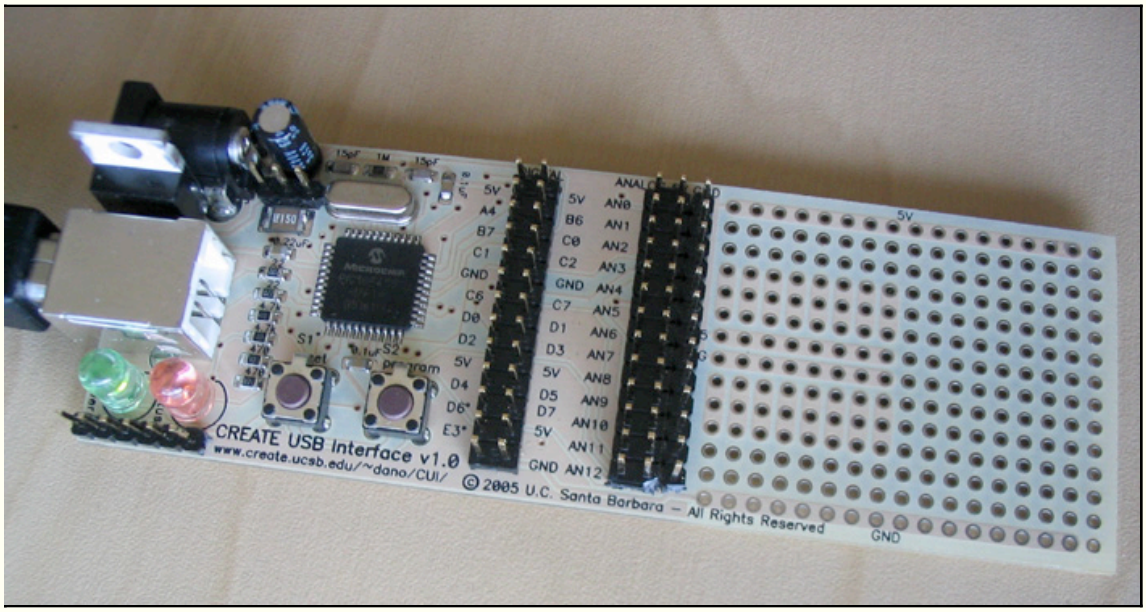
The CREATE USB Interface - where art meets electronics



CUI version 1.0

Dan Overholt
U.C. Santa Barbara
Center for Research in Electronic Art Technology

The need for the "CUI"

In the Media Arts and Technology program, we explore new metaphors for artistic interactivity that connect the physical world with the virtual realm. We develop new techniques for computing that generate music and visual arts in a myriad of ways; but in order to put forth these techniques, we must create new sensors, and build interfaces that can better grasp their control and generation. The CUI allows us to bind physical processes or actions to corresponding digital expressions.

To simplify this process of connecting the real world to the virtual, the CUI provides the necessary electronics to capture sensor input or control actuator output. The CUI comes with a USB port, a power LED, a Reset switch, a Programming switch, and a prototyping area. At the heart of the board is the PIC18F4550, a versatile microcontroller made by Microchip Technologies, Inc. The PIC18F4550 features thirteen A/D inputs, eighteen general I/O ports, and an efficient RISC-like instruction set. The CUI only uses one of the PIC's general I/O ports; the remaining ports are available for user applications to be built in the prototyping area. The total cost of the CUI is around $30 US when built in small quantities.

About this Document

This document is written primarily for anyone who wants to create an interface using the CUI. It is also useful for anyone who wants to understand the process of developing applications for microcontrollers in general, and for Microchip PIC processors in particular.

Here's what you'll find in this document:

- Section 1 offers some ideas of the kinds of interfaces you could build using the CUI and gives an overview of the development process.
- Section 2 gives a guided tour of the CUI board, and includes the board layout, the schematic, and a detailed description of each major component on the board.
- Section 3 tells you everything you'll need to know to build your own CUI board, including purchasing components for the board and fabrication hints.
- Section 4 details the process of designing and programming the CUI board.
- Section 5 offers some debugging techniques and programming tips.
- Appendix A contains a few related web links.

Acknowledgements

The author gratefully acknowledges the support and inspiration from the following people. Curtis Roads, JoAnn Kuchera-Morin, Stephen Travis Pope, George Legrady, and other faculty members of the MAT, ART, and CS departments.

Wesley Smith and Rama Hoetzlein kept things moving through the formative stage of the CUI project. They volunteered many hours to help develop the code and teach other students about PICs and basic electronic circuitry. All of the students involved in the Media Interface Technology course have made this a fun project. And thanks to Robert Poor for the web page layout, taken from his iRX page.

1.0 Overview

What you get out of the CUI depends on what you put into it. It's unlikely that there will be any "pre-canned" configurations that exactly suit your needs--you will generally need to add custom hardware on the board and create a PIC program specifically for your application. But that's the fun part of the process!

To give you a taste of what's possible with a PIC and CUI board, here are some projects that people could create using the CUI (or *should* be able to create):

- A musical interface. The CUI board can be programmed to convert analog sensor inputs to a standard USB protocol, allowing you to control an interactive composition or performance setup.
- A novel visual interface. The PIC can measure input from a sensor such as an accelerometer, and the host computer can run a visualization algorithm which interprets user input as gestural data to control different aspects of the visual output.
- An Ultrasonic Ranging device (for example, using devices such as the SRF04). The PIC is sufficiently fast to generate outgoing pulses and measure the results accurately.
- A game controller. The PIC can be programmed to behave as a joystick-type device by enumerating (telling the host computer about itself) as a game pad. The addition of sensors for input and actuators for force-feedback can provide an easy way of getting data in and out of various host applications (e.g. Max/MSP/Jitter, Pd, SuperCollider, Processing, etc). Using the standard game controller input method bypasses the need to write your own driver for the USB protocol by piggy-backing on the HID (Human Input Device) standard.

Overview of the Development Process

This section briefly describes the steps you'll need to take to create your very own CUI application.

Ready...

The first step is to make sure you have all the materials at hand. You'll need both some hardware and some software:

- *Acquire one or more CUI boards*. Section 3 gives you information on how to fabricate the CUI board, where to purchase the required components, and how to assemble your own boards.
- *Obtain a copy of MPLAB*. MPLAB is an integrated development environment from Microchip. It includes a text editor, assembler, simulator, and programmer support. Best of all, you can download it from the WEB free of charge.
- *Obtain the PIC C18 C-compiler*. Student Edition, version 2.4 or later. Install it in the default location--examples in this document presume that you're using the C18 compiler. This is also a free download from Microchip, but some functionality is limited after 60 days.
- *Acquire the CUI example code*. Download the CUI examples based on Microchip's HID connectivity solutions. The CUI Example Code contains everything needed to get to the point of "Hello World" with the PIC. It should be extracted to the root directory of your hard disc, preserving the names and directory structures within.

Set...

It's unlikely that you will find any pre-compiled firmware for the PIC and CUI that will do exactly what you want. Usually, you'll need to write a program for the PIC (or perhaps modify an existing one). In most cases, you'll also be adding extra components and wires to the CUI board.

The CUI was meant to be customized, so go for it!

- *Understand the fundamentals of the PIC architecture*. Appendix A has pointers to the Microchip site, where you can find details such as the data sheet for the PIC18F4550.
- *Understand the fundamentals of the USB protocol*. Appendix A also has pointers to useful documentation and software for developing USB HID devices.
- *Find related firmware*. Perhaps someone has already created something similar to what you need. There are three rich sources of firmware: the Microchip Application Notes (but most of those are written in assembly), the C18 C Compiler examples, and other user's CUI firmware from previous projects (check the Sensors Wiki to see if there's anything similar to what you want to do). Consult these resources first; it can save you lots of time.
- *Decide on what modifications, if any, you must make to your CUI board*. Section 4 details the features and differences of the various I/O pins available.

Go...

Creating and downloading code for the PIC is a five step process, described in the picture and in the text below.

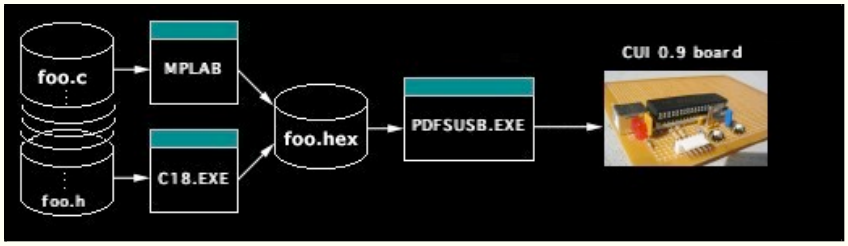
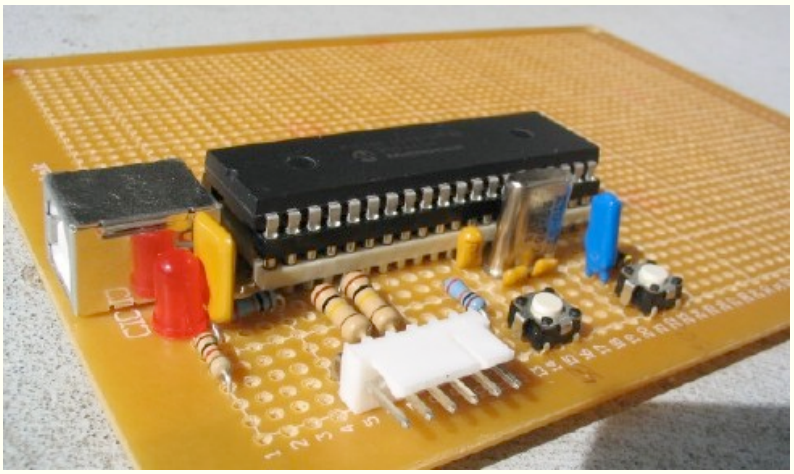


Figure 1: The development process

1. *Write your program* using your favorite text editor or the editor included in Microchip's MPLAB. Generally, you'll write your program in C. In exceptional cases, you may write your program in assembly code.

2. *Compile your program*. Normally, you'll use the MPLAB environment to invoke the C18 C-compiler. C18 will compile your C code into a .hex file. If you've written assembly code, MPLAB will assemble your .asm file into a .hex file.
3. *Program the PIC18F4550* by downloading the .hex file via bootloader using the PDFSUSB.EXE program (located at C:\CUI\Pdfsusb\ by default). In order to get the CUI to show up in the PDFSUSB.EXE menu, you must reset your board while holding down the "program" button. **NOTE:** if this is the first time you have ever programmed a newly built CUI board, you must first burn the bootloader code into the PIC. Section 5 details this one-time procedure that requires the use of a PIC programmer such as the ICD2.
4. *Run your application*. This is as simple as hitting the "execute" button in PDFSUSB.EXE, or hitting the reset button on your board, and watching it run. If it runs okay, hooray! If it doesn't, go to the next step.
5. *(Oops.) Debug your application*. Section 4 offers tips and techniques for debugging.

2.0 Guided Tour of the CUI "0.9" DIY approach



This section describes the layout and schematic of the CUI 0.9. It will show you where each component is located, and d component does. Figure 2 shows the layout of the components on the CUI 0.9 board, Figure 3 is the schematic for the bo

2.1 Layout of the CUI 0.9

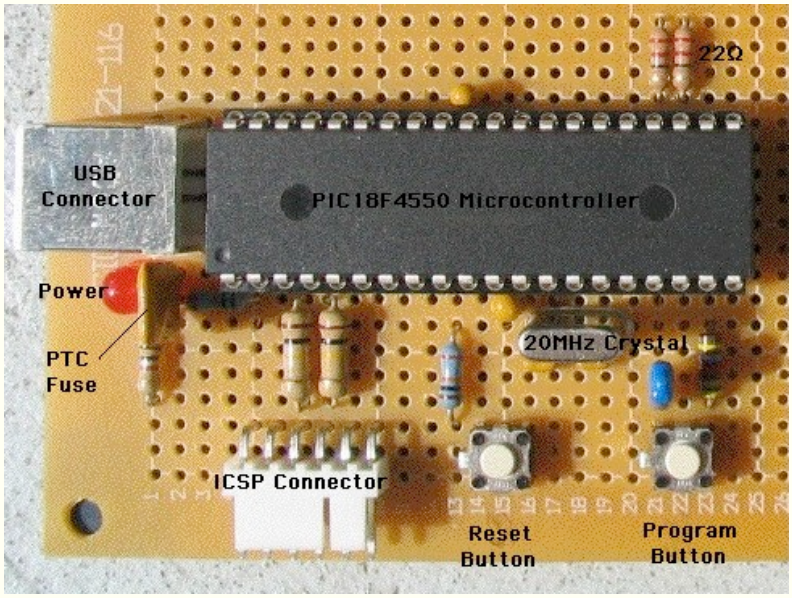


Figure 2: Layout of the CUI board

Note that this layout is slightly out of date and needs to be corrected. The resistors above the ICSP (In-Circuit Serial Pro not needed, and the pull-up resistors for the Reset and Program buttons values have changed. The 100uF capacitor is not see the 15pF capacitors next to the 20 MHz crystal. The bypass capacitors and VUSB capacitor are not labelled in the pl

2.2 Schematic of the CUI 0.9

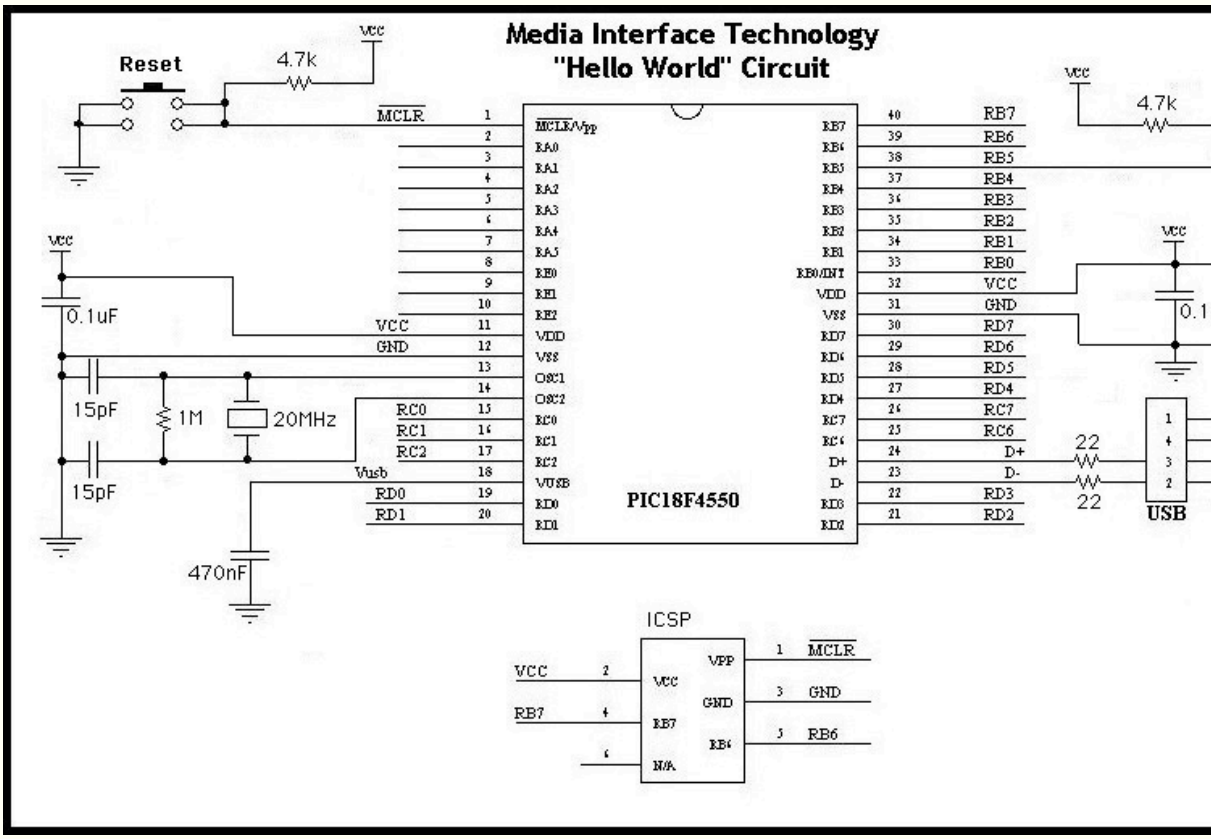


Figure 3: Schematic of the CUI 0.9 board

2.3 Component Description

2.3.1 USB Connector

The USB Connector is a standard "type B" connector. You can plug any "A-to-B" type cable into it, with other end (the
your computer. There are four connections in a USB cable, two of which supply power to the CUI board, while the other t
communications lines D+ and D-. This is how information is transferred between the host computer and the PIC both wh
and while your firmware sends or receives data with the computer if it is a HID application.

2.3.2 Power indicator

When USB power is applied, the LED is lit. Note that some CUI circuits may not be able to use USB power if they require
current (this is maximum amount of current that is allowed to be drawn from a single USB port). The alternative is to use
regulator such as the LM7805, which then provides +5V at 1000mA (1Amp) from a 7-15VDC "wall wart" power supply.
you use a 9V battery connector and power the CUI 0.9 from a standard 9V battery for stand-alone applications.

2.3.4 Push buttons "Reset" and "Program"

The two buttons on the CUI are used during the process of programming your application. They are labelled "Reset" and
Reset button is the equivalent of unplugging the USB cable and plugging it back in (which should cause your computer to
initialize the corresponding driver). If you push the Reset button while holding down the Program button, the CUI will e
which will allow a new application to be loaded into the PIC (via the PDFSUSB.EXE program).

2.3.5 ICSP Connector

The In-Circuit Serial Programming (ICSP) connector is only used one time for each new CUI board. It allows the bootloa
PIC's memory via a standard PIC programmer such as Microchip's ICD2. Section 5 details the process of installing the b

2.3.6 PIC I/O Ports

All of the unused pins of the PIC on the CUI board are user I/O ports that can be connected to custom circuit extensions a
etc. You can solder some "Molex headers" into the board if you want a simple means to temporarily connect wires and co
board. The layout of the I/O ports follows the pinout of the PIC18F4550 as shown below. A description of the characteri
Section 4.

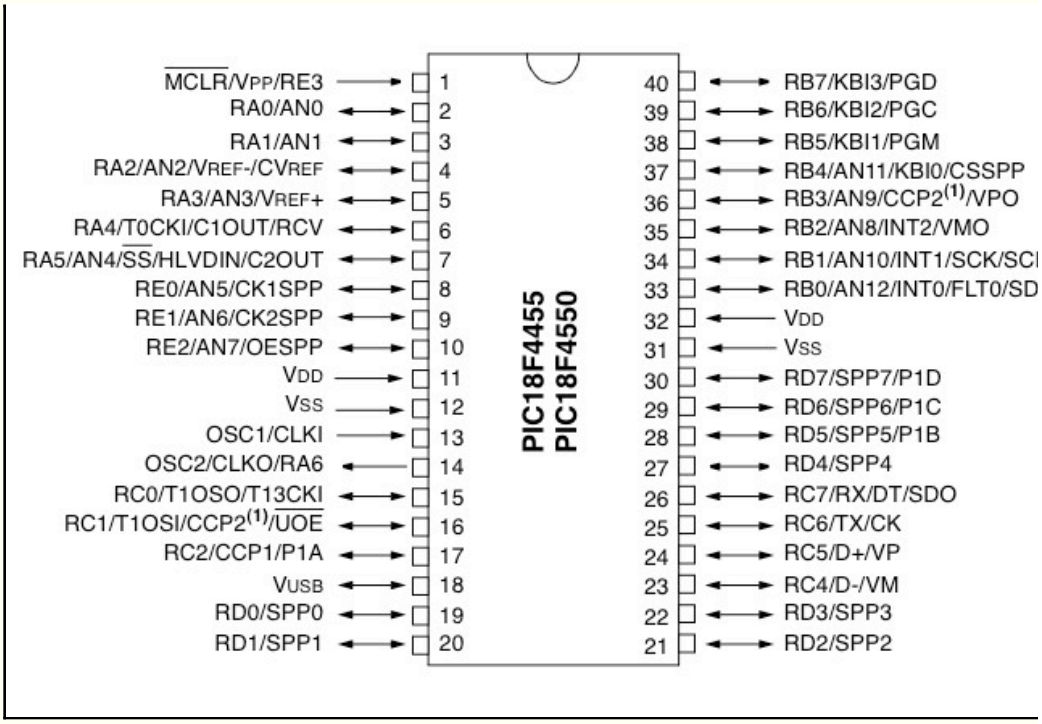


Figure 5: PIC I/O Ports

2.3.7 PIC18F4550

The PIC is the heart of the board. It's a programmable microcontroller with 32Kbytes of flash program memory and 2kb SRAM. It has 13 A/D inputs and 18 general purpose I/O ports. On the CUI board, one of the general purpose I/O pins is c button to enter bootloading mode. You can use the remaining ports for anything you wish.

2.3.8 20MHz Crystal oscillator

This particular part has the job of providing a clock signal to the PIC, and it's corresponding 15pF capacitors and 1Mohr make it oscillate properly. Crystals with other frequencies are available, but 20MHz was chosen since it's the same as th Microchip.

2.3.9 PTC Fuse

This part protects your computer and the CUI board from damage in the event of a short circuit or over-current condition. Temperature Coefficient (PTC) type fuse, which means that it is resettable, and will automatically allow your circuit to w circuit is fixed.

3.0 Building a CUI 0.9

This section details the process of building complete CUI 0.9 boards from scratch:

- Purchase the components
- Solder the components onto the board

3.1 Purchasing the components

All of the components are included in the lab kit for class, or they are available from Digi-Key, Mouser, or other mail-order electronics parts suppliers. While those mentioned are not always the cheapest, they have most all of the compoents in stock, take online card orders, and will get you your parts however fast you want them (FedEx, etc). If you have more time than money on your hands, you may be able to shop around and get a better price on some of the parts shown here. Additional parts, such as a 7805 voltage regulator or special sensors you wish to use can also be acquired from Digi-Key, Mouser, etc., or possibly at a local store such as Marvac or Radio Shack if the component is not too esoteric.

The following lists details the parts required to build a single CUI board. The first column specifies component name, and the Mouser part # is shown in the second column (or other alternatives if Mouser does not stock them). When possible, I've shown Mouser's prices (as of April 2005), but these reflect pricing when purchasing supplies for more than just one CUI (10 or more). If you would like the list of parts in Excel format, this is the bill of materials spreadsheet.

description	Mouser #	quantity	price (each)	total
PIC18F4550	www.sparkfun.com has some	1	10.9	10.9
Perf board	local electronics store	1	3.99	3.99
USB cable	local electronics store	1	3.99	3.99
Push button	652-SDTG-610-N	2	0.2	0.4
40-pin socket	517-ICO-406-S8A-T	1	0.22	0.22
PTC fuse	576-030R0160U	1	0.39	0.39
20MHz xtal	815-AB-20-B2	1	0.36	0.36
ICSP header	571-6404556	1	0.15	0.15
USB typeB	154-2442	1	0.39	0.39
Linear pot	312-619B-50K	1	1.8	1.8
LED red	604-L132XID	1	0.12	0.12
LED green	604-L132XPGD	1	0.14	0.14
LED yellow	604-L132XYD	1	0.12	0.12
R - 1M	291-1M	1	0.07	0.07
R - 4.7k	291-4.7K	2	0.07	0.14
R - 470	291-470	3	0.07	0.21
R - 22	291-22	2	0.07	0.14
C - 15pF	80-CK05BX150K	2	0.24	0.48
C - 0.1uF	539-CK05104K	2	0.31	0.62
C - 470nF	581-TAP474K035SCS	1	0.4	0.4
C - 100uF	647-UVR0J101MDD	1	0.11	0.11

3.2 Soldering the components onto the board

Soldering the components onto the CUI board isn't difficult. Assuming you've already had basic experience with soldering, here are some hints:

The only components that *don't* care which way they're inserted are the resistors, the ceramic capacitors (the 100uF is an electrolytic cap so it needs to be oriented with its minus leg to ground), and the crystal

oscillator. For all others, use the schematic diagram as a guide. Note that for the LED, the longer wire is the positive '+' side.

There are a few components that should be placed as close as possible to the PIC, should you decide to change the layout on your board. These are the 0.1uF bypass capacitors, and the 20MHz crystal oscillator - the bypass capacitors help keep the power supply clean, and the crystal needs to be close to ensure a good clock signal is provided to the PIC (shorter traces provide lower inductance for this high frequency signal).

To keep components from falling out while you're soldering others (since you have to flip the board over to solder the leads) it may help to work from the "shortest" to the "tallest" components. Be careful not to overheat the sensitive components during soldering, especially the PTC, and the smaller capacitors.

4.0 Developing an Application

The steps towards creating your PIC application are straightforward and have been outlined in Section 2. In this section, we jump into the details.

4.1 Not all I/O Ports are created Equal

The PIC18F4550 has thirteen A/D ports: RA0 - RA3, RA5, RB0 - RB4, and RE0 - RE2. Among these, the order is somewhat random so be sure to look up the correct pin number in the data sheet. (Note: you can still use these pins as general purpose I/O pins if you're willing to give up A/D converter inputs). Some ports have particular traits that may make them more or less suited for your particular application. The following table summarizes the differences; for the full scoop, consult the Microchip data sheets.

Except as noted, all ports support TTL levels and are bi-directional under program control.

Table 3. Summary of I/O ports on PIC18F4550

Port	Other Functions	Traits
RA0	Analog input 0	
RA1	Analog input 1	
RA2	Analog input 2	
RA3	Analog input 3	
RA4	Digital I/O	Schmitt Trigger input, can be programmed to be input to TMR0 clock
RA5	Analog input 4	
RB0	Analog input12	can be programmed for external interrupt (INT0)
RB1	Analog input10	can be programmed for external interrupt (INT1)
RB2	Analog input 8	can be programmed for external interrupt (INT2)
RB3	Analog input 9	
RB4	Analog input11	can be programmed for interrupt on change
RB5	Digital I/O	can be programmed for interrupt on change
RB6	Digital I/O	can be programmed for interrupt on change
RB7	Digital I/O	can be programmed for interrupt on change
RC0	Digital I/O	
RC1	Digital I/O	can be programmed for PWM output
RC2	Digital I/O	can be programmed for PWM output
RC6	Digital I/O	can be programmed for UART TX line
RC7	Digital I/O	can be programmed for UART RX line
RD0	Digital I/O	
RD1	Digital I/O	
RD2	Digital I/O	
RD3	Digital I/O	
RD4	Digital I/O	
RD5	Digital I/O	
RD6	Digital I/O	
RD7	Digital I/O	
RE0	Analog input5	
RE1	Analog input6	
RE2	Analog input7	

The CUI 0.9 board leaves many of these pins entirely untouched. This makes it possible to build your own circuit around the microcontroller. Port B, when used as inputs, can be programmed in software for internal weak pull-ups (this is useful when connecting simple switches or buttons to the PIC).

4.2 C versus Assembly code

Whenever possible, program in C instead of in assembly language. You'll find the development and maintenance of firmware programs to be much faster and easier. However, there are some times when you may still need to program in assembly (or put sections of assembly "in-line" using the #asm construct):

- Time critical applications, where you must be able to account for every cycle. In general, you can't

rely on what code the C compiler will generate nor what run-time support code will be run in the course of execution.
- Code critical applications. The C compiler can't assume much about the state of ports and registers, so it generates some extra code in the name of saving and restoring state.

The PIC18F4550 is a monster compared to some of the smaller devices, so most likely you won't have to worry about assembly though.

4.3 Compilers

There are a number of different PIC C compilers available. Till now, we have stuck with C18 from Microchip, but there are also CCS, Hitech, and GCC compilers available. The C18 compiler comes with good examples, which is one benefit, but the 60 day student trial is possibly a hassle. However, we have yet to run into any real problems with this. As for assemblers, it's easiest just to use MPASM, the free PIC assembler from Microchip that's built into MPLAB.

4.4 Burning your program into the PIC

The wonderful thing about a bootloader is that you don't need to have a device programmer to load new code into your PIC. In a few rare circumstances though, you may want one anyway. Though there are many PIC programmers available, we've settled on the ICD2 programmer from Olimex. If needed, it can purchased online via Spark Fun Electronics, and it has proven to be robust and reliable so far, with upgrades for the programmer after they introduce new chips a non-issue (it emulates a "real" ICD2 as far as MPLAB is concerned).

The easiest way to download your bootloader into the PIC is through the ICD2 programmer menus in MPLAB.

4.5 PIC Simulators

A PIC simulator lets you step through your program, set breakpoints, examine registers; some of these things can be done with the real chip as well if you have the ICD2 (In-Circuit Debugger) hooked up to your CUI board. If you don't have an ICD2, the simulator built into MPLAB can give good results.

5.0 Debugging Tips & Techniques

Following is a random collection of tips, techniques, art and lore. Your mileage may vary, but little things such as these can sometimes be the difference between success and complete lunacy.

5.1 First principles

When you write your program, it is *always* a good idea to make an LED blink, just to indicate that the chip is alive and running. Obviously, this cannot be the power LED, but another LED wired up to a Digital I/O pin that is configured as an output.

So assume you've written your program, compiled and loaded the code, and powered up your CUI. And the LED doesn't blink. Now what?

- Are you getting power? Measure with a multimeter between the +5V and GND on your board. You should see a stable +5 volts. If not, make sure that the PIC is inserted correctly: the "dimple" (indicating pin 1) should be pointing towards the USB connector at the "top" of the board.
- Is the PIC's oscillator running? Look between ground and pin 13 or 14 on the PIC using a 10x oscilliscope probe (the impedance of a 1x probe is sufficiently small to kill the oscillations.) If you don't see a 20MHz waveform, did you remember to configure the programming fuses for an "HS" oscillator type?

5.2 Debugging

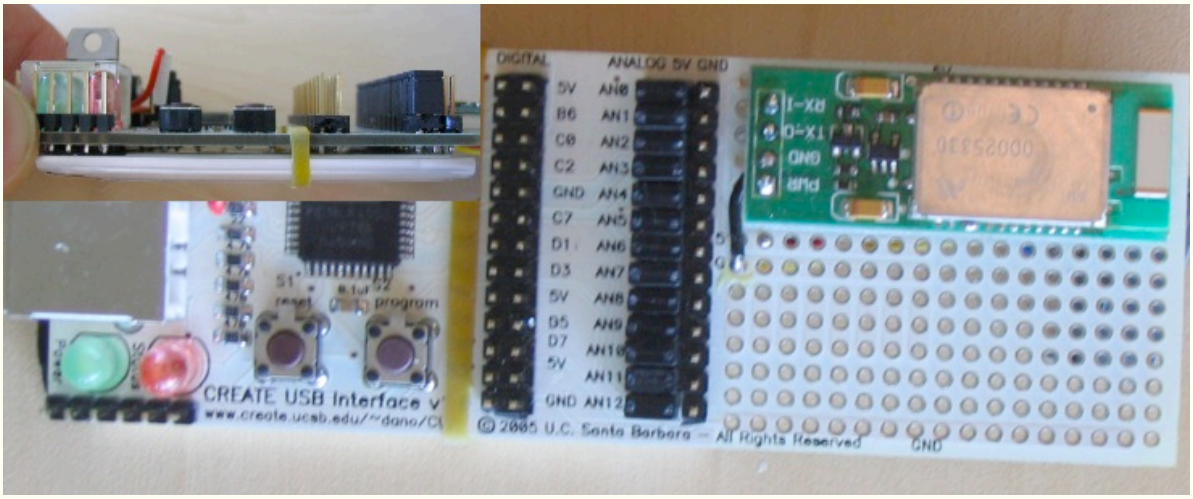
- Program all unused I/O pins to be outputs. This way, you can set the state of a particular I/O pin to be TRUE when you enter a routine and FALSE when you leave it. At the very least, this practice, along with an oscilloscope, can give you confirmation that a particular piece of code is being executed.
- Make the board do something visible or audible for help debugging. In the case of the CUI 0.9, consider adding different color LEDs for status indicators, and turning them on or off before or after the different parts of your program are executed.

5.3 Burning the bootloader

- As mentioned before, the very first time you program your CUI it has to be done with the ICD2 programmer instead of PDFSUSB.EXE. This is because the PIC is completely blank from the factory, and you must program the hex file for the bootloader into the PIC in order to enable future programming sessions to be done via PDFSUSB.EXE.
- The bootloader hex file is located in the CUI Example Code that you downloaded earlier. In MPLAB, open the hex file C:\CUI\Boot_output\MCHPUSB.hex, select the ICD2 as the programmer, and click program device. This bootloader is slightly different from the bootloader that can be downloaded directly from Microchip's website, so make sure to use this one (Microchip's will *not* work with the CUI circuit).
- To enter bootloader mode, hold down the program button on the CUI while plugging in the USB cable, or resetting the board. Your computer will not recognize the bootloader firmware, so you must install the correct driver (only necessary this one time). When windows says "Found New Hardware", click "No, Not This Time", then choose "Install from List". Then click on "Browse..." and choose the directory "C:\CUI\MCHPUSB Driver\Release" - then "Next", "Continue Anyway", and "Finish". Well, that's it - you've got a working CREATE USB Interface now, to be modified and customized to do whatever you want!

Pre-made, ready-to-use CUI v1.0

Feel free to email me if you're interested in a getting a pre-built/bootloaded CUI v1.0 for $50. It comes ready to use with Max/MSP/Jitter, SuperCollider, or any other program that receives USB HID data - just hook up your own sensors to the 13 analog inputs (10-bit resolution) and buttons/switches to the digital inputs, etc... You can also get ready made sensors from the phidgets website, and of course you can re-program it using the bootloader (no need for a PIC programmer), and use the prototyping area to do whatever you want.



CUI v1.0 can be custom-ordered with Bluetooth for wireless functionality (works with Max/MSP and Pd, same features as wired version), powered by rechargeable Li-Ion battery!

Also, a forum has been started for those working with the CUI v1.0: CUI ideas and support forum

Plus a couple of mailing lists for those working with the CUI: CUI-users subscribe here and CUI-dev subscribe here

Links to example Max/MSP/Jitter, Pd, and SuperCollider Patches:
USB CUI for Macintosh
USB CUI for Windows
Bluetooth CUI for Macintosh and Windows
Bluetooth CUI for PD
USB CUI for for Max4.6 - on intel Macs with Max/MSP 4.6, the [hi] object seems to have changed, this firmware update and corresponding test patch resolves the problem. Also includes "CUI IO v1.0" firmware for compatibility with [hidio] object for both Max and Pd (currently in progress 17 January 2007).
SuperCollider3 HIDControl - port of Jan truzschler's HIDControl Ugen to intel Macs - also removes the autoscaling to avoid clicks.

Here are some new and useful tools for the CUI made by users!
CUIOSC by MarkDavid Hosale - a stand alone OS X program that converts the CUI's USB input to OSC (Open Sound Control).
Boot Down by Craig Schimmel - a CUI bootloader utility for OS X that allows firmware updates without a windows box (be careful not to overwrite the bootloader itself though).

App A

Related Links

Publications about the CUI

- Musical Interaction Design with the CREATE USB Interface: Teaching HCI with CUIs instead of GUIs - presented at the International Computer Music Conference (New Orleans, USA, 6-11 November 2006).

Microcontroller Info

- PIC18F4550 - information at Microchip.com (datasheet, help forums, etc).
- C18 User's Guide - Microchip's C18 C Compiler User's Guide.
- C18 Compiler Libraries - References specific C function calls to access on-chip hardware peripherals,

etc.

USB Info

- The USB specification - more info than you really need, chapter 9 is the most relevant.
- HID Usage Tables - setting up a HID device, the gamepad example on page 141 is particularly useful.
- HID Descriptor Tool - a useful GUI tool for constructing HID usage tables that enumerate correctly.
- The HID Page - a good resource with example code, etc. by Jan Axelson the author of *USB Complete*.

Host Info

- USB Sniffer/Analyzer - USB debugging tool that lets you see bytes transfered between the host and the device.
- HID Explorer - if you use Mac OS X, this debugging tool allows you to examine device properties and data.
- USB Monitor - a useful utility for OS X; notifies you whenever a USB device is added or removed.

Parts Supply Houses

- Digikey: Great online ordering. Fast, costs a little bit more.
- Mouser: About like Digikey. Fast, also costs a little bit more.
- Allied Electronics: Great service, slightly slower.
- Hosfelt: Great surplus supplier

...[Media Interface Technology]
...[Dan Overholt - email]