การทดลองที่ 4

การใช้งานอินเตอร์รัพท์ วงจรจับเวลา และ การสื่อสารแบบอนุกรม

(Implementation of Interrupt, Timer and serial communication.)

วัตถุประสงค์

1. เพื่อให้เข้าใจการทำงานแบบอินเตอร์รัพท์ และสามารถเขียนโปรแกรมรองรับการอินเตอร์รัพท์ได้
2. เพื่อให้สามารถใช้งานโมดูลจับเวลา และโมดูลการสื่อสารแบบอนุกรมได้
3. เพื่อให้สามารถเชื่อมต่อโมดูลกับขาสัญญาณภายนอกของไมโครคอนโทรเลอร์ได้

บทนำ

การทดลองนี้ ได้แสดงวิธีการใช้งานโมดูลการแปลงสัญญาณอนาลอกแบบเดลต้าซิกมาลำดับที่ 2 ร่วมกับแอมปริไฟล์แบบปรับอัตราขยายได้ และนำผลการแปลงแสดงออกทางแอลซีดีแบบดอทแมทริก

การทดลองที่ 4.1 แสดงการใช้งานอินเตอร์รัพท์

การทดลอง เนื่องจากขั้นตอนต่างๆได้กล่าวไว้โดยละเอียดแล้วในการทดลองที่ผ่านมาแล้ว ดังนั้นในที่นี้จะกล่าวถึงเพียงย่อๆ หากนักศึกษายังไม่เข้าใจให้ย้อนกลับไปดูการทดลองดังกล่าว สำหรับในการทดลองนี้จะได้อธิบายอย่างละเอียดเกี่ยวกับขั้นตอนการเรียกใช้โมดูลและการเชื่อมต่อโมดูลเข้าด้วยกัน

1. ให้รันโปรแกรม PSoC Designer แล้วทำการสร้างโปรเจคใหม่ โดยใช้ไมโครคอนโทรลเลอร์เบอร์ CY8C29466-24PXI ส่วน Main file ให้เลือกเป็นภาษา C
2. เมื่อเข้าสู่ PSoC designer ในหน้าแรกจะเป็น User Module Selection View ในหน้าต่างนี้ให้เลือกใช้โมดูล แอลซีดี (LCD) โดยมีขั้นตอนดังนี้
3. หลังจากเลือกโมดูลต่างๆแล้ว ให้คลิก  (Interconnect View) เพื่อทำการวางอุปกรณ์และกำหนดค่าพารามิเตอร์ ของ LCD และ Port_1_0 ถึง Port_1_2 เพื่อทำเป็นสัญญาณอินเตอร์รัพท์ดังนี้

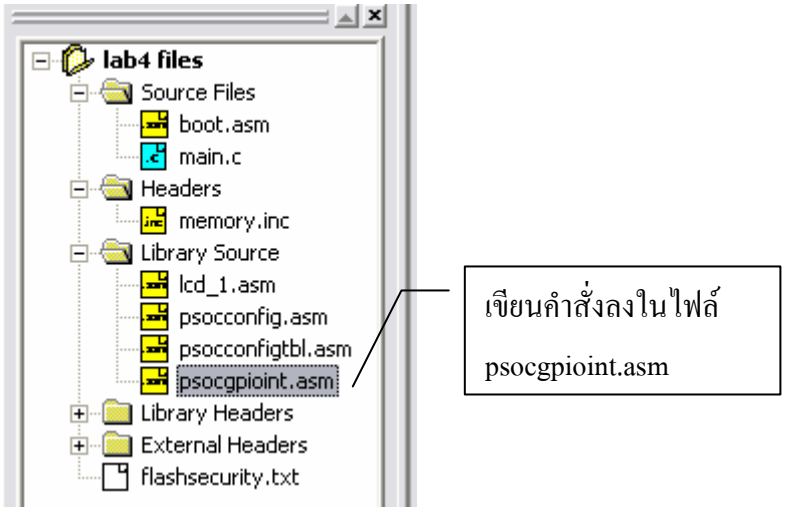
Name	Port	Select	Drive	Interrupt
Port_0_7	P0[7]	StdCPU	High Z Analog	DisableInt
Port_1_0	P1[0]	StdCPU	Pull Up	FallingEdge
Port_1_1	P1[1]	StdCPU	Pull Up	FallingEdge
Port_1_2	P1[2]	StdCPU	Pull Up	FallingEdge

4. เมื่อเสร็จแล้วให้กด  (Generate Application) เพื่อสร้างซอร์ไฟล์ต่างๆ
5. ให้คลิกที่ปุ่ม  (Application Editor) เพื่อเปิดหน้าต่าง Application เพื่อเขียน Source file ให้แก้ไขโปรแกรม main.c เป็น ดังนี้

โปรแกรมที่ 4.1

```
//-------------------------------------------------------------------------
// C main line
//-------------------------------------------------------------------------
#include <m8c.h>        // part specific constants and macros
#include "PSoCAPI.h"    // PSoC API definitions for all User Modules
#include <stdlib.h>

#define P1_0 ((PRT1DR&1)==1)          // Define port P1.0
#define P1_1 ((PRT1DR&2)==2)          // Define port P1.1
#define P1_2 ((PRT1DR&4)==4)          // Define port P1.2

#pragma interrupt_handler GPIO_Interrupt

unsigned int i, j, k;
char TextBuff[5];

void GPIO_Interrupt(void)
{
        if(P1_0==0) i++;                //if sw0 pressed
        else if (P1_1==0) j--;          //if sw1 pressed
        else if (P1_2 == 0) k = k+2;    //if sw2 pressed
}

void main()
{
        PRT1DR|=7;                      // Set Port P0.0 - P0.2 as input
        LCD_1_Start();                  // Initialize LCD
        M8C_EnableGInt;                 //Enable global interrupts
        M8C_EnableIntMask(INT_MSK0, INT_MSK0_GPIO);       //Enable GPIO interrupts

        i = 0;
        j = 100;
         k = 200;

         while(1)
         {
                LCD_1_Position(0,0);
                LCD_1_PrCString("SW0 ");
                itoa(TextBuff,i,10);                //convert integer to string base 10
                LCD_1_PrString(TextBuff);
                LCD_1_PrCString("   ");

                LCD_1_Position(0,8);
                LCD_1_PrCString("SW1 ");
                itoa(TextBuff,j,10);                //convert integer to string base 10
                LCD_1_PrString(TextBuff);
                LCD_1_PrCString("   ");

                LCD_1_Position(1,0);
                LCD_1_PrCString("SW2 ");
                itoa(TextBuff,k,10);                //convert integer to string base 10
                LCD_1_PrString(TextBuff);
                LCD_1_PrCString("   ");
        }
}
```

เพิ่มคำสั่งลงใน Library Source โดยคลิกเปิดไฟล์ psocgpioint.asm ที่อยู่ใน Library Source ตามรูป แล้วเขียนคำสั่ง ljmp _GPIO_Interrupt เพิ่มลงในไฟล์ในตำแหน่งตามรูป



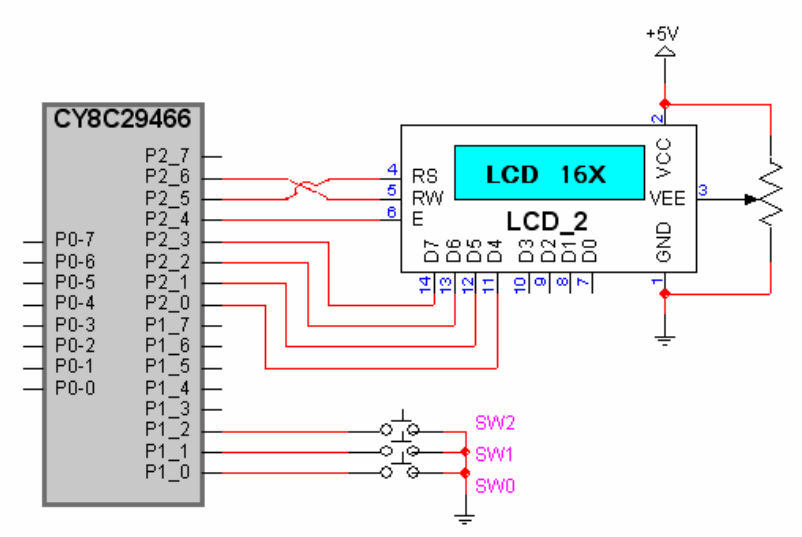

```
;-------------------------------------------------------------------------
;  FUNCTION NAME: PSoC_GPIO_ISR
;
;  DESCRIPTION: Unless modified, this implements only a null handler stub.
;
;-------------------------------------------------------------------------
;
PSoC_GPIO_ISR:


   ;@PSoC_UserCode_BODY@ (Do not change this line.)
   ;---------------------------------------------------
   ; Insert your custom code below this banner
   ;---------------------------------------------------
           ljmp      _GPIO_Interrupt
   ;---------------------------------------------------
   ; Insert your custom code above this banner
   ;---------------------------------------------------
   ;@PSoC_UserCode_END@ (Do not change this line.)

   reti

; end of file PSoCGPIOINT.asm
```

เขียนเพิ่ม

แล้วทำการบันทึกไฟล์

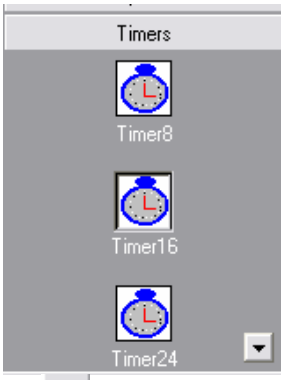
6. ให้ทำการแปลโปรแกรมโดยคลิกที่ปุ่ม (Build) เพื่อแปลโปรแกรม หากไม่มีข้อผิดพลาด จะได้ไฟล์ Hex ที่มีชื่อเหมือนกับชื่อของโปรเจค อยู่ในโฟลเดอร์ Output สำหรับใช้ดาว์นโหลดลงชิพไมโครคอนโทรลเลอร
7. ทำการติดตั้ง LCD เข้ากับบอร์ด

8. ให้ต่อสาย USB แล้วดาว์นโหลดโปรแกรมลงชิพ **<u>ขั้นตอนนี้ขอให้ทำด้วยความระมัดระวังมิฉนั้นอาจเกิดความเสียหายคอมพิวเตอร์และบอร์ดทดลองได้</u>**
9. เมื่อดาว์นโหลดเสร็จ ให้ทำการทดสอบการทำงาน โดยการต่อสวิตช์ กดติดปล่อยดับเข้าที่ P1.0 ถึง P1.2 แล้วทำการกดสวิต เพื่อป้อนโลจิก '0' เข้าที่พอร์ทเพื่ออินเตอร์รัพท์ แล้วสังเกตผลที่เกิดขึ้นบน LCD



การทดลองที่ 4.2 แสดงการใช้งาน Timer และ อินเตอร์รัพท์

1. ให้รันโปรแกรม PSoC Designer แล้วทำการสร้างโปรเจคใหม่ โดยใช้ไมโครคอนโทรลเลอร์เบอร์ CY8C29466-24PXI ส่วน Main file ให้เลือกเป็นภาษา C
2. เมื่อเข้าสู่ PSoC designer และในหน้าต่าง User Module Selection View ในหน้าต่างนี้ให้เลือกใช้โมดูล แอลซีดี (LCD) และโมดูล Timer 16 จากกลุ่ม Timer



3. หลังจากเลือกโมดูลต่างๆแล้ว ให้คลิก  (Interconnect View) เพื่อทำการวางอุปกรณ์และกำหนดค่าพารามิเตอร์ โดยพารามิเตอร์ของ LCD และ Port_0_1 ถึง Port_0_2 ให้ทำเช่นเดียวกับการทดลองที่ 4.1 ส่วนพารามิเตอร์ของ Timer16 ให้ทำดังนี้ (อย่าลืมคลิกวางโมดูล  ก่อนไม่เช่นนั้นจะกำหนดพารามิเตอร์ไม่ได้)

กำหนดพารามิเตอร์ของ Timer16-1

Timer16_1

User Module Parameters	Value
Clock	VC2
Capture	Low
TerminalCountOut	None
CompareOut	None
Period	9999
CompareValue	500
CompareType	Less Than Or Equal
InterruptType	Terminal Count
ClockSync	Sync to SysClk
TC_PulseWidth	Full Clock
InvertCapture	Normal

กำหนดพารามิเตอร์ของ Global Resource

Global Resources	Value
Power Setting [Vcc / SysClk freq	5.0V / 24MHz
CPU_Clock	SysClk/8
32K_Select	Internal
PLL_Mode	Disable
Sleep_Timer	512_Hz
VC1= SysClk/N	8
VC2= VC1/N	3
VC3 Source	VC1
VC3 Divider	39
SysClk Source	Internal
SysClk*2 Disable	No
Analog Power	SC On/Ref Low
Ref Mux	(Vdd/2)+/-BandGap
AGndBypass	Disable
Op-Amp Bias	Low
A_Buff_Power	Low
SwitchModePump	OFF
Trip Voltage [LVD (SMP)]	4.81V (5.00V)
LVDThrottleBack	Disable
Watchdog Enable	Disable

4. เมื่อเสร็จแล้วให้กด  (Generate Application) เพื่อสร้างซอร์ไฟล์ต่างๆ
5. ให้คลิกที่ปุ่ม  (Application Editor) เพื่อเปิดหน้าต่าง Application เพื่อเขียน Source file ให้แก้ไขโปรแกรม main.c เป็น ดังนี้

```
//-------------------------------------------------------------------------
// Timer with start, stop, and cleare function
//-------------------------------------------------------------------------

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

#define P1_0 ((PRT1DR&1)==1)          // Define port P1.0
#define P1_1 ((PRT1DR&2)==2)          // Define port P1.1
#define P1_2 ((PRT1DR&4)==4)          // Define port P1.2

#pragma interrupt_handler GPIO_Interrupt       // Set interrupt handler for GPIO
#pragma interrupt_handler Timer_Interrupt      // Set interrupt handler for timer

DWORD Time = 0;
char TextBuff[5];

void GPIO_Interrupt(void)
{
        if(P1_0==0) Timer16_1_Start();
        else if (P1_1==0) Timer16_1_Stop();
        else if (P1_2 == 0) Time = 0;
}

void Timer_Interrupt(void)
{
        Time++;
}

void main()
{
        PRT1DR|=7;              // Set Port P1.0 - P1.2 as input
        LCD_1_Start();          // Initialize LCD

        M8C_EnableGInt;         //Enable global interrupts
        M8C_EnableIntMask(INT_MSK0, INT_MSK0_GPIO);       //Enable GPIO interrupts
        Timer16_1_EnableInt();                            // Enable Timer16_1 interrupt

        while(1)
        {
                LCD_1_Position(0,0);
                LCD_1_PrCString("Start Stop Clear");
                LCD_1_Position(1,0);
                LCD_1_PrCString("    ");
                itoa(TextBuff,Time/6000,10);              //convert integer to min
                LCD_1_PrString(TextBuff);
                LCD_1_PrCString(":");

                itoa(TextBuff,(Time%6000)/100,10);   //convert integer to sec
                LCD_1_PrString(TextBuff);
                LCD_1_PrCString(":");
                itoa(TextBuff,Time%100,10);               //convert integer to string base 10
                LCD_1_PrString(TextBuff);
                LCD_1_PrCString("   ");
        }
}
```

เพิ่มคำสั่งลงใน Library Source โดยคลิกเปิดไฟล์ psocgpioint.asm และ timer16_1int.asm ที่อยู่ใน Library Source ตามรูป แล้วเขียน เพิ่มลงในไฟล์ในตำแหน่งตามรูป

psocgpioint.asm

```
;-------------------------------------------------------------------------
;  FUNCTION NAME: PSoC_GPIO_ISR
;
;  DESCRIPTION: Unless modified, this implements only a null handler stub.
;
;-------------------------------------------------------------------------
;
PSoC_GPIO_ISR:

   ;@PSoC_UserCode_BODY@ (Do not change this line.)
   ;---------------------------------------------------
   ; Insert your custom code below this banner
   ;---------------------------------------------------
            ljmp       _GPIO_Interrupt
   ;---------------------------------------------------
   ; Insert your custom code above this banner
   ;---------------------------------------------------
   ;@PSoC_UserCode_END@ (Do not change this line.)

   reti

; end of file PSoCGPIOINT.asm
```

เขียนเพิ่ม

timer16_1int.asm

```
;-------------------------------------------------------------------------
;  FUNCTION NAME: _Timer16_1_ISR
;
;  DESCRIPTION: Unless modified, this implements only a null handler stub.
;
;-------------------------------------------------------------------------

_Timer16_1_ISR:

   ;@PSoC_UserCode_BODY@ (Do not change this line.)
   ;---------------------------------------------------
   ; Insert your custom code below this banner
   ;---------------------------------------------------
   ;   NOTE: interrupt service routines must preserve
   ;   the values of the A and X CPU registers.
            ljmp       _Timer_Interrupt
   ;---------------------------------------------------
   ; Insert your custom code above this banner
   ;---------------------------------------------------
   ;@PSoC_UserCode_END@ (Do not change this line.)

   reti
; end of file Timer16_1INT.asm
```

เขียนเพิ่ม

6. แล้วทำการบันทึกไฟล์ แล้วให้ทำการแปลโปรแกรม

7. ทำการติดตั้ง LCD เข้ากับบอร์ด และต่อสวิทช์ กดติดปล่อยดับเข้าที่ P0.0 ถึง P0.2 เหมือนการทดลองที่ 4.1
8. ให้ต่อสาย USB แล้วดาว์นโหลดโปรแกรมลงชิพ **ขั้นตอนนี้ขอให้ทำด้วยความระมัดระวังมิฉนั้นอาจเกิดความเสียหายคอมพิวเตอร์และบอร์ดทดลองได้**
9. เมื่อดาว์นโหลดเสร็จ ให้ทำการทดสอบการทำงาน แล้วสังเกตผลที่เกิดขึ้นบน LCD

อธิบายเพิ่มเติม

การทดลองนี้ต้องการให้ซีพียูถูกอินเตอร์รัพท์ทุกๆ 100 Hz ดังนั้นเมื่อ Timer16-1ใช้สัญญาณนาฬิกาจาก VC2 ซึ่งสามารถคำนวณหาความถี่ได้ดังนี้

VC2 = VC1/3 จากกำหนดพารามิเตอร์ของ Global Resource

VC1 = SysClk/8 จากกำหนดพารามิเตอร์ของ Global Resource

VC2 = SysClk/(8x3) = 24MHz/24 = 1 MHz

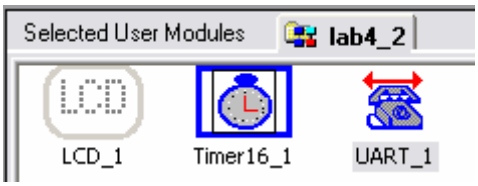
เมื่อต้องการให้อินเตอร์รัพท์ทุกๆ 100 Hz ค่าเวลาที่ Timer จะอินเตอร์รัพท์ (Period) จึงเท่ากับ

Period = 1 MHz/100 Hz = 1000000/100 =10000

แต่การอินเตอร์รัพท์ของ Timer ตั้งไว้เป็นแบบ Terminal Count ซึ่งหมายถึงว่า สัญญาณอินเตอร์รัพท์จะเกิดขึ้นในสัญญาณลูกถัดจากการนับครบค่าที่ตั้ง ดังนั้น ค่า Period จึงต้องกำหนดเป็น 10000 – 1 = 9999 จากการกำหนดค่าทั้งหมดนี้ทำให้การอินเตอร์รัพท์เกิดขึ้นทุกๆ 0.01 วินาที (100 Hz) นั่นคือค่าตัวแปร Time จะมีค่าเพิ่มขึ้นที่ 1 ทุกๆ 0.01 วินาทีเช่นกัน เมื่อต้องการให้แสดงผลเป็น นาที วินาที และ เศษของวินาที จึงต้องทำการหารด้วย 100 และ 60 เพื่อให้ได้ค่า นาที และวินาทีตามลำดับ

การทดลองที่ 4.3 แสดงการใช้งาน อินเตอร์รัพท์ Timer และ การสื่อสารแบบอนุกรม UART

1. ต่อเนื่องจากการทดลองที่ 2 ให้กลับเข้าสู่หน้าต่าง User Module Selection View ในหน้าต่างนี้ให้เลือกโมดูล UART ที่อยู่ในกลุ่ม Digital Comm เพิ่ม ขึ้นจากที่เลือก แอลซีดี และโมดูล Timer 16 ไว้แล้ว



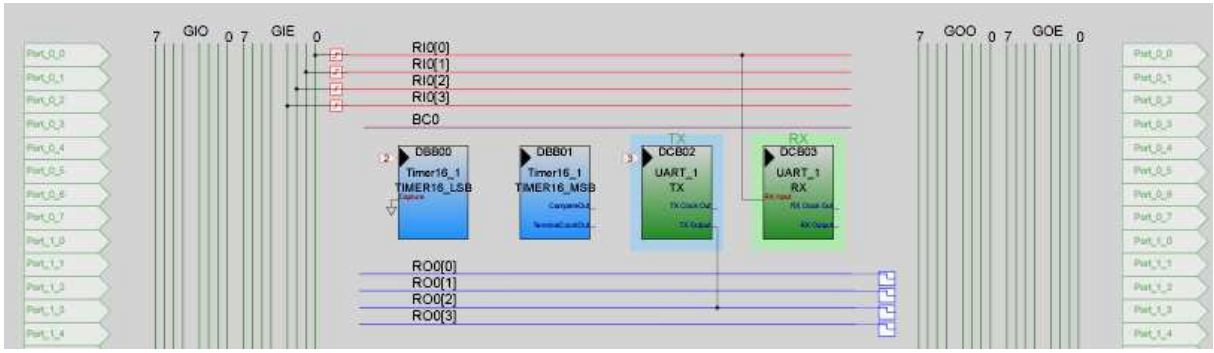
2. หลังจากนั้นให้คลิก  (Interconnect View) เพื่อวาง UART และกำหนดค่าพารามิเตอร์ โดยพารามิเตอร์ของ Global Resource LCD และ Timer16 ให้คงเดิมเหมือนการทดลองที่ 4.2 ส่วน UART ให้กำหนดพารามิเตอร์ตามนี้

พารามิเตอร์ของ UART_1

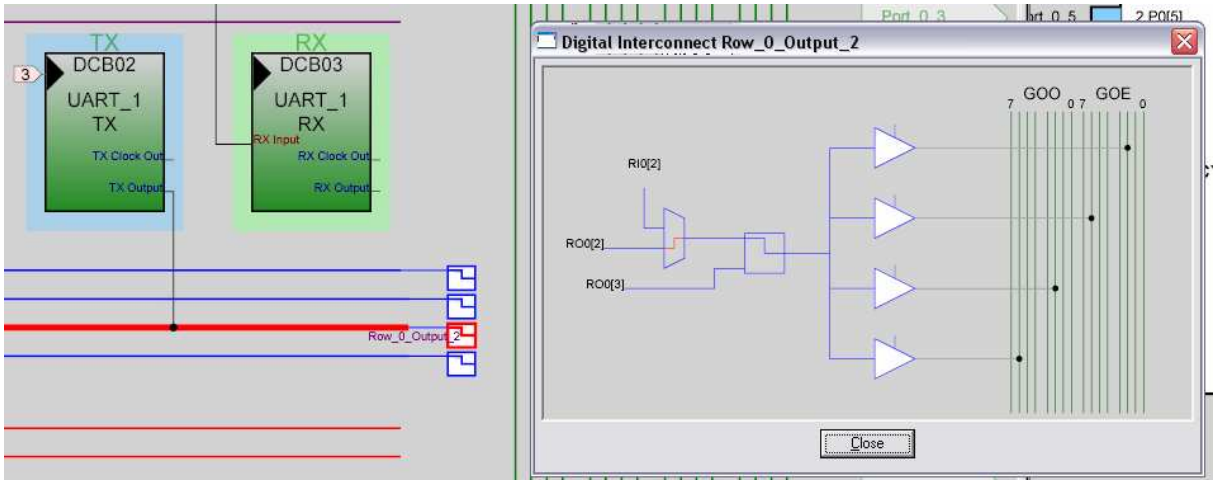
UART_1

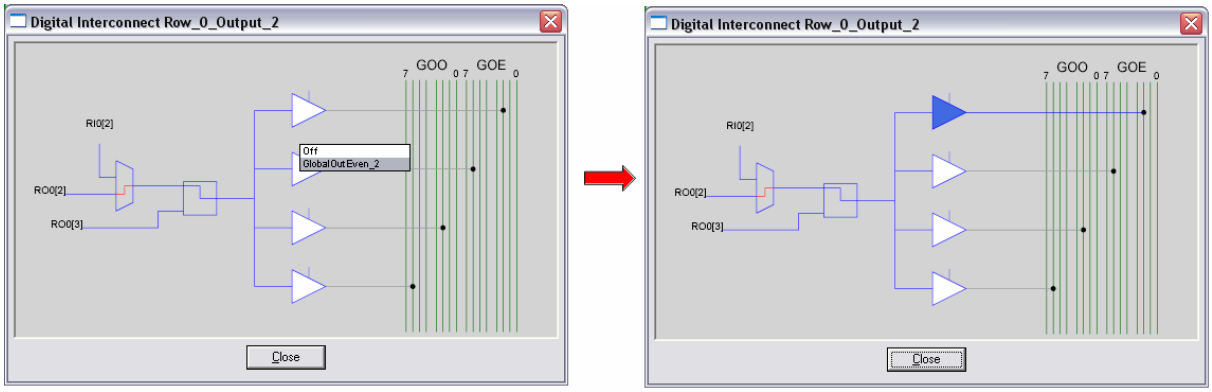
User Module Parameters	Value
Clock	VC3
RX Input	Row_0_Input_0
TX Output	Row_0_Output_2
TX Interrupt Mode	TXComplete
ClockSync	Sync to SysClk
RxCmdBuffer	Enable
RxBufferSize	16
CommandTerminator	13
Param_Delimiter	32
IgnoreCharsBelow	32
Enable_BackSpace	Disable
RX Output	None
RX Clock Out	None
TX Clock Out	None
InvertRX Input	Normal

3. ทำการเชื่อมต่ออินพุทและเอาท์พุทของโมดูล UART_1 เข้ากับขาไมโครคอนโทรเลอร์ โดยจะสังเกตได้ว่า จากค่าพารามิเตอร์ที่กำหนด สัญญาณ TX Output และ สัญญาณ RX Input จะต่ออยู่กับ Row_0_Output_2 (RO0[2]) และ Row_0_Input_0 (RI0[0])

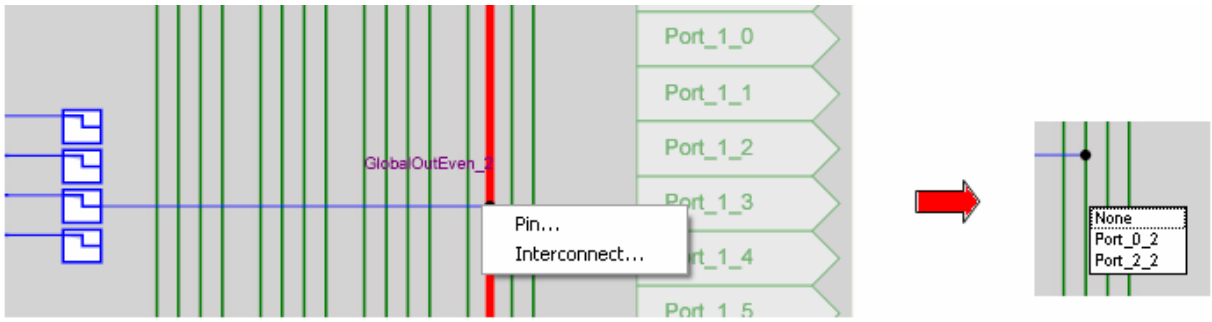


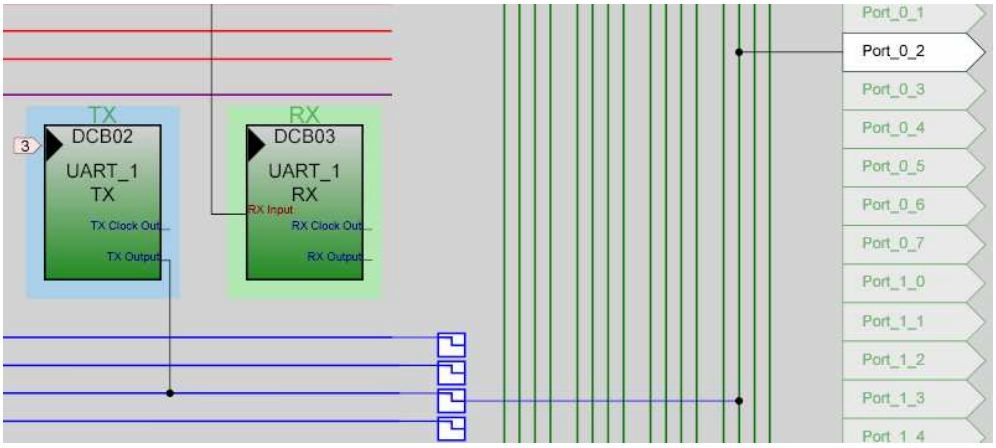
การต่อสัญญาณ TX Output จากโมดูลออกไปที่ขาให้ใช้เมาส์คลิกขวาที่สัญญาณ TX Output จะปรากฏวงจรแสดงการต่อของสัญญาณไปยัง GOE มีทั้งหมด 4 ทางเลือก ให้คลิกขวาที่ทางเลือก ต่อบนสุด แล้วคลิกขวาอีกครั้งเลือกเป็น GlobalOutEven_2 เพื่อต่อออกไปยัง GOE _2 แล้วคลิก Close



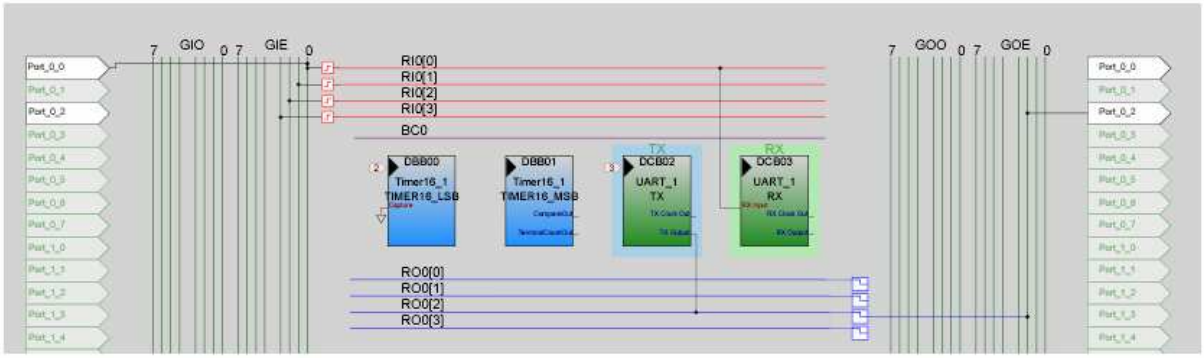


ที่สายสัญญาณ GOE_2 นี้สามารถที่จะตรวจสอบได้ว่า จะต่อกับขาใดได้บ้าง โดยให้คลิกขาวที่สัญญาณนั้น จะปรากฏ Pop up ว่าต้องการต่อกับ Pin หรือ Interconnect เมื่อคลิกขวาที่ Pin จะปรากฏตัวเลือกว่าต้องการต่อกับขา Port_0_2 หรือ Port_2_2 หรือไม่ต่อ (None) ให้เลือกเป็น Port_0_2 จะได้ผลดังรูป





4. ส่วนการต่อสัญญาณ RX_Input ก็ทำในทำนองเดียวกัน ให้ต่อออกยังขา Port_0_0



5. เมื่อเสร็จแล้วให้กด  (Generate Application) เพื่อสร้างซอร์ไฟล์ต่างๆ

6. ให้คลิกที่ปุ่ม  (Application Editor) เพื่อเปิดหน้าต่าง Application เพื่อเขียน Source file ให้แก้ไขโปรแกรม main.c เป็น ดังนี้

โปรแกรม main4.3

```
//-------------------------------------------------------------------------
// Count-down Timer clock with UART communication
//-------------------------------------------------------------------------

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

#pragma interrupt_handler Timer_Interrupt         // Set interrupt handler for timer

DWORD Time = 0;
char TextBuff[16];

void LCD_PrintTime(void)
{
      LCD_1_Position(1,0);
      itoa(TextBuff,Time/6000,10);         //convert integer to min
      LCD_1_PrString(TextBuff);
      LCD_1_PrCString(":");
```

```
    itoa(TextBuff,(Time%6000)/100,10);   //convert integer to sec
    LCD_1_PrString(TextBuff);
    LCD_1_PrCString(":");

    itoa(TextBuff,Time%100,10);         //convert integer to string base 10
    LCD_1_PrString(TextBuff);
    LCD_1_PrCString("    ");
}

void Timer_Interrupt(void)
{
    Time--;
}

void main()
{
    BYTE i;
    LCD_1_Start();                                  // Initialize LCD
    UART_1_CmdReset();                              // Reset Command Buffer
    UART_1_IntCntl(UART_1_ENABLE_RX_INT);           // Enable Receiver
    UART_1_Start(UART_1_PARITY_NONE);               // Start UART
    M8C_EnableGInt;                                 //Enable global interrupts
    M8C_EnableIntMask(INT_MSK0, INT_MSK0_GPIO); //Enable GPIO interrupts
    Timer16_1_EnableInt();                          // Enable Timer16_1 interrupt
    LCD_1_Position(0,0);
    LCD_1_PrCString("BaudRate = 9600 ");
    UART_1_CPutString("\n\rEnter Time\n\r");
    while(1)
    {
        if (UART_1_bCmdCheck()!=0)              //Non-zero value if a command has been recieved.
        {
            LCD_1_Position(0,0);
            LCD_1_PrCString("               ");

            LCD_1_Position(0,0);
            for (i=0;i<UART_1_bCmdLength();i++)
            {
                UART_1_PutChar(UART_1_aRxBuffer[i]);
                TextBuff[i] = UART_1_aRxBuffer[i];
            }
            UART_1_PutCRLF();
            UART_1_CmdReset();
            Time = atol(TextBuff);
            LCD_1_PrCString("Time: ");
            LCD_1_PrString(TextBuff);
            UART_1_CPutString("Timer Start \n\r");
            Timer16_1_Start();
            while(Time !=0)
            {
                LCD_PrintTime();
            }
            Timer16_1_Stop();
            LCD_PrintTime();
            UART_1_CPutString("Time out\n\r");
            UART_1_CPutString("\n\rEnter Time\n\r");
        }
    }
}
```

และ โปรแกรม timer16_1int.asm แก้ไขเหมือนกับการทดลองที่ 4.2

7. แล้วทำการบันทึกไฟล์ และแปลโปรแกรม
8. ทำการติดตั้ง LCD เข้ากับบอร์ด และสัญญาณระหว่าง Port_0 กับขั้วต่อ สัญญาณของ RS-232 ดดำพ โดยต่อสายจาก **P00 เข้ากับขา RX และ P02 เข้ากับขา TX**
9. ต่อสาย RS-232 จากบอร์ดทดลองเข้ากับ พอร์ท RS-232 ของเครื่องพีซี
10. ให้ต่อสาย USB แล้วดาว์นโหลดโปรแกรมลงชิพ **ขั้นตอนนี้ขอให้ทำด้วยความระมัดระวังมิฉนั้นอาจเกิดความเสียหายคอมพิวเตอร์และบอร์ดทดลองได้**
11. เมื่อดาว์นโหลดเสร็จ ให้ทำการทดสอบการทำงาน โดยบนเครื่องพีซีให้รันโปรแกรม Hyper Terminal โดยกำหนด Buadrate เป็น 9600 เมื่อจ่ายไฟเลี้ยงให้บอร์ดทดลอง แล้วสังเกตผลที่เกิดขึ้นบน LCD และบนจอของ Hyper terminal ให้พิมพ์ตัวเลขจำนวนเวลา แล้วกดคีย์ Enter ลัวสังเกตการทำงาน

อธิบายเพิ่มเติม

การทดลองนี้ต้องการใช้ UART ที่ความเร็ว 9600 บิตต่อวินาที ณ ที่ความเร็วนี้ โมดูล UART ต้องการสัญญาณนาฬิกาที่มีความถี่สูงกว่า 8 เท่า ดังนั้นเมื่อกำหนดให้ใช้สัญญาณนาฬิกาจาก VC3 จึงต้องกำหนดพารามิเตอร์ของ VC3 ดังนี้

VC3 = 8 x Baudrate = 8 x 9600 = 76.8 K

VC3 source = VC1

ดังนั้น VC3 = VC1/Divider

VC1 = SysClk/8 = 24 MHz/8 = 3 MHz จากกำหนดพารามิเตอร์ของ Global Resource

VC3 = 76.8 K = 3 MHz/ Divider

Divider = 3000/76.8 = 39.0625

ดังนั้นจึงกำหนดตัวหารสำหรับ VC3 เป็น 39

คำถาม

1. ให้อธิบายการทำงานของโปรแกรม ทั้ง 3 โปรแกรม
2. ให้เขียนโปรแกรม ที่มีการใช้งาน การอินเตอร์รัพท์ Timer และ UART ที่แตกต่างจากโปรแกรมทดลอง มากลุ่มละ 1 โปรแกรม โดยให้อธิบายการทำงานของโปรแกรมที่เขียนด้วย

หมายเหตุสำหรับข้อสองการคิดคะแนนจะคิดจาก 100% หารด้วยจำนวนกลุ่มที่ใช้โมดูลแบบเดียวกัน