```
//--------------------------------------------------------------------------
// use PID for vt.py
//
// This code controls a stepper motor using 3 PWM signals. The speed and
// direction of the motor is received over UART from the Python code.
//--------------------------------------------------------------------------
#include "project.h"

//union is used to convert data received over UART
 union forUartCon {
        uint8 int8val;
        int intvalue;
    } fuc;

int main(void)
{
    //variables
    float Motor_Speed=0.0;      //steps/s, max speed=1000step/s -> 0.244 rev/s
    int Period=0;               //PWM period, controls speed, 4 steps/period
    int Clock_Speed=10000;      //controls 3 PWM signals
    int Direction=2;            //1 means cw, 2 means ccw
    uint8 Receive;              //UART variable
    unsigned char testA[3];     //stores 3 bytes received through UART
    int m=0;                    //counter variable
    int testB[3];               //stores converted data from UART

    //start 3 PWM signals, UART and LCD
    PWM_1_Start();      //alpha
    PWM_2_Start();      //beta
    PWM_3_Start();      //gamma
    UART_1_Start();
    LCD_Char_1_Start();

    LCD_Char_1_ClearDisplay();  //LCD prompts to run 'PID for vt.py'
    LCD_Char_1_Position(0,0);
    LCD_Char_1_PrintString("Initializing:");
    LCD_Char_1_Position(1,0);
    LCD_Char_1_PrintString("Start Python now");


    //infinite for loop
    for(;;)
    {

    m=0;                                    //counter
    Receive=UART_1_GetChar();               //get UART data
    while (Receive==0)                      //loop waits for UART data
    {
        Receive=UART_1_GetChar();           //continue checking for data
        if (Receive>0)                      //data is received
        {
         while (m<3)                        //python sends 3 bytes of data
```

```
        {
            testA[m]=Receive;           //save data in this array
            m=m+1;                      //increment counter
            CyDelay(30);                //delay 30ms
            Receive=UART_1_GetChar();   //get next byte
        }
        Receive=1;                      //breaks out of while loop
    }
}


//convert 3 bytes to 3 integers, needed for motor speed equation
m=0;                                        //reset counter
while (m<3)                                 //convert 3 bytes
    {
        fuc.int8val=testA[m];               //send byte to union
        testB[m]=fuc.intvalue;              //get integer from union
        m=m+1;                              //increment counter
    }

Motor_Speed=256*testB[2]+testB[1];      //convert 2 bytes to 1 integer
Direction=testB[0];                     //no equation needed for direction

LCD_Char_1_ClearDisplay();      //display the motor speed and direction
LCD_Char_1_Position(0,0);
LCD_Char_1_PrintNumber(Motor_Speed);
LCD_Char_1_Position(1,0);
LCD_Char_1_PrintNumber(Direction);

//use data to set PWM signals (1:alpha, 2:beta, 3:gamma)
Period=4.0*Clock_Speed/Motor_Speed;     //calculate PWM period

PWM_1_WritePeriod(Period);          //set alpha signal period
PWM_1_WriteCompare(Period/2);       //alpha 50% duty cycle
PWM_2_WritePeriod(Period);          //set beta signal period
PWM_2_WriteCompare(Period/2);       //beta 50% duty cycle
PWM_3_WritePeriod(Period/2);        //set gamma signal period
PWM_3_WriteCompare(Period/4);       //gamma 50% duty cycle

if (Direction==1)                   //if data received is clockwise
{
    PWM_1_SetCompareMode(1);        //alpha is low to high in a period
    PWM_2_SetCompareMode(1);        //beta is centeraligned, starts high
    PWM_3_SetCompareMode(3);        //gamma is high to low in a period
    PWM_1_WriteCounter(0);          //keeps 8 steps in sync
    PWM_2_WriteCounter(0);
    PWM_3_WriteCounter(0);
}

if (Direction==2)                   //if data received is counter clockwise
{
    PWM_1_SetCompareMode(3);        //reverse alpha signal -> high to low
```

```
        PWM_2_SetCompareMode(3);        //reverse beta signal -> starts low
        PWM_3_SetCompareMode(3);        //gamma never changes
        PWM_1_WriteCounter(0);          //keeps 8 steps in sync
        PWM_2_WriteCounter(0);
        PWM_3_WriteCounter(0);
    }
    //return to the beginning of the infinite for loop
    }
}
```