# PIC Tutorial

For more information, refer to the CCS internal help or the CCS S manual stored at D:\Documents\Dropbox\MSCP-elec share\Manuals (“MSCP-elec share” may be referred to as “Battery Protection Share” for some people).

1. Download/install MPLab IDE 8.8 from <http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1406&dDocName=en019469>
2. Copy over the CCS folder from the solarcar server. If you are at home you may need to VPN into the McMaster network (instructions at <http://www.mcmaster.ca/uts/network/vpn/>)

* CCS folder is at [\\pc-solarcar.eng.mcmaster.ca\Files\Software\CCS PCWHD v4.120](file:///\\pc-solarcar.eng.mcmaster.ca\Files\Software\CCS%20PCWHD%20v4.120)

1. Installing CCS

* run pcwhdupd.exe
* it will ask for registration files. Don’t give any registration files and complete the installation
* run the keygen (in the CCS folder) and place files in the main part of the PIC C folder (C: -> program files (x86) -> PICC, unless you put it somewhere else). The keygen produces the registration files and keeping them in the PICC folder will enable them.
* run ideutilsupd.exe (in CCS folder)
* download and install the MPLab plug-in for CCS found here: <http://www.ccsinfo.com/downloads.php>
* to check if everything worked correctly:
* open CCS and attempt to create a new project using the PIC wizard.
* If your registration files (.crg) are no good you will not be able to select any PICs from the dropdown list. You can fix this by copying the .crg’s from \\pc-solarcar.eng.mcmaster.ca\Files\Software\CCS PCWHD v4.120\example crg into your PICC folder. You may need to restart CCS to complete the fix

1. Creating a new project
   * Open CCS (shortcut is called “PIC C Compiler”). From the “Project” menu or ribbon, click on “PIC Wizard”. It will ask you where you want to save your new project. Making a new folder for it is suggested as the compiler will create a lot of files.
   * If you want the project to be accessible to other team members, locate the new folder in D:\Documents\Dropbox\MSCP-elec share\CCS (“MSCP-elec share” may be referred to as “Battery Protection Share” for some people)
   * The PIC Wizard will now open and you will be presented with the “General” tab. The box on the left side of the wizard allows you to select between several tabs. In order to get a basic blinker working, you will only need the “General” tab and possibly the “Intr Oscillator Config” tab. The others are described here if you’re interested.
     1. General: Here you can select your PIC (e.g. PIC16F883) and set your oscillator frequency (most common are 20 MHz external crystal or 8 MHz internal RC). Remember to select the oscillator you are using from the drop-down menu. If you are given the option to disable the watchdog timer, you should do this here as well.
     2. Examples: This should create for you a basic LED blinker code on a pin that you select. Disclaimer: may not work.
     3. Analog: Used to set up analog inputs (use your PIC for measuring voltages). Select which pins you want enabled as analog inputs. If your PIC has a 10-bit ADC (most), use ”Units: 0-1023”; if it is 12-bit use “Units: 0-4095”.
     4. Communications: Used to set up some types of digital communication including RS232 (aka UART or serial port) and I2C. RS232 is often used for communicating with a PC, while I2C is used for communicating with other chips.
     5. SPI: Used to set up Serial Peripheral Interface, which is another type of digital communication. SPI can also be used to communicate with shift registers.
     6. Drivers: Allows you to include libraries for interfacing with several other common chips (EEPROMs, real-time clocks, A/D converters etc.). Unless you are using one of the chips referenced in this tab, you can skip it.
     7. Header files: Allows you to include standard C libraries (string.h, math.h, etc.). Note that complex math operations (square root, logarithm, trigonometry etc.) will be very slow on a PIC, as is floating point (decimal support).
     8. High/low voltage: The PIC has a module called the HLVD (high/low-voltage detect) that is used to monitor the power supply voltage. This tab is used to enable and configure it.
     9. Interrupts: An interrupt service routine (ISR) is a block of code that is triggered in response to some event (the interrupt). You can set up code that executes at a fixed time interval (timer interrupt) or when a button is pressed (external interrupt), for example. There are many other interrupts as well.
     10. Intr Oscillator Config: This tab allows you to enable the PIC’s internal oscillator and set its operating frequency. Used when you do not need the precise timing of an external crystal for a system clock.
     11. I/O Pins: Allows you to define different pins as digital inputs or outputs. Usually skipped because the digital I/O routines in CCS will set the pin’s “direction” (input or output) as necessary when they are called.
     12. Timers: Used to configure the PIC’s onboard counters. A counter is a simple module holding a number that increases its value by one at some fixed time interval. Used for timing external events, triggering periodic interrupts, and so on.
     13. PCH Timers: Allows configuration of an extra two counters only available on PIC18s.
     14. LCD (Internal): On PICs that have an onboard LCD segment driver, this tab is used to set it up. Does not apply to any PICs in current Solar Car use.
     15. LCD (External): Allows you to include a driver for standard 14-pin LCDs (using a 7-pin interface). We generally use the flex\_lcd.c driver instead.
     16. Capacitive Touch: Configuration for capacitive touchscreen interface module. Not tested by Solar Car yet.