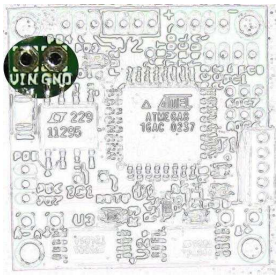
## *Introduction*

This document covers the programming and usage of the MEGAbitty controller board. More information and support on the MEGAbitty board family can be obtained from http://groups.yahoo.com/group/megabitty/. If your MEGAbitty controller board is still in pieces, refer to the "MEGAbitty Assembly Instructions" available in the "files" section of the above site.

## *Connections*

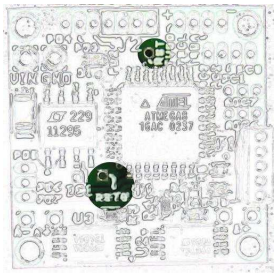
The MEGAbitty controller is designed to work with a 6V-12V DC power supply to provide two 500mA motor driver ports and 700mA of regulated 5V power for the microcontroller and peripherals. An Atmel MEGA8 AVR microcontroller provides 16MIPS of processing power, an in-system programming (ISP) port, a TTL-level serial port, a TWI (I2C compatible) port, 8 A2D inputs, an analog comparator input, two external interrupts, and many general purpose I/O connections. Two of the A2D inputs are pinned out for easy connection to Sharp distance sensors, such as the Sharp GP2D12.

### Power



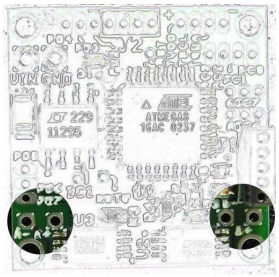

A single power connection ("Vin" & "GND") in the upper left corner of the board supplies both the motor drivers and the 5V regulator. Input voltages of 6-12V are acceptable, but keep in mind that the full applied voltage (minus a small voltage-drop across the H-Bridge) will be applied to the motors. A 5V low-dropout voltage regulator supplies up to 700mA (with proper heat-sinking) to the Mega8 and attached peripherals.

### Reset



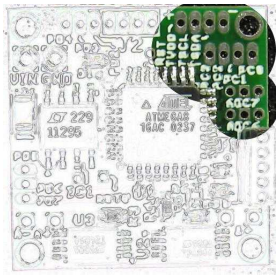

A normally-open push-button reset switch can optionally be connected between the "RST" pad directly above the Mega8 (U1) and the "RSTG" pad at the lower left corner of the Mega8. If a small, low-profile pushbutton switch is used, it can be mounted on top of the Mega8 with a dab of glue or double-sided tape. A couple switches appropriate for this are listed in the "MEGAbitty Assembly Instructions".

### Motor outputs



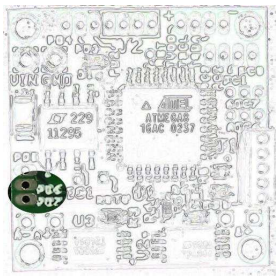

Two motor driver outputs, A+/A- & B+/B-, can deliver up to 500mA (with proper heat-sinking) to each of two DC motors. An H-bridge powers each motor output using locked anti-phase PWM at frequencies of up to 200kHz. The H-bridges are powered directly via MEGAbitty's unregulated input power, "Vin", so motor voltages of up to 12V (minus H-bridge overhead) are obtainable.

### ADCs



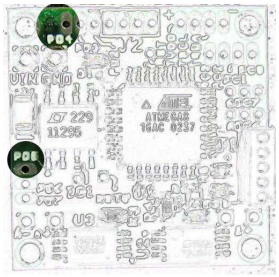

All eight of the Mega8's analog-to-digital converter inputs are accessible on the MEGAbitty board. Two of these, ADC6 and ADC7, are pinned out with a dedicated +5V and GND pad for Sharp distance sensor modules, such as the GP2D12. Four other ADC inputs are brought out to dedicated pads, PC0-3, located in the upper right corner. The remaining two share the TSCL and TSDA pads with the TWI bus (also in the upper right corner.)

## Differential analog comparator



Two analog voltages may be compared against each other via the Mega8's analog comparator, which is accessible via the PD6 and PD7 pads located in the lower left corner of the board.

## Timer/Counters

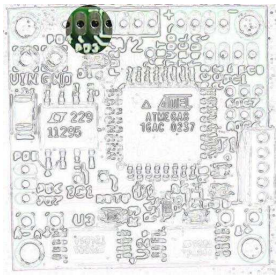


The Mega8 features an 8bit counter (Timer/Counter 0) and a 16bit counter (Timer/Counter1) that can be used to count or time external events on the PD4 and PD5 lines, respectively. These lines are brought out to pads of the same name, with PD4 located in the upper left corner of the board, and PD5 in the lower left corner. Note that Timer/Counter 1 is normally used for PWM control of the motor outputs, and so will not typically be available for event counting.

The 16bit Timer/Counter 1 is used for PWM control of MEGAbitty's two motor drivers via the Mega8's "OC1A" and "OC1B" pins. These pins are directly connected to the motor A and motor B H-Bridge's PWM inputs.

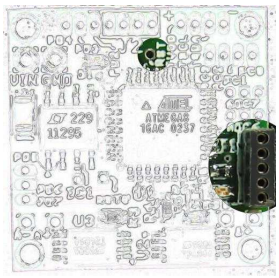
The 8bit Timer/Counter 2 cannot count external events, but it can be used for PWM control of a device connected to "MOSI" pin of the ISP connector (J1). Note that J1 contains a "+5" and "GND" pin next to "MOSI" to simplify connecting to an external device, such as a servo.

## External Interrupts



Two external interrupts are available via the pads PD2 & PD3 located in the upper left corner of the board. These interrupts may be programmed to be low-level sensitive, rising edge sensitive, falling edge sensitive, or sensitive to either edge.

## ISP



The In-System Programming (ISP) connector (J1) must be used to program the Mega8 until a bootloader is installed, and any time fuse bits need to be changed. MEGAbitty is intended to be used with a bootloader, which allows re-programming over the serial bus (J2). Use of the ISP bus as the primary means of programming is discouraged because of a few difficulties due to sacrifices made to keep the board small.

The first difficulty is that the "Reset" line is not included in the ISP connector. When connecting to J1, a separate jumper wire needs to be connected to the "RST" pad. The second difficulty is that the B motor-enable line is shared with the ISP SCLK line. This shouldn't be a problem for ISP programming, but can cause the B motor to twitch while programming. If the ISP bus is to be used heavily, consider modding the board to use the motor A enable line to enable both A and B motors, and cut the B motor enable line at the connection to the J1 SCLK pad. This frees up the SCLK line for dedicated ISP usage and eliminates motor twitching.

Care needs to be taken in powering the board when using the ISP bus. The recommended approach is to power the MEGAbitty board through the normal power connection ("VIN" & "GND"), and let

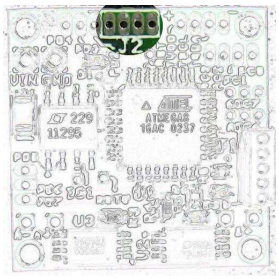
MEGAbitty power the programmer. If the programmer cannot be configured to be powered by the target (MEGAbitty), then the "+" pin on J1 should not be connected to the programmer.

If MEGAbitty *is* powered from the programmer (not recommended), the voltage regulator must be bypassed by temporarily jumpering between the regulated side, "+5V", and the unregulated side "VIN". This is necessary to insure the H-Bridges are powered. If this is not done, the H-Bridge control inputs will see a higher voltage than the H-Bridge supply, which could damage the H-Bridges. Just be sure to remove the jumper before powering up MEGAbitty via the normal power connector!

Under no circumstances should MEGAbitty and the programmer be allowed to power each other simultaneously. Doing so will likely overload both power supply and could cause damage. If Atmel's STK500 development board is used as the programmer, it defaults to powering the target, but may be configured to derive power from the target by removing the "Vtarget" jumper. See the STK500 manual for more information.

See the "MEGAbitty Programming" section for more information on using the ISP port for Mega8 programming.

## Serial

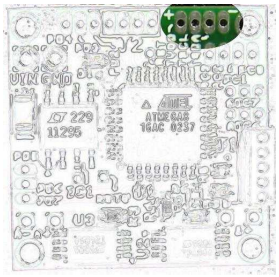


MEGAbitty's TTL-level USART signals "Rx" and "Tx" are connected to serial port header J2 in the upper left corner of the board. J2 also includes +5V and GND connections. The Serial port is intended to be used as the primary programming interface for the Mega8. Once a bootloader is installed via the ISP interface (and the appropriate fuse bits are set), MEGAbitty may be reprogrammed via a computer's standard serial port. To enable this, a RS232 level translator is required to translate the Mega8's TTL level I/O to the higher-voltage RS232 signaling levels. A simple circuit using a MAX232 IC and four caps can easily be made, or an inexpensive kit or pre-built unit can be purchased from the vendors listed in Appendix B.

Attempting to hook MEGAbitty's "Rx" and "Tx" lines directly to a computer's serial port without a level translator will likely damage the Mega8 and possibly the computer.

See the "MEGAbitty Programming" section for more information on Mega8 programming using the serial port.

## TWI



Atmel's Two-Wire Interface (TWI) provides an $I^2C$ like interface for communicating with other processors (other MEGAbittys) and $I^2C$ peripherals. The TWI bus is accessible via the four pads in the upper right corner of the MEGAbitty board: "+5V", "TSCL", "TSDA", & "GND".

## SPI

While the Mega8 supports the SPI bus, it conflicts with the PWM output used for controlling motor B. Since MEGAbitty is targeted at a robot controller with two motor drivers, SPI bus functionality has been sacrificed. The SCLK and SS_n lines of the SPI bus are

shared with the B motor enable and PWM lines, respectively. This makes simultaneous SPI and motor operation difficult. If SPI operation is desired, a couple mods can be made, but will limit motor usability. First, the SCLK line needs to be separated from the motor B enable. To do this, cut the *bottom* layer trace connected to J1's SCLK pad. The pull-up resistor R4 will insure that motor B remains disabled. The second mod, and the more cumbersome one, is to tap off the B motor PWM line for a SS connection. Unfortunately, there are no vias on this line or pads other than at the Mega8 and the H-Bridge. Other than the SS_n line, the SPI bus is accessible via J1 – the same connector used by the ISP bus.

## *MEGAbitty Programming*

MEGAbitty is intended to be programmed via a serial port using a bootloader resident on the Mega8. If you bought a pre-stuffed board then a bootloader has been preinstalled, so, skip ahead to the "Downloading User Code" section. If you assembled MEGAbitty from a kit, then a few steps need to be completed first:

- Enable the external 16MHz oscillator
- Redirect the reset vector to the boot loader area of memory
- Load the bootloader code

The first two steps require setting fuse bits in the Mega8, and the final step requires downloading a hex-format code file. All steps require MEGAbitty to be connected to a programmer via the ISP connector. Programmers can be bought for ~$30 or made relatively easily and cheaply. The ISP cable needs to be made using the 5-pin 0.05" header included in the MEGAbitty kit. See Appendix A for cable construction programmer sources. Atmel's AVR Studio is used as the programming software, and is available for free from Atmel's web site:
http://www.atmel.com/dyn/products/tools_card.asp?tool_id=2725

### Fuse bit settings

The Mega8 first powers up using its internal 8MHz oscillator. A few fuse bits need to be set to switch to the external 16Mhz ceramic resonator:

- CKOPT=1
- CKSEL=1111
- SUT=00

Follow the steps below carefully! Programming the fusebits wrong can render your Mega8 useless! A common slip-up is to program the clock settings wrong. Doing so can leave the chip without a heartbeat and prevent further programming. See Appendix XX (yet to be written) for performing Mega8 CPR.

The following steps target AVR Studio4 and the STK500 as the programmer. Steps for the AVRISP programmer should be similar.

1. Open AVR Studio and hit the "Cancel" button to escape out of the project-selection dialog box. For generating .hex files from assembly code, or for simulating code, you'll want to create a project. For our purposes, however, we don't need one.
2. Connect MEGAbitty to the programmer via the ISP cable and power up the STK500. Make sure MEGAbitty's red LED is not on. If it is, then power off the STK500 and remove the "Vtarget" jumper. It is important that MEGAbitty has its own power source and be allowed to power the programmer. See the "Connections/ISP" section above for details. If the red LED is not on, then power up MEGAbitty. Now you should see MEGAbitty's red light come on.
3. Open the programming tool by selecting "Tools->STK500/AVRISP/JTAG ICE…" It should pop up a "STK500" window with a status box at the bottom. Make sure it was able to detect the STK500.
4. Under the "Program" tab, set the following options:
   - ✓ Select "ATmega8" from the "Device" list.

- ✓ Set the "Programming mode" to "ISP"

5. Select the "Fuses" tab. In the status box, make sure it was able to read the fuse bits.
6. Checkmark the following boxes:
   - ✓ "Boot Flash section size = 1024 words Boot start address=$0C00;"
     (Note that the MegaLoad bootloader will actually fit in 512 words but choosing a larger size reserves room for future upgrades.)
   - ✓ "Boot Reset vector Enabled (default address=$0000);"
     (If you plan to not use a bootloader and only use the ISP cable for programming, then leave this box unchecked. Only recommended for experts!)
   - ✓ "Ext. Crystal/Resonator High Freq.; Start-up time: 1K CK + 64 ms;"
     (This is the important one. Make sure the description matches exactly! Many of the clock options sound similar.)
7. Make sure the following boxes are **NOT** checked:
   - "Reset Disabled (Enable PC6 as i/o pin);"
     (checking this box will disable further programming via ISP! The only way to recover is to use "Parallel Programming Mode" which isn't possible on MEGAbitty.)
   - "CKOPT fuse (operation dependent of CKSEL fuses);"
8. **Double check** that the appropriate boxes are checked and the dangerous boxes are not checked.
9. Hit the "Program" button to set the fuses. If you watch the blue and green LED's (D3 & D4) on MEGAbitty, they should flash briefly during programming.
10. Check the status box to make sure the programming succeeded.
11. Check that the fuses verified ok – if there is not a message in the status box about verifying the fuses, click the "Verify" button.

If all went well, MEGAbitty should be blazing along at 16MHz now, but doing absolutely nothing. The next step is to give it something to do.

## Installing a bootloader

A bootloader is a small program that resides in the Mega8's flash and allows new user programs to be downloaded via a PC's serial port. Normally when power is applied to the Mega8, it looks at location 0x0000 in flash for its first instruction. A user program normally starts at location 0x0000. The "Boot Reset Vector" fuse bit enabled in the "Fuse bit settings" section above causes the Mega8 to look at a location near the end of memory for its first instruction. When it does, it finds the bootloader and executes it first. The bootloader tries to communicate via Mega8's UART and if doesn't get a response from the PC, it jumps back to location 0x0000 to execute the user program. If a PC is attached, the bootloader establishes communications and waits for a program to be downloaded. As the bootloader receives code from the PC, it checks it for transmission errors and programs it into the lower portion of flash.

A bootloader is available for free from http://www.microsyl.com/ -- look for and download "MegaLoad". MegaLoad consists of two parts: A Windows application for downloading user code to a target; and the bootloader code that resides on the target. Install MegaLoad and look in the installed directory for "ATMega8.zip". This zip file contains the source code and a compiled "ATMega8.hex" file for the bootloader. Save "ATMega8.hex" someplace easy to find, as it will be needed in the steps below.

The following steps target the STK500 as the programming tool. Steps for the AVRISP should be similar. If you still have the "STK500" window up from setting the fuse bits, skip to step 4.

1. Open AVR Studio and hit the "Cancel" button to escape out of the project-selection dialog box.
2. Connect MEGAbitty to the programmer via the ISP cable and power up the STK500. Make sure MEGAbitty's red LED is not on. If it is, then power off the STK500 and remove the "Vtarget" jumper. It is important MEGAbitty have its own power source and be allowed to power the programmer. See "Connections/ISP" for details. If the red LED is not on, then power up MEGAbitty. Now you should see MEGAbitty's red light come on.
3. Open the programming tool by selecting "Tools->STK500/AVRISP/JTAG ICE…" It should pop up a "STK500" window with a status box at the bottom. Make sure it was able to detect the STK500.
4. Under the "Program" tab, set the following options:
   - ✓ Select "ATmega8" from the "Device" list.
   - ✓ Set the "Programming mode" to "ISP"
   - ✓ Enable "Erase Device Before Programming"
   - ✓ Enable "Verify Device After Programming"
5. In the "Flash" section, click the "**…**" button next to the "Input HEX File" box and browse to the "ATMega8.hex" file.
6. Hit the "Program" button to load the bootloader on to MEGAbitty. If you watch the blue and green LED's (D3 & D4) on MEGAbitty, they should flash briefly during programming.
7. Check the status box to insure programming succeeded and verified ok. If there isn't a message like "FLASH contents is equal to file… OK", hit the "Verify" button.
8. Power down MEGAbitty and disconnect it from the programmer.

Now MEGAbitty has something to do – look for an attached computer and download new programs. That's not very robot centric however, and MEGAbitty is capable of so much more. The next step is to give it something *real* to do.

## Downloading user code

With the bootloader installed, downloading user code is very simple. Three things are required first, however:

- A serial cable with RS232 level translator and 0.05" header to interface with J2 on MEGAbitty (See Appendix B for RS232 level translator sources)
- The MegaLoad windows application
- A ".hex" file.

The ".hex" file is generated from user-written code by a compilation and assembly process. AVR Studio will generate a ".hex" file from a program written in assembly. Higher level language compilers, such as CodeVision for C, or BASCOM for BASIC, generate a .hex file from C or BASIC code. Some compilers may only go half way and compile code to an assembly ".asm" file. Typically AVR Studio can then be used to compile the ".asm" file into a ".hex" file.

To start with, just try using a precompiled and assembled ".hex" file, such as "ledFader.hex" that is available from the "files" area on the MEGAbitty yahoo group:
http://groups.yahoo.com/group/megabitty/

With the above items on-hand, do the following:

1. Attach the RS232 level-translator serial cable to a free computer serial port and to MEGAbitty.
2. Start up the MegaLoad windows application.
3. Select the appropriate COM port and browse to the ".hex" file (i.e. "ledFader.hex") to download.
4. Apply power to MEGAbitty

That's it! If all went well, MEGAbitty's blue and green LED's should start fading back and forth a couple seconds after applying power. Upon power-up, MEGAbitty started trying to communicate with the computer; the computer responded; the MegaLoad Windows app dumped the new code onto MEGAbitty; and the bootloader handed over control to the newly downloaded program.

Power down MEGAbitty and disconnect it from the serial cable. Reapply power. There will be an approximately four second delay before the LED's start fading back & forth. During this delay, the bootloader is trying to find a computer to talk to. After four seconds or so, it gives up and jumps to the led fader program.

## *Creating your own user code*

With MEGAbitty up and running, you have a tiny but powerful controller and motor driver with lots of features just waiting to become the main brain of your next invention. Most likely that's going to require more sophisticated code than fading a couple LEDs back and forth. The following section provides a couple starting points for various programming languages.

### Assembly

(yet to be written)

**Resources:**

AVR Studio4 http://www.atmel.com/dyn/products/tools_card.asp?tool_id=2725

*Newbie's Guide To AVR Development* http://www.avrfreaks.org

### BASIC

(yet to be written)

**Resources:**

BASCOM-AVR (BASIC compiler) http://www.mcselec.com/bascom-avr.htm

FASTAVR (BASIC compiler) http://www.fastavr.com/

### C

(yet to be written)

**Resources:**

CodeVision (C compiler) http://www.hpinfotech.ro/html/cvavr.htm

*AVR033: Getting Started with the CodeVisionAVR C Compiler* http://www.atmel.com

GCC/WinAVR (C compiler) http://www.avrfreaks.org/AVRGCC/index.php

*Avr-gcc/AVRstudio beginners guide* http://www.avrfreaks.org

ImageCraft (C compiler) http://www.imagecraft.com/software/

*AVR031: Getting Started with ImageCraft C for AVR* http://www.atmel.com

IAR (C compiler) http://www.iar.com/

*AVR030: Getting Started With C for AVR* http://www.atmel.com

*AVR035: Efficient C Coding for AVR* http://www.atmel.com

## *Appendix A: ISP Programmer*

To use the ISP port, you need to make a cable to interface with the STK500, AVRISP, or other ISP programmer. Figure 2 & Figure 3 provide the standard 6-pin and 10-pin ISP connector pinouts used by most programmers. MEGAbitty's ISP connector pinout is shown in Figure 4. The 5-pin 0.05" pitch header required to interface with the MEGAbitty ISP port is included in the MEGAbitty kit. The cable and the 0.01" pitch connector for the cable's programmer side is left up to the user to provide. When choosing wire to use, look for small gauge wires to ease soldering to the 50-mil pitch header, as there is not much room between pins. Smaller gauge ribbon cable, or phone cable works well. After soldering and making sure there are no shorts or bad joints, consider potting the connector in hot-glue to prevent shorts from forming between the closely spaced soldered pins. Make sure to push hot-glue into the spaces between pins to provide an insulative barrier.

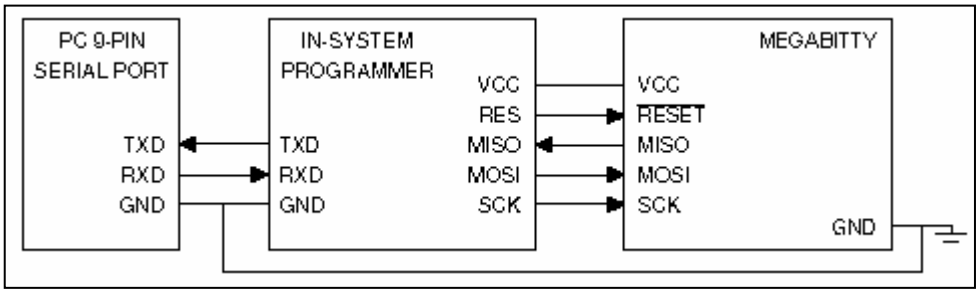



**Figure 1**: The programmer connects to a computer via a serial port, and to MEGAbitty via the ISP connector.

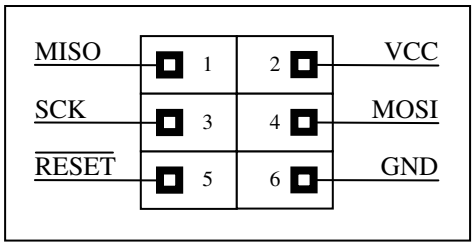



**Figure 2**: Standard 6-pin ISP header. 2x3, 0.10" square-pin connector.

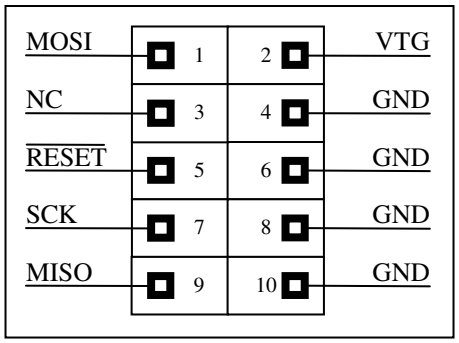



**Figure 3**: Standard 10-pin ISP header. 2x5, 0.10" square-pin connector.

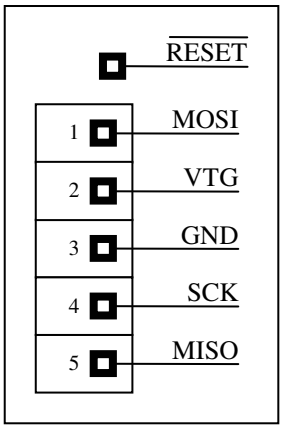



**Figure 4**: MEGAbitty ISP connector. 1x5 0.05" connector; reset line separate.

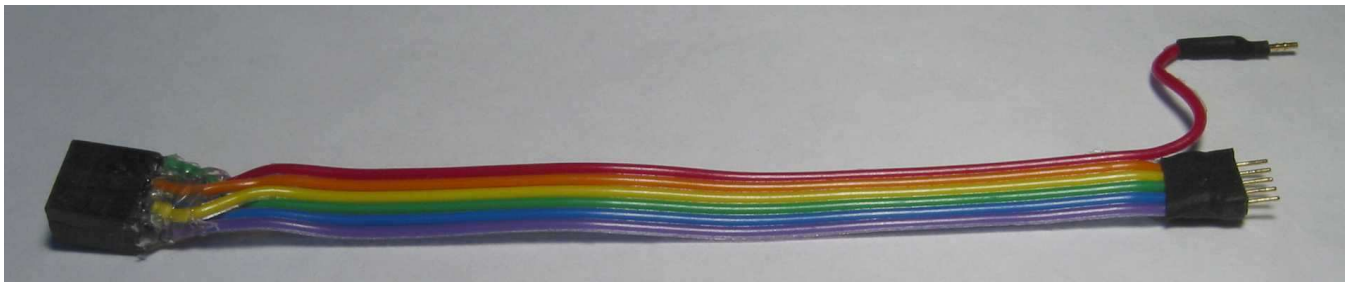


**Figure 5**: An example ISP adapter cable for MEGAbitty. A 6-pin 0.01" pitch socket (left) connects to the programmer. A 5-pin 0.05" pitch header (right) plugs into MEGAbitty's ISP connector, and a free-floating wire connects to RESET.

### ISP programmer sources:

**Atmel STK500**

The STK500 is more than just an ISP programmer. It is Atmel's AVR development board and supports most of the AVR family, including the Mega8. If you plan to play around with AVR's beyond MEGAbitty, consider picking up this board ($80 @ DigiKey). Note that you need a 12V supply (wall-wart) that is not included with the STK500.
http://www.atmel.com/dyn/products/tools_card.asp?tool_id=2735

**Atmel AVRISP**

The AVRISP is the ISP programmer portion of the STK500 by itself, boxed up for portability. It does exactly what's needed for ISP programming on MEGAbitty and other AVR's, but not much more. ($30 @ DigiKey).
http://www.atmel.com/dyn/products/tools_card.asp?tool_id=2726

**Atmel AppNote AVR910**

The AVR910 application note provides the details needed to make your own ISP programmer relatively cheaply. This do-it-yourself programmer uses the AT90S1200 AVR to translate serial commands to ISP commands with firmware that Atmel provides. Unfortunately, you'll need to install the AT90S1200 firmware somehow, which requires a programmer itself! Even if you don't build the programmer, this appnote provides a good overview on how ISP programming works.
http://www.atmel.com/dyn/resources/prod_documents/DOC0943.PDF

## *Appendix B: Serial Programmer (for use w/ a bootloader)*

Programming MEGAbitty over a serial port is the most convenient method for downloading new code; however, a special serial cable is needed. This is because MEGAbitty's serial port is only TTL (0 to +5V) compatible, while RS232 signaling levels range between +/- 10V or more. A number of chips are available for translating between TTL and RS232 signaling levels – most are a copy or a variant of Maxim's MAX232 (http://pdfserv.maxim-ic.com/arpdf/MAX220-MAX249.pdf). All that is needed to make a RS232 translator for MEGAbitty is a MAX232, four capacitors, and connectors. All parts are available at DigiKey, or a number of vendors offer kits or assembled products for relatively little money:

**RS232<->TTL Translator sources:**

**Acroname**

Acroname's affordable TTL<->RS232 converter is designed to work with Acroname's BrainStem modules, but will work with MEGAbitty as well. You still need to make a cable to connect MEGAbitty's 0.05" connector to the RS232 converter.
http://www.acroname.com/robotics/parts/S13-SERIAL-INT-CONN.html

**Kronos Robotics**

Kronos Robotics' affordable TTL<->RS232 converter in kit form – will also work well with the MEGAbitty. Again, you still need to make a cable to connect to MEGAbitty's 0.05" socket.
http://www.kronosrobotics.com/detail.asp?product_id=EZ232