	Z
L	L	Z

Z = High output impedance
(*) Relative to the considered channel

Figure 1: Switching Times

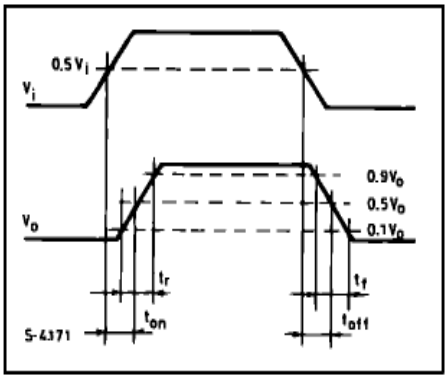
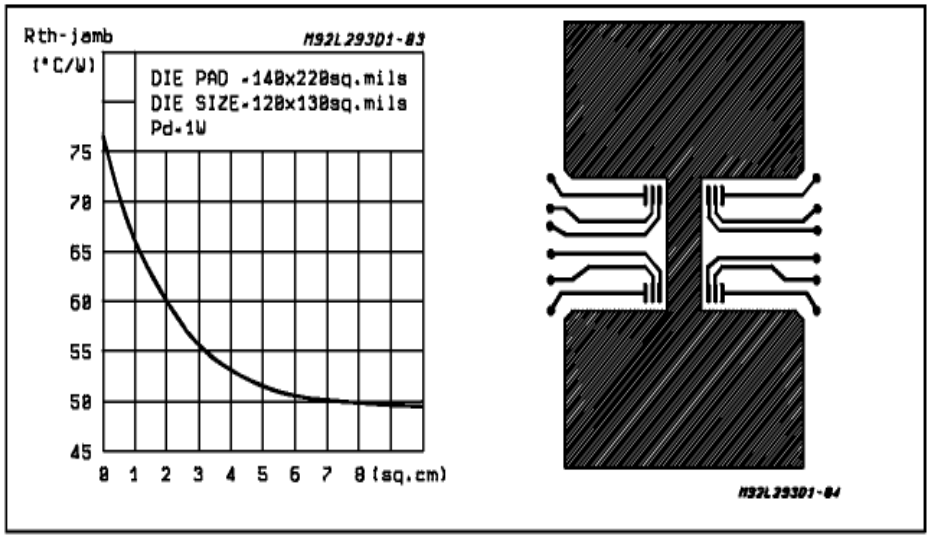
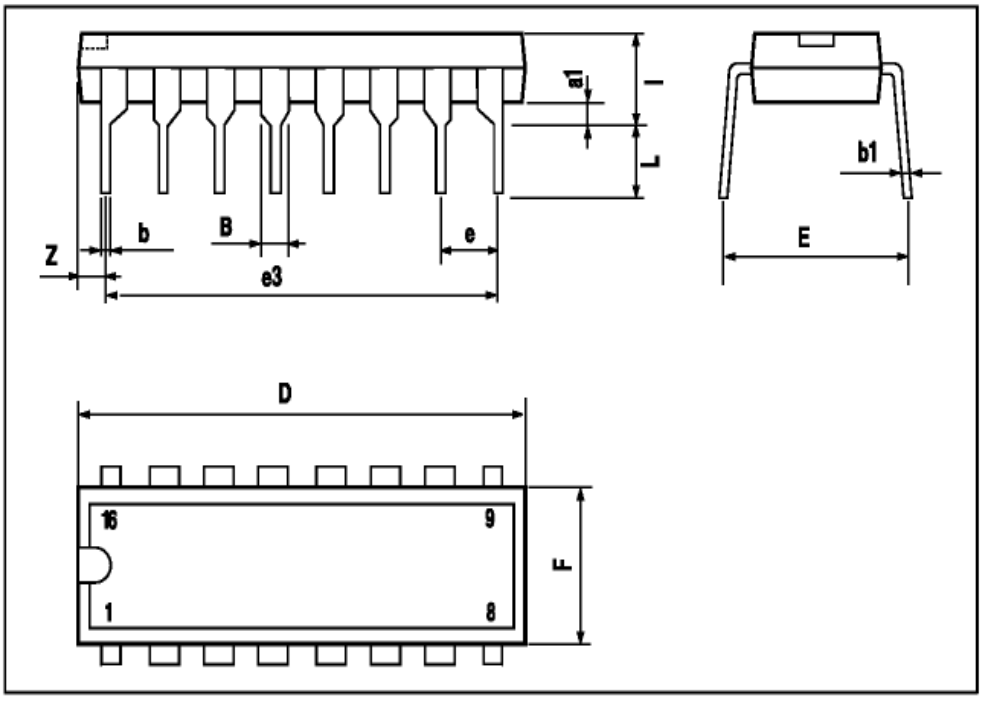


Figure 2: Junction to ambient thermal resistance vs. area on board heatsink (SO12+4+4 package)



POWERDIP16 PACKAGE MECHANICAL DATA

DIM.	mm			inch		
	MIN.	TYP.	MAX.	MIN.	TYP.	MAX.
a1	0.51			0.020		
B	0.85		1.40	0.033		0.055
b		0.50			0.020	
b1	0.38		0.50	0.015		0.020
D			20.0			0.787
E		8.80			0.346	
e		2.54			0.100	
e3		17.78			0.700	
F			7.10			0.280
I			5.10			0.201
L		3.30			0.130	
Z			1.27			0.050



SO20 PACKAGE MECHANICAL DATA

DIM.	mm			inch		
	MIN.	TYP.	MAX.	MIN.	TYP.	MAX.
A			2.65			0.104
a1	0.1		0.2	0.004		0.008
a2			2.45			0.096
b	0.35		0.49	0.014		0.019
b1	0.23		0.32	0.009		0.013
C		0.5			0.020	
c1		45			1.772	
D		1	12.6		0.039	0.496
E	10		10.65	0.394		0.419
e		1.27			0.050	
e3		11.43			0.450	
F		1	7.4		0.039	0.291
G	8.8		9.15	0.346		0.360
L	0.5		1.27	0.020		0.050
M			0.75			0.030
S	8° (max.)