






Team Ford First   

Advanced Programming and Sensors




Bob Koehl
Team Ford FIRST
ace4bert@peoplepc.com

Team Ford First   

Major Sources of C Documentation

[Programming Reference Guide](#)
[main.c](#)
[user_routines.c](#)
[user_routines_fast.c](#)
[printf_lib.c](#) (other format % symbols)




November 2005 2

Team Ford First   

Review Integer data types

char -128 to +127
unsigned char 0 to 255
int -32,768 to +32,767
unsigned int 0 to 65,535
long -2,147,483,684 to +2,147,483,683
unsigned long 0 to 4,294,967,295
short long -8,388,608 to +8,388,607
unsigned short long 0 to 16,777,215




November 2005 3

Team Ford First   

Input vs Output

- The term "Input" and "Output" is from the perspective of the controller.
- An "Input" is "To" the controller from the robot or the operator interface.
- An "Output" is "From" the controller to the robot or the operator interface.




November 2005 4

Team Ford First   

Digital

- Digital is a device that has one of two possible states:
 - "ON" State
 - "OFF" State
- The Value of the ON State is either 0 or 1.
- The Value of the OFF State is the opposite of the ON State.

November 2005 5


Team Ford First   

Review OI IO

OI (the Operator Interface panel) has (4) ports. Each port may have a joy stick or custom IO (Inputs and Outputs) connected.

Inputs are either Digital or Analog:
Digital: value of 0 or 1
0 == NOT pressed; 1 == pressed
Analog: value between 0 and 255
Default variable names are defined in the document:
[FRC_RC_Default_Software_Ref_Guide](#)


November 2005 6

Team Ford First 

OI Digital States

- Operator Interface (OI) Device States:
 - "ON" when the Condition == 1;
 - "OFF" when the Condition == 0;
- Inputs include: *p1_sw_trig*, *p1_sw_top*; *p1_sw_aux1*; *p1_sw_aux2*, etc. for 4 ports.
- Outputs include: *Pwm1_green*; *Pwm1_red*; *Pwm2_green*; *Pwm2_red*; *Relay1_green*; *Relay1_red*; *Relay2_green*; *Relay2_red*; *Switch1_LED*; *Switch2_LED*; *Switch3_LED*

November 2005 7


Team Ford First 

Other OI Digital Outputs

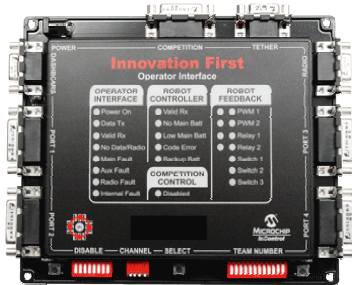
There is a 4 digit display at the bottom of the OI that will display 4 different items based on how many times the "Select" button is pressed:

1. Team Number
2. Channel Number (begins with lower case "c")
3. Battery Voltage (has a decimal point)
4. User Display (begins with lower case "u" and is programmable through the variable named *User_Mode_Byte*). In this mode, all Robot Feedback lights on the OI are turned OFF.


November 2005 8

Team Ford First 

View of Operator Interface (OI)




November 2005 9

Team Ford First 

RC Digital States

- Robot Controller (RC) INPUT Device States:
 - "CLOSED" when the Condition == 0;
 - "OPEN" when the Condition == 1;
- Inputs are named *rc_dig_in01* through *rc_dig_in16*.
- The first 6 inputs (*rc_dig_in01* through *rc_dig_in06*) can be configured as INTERRUPTS for very advanced features.
- Any INPUT may be configured as an OUTPUT. See *User_Initialization()* in *user_routines.c*
- OUTPUTS are named *rc_dig_out01* through *rc_dig_out16* and are 1 when ON and 0 when OFF and may be used to control 5 Volt DC devices.


November 2005 11

Team Ford First 

Other RC Digital States

- Other RC digital outputs are: *relay1_fwd*; *relay1_rev*; *relay2_fwd*; *relay2_rev*; etc. through *relay8_fwd*; *relay8_rev*.
- When connected to "SPIKE" relays they can be used to control motors, pneumatic (air) control valves, or any other 12 Volt DC device.
- Setting the relay output = 1 turns ON the SPIKE and = 0 turns OFF the SPIKE.




November 2005 12

Team Ford First 

Analog

- Analog Devices can have any voltage value between 0 and 5 volts, not just the ON or OFF of a Digital Device.
- The OI controller (which we do not program) uses an analog to digital converter the change the voltage to number that represents the 0 to 5 volts at the device.
- **OI Analog** values include: *p1_x*; *p1_y*; *p1_wheel*; *p1_aux*; *p1_aux1*; *p1_aux2*; etc. for each of ports 1 through 4 on the Operator Interface.
- **OI Analog** are of type *unsigned char* which means the values are between 0 and 255.




November 2005 13

Team Ford First   

RC Analog Inputs

- There are 16 Analog Inputs on the Robot Controller (**RC**).
- The *connection point* for each of those inputs are defined as *rc_ana_in01* through *rc_ana_in16*.
- To read the value of what at that connection point we need to use the function *Get_Analog_Value(connection point);*
- This will convert the voltage at the connection point to a 10 bit (0 to 1023) value representing that voltage. A variable of type *unsigned int* should be used when reading the value.




November 2005 14

Team Ford First   

RC Outputs




- There are 16 PWM Outputs on the Robot Controller.
- When connected to a Victor 884 controller, a motor can be speed controlled from full speed reverse (0) to full speed forward (254) and any other speed including stop (127).
- The *connection point* for each of those outputs are variables defined as *pwm01* through *pwm16*.
- These outputs are of variable type *unsigned char* with a value between 0 and 255.

November 2005 15

Team Ford First   

Other Possible Analog Inputs

November 2005 16




Team Ford First   

Potentiometer

Potentiometer is an analog device that generates a voltage that tells how many degrees the pot has been rotated.

0 Volts means the pot is fully turned one direction.
5 Volts means the pot is full turned the other direction.
Standard pots have 270 degrees of rotation.
Pots may be multi turn (5 or 10 turns to go from 0 to full) or continuous turn (back to 0 every 360 degrees of rotation).




November 2005 17

Team Ford First   

Potentiometer Usage

How can a potentiometer be used on a robot?
What if we wanted to scale the maximum speed of the robot to something other than 0 to 254 yet still use the full range of the joystick?
What if we wanted to hold the position of a robot arm that wanted to move due to gravity?
What if we wanted to stop a robot arm and there was no way to install limit switches?

November 2005 18


Team Ford First   

Gyro

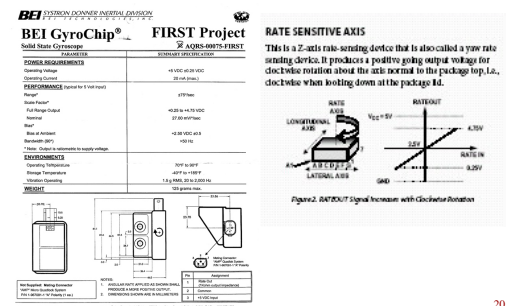
Gyro is an analog device that generates a voltage that is proportional to the number of degrees per second the device is rotating.

The gyro nul value is the voltage from the device when it is not rotating.
The gyro full scale value is the voltage when the gyro is rotating in one direction at its maximum rate.


November 2005 19

Team Ford First 

Gyro Data Sheets



20

Team Ford First 


Gyro Usage

Gyro provides a voltage near 2.5 Volts when not rotating.

Gyro provides a voltage near 4.75 Volts when rotating clockwise at 150 deg/sec.

Gyro provides a voltage near 0.25 Volts when rotating counter clockwise at 150 deg/sec.

November 2005 21

Team Ford First 

Gyro Usage


How can we use a device that tells us how many degrees per second it is rotating?

What would you expect the gyro input to be if we are driving the robot straight forward?

What would you expect the gyro input to be if we are doing a "graceful" turn clockwise?


What would you expect the gyro input to be if we are doing a "spin" counter clockwise?

November 2005 22

Team Ford First 

When are we going to talk about Programming???

November 2005 23

Team Ford First 

Review Program Scan


The "Default Program" is the "endless loop" of the following:

```

while (1) /* This loop will repeat indefinitely. */
{
  if (statusflag.NEW_SPI_DATA) /* 26.2ms loop 38.2 loops/sec.*/
    Process_Data_From_Master_uP(); // located in user_routines.c
  // Process_Data_From_Master_uP() calls the Default_Routine()
  if (autonomous_mode)
    User_Autonomous_Code(); /* Located in user_routines_fast.c */
  Process_Data_From_Local_IO(); /* Located in user_routines_fast.c */
  // Can be used for processing Robot IO that changes very rapidly
} /* end of while (1) */
    
```

[2004 Programming Reference Guide](#)

November 2005 24

Team Ford First 


Using Robot Analog Inputs

```

/* Example of using the Analog Inputs to map an analog input to
a PWM output. */
unsigned int sensor1;
unsigned char sensor1_8bits;
sensor1 = Get_Analog_Value( rc_ana_in01 ); /* Assign the 10
bit analog reading to an integer variable. */
sensor1_8bits = (unsigned char)(sensor1 >> 2); /* Only take the
8 most significant bits, */ >>2 is the same as dividing by 4
pwm01 = sensor1_8bits; /* because PWM OUTPUTS can only
be 8 bits. */
    
```

[2004 Programming Reference Guide](#)

November 2005 25

Team Ford First 


Review *if* decisions

```

if (condition is TRUE) //NO semi colon here
{
    command;           //executed if the condition is true
    command;           //executed if the condition is true
}
//} end of if condition is true

== (double =) equal   if( p1_x == 127) if(sensor1 < 920)
!=not equal          if( p1_y != p1_y_old)
> greater than       if( p1_y > 107) if(sensor1 > 36)
>= greater than or equal if( p1_x >= 108) if(sensor1 >= 54)
< less than          if( p1_y < 137)
<= less than or equal if( p1_x <= 138)
    
```

November 2005 26


Team Ford First 

multiple *conditions*

OR conditions use the symbols **||**
if(p1_sw_trig == 1 || p1_sw_top == 1)
 If **either** the trig is equal to 1 **or** the top is equal to 1
 the condition is TRUE

AND conditions use the symbols **&&**
if(p1_sw_trig == 1 && p1_sw_top == 1)
 If **both** the trig is equal to 1 **and** the top is equal to 1
 the condition is TRUE

November 2005 27

Team Ford First 

if ... else commands

Take the *if*(condition is TRUE) command;


Expand to an *if ... else* command:

```

if(condition is TRUE) command_1;
else command_2;
    
```

If the condition is TRUE, perform command_1.
 else, since the condition is NOT TRUE,
 perform command_2

November 2005 28

Team Ford First 

Make an output follow an input

```


if(condition)
    turn_ON_device;
else
    turn_OFF_device;
    
```

Why would we learn to do this?

This programming method is a simplest way to "make something happen" ONLY while a button is held.

It is often used to "jog" a motor that may be connected to a robotic hand or gripper.

November 2005 29

Team Ford First 

Output stays ON until told to turn OFF

Stay ON until turned OFF

<pre> if(trig) turn_ON; else if(top) turn_OFF; </pre>	<pre> if(trig) turn_ON; if(top) turn_OFF; </pre>
---	--


Can this be done without using an *if* statement?

```

output = (trig | output) & ~top;
    
```

"|" is logical OR; "&" is logical AND; "~" is logical NOT

November 2005 31

Team Ford First 

Select Paths with *switch*




While *if* is good for choosing between two alternatives, it quickly becomes cumbersome when several alternatives are needed.

C's solution to this problem is the *switch* statement. The *switch* statement is C's multiple selection statement. It is used to select one of several alternative paths in program execution and works like this:

A variable is successively tested against a list of integer or character constants.

When a match is found, the statement sequence associated with the match is executed.

November 2005 32




Team Ford First   

switch ...case

The general form of the **switch** statement is:

```
switch(expression)
{
  case constant1: statement sequence; break;
  case constant2: statement sequence; break;
  case constant3: statement sequence; break;
  ...
  default: statement sequence; break;
}
```

November 2005 33




Team Ford First   

switch ...case

Each case is labeled by one or more constant expressions (or integer-valued constants). The **default** statement sequence is performed if no matches are found. The **default** is optional. If all matches fail and **default** is absent, no action takes place.

When a match is found, the **statement sequence** associated with that **case** are executed until **break** is encountered.




November 2005 34

Team Ford First   

switch ...case

```
switch (i)           //based on the value of i
{                   //start of switch
  case 1: printf("one"); break;
  case 2: printf("two"); break;
  case 3: printf("three"); break;
  case 4: printf("four"); break;
  default: printf("unrecognized number");
}                   //end of switch
```

November 2005 35

Team Ford First   

Programming Assignment

The assignment for next session:




We are using the Default Code to drive our robot with a single joystick (*p1_y* "limit mixed with" *p1_x*).

The right motors are connected to *pwm13* and *pwm14*.
The left motors are connected to *pwm15* and *pwm16*.
We have a 360 deg pot connected to *rc_ana_in01*.
We have a 75 deg/sec gyro connected to *rc_ana_in16*.
We have a 270 deg pot connected to *rc_ana_in13*.

When the robot is being commanded to drive straight, it has a tendency to drift off line.

Fix it so it goes straight when commanded to go straight.

November 2005 36

Team Ford First   

Programming Assignment

Questions to ask:

How do we know if we are trying to drive in a straight line?




How will we know if we are NOT driving in a straight line?

- Will the pot help?
- Will a gyro help?

How will we adjust the motor speeds to try to get us back to a straight line?

- Can we use the gyro or the pot to slow down the motors on one side and speed up the wheels on the other side?

November 2005 37

Team Ford First   

Trying to drive a straight line

How do we know if we are trying to drive in a straight line?




- *p1_x* is approximately equal to 127?
- left motor speeds are close to right motor speeds?

Either will work!!

What will that look like in the program?

```
if(p1_x > 120 && p1_x < 134)
if(abs(pwm_left - pwm_right) < 6)
```

November 2005 38

Team Ford First   

Not driving straight

If the chassis is rotating, then the robot is NOT going straight.

Use the gyro to detect that the robot chassis is rotating.

Read the gyro into an *unsigned int*:




```
gyro_int = Get_Analog_Value(rc_ana_in16);
```

gyro_int will be a value between 0 and 1023.

Remember the gyro has a "nul" value near 512 when the chassis is NOT rotating.

If *gyro_int* is greater than the "nul" the chassis is rotating Clockwise; if *gyro_int* is less than the "nul" the chassis is rotating Counter Clockwise.

November 2005 39

Team Ford First   




Chassis Rotating Clockwise

If the Chassis is rotating Clockwise (*gyro_int* greater than "nul"), what should be done to straighten the rotation?

Rotating Clockwise means the Left motors are going faster than the Right motors.

To stop the rotation, the Left motors need to slow down AND the Right motors need to speed up.

November 2005 40

Team Ford First   




Chassis Rotating Counter Clockwise

If the Chassis is rotating Counter Clockwise (*gyro_int* less than "nul"), what should be done to straighten the rotation?

Rotating Counter Clockwise means the Right motors are going faster than the Left motors.

To stop the rotation, the Right motors need to slow down AND the Left motors need to speed up.

November 2005 41

Team Ford First   

Determine the gyro "nul"

How do we get this gyro "nul" value?

One way is to read the gyro during the program initialization phase.

In [user_routines.c](#) (line 162) in the function [User_Initialization\(\)](#), there is room to add any initialization we need.

```
gyro_nul = Get_Analog_Value(rc_ana_in16);
```




gyro_nul has been setup as an (*unsigned int*).

Another way is to assign a value as follows:

```
unsigned int gyro_nul = 526;
```

Experiment to find the nul value using a printf() statement, Dashboard Viewer, or the OI LED.

November 2005 42

Team Ford First   

Calculate the Rotation

Define a variable of type *int* to calculate the amount of rotation plus or minus from the nul.




```
gyro_rate = gyro_int - gyro_nul;
```

This is a plus or minus integer value.

gyro_int can be from 0 to 1023.
gyro_nul will be around 512 (+/- 50)
gyro_rate can be between -512 and +511

The rate is plus if the chassis is rotating Clockwise; the rate is minus if the chassis is rotating Counter Clockwise.

November 2005 43

Team Ford First   

Determine motor speeds

Remember we said if the chassis rotates Clockwise, we slow down the Left and speed up the Right.

gyro_rate is **positive** for Clockwise rotation.




```
Left = Left - Scale_factor * gyro_rate;
```

```
Right = Right + Scale_factor * gyro_rate;
```

When *gyro_rate* is **negative** for Counter Clockwise rotation, the equations automatically change which motor gets slowed and which motor goes faster because of the sign on *gyro_rate*.

This just seems too easy, doesn't it???

November 2005 44

Team Ford First   

Define *scale_factor*

scale_factor is what really makes the robot respond to the gyro without over or under reacting.




Too high a *scale_factor* will cause the robot to over correct and weave back and forth.

Too low a *scale_factor* will result in the robot never getting close to running straight.

The *gyro_rate* is giving us a number (-512 to +511) that is much larger than what we are controlling (127 +/- 127 or 0 to 127 to 254)

Let' s try to divide *gyro_rate* by 4 so that it can affect the speed change by (512/4) or +/-128

November 2005 45

Team Ford First   

Limiting Function

Will the function *Limit_Mix()* work?

What if we make the equations:




Left = *Limit_Mix*(2000 + Left - (*gyro_rate*/4));

Right = *Limit_Mix*(2000 + Right + (*gyro_rate*/4));

Will limiting the results to the range of 0 to 254 be good enough?

Probably, so let' s do the whole program for gyro adjusting while trying to drive straight.




November 2005 46

Team Ford First   

Look at the Program

- [user_routine.c](#) is revised to include the gyro.




November 2005 47

Team Ford First   

Other Sensors or Programs to evaluate

- www.kevin.org talks about:
 - [encoders](#) and [gyros](#)
- www.radioshack.com/frctalks about VEX Robot I/O including Switches, Range Finders, Line Tracking Sensors, Optical Shaft Encoders

November 2005 48

Team Ford First   

Questions ????

November 2005 49