

Chuck's Programming Notes

Updated: December 10, 2004

NOTE: All code below should be placed between ...

```

Getdata(&rxdata);
    //your code here
Putdata(&txdata);

```

This applies to both the normal and autonomous modes of operation.

1. Display numerical data to Operator Interface (OI) Panel. Remember to press the Select button on the OI panel so that the lower-case 'U' is displayed in order for this to work.

```
User_Mode_byte = p1_y;
```

p1_y can be any variable displaying 0 to 255.

2. Reading Analog Input from Potentiometer. This will be a value between 0 and 1023 because this is a 10-bit number. An 8-bit number displays a value between 0 and 511.

```

int pot;
pot = Get_Analog_Value(rc_ana_in01);

```

3. Display Analog Input to OI Panel.

NOTE: If the analog input is linear then this should work okay.

```
User_Mode_byte = Get_Analog_Value(rc_ana_in01)/4;
```

4. One-second timer in Autonomous Mode.

```

nCount++;
if(nCount > 38){ //occurs every second in a 26.5 mSec loop
    nCount=0;
}

```

5. One-second timer in Normal Mode Using OI Switch 1 and 2 as a display.

```

static long nCount;
static int bOn;

nCount++;
if(nCount > 5500){
    nCount=0;
    if (bOn >0){
        bOn = 0;
        Switch1_LED = 1;
        Switch2_LED = 0;
    }
    else{
        bOn = 1;
        Switch1_LED = 0;
        Switch2_LED = 1;
    }
}
}

```

6. Toggle OI Switch on and off.

```

if (bAuto==1)
    Switch1_LED = 1;
else
    Switch1_LED = 0;

```

7. Operate joystick Y axis to follow a potentiometer connected to analog input. Control OI Switch LEDs to instruct operator which direction to move the joystick. Zero difference is considered when diff is ± 5 .

```

int pot;
int diff = 0;

pot = Get_Analog_Value(rc_ana_in01)/4;

if (p1_y < pot - 5){
    diff = pot - p1_y;
    Switch1_LED = 0;
    Switch2_LED = 0;
    Switch3_LED = 1;
}
else if (p1_y > pot + 5){
    diff = p1_y - pot;
    Switch1_LED = 1;
    Switch2_LED = 0;
    Switch3_LED = 0;
}
else {
    diff = 0;
    Switch1_LED = 0;
    Switch2_LED = 1;
    Switch3_LED = 0;
}

User_Mode_byte = pot; //operate OI Select Switch to see this

```

8. Joystick position relative to analog input 1 potentiometer is represented by 7 Green LEDs on the OI (PWM1, PWM2, Relay1, Relay2, Switch1, Switch2, Switch3). The 7 LEDs stacked vertical on the OI looks like a level indicator for volume on a stereo.

```

//displays difference between joystick and pot on RC
int pot;
int diff = 0;

Pwm1_red = 0;
Pwm2_red = 0;
Relay1_red = 0;
Relay2_red = 0;

pot = Get_Analog_Value(rc_ana_in01)/4;

if (p1_y < pot - 50){
    diff = pot - p1_y;
    Pwm1_green = 0;
    Pwm2_green = 0;
}

```

```
    Relay1_green = 0;
    Relay2_green = 0;
    Switch1_LED = 0;
    Switch2_LED = 0;
    Switch3_LED = 1;
}
else if (p1_y < pot - 25){
    diff = pot - p1_y;
    Pwm1_green = 0;
    Pwm2_green = 0;
    Relay1_green = 0;
    Relay2_green = 0;
    Switch1_LED = 0;
    Switch2_LED = 1;
    Switch3_LED = 0;
}
else if (p1_y < pot - 5){
    diff = pot - p1_y;
    Pwm1_green = 0;
    Pwm2_green = 0;
    Relay1_green = 0;
    Relay2_green = 0;
    Switch1_LED = 1;
    Switch2_LED = 0;
    Switch3_LED = 0;
}
else if (p1_y > pot + 50){
    diff = p1_y - pot;
    Pwm1_green = 1;
    Pwm2_green = 0;
    Relay1_green = 0;
    Relay2_green = 0;
    Switch1_LED = 0;
    Switch2_LED = 0;
    Switch3_LED = 0;
}
else if (p1_y > pot + 25){
    diff = p1_y - pot;
    Pwm1_green = 0;
    Pwm2_green = 1;
    Relay1_green = 0;
    Relay2_green = 0;
    Switch1_LED = 0;
    Switch2_LED = 0;
    Switch3_LED = 0;
}
else if (p1_y > pot + 5){
    diff = p1_y - pot;
    Pwm1_green = 0;
    Pwm2_green = 0;
    Relay1_green = 1;
    Relay2_green = 0;
    Switch1_LED = 0;
    Switch2_LED = 0;
    Switch3_LED = 0;
}
else {
    diff = 0;
```

```

Pwm1_green = 0;
Pwm2_green = 0;
Relay1_green = 0;
Relay2_green = 1;
Switch1_LED = 0;
Switch2_LED = 0;
Switch3_LED = 0;
}

```

9. There are six user bytes. These are bytes that can be written to by the programmer for the purpose of passing the data to the Dashboard. User bytes 1 and 2 are special and require additional code to program. Bytes 3 through 6 are easier to use.

```

User_Byte3 = 103;    or    txdata.user_byte3 = 103;
User_Byte4 = 104;

//get analog value
User_Byte5 = Get_Analog_Value(rc_ana_in01);

//get joystick
User_Byte6 = (int) p1_y;

//notice that .allbits must be added for bytes 1 and 2
User_Byte1.allbits = 101;    or    txdata.user_byte1.allbits = 101;
User_Byte2.allbits = 102;

```

10. This routine allows the joystick on port 1 to control a motor via pwm01. The motor is prevented from responding instantaneously due to built in time delays.

```

nCount++;
if (nCount > 10){
    nCount = 0;
    if (p1_y > pwm01){
        diff = p1_y - pwm01;
        pwm01+= diff * .25; //increase by __ of difference
        if (pwm01 > 250) pwm01 = 250;
    }
    else if (p1_y < pwm01){
        diff = pwm01 - p1_y;
        pwm01-= diff * .25; //decrease by __ of difference
        if (pwm01 < 5) pwm01 = 5;
    }
    else
        pwm01 = p1_y; //match joystick value
}

```

11. A NO SPST switch connected to a digital input can be used to direct program control. This switch can affect normal mode or autonomous mode of operations.

```

if (rc_dig_in03 == 1){
    //do this if switch is open
}
else{
    //do this if switch is closed
}

```

12. Autonomous mode using predefined 'pwm01' values. First, add one line of code (**BOLD**) to the following in user_routines_fast.c.

```
void User_Autonomous_Code(void)
{
  while (autonomous_mode) /* DO NOT CHANGE! */
  {
    if (statusflag.NEW_SPI_DATA) /* 26.2ms loop area */
    {
      Getdata(&rxdata); /* DO NOT DELETE, or you will be stuck
here forever! */

      /* Add your own autonomous code here. */
      AutoDrive();

      Putdata(&txdata); /* DO NOT DELETE, or you will get no PWM
outputs! */
    }
  }
}
```

Second, create a file named "auto.h" and add it to the project. Add this code to the "auto.h" file.

```
#include "ifi_aliases.h"
#include "ifi_default.h"
#include "ifi_utilities.h"

void AutoDrive (void);
```

Third, create another file named "auto.c" and add it to the project. Add this code to the "auto.c" file.

```
#include "auto.h"

unsigned char p1[] = {127,150,150,150,150,150,150,150,150,150,127};

void AutoDrive (void)
{
  static unsigned char i;
  static unsigned char k;
  unsigned char t = 37;
  if (i < 10)
  {
    k++;
    if(k>t)
    {
      pwm01 = p1[i];
      k = 0;
      i++;
    }
  }
  else
    pwm01 = 127;
}
```

Explanation. The array 'p1' stores 10 values representing desired 'pwm01' commands. The function AutoDrive is called every 26.5 mSec. The variables 'i and k' are remembered because of the keyword 'static'. Every 38 times, 'k is > then t'. The 'pwm01' command is set to the value in the array at position 'i' . This repeats until the array is exhausted. At this time the pwm01 is set to 127. Modify the value of 't' to see the results.

The value of pwm01 will be updated every ($t \times 26.5 \text{ mS}$). In this case $t = 37$. Time is $37 \times 26.5 \text{ mSec}$ or 0.98 seconds (approximately 1 second).