

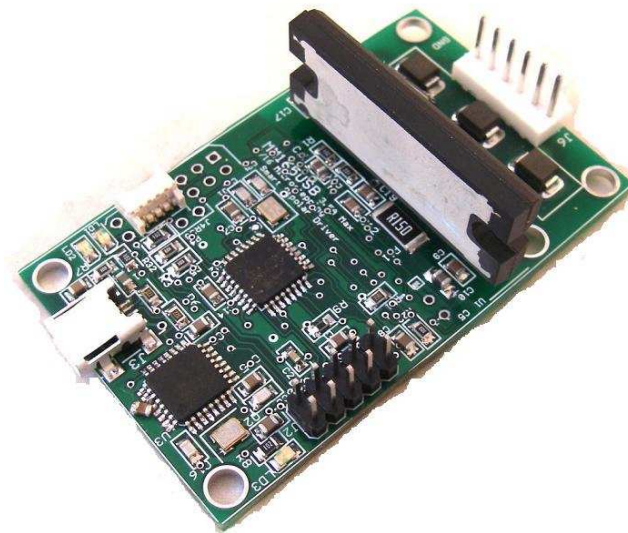
MM165U Smart Single Axis Stepper Motor Driver

1/16th Microstepping 3.5A Bipolar Driver

Technical Reference Manual

PCB Rev 1.0

Motor Control Software Version 1.00



www.soc-robotics.inc

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.

© Copyright 2011. SOC Robotics, Inc. All rights reserved.

SOC Robotics, Inc. makes no warranty for the use of its products, other than those expressly contained in the Company’s standard warranty which is detailed in SOC Robotics Terms and Conditions located on the Company’s web site. The Company assumes no responsibility for any errors which may appear in this document, reserves the right to change devices or specifications detailed herein at any time without notice, and does not make any commitment to update the information contained herein. No licenses to patents or other intellectual property of SOC Robotics are granted by the Company in connection with the sale of SOC Robotics products, expressly or by implication. SOC Robotics products are not authorized for use as critical components in life support devices or systems.

Pentium is a registered trademark of Intel Corporation.

Windows, Windows NT and Windows XP are registered trademarks of Microsoft Corporation.

Marks bearing ® and/or ™ are trademarks of SOC Robotics, Inc.

Terms and product names in this document may be trademarks of others.

1935A-08/00/5M

Table of Contents

Warranty Statement	2
1.0 Introduction	4
2.0 Detailed Description	9
2.1 Introduction.....	9
2.2 Processor.....	9
2.3 TWI Port.....	10
2.4 Auxiliary IO Port.....	10
2.5 ISP Programming Port.....	10
2.6 Molex Connector.....	11
2.7 Related Peripherals.....	11
2.8 Applications.....	12
3.0 Hardware Expansion Port Summary	13
3.1 Introduction.....	13
3.2 Motor Control Port.....	14
3.3 Auxiliary IO Port	14
3.4 TWI I2C Expansion Port	15
3.5 ISP Programming Port.....	15
4.0 Software Operation	16
4.1 Theory of Operation.....	16
4.2 Auxiliary IO.....	16
4.3 Step Drive Connector	16
4.4 Driver Commands.....	17
4.5 MM165U Command Overview	18
4.6 Detailed Command Description.....	21
5.0 Electrical and Mechanical Description	30
5.1 Component Layout.....	30
5.2 Electrical Specifications.....	30
5.3 Mechanical Dimensions	30
6.0 Circuit Schematics	31

1.0 Introduction

Features:

- Bipolar 3.5A Stepper Motor Driver
- Maximum motor drive voltage 35V
- Full, half, eighth and sixteenth microstepping
- Maximum 3.5 A per phase
- Backlash compensation built-in
- On board 20MHz RISC AVR processor – ATmega328
- USB 2.0 Interface for serial Communication and logic power
- G Code like commands supported
- Local joystick and limit switch support
- TWI I2C Communication Protocol Built-in
- 32K Flash1, 4K SRAM, 1K EEPROM
- Software controlled step mode, decay and torque setting
- TB6560 Bipolar Driver chip
- ISP Programming Port
- GNU C Compiler, Third Party Commercial C Compiler
- Extremely small form factor (2.54x1.54 in)
- 5VDC @ 12ma Power input for logic
- Motor drive input 10-40VDC at 3.5A



Hardware

The MM165U is a 1/16th Microstepping Unipolar 3A per phase stepper motor driver that converts step and direction signals to the appropriate high voltage stepper motor drive signals. The board is designed to operate at 5VDC. Motor voltage can vary from 10-40VDC. The motor drive chip is controlled by a dedicated onboard 8 bit RISC processor (ATmega328) that provides various expansion and enhancement features. Applications can communicate with the board via an I2C interface or using encoded commands on the step and direction inputs. A Limit and eStop switch inputs can be monitored and acted on but are turned off by default. An optional joystick can be connected directly to the driver to control stepper operation. A second processor (AT90USB162) provides a USB interface.

The MM165U consumes about 20ma in active state not including motor drive current. The motor driver is capable of 3.5A drive at up to 35VDC.

Motor Control Software

The MM165U is shipped with a sophisticated software application burned into the on board ATmega168 processor that controls the TB6560 bipolar driver chip, monitors step/direction inputs, limit switch state, encoded commands on the step/direction input lines and commands received on the I2C port. The MM165U can operate as a single axis G Code processing system by acting on commands sent to it via the I2C or Step/Direction input lines. See the section on software operation for a full description of the complete command structure.

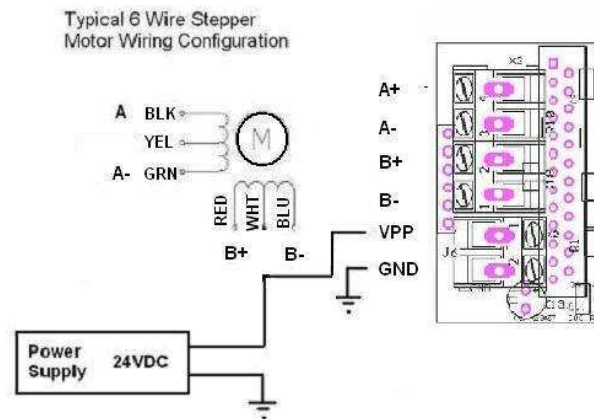
The control program in the MM165U processor can be re-Flashed using programming cable Part No. MCM-8 and the appropriate host software MK4Prog.exe.

The MM165U can be custom 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.

Stepper Motor Connection

Motor Drive voltage should be between 10-35VDC. A potentiometer sets average current and should be adjusted while the board is running. An optional heat sink is available to provide radiant cooling so forced air cooling is not required in most applications. Under extreme operating conditions forced air cooling should be used. The diagram below shows stepper motor connections for the SOC Robotics SM2006 and SM3006 stepper motors – wire color for other stepper motors may vary although most coil phase diagrams are similar. It is good practice to wire a fuse between the motor power supply and the MM165U. Do not turn motor power on if the logic side of the MM165U is not turned on – damage to the TB6560 driver chip may result. If the motor will not turn then check your wiring.

MM165 Motor Connection for SM2006/SM3006



Adjusting Motor Drive Current

The MM165U motor drive current is set by software in four increments. Default setting is 3.5A – the maximum setting. It is possible to set the motor current in such a way that the control chip delivers too much current to the motor resulting in motor overheating. If the motor missteps at high step rates then your motor maybe undersized for your application and can not develop enough torque – in this case use a bigger motor. Increasing drive voltage will increase maximum step rate. The maximum step rate of a stepper motor is determined by drive voltage, coil inductance and step mode. Changing step mode from full step to half step can reduce effective drive torque by 40%.

2.0 MM165U Detailed Description

2.1 Introduction

The MM165U is a compact single axis bipolar stepper motor microstepping driver with a dedicated onboard processor. The processor responds to step/direction signals on the primary control input port, receives and interprets commands received on the I2C interface or encoded on the step/direction signal lines and processes sensor data received on the auxiliary IO port.

MM165U Connector Pin Assignment

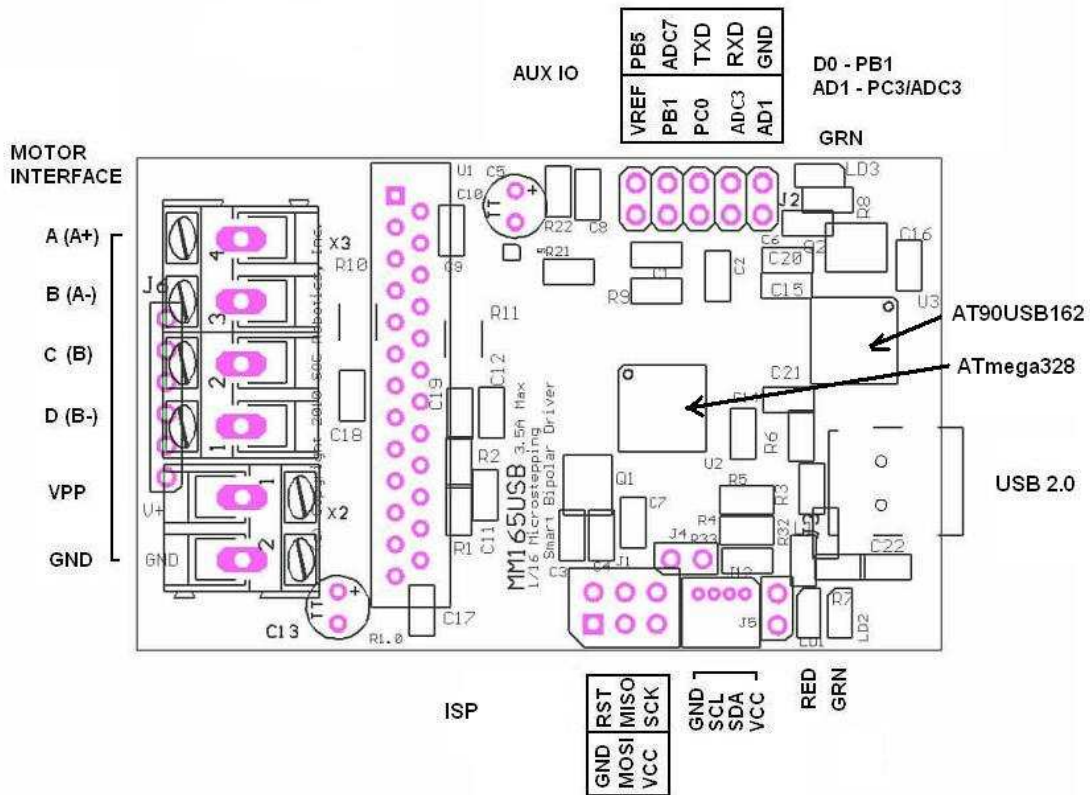


Figure 2-1. Primary Components on top side of PCB.

2.2 Processors

The MM165U has an 8bit RISC AVR processor (ATmega328) running at 20MHz. Note that the ATmega328 requires 5VDC to run at 20MHz. The program running in the Atmega328 monitors the step/direction input lines and drives the TB6560 microstepping driver chip. The list of commands supported by the processor firmware is described in the software operation section of this manual.

A second processor implements the USB serial CDC function using an AT90USB162 running at 16MHz. The serial port of the AT90USB162 is connected to the serial port of the ATmega328 providing virtual serial communications to the Windows desktop. A .inf file must be installed before a serial link is created on the desktop. The inf file is available for download from our web site along with the complete Ferret application as the USB10 Project File. Default baud rate is 38,400, no parity, 8 data bits and one stop bit. The USB processor is running an application called Ferret

2.3 TWI Port

The MM165U has a TWI I2C port for communicating with smart peripherals or other I2C peripherals. The TWI port uses a 4 pin Molex connector with power and ground so the MM165U can power other peripherals or be powered itself via this connector. Default I2C address is 0x14. I2C address is a configuration parameter and can be changed in software. The Molex connector is not installed but can be optionally added to the board.

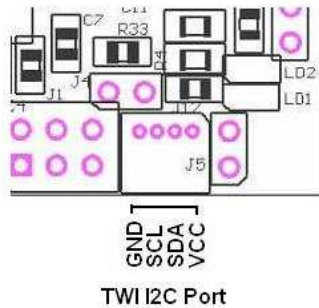


Figure 2-2. MM165 I2C port pin assignments.

2.4 Auxiliary IO Port

The MM165U has an auxiliary IO port. The auxiliary IO port supports three analog input lines and several digital IO lines. Version 1.0 software is capable of monitoring the position of an optional joystick attached to this port. If the center tap of a joystick is attached to pin ADC7 with one end of joystick potentiometer attached to GND and the other end to VREF the joystick can control the stepper driver directly. With the joystick at center position no movement occurs. Moving the joystick left or right causes the stepper motor to turn clockwise or counterclockwise. Speed is proportional to the degree by which the joystick off center position. By default joystick operation is disabled in software and must be enabled by the user. Maximum step rate and hysteresis (dead band) about the center point are programmable parameters.

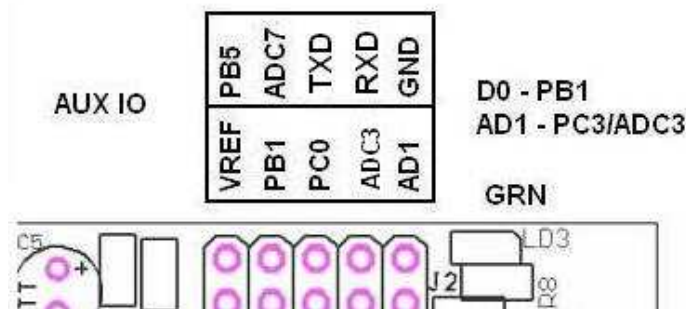


Figure 2-3. MM165U Auxiliary IO port pin assignments.

2.5 ISP Programming Port

The MM165U has two ISP programming ports – a 4 pin Molex connector and a 6 pin header. The MM165U’s Flash can be programmed by attaching a 4 line cable from the MK4 programming port to the Molex connector. The 6 pin header is not installed but can be added as an optional component. Alternatively the MM165U flash can be programmed using a separately supplied ISP10 Parallel Port Programmer and ISP610 adapter.

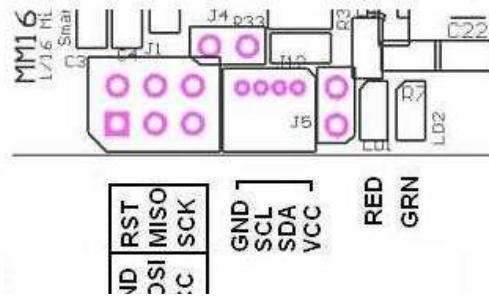


Figure 2-4. ISP610 ISP Adapter and correct attachment to the MM165U.

2.6 Molex Connectors

The MM165U uses two small Molex picoBlade 4 pin connectors with 1.25mm pin spacing (4 pin Molex Part No. 53048-0410 - Digikey Part No. WM1744-ND). These connectors mate with female Molex 4 housing connectors (4 pin housing Molex Part No. 51021-0400 - Digikey Part No. WM1722-ND).

The housing connectors crimp terminal accepts 26-28AWG wire (Molex Part No. 50079-8000 - Digikey Part No. WM1722-ND - Crimp tool 63811-0300) or 28-32AWG (Molex Part No. 50058-8000 - Digikey Part No. WM1775-ND - Crimp tool 63811-0200).

2.8 Applications

The MM165U is a bipolar stepper motor driver that responds to step and direction signal inputs but is also able to communicate with other smart controllers via the I2C interface. In fact, large numbers of MM165U’s can be ganged together and be controlled via a single I2C master. This enables the creation of rich motion control systems beyond simple three or four axis systems.

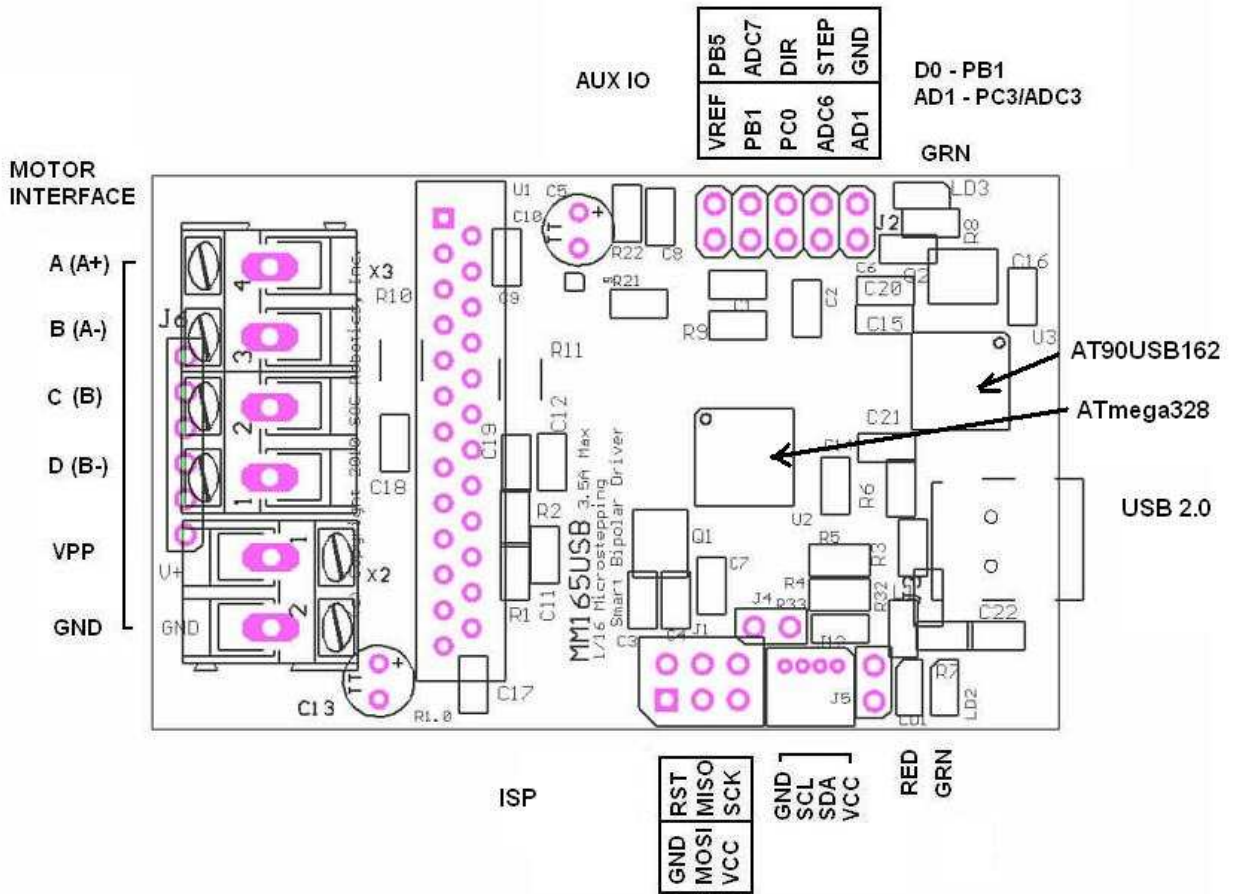
The MM165U also accepts sensor input on the Auxiliary IO and ISP programming lines providing an upgrade to a closed loop design in which sensor feedback is used to tune stepper motor operation. The example of a possible external sensor is a rotary magnetic sensor and a linear capacitive position sensor.

3.0 MM165U Hardware Expansion Port Summary

3.1 Introduction

The MM165U has three I/O ports: a motor control port, programming port and an auxiliary IO port as shown in the connector layout diagram below. Two alternative configurations of the board are available: Configuration 1 replaces the 0.1" motor control port header with an 8 pin picoBlade Molex connector and configuration 2 replaces the motor 0.1" motor control port header with a 4 pin picoBlade Molex connector for the I2C port.

MM165U Connector Pin Assignment



3.2 USB 2.0 Port

The MM165U has a USB 2.0 port for connecting the driver to the Windows desktop. When the MM165 is first connected to a Windows PC a driver request is made. The USB interface is implemented by a software application called Ferret running on the AT90USB162 processor on the board. Documentation for Ferret is provided in the USB10 Technical Reference Manual. Complete source code for the USB application and a Windows serial CDC driver is available for download from our web site as the USB10 MyUSB application. The .inf file is in folder "MyUSB 1.5.3/Demos/USBtoI2C" and is named "MyUSB USBtoSerial.inf".

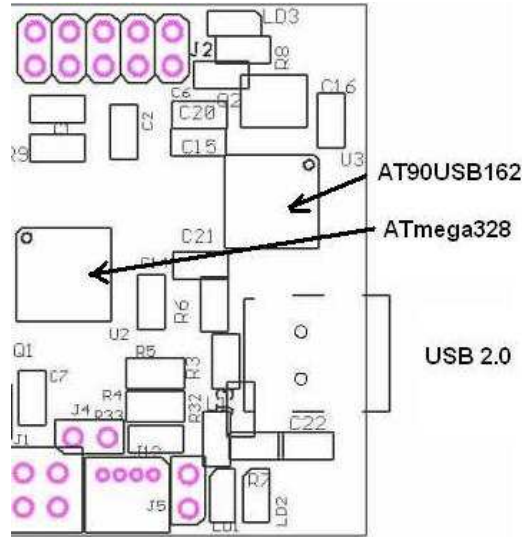


Figure 3-3. USB Port.

3.3 Auxiliary IO Port

The auxiliary IO port accepts analog input, digital IO and step/direction inputs. The digital IO can be used to connect external sensors such as rotary and linear position sensors or to allow the MM165U to control external devices such as relays. The analog inputs allow the MM165U to measure external analog sensors. Version 1.0 software supports optional joystick control on this port.

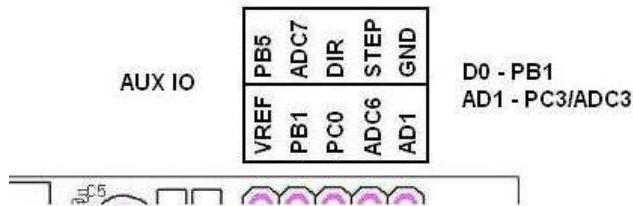


Figure 3-4. Expansion Port Pin Assignment J2.

If the center tap of a joystick is attached to pin ADC7 with one end of joystick potentiometer attached to GND and the other end to VREF the joystick can control the stepper driver directly. With the joystick at center position no movement occurs. Moving the joystick left or right causes the stepper motor to turn

clockwise or counterclockwise. Speed is proportional to the degree by which the joystick off center position. By default joystick operation is disabled in software and must be enabled by the user. Maximum step rate and hysteresis (dead band) about the center point are programmable parameters.

3.4 I2C Expansion Port

The MM165U has an I2C communication port on connector J3. The I2C connector is not installed on the standard MM165U but is available by special order. The MM165U can be controlled via the I2C port. See the section of Drive Commands for more information. Note that the I2C signals are also on the Motor Control Port connector. The MM165U is a slaved I2C device and can be controlled by other SOC Robotics Smart Peripherals such as joysticks, LCD display controllers data acquisition modules. Default I2C address is 0x14 Hex. This address can be changed via the I2C interface using special configuration commands.

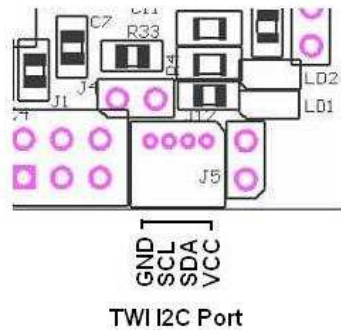


Figure 3-5. TWI I2C Port with 4 Pin Molex picoBlade Connector.

3.5 ISP Programming Port

The MM165U has two ISP Programming ports - a six pin header compatible with the Atmel 6 pin header ISP programming standard and a four pin picoBlade Molex connector compatible with the USB10U, MK4, MK14, MK54 and MK200 controllers. The ISP programming port allows the onboard ATmega168 processor Flash to be reprogrammed. See the Atmel ISP programming specification for detailed ATmega168 programming procedures.

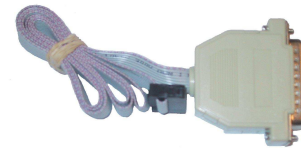
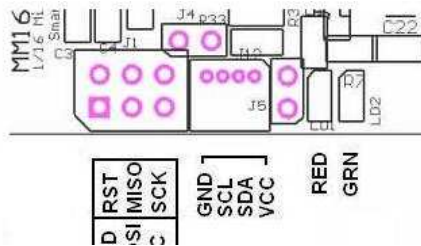


Figure 3-6. ISP Programming Port and MCM-8 programming cable.

4.0 Software Operation

4.1 Theory of Operation

The MM165U is controller by commands sent on the USB 2.0 serial interface or I2C port. The MM165U is shipped with half step mode enabled. The MM165U supports full, wave, one eighth and one sixteenth step modes. Step mode can be changed by sending a command to the board via the I2C interface or by sending commands on the serial USB 2.0 interface. Default USB serial interface is baud rate is 38,400, no parity, 8 data bits and one stop bit.

The USB interface is implemented by a software application called Ferret running on the AT90USB162 processor on the board. Documentation for Ferret is provided in the USB10 Technical Reference Manual. Complete source code for the USB application and a Windows serial CDC driver is available for download from our web site as the USB10 MyUSB application. The .inf file is in folder "MyUSB 1.5.3/Demos/USBtoI2C" and is named "MyUSB USBtoSerial.inf".

The motor drive chip TB6560PR is a bipolar microstepping driver with built in current sense resistors and logic to control drive current for a specific drive voltage. Motor torque is set in software – default is full torque. The TB6560 is designed to drive 2-3.5A motors but should be able to drive smaller motors without any trouble. The TB6560 has an over temperature Protection circuit that turns the driver off if die temperature exceeds a maximum value. Power must be cycled to re-enable the chip. The board power is applied through connector J10 and requires 5V regulated DC. The standard board runs at 20MHz.

4.2 Auxiliary IO

The MM165U has a number of auxiliary digital and analog inputs plus digital outputs that allow connection of rotary or linear position sensors and limit switches. The processor can be field programmed with new functions to process signals from these sensors and limit switches. An optional center tap joystick can be attached to the Auxiliary IO port.

4.3 Driver Commands

The MM165U accepts commands in two different communication formats: commands sent on the USB Serial Interface and commands sent through the I2C interface.

Command Features

Configuration parameters can be stored in on-chip EEPROM for automatic power up configuration control. A special built-in test/setup mode provides a simplified method for setting up the board. Configuration commands can be sent to the motor controller in real time using a special communications protocol based on the Step and Direction lines.

- New communication protocol for bidirectional data flow between the host and MM165U via the STEP/DIRECTION and Limit Switch lines and I2C
- The following parameters can be set by the user while the MM165U is operating:
 - o Pass through mode enabled for noisy environments
 - o Limit switch detection- enable, disable, smart enable, polarity settable
 - o EStop switch detection- enable, disable, polarity settable
 - o Step mode selection – wave, full, half, eighth, sixteenth
 - o Automatic motor shut off – enable, disable and time period settable
 - o Motor power - enable/disable
 - o Motor decay mode and torque setting
 - o Step rate, direction and number of steps to execute
 - o Maximum step rate

- Enable joystick following and set max pps and dead band at center tap
 - Distance traveled for each step in inch or metric
 - Feed Rate – ipm or cmpm
 - Linear or rotary distance commands in inch or metric
 - G1 G code commands accepted directly
 - Acceleration curve settable
 - Single Step - Clockwise or counterclockwise
 - Step Movement Direction Polarity – Clockwise or counterclockwise
 - Direction polarity settable – high – clockwise, high –counterclockwise
 - Backlash set
 - Communication mode – enable/disable
 - Save configuration settings in EEPROM
- Configuration parameters can be saved in EEPROM
 - Stored configuration parameters can be read from the motor EEPROM using MK4Prog.exe and used to create a text based configuration script file

I2C Commands

An alternative communication protocol is to send and receive commands via the I2C interface lines. I2C is a Philips two wire (SDA and SCL) communications protocol that is widely supported by both host processors and embedded processors. See related documentation for a detailed description of I2C. The I2C interface defaults to address 0x14 but can be changed by sending a broadcast set address command. Note that each MM165U on a single I2C chain must have a unique address.

Both communication protocols support a common command set. Commands are typically ASCII character sequences. There are three classes of commands – Setup, Drive and Global. Setup commands select various configuration options, Drive commands activate stepper motor operation while Global commands configure the communications address of the board and activate a special multiple MM165U's stepper motor operational state.

4.5 MM165U Command Overview

The MM165U processor responds to commands sent to it via the Step/Direction signal lines. Using a simple encoding procedure it is now possible to communicate with the Motor Controller via the Step, Direction and Limit switch lines. All commands are lower case letters and numbers. It is possible to change Motor Controller operation and configuration on the fly while the controller is running. Configuration changes take effect immediately.

A command consists of two or more ASCII characters. Some commands require entry of integer or floating point numbers. A floating point number entered as an integer is interpreted correctly. Integer or floating point number must be terminated with a semicolon ";" character. Note that all commands are lower case characters. Commands are acted on immediately.

On power up the MM165U processor loads configuration settings from on chip EEPROM. If these settings are changed using the commands below the processor does not automatically store the new settings in EEPROM – the user must explicitly do this using the Save Settings command. If power is lost all changed settings will be lost if they have not been stored.

Entering the ENTER key outputs a command prompt character "-". Entering "?" outputs a signon message and entering "p" outputs a list of current parameters. Other than these two commands all other commands are as described below and are the same for the MM165U and MM165. Note that commands can be entered either via the USB serial interface or I2C interface.

An example of a command sequence is:

```
-s8
-css
```

The first command instructs the driver to set the step mode to eighth step. The next command then stores this new setting in EEPROM so the next time the driver is powered up quarter step mode is activated. Some command functions can be halted by entering the character sequence `q`.

The MM165U supports the following commands. A detailed description of each command follows.

Command Summary

All commands are printable ASCII characters.

- s - Set Step Mode, 1-Full, 2-Half, 8-Eight, 6-Sixteenth `cx`s2
- q - Set Torque Mode, 1-100%, 2-75%, 3-50%, 4-20%
- y - Set Decay Mode, 1-0%, 2-25%, 3-50%, 4-100%
- c - Enter Configuration Change Mode:
 - s - Save settings to EEPROM (needs two 's') `css`
 - t - Pass through mode, a-active, i-inactive
 - r - Reset Motor drive chip
 - h - Turn motor power off
 - c - Command Recognition mode, a-active, i-inactive
 - n - Noise reduction mode, a-active, i-inactive
 - l - loop count for noise reduction - default 2, (int)
 - f - Step/direction signals recognition enabled, y=yes, n-no
 - i - I2C Mode o-on, f-off, a-address (int)
 - e - EStop switch recognition, y=yes, n-no
 - l - Limit switch recognition, y=yes, s-smart, n-no
 - j - Joystick mode recognition, y=yes, n-no
 - h - hysteresis, default 12, (int)
 - s - scale, default 80.0, (float)
 - p - Step polarity, p-positive, n-negative TBD
 - d - Direction polarity, c-clockwise, w-counterclockwise
 - a - Automatic motor power shut off mode, o-on, f-off
 - t - Time after which the motor power is shut off, seconds (float) `cat`25.5;
 - b - Set backlash (float) `cb`24;
 - x - Set acceleration parameter (float)
 - m - Step resolution parameters
 - r - Resolution (float), `cmr`0.000125;
 - n - Minimum pps (float), `cmn`20.0;
 - x - Maximum pps (float), `cmx`2300;
 - m - Ramp rate (float)
- d - Drive Motor Commands
 - z - Zero absolute position
 - g - Step continuously until told to stop
 - h - Go to Home position, n-near side, f-far side, o-offset (float)
 - i - Set feed rate (ipm), float `di`10.0;

- s - Set step rate (pps), float ds125;
- p - Pause execution (msec), int dp1000;
- c - Step clockwise
- w - Step counterclockwise
- l - Set slow step rate (pps), float
- f - Set fast step rate (pps), float
- m - Constant step dmw220000;1300;
direction, stepmode, number of steps, step rate (pps)
- x - Ramp for n loops using default low high pps dx220;2000;4;
stepmode, steps at each speed, constant step number, loops
- r - Ramp between low and high pps dr2300;2000;20;2000;
stepmode, slow step rate, high step rate, step increment, constant steps
- d - Distance mode dd1.25;6.27;
stepmode, new position, feed rate (ipm)
- t - Enter Test Mode
 - t - Toggle back and forth - all step modes, hit any key to exit
 - x - Ramp low to high - high to low - all step modes, hit any key to exit
 - v - Ramp low to high - high to low - enter step mode, hit any key to exit
- l - Led Mode Command, 1-mode1, 2-mode2, 3-mode3
- g - Return Data Command
 - b - Return controller busy status, b-busy, n-not busy
 - s - Return step mode
 - l - Return limit and estop switch enable state and value
 - a - Return auto motor shutoff setting and time
- z - Enter Record Command Mode
 - b - Begin recording
 - h - Halt recording
 - e - Execute stored commands
 - a - Arm pin PB5 (SCK) to trigger execution of stored commands , o-on, f-off zao

4.6 Detailed Command Description

Step Mode Command

Set the driver step mode. The MM165U is a microstepping unipolar stepper motor driver capable of full, half, eight and one sixteenth microstepping.

s - Change Step Mode, 1-Full, 2-Half, 8-Eight, 6-Sixteenth

Set the driver step mode. Step mode is set to full, half, eight or one sixteenth microstepping. The following characters set the step mode:

- 1 - Full step mode
- 2 - Half step mode
- 8 - One eight step mode
- 6 - One sixteenth step mode

Examples: s1 s6

Note that changing step mode may not guarantee a smooth step transition. Step mode changes take effect immediately. If making these changes while the board is running you should reset the motor controller.

Torque Mode Command

Set the TB6560 torque mode. The MM165 has four torque settings.

q - Set Torque Mode, 1-100%, 2-75%, 3-50%, 4-20%

Set torque to 100% (3.5A), 75% (2.6A), 50% (1.75A) or 20% (0.7A). The following characters set the step mode:

- 1 - 100% torque
- 2 - 75% torque
- 3 - 50% torque
- 4 - 20% torque

Examples: q1 q4

Decay Mode Command

Set the TB6560 driver decay mode. The MM165 has four decay mode settings.

y - Set Decay Mode, 1-0%, 2-25%, 3-50%, 4-100%

Set the decay to 0%, 24%, 50% or 100%. Typically the decay mode should be set to 100% when running the stepper at high speeds. The following characters set the decay mode.

- 1 - 0% decay
- 2 - 25% decay
- 3 - 50% decay
- 4 - 100% decay

Examples: y1 y3

Configuration Change Commands

The configuration change commands set the basic default operating modes of the MM165U. The user can turn command mode recognition on or off (if command recognition mode is turned off commands are no

longer recognized), save settings to EEPROM, enable a noise reduction mode, enable step/direction recognition, set EStop and limit switch recognition, change direction polarity, automatically turn holding torque on/off and time, set backlash and step resolution, minimum and maximum pps and ramp rate.

- c - Enter Configuration Change mode:
 - s - Save settings to EEPROM (needs two 's') css
 - t - Pass through mode, a-active, i-inactive
 - r - Reset Motor drive chip
 - h - Turn motor power off
 - c - Command Recognition mode, a-active, i-inactive
 - n - Noise reduction mode, a-active, i-inactive
 - l - loop count for noise reduction - default 2, (int)
 - f - Step/direction enable, y-yes, n-no
 - i - I2C Mode o-on, f-off, a-address (int)
 - e - EStop switch recognition, y-yes, n-no
 - l - Limit switch recognition, y-yes, s-smart, n-no
 - j - Joystick mode recognition, y-yes, n-no
 - h - hysteresis, default 12, (int)
 - s - scale, default 80.0, (float)
 - p - Step polarity, p-positive, n-negative TBD
 - d - Direction polarity, c-clockwise, w-counterclockwise
 - a - Automatic motor power shut off mode, o-on, f-off
 - t - Time after which the motor power is shut off, seconds (float) cat25.5;
 - b - Set backlash (float) cb24;
 - x - Set acceleration parameter (float)
 - m - Step resolution parameters
 - r - Resolution (float), cmr0.000125;
 - n - Minimum pps (float), cmn20.0;
 - x - Maximum pps (float), cmx2300;
 - m - Ramp rate (float)

c - Enter Configuration Change Mode

Top level command. Any characters entered after the "c" character are interpreted as configuration change commands.

s - Save Settings to EEPROM (needs two 's') css

Save all configuration parameters to EEPROM. Once saved in EEPROM the next power cycle of the board loads the new settings at power up. Note that if any configuration parameters are changed they remain in effect as long as the power is applied. Configuration changes are not automatically stored - the user must use this command to save any configuration changes. Note that two "s" characters are required.

t - Pass through mode, a-active, i-inactive

Set Pass Through Mode to active or inactive. Pass through mode is a special operating state of the MM165U that disables command recognition on both the I2C and Step/Direction lines. This mode when active ensures the step and direction signal levels are recognized under all conditions. If the setup and hold times for command recognition on the step/direction lines can not be guaranteed then the MM165U may accidentally enter command mode. This can also happen if there is excessive noise on the step/direction signal lines. Once Pass through mode is activated no commands are recognized - the only way to re-enable command mode is to re-flash the processor.

r - Reset Motor Drive Chip

Sets the motor drive chip to a known start state. Note that the motor rotor may be in an intermediate start position if any microstepping modes were selected.

h - Turn motor power off

This command turns motor power off. Motor power is reapplied on the next step input.

c - Command Recognition Mode, a-active, i-inactive

Command recognition mode is enabled or disabled by this command. Once command recognition mode has been disabled the only way to re-enable is to either re-powered to the board (if this setting was not saved) or re-program the processor. In noisy environments it may be necessary to disable command recognition mode so noise spikes on the step/direction lines are not mistakenly interpreted as step signals. By default command recognition mode is active. If a desktop application that is generating setup and holds times are not within specification then it may be necessary to turn Command Recognition mode off - note this applies to Commands encoded in the Step/Direction input lines only and not recognition of commands received on the I2C lines.

n - Noise Reduction Mode, a-active, i-inactive

l - loop count for noise reduction - default 2, (int)

In noisy environments a special noise reduction mode can be enabled. Noise reduction adds delay to the step pulse recognition logic. If step pulses are shorter than 5usec noise reduction mode should not be used. The loop count parameter sets the duration of the noise reduction period. By default noise reduction is disabled.

f - Step/Direction Enable, y-yes, n-no

Recognition of step/direction signals can be enabled or disabled. Step/direction signal recognition maybe disabled if step commands are encoded on the step/direction lines or I2C lines. By default Step/direction recognition is enabled.

i - I2C Mode o-on, f-off, a-address (int)

Commands received on the I2C signal lines can be enabled or disabled. By default I2C recognition is disabled. The default I2C address can also be set. By default I2C mode is disabled.

e - EStop Switch Recognition, y-yes, n-no

EStop switch detection can be enabled, disabled. When disabled the state of the EStop switch has no effect on motor operation. If enabled the motor will stop immediately and remained stopped as long as the EStop switch is active. Motor operation will resume if the EStop switch active state is removed or EStop Switch detection is disabled. EStop switch inputs are pulled high with a 10K resistor so to activate an eStop event the switch must pull the high level to ground. By default EStop recognition is disabled.

l - Limit Switch Recognition, y-yes, s-smart, n-no

Limit switch detection can be enabled, disabled or smart enabled. When disabled the state of the limit switch has no effect on motor operation. If enabled the motor will stop immediately if the Limit Switch becomes active (a low level) and the motor must be moved manually off the stop position. Alternatively if a limit switch event has occurred then by disabling limit switch recognition it is possible to move the stepper motor. If Smart mode is enabled the motor will respond to move commands in the direction away from the switch. Limit switch inputs are pulled high with a 10K resistor so to activate a limit switch event the switch must pull the high level to ground. By default Limit switch recognition is disabled.

j - Joystick mode recognition, y-yes, n-no

A center tap joystick potentiometer can be attached to the auxiliary IO port pin ADC7 with either ends of the potentiometer attached to GND and AREF. If joystick mode recognition is enabled then movement of the joystick off center position will cause the stepper motor to move clockwise or counterclockwise at a

rate set by the degree by which the joystick is off center. The potentiometer output swings from 2.5V at the center to GND or 5V. The joystick voltage output is converted to a digital value using the processors on chip A/D converter. The converted value ranges from 0 to 256. By default there is a hysteresis (dead band) at the center position to compensate for play in the potentiometer. This parameter can be set larger or smaller using the 'h' command followed by an integer position. The default hysteresis is set to 12 representing 5% of joystick range. A second scale factor parameter converts the joystick input to pulses per second by multiplying the joystick input by the scale factor. Assuming the center point is 128 with hysteresis of 12 once the joystick input reaches $128+12=140$ any increase over this will cause the stepper motor to move. A scale factor is applied to the joystick input to convert that number to a pps number. The default scale factor is 80. With a full range input of $256-128-12=116$ this give a maximum pps of $116*80=9280$ pps. Default step mode is quarter step so this results in a maximum full step rate of 2320. Both the hysteresis and scale factor can be change to suit the characteristics of the stepper used. Scale factor changes are made using the 's' command.

Example:

```
cjy      - Enable joystick following
cjh14    - Change joystick hysteresis to 14
cjs50.0; - Change scale factor to 50.0
```

p - Step Polarity, p-positive, n-negative TBD

Change the polarity of the step input signal. This function has not been implemented in the current version of the software. By default step polarity is an active going high pulse.

d - Direction Polarity, c-clockwise, w-counterclockwise

Change polarity of the direction input signal. Sets the direction the stepper motor turns when the direction signal is high. Note that depending on how the motor was wired clockwise and counterclockwise directions may be flipped. This command allows the user to flip the direction of rotation. By default the high level selects clockwise rotation.

a - Automatic Motor Power Shut Off Mode, o-on, f-off

t - Time after which the motor power is shut off, seconds (float)

After a step operation completes holding torque can be activated or deactivated. If activated torque the time hold torque is applied can be set. The time duration is set in 0.1 second increments from a 1/10 of a second to 6000 seconds. Depending on the application if holding torque is applied for long time periods driver and motor overheating may result.

Example:

```
cao      - Enable automatic power shut off
caf      - Disable automatic power shut off
cat1;    - Turn motor power off after 1 second
```

b - Set Backlash

If the backlash of the lead screw is known this command calibrates the absolute position command when direction changes. The backlash is the number of steps required to move the thrust nut in direction change operations.

Example: `cb24;` - Set backlash to 24 steps

x - Set acceleration parameter (float)

The acceleration parameter controls the rate of change of stepping. Stepper motors operate within a specific pull-in pull-out torque band. A stepper motor can not instantly change speed. The acceleration

parameter sets the maximum rate of change. The parameter is a floating point number between 1 and 25,000. 1 gives very slow acceleration while 25,000 is almost instantaneous.

Example: cx1000.0;

m - Step Resolution Parameters

r - Resolution (float), cmr0.000125 ;
 n - Minimum pps (float), cmn20.0 ;
 x - Maximum pps (float), cmx2300 ;
 m - Ramp rate (float)

The distanced traveled for a single step is used to calibrate the absolute position move command. Typically this parameter depends on the tpi of the lead screw and step mode. For example a 20tpi screw moves a carriage by 0.00025 inch in full step mode. The parameter is entered as a floating point number. The minimum pulse per second (pps) setting is used by the test and drive commands to set the slow step rate. The parameter is entered as a floating point number. The maximum pps setting is used by the test and drive commands to set the high step rate. The Ramp rate is a parameter that sets the acceleration for the test, drive and absolute position commands.

Drive Motor Commands

Enter drive motor command function. The drive mode allows step commands to be executed directly by the on board processor. Relative and absolute step position functions are supported. Most of the commands can be terminated early by entering the exit command: e

d - Drive motors (enter direction and step rate), e-exit (entered at any time)
 z - Zero absolute position
 g - Step continuously until told to stop
 h - Go to Home position, n-near side, f-far side, o-offset (float)
 i - Set feed rate (ipm), float
 s - Set step rate (pps), float
 p - Pause execution (msec), int
 c - Step clockwise
 w - Step counterclockwise
 l - Set slow step rate (pps), float
 f - Set fast step rate (pps), float
 m - Constant step dmw220000;1300 ;
 direction, stepmode, number of steps, step rate (pps)
 x - Ramp for n loops using default low high pps dx220;2000;4 ;
 stepmode, steps at each speed, constant step number, loops
 r - Ramp between low and high pps dr2300;2000;20;2000 ;
 stepmode, slow step rate, high step rate, step increment, constant steps
 d - Distance mode dd21.25;6.27 ;
 stepmode, new position, feed rate (ipm)

z - Zero Absolute Position

Set the internal position counter to zero. This command resets the home position. The absolute position command converts a distance command to the required number of steps based on the backlash and step resolution commands.

g - Drive Motor Continuously (c-clockwise, w-counterclockwise, s-stop)

Drive the motor continuously clockwise or counterclockwise until told to stop. The motor will step at a rate set by the pps command with a direction set by the character immediately after the "g". After sending the 'g' character send a 'c' step clockwise, 'w' to setp counterclockwise and 's' to stop stepping.

Example:

dgc	Step motor continuously clockwise
dgw	Step motor continuously counterclockwise
dgs	Stop stepping motor continuously

h - Go to Home Position

Move to the home position which is closest to the stepper motor with the limit switch activated. This command should be used in combination with the Limit Switch Smart setting. There are three parameters that set which end the home position is located at and an offset from the limit switch. The offset allows the home position to be located at any position between either end of the travel.

Example:

dhn	Home position located at near side
dhf	Home position located at far side
dho1.25;	Home position is offset by 1.25 in from limit switch

i - Set Feed Rate (ipm)

Set the feed rate in inch per minute (ipm). This command sets the step rate based on the step resolution. The parameter is entered as an integer or floating point number.

Example: di12.25; - Set feed rate to 12.25 inch per minute.

s - Set Step Rate (pps)

Set the step rate in pulses per second (pps). This command sets the step rate based on the number of steps per second. The parameter is entered as an integer or floating point number.

Example: ds125; - Set step rate to 125 steps per second.

p - Pause execution (msec)

Pause execution for a period of time. The time period is entered as an integer and is in mseconds. This command allows precisely timed delays to be added to stored programs.

Example: dp1000; Pause for 1 second.

c - Step clockwise

Step clockwise one step using the existing step mode. Note that if the desired step direction is not clockwise use the Step Direction Polarity command to change the direction.

w - Step counterclockwise

Step counterclockwise one step using the existing step mode. Note that if the desired step direction is not counterclockwise use the Step Direction Polarity command to change the direction.

s - Set slow step rate (pps)

Set the slow step rate in pulses per second (pps). This command sets the slow speed step rate that is used by several other commands such as the ramp command. The parameter is entered as an integer or floating point number.

Example: `ds1300;` - Set slow step rate to 1300 pps.

f - Set fast step rate (pps)

Set the fast step rate in pulses per second (pps). This command sets the slow speed step rate that is used by several other commands such as the ramp command. The parameter is entered as an integer or floating point number.

Example: `df1300;` - Set fast step rate to 1300 pps.

m - Constant step

Step at a constant rate (pps) given an entered direction, step mode and number of steps. This command allows a set of step parameters to be entered. Variable length parameters such as step rate may be entered as an integer or floating point number. Direction is "c" or "w". Step mode is 1, !, 2, @, 4, 8 or 6.

Entered parameters: direction, stepmode, number of steps, step rate (pps)

Example: `dmw220000;1300;` - Step CCW, half step, 20000 steps at 1300 pps
`dmc41000;400;` - Step CW, quarter step, 1000 steps at 400 pps

x - Ramp for n loops using default low high pps

Ramp from a low step rate to a high step rate, step at the maximum step rate for the entered number of steps and then step from the high step rate to the slow step rate stepping the entered number of steps at each speed and repeat for the entered number of loops. The slow and fast step rate are entered using the "s" and "f" drive commands. Note variable length command parameters are terminated with a ";" character.

Entered parameters: stepmode, steps at each speed, constant step number, loops

Example: `dx220;2000;4;` - Ramp using half step, 20 steps at each ramp
 increment, 2000 steps at max rate and loop
 4 times
`dx410;20000;14;` - Ramp using quarter step, 10 steps at each ramp
 increment, 20000 steps at max rate and loop
 14 times

r - Ramp Between Low and High pps

Ramp from a low step rate to a high step rate, step at the maximum step rate for the entered number of steps and then step from the high step rate to the slow step rate stepping the entered number of steps at each speed and repeat for the entered number of loops. The slow and fast step rate are entered using the "s" and "f" drive commands. Note variable length command parameters are terminated with a ";" character.

Entered parameters: stepmode, slow step rate, high step rate, step increment, constant steps

Example: `dr2300;2000;20;2000;` - Ramp using half step from 300 pps to
 2000 pps incrementing 20 steps per speed
 change and step 2000 steps at the maximum
 rate.
`dr1200;700;10;1000;` - Ramp using full step from 200 pps to
 700 pps incrementing 10 steps per speed
 change and step 1000 steps at the maximum
 rate.

d - Distance Mode

Move to an absolute position at the entered feed rate (ipm) given an entered step mode. The new position is entered as inches, step rate is inch per minute and step mode is 1, !, 2, @, 4, 8 or 6. At first power up the home is set to zero. Positive or negative absolute positions can be entered. Backlash compensation is applied automatically and step resolution defines the number of steps either clockwise or counterclockwise. The function automatically compensates for step mode changes so if a position command says move to absolute position 1.000 at full step and then step to 0.000 at half step the carriage will return the exact starting point.

Entered parameters: stepmode, new position, feed rate (ipm)

Example: dd21.25;6.27; - Ramp using half step from 300 pps to

Test Mode Commands

Enter test mode. Test mode commands drive the stepper motor through a series of direction toggle and ramp motions.

t - Enter Test mode

 t - Toggle back and forth - all step modes, hit any key to exit

 x - Ramp low to high - high to low - all step modes, hit any key to exit

 v - Ramp low to high - high to low - enter step mode, hit any key to exit

t - Enter Test Mode

Enter the test mode state and interpret any subsequent characters as test mode selections.

t - Toggle Back and Forth - All step modes, hit any key to exit

This command causes the drive to step the motor in the clockwise direction for a defined number of steps and then step in a counterclockwise direction the same number of steps at a pre-defined pps rate starting with full step mode and sequencing through all the step modes. Entering cxq caused the sequence to terminate.

x - Ramp Low to High/High to Low - All step modes, hit any key to exit

This command causes the driver to step the motor through a clockwise then counterclockwise motion while ramping the speed from a minimum pps rate to a maximum pps rate sequencing through all step modes from full to sixteenth step.

v - Ramp Low to High/High to Low - Hit any key to exit

This command causes the driver to step the motor through a clockwise then counterclockwise motion while ramping the speed from a minimum pps rate to a maximum pps rate at an entered step mode and loop count.

Entered Parameters: step mode, loop count

Example: tv210 - Ramp in half step mode and repeat 10 times.

LED Display Mode Command

The user can set the display mode of the LEDs to one of three different display status modes. This function has not been implemented in the current software version. Mode 1 is active.

1 - Led mode, 1-mode1, 2-mode2, 3-mode3

Set the led display mode. Command character is "l". Sets mode 1, 2 or 3.

Mode 1 (Default mode in command recognition mode)

Green led displays status of stepper motor power: on - power being supplied to motor, off - no power to motor. Red led displays step rate and blinks when driver is waiting for a step signal- led blinks for each step signal sent to the motor. The slow blinking Red led indicates the driver processor is alive and waiting for a step command.

Mode 2 (Default mode if pass through mode active)

Green led displays status of stepper motor power: on - power being supplied to motor, off - no power to motor. Red led display step direction - on - clockwise, off - counterclockwise.

Mode 3

The on or off condition of the Green and Red LEDs is set using commands. The next character selects the Red or Green Led and the state (on or off).

Entered Parameters: led, state (on-o, off-f)

Examples:

lro - Turn the Red Led on

lgf - Turn Green Led off.

Return Command Status (I2C only)

The MM165U can return the status of several parameters and the busy status of the processor. The MM165U may take a while to execute certain commands. The return status command indicates the status of the processor as either busy or not busy. If commands are sent to the MM165U without regard for the time it takes to execute commands then a command overrun condition can occur. If a sequence of commands must be sent to the device then the gb command provides a method to meter the flow of commands so a command overrun condition does not occur. Note that the return data commands are implemented on the I2C interface only and do not return any data on the step/direction lines. A future version may allow this.

g - Return data command

b - Return controller busy status, b-busy, n-not busy

s - Return step mode

l - Return limit and eStop switch inputs

a - Return auto motor shutoff setting and time

b - Return controller busy status, b-busy, n-not busy

Return the driver busy status. If the MM165U is busy executing a command or function then the get busy command will return a 'b' - if the driver is not busy then 'n' is returned.

Example: gb returns 'b' or 'n'

s - Return step mode

Return the driver step mode. Step mode is set to full, half, eight or one sixteenth microstepping. The following characters are returned to indicate the current set step mode:

- 1 - Full step mode
- 2 - Half step mode
- 8 - One eighth step mode
- 6 - One sixteenth step mode

A single byte is returned set to one of the characters above.

Examples: `gs` returns '1', '2', '8' or '6'

l - Return status of limit and estop switch enable state and level

Return the status of the limit switch and eStop switch state and current level. Four bytes are returned. The state of Limit switch checking is returned as either 'y' for enabled, 'n' for disabled and 's' for smart mode. EStop switch state checking is return as 'y' for enabled or 'n' for disabled. The level is returned a 'h' for high and 'l' for low. Four bytes are returned. The first two are the setting of the limit switch state (enabled, disabled or smart) followed by the level on the limit switch input (high or low). The next two bytes are the setting of the eStop state (enabled or disabled) followed by the level (high or low).

Examples: `gl` returns 'yhyh' if both limit and estop switch checking is enabled and not activated

a - Return auto motor shutoff setting and time

Return the status of the auto motor power shutoff parameter and the time associated with auto motor shutoff. Three bytes are returned. The first byte contains the auto motor shut off parameter that is either 'o' for on or 'f' for off. The turn off time is a two byte binary number in seconds.

Examples: `ga` returns 'o' or 'f' and a two byte binary number

Record Commands Mode

The MM165U processor has approximately 1900 bytes of EEPROM storage space that can be used to record canned command sequences. Commands to be recorded are any valid commands sent to the MM120 via the I2C or Step/Direction lines. For example, if the stepper motor needs to move a platform 1 inch left, pause for five seconds then move back 1 inch the following procedure accomplishes that. First start recording mode by sending the 'b' command. Now send the sequence of commands to execute the desired motion. At the completion of the motion send the 'h' command to halt recording. The motion sequence is not stored in the control processors EEPROM. To execute the stored motion send the 'e' command or pull the Aux IO Port pin PB5 low. Only one stored command sequence can be loaded into EEPROM and the length is limited to 1900 characters. The PB5 pin is pulled high by a 10K resistor so an additional pull up resistor is not required. A simple normally open switch is all that is necessary to pull PB5 low. Note that PB5 pin recognition must be enabled in order to have any effect.

z - Enter record command mode

- b - Begin recording
- h - Halt recording
- e - Execute stored commands
- a - Arm pin PB5 (SCK) to trigger execution of stored commands , o-on, f-off

Example:

`zb`

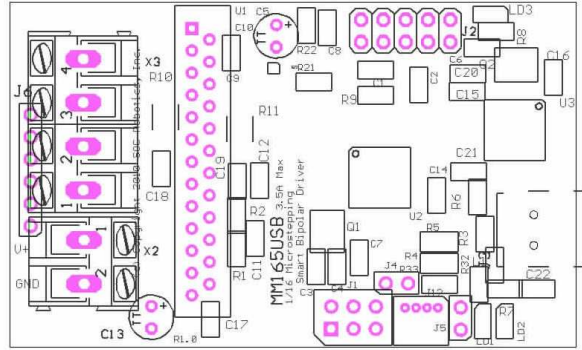
```
dd21.00;5.45;  
dd21.25;6.57;  
dp2000;  
dd21.00;6.57;  
dd20.0;5.45;  
zh  
ze
```

In the example above the record command mode is entered followed by a movement of the carriage from position 0 to position 1.00 at 5.45 inch/minute, then the carriage is instructed to move to 1.25 at 6.57ipm followed by a pause for 2 seconds then it moves back to 1.00 at the same rate and finally returns to the starting position 0.00. Recording mode ends when the halt recording command is entered. The complete motion sequence is then executed by the start execution command or pulling PB5 (pin 2) low.

5.0 Electrical and Mechanical Description

5.1 Component Layout

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



5.2 Electrical Specifications

Electrical

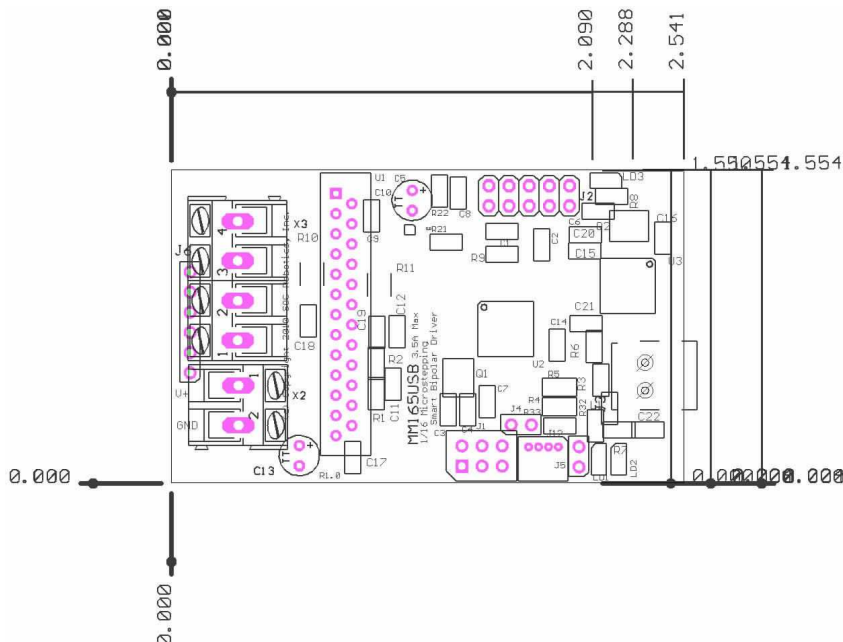
Input power: 5VDC @ 20ma logic
12-30VDC 3.5A for stepper driver

Mechanical

Dimensions: 2.51x1.58 in (one mounting hole)
Weight: 38 grams

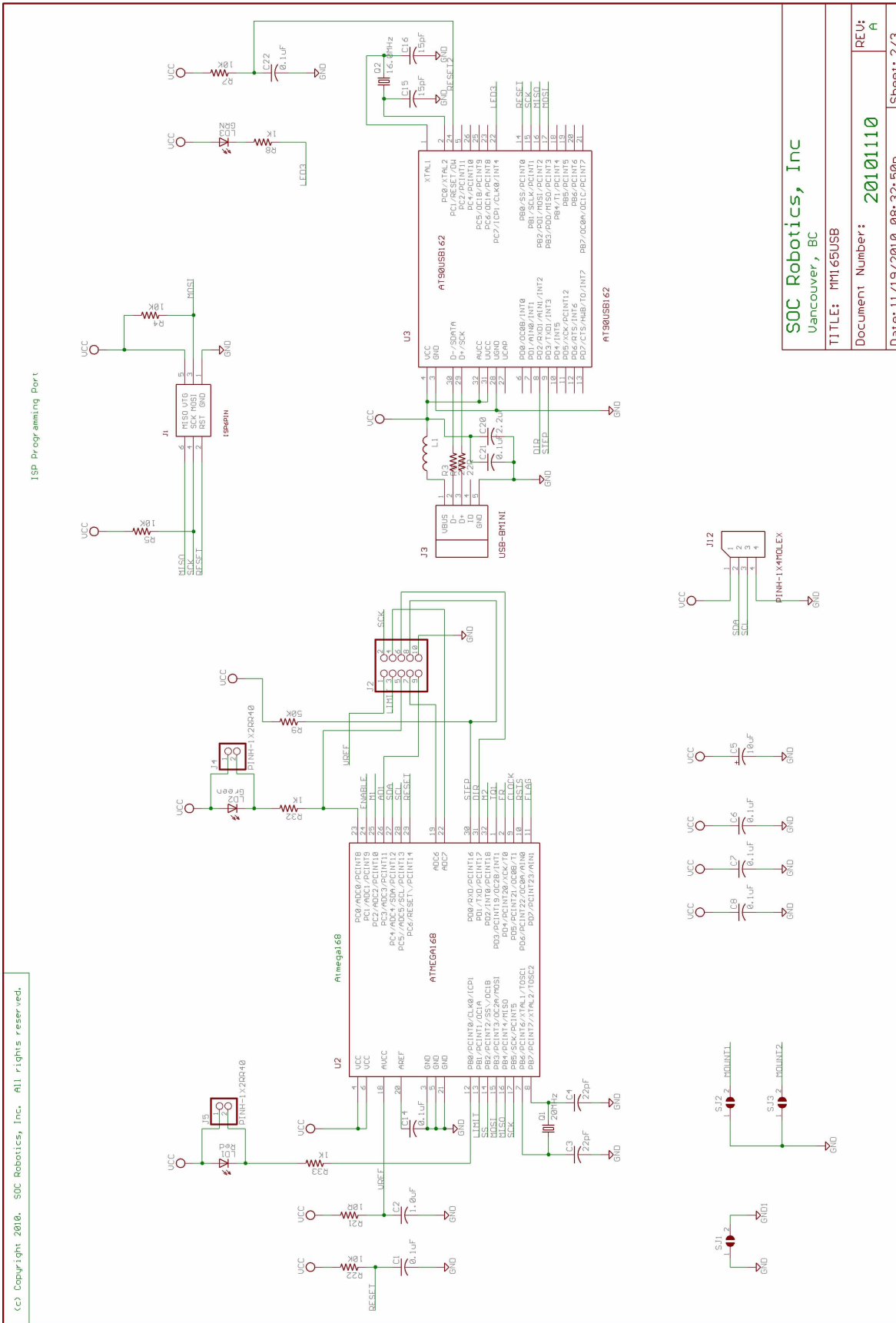
5.3 Mechanical Dimensions

Board dimensions are stated in inches.

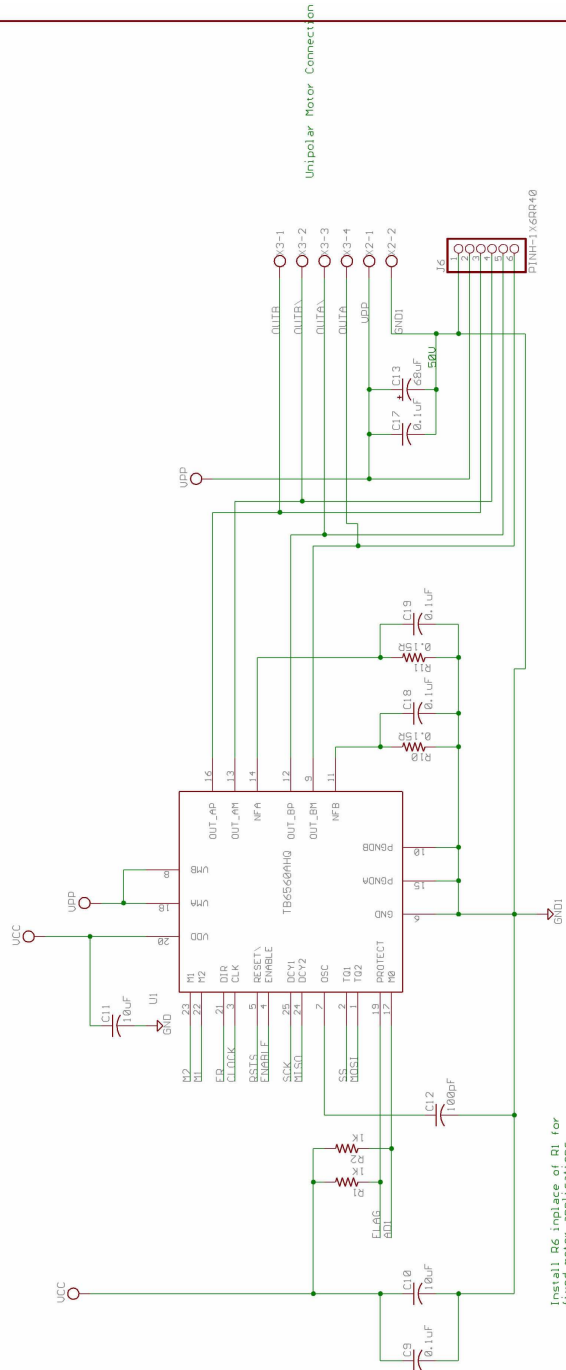


6.0 MM165U Circuit Schematics

<p>(c) Copyright 2010, SOC Robotics, Inc. All rights reserved.</p> <p>MM165USB 1-Axis Smart</p> <p>Bipolar Stepper Motor Driver 3.5A</p> <p>PCB Rev 1.0</p>		<p>SOC Robotics, Inc Vancouver, BC</p>	
<p>TITLE: MM165USB</p>		<p>REU: A</p>	
<p>Document Number: 20101110</p>		<p>Sheet: 1/3</p>	
<p>Date: 11/19/2010 08:32:50p</p>			



(c) Copyright 2010. SOC Robotics, Inc. All rights reserved.



SOC Robotics, Inc
 Vancouver, BC
 TITLE: MM165USB
 Document Number: 20101110
 Date: 11/19/2010 08:32:50p
 REU: A
 Sheet: 3/3



Notes: