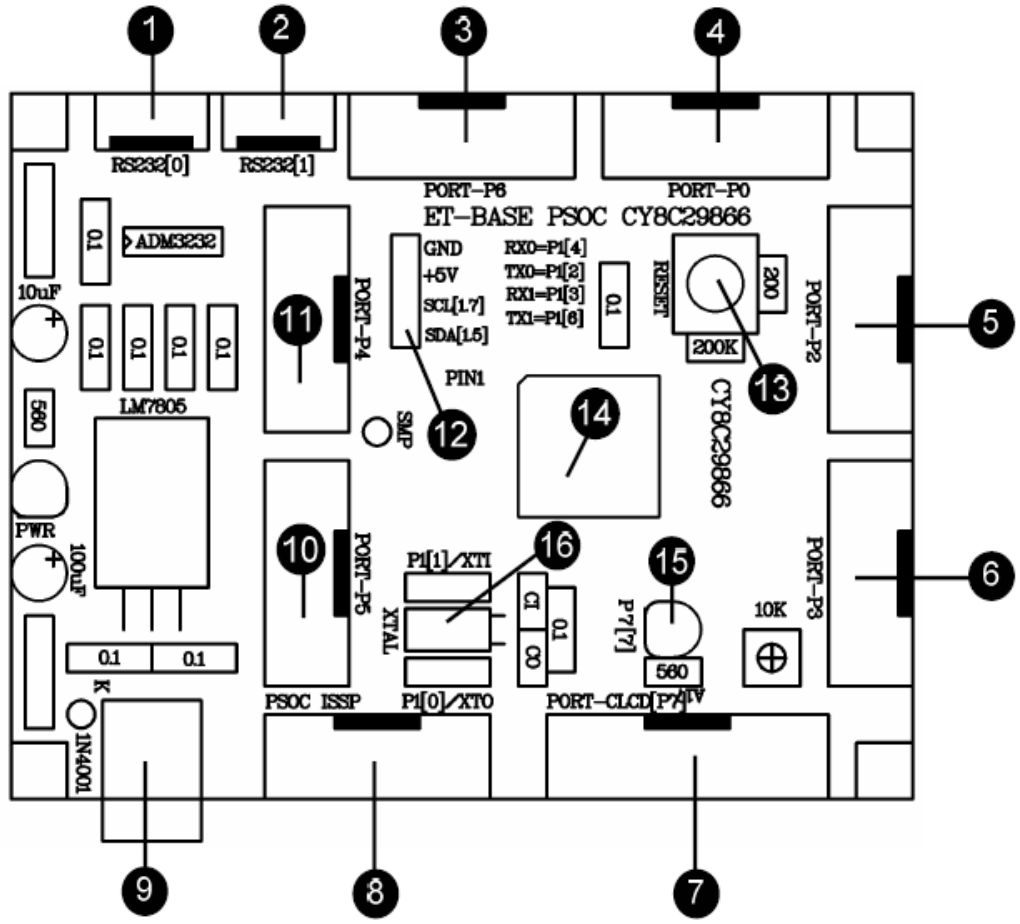
ภาคผนวก ค

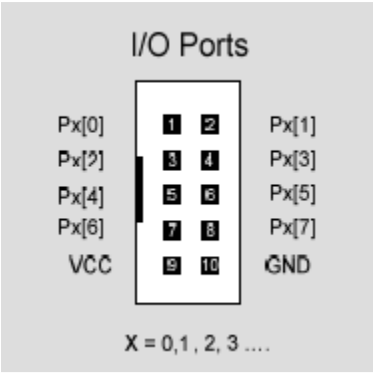
การเขียนโปรแกรมควบคุมความเร็วของ PSoC microcontroller

โครงสร้างบอร์ด PSoC CY8C29866



รูปที่ ค.1 บอร์ด PSoC CY8C29866

หมายเลข 3 ,4 , 5, 6, 10 และ 11 เป็นพอร์ตของขาสัญญาณ I/O ของ PSoC มีการจัดเรียงดังรูป



รูปที่ ค.2 I/O Ports ของบอร์ด PSoC CY8C29866

ตารางที่ ก.1 การกำหนดหมายเลขพอร์ตสำหรับควบคุมความเร็วชุดขับเคลื่อน (Thruster) แต่ละตัว

ชุดขับเคลื่อนที่	1		2		3		4		5		6	
หมายเลขพอร์ต	4	ตัวแปร	5	ตัวแปร	6	ตัวแปร	11	ตัวแปร	10	ตัวแปร	3	ตัวแปร
ขาสัญญาณ PWM	P-0-5	P1	P-2-7	P2	P-3-5	P3	P-4-4	P4	P-5-6	P5	P-6-0	P6
ขาสัญญาณตามเข็ม	P-0-1	L1	P-2-1	L2	P-3-1	L3	P-4-5	L4	P-5-5	L5	P-6-1	L6
ขาสัญญาณทวนเข็ม	P-0-3	R1	P-2-3	R2	P-3-3	R3	P-4-3	R4	P-5-3	R5	P-6-3	R6

การเขียนคำสั่งควบคุมความเร็วชุดขับเคลื่อน (Thruster) ใน PSoC microcontroller

```
//-------------------------------------------------------------------------
// Psoc Control 6 Motors   C main line
//-------------------------------------------------------------------------

#include <m8c.h>        // part specific constants and macros
#include "PSoCAPI.h"    // PSoC API definitions for all User Modules
#include "global.h"                    // Define Input Switch

int DPWM1,DPWM2,DPWM3,DPWM4,DPWM5,DPWM6;          // integer
BYTE Txt_1,Txt_2,Txt_3,Txt_4,Txt_5,Txt_6;
BOOL LR1,LR2,LR3,LR4,LR5,LR6;                     // Left Right
BOOL ERROR;                                       // for Error Status

void main()
{
```

```
char*strPtr;                                        // String Pointer

        UART_1_CmdReset();                          // Initialize receiver/cmd
        UART_1_IntCntl(UART_1_ENABLE_RX_INT);  // Enable RX interrupts
        Timer16_1_WritePeriod(104);                 // Set baud rate 57600 bps
        Timer16_1_WriteCompareValue(52);
        Timer16_1_Start();                          // Turn on baud rate generator
        UART_1_Start(UART_1_PARITY_NONE);       // Enable UART
        M8C_EnableGInt;                             // Turn on interrupt
        ERROR=0;                                    // Initial Error Status
        LR1=0;                                      // Initial Start Turn Right

while(1)
{
        if(UART_1_bCmdCheck())
{
        // Wait for Command
        if(strPtr=UART_1_szGetParam())
{
////////////////////////////////////////////////////////////////////////////////
                if(strPtr[0]=='1')                  // Index of Motor1
{
                        if(strPtr[1]=='L')          // Left/Right Command
{
                        if(strPtr[2]=='0')          // Turn Right
{
                        LR1=0;          // Variable sent for Read Data
                        LR_1_L_Off();  // Enable B Off
                        LR_1_R_On();   // Enable A On (Turn Right)
}
```

```
                    else if(strPtr[2]=='1')      // Turn Left
        {
                    LR1=1;          // Variable sent for Read Data
                    LR_1_L_On();    // Enable B On (Turn Left)
                    LR_1_R_Off();   // Enable A Off
        }
        }
            else if(strPtr[1]=='P')                         // PWM Data Command
        {
DPWM1=(ReadChar(strPtr[2]))*16+(ReadChar(strPtr[3]));       // Get hex char for PulseWidth
if((DPWM1>0)&&(DPWM1<=249))                                 // DPWM1>0 to 249 Start PWM
        {
            PWM16_1_Start();
            PWM16_1_WritePeriod(249);
            PWM16_1_WritePulseWidth(DPWM1);
        }
            else
        {
            PWM16_1_Stop();                         // Stop PWM
        }
            UART_1_PutCRLF();                       // Send "CR" Code
        }
            else if(strPtr[1]=='R')                 // For Read Data
        {
            Txt_1=(ERROR*0x02)+(LR1*0x01);      // Data for Sent to UART
            UART_1_PutSHexByte(Txt_1);          //
            UART_1_PutSHexByte((char)DPWM1);
            UART_1_PutCRLF();
        }
        }
        //UART_1_CmdReset();                        // Reset Command Buffer
```

```
//////////////////////////////////////////////////////////////////////////////
        if(strPtr[0]=='2')                      // Index of Motor2
{
        if(strPtr[1]=='L')                      // Left/Right Command
{
        if(strPtr[2]=='0')                      // Turn Right
{
        LR2=0;                                  // Variable sent for Read Data
        LR_2_L_Off();                           // Enable B Off
        LR_2_R_On();                            // Enable A On (Turn Right)
}
        else if(strPtr[2]=='1')                 // Turn Left
{
        LR2=1;                                          // Variable sent for Read Data
        LR_2_L_On();                                    // Enable B On (Turn Left)
        LR_2_R_Off();                                   // Enable A Off
}
}

        else if(strPtr[1]=='P')                         // PWM Data Command
{
DPWM2=(ReadChar(strPtr[2]))*16+(ReadChar(strPtr[3]));// Get hex char for PulseWidth
        if((DPWM1>0)&&(DPWM1<=249))     // DPWM1>0 to 249 Start PWM
{
        PWM16_2_Start();
        PWM16_2_WritePeriod(249);
        PWM16_2_WritePulseWidth(DPWM2);
}
        else
{
        PWM16_2_Stop();                 // Stop PWM
```

```
}
        UART_1_PutCRLF();                   // Send "CR" Code
}
        else if(strPtr[1]=='R')             // For Read Data
{
        Txt_2=(ERROR*0x02)+(LR2*0x01);      // Data for Sent to UART
        UART_1_PutSHexByte(Txt_2);
        UART_1_PutSHexByte((char)DPWM2);
        UART_1_PutCRLF();
}
}
//UART_1_CmdReset();                        // Reset Command Buffer
  ////////////////////////////////////////////////////////////////////////////////
        if(strPtr[0]=='3')                  // Index of Motor3
{
        if(strPtr[1]=='L')                  // Left/Right Command
{
        if(strPtr[2]=='0')                  // Turn Right
{
        LR3=0;                              // Variable sent for Read Data
        LR_3_L_Off();                       // Enable B Off
        LR_3_R_On();                        // Enable A On (Turn Right)
}
        else if(strPtr[2]=='1')             // Turn Left
{
        LR3=1;                              // Variable sent for Read Data
        LR_3_L_On();                        // Enable B On (Turn Left)
        LR_3_R_Off();                       // Enable A Off
}
}
```

```
                else if(strPtr[1]=='P')                 // PWM Data Command
        {
DPWM3=(ReadChar(strPtr[2]))*16+(ReadChar(strPtr[3]));// Get hex char for PulseWidth
                if((DPWM1>0)&&(DPWM1<=249))     // DPWM1>0 to 249 Start PWM
        {
                PWM16_3_Start();
                PWM16_3_WritePeriod(249);
                PWM16_3_WritePulseWidth(DPWM3);
        }
                else
        {
                        PWM16_3_Stop();                 // Stop PWM
                }
                        UART_1_PutCRLF();               // Send "CR" Code
                }
                        else if(strPtr[1]=='R')         // For Read Data
                {
                        Txt_3=(ERROR*0x02)+(LR3*0x01); // Data for Sent to UART
                        UART_1_PutSHexByte(Txt_3);
                        UART_1_PutSHexByte((char)DPWM3);
                        UART_1_PutCRLF();
                }
                }
                //      UART_1_CmdReset()               // Reset Command Buffer
                ////////////////////////////////////////////////////////////////////////////////
                        if(strPtr[0]=='4')              // Index of Motor4
                {
                        if(strPtr[1]=='L')              // Left/Right Command
                {
                        if(strPtr[2]=='0')              // Turn Right
                {
```

```
                LR4=0;                  // Variable sent for Read Data
                LR_4_L_Off();           // Enable B Off
                LR_4_R_On();            // Enable A On (Turn Right)
            }
                else if(strPtr[2]=='1')     // Turn Left
            {
                LR4=1;                  // Variable sent for Read Data
                LR_4_L_On();            // Enable B On (Turn Left)
                LR_4_R_Off();           // Enable A Off
            }
            }
                else if(strPtr[1]=='P')     // PWM Data Command
            {
DPWM4=(ReadChar(strPtr[2]))*16+(ReadChar(strPtr[3]));// Get hex char for PulseWidth
                if((DPWM1>0)&&(DPWM1<=249))// DPWM1>0 to 249 Start PWM
            {
                PWM16_4_Start();
                PWM16_4_WritePeriod(249);
                PWM16_4_WritePulseWidth(DPWM4);
            }
                else
            {
                PWM16_4_Stop();         // Stop PWM
            }
                UART_1_PutCRLF();       // Send "CR" Code
            }
                else if(strPtr[1]=='R')     // For Read Data
            {
                Txt_4=(ERROR*0x02)+(LR4*0x01);      // Data for Sent to UART
                UART_1_PutSHexByte(Txt_4);
                UART_1_PutSHexByte((char)DPWM4);
```

```
                UART_1_PutCRLF();
            }
            }
            //      UART_1_CmdReset();          // Reset Command Buffer
            ////////////////////////////////////////////////////////////////////////////////
                if(strPtr[0]=='5')              // Index of Motor5
            {
                if(strPtr[1]=='L')              // Left/Right Command
            {
                if(strPtr[2]=='0')              // Turn Right
            {
                LR5=0;                          // Variable sent for Read Data
                LR_5_L_Off();                   // Enable B Off
                LR_5_R_On();                    // Enable A On (Turn Right)
            }
                else if(strPtr[2]=='1')         // Turn Left
            {
                LR5=1;                          // Variable sent for Read Data
                LR_5_L_On();                    // Enable B On (Turn Left)
                LR_5_R_Off();                   // Enable A Off
            }
            }
                else if(strPtr[1]=='P')         // PWM Data Command
            {
DPWM5=(ReadChar(strPtr[2]))*16+(ReadChar(strPtr[3]));// Get hex char for PulseWidth
                if((DPWM1>0)&&(DPWM1<=249));// DPWM1>0 to 249 Start PWM
            {
                PWM16_5_Start();
                PWM16_5_WritePeriod(249);
                PWM16_5_WritePulseWidth(DPWM5);
            }
```

```
        else
{
        PWM16_5_Stop();             // Stop PWM
}
        UART_1_PutCRLF();           // Send "CR" Code
}
        else if(strPtr[1]=='R')     // For Read Data
{
Txt_5=(ERROR*0x02)+(LR5*0x01);  // Data for Sent to UART
        UART_1_PutSHexByte(Txt_5);
        UART_1_PutSHexByte((char)DPWM5);
        UART_1_PutCRLF();
}
}
//      UART_1_CmdReset();          // Reset Command Buffer
////////////////////////////////////////////////////////////////////////////////
        if(strPtr[0]=='6')          // Index of Motor6
{
        if(strPtr[1]=='L')          // Left/Right Command
{
        if(strPtr[2]=='0')          // Turn Right
{
        LR6=0;                      // Variable sent for Read Data
        LR_6_L_Off();               // Enable B Off
        LR_6_R_On();                // Enable A On (Turn Right)
}
        else if(strPtr[2]=='1')     // Turn Left
{
        LR6=1;                      // Variable sent for Read Data
        LR_6_L_On();                // Enable B On (Turn Left)
        LR_6_R_Off();               // Enable A Off
```

```
                }
                }
                        else if(strPtr[1]=='P')                 // PWM Data Command
                {
DPWM6=(ReadChar(strPtr[2]))*16+(ReadChar(strPtr[3]));// Get hex char for PulseWidth
                        if((DPWM1>0)&&(DPWM1<=249))// DPWM1>0 to 249 Start PWM
                {
                        PWM16_6_Start();
                        PWM16_6_WritePeriod(249);
                        PWM16_6_WritePulseWidth(DPWM6);
                }
                        else
                {
                        PWM16_6_Stop();                 // Stop PWM
                }
                        UART_1_PutCRLF();               // Send "CR" Code
                }
                        else if(strPtr[1]=='R')                 // For Read Data
                {
                Txt_6=(ERROR*0x02)+(LR6*0x01);      // Data for Sent to UART
                        UART_1_PutSHexByte(Txt_6);
                        UART_1_PutSHexByte((char)DPWM6);
                        UART_1_PutCRLF();
                }
                }
                        UART_1_CmdReset();              // Reset Command Buffer
                 }
                        UART_1_CmdReset();
                }
        }
}
```