How to Get Started Programming the EDU/VEX/FRC Robot Controllers

David Giandomenico
Parent Mentor for Team #846 – The Funky Monkeys
Lynbrook High School Robotics Program, San Jose CA
<dgiandomenico@lynbrookrobotics.com> Tel:(408)343-1183

Welcome to the WRRF Workshop in programming the EDU/VEX/FRC robot controllers. In this workshop you will learn how to control your robot's motors using both potentiometers and switches as inputs. You will even learn how to automate your robot. Along the way you will be learning basics about the 'C' programming language – just enough to get you started.
The code used in this workshop is available for viewing and download at LynbrookRobotics website.
See:

http://lynbrookrobotics.com/robotcode/
http://lynbrookrobotics.com/robotcode/robot_code.htm

The EDU, VEX, and FRC controllers are all very similar, each being based on the same microcontroller, and each control board having been designed by the same manufacture, Innovation FIRST, Inc. Input and output variables differ only slightly between the three controllers. We will be using the EDU controller (a.k.a. MINI controller) because it is compact and because every FIRST team received one.

Introduction
Since 2004, robots in the FIRST Robotics Competitions use the 'C' programming language. 'C' is well suited to access a microcontroller at the hardware level, allows powerful coding, and generates compact code. Higher level languages, such as C++ or Java require more support code or lack the ability to program at the hardware level. Students who know C++ or Java should find the transition to 'C' straightforward. The greatest hurdle is creating a complex system to control motors and sensors in a short period of time.

FIRST Robots use IFI Robotics' FRC Control system, which is composed of an operator interface for the joysticks and the robot-side controller. The two halves are linked by a two-way data radio modem. The robot-side controller contains two MICROCHIP PIC18F8520 microcontrollers, in the roles of a Master and User controller. From the programmer's viewpoint, only the User controller is visible and provides access to all of the needed robot functions. The IFI Robotics' FRC Robot Controller has up to 32KBytes of memory and an operational speed of 10MIP

The C compiler must be written specifically for the microcontroller used in the robot controller, which in this case, is the MICROCHIP PIC18F8520. Since the robot controller's instruction set differs from the instruction set of the CPU in a desktop computer, common C compilers such as MS Visual C cannot be used. FIRST teams use MICROCHIP's free MPLab IDE in combination with MICROCHIP's commercial ($$) MPLab C18 compiler. MPLab software operates on a MS Windows based computer, and the compiled code is then downloaded through a serial cable to the robot controller.

Code Resources
The primary code sources for FIRST robots are available through IFI Robotics' and Kevin Watson's (of NASA's Jet Propulsion Laboratory) websites. IFI Robotics provides default code for the current and prior years, while Kevin Watson's site contains the best collection of code for higher level control features, such as navigation, gyros, encoders, and camera's.

Kevin Watson's Code Repository: **http://kevin.org/frc/**
Code for navigation, gyros, encoders, and camera's & links to MICROCHIP documents:

IFI Robotics
Default code: http://www.ifirobotics.com/rc.shtml#Programming
Code from prior years: http://www.ifirobotics.com/first-legacy.shtml

Software
MPLab IDE - free download from MICROCHIP (www.microchip.com). Search for download in development tools area. The IDE (Integrated Development Environment) allows you to view and edit source code. However, to compile the code, you need the add-in "MPLab C18" 'C' compiler.

MPLab C18 Compiler - commercial product provided at no cost to FIRST teams. Contact your team's software manager.

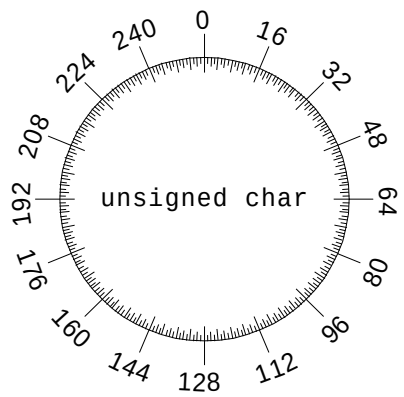
IFI Loader - IFI Robotics (bottom of page: http://www.ifirobotics.com/rc.shtml); MS Windows based software used to upload the compiled code from the PC to the IFI FRC Robot Controller.

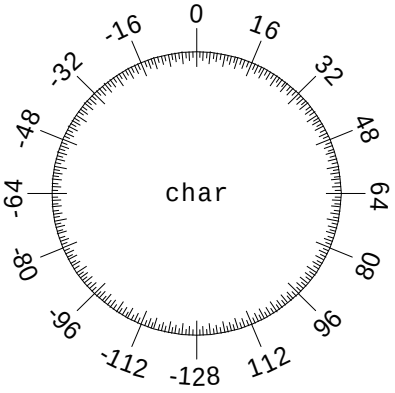
Dashboard - IFI Robotics; Optional very useful MS Windows based utility software used to display data sent from the FRC robot controller to the FRC operator interface (FRC only).
http://www.ifirobotics.com/dashboard_viewer.shtml

'C' Language References:
The C Programming Language, 2nd edition. Brian W. Kernighan and Dennis Ritchie
(Written by the authors of the 'C' programming language itself)

Online (www) 'C' references: Numerous. **Try a google search ('C' reference).**

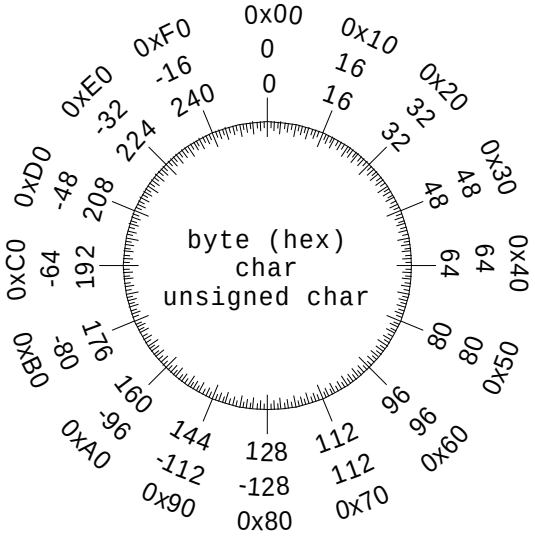
Integer Types in ‘C’






```
Range: {0,255}                    Range: {-128,+127}
Rollover Point: 255+1 = 0         Rollover Point: 127+1 = -128
```

Integer types in ‘C’ use modulus arithmetic, like the face of a clock.



Whether an integer type variable is `signed` or `unsigned`, it is stored in the same way: in binary. In the figure on the left, the outer ring of numbers shows the binary value (represented in hexadecimal as 0xNN) for an 8-bit byte, while the two inner rings show the corresponding `signed` and `unsigned` decimal values.

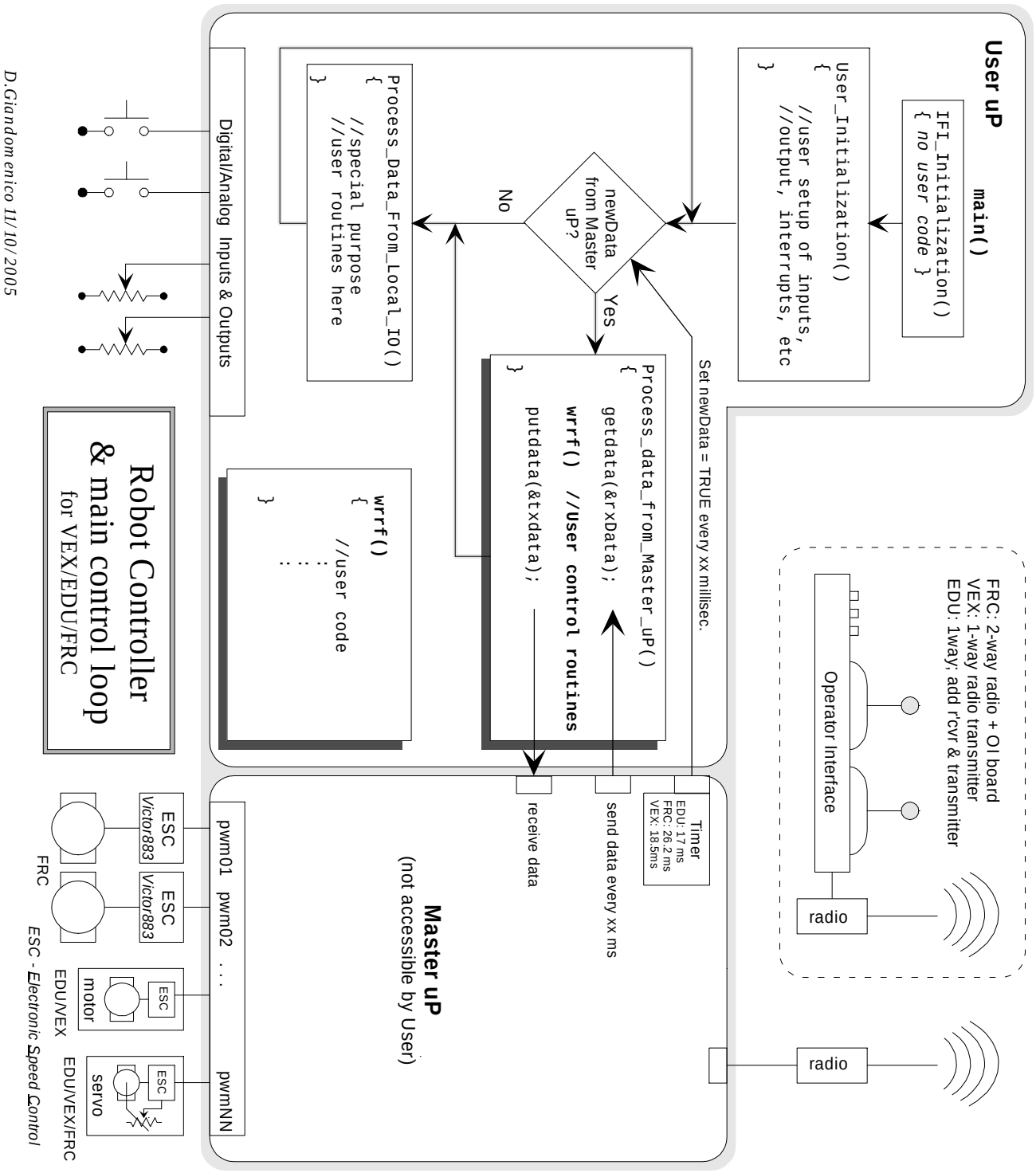
Notable signed `char` numbers:

```
    -1 (dec) = 1111 1111 b = 0xFF
  -128 (dec) = 1000 0000 b = 0x80
```

The Size and Range of Integer Types in ‘C’

	Size (bytes)	# of values	Range unsigned	Range signed - default
char	1	2^8 = 256	**0 to 255**	**-128 to +127**
short	2	2^{16} = 65,536	0 to 65,535	-32,768 to +32,767
int	2*	2^{16} = 65,536	0 to 65,535	32,768 to +32,767
long	4	2^{32} = 4,294,967,296	0 to 4,294,967,296	-2,147,483,648 to +2,147,483,647

*`int` may be 2 or 4 bytes. In the C18/MPLab compiler, an `int` is 2 bytes.



In the EDU/VEX/FRC controllers, data from the *Operator Interface* is collected by the *Master uP* (microprocessor) every 17 – 26.2ms, or 38-59 Hz (times per second) depending on the controller type. The *User uP* is notified that data is ready when the *Master uP* sets the variable `statusflag.NEW_SPI_DATA` true. The main control loop, then executes `Process_data_from_Master_uP()`, which retrieves the new input data from the Master uP, calls the user's control routine (here shown as `wrrf()`) and then sends motor control data back to the Master uP. Onboard data from switches and sensors is immediately available to the User uP without waiting for the Master uP.