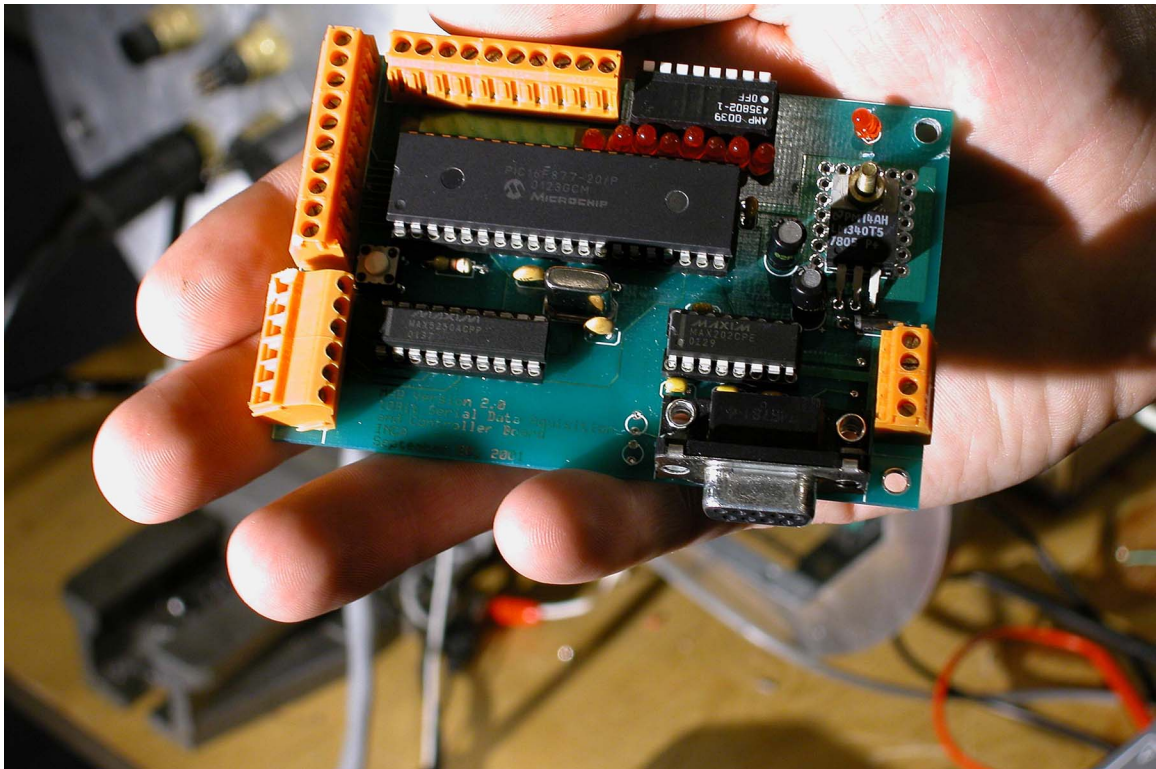


Centre for Instrumentation, Control and Automation User's Guide to the MAD² Microcontroller Board

Mark Simms

September 19, 2002



Analog Input	8 ports, 8/10-bit resolution
Digital I/O	8/16 ports
Timers	3 timers, up to .2us resolution
Serial	RS-232, up to 57.6k bps
LED's	8
In-circuit programming	Using the PIC's flash capability

Table 1: MAD^{2.1}Capabilities

The MAD^{2.1} microcontroller board is a PIC16F877-based controller board for use in a variety of applications. It can be used as a general purpose I/O card, or as a stand-alone embedded controller.

The MAD v2.0 was designed by Lloyd Smith, Steven Taylor and Mark Simms in September, 2001. The MAD v2.1 was updated by Steven and Mark in September 2002 to use a new analog output chip, and a different LED layout.

The board has the following capabilities:

The board is typically programmed using the CCS C compiler from CCS Computing (<http://www.ccsinfo.com/>).

1 Hooking up the MAD

In order to use the MAD^{2.1}, you need to connect it to a power supply (and possibly a PC to access the serial port). With a copy of the pinout diagram and table (see Appendix A) in hand, collect the following equipment:

- 1 MAD card
- A power supply capable of delivering +6V @ 100 mA. In this lab this will typically be a DC lab power supply, or perhaps a battery (or two).
- A flat-head screwdriver, with a small enough head to adjust the headers on the board.

With the equipment in hand, connect up the MAD^{2.1} as follows:

1. Connect the positive rail from your power supply (or the +ve lead on the battery) to the V_{in} pin on the MAD^{2.1} card.
2. Connect the negative/common rail from your power supply (or the -ve lead on the battery) to the $GND IN$ pin on the MAD^{2.1} card.

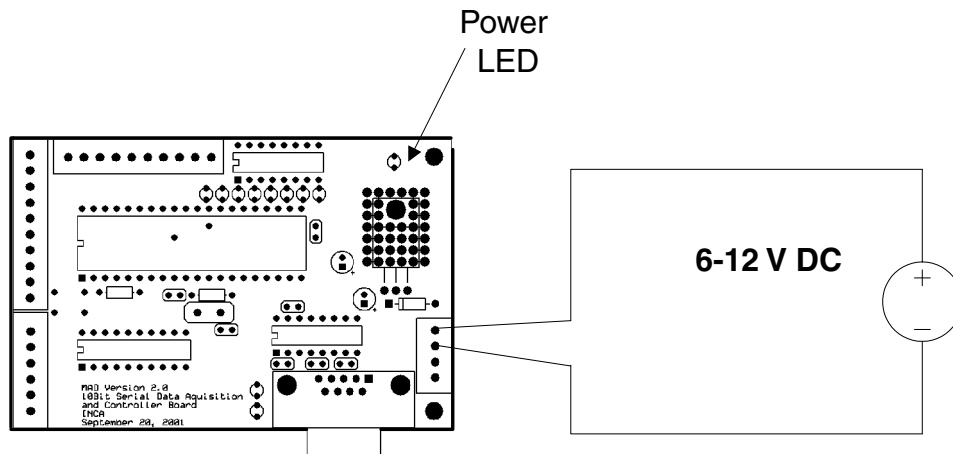


Figure 1: Connecting Power

3. Dial the power supply voltage down to 0. Turn on the power supply, and increase the voltage on the supply to roughly 6.0 V or higher. The power LED on the MAD^{2.1} card should then light up. This is to avoid a rush of current when the switching power supply comes on (which can fry components).

If using a battery, this step is obviously redundant.

The MAD^{2.1} should now be connected, powered up, and ready to program.

Note that this is the procedure for direct power supply connection. There is also support for a +9V DC adapter to be plugged into the MAD^{2.1} board, though that is not covered in this guide.

2 Programming the MAD^{2.1}

This section covers how to write code for the PIC16F877 on the MAD^{2.1} card, how to burn code onto the PIC, and how to use the bootloader and the CCS C compiler.

2.1 Using MPLab

MPLab is the application used to write code for the PIC. It can be used to write code in a variety of languages, including assembler and C. The C compiler used in the INCA lab, and highly recommended, is the Custom Computing Services PCM C compiler. Writing C code as opposed to assembler is generally far more efficient in terms of developer time (i.e. your time).

The following sections show the list of steps involved in configuring MPLab to use CCS and compile code for the PIC.

2.1.1 Creating a new Project

1. Create a directory to store your application in. For example *C*:
picapps
test
.
2. Start the MPLab application.
3. From the menu at the top of the screen choose *Project*, then *New Project*. Save the new project in a suitable location. For example, *C*:
picapps
test
test.pjt.
4. Configure the project for the MAD^{2.1} board. To do this:
 - Select the correct processor, the PIC16F877. As seen in figure 2, click on the *Change* button next to development mode. In the popup dialog, as seen in figure 3, click on *None (Editor Only)* and the *PIC16F877* processor from the pull-down list.
 - Select the correct development tool suite, CCS. From the pulldown *Language Tool Suite*, select CCS. You will be informed that all of your program settings will be lost – this is normal.
5. Configure the CCS compiler. To do this click on the [project].hex file, then on *Node Properties*. In the popup dialog that appears, as seen in figure 4, click on the *PCM* compiler. If you wish, other compiler directives may be set here – see the CCS manual for details.
6. Create the source file. From the menu, *File, New*. Then, *File, Save As*; save your new source file with an appropriate name, such as *project.c*.

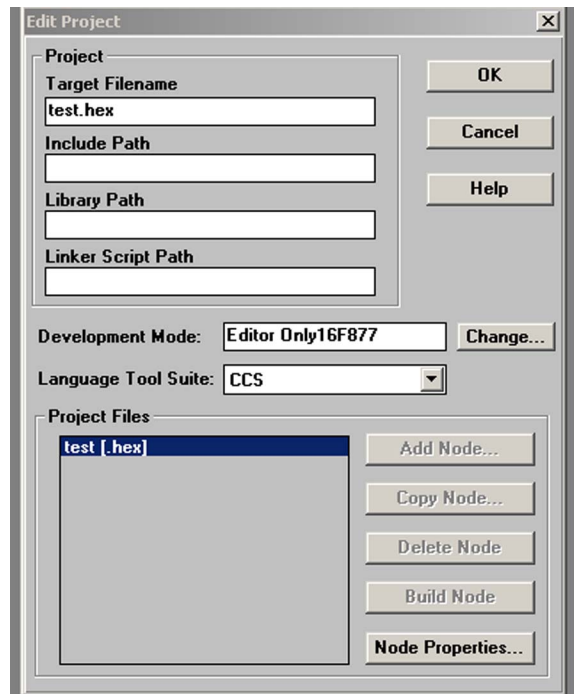


Figure 2: MPLab: New Project

MPLab should now be properly configured to compile an application for the MAD^{2.1}.

2.2 Simple Test Program

A sample test program can be seen in figure 5. It is set up for the MAD^{2.1} with a PIC16F877, a 20 Mhz clock, bootloader and serial connection.

The first line in the file, `#include...` includes the header file that defines how the PIC16F877 operates. The next line sets the configuration fuses¹. The third line sets the

¹See the CCS manual of the PIC data sheet for details

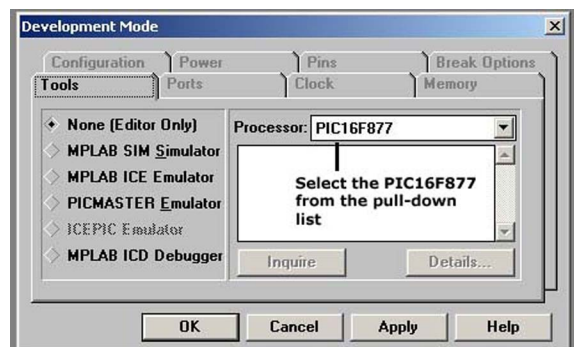


Figure 3: MPLab: Selecting the Development Mode

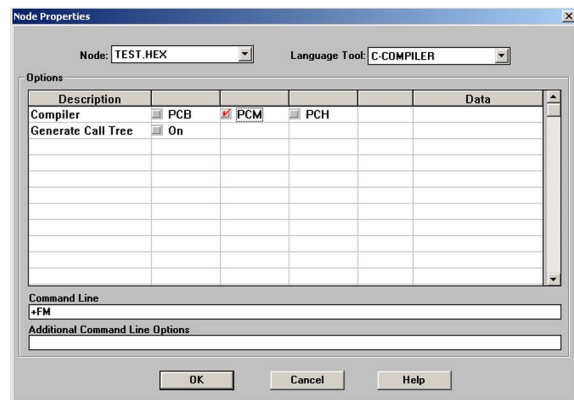


Figure 4: MPLab: Compiler Settings

program to use a 20 Mhz clock (which is what the MAD^{2.1} uses). The fourth line configures the RS-232 connection.

The fifth line, `%ORG...` instructs the compiler to reserve the last 255 bytes of memory. This is where the bootloader resides – if you forget this line you can corrupt or erase the bootloader from the PIC.

Every program you will develop for the MAD^{2.1} using the CCS compiler will generally start with these five lines of code.

Program .1: The C Source

```

1  #include <16F877.H>                // Using the PIC16F877
2  #fuses HS,NOBROWNOUT,NOPUT        // Configuration Fuses
3  #use delay(clock=2000000)          // 20 Mhz clock
4  #use rs232(baud=9600, xmit=PIN_C6, rcv=PIN_C7, PARITY=N, BITS=8)
5  #ORG 0x1F00,0x1FFF {}             // Reserve Memory for the bootloader
6
7  void main() {
8      puts("Pumpkin Simple Test Code, v1.0"); // Print a message to the
9                                              // serial port
10     while (TRUE) {
11         output_high(PIN_D0); // Make a status LED blink with a period
12         delay_ms(500); // of 1 second
13         output_low(PIN_D0);
14         delay_ms(500);
15     }
16 }

```

Figure 5: Sample MAD^{2.1} Test Program

Inside the main function of the program, the PIC will print out a short status message over the serial port, and flash an LED on and off with a period of 1 second. If you try this out and get serial output, but no LED flashing, check the settings on the DIP switches.

Type in this program, and save it. Then compile your application to binary code by selecting *Project, Make Project* from the main menu. If you have configured everything properly the application should compile without any errors.

Congratulations, your first .hex file (or compiled binary code file) is ready to be downloaded onto the PIC.

2.3 Downloading or Burning Code onto the MAD^{2.1}

There are two methods of downloading code onto the PIC on the MAD^{2.1}board. These are:

1. **Burning.** The PIC16F877 is removed from the MAD^{2.1}board, and placed into an EEPROM burner, such as the PICStart+. The burner then programs the on-board PROM in the PIC with the new code.

This generally requires 2-3 minutes of burn time, and requires the PIC to be removed from the MAD^{2.1}board. This method should only be used when burning a new bootloader onto a PIC, or when putting the final version of program onto the PIC before installing in a “production” system.

2. **Bootloading.** The usual method of downloading new code to the MAD^{2.1} is via the serial port, using the bootloader.

The PIC16F877 has stores its program code in Flash, which the PIC itself can modify. The bootloader takes advantage of this capability by allowing fast and efficient in-circuit programming.

The PIC is initially burned with a small bootloader program that sits at the top of memory. Whenever the PIC starts up, the bootloader program checks to see if the user is trying to send any new code down over the serial link. If so, it downloads this code and writes it to the on-board Flash memory. If not, it times out (usually after one second) and executes whatever code was previously bootloaded onto the PIC.

The bootloader can download a full-sized (8kB) program in about 15 seconds without changing the circuit – this is the most efficient way of developing and downloading code onto the MAD^{2.1}.

2.4 Using the Bootloader

To use the bootloader, start the PC-side bootloader application, as seen in figure 6.

To download a program to the MAD^{2.1}, select the appropriate .hex by clicking on the *Select* button, and selecting the file from the pop-up dialog.

Then click on the *Write* button. The PC-side bootloader program will then wait for the MAD^{2.1} to restart. Click the reset button on the MAD^{2.1}, as shown in figure 7.

The progress meter on the bootloader should then advance as it downloads the program to the MAD^{2.1}. When the download is complete, reset the MAD^{2.1} again, and your new application should start up.

To test this start up HyperTerminal, and configure a serial connection with 9600-8N1, as seen in figure 8. Reset the MAD^{2.1}, and you should see the status message from the MAD^{2.1}. This status message can be seen in figure 9.

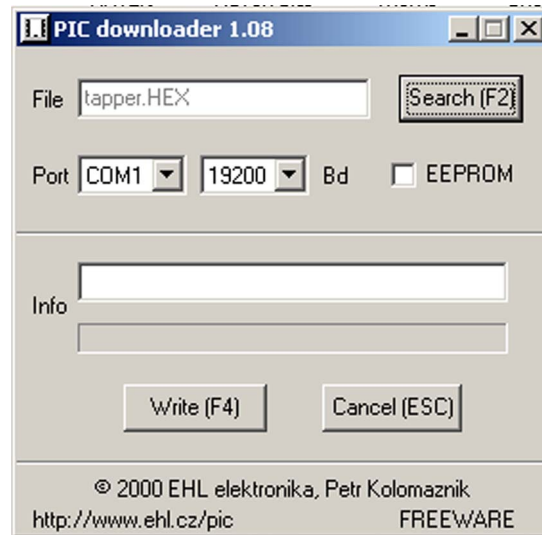
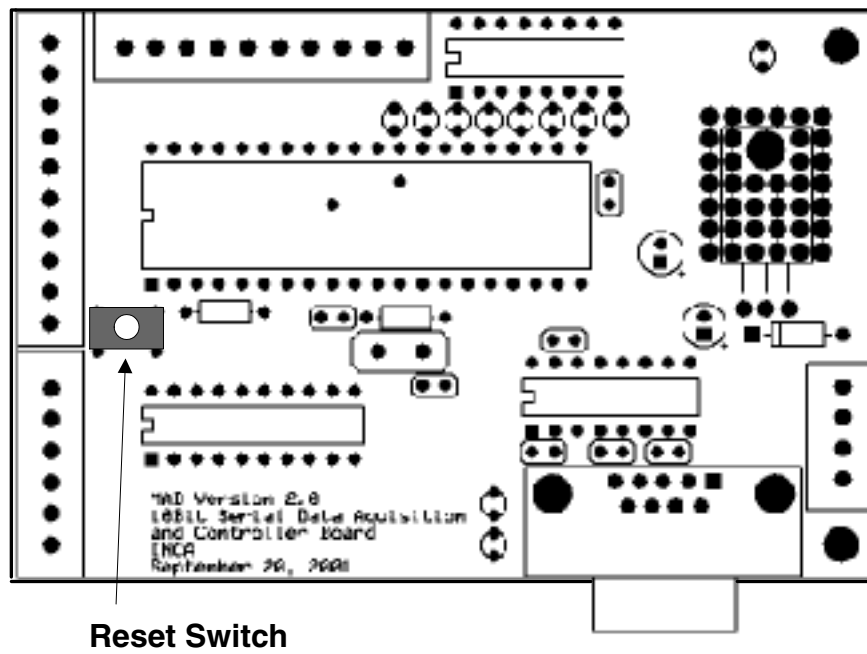


Figure 6: The PC-side Bootloader

Figure 7: MAD^{2.1}: The Reset Button

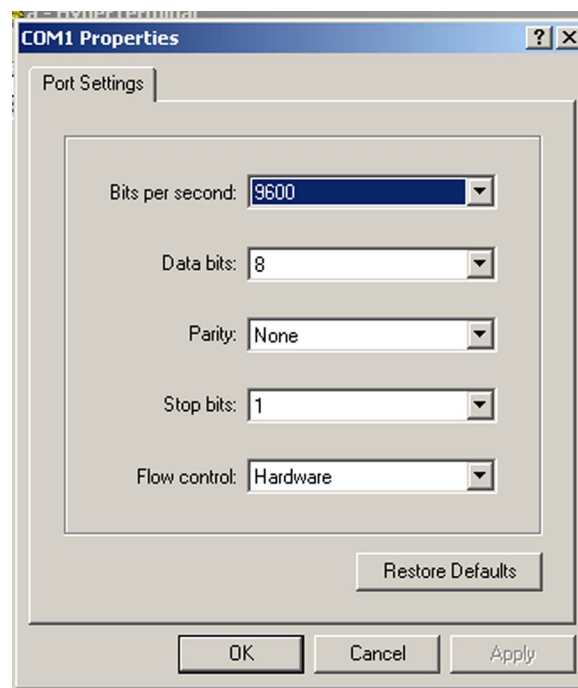


Figure 8: Configuring Hyperterminal

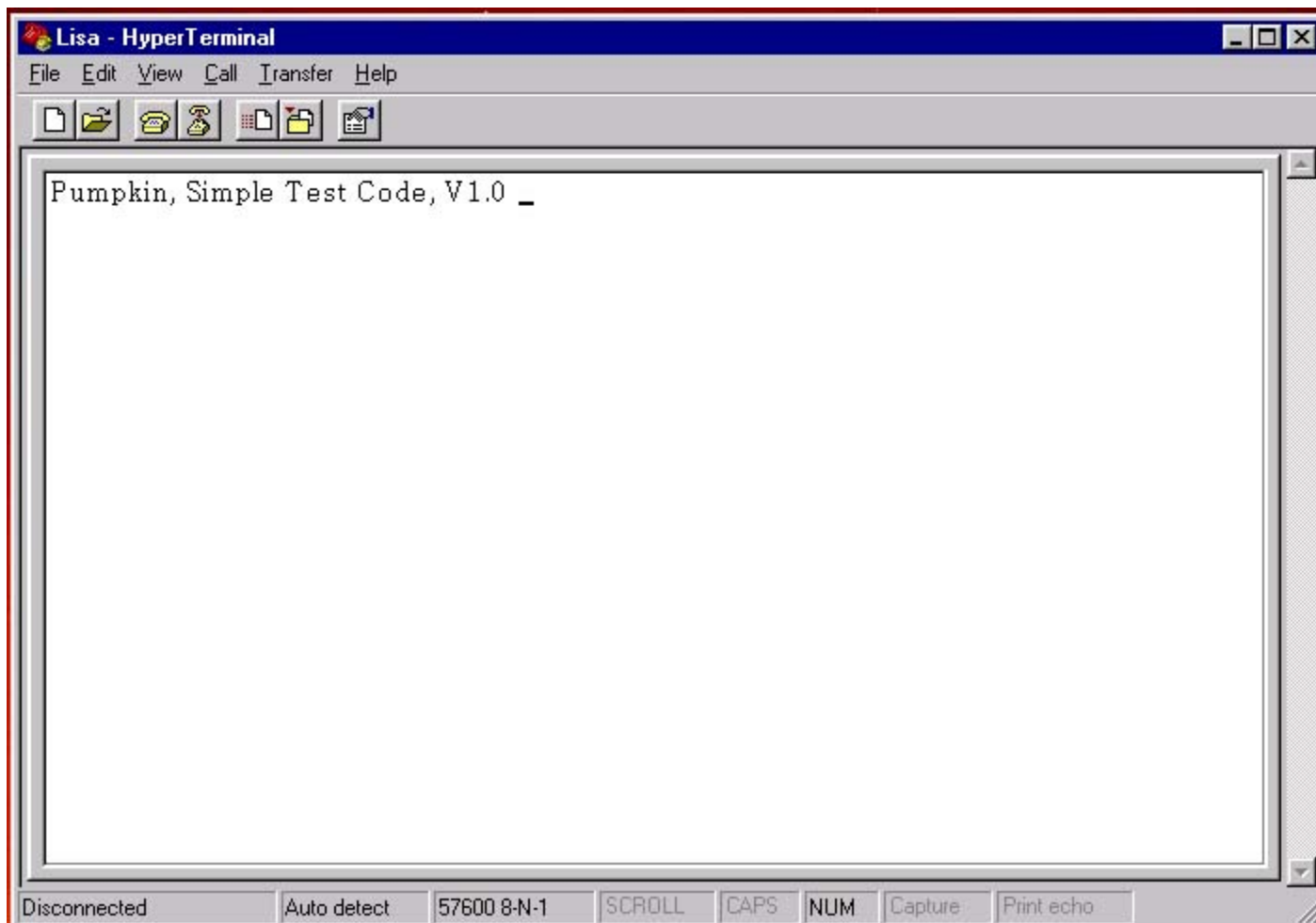


Figure 9: Hyperterminal: Displaying output from the MAD^{2.1}

