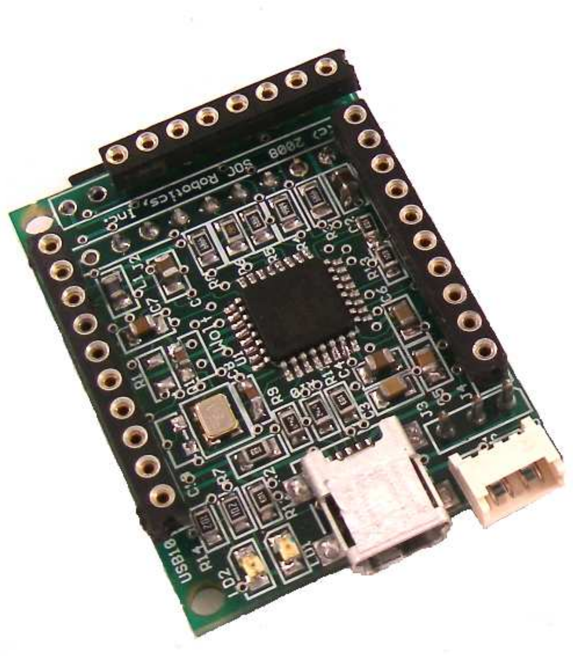
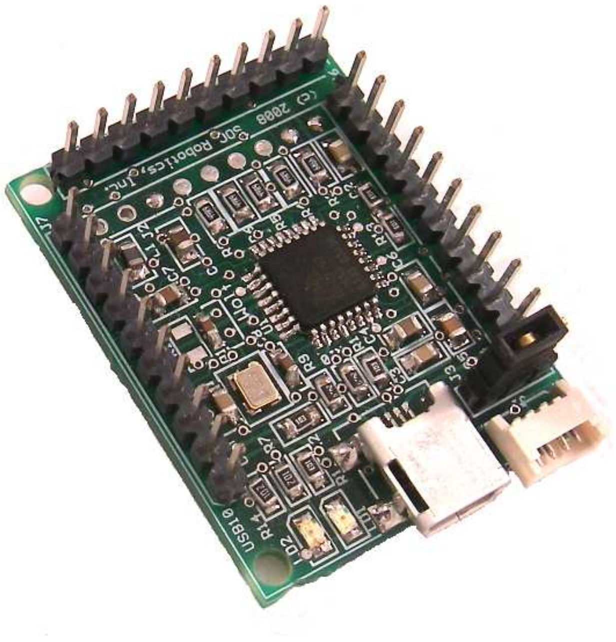
USB10 Smart AVR USB Processor
Ferret

Preliminary Technical Reference Manual
PCB Rev 1.0


USB10M


USB10

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.

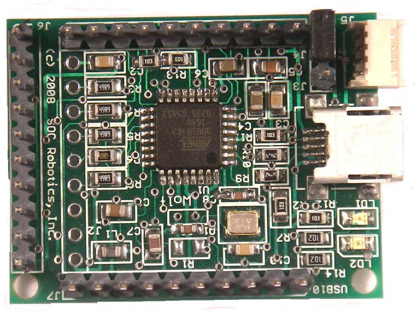
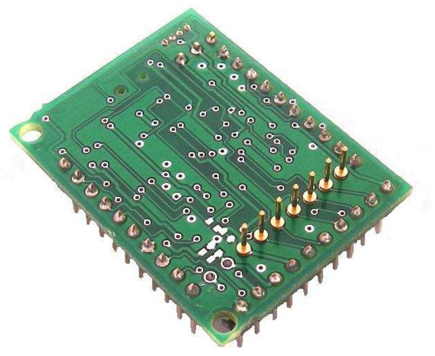
THE FOREGOING WARRANTY CONSTITUTES SOC ROBOTICS EXCLUSIVE LIABILITY, AND THE EXCLUSIVE REMEDY OF BUYER, FOR ANY BREACH OF ANY WARRANTY OR OTHER NONCONFORMITY OF THE GOODS COVERED BY THIS AGREEMENT. THIS WARRANTY IS EXCLUSIVE, AND IN LIEU OF ALL OTHER WARRANTIES. SOC ROBOTICS MAKES NO OTHER WARRANTIES, 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.

Table of Contents

1.0 Introduction

Features:

- USB 2.0 compatible
- AT90USB162 AVR Processor
- 16MHz on board crystal (clock set to 8MHz)
- 21 Digital IO lines
- SPI Port
- TWI I2C Port (software implementation)
- 16K Internal Program Flash (4K taken by boot loader)
- 512 Internal EEPROM
- 512 Internal SRAM
- ISP Programming Port (requires CISP adapter)
- Machine pins on J4, J6, J7 to enable stacking
- GNU C Compiler, Third Party Commercial C Compiler
- Example application included for rapid application development
- Extremely Small form factor (1.51x1.16 in)
- 3.3-5VDC @ 8-25ma Power input (clock dependent)

Hardware

The **USB10** is an extremely compact smart USB peripheral based on the Atmel AVR AT90USB162 embedded processor. With 21 digital IO lines the **USB10** is an excellent platform for USB controlled embedded applications. The AT90USB162 has a master/slave SPI port and high speed full duplex UART. The board is clocked by a 16MHz crystal reduced internally to 8MHz so the board can run at 3.3V or 5V while maintaining reliable USB clock synchronization. The board can be strapped for 3.3V, 5V or target voltage using a jumper. A small 4 pin picobBlade Molex connector compatible with other SOC Robotics embedded processor products supports I2C.

The **USB10** is shipped with a pre-programmed application called the Ferret Control Program that implements several basic functions including a USB to I2C communication device, ISP programmer and motor controller. The Ferret Control Program is provided with complete source code. Additional applications can easily be added to the existing code base. Atmel provides a complete USB software CDC application. An open source USB software package called MyUSB is available to facilitate custom software applications. The Ferret Control Program is based on MyUSB.

The **USB10** consumes 8-25ma depending on clock rate. By changing the internal clock and reducing power to certain peripherals it is possible for the Wasp to operate with very low power consumption. The **USB10** is shipped with a 4K Boot Loader in high memory that allows the AT90USB162 to be programmed via the USB line. A free third party boot loader allows the device to be programmed directly by AVR Studio. The device can also be programmed using a conventional ISP Programming Adapter such as the ISP10 with CISP adapter. The AT90USB162 also has debugWire support.

The **USB10** PCB connectors J4, J6 and J7 are positioned on 0.1" pin spacing so prototype daughter cards can easily be attached to the top or bottom of the board making custom circuit design possible without the need for a custom PCB. An Eagle CAD layout template of the connectors is available to facilitate development of custom circuits.

Software

The **USB10** is programmed in C using either an open source GNU C Compiler, AVR Studio V4.13 or higher with GNU C integrated within 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 project files. Example code is provided to setup a serial communications link with a Windows PC.

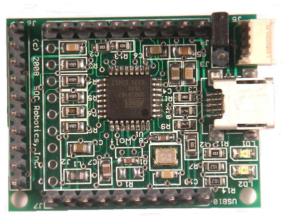
The example Project files that come with the USB10 were written using AVR Studio 4 Windows IDE. With the release of AVR Studio V4.13 the open source AVR GNU C Tool chain is now integrated with the AVR Studio V4 Windows IDE and is recommended for cost sensitive development projects. The ImageCraft IDE is a low cost commercial C development environment and includes support for 64bit floating point numbers in the PRO version.

Configurations

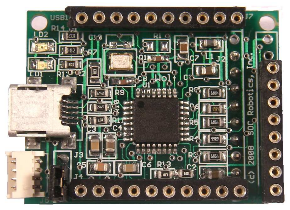
The **USB10** is available in several different configurations summarized in the table below. Note that in order to run at the maximum clock speed of 16MHz the board must be powered at 5V.

Product Name	Crystal Speed	Flyleads	Motor Control	Machine Pins
USB10	16MHz	No	No	No
USB10W	16MHz	No	No	Yes
USB10P	16MHz	Yes	No	Yes
MC-USB	16MHz	No	Yes	Yes

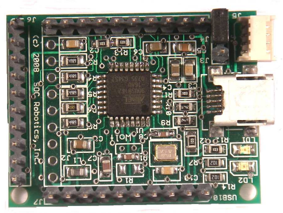
USB10

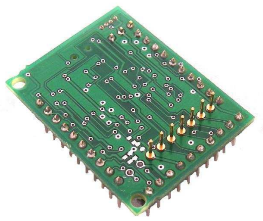


USBW

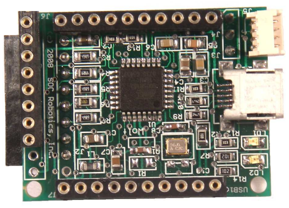


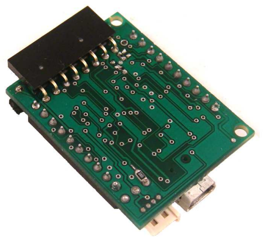
USB10P





MC-USB10

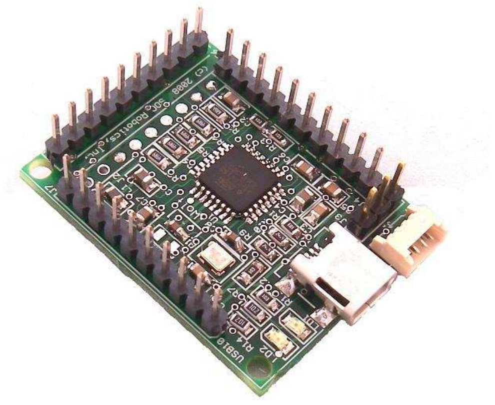
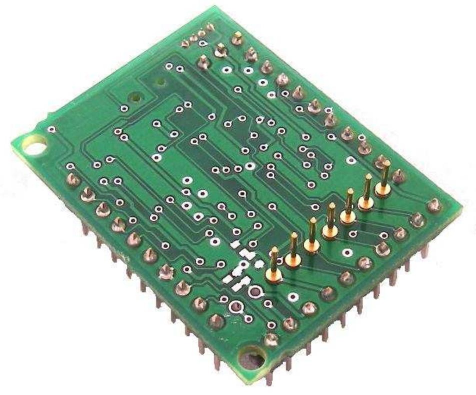




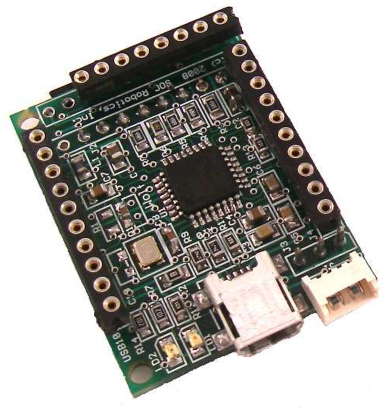
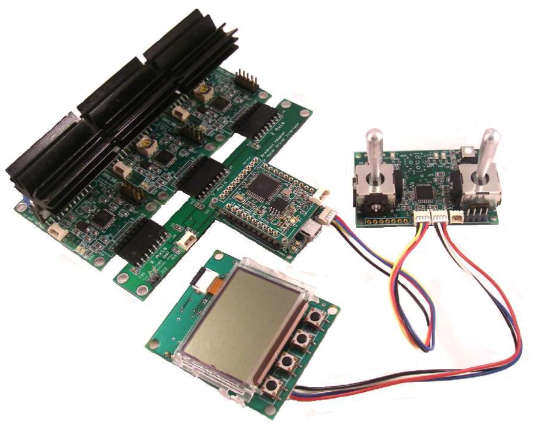
2.0 USB10 Detailed Hardware Description

2.1 Introduction

The **USB10** is USB 2.0 smart controller with a programmable AVR AT90USB162 8 bit RISC processor on board. The **USB10** is a general purpose USB peripheral that can be programmed by the user to perform a wide range of functions. With a rich IO suite the **USB10** can operate as a TWI Master/Slave, SPI Master/Slave, JTAG controller, ISP programmer, PWM generator or general purpose digital IO controller. Atmel supplies complete source code to allow the **USB10** to become a Windows XP CDC peripheral (along with other USB functions). The **USB10** connector pin layout is compatible with the pin layout of the **Wasp**, **WaspARM** and **WaspX** embedded controllers all of which can directly attach to the device.

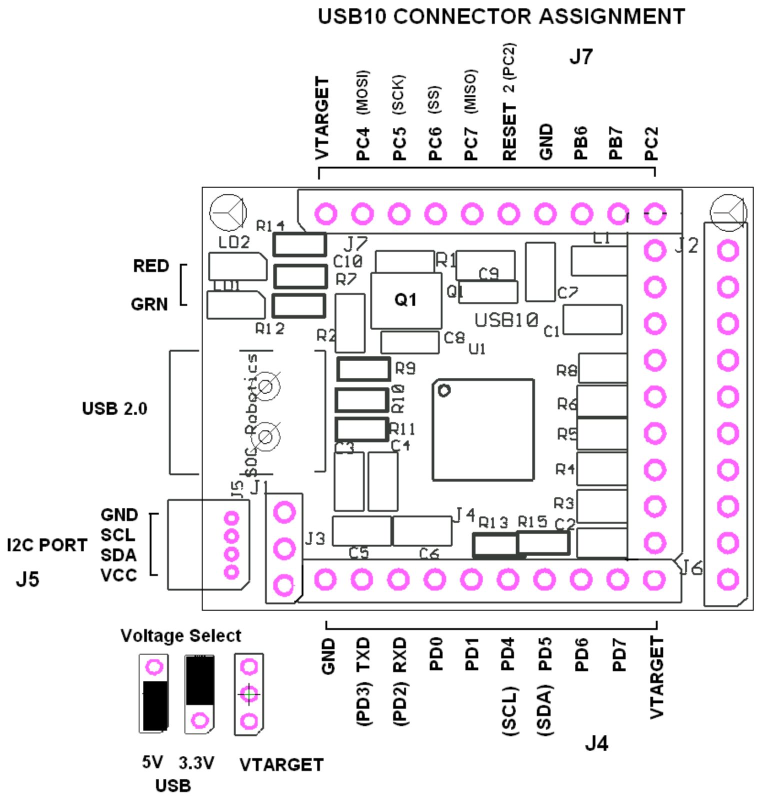



The **USB10** serial port lines on connector J4 is compatible with the serial lines on the **Wasp** and **WaspARM** processors so direct serial communication is possible with these embedded processors. The **USB10** is available in another configuration known as the **MC-USB** for motion control applications.

Note that the **USB10** is shipped with a 16MHz crystal software configured to run at 8MHz. The AT90USB162 can run at the full 16MHz speed but only if the board is supplied with 5V.

The USB10 has three I/O expansion ports, USB Port, I2C port and two display LEDs.



Alternate Functions J6

JTAG	ISP	IO	MC-USB
GND	GND	GND	GND
5VDC	5VDC	5VDC	5VDC
3.3VDC	3.3VDC	GND	GND
GND	GND	PB5	ESTOP
RST	RST	PB4	LIMIT
TDO	MISO	PB3	SCL
TCK	SS	PB0	SDA
TMS	SCK	PB1	DIR
TDI	MOSI	PB2	STEP
VTARGET			VCC

2.2 Expansion PORTS C/B

AVR PORTS C/B are routed to connector J7. The RESET2 pin is connected to PC2 and in combination with PC4 to 7 can be used to implement a software implemented ISP programming function. The Ferret Control Program (described latter in this document) implements an ISP programming function using these pins. The seven pins VTARGET, PC4,5,6,7, RESET2 and GND are compatible with the CISP programming adapter and can be used to program other AVR devices via USB using the utility USB10Prog.exe.

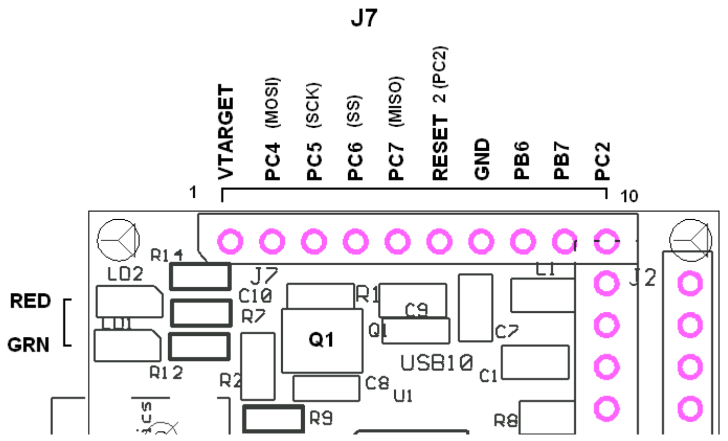
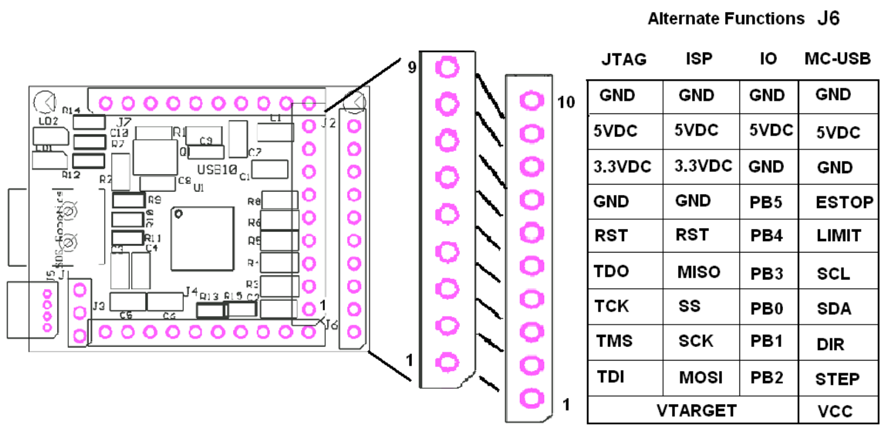


Figure 3-3. **PORTC/B Pin Assignment J7.**

Note that the Rev 1.0 PCB has been modified so that signal PC2 is connected to RESET2 (pin 6-J7). This pin was connected the AT90USB162's reset line. This modification allows the **USB10** to directly program a Wasp attached to the top of the board.

2.3 Expansion PORT B

AVR PORT B is routed to connectors J2 and J6. J2 is a 9 pin connector while J6 is a 10 pin connector. Most of the PORT B signals are routed to the J2 through 49 ohm load resistors. In order to accommodate different connector configurations



Alternate Functions J6

JTAG	ISP	IO	MC-USB
GND	GND	GND	GND
5VDC	5VDC	5VDC	5VDC
3.3VDC	3.3VDC	GND	GND
GND	GND	PB5	ESTOP
RST	RST	PB4	LIMIT
TDO	MISO	PB3	SCL
TCK	SS	PB0	SDA
TMS	SCK	PB1	DIR
TDI	MOSI	PB2	STEP
VTARGET			VCC

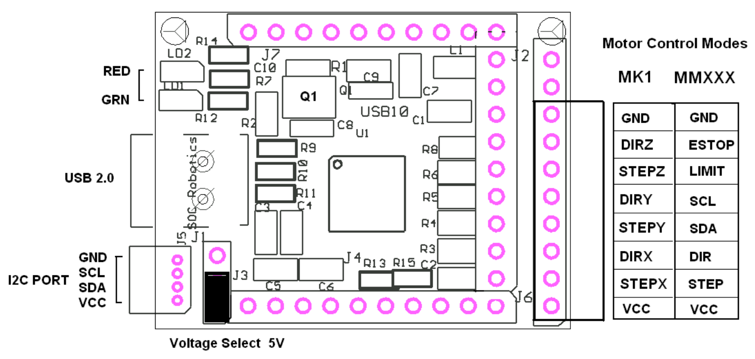
Figure 3-4. **PORTB Pin Assignment J2.**

The function of Pins 8,9 of J2 and 8,9,10 of J6 can be reconfigured with 402 shorting resistors mounted on the bottom of the PCB. The table in the picture above shows the different functions of these pins. See section 2.7 for more information.

The Ferret Control Program (described latter in this document) implements two different functions on PORT B - an ISP programming function and motor control function. PORT B is connected to the AT90USB162s SPI peripheral. The USB10P has special pins mounted on the bottom of J2 so the **USB10P** can be plugged into other SOC Robotics AVR based products.

The **MC-USB** is a version of the **USB10** that interprets J6 as motor control lines. The **MC-USB** directly drives the MM120, MM130, MM133, MM220 and MK1 motor control products. The Ferret Control Program has a motor control function that is enabled under software control (see the section on Ferret).

USB10 CONNECTOR MOTOR ASSIGNMENT



Motor Control Modes

MK1	MMXXX
GND	GND
DIRZ	ESTOP
STEPZ	LIMIT
DIRY	SCL
STEPY	SDA
DIRX	DIR
STEPX	STEP
VCC	VCC

2.4 Expansion PORT D

AVR PORT D is routed to connector J4. PORT D has a full duplex serial UART, software implemented I2C and general purpose digital IO signals. The UART and I2C lines are compatible with the equivalent signals on a Wasp processor. The Ferret Control Program implements a serial pass through function in which a Desktop terminal or software application can communicate with an attached Wasp using the serial port.

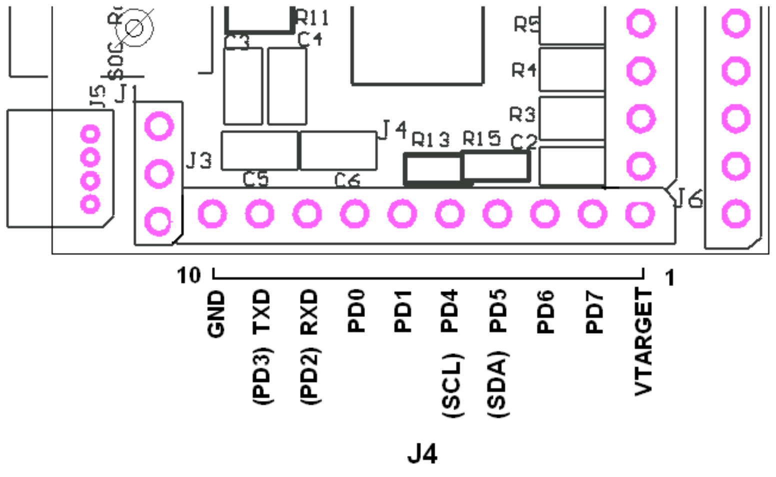
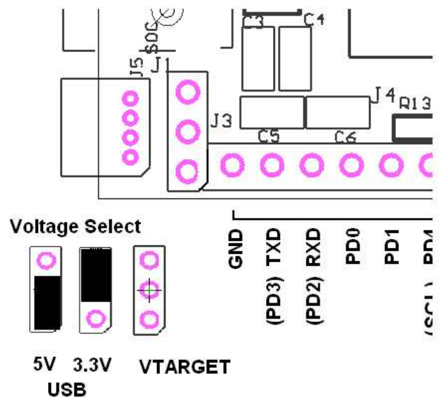
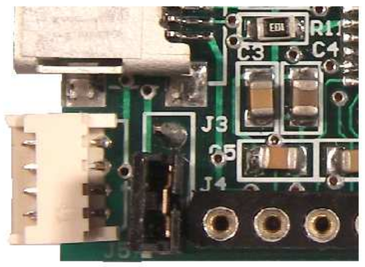


Figure 3-5. **PORT D Pin Assignment J3.**

2.5 Power Select Header

The **USB10** has a Power Select header J3 for selecting 3.3V, 5V or Target voltage. The AT90USB162 regulates the USB 5V power to 3.3V. The processor can run at either setting. Note that for reliable USB 2.0 operation the voltage should be at least 3.3V. The AT90USB162 has an on chip voltage regulator that converts the 5V USB voltage to 3.3V.

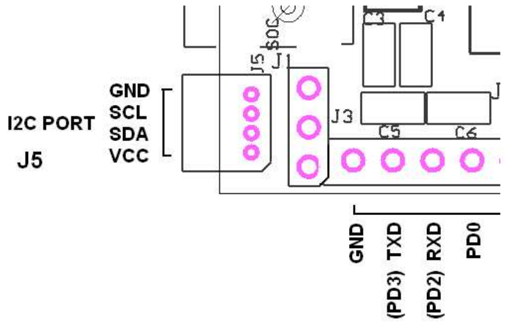


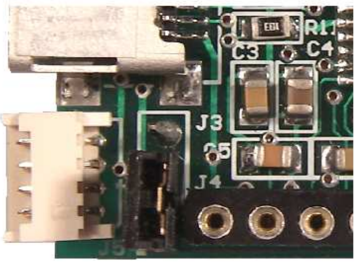


The 3.3V output of the on chip voltage regulator is not designed to supply a lot of current and may cause a brown out interrupt at power up.

2.6 TWI I2C Expansion Port

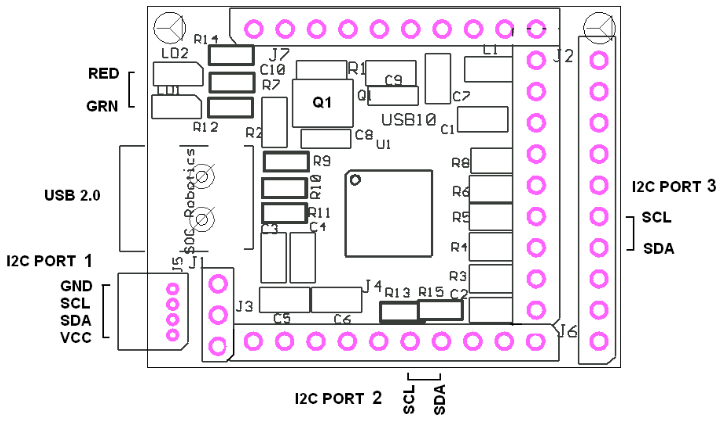
The TWI I2C lines are routed to a separate 4 pin Molex connector J5 (4 pin Molex picoBlade connector). J5 is compatible with the SOC Robotics Smart Peripheral family of motor controllers, LCD displays and data acquisition modules. The AT90USB162s does not have an TWI I2C hardware peripheral so I2C functionality must be implemented in software.





The Ferret Control Program implements an I2C function in software that allows a Desktop Terminal or software application to write or read data on the I2C interface via USB. The Ferret Control Program supports three different I2C hardware pin assignments as shown in the picture below. I2C Port 2 on connector J4 is compatible with the equivalent I2C pin assignment of the Wasp embedded processor. I2C Port 3 on connector J6 is compatible with the I2C implementation in the MM series of motor driver boards.

USB10 I2C CONNECTOR ASSIGNMENT



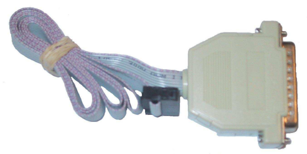
2.7 ISP Programming Port

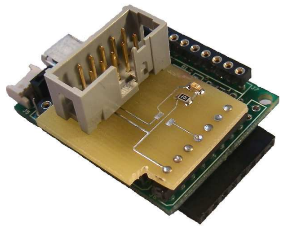
The **USB10** can be programmed using ISP or can be used to program other AVR processors.

ISP Programming the AT90USB162

The **USB10** must be modified to ISP program the AT90USB162. The processor reset line is not accessible to any pin so a wire must be added to the board from the reset pin to the RESET line of the CISP.

See the Atmel ISP programming specification for detailed AT90USB162 programming procedures. The SPI ISP programming lines are on connector J2. A specially modified CISP adapter converts the 7 pin ISP signals to a standard 10 pin Atmel ISP header. The ISP10 parallel port ISP programmer (shown below) attaches to the CISP. Anyone of the following software utilities - ISProg.exe (SOC Robotics, Inc utility), AVRDude, ICCAVR IDE or PonyProg is used to download programs to the AT90USB162.

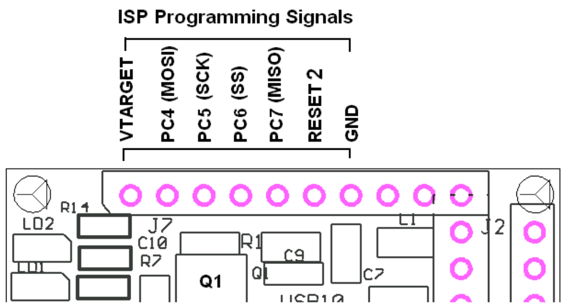


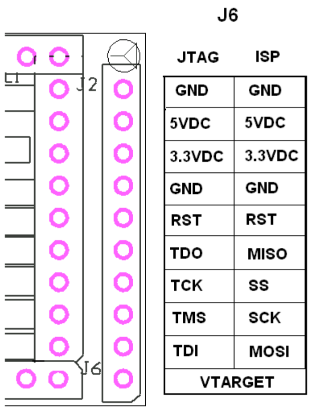


The specially modified CISP programming adapter is attached to J2. A wire is then run from the CISP RESET signal to the processors RESET pin (as shown in the picture). The CISP is a special order item – use ordering code CISPU. Consult the factory for modification instructions.

Programming other AVR processors using the USB10

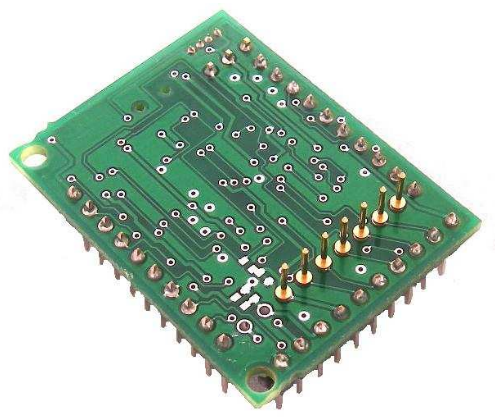
The **USB10** has two ISP programming Ports – one on J7 and the other on J2-J6. J2-J6 connects directly to the AT90USB162 SPI ISP programming signals: MOSI, SCK, MISO. RESET is controlled by PB4. The Ferret Control Program implements an ISP programming sub-function that communicates with a desktop application called USB10prog.exe. USB10Prog.exe is available from the SOC Robotics web site.





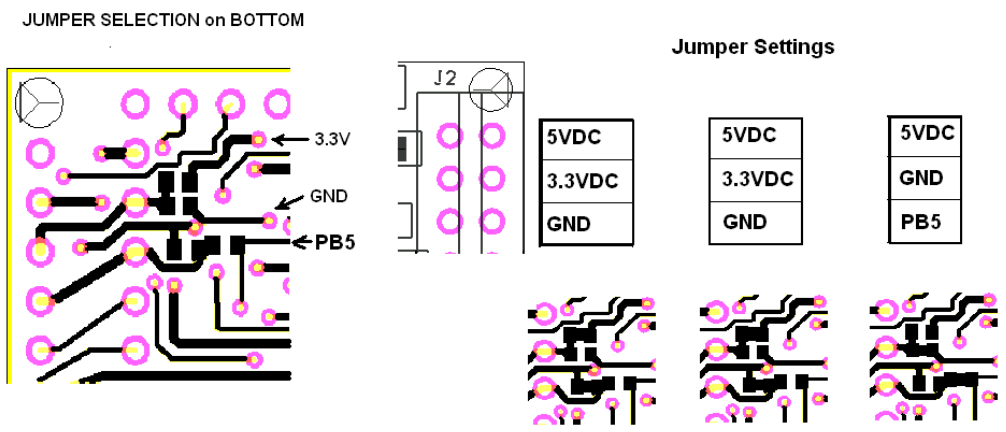
JTAG	ISP
GND	GND
5VDC	5VDC
3.3VDC	3.3VDC
GND	GND
RST	RST
TDO	MISO
TCK	SS
TMS	SCK
TDI	MOSI
VTARGET	

The **USB10P** has special pins on the bottom of the PCB that allow the board to be plugged on to other SOC Robotics AVR based boards such as the Wasp, Ant6, LED8, etc.



2.8 Connector J6 Power Strapping

By attaching 0R 402 resistors to the bottom of the board the **USB10** can be configured to emulate the connection option of other SOC Robotics programming adapters.



3.0 USB10 Detailed Software Description

3.1 Introduction

There are several programming utilities and sample applications for the **USB10.** A rich and growing inventory of open source USB software is available to simplify the creation of USB enabled applications.

The AT90USB162 processor is programmed one of three ways:

- o Standard SPI ISP programming – pull RESET low and toggle SPI lines.
- o Through the USB port using an on chip boot loader (FLIP or AVR Studio).
- o Through the Reset line using a one wire Debugger.

A simple I2C application is programmed into the **USB10** at the factory. The source code for this application is available from the **USB10** Product Page at www.soc-robotics.com.

3.2 Programming the USB10 Flash/EEPROM/Fuses

Before the **USB10** can be programmed or communicated with the appropriate Windows device drivers must be installed. Before plugging the **USB10** into your PCs' USB cable download and installed the free programming application **FLIP** from the Atmel website (go to the AVR/AT90USB162 product page). Atmel Application Note "AVR282 - USB Firmware Upgrade for AT90USB" describes how to install the application. Run the **FLIP** installation software – the AT90USB162 driver is automatically installed. Now plug in the **USB10** and follow the directions in AVR282. FLIP detects the USB10 and allows you to program the device.

If you decide to use the free Atmel CDC example software then the FLIP installed driver is all you need. If you decide to use the open source USB software MyUSB then the appropriate .inf file for this software must be installed. The .inf file is in the project folder that comes with MyUSB. MyUSB is available at `www.fourwalledcubicle.com/MyUSB.php`. A specially configured version of MyUSB for the **USB10** is available from the **USB10** Product Page at www.soc-robotics.com.

ISP Programming

The **USB10** can be programmed using a standard ISP compatible programming tool such as our parallel port **ISP10** programming adapter (with an optional CISP adapter that converts the 10 pin ISP connector to a 7 pin connector) or any ISP AVR serial or USB programmer. Note that device fuses can only be changed using ISP programming.

USB Programming

The **USB10** can also be programmed via the USB line. The **USB10** comes with a small bootloader programmed into the top 4K bytes of memory. The bootloader communicates with a Host PC using either FLIP or AVR Studio 4 (both available from Atmel).

Bootloader

A bootloader is available from Atmel (**`AT90USB162 USB Bootloader v1.0.5`**) that is loaded into the top 4K of the AT90USB162 Flash. The bootloader allows AVR Studio 4 to load programs into on chip flash via the USB port. The bootloader is activated by holding PD7 and RESET low at the same time.

Alternatively an Atmel programming utility called FLIP can be used to load programs into the AT90USB162.

Virtual Serial Application (CDC)

There are several USB code examples available – one from Atmel and an open source version called MyUSB.

A sample CDC application is available for download from Atmel that creates a USB serial communication device.

A third party open source USB application called MyUSB that implements a comprehensive set of embedded USB functions is available from `www.fourwalledcubicle.com/MyUSB.php`. MyUSB implement most of the Atmel CDC example and mode. The Ferret Control Program is implement using MyUSB.

3.3 Ferret Control Program

The **USB10** is shipped with an application called the Ferret Control Program that implements five basic functions. The application is based in MyUSB and the source code is available for download from the **USB10** Product Page at www.soc-robotics.com. The Ferret Control Program is a good starting point to add additional functionality to the **USB10**.

The Ferret Control Program performs the following functions:

- o Create a real time link between the PC Virtual Serial Port and the AT90USB162 UART.
- o Send and Receive data on the I2C port (two versions)
- o Send Motor Control commands to the analog/digital Port lines
- o Convert Intel Hex files into SPI ISP programming functions on either Digital Port J2 or J1
- o Start the boot loader
- o Output a Signon Message giving Software Version level

Functions are selected by entering simple ASCII commands in which the first few characters select the desired operating mode:

c - Send or receive I2C command
s - Enter Serial Transfer mode and transfer UART data bi-directionally
m – Send a Motor Control command
l - Turn the Red or Green LED on or off
p – Enter SPI programming mode and
b – Enter the boot loader
? – Output help message

In the command descriptions below note that ASCII characters may be separated by space key, tab or comma – the application treats these characters as white space and are ignored. A command is not executed until the RETURN or ENTER key is pressed. If white space characters are included in the data stream they are transmitted up to the length of the data variable. Numbers are treated as decimals by default in the range 0 to 64 limiting the maximum data string to 64 characters. Hexadecimal values are entered by preceding the number with a backslash character '\' – for example hexadecimal 0x13 is represented as \x13. The command selection character can be upper or lower case. The ESCAPE character 0x1b is represented by "\c".

I2C Mode

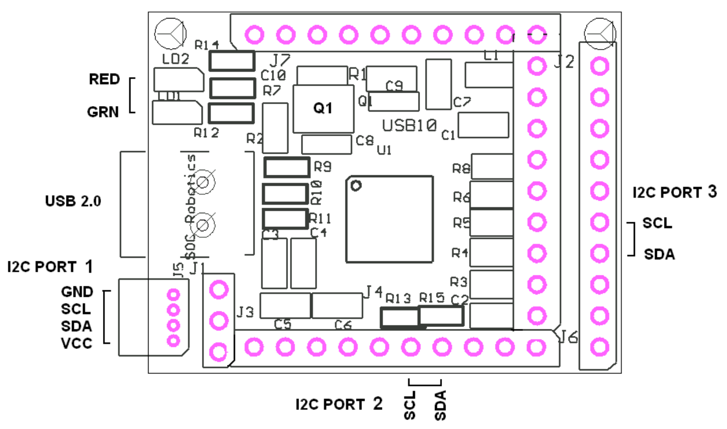
The I2C mode allows the **USB10** to communicate with an I2C device using characters transmitted by the PC on the USB line. The I2C bus has two signals: SDA and SCL and can operate as either a Master or a

Slave device. The **USB10** is pre-configured to operate as a Master. 128 different slave I2C devices can be addressed. The Master implementation does not support Multi-Master operation so care must be taken in Multi-Master configurations.

I2C commands are either write or read. A write command sends a string of characters to a destination slave device. The current implements assumes all writes go to sub-address zero. All reads are also from sub-address zero so if you need to retrieve data from a slave device the slave must be notified which data to send by a previous write command.

The **USB10** supports three different I2C buses. Traffic is directed to a specific bus by entering a number after the I2C Command character is entered. I2C bus 1 and 2 share signals so traffic can not operate simultaneously on both buses. I2C bus 3 is independent of bus 1 and 2 and can operate simultaneously.

USB10 I2C CONNECTOR ASSIGNMENT



A detailed description of the write and read commands follows.

To write to a device use the following format.

I2C write command format:

cn, ADDRESS, w, BYTES_TO_WRITE, data

Where "cn" selects I2C mode and bus based the value of n which can be 1, 2 or 3, ADDRESS is a valid i2c address in decimal, 'w' selects the write command, BYTES_TO_WRITE is the length of data to follow and data is a variable length string of ASCII characters.

Examples:

c1 20 w 4 gs23	- Write the four byte string "gs23" to I2C device at address 20 using bus 1
c1,22,w,2,45	- Write two byte string "45" to I2C device at address 22 using bus 1
c2,\x16,w,2,45	- Write two byte string "45" to I2C device at address 22 using bus 2

I2C read command format:

cn, ADDRESS, r, BYTES_TO_READ

Where "cn" selects I2C mode and bus based the value of n which can be 1, 2 or 3, ADDRESS is a valid i2c address in decimal, 'r' selects the read command and BYTES_TO_WRITE is the length of data to expect.

The read command is usually preceded by a write command that directs the I2C device to return the appropriate data selected by the last write command in the next read command.

Serial Mode

The Serial Mode connects the **USB10s** AT90USB162s UART to the USB line. All subsequent characters sent to the USB10 are output on the AT90USB162s Tx UART line. All characters received on the RX line are returned to the USB.

When in Serial Mode if the PC transmits the ESCAPE character "\c" the **USB10** will exit Serial Mode.

If you wish to send a hex number in the data stream from the PC you can do so by using the hex number format.

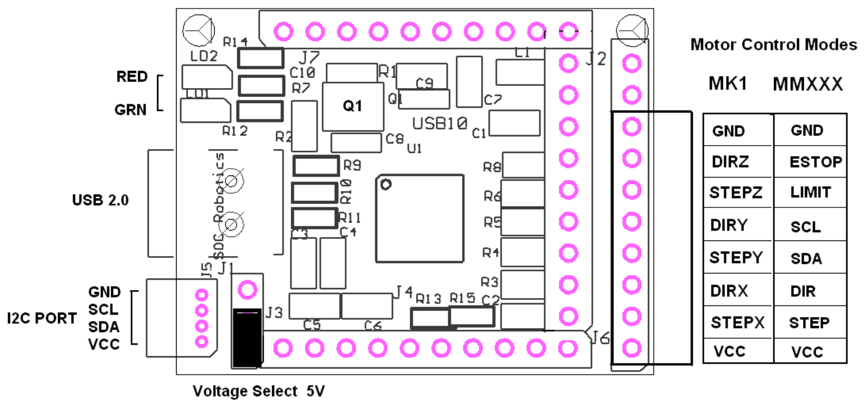
USB10 baud rate is set by the PC's virtual USB properties dialogue, terminal application or application program. Any baud rate, parity, stop bits or data length changes are reflected immediately in the AT90USB162 UART.

Motor Control Mode

The **USB10** can send data or step/direction signals to an attached Motor Driver board such as the MK1, MM120, MM130, MM133 or MM220 using the Motor Control Mode. Two modes are supported Mode 1 and Mode 2. The Motor Control Mode 1 encodes ASCII characters using a one way SPI like signaling format. Digital port J6 is mapped either to the MK1 Breakout board that supports three separate stepper motor drivers or a single axis MM series board. Mode 2 converts single byte commands into step and direction signal pulses.

The port mapping is shown in the picture below.

USB10 CONNECTOR MOTOR ASSIGNMENT



MK1	MMXXX
GND	GND
DIRZ	ESTOP
STEPZ	LIMIT
DIRY	SCL
STEPY	SDA
DIRX	DIR
STEPX	STEP
VCC	VCC

When an MM series board is attached to the **USB10** only X Axis commands should be sent to the driver and I2C commands can be sent by selecting I2C bus 3.

M1 Commands

M1 Motor control command format is as follows:

```
m1, AXIS, BYTES_TO_WRITE, data
```

where 'm' selects motor control mode, AXIS is the either 'x', 'y' or 'z', BYTES_TO_WRITE is the number of bytes in the data packet (decimal format) and data is a string of printable ASCII characters.

Examples:

m1 x 2 tx - Transmit two ASCII characters 't' and 'x' to the X Axis driver
m1,y,11,dd11.0;5.0; - Transmit eleven ASCII characters to the Y Axis driver

M2 Commands
M2 Motor control command format is as follows:

```
m2bbbb..b
```

Where m2 activates motor control mode and b is a specially encode byte in which the value of each bit determines the step and direction signal for the X, Y and Z axis. The bytes is encoded as follows:

Data Bit	Function
D7	1 – Exit Motor Mode
D6	Ignored
D5	DirZ
D4	StepZ
D3	DirY
D2	StepY
D1	DirX
D0	StepX

If bits D0, D2 or D4 are high the USB10 pulses the appropriate axis line with a high going pulse for 5useconds. If D0, D2 or D4 is low no step pulse is generated. The value is the Direction bit is directly mapped to the output.

Sending a byte with bit 7 set high terminates motor mode and returns the USB10 to top level command mode.

Led Control Mode

The USB10 has two LEDs – a Green LED and Red LED. The LED control command allows the user to set the state of each LED.

The following LED control commands are supported:

```
l,LEDID,STATE
```

Where 'l' selects LED control mode, LEDID is either 'g' or 'r' for red or green LED and STATE is either 'o' or 'f' for on or off.

ISP Programming Mode

The **USB10** can program an AVR device using SPI on J6 or bit banging the digital IO lines on port 7. A desktop application called USB10Prog.exe is used to program or upload the contents of AT90USB10

EEPROM or FLASH and select which programming port to use. The details of the protocol is fairly involved and will not be described here. USB10Prog.exe can be downloaded from the USB10 product page at www.soc-robotics.com.

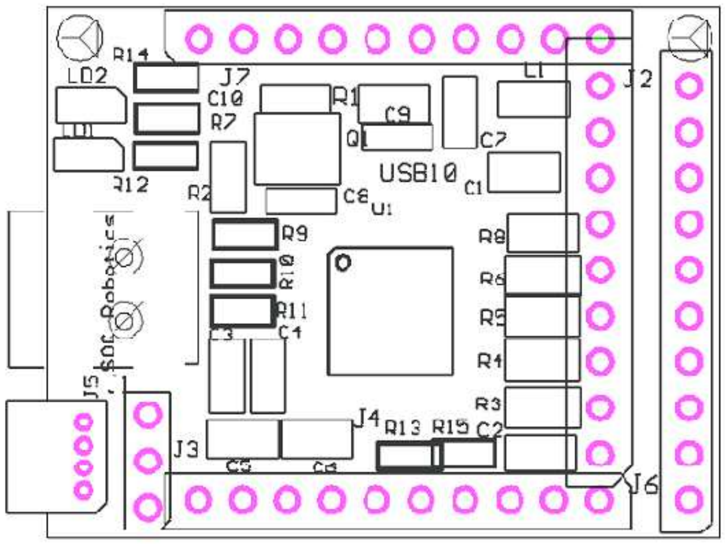
Boot Mode

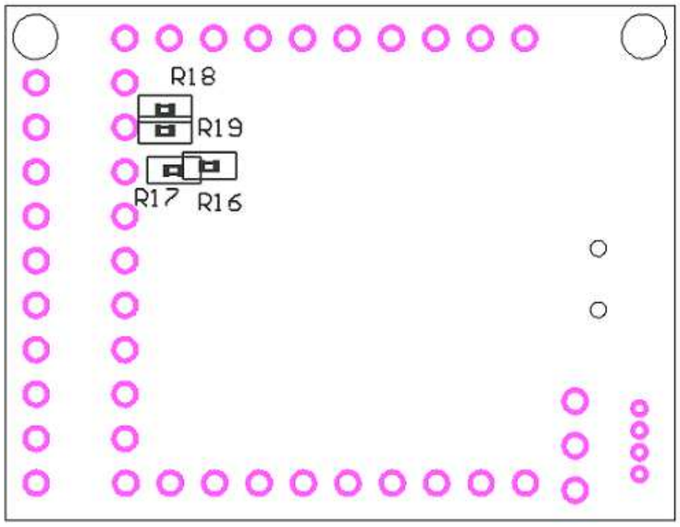
To simplify reprogramming the processor the boot command forces the processor to start the boot routine. An Atmel application called FLIP is then used to load a new application. The command is entered as a 'b' followed by RETURN. The boot process starts immediately. Load a new program using FLIP. FLIP can then start the application. Some terminal programs lock and should be restarted. If the new program fails to run properly hold PD7 and RESET low then re-apply power – this will restart the boot loader.

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:	3.3-5VDC @ 12ma
	Voltage is jumper selectable 3.3V, 5V or VTarget
Sleep Mode:	0.7ma

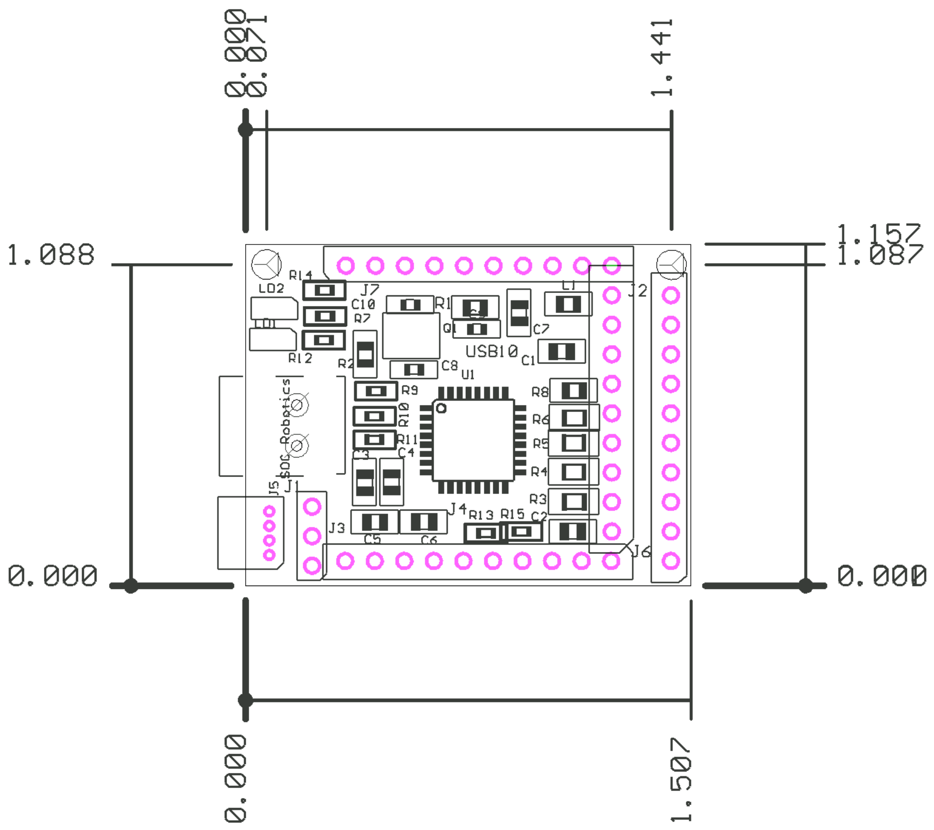
Mechanical

Dimensions:	1.51x1.16 in (one mounting hole)
Weight:	6 grams

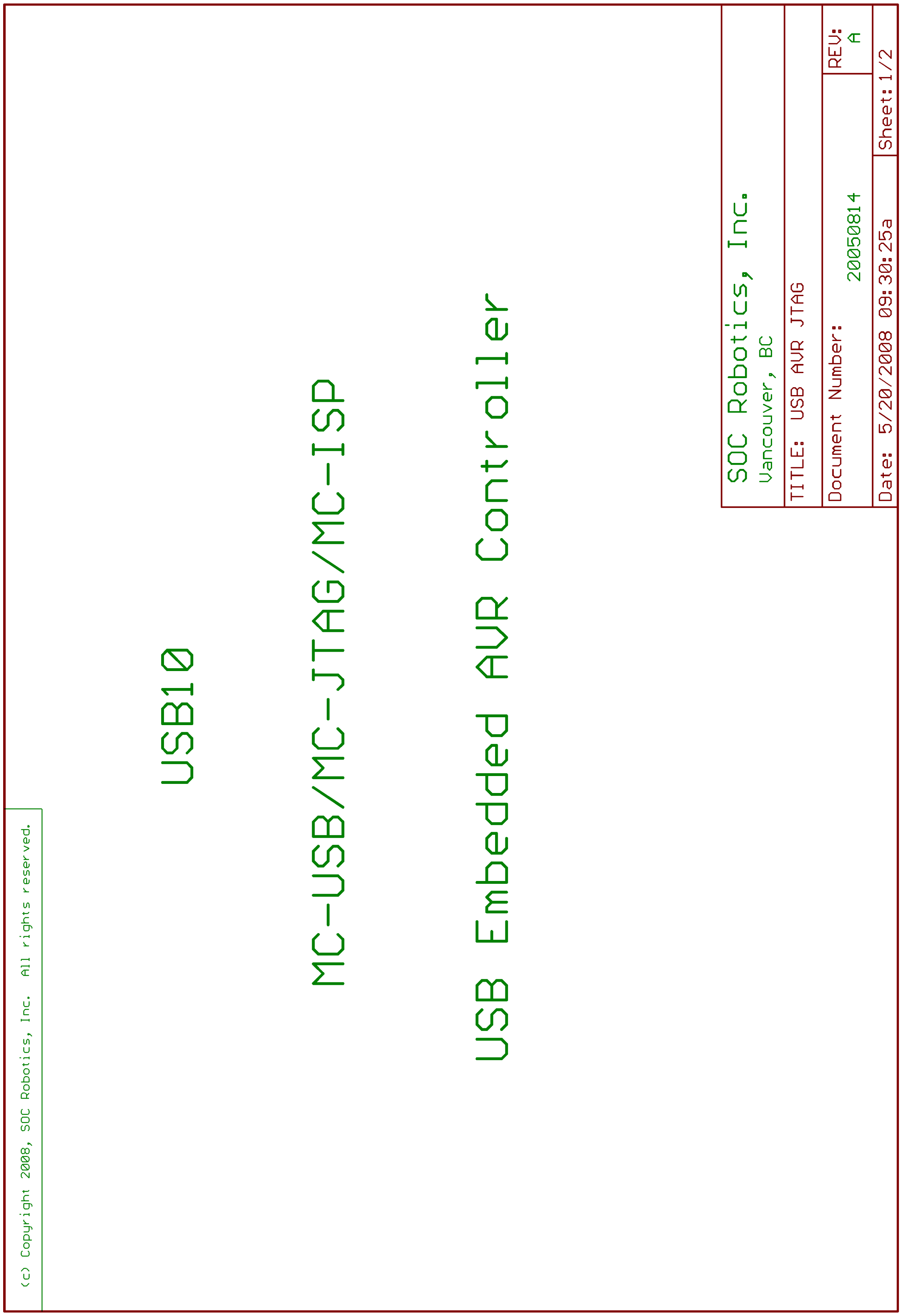
4.3 Mechanical Dimensions

Board dimensions are stated in inches. Connectors J1, J2 and J3 are positioned on 0.1″ pin spacing so the Wasp is easily mounted directly on any standard 0.1″ prototyping board.

A sample schematic with connector library and board layout in Eagle CAD format is available at **www.soc-machines.com/download/usb10layout.htm**.



5.0 USB10 Schematics



USB10

MC-USB/MC-JTAG/MC-ISP

USB Embedded AVR Controller

SOC Robotics, Inc.
Vancouver, BC

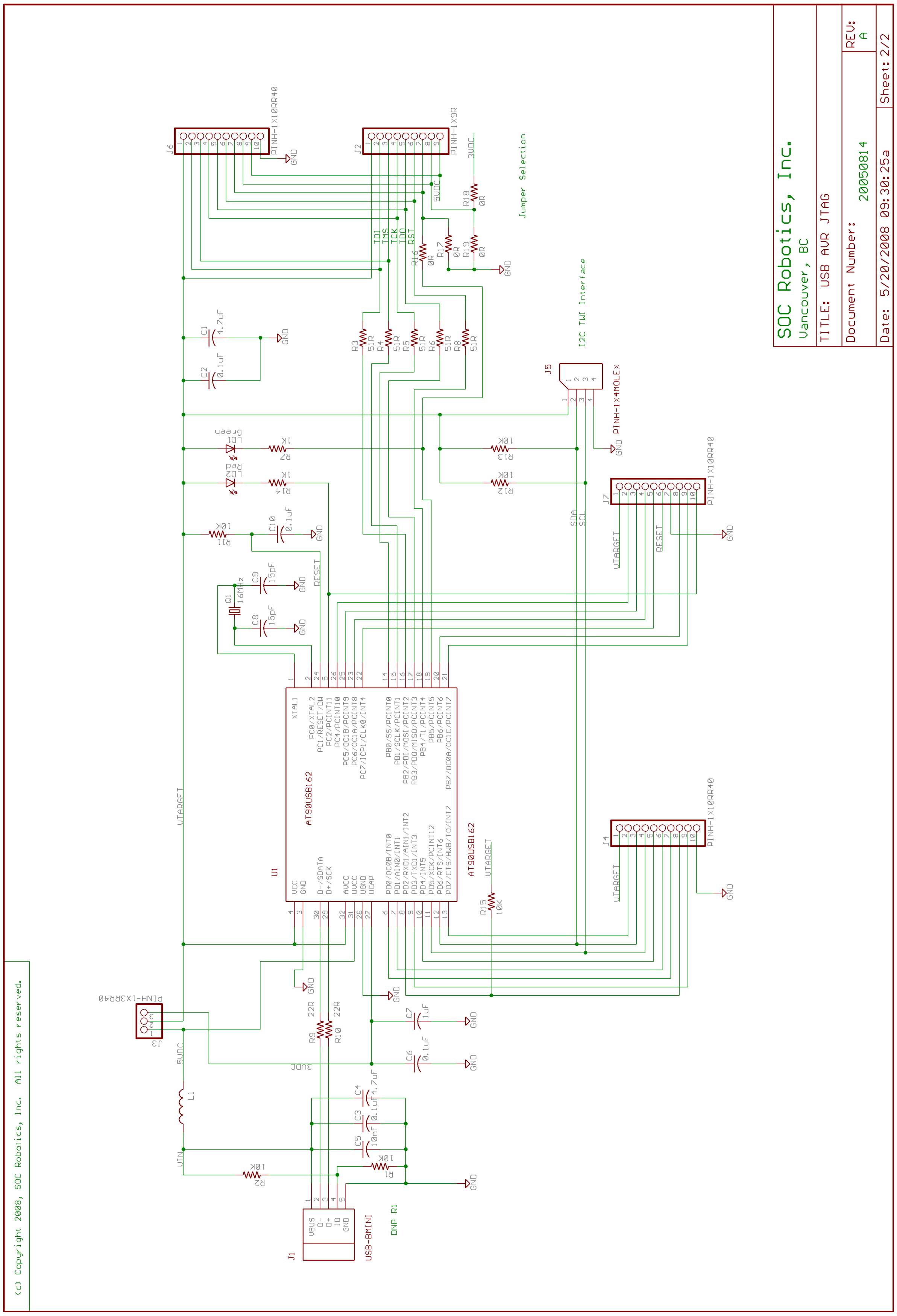
TITLE: USB AVR JTAG

Document Number: 20050814

REV: A

Date: 5/20/2008 09:30:25a

Sheet: 1/2


SOC Robotics, Inc.
Vancouver, BC
TITLE: USB AVR JTAG
Document Number: 20050814
REV: A
Date: 5/20/2008 09:30:25a
Sheet: 2/2
(c) Copyright 2008, SOC Robotics, Inc. All rights reserved.
Jumper Selection
I2C TWI Interface
AT90USB162
U1
J1 USB-BMINI
DNP R1
J2 PINH-1X9R
J3 PINH-1X3RR40
J4 PINH-1X10RR40
J5 PINH-1X4MOLEX
J6 PINH-1X10RR40
J7 PINH-1X10RR40
Q1 16MHz
LD1 Green
LD2 Red
VTARGET
RESET
5VDC
3VDC
VIN
GND
TDI
TMS
TCK
TDO
RST
SDA
SCL

Notes: