# AN667

# Smart Battery Charger with SMBus Interface

*Authors:* *Laszlo Kiraly, Linear Technology*
*Robert Schreiber, Microchip Technology*

## INTRODUCTION

This application note provides the schematics, software listings, and circuit board layout for a PIC16C73 based Smart Battery Charger. The Demo Board, DC101, is available to selected customers through Linear Technology Corp. product marketing.

The DC101 (Figure 1) is the Smart Battery Charger (SBC) portion of a Smart Battery System. A simplified block diagram of a Smart Battery System is shown in Figure 3 (refer to "The Smart Battery Charger Specification," Duracell).

The Smart Battery (SB) contains circuitry which provides charging information to the SBC. The SBC receives this information in terms of ChargingVoltage(), ChargingCurrent(), and AlarmWarning() commands from the SB. The SBC and SB communicate via the System Management Bus (SMB), which is an implementation of the $I^2C$ bus (refer to "The Smart Management Bus Specification," Intel; and "The $I^2C$ Bus and How to Use It," Philips Semiconductor). The SBC sets the charging current and voltage based on input from the SB. The SBC charger also has the intelligence to monitor the SB's thermistor. The thermistor provides temperature information for charge termination, and battery chemistry information.

**FIGURE 1: DC101 SMART BATTERY CHARGER**

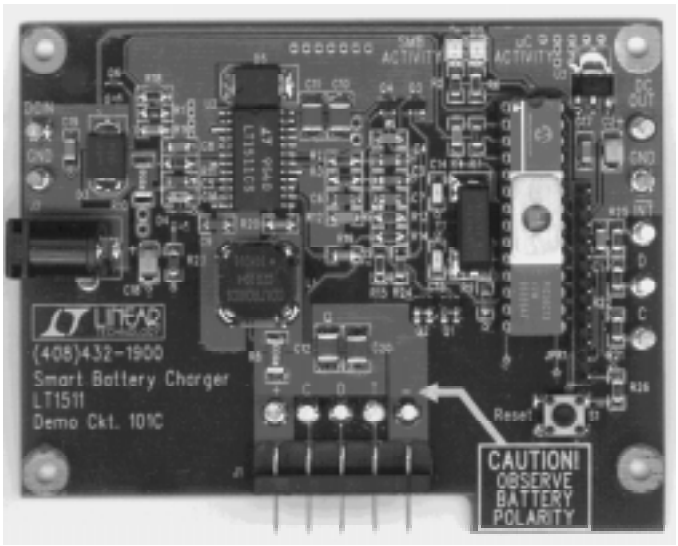



**FIGURE 2: SIMPLIFIED BLOCK DIAGRAM**

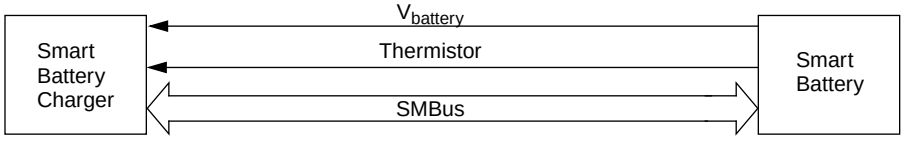
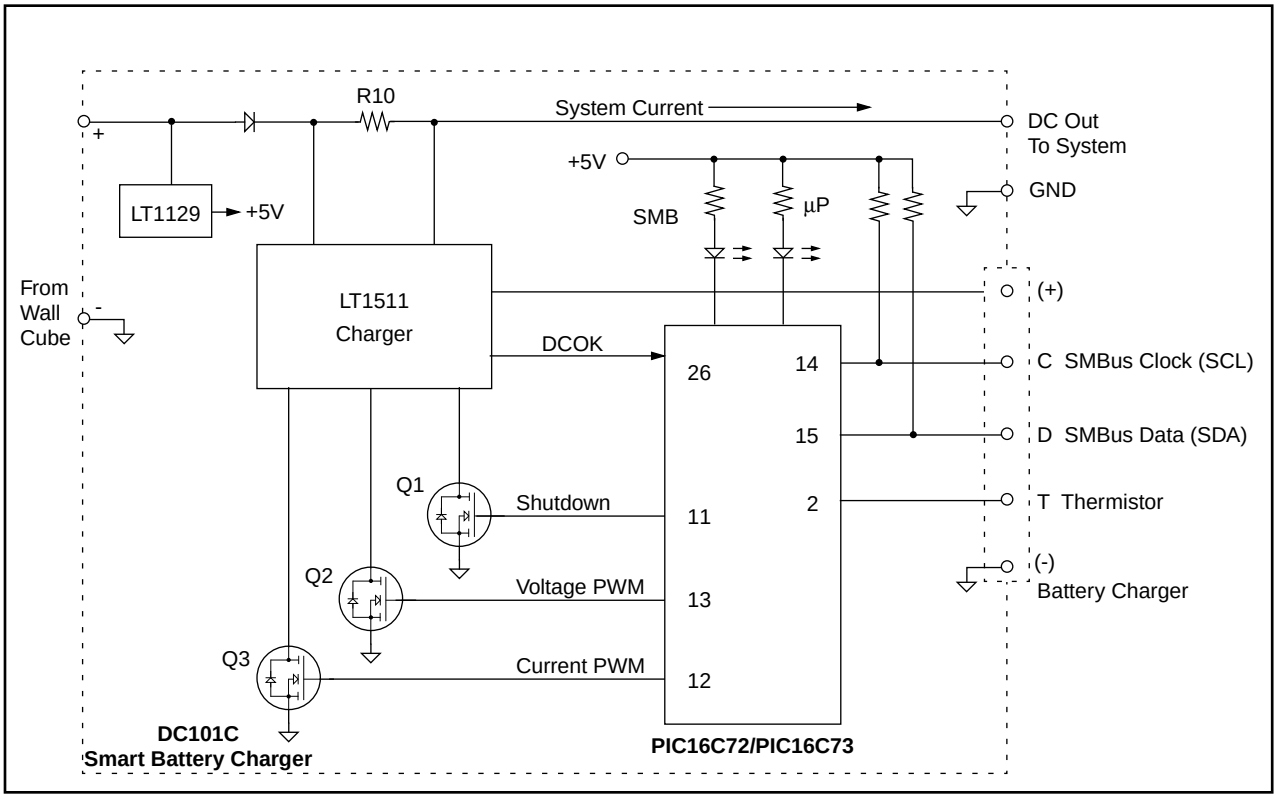


**FIGURE 3: BLOCK DIAGRAM**




## OPERATING THE BOARD

**Input voltage.** The nominal input voltage of the board is 24V DC (Refer to Appendix A for Performance Summary). The input voltage must be higher than the battery voltage by a minimum of 3V. The minimum input voltage is 16V, limited by the undervoltage lockout circuit in the LT1511 and set by resistors R16, R17 and R18. The highest input voltage is 27.4V, that is limited by the maximum input voltage of the LT1511. The input is protected against reverse polarity up to 30V.

**Input current.** The sum of the system current and the charger input current is limited by the LT1511 to 2A. When both the system current and the charger input current requirements are high, the charger reduces the charging current to meet the 2A current limit.

**RED LED. It** indicates SMBus activity. It lights-up for about 1 second when the charger recognizes its own address (12hex) on the SMBus.

**Green LED.** Flashing green LED indicates microprocessor activity and charger status.

Fast blinking (approximately 8 Hz) indicates normal microprocessor activity and either trickle charge or shut down charger status. After valid voltage and current data have been received the blinking speed of the LED slows down to about 2 Hz, indicating normal charging.

**Battery removal, thermistor measurements.** The charger periodically checks the thermistor in the battery through the T terminal. When the thermistor is out of normal operating temperature range, the charger switches to trickle charge mode and increases the flashing frequency of the green LED to indicate an abnormal charging condition. When the resistance of the thermistor is in the 500Ω < Rth < 1.5 kΩ range, the charger assumes Li Ion battery is at the output, and instead of trickle charging the battery it shuts-down the charger until a valid voltage and current request arrives from the battery.

An open thermistor forces the charger into trickle charge mode and the charger disregards data on the SMBus.

The schematics, parts list, and circuit layout are shown in Appendix A.

## SYSTEM MANAGEMENT BUS (SMBUS)

When charge in the Smart Battery (SB) drops below 85% of the nominal capacity, it initiates communication over the SMBus every 64 seconds. After sending a START sequence the battery addresses the Smart Battery charger and waits for acknowledgment (ACK) from it. If the charger fails to acknowledge the word, the battery terminates further communication by placing a STOP sequence onto the SMBus. If the charger acknowledges (ACK) the reception of first word, the battery continues the communication sequence and it sends six more words to the charger. The complete current and voltage request communication sequence is shown below:

```
START
address (12 hex)
ChargingCurrent() command code (14 hex),
current_LSB
current_MSB,
address (12 hex)
ChargingVoltage() command code (15 hex)
voltage_LSB
voltage_MSB
STOP
```

The idealized SMBus waveforms shown in Figure 4 illustrate SMBus communication between the battery and the charger. The first seven bits after the start sequence is the battery address. The R/$\overline{W}$ tells the charger that the battery attempts to write to the charger.

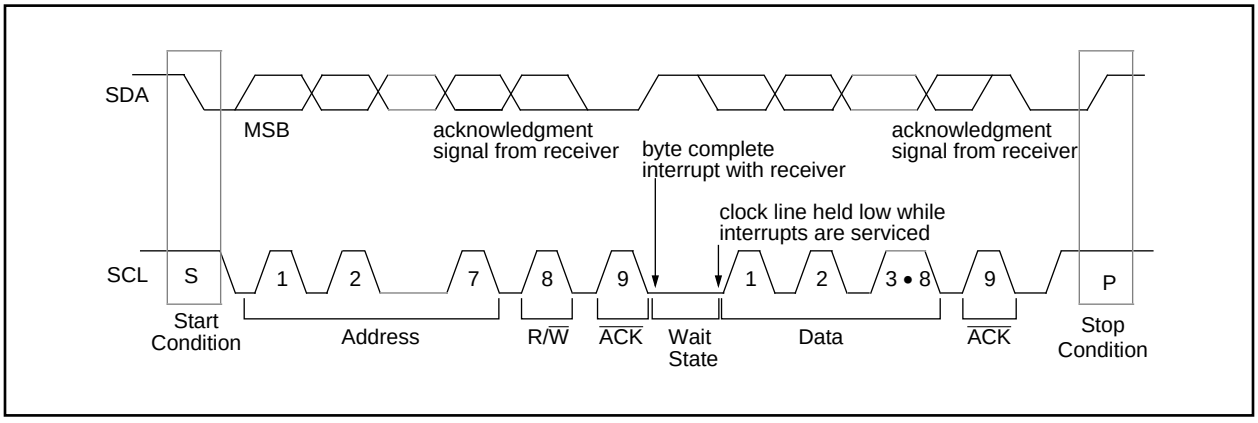
During the acknowledge period (ACK), the charger becomes active and it pulls the data line (SDA) low indicating reception of a data word. When reception of the data word is not acknowledged by the charger, the battery terminates the communication by sending a STOP sequence to the bus. The clock pulses for the communication are always generated by the battery (BUS MASTER).

## CONCLUSION

The PIC16C73 contains the on-board peripherals necessary to easily implement an SMBus battery charger. The $I^2C$ module allows for the SMBus communications and the PWM modules allow for voltage and current control. This high level of integration reduces the external components required and increases the flexibility of the design.

A complete software listing is shown in Appendix B.

**FIGURE 4: SMBus WAVEFORMS**



## APPENDIX A:

**TABLE A-1: PERFORMANCE SUMMARY**

| Parameter | Conditions | MIN | TYP | MAX | UNIT |
|---|---|---|---|---|---|
| Input Voltage | | 16.0 | 24.0 | 27.0 | V |
| Input Current | Hardware Limited | | | 2.0 | A |
| Output Current | Software Limited | 0 | | 2.5 | A |
| Output Voltage | Software Limited | 4 | | 20 | V |
| Thermistor Resistance | NiMH Battery | 3k | | 30k | Ohms |
| | Li Ion | 500 | 1000 | 1500 | Ohms |

**FIGURE A-1: SCHEMATIC DIAGRAM**

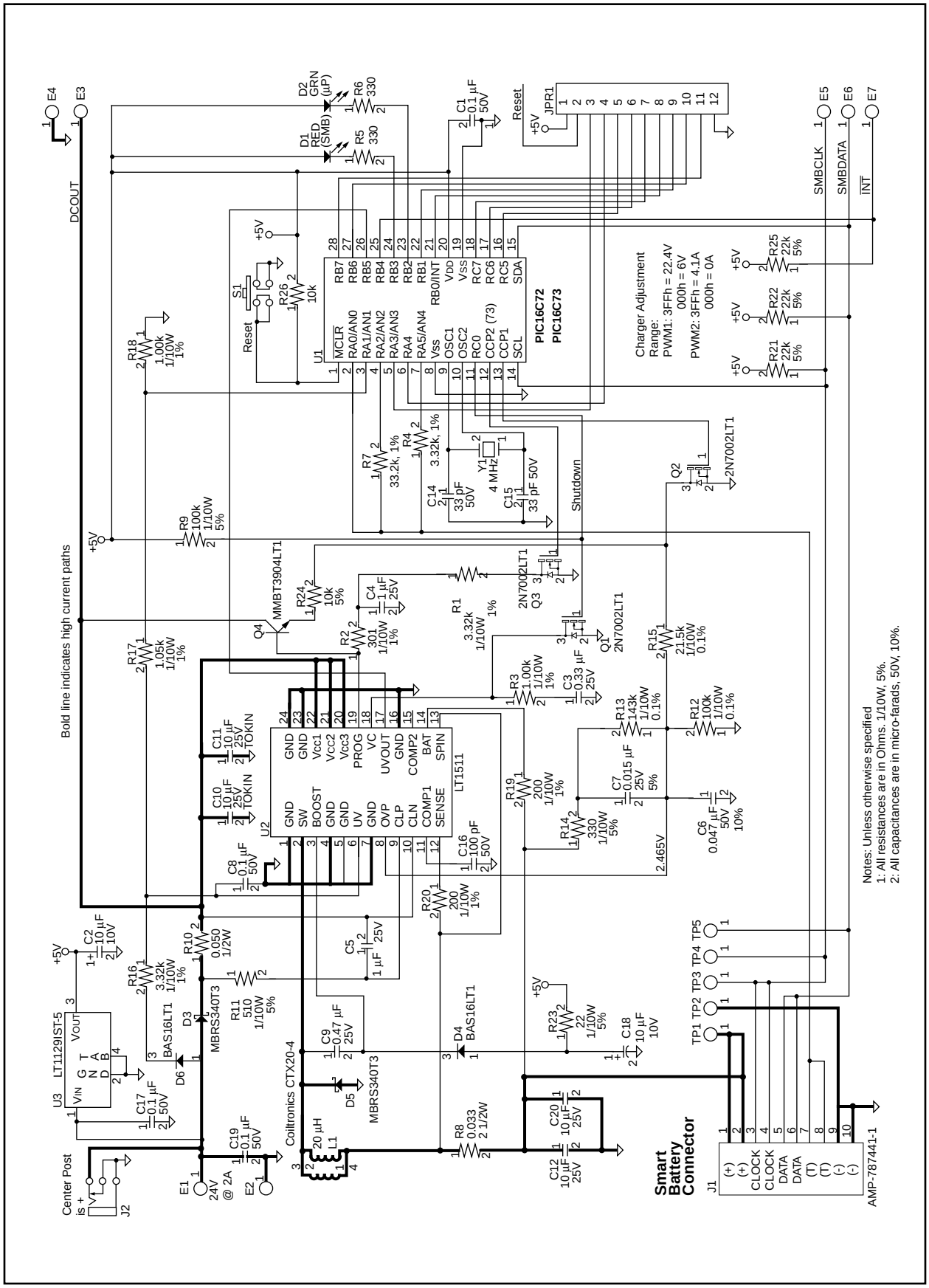


**TABLE A-2: PARTS LIST**

| Item | Quantity | Reference | Part Description | |
|---|---|---|---|---|
| 1 | 4 | C1,C8,C17,C19 | CAP., CHIP 0.1 µF 50V | AVX 12065C104KATMA |
| 2 | 2 | C2,C18 | CAP., TANT. 10 µF 10V | AVX TAJB106M010 |
| 3 | 1 | C3 | CAP., CHIP 0.33 µF 25V | AVX 12063G334ZAT2 |
| 4 | 2 | C4,C5 | CAP., CHIP 1 µF 25V | AVX 12063G105ZATMA |
| 5 | 1 | C6 | CAP., CHIP 0.047 µF 50V 10% | AVX 12065C473KAT |
| 6 | 1 | C7 | CAP., CHIP 0.015 µF 25V 5% | AVX 12063C153KAT2 |
| 7 | 1 | C9 | CAP., CHIP 0.47 µF 25V | AVX 12063G474ZAT2 |
| 8 | 4 | C10,C11,C12,C20 | CAP., CERAMIC 10 µF 25V | MARCON THC550EIE106Z |
| 10 | 2 | C14,C15 | CAP., CHIP 33 pF 50V | AVX 1206A330KAT2 |
| 11 | 1 | C16 | CAP., CHIP 100 pF 50V | VITRAMON VJ1206A101KXA |
| 12 | 1 | D1 | LED, SF1-BR RED | DATA DISPLAY PRODUCT |
| 13 | 1 | D2 | LED, SF1-G GRN | DATA DISPLAY PRODUCT |
| 14 | 2 | D3,D5 | DIODE | MOTOROLA MBRS340T3 |
| 15 | 2 | D4,D6 | DIODE | MOTOROLA BAS16LT1 |
| 16 | 12 | TP1--TP5, E1--E7 | TESTPOINT, TURRET | KEYSTONE 1502-2 |
| 17 | 1 | JPR1 | JUMPER, HEADER 12 PIN | COMM CON CONN. 3801S-12-G1 |
| 18 | 1 | J1 | CONNECTOR, | AMP-787441-1 |
| 19 | 1 | J2 | CONNECTOR, | CUI-PJ-002A CUI-STACK |
| 20 | 1 | L1 | INDUCTOR, 20 µH | COILTRONICS CTX20-4 |
| 21 | 3 | Q1,Q2,Q3 | MOSFET N, Channel | MOTOROLA 2N7002LT |
| 22 | 1 | Q4 | TRANS. NPN, | MOTOROLA MMBT3904LT1 |
| 23 | 1 | R2 | RES., 301, 1/8W, 1% | AVX CR32-3010F-T |
| 24 | 2 | R3,R18 | RES., 1.00k, 1/8W, 1% | AVX CR32-1001F-T |
| 25 | 3 | R1,R4,R16 | RES., 3.32k, 1/8W, 1% | AVX CR32-3321F-T |
| 26 | 3 | R5,R6,R14 | RES., 330, 1/8W, 5% | AVX CR32-331J-T |
| 27 | 1 | R7 | RES., 33.2k, 1/8W, 1% | BECKMAN BCR1/8-3322F-T |
| 28 | 1 | R8 | RES., 0.033, 1/2W, 5% | IRC LR2010-01-R033-J |
| 29 | 1 | R9 | RES., 100k, 1/8W, 5% | AVX CR32-104J-T |
| 30 | 1 | R10 | RES., 0.050, 1/2W, 5% | IRC LR2010-01-R050-J |
| 31 | 1 | R11 | RES., 510, 1/8W, 5% | AVX CR32-511J-T |
| 32 | 1 | R12 | RES., 100k, 1/8W, 0.1% | IRC W1206R-03-1003-B |
| 33 | 1 | R13 | RES., 143k, 1/10W, 0.1% | IRC W1206R-03-1433-B |
| 34 | 1 | R15 | RES., 21.5k, 1/10W, 0.1% | IRC W1206R-03-2152-B |
| 35 | 1 | R17 | RES., 1.05k, 1/8W, 1% | AVX CR32-1051F-T |
| 36 | 2 | R20,R19 | RES., 200, 1/8W, 1% | AVX CR32-2000F-T |
| 37 | 3 | R21,R22,R25 | RES., 22k, 1/10W, 5% | AVX CR-32-223J-T |
| 38 | 1 | R23 | RES., 22, 1/10W, 5% | AVX CR32-220J-T |
| 39 | 2 | R24,R26 | RES., 10k, 1/8W, 5% | DALE CR1206-103J |
| 40 | 1 | S1 | PB-SWITCH, MJTP1230 | MORS-ASC MJTP1230 |
| 41 | 1 | U1 | I.C., PIC16C73, PIC16C72 | MICROCHIP IC, Microcontroller |
| 42 | 1 | U2 | I.C., LT1511 | LINEAR TECHNOLOGY IC, Battery Charger |
| 43 | 1 | U3 | I.C., LT1129IST-5 | LINEAR TECHNOLOGY IC, Voltage Regulator |
| 44 | 1 | Y1 | CRYSTAL, 4 MHz | EPSON(USA) MA-505-4.00M-C2 |
| 45 | 1 | XU1 | I.C., SOCKETS | COMM CON 7167-14-G2 |

**FIGURE A-2: CIRCUIT COMPONENT SIDE**

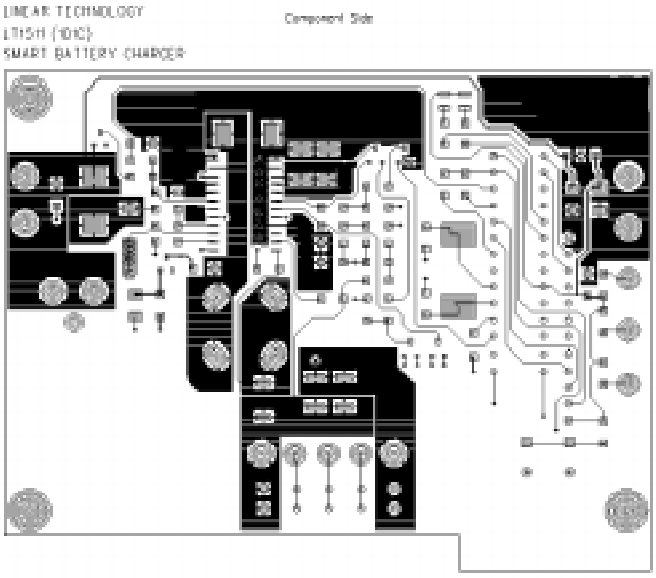



**FIGURE A-3: COMPONENT SIDE SILKSCREEN**

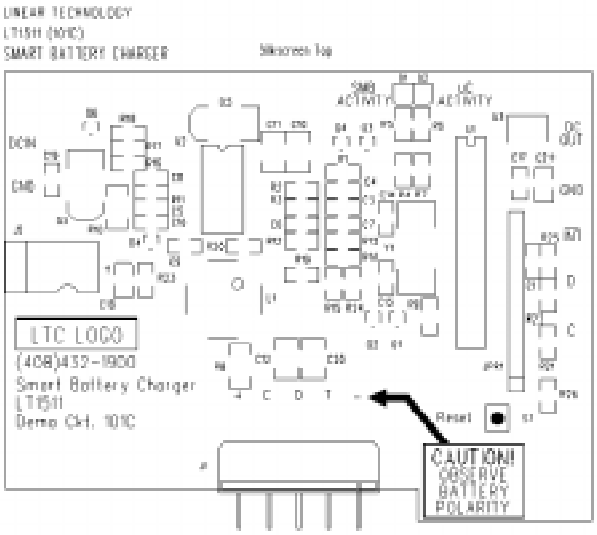


**FIGURE A-4: COMPONENT SIDE SOLDERSIDE**

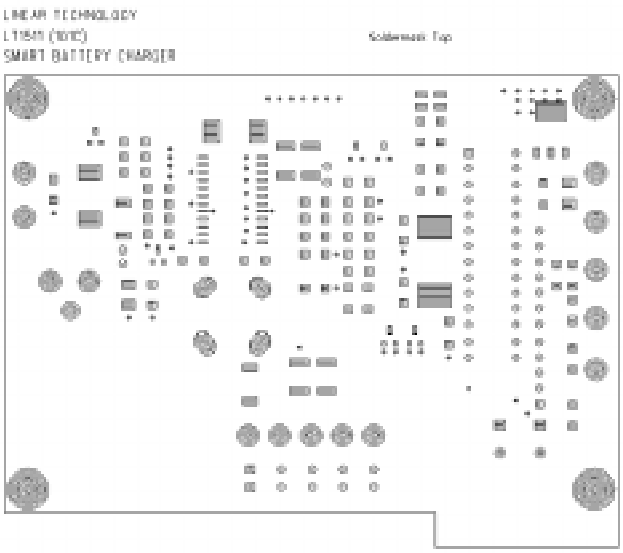



**FIGURE A-5: CIRCUIT SOLDER SIDE**

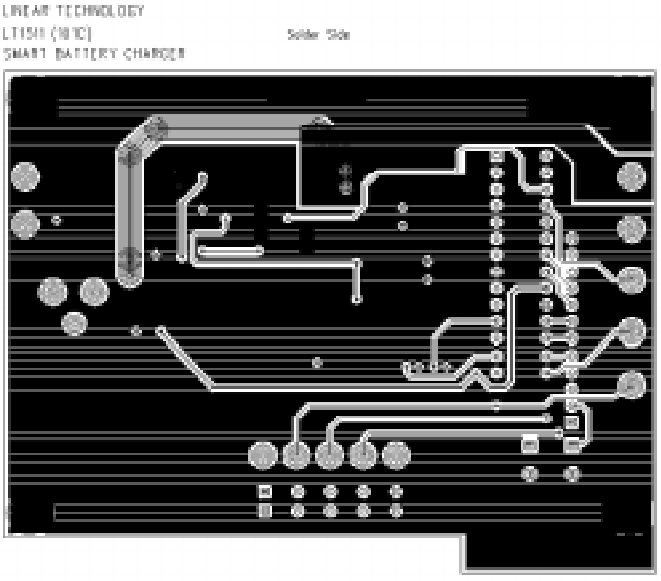
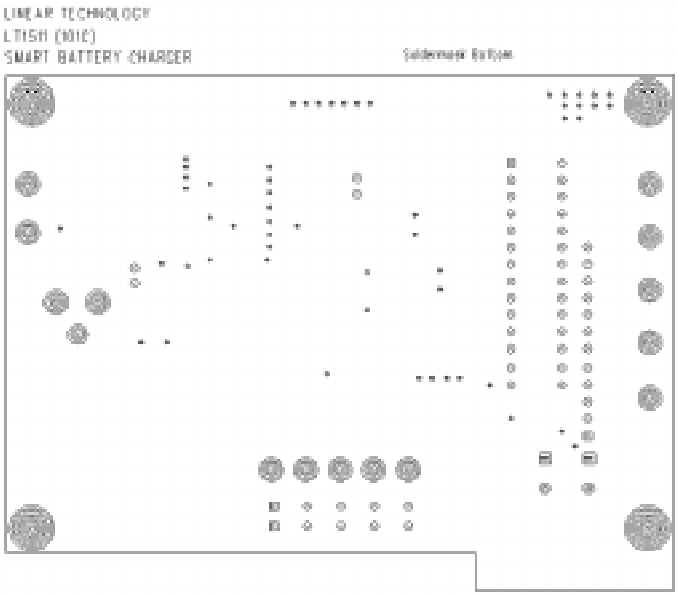
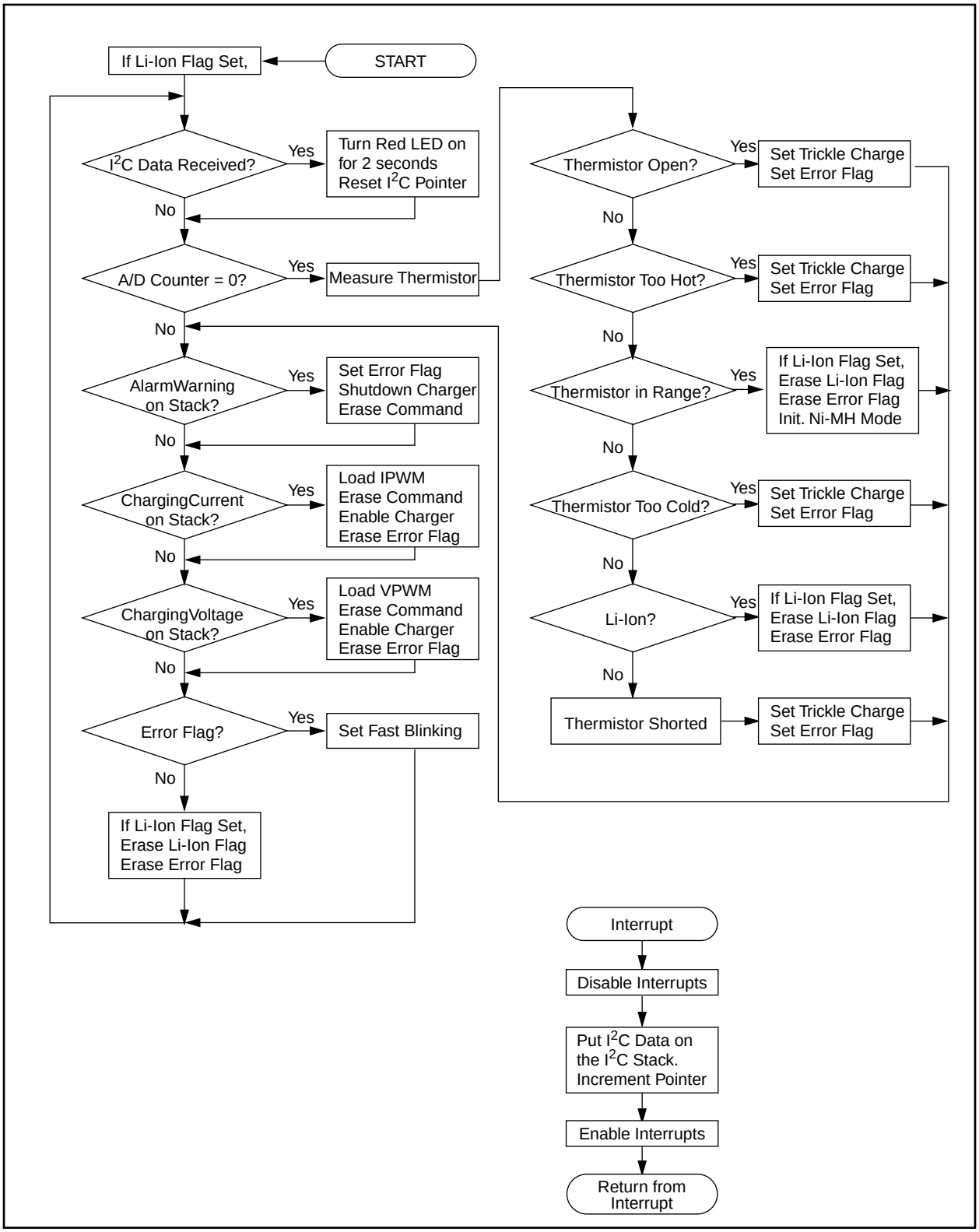


**FIGURE A-6: SOLDER SIDE SOLDER MASK**



## APPENDIX B:

**FIGURE B-1: SOFTWARE FLOWCHART**



```
//-------------------------------------------------------------------------
//                        CHRGR101.C                       Version: 1.01
//                      By: Laszlo Kiraly
//                    Email:KiralyL@aol.com
//                     Linear Technology
//                   Applications Department
//          1630 McCarthy Blvd, Milpitas, CA 95035, USA
//          Phone: (408) 432-1900,  Fax: (408) 434-0507
/*-------------------------------------------------------------------------
                                                  April 16,1996
This program:
 Sets-up PWMs and I2C communication.
 It receives data from I2C (including its own address) and stores them
 in i2c_data[10] as they come in, including its own address (0x12).

              0 charger i2c address (always 0x12)
              1 ChargingCurrent() CMD (0x14)
              2 charging current data byte L
              3 charging current data byte H       (unsigned int, mA)
              4 charger i2c address (0x12)
              5 ChargingVoltage CMD (0x15)
              6 charging voltage data byte L
              7 charging voltage data byte H       (unsigned int, mV)

Revision changes:

- scales current word and loads value to IPWM (PWM2).
- scales voltage word and loads value to VPWM (PWM1).
- 16CC73.H bits for CCP1CON registers need to be specified.
- thermistor limits were changed   (2/21/96)
- using easy math (2/21/96)
//
- comments were added                                          (4/11/96)
- init() function was created                                  (4/11/96)
- AlarmWarning() function now checks b15:b12 bits.             (4/11/96)
* At LT1511 UV shutdown (DCOK-L) uP pulls its shut-down high.
- The uP enables the charger after valid data received from battery.(4/11/96)
- No broadcast from battery timeout (180 sec. nom.) now implemented.(4/11/96)
- SMBus reset function has been added.                         (4/15/96)
- GIE disable and enable was removed (__int handles them)      (4/16/96)
-------------------------------------------------------------------------
*/
#include "16C73a.H"
#include "math.h"
//
void delay( void );                   // 840msec delay for red LED
void delay1( void);                   // 20ms delay for while() loop
void delay_5us( void );
void timer0( char );
void load_ipwm( void );               // scales received value and loads PWM1
void load_vpwm( void );               // scales ChargingCurrent, loads PWM2
void initiv( char );                  // sets voltage and current PWMs
void ad_th( void );                   // measures thermistor, controls charger
void init_var( void );                // initializes variables (general)
void clear_smbus( void );             // sends start-stop sequence to SMBus
//
char i2c_data[10];                    // i2C stack
char i2c_counter;                     // i2c stack pointer
//
bits  flag;
                                      // flag.0 is set on return from I2C INT.
                                      // flag.1 blinking speed 1 = high speed
                                      // flag.2 themistor/resistor out of range
                                      // flag.3=1 comm. timeout, inibits cntr
char LED_counter, ad_counter;
long unsigned ad_val;
```

```
long unsigned com_timeout_cntr;
long unsigned clear_smbus_cntr;
;
char val;
//
//---------------------------------------------------------------------
void __INT( void)
{
  SSPCON.CKP = 0;                       // HOLD CK 0
  flag.0 = 1;

  i2c_data[i2c_counter] = SSPBUF;
  i2c_counter++;
  if(i2c_counter > 7)
  {
     i2c_counter=0;                     // reset counter if overflows
  }
  PIR1.SSPIF = 0;
  SSPCON.CKP = 1;                       // release ck
}
//
// ------------------------------------------------------------
void main()
{
  init_var();                           // initializes variables (hardware setup)
  //
  // timer0(150);
  //
  delay();                              // wait 1sec until voltages settle
  //
  i2c_counter =0;                       // clear i2c data pointer
  ad_counter = 5;                       // clear
  //
  com_timeout_cntr = 0;
  clear_smbus_cntr = 0;                 // sets time between SMBus inits.
  //
  SSPADD = 0x12;                        // define slave address
  //
  flag = 0;
  flag.1 = 1;                           // set fast blinking
  flag.2 = 1;                           // thermistor out of both NiMH and liIon
  flag.3 = 1;                           // disable com_timeout_cntr
  //
  PIE1.SSPIE  = 1;                      // enable i2c interrupt
  INTCON.PEIE = 1;                      // enable peripheral INTS
  INTCON.GIE  = 1;                      // general INT enable
  //
  while(1)                              // infinite loop in MAIN
  {
    delay1();
    //
    // ------------------------------------------------------------
    if(com_timeout_cntr == 6250)        // no communication timeout 100-> 2.88sec
    {
       flag.1 = 1;                      // set fast blinking
       flag.3 = 1;                      // communication timeout cntr disabled
       //
       PORTC.0 = 1;                     // disable charger
       com_timeout_cntr = 0;            // reset timer
       //
       PORTB.3 = 0;                     // turn red LED on for 50ms
       delay1();
       PORTB.3 = 1;                     // red LED off
    }
    //
    if( !flag.3 ) com_timeout_cntr++;   // if no timeout run counter
```

```
else com_timeout_cntr = 0;
if( flag.2) flag.3 = 1;            // if th. out of range reset cntr
                                   // do not override th. based decisions

//
// ---------------------------------------------------------------------
if( flag.3) clear_smbus_cntr++;    // clear SMBus
if( clear_smbus_cntr == 1000)
{
  clear_smbus();
  clear_smbus_cntr = 0;
}
if( flag.2) clear_smbus_cntr = 0;
//
// ---------------------------------------------------------------------
if(ad_counter == 0)                // is it time to measure thermistor ?
{                                  // checks UV (DCOK input) also
  ad_th();                         // if ad_counter starts at 5, td=140ms
ad_counter = 5;
}
ad_counter--;
// ---------------------------------------------------------------------
if(flag.0)                         // retrun from I2C interrupt ?

{
    PORTB.3 = 0;                   // red LED on
    delay();                       // 1 second delay
    PORTB.3 = 1;                   // red LED off
    flag.0 = 0;
    i2c_counter = 0;               // reset I2C stack pointer
    // com_timeout_cntr = 0;
    flag.1 = 0;                    // change to slow blinking
}
// ---------------------------------------------------------------------
if(i2c_data[1] == 0x16)            // AlarmWarning() on stack ?
{
   if( i2c_data[3] > 0x0f)
   {
     PORTC.0 = 1;
     flag.1 = 1;                   // set fast blinking
     i2c_data [1] = 0xF6;          // erase command from stack
     PORTC.0 = 1;                  // shutdown = 1 (inhibit charger)
     com_timeout_cntr = 0;
     flag.3 = 0;                   // enable com_timeout_cntr
   }
}
// ---------------------------------------------------------------------
if(i2c_data[1] == 0x14)            // ChargingCurrent() on stack ?
{
  load_ipwm();                     // set PWM
  i2c_data[1] = 0xF4;              // erase command
  flag.1 = 0;                      // set slow blinking
  com_timeout_cntr = 0;
  flag.3 = 0;                      // enable com_timeout_cntr
}
//---------------------------------------------------------------------
if(i2c_data[5] == 0x15)            // ChargingVoltage() on stack ?
{
  load_vpwm();                     // set output voltage
  i2c_data[5] = 0xF5;              // erase command
  flag.1 = 0;                      // set slow blinking
  PORTC.0 = 0;                     // shutdown = 0 enable charger
  com_timeout_cntr = 0;
  flag.3 = 0;                      // enable com_timeout_cntr
}
// -------------- setting LED blinking speed --------------------------
```

```
    //
    if(flag.1)                          // fast blinking green requested ?
    {                                   // or shutdown
      LED_counter++;
      if(LED_counter > 1)               // is it time to change status of LED ?
      {
         LED_counter = 0;               // if yes, clear counter
         if(PORTB.2) PORTB.2=0;         // and toggle LED
         else PORTB.2 = 1;
      }
    }
    if(!flag.1)                         // slow flashing green - charging
    {
      LED_counter++;
      if(LED_counter > 4)               // time to change LED status ?
      {
         LED_counter = 0;               // if yes, reset counter
         if(PORTB.2) PORTB.2=0;         // and toggle LED
         else PORTB.2 = 1;
      }
    }
  } // end of while
}   // end of main
// -------------------------------------------------------------
//
void clear_smbus( void )
{
  TRISC.SDA = 1;                        // set as input
  TRISC.SCL = 1;                        // set as input
  //
  PORTC.SDA = 0;                        // to pull down SDA line when TRISC.SDA=L

  SSPCON.SSPEN = 0;                     // configure SDA and SCL pins as i/o pins
  //
     if( !PORTC.SCL) goto clsm1;        // SMBus traffic ? jump if yes
       delay_5us();
     if( !PORTC.SCL) goto clsm1;        // SMBus traffic ? jump if yes
       delay_5us();
     if( !PORTC.SCL) goto clsm1;        // SMBus traffic ? jump if yes
       //
  TRISC.SDA = 0;                        // SDA ->   ~~~\___    start
  delay_5us();
  TRISC.SDA = 1;                        // SDA ->   ___/~~~    stop
  //
  clsm1:
  PORTC.SDA = 1;
  TRISC.SDA = 1;                        // set as output
  TRISC.SCL = 1;                        // set as output
  SSPCON.SSPEN = 1;                     // cofigure SDA and SCL as serial port pins
}
//
//-------------------------------------------------------------
// initialize voltage and current PWMs.
//-------------------------------------------------------------
void initiv( char a )
{
  if( a==1)
  {
    i2c_data[1] = 0x14;                 // ChargingCurrent command
    i2c_data[2] = 0x64;                 // charging current L byte
    i2c_data[3] = 0x00;                 //                  H byte
    i2c_data[5] = 0x15;                 // ChargingVoltage command
    i2c_data[6] = 0xFF;                 // charging voltage L byte
    i2c_data[7] = 0xFF;                 //                  H byte
  }
  if( a==0)
```

```
  {
    i2c_data[1] = 0x14;               // Charging Current command
    i2c_data[2] = 0x32;               // charging current L byte
    i2c_data[3] = 0x00;               //                  H byte
    i2c_data[5] = 0x15;               // Charging Voltage command
    i2c_data[6] = 0xFF;               // charging voltage L byte
    i2c_data[7] = 0x8F;               //                  H byte
  }
  load_ipwm();
  load_vpwm();
  //
  i2c_data[1] = 0;
  i2c_data[5] = 0;
}
//-------------------------------------------------------------
// Measures thermistor and controls the charger accordingly.
// -------------------------------------------------------------
void ad_th( void)
{
  char *ptr;
  char i,j;

  PORTA.5 = 1;                         // use 3.32k pull-up
  ad_val = 0;

  for( j=0; j<4; j++)
  {
    PIR1.ADIF = 0;
    ADCON0.GO = 1;
    while(ADCON0.GO);

    ad_val = ad_val + ADRES;
  }
  PORTA.5 = 0;                         // turn 3.32k pull-up off
  ad_val = ad_val / 4;
  //
  ptr = &ad_val;
  val = *ptr;
  //
  if( val > 220)                       // -- thermistor too cold---------
  {
     flag.1 = 1;
     flag.2 = 1;
     initiv(1);
  }

  if( (val < 221) && (val > 121) )     // thermistor in-range -----------
  {
     if(flag.2)                        // returns from thermistor error
     {
        initiv(1);                     // set trickle current
        PORTC.0 = 0;                   // start charger
        flag.2 = 0;                    // clear thermistor error flag
     }
  }

  if( (val < 122) && (val > 34) )      // thermistor too hot -----------
  {
     flag.1 = 1;                       // LED fast blinking
     flag.2 = 1;                       // set thermistor  flag
     initiv(1);                        // set trickle charge
  }

  if( (val<35) && (val > 10) )         // Li ION
  {
    if(flag.2)
```

```
    {
       initiv(0);
       PORTC.0 = 0;                    // start charger
       flag.2 = 0;                     // clear error flag
    }
  }
  if( val< 11)                         // Thermistor shorted
  {
     flag.1 = 1;                       // green LED fast blinking
     flag.2 = 1;                       // set thermistor flag
     initiv(1);                        // set trickle charge
  }
}
//-----------------------------------------------------------------
// reads current values from i2c_dat[2] and i2c_data[3] locations,
// (L and H bytes) limits  the current, scales current word, and
// loadx PWM registers (10 bit mode)
//-----------------------------------------------------------------
void load_ipwm( void )
{
  long idata;
  char *ptr;
  bits ilowbits;
  //
  ptr = &idata;                       // get address of idata
  //
  *ptr = i2c_data[2];                 // load L byte of idata
  *(ptr+1) = i2c_data[3];             // load H byte of idata
  //
  if(idata > 2600) idata = 2600;
  //
  idata >>=2;                         // scale idata 4096mA / 1024 = 4
  ilowbits = *ptr;                    // save L byte
  //
  idata >>=2;                         // upper 8 bit of data
  //
  CCP2CON.CCP2X = ilowbits.1;         // load lower two LSBits of 10 bit word to PWM
  CCP2CON.CCP2Y = ilowbits.0;
  CCPR2L = *ptr;                      // load upper 8 bit of 10 bit PWM word
}
//
//-----------------------------------------------------------------------
// Reads voltage data from i2c_data[6] (L byte) and i2c_data[7] locations,
// assembles a 16 bit word. Limits the voltage at max 20V, scales vdata
// voltage data word and loads PWM with 10 bit data.
//-----------------------------------------------------------------------
void load_vpwm( void )
{
  long unsigned vdata;
  char *ptr;
  bits vlowbits;
  //
  ptr = &vdata;                       // get address of idata
  *ptr = i2c_data[6];                 // load L byte of idata
  *(ptr+1) = i2c_data[7];             // load H byte of idata
  //
  if(vdata < 8000) vdata = 8000;      // PWM1 0x000 =8V,  0x3FF = 20V
  if(vdata > 18000) vdata=18000;      // limit incoming voltage to 20V
  //
  vdata = vdata-6000;                 //
  //
  vdata >>= 4;                        // scale vdata 22-6=16V,  16V/1024=16
  vlowbits = *ptr;                    // save L byte
  vdata >>=2;                         // upper 8 bit of data
  //
  CCP1CON.CCP1X = vlowbits.1;         // load lower two LSBits of 10 bit word
```

```
  CCP1CON.CCP1Y = vlowbits.0;       // to PWM 9-th and 10-th bits.
  CCPR1L = *ptr;                    // load upper 8 bit of 10 bit PWM word
}
//---------------------------------------------------------------------
void delay(void)
{
  char i, j, k;                     // software delay, about 1 sec
  for(k=0; k<2; k++)                // measured: 840 msec.
  {
    for(j=0; j<255; j++)
    {
      for(i=0; i<255; i++);         // inner loop delay: 1.647ms
    }
  }
}
// ---------------------------------------------------------------------
void delay1( void )
{
  char i,j;
  for(j=0; j<20; j++)
  {
    for(i=0; i<155; i++);           // delay 1ms
  }
}
// ---------------------------------------------------------------------
void delay_5us( void )          // 5us delay
{
#asm
  nop
  nop
  nop
  nop
#endasm
}
//---------------------------------------------------------------------------
void timer0( char a )           // td=(256-a)*256us @ 4MHz (prescaler /256)
{
  TMR0 = a;                     // reload load timer
  INTCON.T0IF = 0;              // reset TMR0 interrupt flag
}
//---------------------------------------------------------------------------
//
void init_var( void )
{
//------INIT TMR0--------------
//
  OPTION = 0b11010111;// init timer 1, no pull-up at b (bit7=1)
                                // bit7 Port B pull-up enable (1=disable)
                                //      INTEDG 1=INT on rising of RBO/INT
                                //      T0CS TMR0 ck source 0 = internal
                                //      T0SE TMR0 source edge 1=H->L on RA4
                                // bit3 PSA prescaler assign. 1=WDT, 0=TMR0
                                //      PS2:PS0 prascaler div. rate 2,4,8,16..

//--- TMR2 and PWMs ----------
//
  T2CON = 0x04;// init timer, bit2 turns it on
                                // bit7 unimplemented
                                // bit6:bit3 postscaler select 1,2,3..16
                                // bit2 TMR2ON 1=TMR2 on 0=TMR2 off
                                // bit1:bit0 prescaler div. 1, 4 or 16
//
  CCP1CON = 0b00001100;// init PWM1
                                // bit7 unimplemented
                                //      unimplemented
                                //      bit1 for 10 bit mode (0 for 8bit)
```

```
                              //      bit0 for 10 bit mode (0 for 8bit)
                              // bit3:0 mode select 11xx= PWM mode
//
  CCP2CON = 0b00001100;// init PWM2

// --- init port A --
  PORTA = 0;
  //        76543210
  TRISA = 0b11011011;         // RA5  3.32k       OUT
                              // RA4  header 4    IN
                              // RA3  header 3    IN
                              // RA2  33.2k       OUT
                              // RA1  UV          IN
                              // RA0  analog in   IN
// ----init A/D------
  //         76543210
  ADCON0 = 0b01000001;        // Analog digital converter module
                              // bit7 ADCS1   A/D clock select
                              //      ADCS0   01= fosc/8 -> tconv=16us
                              //      CHS2    channel selection
                              //      CHS1
                              // bit3 CHS0    000 -> RA0
                              //      GO/DONE_ start conv/finished
                              //      unimplemented
                              //      ADON 1= a/d on 0=a/d off
  //         76543210
  ADCON1 = 0b00000100;        // b7:b3 not implemented
                              // b2:b0 analog port pin config.
                              // 100   RA0=analog  RA1=analog
                              //       RA2=digital RA3=digital, Vref=VDD

// ----init port B---
//
  PORTB  = 12;
  //        76543210
  TRISB = 0b11110011;         // RB7 (pin 28) -> header 11    IN
                              // RB6 (pin 27) -> header 10    IN
                              // RB5 (pin 26) DCOK input      IN
                              // RB4 (pin 25) -> header 5     IN
                              // RB3 red  LED                 OUT
                              // RB2 grn. LED                 OUT
                              // RB1 (pin 22) -> header 9     IN
                              // RBO (pin 21) -> header 8     IN
// ----init.port C---

  PORTC = 1;
  //        76543210
  TRISC = 0b11011000;// init PORTC  C.2 = CCP1 C.1 = CCP2
                              // RC7 (pin18) header 7
                              // RC6 (pin17) header 6
                              // RC5 (pin16) header 5
                              // RC4 (pin15) SDA  I2C data  -> input
                              // RC3 (pin14) SCL  I2C clock -> input
                              // RC2 (pin13) CCP1 PWM1 pin, -> output
                              // RC1 (pin12) CCP2 PWM2 pin, -> output
                              // RC0 (pin11) shutdown       -> output

  // -------------------------
  CCPR1L = 10;                // pulse width1
  CCPR2L = 10;                // pulse width2
  PR2 = 255;                  // period time 200-> 200us

  //---setting-up I2C communication ----------
  //         76543210
  SSPCON = 0b00110110;        // sync serial port control register
                              // B7 WCOL=0    Write collision det. (SW reset)
```

```
                                //    SSPOV=0   receive collision det (SW reset)
                                //    SSPEN=1   enable ser port ( SCK SDO open D)
                                //    CKP = 1   0=enable clock
                                // B2:B0 SSPM2:SSPM0=110 slave mode.
// --- timer interrupts---
  INTCON.T0IF = 1;              // reset TMR0 int flag
  INTCON.T0IE = 0;              // 1=enable TMR0 interrupt

  PIE1.SSPIE = 1;               // enable i2c interrupt
  PIR1.SSPIF = 0;               // reset i2c interrupt flag

//  T1CON = 0x00;               // init TMR1, internal ck, prescaler div=1

  i2c_data[0] = 0;              // clear address locations
  i2c_data[1] = 0;
  i2c_data[2] = 0;
  i2c_data[3] = 0;
  i2c_data[4] = 0;
  i2c_data[5] = 0;
  i2c_data[6] = 0;
  i2c_data[7] = 0;
  i2c_data[8] = 0;
  i2c_data[9] = 0;
}
```

**Note the following details of the code protection feature on PICmicro® MCUs.**

- The PICmicro family meets the specifications contained in the Microchip Data Sheet.
- Microchip believes that its family of PICmicro microcontrollers is one of the most secure products of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the PICmicro microcontroller in a manner outside the operating specifications contained in the data sheet. The person doing so may be engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as “unbreakable”.
- Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our product.

If you have any further questions about this matter, please contact the local sales office nearest to you.



DNV Certification, Inc. USA — ANSI-RAB QMS ACCREDITED — DNV MSC The Netherlands Accredited by the RvA — DNV — ISO 9001 / QS-9000 REGISTERED FIRM

*Microchip received QS-9000 quality system certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona in July 1999. The Company’s quality system processes and procedures are QS-9000 compliant for its PICmicro® 8-bit MCUs, KEELOQ® code hopping devices, Serial EEPROMs and microperipheral products. In addition, Microchip’s quality system for the design and manufacture of development systems is ISO 9001 certified.*

# WORLDWIDE SALES AND SERVICE

## AMERICAS

**Corporate Office**
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200 Fax: 480-792-7277
Technical Support: 480-792-7627
Web Address: http://www.microchip.com

**Rocky Mountain**
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7966 Fax: 480-792-7456

**Atlanta**
500 Sugar Mill Road, Suite 200B
Atlanta, GA 30350
Tel: 770-640-0034 Fax: 770-640-0307

**Boston**
2 Lan Drive, Suite 120
Westford, MA 01886
Tel: 978-692-3848 Fax: 978-692-3821

**Chicago**
333 Pierce Road, Suite 180
Itasca, IL 60143
Tel: 630-285-0071 Fax: 630-285-0075

**Dallas**
4570 Westgrove Drive, Suite 160
Addison, TX 75001
Tel: 972-818-7423 Fax: 972-818-2924

**Detroit**
Tri-Atria Office Building
32255 Northwestern Highway, Suite 190
Farmington Hills, MI 48334
Tel: 248-538-2250 Fax: 248-538-2260

**Kokomo**
2767 S. Albright Road
Kokomo, Indiana 46902
Tel: 765-864-8360 Fax: 765-864-8387

**Los Angeles**
18201 Von Karman, Suite 1090
Irvine, CA 92612
Tel: 949-263-1888 Fax: 949-263-1338

**New York**
150 Motor Parkway, Suite 202
Hauppauge, NY 11788
Tel: 631-273-5305 Fax: 631-273-5335

**San Jose**
Microchip Technology Inc.
2107 North First Street, Suite 590
San Jose, CA 95131
Tel: 408-436-7950 Fax: 408-436-7955

**Toronto**
6285 Northam Drive, Suite 108
Mississauga, Ontario L4V 1X5, Canada
Tel: 905-673-0699 Fax: 905-673-6509

## ASIA/PACIFIC

**Australia**
Microchip Technology Australia Pty Ltd
Suite 22, 41 Rawson Street
Epping 2121, NSW
Australia
Tel: 61-2-9868-6733 Fax: 61-2-9868-6755

**China - Beijing**
Microchip Technology Consulting (Shanghai) Co., Ltd., Beijing Liaison Office
Unit 915
Bei Hai Wan Tai Bldg.
No. 6 Chaoyangmen Beidajie
Beijing, 100027, No. China
Tel: 86-10-85282100 Fax: 86-10-85282104

**China - Chengdu**
Microchip Technology Consulting (Shanghai) Co., Ltd., Chengdu Liaison Office
Rm. 2401, 24th Floor,
Ming Xing Financial Tower
No. 88 TIDU Street
Chengdu 610016, China
Tel: 86-28-6766200 Fax: 86-28-6766599

**China - Fuzhou**
Microchip Technology Consulting (Shanghai) Co., Ltd., Fuzhou Liaison Office
Unit 28F, World Trade Plaza
No. 71 Wusi Road
Fuzhou 350001, China
Tel: 86-591-7503506 Fax: 86-591-7503521

**China - Shanghai**
Microchip Technology Consulting (Shanghai) Co., Ltd.
Room 701, Bldg. B
Far East International Plaza
No. 317 Xian Xia Road
Shanghai, 200051
Tel: 86-21-6275-5700 Fax: 86-21-6275-5060

**China - Shenzhen**
Microchip Technology Consulting (Shanghai) Co., Ltd., Shenzhen Liaison Office
Rm. 1315, 13/F, Shenzhen Kerry Centre,
Renminnan Lu
Shenzhen 518001, China
Tel: 86-755-2350361 Fax: 86-755-2366086

**Hong Kong**
Microchip Technology Hongkong Ltd.
Unit 901-6, Tower 2, Metroplaza
223 Hing Fong Road
Kwai Fong, N.T., Hong Kong
Tel: 852-2401-1200 Fax: 852-2401-3431

**India**
Microchip Technology Inc.
India Liaison Office
Divyasree Chambers
1 Floor, Wing A (A3/A4)
No. 11, O'Shaugnessey Road
Bangalore, 560 025, India
Tel: 91-80-2290061 Fax: 91-80-2290062

**Japan**
Microchip Technology Japan K.K.
Benex S-1 6F
3-18-20, Shinyokohama
Kohoku-Ku, Yokohama-shi
Kanagawa, 222-0033, Japan
Tel: 81-45-471- 6166 Fax: 81-45-471-6122

**Korea**
Microchip Technology Korea
168-1, Youngbo Bldg. 3 Floor
Samsung-Dong, Kangnam-Ku
Seoul, Korea 135-882
Tel: 82-2-554-7200 Fax: 82-2-558-5934

**Singapore**
Microchip Technology Singapore Pte Ltd.
200 Middle Road
#07-02 Prime Centre
Singapore, 188980
Tel: 65-334-8870 Fax: 65-334-8850

**Taiwan**
Microchip Technology Taiwan
11F-3, No. 207
Tung Hua North Road
Taipei, 105, Taiwan
Tel: 886-2-2717-7175 Fax: 886-2-2545-0139

## EUROPE

**Denmark**
Microchip Technology Nordic ApS
Regus Business Centre
Lautrup hoj 1-3
Ballerup DK-2750 Denmark
Tel: 45 4420 9895 Fax: 45 4420 9910

**France**
Microchip Technology SARL
Parc d'Activite du Moulin de Massy
43 Rue du Saule Trapu
Batiment A - ler Etage
91300 Massy, France
Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79

**Germany**
Microchip Technology GmbH
Gustav-Heinemann Ring 125
D-81739 Munich, Germany
Tel: 49-89-627-144 0 Fax: 49-89-627-144-44

**Italy**
Microchip Technology SRL
Centro Direzionale Colleoni
Palazzo Taurus 1 V. Le Colleoni 1
20041 Agrate Brianza
Milan, Italy
Tel: 39-039-65791-1 Fax: 39-039-6899883

**United Kingdom**
Arizona Microchip Technology Ltd.
505 Eskdale Road
Winnersh Triangle
Wokingham
Berkshire, England RG41 5TU
Tel: 44 118 921 5869 Fax: 44-118 921-5820

01/18/02