Anexos

5.1. Diagrama electrónico

En la figura 5.1 se muestra el dibujo esquemático del sistema electrónico desarrollado para el presente proyecto de tesis.

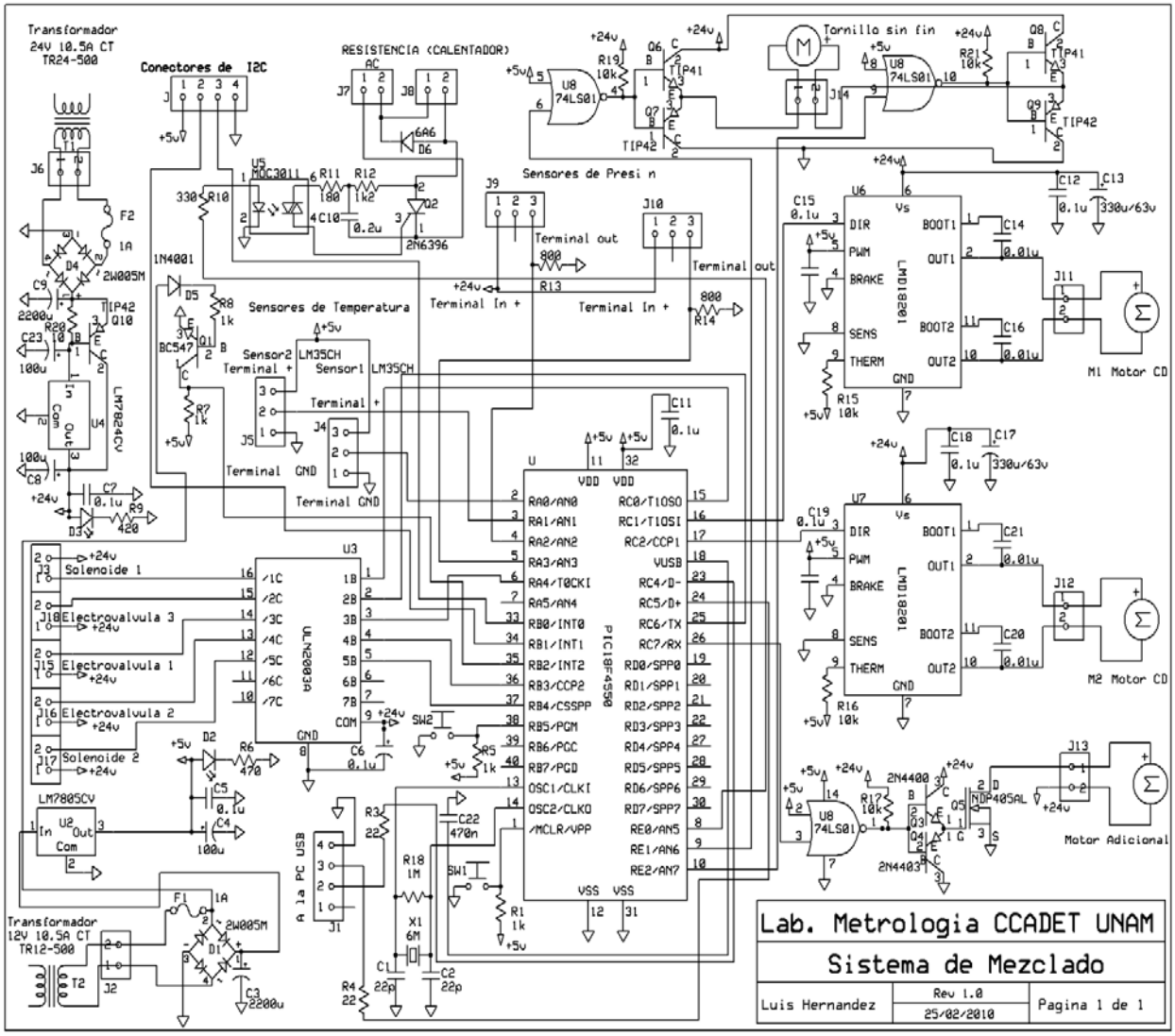


Figura 5. 1 Sistema electrónico.

5.2. Circuito impreso

El circuito impreso desarrollado se muestra en las figuras 5.2 y figura 5.3

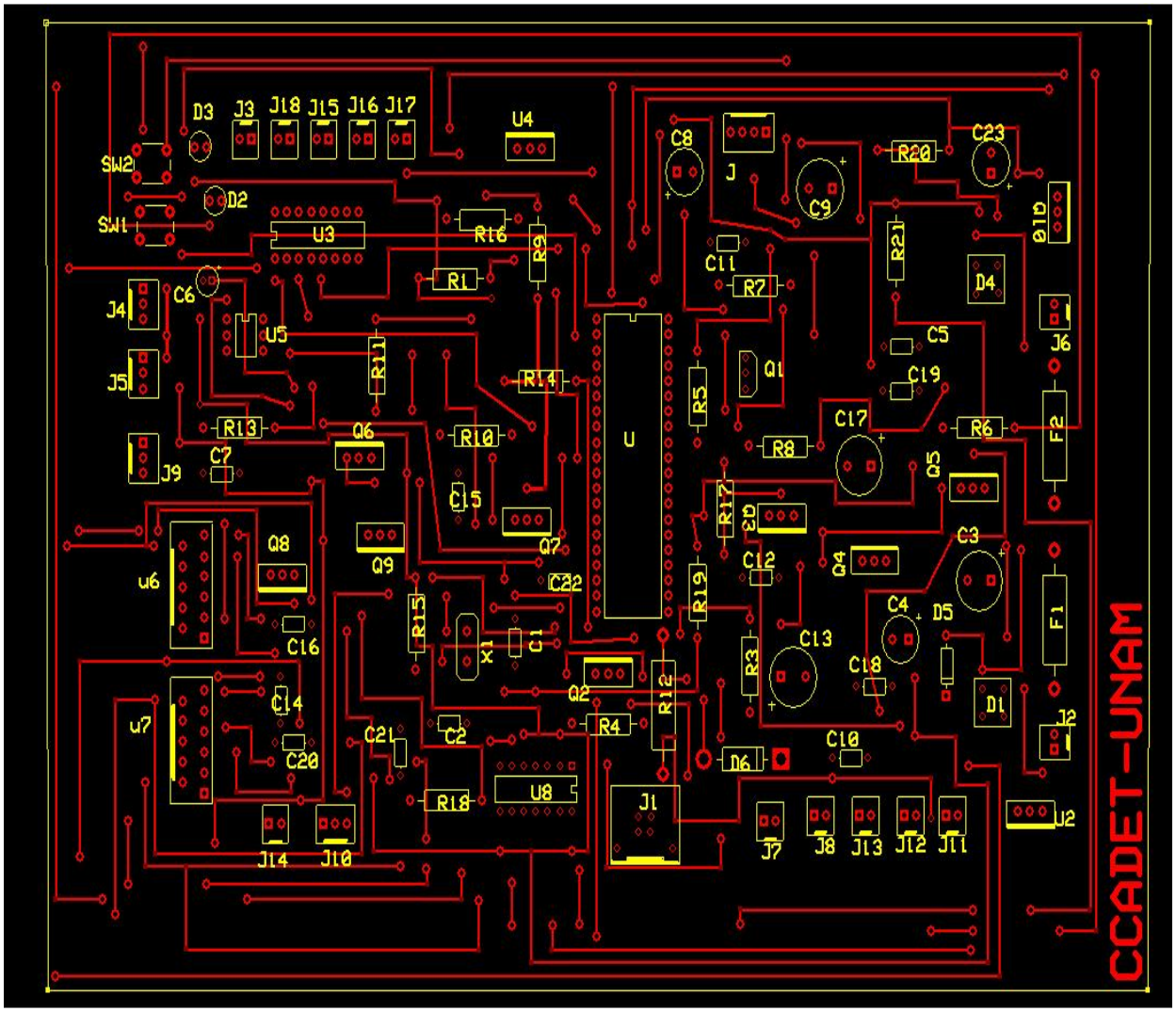


Figura 5. 2 Cara superior del circuito impreso

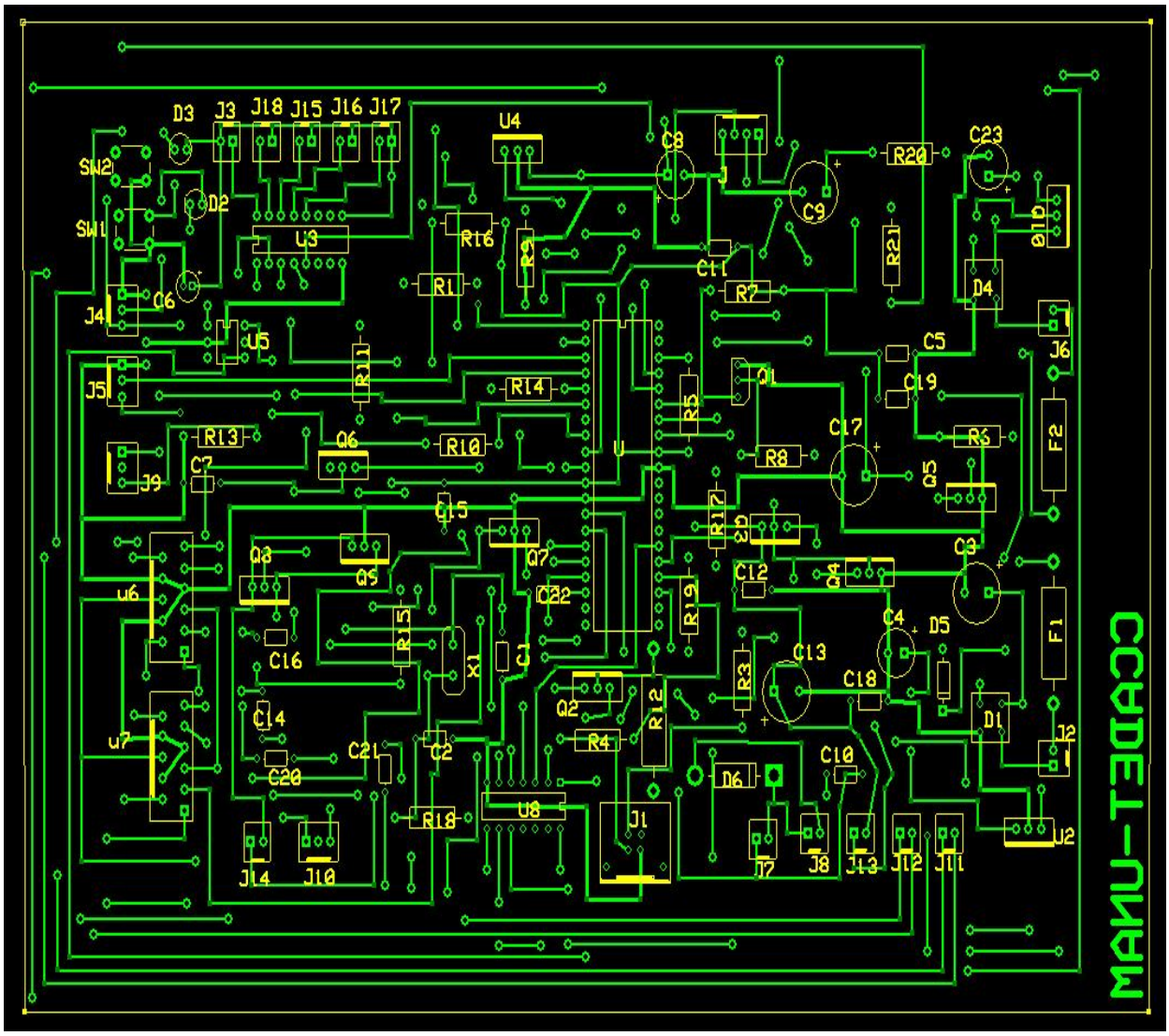


Figura 5. 3 Cara inferior del circuito impreso

5.3. Lista de Materiales

La lista del material utilizado para la elaboración del sistema electrónico es la que se muestra en la tabla 5.1.

Cantidad	Numero de parte	Componente
4	C14, C16, C20, C21	Capacitor monolítico de 0.01uF/50V
9	C5, C6, C7,C11, C12, C19, C15, C18,	Capacitor monolítico de 0.1uF/50V
2	C1, C2	Capacitor cerámico de 22pF
2	C3,C9	Capacitor electrolítico de 2200uF/63V
3	C4,C8,C23	Capacitor electrolítico de 100uF/63V
2	C13,C17	Capacitor electrolítico de 330uF/63V
1	C10	Capacitor de poliéster de 0.47uF
1	C22	Capacitor cerámico de 0.2uF
2	R3, R4	Resistencia de carbón de 22Ω
2	R6	Resistencia de carbón de 470Ω
5	R15, R16,R17,R19, R21	Resistencia de carbón de 10kΩ
1	R20	Resistencia de carbón de 10Ω
4	R1,R5,R7, R8,R9	Resistencia de carbón de 1kΩ
2	R13,R14	Resistencia de carbón de 820Ω
1	R12	Resistencia de carbón de 1k5Ω a 2W
1	R11	Resistencia de carbón de 180Ω
1	R10	Resistencia de carbón de 330Ω
1	R18	Resistencia de 1MΩ
2	D2,D3	Led
2	F1,F2	Fusibles europeos de 2A
1	U	Base para circuito integrado de 40 patas
1	U3	Base para circuito integrado de 16 patas
1	U8	Base para circuito integrado de 8 patas
1	U5	Optoaislador salida triac (MOC3011M)
1	Q2	T-SCR 12A/200V SCR (2N6396)
2	Q6,Q8	TIP 41A
3	Q7,Q9,Q10	TIP 42C
2	U6,U7	Switch H Completo (LMD18201T)
1	Q3	T-NPN 600MA/40V (2N4400)
1	Q4	SI-PNP 40V 0.6A (2N4403)
1	Q5	T-C-N 15A/50V (NDP405AL)
1	U	Circuito integrado PIC18F4550
1	U8	Compuerta NAND (HD74LS01P)
2	D1,D4	Puente Rectificador (2W005M)
1	Q1	SI-NPN 50V 0.2A (BC547B)
1	U2	Regulador de voltaje L7805CV
1	U4	Regulador de voltaje L7824CV
1	X1	Cristal 20MHz
1	U3	Arreglo de 7 Transistores Darlington (ULN2003AP)

1	J1	Conector USB tipo B
1	D5	Diodo Rectificador (1N4001)
1	D6	Rectificador 6A/600V (6A6)
2	SW1,SW2	Microswitch push mini
15	J,J2,J3, J6, J7, J8, J11, J12, J13, J14, J15,J16,J17,J18	Bornera p/ Circuito Impreso 2 Terminales
4	J4, J5, J9, J10	Bornera p/ Circuito Impreso 3 Terminales
6	Q6, Q7, Q8, Q9, Q2, U2	Disipador de calor (TO-220)
2	U4,Q10	Disipador electrónico de 38mm(TO-300)
1	T1	Transformador 24V
1	T2	Transformador 12V

Tabla 5. 1 Lista de material.

5.4. Programa en C para el PIC

El programa en C para PIC desarrollado para el sistema electrónico se muestra a continuación.

```
#include <18F4550.h>
#device adc=10
//#fuses HSPLL,NOWDT,NOPROTECT,NOLVP,NODEBUG,USBDIV,PLL3,CPUDIV1,VREGEN
#fuses HSPLL,NOWDT,NOPROTECT,NOLVP,NODEBUG,USBDIV,PLL5,CPUDIV1,VREGEN
#use delay(clock=48000000)
#use fixed_io(c_outputs=pin_c0,pin_c1,pin_c2,pin_c6,pin_c7)
#use fixed_io(b_outputs=pin_b3,pin_b4,pin_b7)
#use fixed_io(a_outputs=pin_a4)
#use fixed_io(e_outputs=pin_e0,pin_e1,pin_e2)

#define USB_HID_DEVICE     FALSE          //deshabilitamos el uso de las directivas HID
#define USB_EP1_TX_ENABLE  USB_ENABLE_BULK   //turn on EP1(EndPoint1) for IN bulk/interrupt
transfers
#define USB_EP1_RX_ENABLE  USB_ENABLE_BULK   //turn on EP1(EndPoint1) for OUT
bulk/interrupt transfers
#define USB_EP1_TX_SIZE    2             //size to allocate for the tx endpoint 1 buffer
#define USB_EP1_RX_SIZE    4             //size to allocate for the rx endpoint 1 buffer
////////////////////////////////////////////////////////////////////////////
//
// Include the CCS USB Libraries.
//
////////////////////////////////////////////////////////////////////////////
#include <pic18_usb.h>     //Microchip PIC18Fxx5x Hardware layer for CCS's PIC USB driver
#include <PicUSB.h>        //Configuración del USB y los descriptores para este dispositivo
#include <usb.c>          //handles usb setup tokens and get descriptor reports
//*********************************************************************//
//    Definición del envió y recepción de datos
//*********************************************************************//
#define modo      recibe[0]
#define param1    recibe[1]
#define param2    recibe[2]
```

```
#define param     recibe[3]
#define cad_lw    envia[0]
#define cad_hi    envia[1]
 //*************************************************************************//
//**************Función para configuración PWM inicial **********************//
//*************************************************************************//
void pwm_inicial(void)
{
  setup_ccp1(CCP_PWM); //modo PWM CCP1CON<3:0>
  setup_ccp2(CCP_PWM);
  setup_timer_2(T2_DIV_BY_1,255,1); //modo = T2CON<2:0> (precaler),periodo = 255
(PR2),postcaler= T2CON<6:3>
  set_pwm1_duty(511);   //valor de 10 bits
  set_pwm2_duty(511);
}
//*************************************************************************//
//*****************Función para configureación conversion A/D ***************//
//*************************************************************************//
void cad_inicial(void)
{
 setup_adc_ports(AN0_TO_AN3|VSS_VDD);
 setup_adc(ADC_CLOCK_INTERNAL);
}
//*************************************************************************//
int16 alfa;
boolean banderaC;
#int_ext2
void intExterna2()
{
  output_bit(PIN_E0,0);
  banderaC=true;
  setup_timer_1(T1_INTERNAL|T1_DIV_BY_2); // (4/48M)*(65536-1000)*2=5.378ms
  set_timer1(alfa);
  enable_interrupts(INT_TIMER1);
  enable_interrupts(GLOBAL);
}
#int_timer1
void intTimer1()
{
  if(banderaC)
  {
    banderaC=false;
    output_bit(PIN_E0,1);
    delay_ms(1);
    output_bit(PIN_E0,0);
  }
}
void main(void) {
```

```
  int8 recibe[4];              //declaramos variables
  int8 envia[2];
  int16 valor;
  int16 Temp1;
  int32 acumulador;
  int32 resultado;
  int8 i;
  alfa = 16000;
  //***************Configuración PWM ****************************************//
  pwm_inicial();      //Configuiración PWM inicial, 50% ciclo util y PWM=46.875 KHz
  //*************************************************************************//
   //************************************************************************//
  cad_inicial();
  //************************************************************************//
  usb_init();                  //inicializamos el USB
  usb_task();                   //habilita periferico usb e interrupciones
  usb_wait_for_enumeration();     //esperamos hasta que el PicUSB sea configurado por el host
// inicia puertos
  output_bit(PIN_E0,0);
  delay_ms(200);
  output_bit(PIN_C0,0);
  output_bit(PIN_B4,0);
  output_bit(PIN_A4,0);
  output_bit(PIN_B3,0);
  output_bit(PIN_C6,0);
  output_bit(PIN_C7,0);
  output_bit(PIN_E1,1);
  output_bit(PIN_E2,1);
// inicia intExterna
  banderaC=false;
  ext_int_edge(INT_EXT2,L_TO_H);
  enable_interrupts(INT_EXT2);
  enable_interrupts(GLOBAL);
  while (TRUE)
  {
    if(usb_enumerated())        //si el PicUSB está configurado
    {
      if (usb_kbhit(1))        //si el endpoint de salida contiene datos del host
      {
        usb_get_packet(1, recibe, 3); //tomamos el paquete de tamaño 3bytes del EP1 y
almacenamos en recibe
        if (modo == 0) // Modo_MotorPWM 1
        {
          valor = param1+(param2*256);
          setup_ccp1(CCP_PWM); //  Configuración del PWM
          setup_timer_2(T2_DIV_BY_1,255,1);
          set_pwm1_duty(valor);
        }
```

```
if (modo == 1) // Modo_MotorPWM 2
{
  valor = param1 + (param2*256);
  setup_ccp2(CCP_PWM);
  setup_timer_2(T2_DIV_BY_1,255,1);
  set_pwm2_duty(valor);
}
if (modo == 2) // Modo_Solenoide 1
{
  output_bit(PIN_C0,param1);
}
if (modo == 3) // Modo_Solenoide 2
{
  output_bit(PIN_B4,param1);
}
if (modo == 4) // Modo_Electrovalvula 1
{
  output_bit (PIN_A4,param1);
}
if (modo == 5) // Modo_Electrovalvula 2
{
  output_bit (PIN_B3,param1);
}
if (modo == 6) // Modo_Electrovalvula 3
{
  output_bit (PIN_C6,param1);
}
if (modo == 7) // Modo_Motor Adicional
{
  output_bit (PIN_C7,param1);
}
if (modo == 8) // Modo_Tornillo_Sin_Fin
{
  if (param2 == 0) // Parar Tornillo
  {
   output_bit(PIN_E1,1);
   output_bit(PIN_E2,1);
  }
  if (param2 == 1) // Giro a la izquierda
  {
    output_bit(PIN_E1,0);
    output_bit(PIN_E2,1);
  }
  if (param2 == 2) // Giro a la derecha
  {
    output_bit(PIN_E1,1);
    output_bit(PIN_E2,0);
  }
```

```
          if (param2 == 3)
          {
          output_bit(PIN_E1,0);
          output_bit(PIN_E2,0);
          }
        }
        if (modo == 9) // Control_ de_Fase
        {
          alfa = param1+(param2*256);
        }
        if (modo == 10) // Modo_Convertidor A/D Temperatura 1
        {
          acumulador = 0;
          for ( i=1; i<=100; i++)
          {
           set_adc_channel(0);
           delay_us(200);
           resultado = read_adc();
           acumulador = acumulador + resultado;
          }
          Temp1 = (int16)(acumulador/100);
          cad_lw = Temp1 & 0xFF;
          cad_hi = Temp1 >> 8;
          usb_put_packet(1, envia, 2, USB_DTS_TOGGLE); //enviamos el paquete de tamaño 1byte
del EP1 al PC
        }
        if (modo == 11) // Modo_Convertidor A/D Temperatura 2
        {
         acumulador = 0;
          for ( i=1; i<=100; i++)
          {
           set_adc_channel(1);
           delay_us(200);
           resultado = read_adc();
           acumulador = acumulador + resultado;
          }
          Temp1 = (int16)(acumulador/100);
          cad_lw = Temp1 & 0xFF;
          cad_hi = Temp1 >> 8;
         usb_put_packet(1, envia, 2, USB_DTS_TOGGLE);
        }
        if (modo == 12) // Modo_Convertidor A/D Presión 1
        {
        acumulador = 0;
          for ( i=1; i<=100; i++)
          {
           set_adc_channel(2);
           delay_us(200);
```

```
          resultado = read_adc();
          acumulador = acumulador + resultado;
         }
         Temp1 = (int16)(acumulador/100);
         cad_lw = Temp1 & 0xFF;
         cad_hi = Temp1 >> 8;
         usb_put_packet(1, envia, 2, USB_DTS_TOGGLE);
       }
       if (modo == 13) // Modo_convertidor A/D Presión 2
       {
        acumulador = 0;
         for ( i=1; i<=100; i++)
         {
          set_adc_channel(3);
          delay_us(200);
          resultado = read_adc();
          acumulador = acumulador + resultado;
         }
         Temp1 = (int16)(acumulador/100);
         cad_lw = Temp1 & 0xFF;
         cad_hi = Temp1 >> 8;
         usb_put_packet(1, envia, 2, USB_DTS_TOGGLE);
       }
      }
   }
  }
}
```

5.5. Programa en C# para PC

A continuación se muestra un fragmento del programa realizado para implementar la interface de operación. Debido a su gran extensión, solamente se muestran los archivos relevantes. El siguiente código representa las funciones de comunicación entre la interface de operación y el sistema electrónico.

```
public void PWM_PIC(uint Bajo, uint Alto)
        {
            //byte Bajo;
            //byte Alto;
            String cadena;//Declaracion de una variable
            byte* send_buf = stackalloc byte[3];
            send_buf[0] = 0x00;    // Código de Entrada a Modo_PWM
            send_buf[1] = (byte)Bajo;
            send_buf[2] = (byte)Alto;
            SendPacket(send_buf, 3);
            cadena = String.Format("Bajo={0:D},Alto ={1:D}", Bajo, Alto);
            MessageBox.Show(cadena);
        }
        public void PWM2_PIC(uint Bajo, uint Alto)
        {
```

```
        String Cadena;
        byte* send_buf = stackalloc byte[3];
        send_buf[0] = 0x01;
        send_buf[1] = (byte)Bajo;
        send_buf[2] = (byte)Alto;
        SendPacket(send_buf, 3);
        Cadena = String.Format("Bajo={0:D},Alto={1:d}", Bajo, Alto);
        MessageBox.Show(Cadena);
    }
    public void PrendePIC()
    {
        byte* send_buf = stackalloc byte[3];
        send_buf[0] = 0x02;    // Código de Entrada a Modo_Suma
        send_buf[1] = 0x01;
        send_buf[2] = 0x01;
        SendPacket(send_buf, 3);
    }
    public void ApagaPIC()
    {
        byte* send_buf = stackalloc byte[3];
        send_buf[0] = 0x02;
        send_buf[1] = 0x00;
        send_buf[2] = 0x00;
        SendPacket(send_buf, 3);
    }
    public void PrenderSolenoide2()
    {
        byte* send_buf = stackalloc byte[3];
        send_buf[0] = 0x03;
        send_buf[1] = 0x01;
        send_buf[2] = 0x01;
        SendPacket(send_buf, 3);
    }
    public void ApagarSolenoide2()
    {
        byte* send_buf = stackalloc byte[3];
        send_buf[0] = 0x03;
        send_buf[1] = 0x00;
        send_buf[2] = 0x00;
        SendPacket(send_buf, 3);
    }
    public void PrenderElectrovalvula1()
    {
        byte* send_buf = stackalloc byte[3];
        send_buf[0] = 0x04;
        send_buf[1] = 0x01;
        send_buf[2] = 0x01;
        SendPacket(send_buf, 3);
    }
    public void ApagarElectrovalvula1()
    {
        byte* send_buf = stackalloc byte[3];
        send_buf[0] = 0x04;
        send_buf[1] = 0x00;
        send_buf[2] = 0x00;
        SendPacket(send_buf, 3);
    }
```

```
public void PrenderElectrovalvula2()
{
    byte* send_buf = stackalloc byte[3];
    send_buf[0] = 0x05;
    send_buf[1] = 0x01;
    send_buf[2] = 0x01;
    SendPacket(send_buf, 3);
}
public void ApagarElectrovalvula2()
{
    byte* send_buf = stackalloc byte[3];
    send_buf[0] = 0x05;
    send_buf[1] = 0x00;
    send_buf[2] = 0x00;
    SendPacket(send_buf, 3);
}
public void PrenderElectrovalvula3()
{
    byte* send_buf = stackalloc byte[3];
    send_buf[0] = 0x06;
    send_buf[1] = 0x01;
    send_buf[2] = 0x01;
    SendPacket(send_buf, 3);
}
public void ApagarElectrovalvula3()
{
    byte* send_buf = stackalloc byte[3];
    send_buf[0] = 0x06;
    send_buf[1] = 0x00;
    send_buf[2] = 0x00;
    SendPacket(send_buf, 3);
}
public void PrenderMotorAdicional()
{
    byte* send_buf = stackalloc byte[3];
    send_buf[0] = 0x07;
    send_buf[1] = 0x01;
    send_buf[2] = 0x01;
    SendPacket(send_buf, 3);
}
public void ApagarMotorAdicional()
{
    byte* send_buf = stackalloc byte[3];
    send_buf[0] = 0x07;
    send_buf[1] = 0x00;
    send_buf[2] = 0x00;
    SendPacket(send_buf, 3);
}
public void IzquierdaPIC()
{
    byte* send_buf = stackalloc byte[3];
    send_buf[0] = 0x08;
    send_buf[1] = 0x00;
    send_buf[2] = 0x01;
    SendPacket(send_buf, 3);
}
public void DerechaPIC()
```

```
        {
            byte* send_buf = stackalloc byte[3];
            send_buf[0] = 0x08;
            send_buf[1] = 0x00;
            send_buf[2] = 0x02;
            SendPacket(send_buf, 3);
        }
        public void PararPIC()
        {
            byte* send_buf = stackalloc byte[3];
            send_buf[0] = 0x08;
            send_buf[1] = 0x00;
            send_buf[2] = 0x00;
            SendPacket(send_buf, 3);
        }
        public void ControlFase_PIC(uint Bajo, uint Alto)
        {
            String cadena; // Declaracion de una variable
            byte* send_buf = stackalloc byte[3];
            send_buf[0] = 0x09;    // Código de Entrada a Control de Fase
            send_buf[1] = (byte)Bajo;
            send_buf[2] = (byte)Alto;
            SendPacket(send_buf, 3);
         }
        public void ConvCA1PIC()
        {
            byte* send_buf = stackalloc byte[1];
            send_buf[0] = 0x0A;// Código de Entrada a
Modo_Convertidor_A/D TEMP1
            SendPacket(send_buf, 1);
        }
        public void ConvCA2PIC()
        {
            byte* send_buf = stackalloc byte[1];
            send_buf[0] = 0x0B;
            SendPacket(send_buf, 1);
        }
        public void ConvCA3PIC()
        {
            byte* send_buf = stackalloc byte[1];
            send_buf[0] = 0x0C;
            SendPacket(send_buf, 1);
        }
        public void ConvCA4PIC()
        {
            byte* send_buf = stackalloc byte[1];
            send_buf[0] = 0x0D;
            SendPacket(send_buf, 1);
        }
        public float Temp1PIC()
        {
            uint cad_lw;
            uint cad_hi;
            uint result;
            float tempf1;
            float tempf;
            int temp;
```

```
            byte* receive_buf = stackalloc byte[2];
            DWORD RecvLength = 2;
            ReceivePacket(receive_buf, &RecvLength);
            cad_lw = receive_buf[0];
            cad_hi = receive_buf[1];
            result = cad_lw + (cad_hi * 256);
            tempf1 = (float)(result);
            tempf = (float)(((tempf1) * (98.0/204)) - 4);
            temp = (int)(tempf);
            return (temp);
        }
        public float Temp2PIC()
        {
            uint cad_lw;
            uint cad_hi;
            uint result;
            float tempf1;
            float tempf;
            int temp;
            byte* receive_buf = stackalloc byte[2];
            DWORD RecvLength = 2;
            ReceivePacket(receive_buf, &RecvLength);
            cad_lw = receive_buf[0];
            cad_hi = receive_buf[1];
            result = cad_lw + (cad_hi * 256);
            tempf1 = (float)(result);
            tempf = (float)(((tempf1) * (98.0/204)) - 4);
            temp = (int)(tempf);
            return (temp);
        }
        public float Presion1PIC()
        {
            uint cad_lw;
            uint cad_hi;
            uint result;
            float presf1;
            float presf;
            byte* receive_buf = stackalloc byte[2];
            DWORD RecvLength = 2;
            ReceivePacket(receive_buf, &RecvLength);
            cad_lw = receive_buf[0];
            cad_hi = receive_buf[1];
            result = cad_lw + (cad_hi * 256);
            presf1 = (float)(result);
            presf = (float)(((presf1) * 0.002197) + 0.002197);
            return presf1;
        }
        public float Presion2PIC()
        {
            uint cad_lw;
            uint cad_hi;
            uint result;
            float presf1;
            float presf;
            byte* receive_buf = stackalloc byte[2];
            DWORD RecvLength = 2;
            ReceivePacket(receive_buf, &RecvLength);
```

```
            cad_lw = receive_buf[0];
            cad_hi = receive_buf[1];
            result = cad_lw + (cad_hi * 256);
            presf1 = (float)(result);
            presf = (float)(((presf1) * 3.125) - 10);
            return presf1;
        }
```

El siguiente código muestra la configuracion de las propiedades del panel de control.

```
namespace prueba
{
    public partial class frmprueba : Form
    {
        PicUSBAPI usbapi = new PicUSBAPI();
        public frmprueba()
        {
            InitializeComponent();
        }
        Boolean Bandera;
        private void PIC_PWM_Click(object sender, EventArgs e)
        {
            uint pwm1;
            uint Bajo;
            uint Alto;
            pwm1 = uint.Parse(Valor1.Text);
            Bajo = pwm1 & 0xFF;
            Alto = pwm1 >> 8;
            usbapi.PWM_PIC(Bajo,Alto);
        }
        private void PIC_PWM2_Click(object sender, EventArgs e)
        {
            uint pwm1;
            uint Bajo;
            uint Alto;
            pwm1 = uint.Parse(Valor2.Text);
            Bajo = pwm1 & 0xFF;
            Alto = pwm1 >> 8;
            usbapi.PWM2_PIC(Bajo, Alto);
        }
        private void Valor2_TextChanged(object sender, EventArgs e)
        {

        }
        private void button1_Click(object sender, EventArgs e)
        {
            if (Bandera == false)
            {
                usbapi.PrendePIC();
                Bandera = true;
            }
            else
            {
                usbapi.ApagaPIC();
                Bandera = false;
            }
        }
        private void Soleniode2_Click(object sender, EventArgs e)
```

```
{
    if (Bandera == false)
    {
        usbapi.PrenderSolenoide2();
        Bandera = true;
    }
    else
    {
        usbapi.ApagarSolenoide2();
        Bandera = false;
    }
}
private void Electrovalvula1_Click(object sender, EventArgs e)
{
    if (Bandera == false)
    {
        usbapi.PrenderElectrovalvula1();
        Bandera = true;
    }
    else
    {
        usbapi.ApagarElectrovalvula1();
        Bandera = false;
    }
}
private void Electrovalvula2_Click(object sender, EventArgs e)
{
    if (Bandera == false)
    {
        usbapi.PrenderElectrovalvula2();
        Bandera = true;
    }
    else
    {
        usbapi.ApagarElectrovalvula2();
        Bandera = false;
    }
}
private void Electrovalvula3_Click(object sender, EventArgs e)
{
    if (Bandera == false)
    {
        usbapi.PrenderElectrovalvula3();
        Bandera = true;
    }
    else
    {
        usbapi.ApagarElectrovalvula3();
        Bandera = false;
    }
}
private void MotorAdicional_Click(object sender, EventArgs e)
{
    if (Bandera == false)
    {
        usbapi.PrenderMotorAdicional();
        Bandera = true;
```

```
            }
            else
            {
                usbapi.ApagarMotorAdicional();
                Bandera = false;
            }
        }
        private void PICizq_Click(object sender, EventArgs e)
        {
            usbapi.IzquierdaPIC();
        }
        private void button2_Click(object sender, EventArgs e)
        {
            usbapi.PararPIC();
        }
        private void PICder_Click(object sender, EventArgs e)
        {
            usbapi.DerechaPIC();
        }
        private void PIC_FASE_Click(object sender, EventArgs e)
        {
            uint fase;
            uint Bajo;
            uint Alto;
            fase = uint.Parse(Valor3.Text);
            Bajo = fase & 0xFF;
            Alto = fase >> 8;
            usbapi.ControlFase_PIC(Bajo, Alto);
        }
        private void PIC_CAD1_Click(object sender, EventArgs e)
        {
            usbapi.ConvCA1PIC();
            Temp1.Text = (usbapi.Temp1PIC()).ToString();
        }
        private void PIC_CAD2_Click(object sender, EventArgs e)
        {
            usbapi.ConvCA2PIC();
            Temp2.Text = (usbapi.Temp2PIC()).ToString();
        }
        private void PIC_CAD3_Click(object sender, EventArgs e)
        {
            usbapi.ConvCA3PIC();
            Presion1.Text = (usbapi.Presion1PIC()).ToString();
        }
        private void PIC_CAD4_Click(object sender, EventArgs e)
        {
            usbapi.ConvCA4PIC();
            Presion2.Text = (usbapi.Presion2PIC()).ToString();
        }
    }
}
```

5.6. Datasheet para el puente H integrado

National Semiconductor

September 1996

LMD18201 3A, 55V H-Bridge

General Description

The LMD18201 is a 3A H-Bridge designed for motion control applications. The device is built using a multi-technology process which combines bipolar and CMOS control circuitry with DMOS power devices on the same monolithic structure. The H-Bridge configuration is ideal for driving DC and stepper motors. The LMD18201 accommodates peak output currents up to 6A. Current sensing can be achieved via a small sense resistor connected in series with the power ground lead. For current sensing without disturbing the path of current to the load, the LMD18200 is recommended.

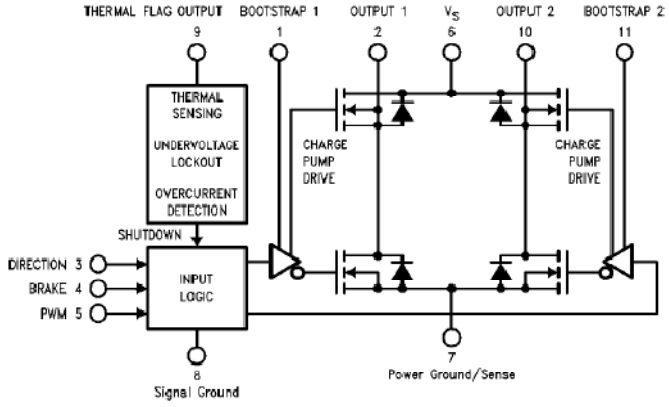
Features

- Delivers up to 3A continuous output
- Operates at supply voltages up to 55V
- Low $R_{DS(ON)}$ typically 0.33Ω per switch
- TTL and CMOS compatible inputs
- No "shoot-through" current
- Thermal warning flag output at 145°C
- Thermal shutdown (outputs off) at 170°C
- Internal clamp diodes
- Shorted load protection
- Internal charge pump with external bootstrap capability

Applications

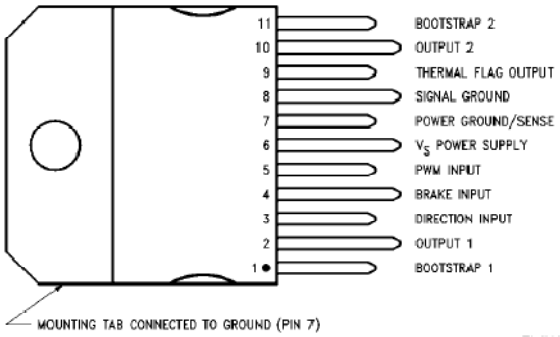
- DC and stepper motor drives
- Position and velocity servomechanisms
- Factory automation robots
- Numerically controlled machinery
- Computer printers and plotters

Functional Diagram



TL/H/10793–1

Connection Diagram and Ordering Information



TL/H/10793–2

Top View

Order Number LMD18201T
See NS Package Number TA11B

Absolute Maximum Ratings (Note 1)

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Total Supply Voltage (V_S, Pin 6)	60V
Voltage at Pins 3, 4, 5 and 9	12V
Voltage at Bootstrap Pins (Pins 1 and 11)	V_{OUT} + 16V
Peak Output Current (200 ms)	6A
Continuous Output Current (Note 2)	3A
Power Dissipation (Note 3)	25W
Sense Voltage (Pin 7 to Pin 8)	+0.5V to −1.0V
Power Dissipation (T_A = 25°C, Free Air)	3W
Junction Temperature, $T_{J(max)}$	150°C
ESD Susceptibility (Note 4)	1500V
Storage Temperature, T_{STG}	−40°C to +150°C
Lead Temperature (Soldering, 10 sec.)	300°C

Operating Ratings (Note 1)

Junction Temperature, T_J	−40°C to +125°C
V_S Supply Voltage	+12V to +55V

Electrical Characteristics

The following specifications apply for V_S = 42V, unless otherwise specified. **Boldface** limits apply over the entire operating temperature range, −40°C ≤ T_J ≤ +125°C, all other limits are for T_A = T_J = 25°C. (Note 5)

Symbol	Parameter	Conditions	Typ	Limit	Units
$R_{DS(ON)}$	Switch ON Resistance	Output Current = 3A (Note 6)	0.33	0.4/**0.6**	Ω (max)
$R_{DS(ON)}$	Switch ON Resistance	Output Current = 6A (Note 6)	0.33	0.4/**0.6**	Ω (max)
V_{CLAMP}	Clamp Diode Forward Drop	Clamp Current = 3A (Note 6)	1.2	1.5	V (max)
V_{IL}	Logic Low Input Voltage	Pins 3, 4, 5		**−0.1** **0.8**	V (min) V (max)
I_{IL}	Logic Low Input Current	V_{IN} = −0.1V, Pins = 3, 4, 5		**−10**	µA (max)
V_{IH}	Logic High Input Voltage	Pins 3, 4, 5		**2** **12**	V (min) V (max)
I_{IL}	Logic High Input Current	V_{IN} = 12V, Pins = 3, 4, 5		**10**	µA (max)
	Undervoltage Lockout	Outputs Turn OFF		9 11	V (min) V (max)
T_{JW}	Warning Flag Temperature	Pin 9 ≤ 0.8V, I_L = 2 mA	145		°C
$V_{F(ON)}$	Flag Output Saturation Voltage	T_J = T_{JW}, I_L = 2 mA	0.15		V
$I_{F(OFF)}$	Flag Output Leakage	V_F = 12V	0.2	10	µA (max)
T_{JSD}	Shutdown Temperature	Outputs Turn OFF	170		°C
I_S	Quiescent Supply Current	All Logic Inputs Low	13	25	mA (max)
$t_{D(ON)}$	Output Turn-On Delay Time	Sourcing Outputs, I_{OUT} = 3A Sinking Outputs, I_{OUT} = 3A	300 300		ns ns
t_{ON}	Output Turn-On Switching Time	Bootstrap Capacitor = 10 nF Sourcing Outputs, I_{OUT} = 3A Sinking Outputs, I_{OUT} = 3A	 100 80		 ns ns
$t_{D(OFF)}$	Output Turn-Off Delay Times	Sourcing Outputs, I_{OUT} = 3A Sinking Outputs, I_{OUT} = 3A	200 200		ns ns
t_{OFF}	Output Turn-Off Switching Times	Bootstrap Capacitor = 10 nF Sourcing Outputs, I_{OUT} = 3A Sinking Outputs, I_{OUT} = 3A	 75 70		 ns ns
t_{PW}	Minimum Input Pulse Width	Pins 3, 4 and 5	1		µs
t_{CPR}	Charge Pump Rise Time	No Bootstrap Capacitor	20		µs

Electrical Characteristics (Continued)

Note 1: Absolute Maximum Ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications do not apply when operating the device beyond its rated operating conditions.

Note 2: See Application Information for details regarding current limiting.

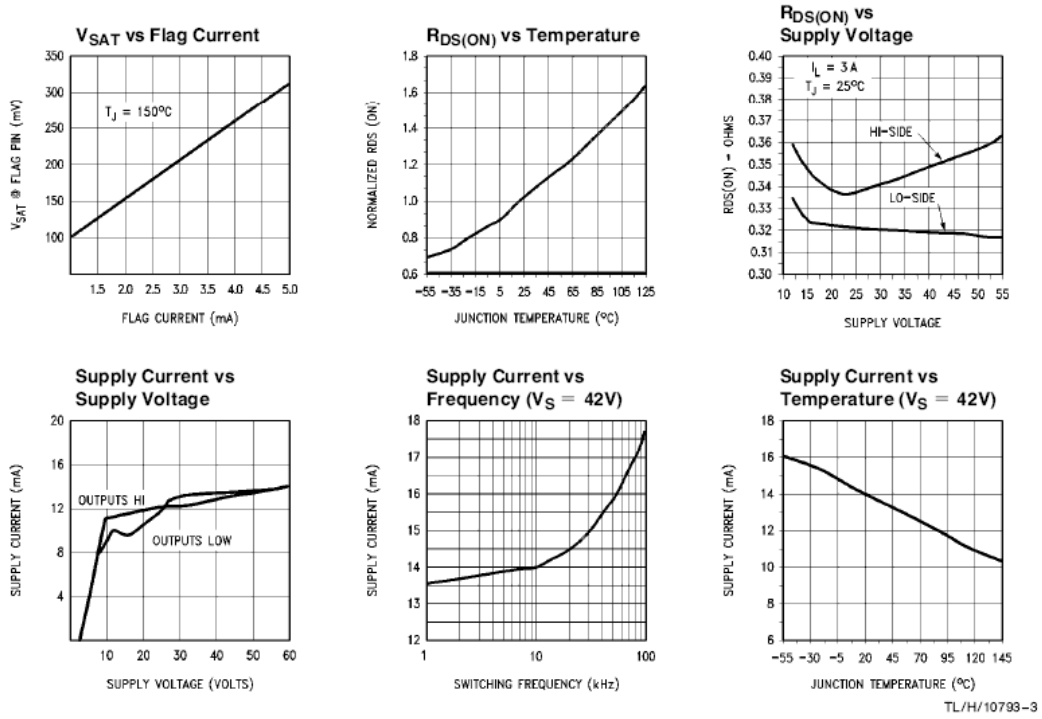
Note 3: The maximum power dissipation must be derated at elevated temperatures and is a function of $T_{J(max)}$, θ_{JA}, and T_A. The maximum allowable power dissipation at any temperature is $P_{D(max)} = (T_{J(max)} - T_A)/\theta_{JA}$, or the number given in the Absolute Ratings, whichever is lower. The typical thermal resistance from junction to case (θ_{JC}) is 1.0°C/W and from junction to ambient (θ_{JA}) is 30°C/W. For guaranteed operation $T_{J(max)} = 125$°C.

Note 4: Human-body model, 100 pF discharged through a 1.5 kΩ resistor. Except Bootstrap pins (pins 1 and 11) which are protected to 1000V of ESD.

Note 5: All limits are 100% production tested at 25°C. Temperature extreme limits are guaranteed via correlation using accepted SQC (Statistical Quality Control) methods. All limits are used to calculate AOQL, (Average Outgoing Quality Level).

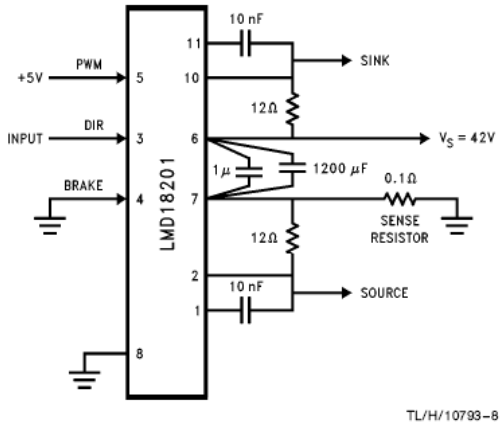
Note 6: Output currents are pulsed ($t_W < 2$ ms, Duty Cycle < 5%).

Typical Performance Characteristics



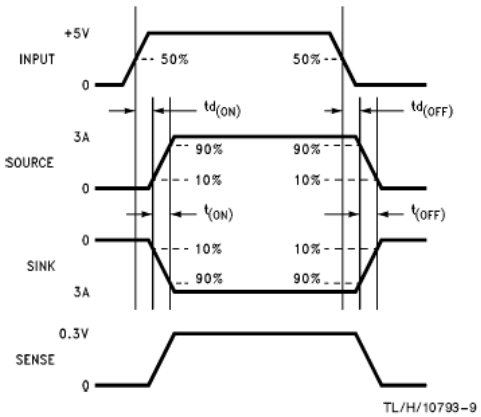
TL/H/10793–3

Test Circuit



TL/H/10793–8

Switching Time Definitions



TL/H/10793–9

Pinout Description (See Connection Diagram)

Pin 1, BOOTSTRAP 1 Input: Bootstrap capacitor pin for half H-Bridge number 1. The recommended capacitor (10 nF) is connected between pins 1 and 2.

Pin 2, OUTPUT 1: Half H-Bridge number 1 output.

Pin 3, DIRECTION Input: See Table I. This input controls the direction of current flow between OUTPUT 1 and OUTPUT 2 (pins 2 and 10) and, therefore, the direction of rotation of a motor load.

Pin 4, BRAKE Input: See Table I. This input is used to brake a motor by effectively shorting its terminals. When braking is desired, this input is taken to a logic high level and it is also necessary to apply logic high to PWM input, pin 5. The drivers that short the motor are determined by the logic level at the DIRECTION input (Pin 3): with Pin 3 logic high, both current sourcing output transistors are ON; with Pin 3 logic low, both current sinking output transistors are ON. All output transistors can be turned OFF by applying a logic high to Pin 4 and a logic low to PWM input Pin 5; in this case only a small bias current (approximately −1.5 mA) exists at each output pin.

Pin 5, PWM Input: See Table I. How this input (and DIRECTION input, Pin 3) is used is determined by the format of the PWM Signal.

Pin 6, V_S Power Supply

Pin 7, POWER GROUND/SENSE Connection: This pin is the ground return for the power DMOS transistors of the H-Bridge. The current through the H-Bridge can be sensed by adding a small, 0.1Ω, sense resistor from this pin to the power supply ground.

Pin 8, SIGNAL GROUND: This is the ground return for the internal logic circuitry used to control the PWM switching of the H-Bridge.

Pin 9, THERMAL FLAG Output: This pin provides the thermal warning flag output signal. Pin 9 becomes active-low at 145°C (junction temperature). However the chip will not shut itself down until 170°C is reached at the junction.

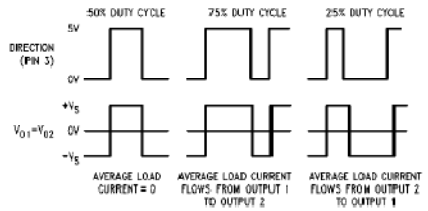
Pin 10, OUTPUT 2: Half H-Bridge number 2 output.

Pin 11, BOOTSTRAP 2 Input: Bootstrap capacitor pin for half H-Bridge number 2. The recommended capacitor (10 nF) is connected between pins 10 and 11.

TABLE I. Logic Truth Table

PWM	Dir	Brake	Active Output Drivers
H	H	L	Source 1, Sink 2
H	L	L	Sink 1, Source 2
L	X	L	Source 1, Source 2
H	H	H	Source 1, Source 2
H	L	H	Sink 1, Sink 2
L	X	H	NONE

Locked Anti-Phase PWM Control



TL/H/10793–4

Application Information

TYPES OF PWM SIGNALS

The LMD18201 readily interfaces with different forms of PWM signals. Use of the part with two of the more popular forms of PWM is described in the following paragraphs.

Simple, locked anti-phase PWM consists of a single, variable duty-cycle signal in which is encoded both direction and amplitude information. A 50% duty-cycle PWM signal represents zero drive, since the net value of voltage (integrated over one period) delivered to the load is zero. For the LMD18201, the PWM signal drives the direction input (pin 3) and the PWM input (pin 5) is tied to logic high.

Sign/magnitude PWM consists of separate direction (sign) and amplitude (magnitude) signals. The (absolute) magnitude signal is duty-cycle modulated, and the absence of a pulse signal (a continuous logic low level) represents zero drive. Current delivered to the load is proportional to pulse width. For the LMD18201, the DIRECTION input (pin 3) is driven by the sign signal and the PWM input (pin 5) is driven by the magnitude signal.

USING THE THERMAL WARNING FLAG

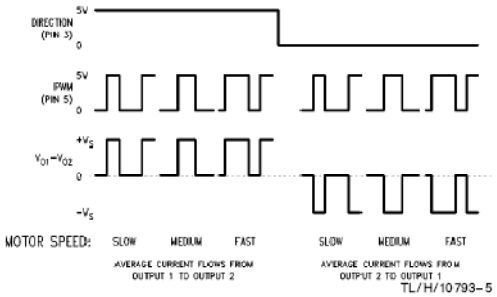
The THERMAL FLAG output (pin 9) is an open collector transistor. This permits a wired OR connection of thermal warning flag outputs from multiple LMD18201's, and allows the user to set the logic high level of the output signal swing to match system requirements. This output typically drives the interrupt input of a system controller. The interrupt service routine would then be designed to take appropriate steps, such as reducing load currents or initiating an orderly system shutdown. The maximum voltage compliance on the flag pin is 12V.

SUPPLY BYPASSING

During switching transitions the levels of fast current changes experienced may cause troublesome voltage transients across system stray inductances.

It is normally necessary to bypass the supply rail with a high quality capacitor(s) connected as close as possible to the V_S Power Supply (Pin 6) and POWER GROUND (Pin 7). A 1 μF high-frequency ceramic capacitor is recommended. Care should be taken to limit the transients on the supply pin below the Absolute Maximum Rating of the device. When operating the chip at supply voltages above 40V a voltage suppressor (transorb) such as P6KE62A is recommended from supply to ground. Typically the ceramic capacitor can be eliminated in the presence of the voltage suppressor. Note that when driving high load currents a greater amount of supply bypass capacitance (in general at least 100 μF per Amp of load current) is required to absorb the recirculating currents of the inductive loads.

Sign/Magnitude PWM Control



TL/H/10793–5

Application Information (Continued)

CURRENT LIMITING

Current limiting protection circuitry has been incorporated into the design of the LMD18201. With any power device it is important to consider the effects of the substantial surge currents through the device that may occur as a result of shorted loads. The protection circuitry monitors the current through the upper transistors and shuts off the power device as quickly as possible in the event of an overload condition (the threshold is set to approximately 10A). In a typical motor driving application the most common overload faults are caused by shorted motor windings and locked rotors. Under these conditions the inductance of the motor (as well as any series inductance in the V_{CC} supply line) serves to reduce the magnitude of a current surge to a safe level for the LMD18201. Once the device is shut down, the control circuitry will periodically try to turn the power device back on. This feature allows the immediate return to normal operation once the fault condition has been removed. While the fault remains however, the device will cycle in and out of thermal shutdown. This can create voltage transients on the V_{CC} supply line and therefore proper supply bypassing techniques are required.

The most severe condition for any power device is a direct, hard-wired ("screwdriver") long term short from an output to ground. This condition can generate a surge of current through the power device on the order of 15 Amps and require the die and package to dissipate up to 500W of power for the short time required for the protection circuitry to shut off the power device. This energy can be destructive, particularly at higher operating voltages (>30V) so some precautions are in order. Proper heat sink design is essential and it is normally necessary to heat sink the V_{CC} supply pin (pin 6) with 1 square inch of copper on the PC board.

INTERNAL CHARGE PUMP AND USE OF BOOTSTRAP CAPACITORS

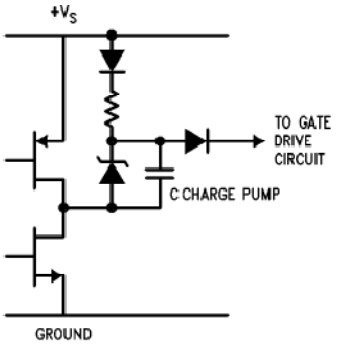
To turn on the high-side (sourcing) DMOS power devices, the gate of each device must be driven approximately 8V more positive than the supply voltage. To achieve this an internal charge pump is used to provide the gate drive voltage. As shown in *Figure 1*, an internal capacitor is alternately switched to ground and charged to about 14V, then switched to V_S thereby providing a gate drive voltage greater than V_S. This switching action is controlled by a continuously running internal 300 kHz oscillator. The rise time of this drive voltage is typically 20 μs which is suitable for operating frequencies up to 1 kHz.

For higher switching frequencies, the LMD18201 provides for the use of external bootstrap capacitors. The bootstrap principle is in essence a second charge pump whereby a large value capacitor is used which has enough energy to quickly charge the parasitic gate input capacitance of the power device resulting in much faster rise times. The switching action is accomplished by the power switches themselves *(Figure 2)*. External 10 nF capacitors, connected from the outputs to the bootstrap pins of each high-side switch provide typically less than 100 ns rise times allowing switching frequencies up to 500 kHz.

INTERNAL PROTECTION DIODES

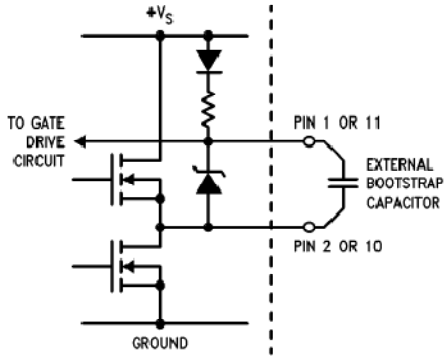
A major consideration when switching current through inductive loads is protection of the switching power devices from the large voltage transients that occur. Each of the four switches in the LMD18201 have a built-in protection diode to clamp transient voltages exceeding the positive supply or ground to a safe diode voltage drop across the switch.

The reverse recovery characteristics of these diodes, once the transient has subsided, is important. These diodes must come out of conduction quickly and the power switches must be able to conduct the additional reverse recovery current of the diodes. The reverse recovery time of the diodes protecting the sourcing power devices is typically only 70 ns with a reverse recovery current of 1A when tested with a full 3A of forward current through the diode. For the sinking devices the recovery time is typically 100 ns with 4A of reverse current under the same conditions.



TL/H/10793–6

FIGURE 1. Internal Charge Pump Circuitry



TL/H/10793–7

FIGURE 2. Bootstrap Circuitry

Typical Applications

BASIC MOTOR DRIVER

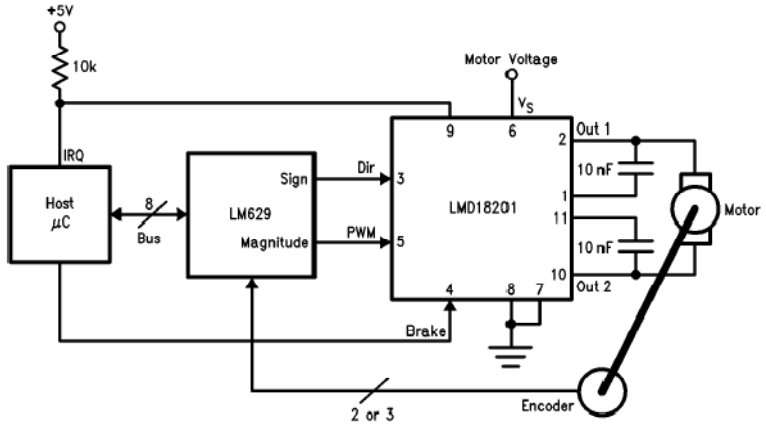
The LMD18201 can directly interface to any Sign/Magnitude PWM controller. The LM629 is a motion control processor that outputs a Sign/Magnitude PWM signal to coordinate either positional or velocity control of DC motors. The LMD18201 provides fully protected motor driver stage.

CURRENT SENSING

In many motor control applications it is desirable to sense and control the current through the motor. For these types of applications a companion product, the LMD18200, is also available. The LMD18200 is identical to the LMD18201 but has current sensing transistors that output a current directly proportional to the current conducted by the two upper DMOS power devices to a separate current sense pin. This technique does not require a low valued, power sense resistor and does not subtract from the available voltage drive to the motor.

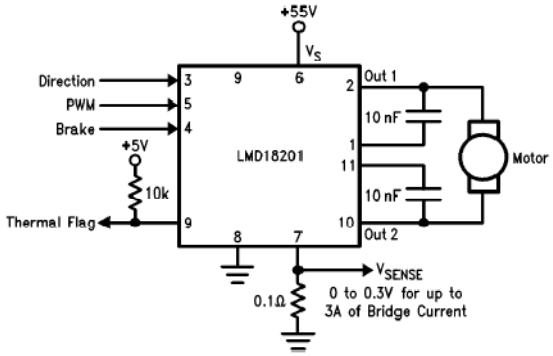
To sense the bridge current through the LMD18201 requires the addition of a small sense resistor between the power ground/sense pin (Pin 7) and the actual circuit ground. This resistor should have a value of 0.1Ω or less to stay within the allowable voltage compliance of the sense pin, particularly at higher operating current levels. The voltage between power ground/sense (Pin 7) and the signal ground (Pin 8) must stay within the range of −1V to +0.5V. Internally there is approximately 25Ω between pins 7 and 8 and this resistance will slightly reduce the value of the external sense resistor. Approximately 70% of the quiescent supply current (10 mA) flows out of pin 7. This will cause a slight offset to the voltage across the sense resistor when the bridge is not conducting. During reverse recovery of the internal protection diodes the voltage compliance between pins 7 and 8 may be exceeded. The duration of these spikes however are only approximately 100 ns and do not have enough time or energy to disrupt the operation of the LMD18201.

Basic Motor Driver



TL/H/10793–10

Current Sensing



TL/H/10793–11

5.7. Datasheet para el sensor LM35

National Semiconductor

November 2000

LM35
Precision Centigrade Temperature Sensors

General Description

The LM35 series are precision integrated-circuit temperature sensors, whose output voltage is linearly proportional to the Celsius (Centigrade) temperature. The LM35 thus has an advantage over linear temperature sensors calibrated in ˚ Kelvin, as the user is not required to subtract a large constant voltage from its output to obtain convenient Centigrade scaling. The LM35 does not require any external calibration or trimming to provide typical accuracies of ±¼˚C at room temperature and ±¾˚C over a full −55 to +150˚C temperature range. Low cost is assured by trimming and calibration at the wafer level. The LM35's low output impedance, linear output, and precise inherent calibration make interfacing to readout or control circuitry especially easy. It can be used with single power supplies, or with plus and minus supplies. As it draws only 60 µA from its supply, it has very low self-heating, less than 0.1˚C in still air. The LM35 is rated to operate over a −55˚ to +150˚C temperature range, while the LM35C is rated for a −40˚ to +110˚C range (−10˚ with improved accuracy). The LM35 series is available packaged in hermetic TO-46 transistor packages, while the LM35C, LM35CA, and LM35D are also available in the plastic TO-92 transistor package. The LM35D is also available in an 8-lead surface mount small outline package and a plastic TO-220 package.

Features

- Calibrated directly in ˚ Celsius (Centigrade)
- Linear + 10.0 mV/˚C scale factor
- 0.5˚C accuracy guaranteeable (at +25˚C)
- Rated for full −55˚ to +150˚C range
- Suitable for remote applications
- Low cost due to wafer-level trimming
- Operates from 4 to 30 volts
- Less than 60 µA current drain
- Low self-heating, 0.08˚C in still air
- Nonlinearity only ±¼˚C typical
- Low impedance output, 0.1 Ω for 1 mA load

Typical Applications

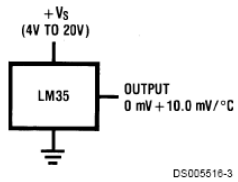
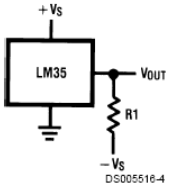


FIGURE 1. Basic Centigrade Temperature Sensor (+2˚C to +150˚C)

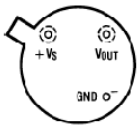


Choose $R_1 = -V_S/50\ \mu A$
V_{OUT} = +1,500 mV at +150˚C
= +250 mV at +25˚C
= −550 mV at −55˚C

FIGURE 2. Full-Range Centigrade Temperature Sensor

Connection Diagrams

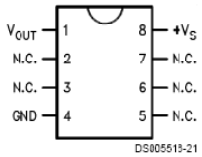
TO-46
Metal Can Package*



*Case is connected to negative pin (GND)

Order Number LM35H, LM35AH, LM35CH, LM35CAH or LM35DH
See NS Package Number H03H

SO-8
Small Outline Molded Package



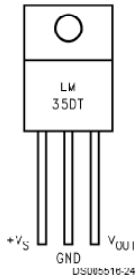
N.C. = No Connection

Top View
Order Number LM35DM
See NS Package Number M08A

TO-92
Plastic Package

Order Number LM35CZ, LM35CAZ or LM35DZ
See NS Package Number Z03A

TO-220
Plastic Package*



*Tab is connected to the negative pin (GND).
Note: The LM35DT pinout is different than the discontinued LM35DP.

Order Number LM35DT
See NS Package Number TA03F

Absolute Maximum Ratings (Note 10)

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/ Distributors for availability and specifications.

Supply Voltage	+35V to −0.2V
Output Voltage	+6V to −1.0V
Output Current	10 mA
Storage Temp.;	
TO-46 Package,	−60˚C to +180˚C
TO-92 Package,	−60˚C to +150˚C
SO-8 Package,	−65˚C to +150˚C
TO-220 Package,	−65˚C to +150˚C
Lead Temp.:	
TO-46 Package, (Soldering, 10 seconds)	300˚C
TO-92 and TO-220 Package, (Soldering, 10 seconds)	260˚C
SO Package (Note 12)	
Vapor Phase (60 seconds)	215˚C
Infrared (15 seconds)	220˚C
ESD Susceptibility (Note 11)	2500V
Specified Operating Temperature Range: T_{MIN} to T_{MAX} (Note 2)	
LM35, LM35A	−55˚C to +150˚C
LM35C, LM35CA	−40˚C to +110˚C
LM35D	0˚C to +100˚C

Electrical Characteristics

(Notes 1, 6)

Parameter	Conditions	LM35A			LM35CA			Units (Max.)
		Typical	Tested Limit (Note 4)	Design Limit (Note 5)	Typical	Tested Limit (Note 4)	Design Limit (Note 5)	
Accuracy (Note 7)	T_A=+25˚C	±0.2	±0.5		±0.2	±0.5		˚C
	T_A=−10˚C	±0.3			±0.3		±1.0	˚C
	T_A=T_{MAX}	±0.4	±1.0		±0.4	±1.0		˚C
	T_A=T_{MIN}	±0.4	±1.0		±0.4		±1.5	˚C
Nonlinearity (Note 8)	T_{MIN}≤T_A≤T_{MAX}	**±0.18**		**±0.35**	**±0.15**		**±0.3**	˚C
Sensor Gain (Average Slope)	T_{MIN}≤T_A≤T_{MAX}	**+10.0**	**+9.9, +10.1**		**+10.0**		**+9.9, +10.1**	mV/˚C
Load Regulation (Note 3) 0≤I_L≤1 mA	T_A=+25˚C	±0.4	±1.0		±0.4	±1.0		mV/mA
	T_{MIN}≤T_A≤T_{MAX}	**±0.5**		**±3.0**	**±0.5**		**±3.0**	mV/mA
Line Regulation (Note 3)	T_A=+25˚C	±0.01	±0.05		±0.01	±0.05		mV/V
	4V≤V_S≤30V	**±0.02**		**±0.1**	**±0.02**		**±0.1**	mV/V
Quiescent Current (Note 9)	V_S=+5V, +25˚C	56	67		56	67		µA
	V_S=+5V	**105**		**131**	**91**		**114**	µA
	V_S=+30V, +25˚C	56.2	68		56.2	68		µA
	V_S=+30V	**105.5**		**133**	**91.5**		**116**	µA
Change of Quiescent Current (Note 3)	4V≤V_S≤30V, +25˚C	0.2	1.0		0.2	1.0		µA
	4V≤V_S≤30V	**0.5**		**2.0**	**0.5**		**2.0**	µA
Temperature Coefficient of Quiescent Current		**+0.39**		**+0.5**	**+0.39**		**+0.5**	µA/˚C
Minimum Temperature for Rated Accuracy	In circuit of *Figure 1*, I_L=0	+1.5		+2.0	+1.5		+2.0	˚C
Long Term Stability	T_J=T_{MAX}, for 1000 hours	±0.08			±0.08			˚C