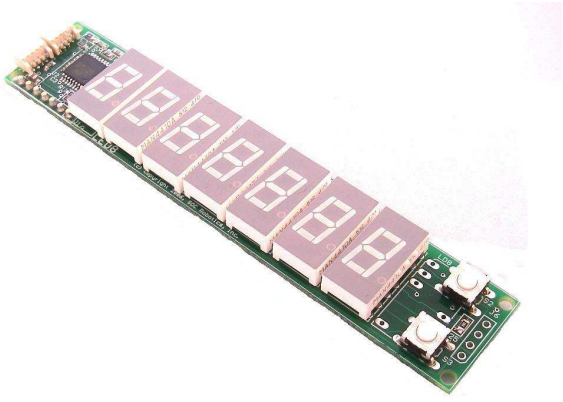
LED8 Display Controller

Technical Reference Manual

PCB Rev 1.0



LED8S

LED8

www.soc-robotics.com

Warranty Statement

SOC Robotics warrants that the Product delivered hereunder shall conform to the applicable SOC Robotics Data Sheet or mutually agreed upon specifications and shall be free from defects in material and workmanship under normal use and service for a period of 30 days from the applicable date of invoice. Products that are "samples", "design verification units", and/or "prototypes" are sold "AS IS," "WITH ALL FAULTS," and without a warranty. If, during such warranty period, (i) SOC Robotics is notified promptly in writing upon discovery of any defect in the goods, including a detailed description of such defect; (ii) such goods are returned to SOC Robotics, DDP SOC Robotics facility accompanied by SOC Robotics Returned Material Authorization form; and (iii) SOC Robotics examination of such goods discloses to SOC Robotics satisfaction that such goods are defective and such defects are not caused by accident, abuse, misuse, neglect, alteration, improper installation, repair, improper testing, or use contrary to any instructions issued by SOC Robotics. SOC Robotics shall (at its sole option) either repair, replace, or credit Buyer the purchase price of such goods. No goods may be returned to SOC Robotics without SOC Robotics Returned Material Authorization form. Prior to any return of goods by Buyer pursuant to this Section, Buyer shall afford SOC Robotics the opportunity to inspect such goods at Buyer's location, and any such goods so inspected shall not be returned to SOC Robotics without its prior written consent. SOC Robotics shall return any goods repaired or replaced under this warranty to Buyer transportation prepaid, and reimburse Buyer for the transportation charges paid by Buyer for such goods. The performance of this warranty does not extend the warranty period for any goods beyond that period applicable to the goods originally delivered.

THE FOREGOING WARRANTY CONSTITUTES SOC ROBOTICS EXCLUSIVE LIABILITY, AND THE EXCLUSIVE REMEDY OF BUYER, FOR ANY BREACH OF ANY WARRANTY OR OTHER NONCONFORMITY OF THE GOODS COVERED BY THIS AGREEMENT. THIS WARRANTY IS EXCLUSIVE, AND IN LIEU OF ALL OTHER WARRANTIES. SOC ROBOTICS MAKES NO OTHER WARRANTIES, EXPRESS, IMPLIED, OR STATUTORY, INCLUDING WITHOUT LIMITATION ANY WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. THE SOLE AND EXCLUSIVE REMEDY FOR ANY BREACH OF THIS WARRANTY SHALL BE AS EXPRESSLY PROVIDED HEREIN.

Limitation on Liability

Notwithstanding anything to the contrary contained herein, SOC Robotics shall not, under any circumstances, be liable to Buyer or any third parties for consequential, incidental, indirect, exemplary, special, or other damages. SOC Robotics total liability shall not exceed the total amount paid by Buyer or SOC Robotics hereunder. SOC Robotics shall not under any circumstances be liable for excess costs of re-procurement.

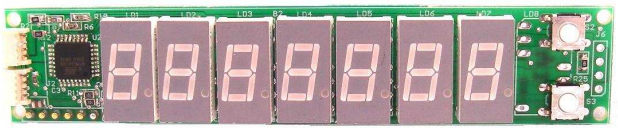
1935A–08/00/5M

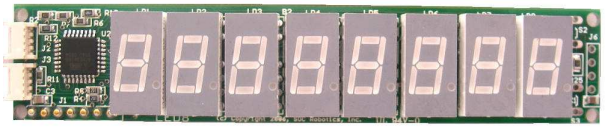
Table of Contents

Introduction

Features:

- 7 or 8 LEDs (Green or Red – 0.4" high)
- On board AVR processor Atmega168
- 16K Flash, 1K SRAM, 512 EEPROM
- 8MHz Internal Osc (optional 20MHz crystal)
- Two I2C ports
- Two switches (or optional 8th LED)
- Analog input port
- CISP ISP programming interface
- Serial UART interface
- Optional 3.0V 1/2AA Battery Holder
- 3-5V operation
- Dimensions: 4.44 x 0.85 inch





Hardware

The **LED8** is a smart seven or eight digit LED display board with an onboard Atmega168 AVR RISC processor running at 8MHz. The **LED8** processor has 16K Flash, 1K SRAM and 512bytes EEPROM. The board comes preloaded with an application that displays data received on the I2C port. The processor is programmed via a CISP port that is compatible with the Atmel ISP programming protocol. Mounting holes allow a 3V 1/2AA battery holder to be mounted on the back of the board for mobile applications. The **LED8** is available as a seven digit display with two switches or an eight digit display without switches.

Software

The **LED8** is programmed in C using either a GNU C Compiler, AVR Studio V4.13 or higher with GNU C integrated with the IDE or a third party IDE such as ICCAVR from ImageCraft. Check the SOC Robotics web site www.soc-robotics.com for program examples and ICCAVR project files. Example code is provided to communicate on the I2C port, read/write EEPROM and how to use the UART.

The example Project files that come with the LED8 were written using the ImageCraft ICCAVR Windows IDE. With the release of AVR Studio V4.13 the open source AVR GNU C Tool chain is now integrated with the AVR Studio V4 Windows IDE and is recommended for cost sensitive development projects. The ImageCraft IDE is a low cost commercial C development environment and includes support for 64bit floating point operation in the PRO version.

An Arduino boot loader is under development and should be available in early April 2010 for free download from our web site.

Configurations

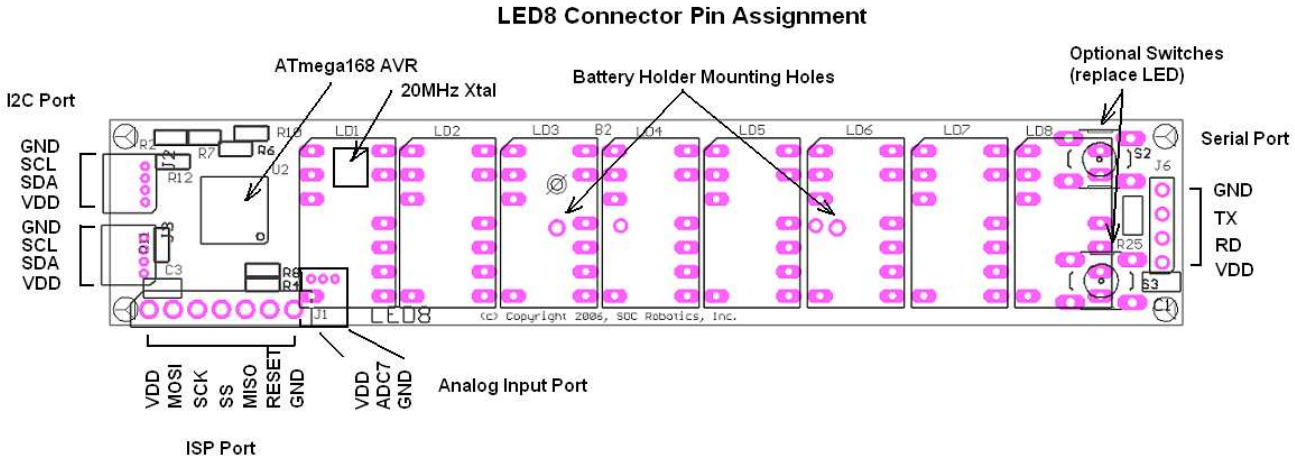
The **LED8** is available in two different configurations – 7 LEDs with two switches or eight LEDs.

The default internal oscillator runs the processor at 8MHz which allows operation at either 3.3V or 5V. An optional 10MHz oscillator can be installed for 3.3V operation or a 20MHz crystal for 5V operation.

2.0 LED8 Detailed Description

2.1 Introduction

The LED8 is a compact embedded multi-digit display processor with its own dedicated 8 bit RISC processor. Programmed in C and optionally powered by a standalone battery the LED8 is an excellent embedded processor platform for small mobile applications that require a large bright 7 or 8 digital display.



2.2 Processors

The LED8 has an 8bit RISC AVR ATmega168 processors with 16K Flash, 512bytes EEPROM and 1K SRAM. The processor has a UART, SPI port, TWI I2C, 8 Channel 10 bit A/D, digital IO ports, interrupts and timer/counter pins. A device datasheet is available from the Atmel web site providing very detailed information on the internal peripherals. Not all on chip peripherals are brought to connectors. The LED8 has a pad for an optional external crystal – either a 20MHz or 10MHz crystal can be installed. The ATmega168 requires 5VDC to run at 20MHz. Operation from an external crystal allows faster baud rates.

A software driven display function manages the activation of each common anode line so that one character at a time is displayed on each LED. All seven or eight LEDs are then multiplexed so the entire display shows a 7 or 8 digit number. The source code for this function is included with the product. By varying the update rate the intensity of the display can be varied.

An I2C slave function is also implemented on the board so that an external I2C master can use the LED8 for display purposes. The source code for this function is included with the application.

The Atmega168 is a general purpose processor so the user can reprogram the LED8 for other custom functions in a I2C or Serial Uart connected application or standalone application with the 1/2AA battery holder.

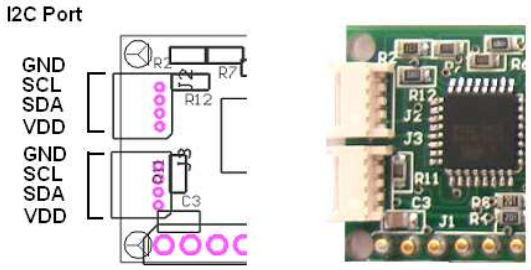
Nominal power consumption of the board is 24ma. If internal peripherals are turned off and the processor placed in a sleep state power drops below 1ma.

2.3 TWI I2C Port

The LED8 has two TWI I2C ports for communicating with smart peripherals or other I2C peripherals. The TWI port uses a 4 pin Molex picoBlade connector with power and ground so the LED8 can power other

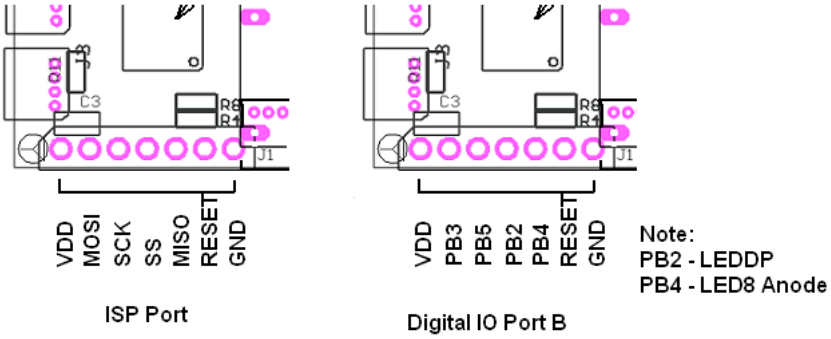
peripherals or be powered itself via this connector. Two connectors are available so the LED8 can be daisy chained in a multi I2C peripheral application.

The I2C ports J12 and J8 are common. Note that all devices connected on a common I2C daisy chain must be at the same voltage. A voltage translation board TM1 or TM1R is available to convert before two different voltages.



2.4 SPI/ISP Programming Port

The LED8 has an SPI port J1 – two of these pins can be used for general purpose digital IO – the other two pins are connected to the LED DP and the eight LED anode and can't be used. The SPI port is used to program the device.



The SPI port is used as an ISP programming port. The LED8 is programmed using an ISP10 Parallel Port programming adapter or any 10pin Atmel ISP compatible programming adapter. The LED8 is supplied with a CISP adapter that converts the LED8 programming pins on J1 to an Atmel compatible 10pin adapter. The CISP attaches to connector J1 located on the bottom side of the board – attach as shown in the picture below. The LED8 can also be programmed using a USB10 USB 2.0 peripheral.

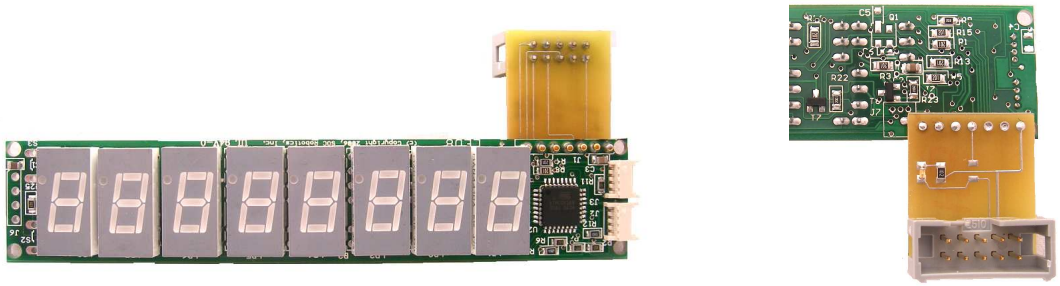


Figure 2-3. **CISP ISP adapter showing correct attachment to the LED8.**

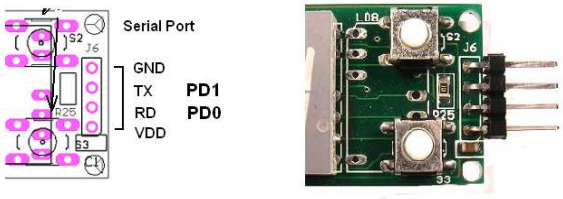
2.5 Molex Connectors

The LED8 uses two small Molex picoBlade 4 pin and 3 pin connectors with 1.25mm pin spacing (4 pin Molex Part No. 53048-0410 - Digikey Part No. WM1744-ND - 5 pin Molex Part No. 51021-0500 – Digikey part no. WM1745-ND). These connectors mate with female Molex 4 and 5 pin housing connectors (4 pin housing Molex Part No. 51021-0400 - Digikey Part No. WM1722-ND – 5 pin housing Molex Part No. 51021-0500 - Digikey Part No. WM1723-ND).

The housing connectors has two different crimp terminal types: 26-28AWG (Molex Part No. 50079-8000 - Digikey Part No. WM1722-ND – Crimp tool 63811-0300) and 28-32AWG (Molex Part No. 50058-8000 - Digikey Part No. WM1775-ND - Crimp tool 63811-0200).

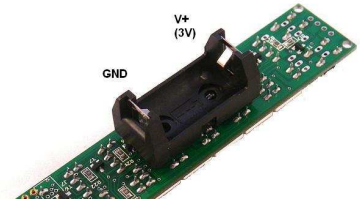
2.6 Serial Port

The LED8 has a four pin serial port that connects to the processor's RXD and TXD UART lines. Software sets the baud rate and an interrupt driver serial routine is provided to setup a serial communications link. The serial port can also be configured under software control as two digital IO lines both of which can be configured at pin change interrupts. The install Arduino bootloader communicates via the serial port.



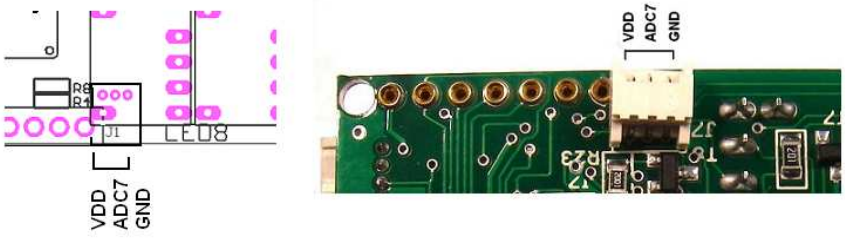
2.7 Optional Battery Holder

The LED8 has holes for installation of a 1/2AA battery holder to enable mobile operation.



2.8 Analog Input Port

The LED8 has a three pin analog input port on J7. J7 connects to ADC7. ADC7 is a 10bit A/D with a maximum sample rate of 35kHz. Analog data input on ADC7 can be displayed on the LED display in real time.

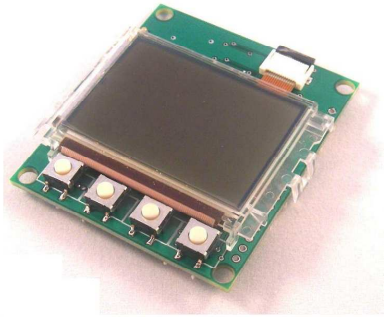


Analog Input Port
(Top view)

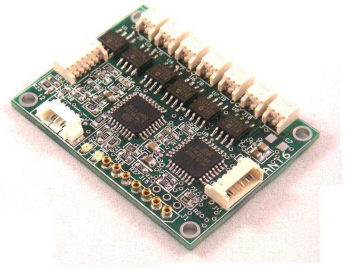
Analog Input Port
(Bottom view)

2.9 Related Peripherals

The LED8 attaches to or can communicate with other SOC Robotics Embedded Processor devices such as the SmartLCD, USB10, Cricket, Ant6 and other peripherals.



. SmartLCD display that communicates via I2C.



Ant6 6 Channel H-Bridge Controller

2.10 Applications

The LED8 is a small, optionally battery powered, 7 or 8 digit LED display device with general purpose digital IO, a 10 bit analog input port, serial UART port and two I2C port.

The LED8 can be programmed for custom applications by building on the supplied default application. The default application configures the LED8 as a slave I2C device for character or number display.

3.0 Software and Applications

3.1 Introduction

The LED8 is a Smart Display peripheral that executes a communications program running on the on board Atmega168 processor. The LED8 is shipped with an application that responds to I2C commands sent by the MC433G, MK54, MK200, SAM58GC or MK400 GStep controllers. If four LED8's are used in an application then the default I2C address must be changed to suit the number of LED8's. A typical MC433G controller assumes there are four LED8's at address 0x20, 0x21, 0x22 and 0x24. Four LED8s mounted on a common interconnect PCB is available and is called the DRO48 or DRO58 for 4 axis and 5 axis Digital Read Outs respectively.

Custom programs can be written to control display operation by monitoring the analog input and/or digital inputs.

3.2 Display Application Description for MC433G

The default software programmed into LED8 monitor commands received on the I2C bus. The LED8 monitors the status of the two switches and returns this information when requested by the I2C master. The default address of the LED8 is 0x20.

3.3 I2C/Serial Command Summary

Commands to control display state are sent to the LED8 on the I2C or Serial Port.

I2C Port

The I2C port is a two wire party line communication protocol running at 50-400KHz (distance dependent). Each device on an I2C bus must have a unique address. The LED8's address is 0x20 (32 decimal). Commands sent to the LED8 are either write or read commands. Write commands contain the device address followed by one or more ASCII characters. Read commands are preceded by a write command that selects the parameter of interest returned in the next read command. Sub-addressing is not supported.

The following write commands are supported by the Version 1.0 software:

Command (Hex)	Function
0x00 or 'i'	Write integer – command followed by two byte integer
0x01 or 'h'	Write hex – command followed by two byte integer
0x02 or 'b'	Write binary – command followed by two byte integer
0x03 or 'f'	Write float – command followed by four byte floating point value
0x04 or 'p'	Write position - command followed by four byte long
0x05 or 's'	Write position scale – command followed by four byte float
0x06 or 'd'	Set LED8 to default state – set decimal place two byte integer

The Set LED8 to default state command assumes a two byte integer follows. The integer defines the number of significant digits to display. The difference between the Write Float and Write Position is that the Write float command is followed by a four byte floating point number that is displayed as sent. The Write Position command is a long which is converted to a float by multiplying the long value by the position scale value sent by the Write Position Scale command. The default position scale is 0.00015.

A single read command is supported. Read sub-addressing is not used in this version.

The I2C read command retrieves the current state of the two switches of the 7 Segment Display. The state of both switches is returned in a single byte. Switch 2 (top switch) value is returned with bit 0 set to 1 if

the switch 2 is not pressed and 0 if the switch is pressed. Switch 3 is returned with bit 1 set to 1 if the switch is not pressed and 0 if the switch is pressed. The control software must interpret the byte appropriately.

The following C source shows how to use several of the commands:

Initialize LED8 by sending TWI address and location of decimal point.

```
// Configure LED8 display device via TWI to accept float with
// decimalpt position set
void  Init_LED8_TWI(unsigned char twiaddress,int decimalpt)
{
        void *intvalue;

        messageBuf[0] = 6;
        intvalue = &messageBuf[1];
        *(int*)intvalue = decimalpt;
        TWI_Start_Write(twiaddress,messageBuf,3);
}
```

Send a floating point value to the LED8 at TWI address.

```
// Write float value to LED8 via TWI
void Write_Float_LED8(unsigned char twiaddress,float fvalue)
{
        void *value;

        messageBuf[0] = 3;
        value = &messageBuf[1];
        *(float *)value = fvalue;
        TWI_Start_Write(twiaddress,messageBuf,5);
}
```

Retrieve current state of the LED8 switches by sending a read request to TWI address.

```
// Read state of LED8 switches
unsigned char Read_LED8_Switches(char twiaddress)
{
       TWI_Start_Read(twiaddress,3);
       TWI_Get_Data_From_Transceiver(messageBuf,1);
       return messageBuf[0];
}
```

Detect and initialize four LED8's (DRO48) at TWI addresses 0x20, 0x21, 0x22 and 0x24.

```
#define    TWI_MTX_ADR_NACK  0x20
// Check existance of LED8's: 0x20, 0x21, 0x22, 0x24
void Enumerate_LED8()
{
        led8state = 0;

        // Check for X axis LED8
        Init_LED8_TWI(LED8XAXIS,LEDFIVEDECIMAL);
         if(TWI_Get_State_Info()!=TWI_MTX_ADR_NACK) {
                led8state |= 0x01;
                delaymsec(10);
                Write_Float_LED8(LED8XAXIS,0.0);
                }

        // Check for y axis LED8
        Init_LED8_TWI(LED8YAXIS,LEDFIVEDECIMAL);
         if(TWI_Get_State_Info()!=TWI_MTX_ADR_NACK) {
               led8state |= 0x02;
                delaymsec(10);
                Write_Float_LED8(LED8YAXIS,0.0);
                }


        // Check for z axis LED8
        Init_LED8_TWI(LED8ZAXIS,LEDFIVEDECIMAL);
         if(TWI_Get_State_Info()!=TWI_MTX_ADR_NACK) {
                led8state |= 0x04;
```

```
                delaymsec(10);
                Write_Float_LED8(LED8ZAXIS,0.0);
                }

            // Check for a axis LED8
            Init_LED8_TWI(LED8AAXIS,LEDFIVEDECIMAL);
             if(TWI_Get_State_Info()!=TWI_MTX_ADR_NACK) {
                led8state |= 0x08;
                delaymsec(10);
                Write_Float_LED8(LED8AAXIS,0.0);
                }

            // Check for a axis LED8
            Init_LED8_TWI(LED8DAXIS,LEDFIVEDECIMAL);
             if(TWI_Get_State_Info()!=TWI_MTX_ADR_NACK) {
                led8state |= 0x10;
                delaymsec(10);
                Write_Float_LED8(LED8DAXIS,0.0);
                }
}
```

Display four floating point numbers representing the current position of the X, Y, Z and A axis of the MC433G.

```
void Display_LED84()
{
    float xpos,ypos,zpos,apos,dpos;

    if((led8state&0x01)) {
           xpos = ((float)xoldpos)*xresolution;
           if(xdirpolarity==NEGATIVE)  xpos = -xpos;
           Write_Float_LED8(LED8XAXIS,xpos);
               }

       if((led8state&0x02)) {
              ypos = ((float)yoldpos)*yresolution;
              if(ydirpolarity==NEGATIVE)  ypos = -ypos;
              Write_Float_LED8(LED8YAXIS,ypos);
              }

       if((led8state&0x04)) {
              zpos = ((float)zoldpos)*zresolution;
              if(zdirpolarity==NEGATIVE)  zpos = -zpos;
              Write_Float_LED8(LED8ZAXIS,zpos);
              }

       if((led8state&0x08)) {
              apos = ((float)holdpos)*aresolution;
              if(adirpolarity==NEGATIVE)  apos = -apos;
              Write_Float_LED8(LED8AAXIS,apos);
              }

       if((led8state&0x10)) {
              dpos = ((float)doldpos)*dresolution;
              if(ddirpolarity==NEGATIVE)  dpos = -dpos;
              Write_Float_LED8(LED8DAXIS,dpos);
              }
}
```

Serial Port

Version 1.0 LED8 software does not respond to commands on the serial port but the current status of LED8 state is output on the serial port at 9600 baud, no parity and one stop bit.

3.4 MK4U, MK54, SAM58GC, MK200, MK800 Interaction

The Modular Motion family of stepper motor controllers are all compatible with the LED8.

3.5 Chinook, Whistler, P0, P1 Processor Interaction

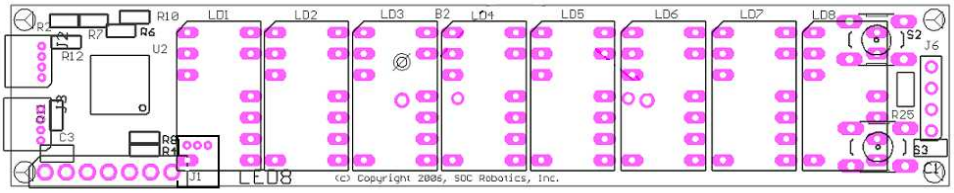
The Chinook (Wasp, Cricket, Ant6, SmartLCD) , Whistler (WaspARM,SAM48), and Blackfin (P0,P1) processor families can all control the LED8 via the I2C interface.

Note that a LED8 with 20MHz crystal must operate at 5V.

4.0 Electrical and Mechanical Description

4.1 Component Layout

Components are mounted on both sides of the board. Not all components may be mounted. See the section on optional components for more information.



4.2 Electrical Specifications

Electrical

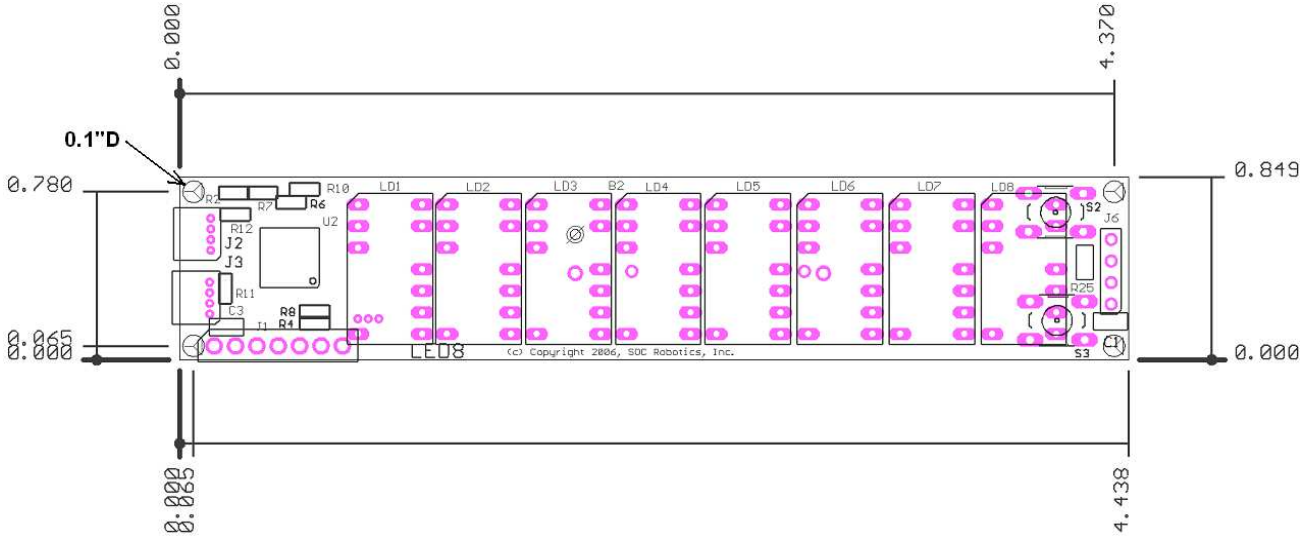
Input power: 5VDC @ 18ma
Sleep Mode: 0.7ma

Mechanical

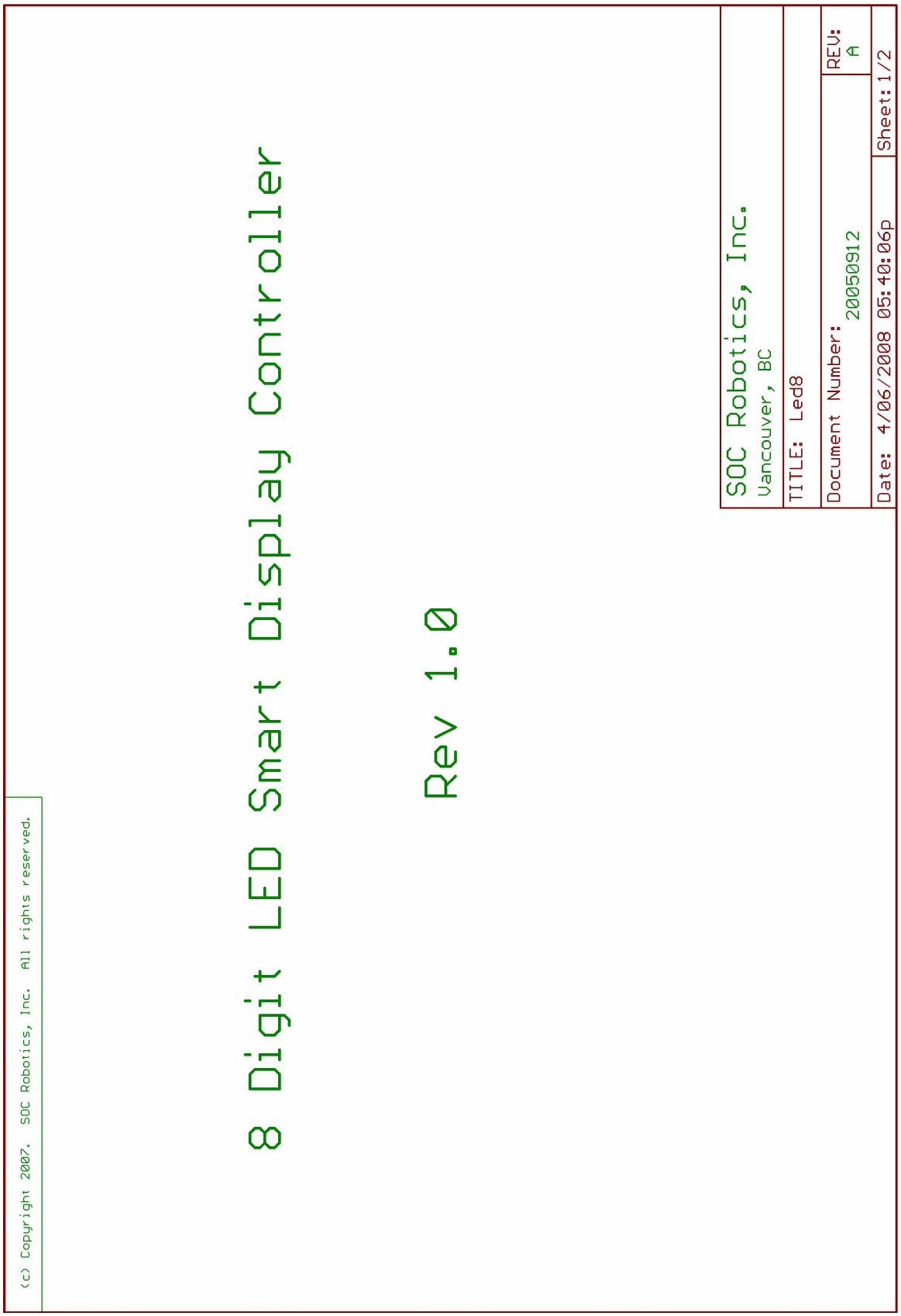
Dimensions: 4.44x0.85 in (four mounting holes)
Weight: 19 grams

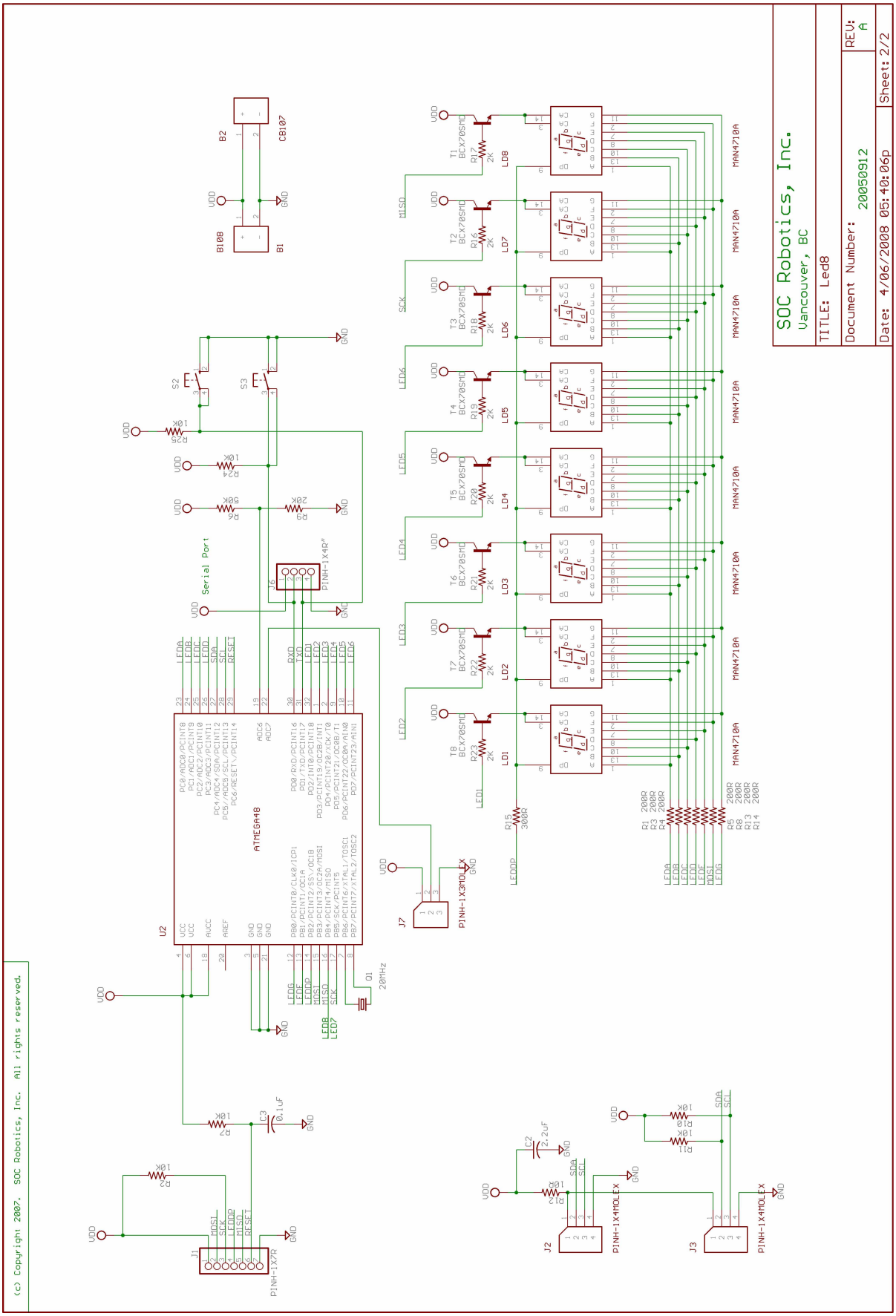
4.3 Mechanical Dimensions

Board dimensions are stated in inches.



5.0 LED8 Circuit Schematics




SOC Robotics, Inc.
Vancouver, BC
TITLE: Led8
Document Number: 20050912
REV: A
Date: 4/06/2008 05:40:06p
Sheet: 2/2
(c) Copyright 2007. SOC Robotics, Inc. All rights reserved.
ATMEGA48
Serial Port
PINH-1X4R"
PINH-1X3MOLEX
PINH-1X4MOLEX
PINH-1X7R
MAN4710A
20MHz

Notes: