การใช้งานอินเตอร์รัพท์ วงจรจับเวลา และ การสื่อสารแบบอนุกรม

รศ.ณรงค์ บวบทอง
ภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์
คณะวิศวกรรมศาสตร์
มหาวิทยาลัยธรรมศาสตร์

อย่าขาด

เงินและงานการศึกษาควรมาก่อน

อย่ารีบร้อนหารักงานจักเสีย

หากขาดเงินขาดงานพาลขาด.....

งานไม่เสีย เงินคงได้ ..... คงมา

หัวข้อ

1. **วัตถุประสงค์**
2. **อินเตอร์รัพท์**
3. **วงจรจับเวลา**
4. **การสื่อสารแบบอนุกรม**

วัตถุประสงค์

- เพื่อให้เข้าใจการทำงานแบบอินเตอร์รัพท์ และสามารถเขียนโปรแกรมรองรับการอินเตอร์รัพท์ได้
- เพื่อให้สามารถใช้งานโมดูลจับเวลา และโมดูลการสื่อสารแบบอนุกรมได้
- เพื่อให้สามารถเชื่อมต่อโมดูลกับขาสัญญาณภายนอกของไมโครคอนโทรเลอร์ตระกูล PSoCได้

การอินเตอร์รัพท์

- การอินเตอร์รัพท์คืออะไร ?

Suddenly more data becomes available for the assignment.

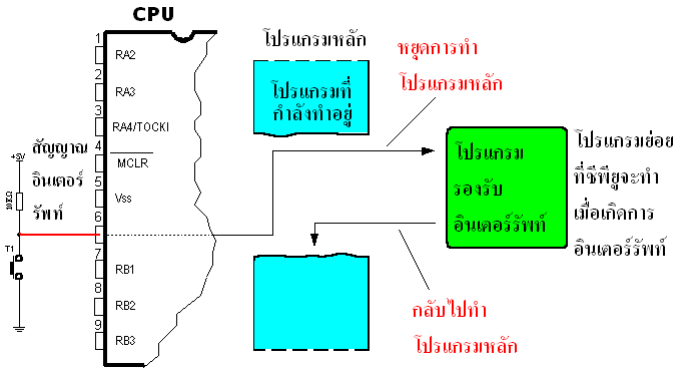
The 6502's interrupt latency is one of the 8-bit world's fastest.

ห้ามไม่ให้อินเตอร์รัพท์ได้หรือไม่ ?

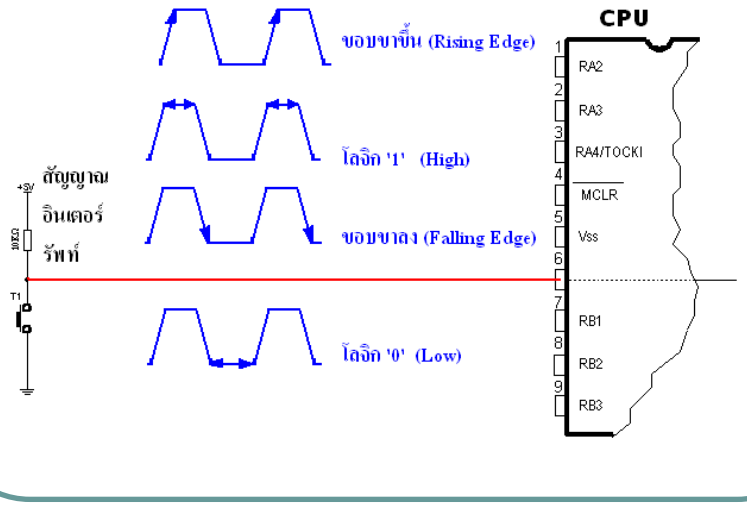
อยากขัดจังหวะ
หรือพี่น้อง

ห้ามไม่ให้อินเตอร์รัพท์ได้หรือไม่ ?

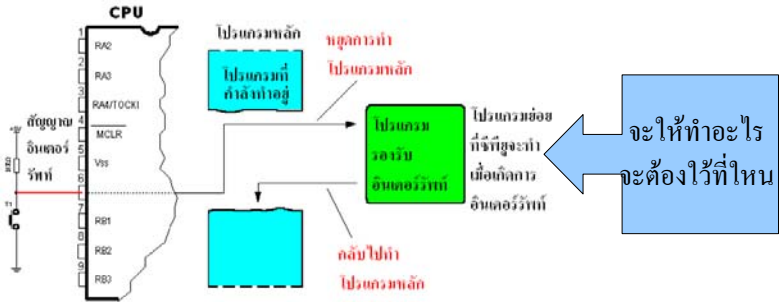
การอินเตอร์รัพท์ของซีพียู



สัญญาณอินเตอร์รัพท์



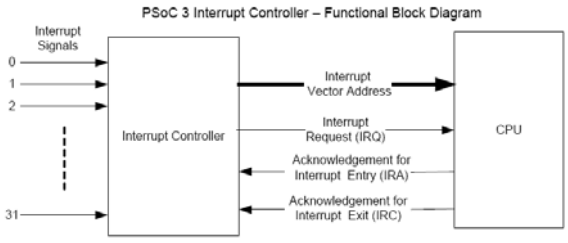
โปรแกรมรองรัพท์อินเตอร์รัพท์



สรุปถ้าต้องการให้อินเตอร์รัพท์ ต้องทำอะไรบ้าง ?

- อนุญาตให้อินเตอร์รัพท์ได้
- สัญญาณประเภทใด
- เตรียมโปรแกรมที่จะให้ทำเมื่อมีอินเตอร์รัพท์และนำไปไว้ที่ตกลงกันไว้

การอินเตอร์รัพท์ของ PSoC



- การกำหนดสถานะให้อินเตอร์รัพท์ได้หรือไม่ได้
- อะไรอินเตอร์รัพท์ได้บ้าง
- อินเตอร์รัพท์แล้วไปทำงานที่ไหน
- ให้ทำงานอะไร

การกำหนดสถานะให้อินเตอร์รัพท์ได้หรือไม่ได้ของ PSoC

- ถ้าต้องการให้อินเตอร์รัพท์ซีพียูได้ต้อง ให้

  global interrupt permit = ‘1’ (GIE = 1)

ภาษาซี

```
M8C_EnableGInt;
```

ถ้าไม่กำหนดจะเป็น ไม่ยอมรับการอินเตอร์รัพท์ (Disable)

ใน PSoC อะไรอินเตอร์รัพท์ได้

Reset	จากการรีเซ็ท
Low supply voltage	จากการตรวจสอบแรงดันแหล่งจ่าย
Analog	จากการแปลงสัญญาณอนาลอก
Timer	จากวงจรจับเวลา
GPIO	การเปลี่ยนแปลงสัญญาณที่ขา
UART	จากการสื่อสารข้อมูลแบบอนุกรม
อื่นๆ	

PSoC กำหนด ตำแหน่งของโปรแกรมรองรับอินเตอร์รัพท์ไว้ที่ใด

Vector #		Address (16-bit)
1	Reset	0000h
2	Low supply voltage	0004h
3	Analog column 0	0008h
4	Analog column 1	000Ch
5	Analog column 2	0010h
6	Analog column 3	0014h
7	VC3	0018h
8	GPIO	001Ch

0004h: LJMP to code for handling low voltage indicator

0010h: LJMP to code for handling analog col 2 interrupt

001Ch: LJMP to code for handling GPIO interrupt

ตำแหน่งของโปรแกรมรองรับอินเตอร์รัพท์ของ PSoC ทั้งหมด

Name: Identifies source of interrupt

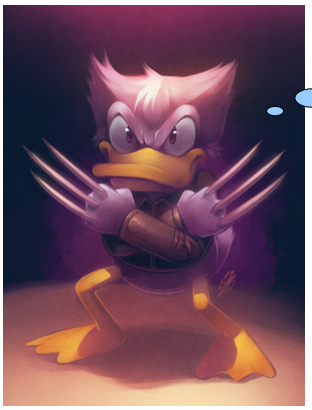
Device Availability

Priority: If interrupts happen at the same time, highest priority interrupt wins

Address: Address of interrupt vector
Note: Only 4 bytes, usually just an LJMP

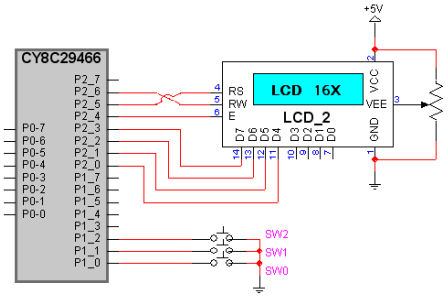
Interrupt Priority	Interrupt Address	PSoC Devices CY8 – C29x66	C27x43	C24x94	C24x23	C24x23A	C22x13	C21x34	C21x23	PSoC Devices CY7 – C64215	C603xx	CYWUSB6935	Interrupt Name
0 (Highest)	0000h	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	Reset
1	0004h	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	Supply Voltage Monitor
2	0008h	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	Analog Column 0
3	000Ch	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	Analog Column 1
4	0010h	✓	✓										Analog Column 2
5	0014h	✓	✓										Analog Column 3
6	0018h	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	VC3
7	001Ch	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	GPIO
8	0020h	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	PSoC Block DBB00
9	0024h	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	PSoC Block DBB01
10	0028h	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	PSoC Block DCB02
11	002Ch	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	PSoC Block DCB03
12	0030h	✓	✓										PSoC Block DBB10
13	0034h	✓	✓										PSoC Block DBB11
14	0038h	✓	✓										PSoC Block DCB12
15	003Ch	✓	✓										PSoC Block DCB13
16	0040h	✓		USB Bus Reset						USB Bus Reset			PSoC Block DBB20
17	0044h	✓		USB Start of Frame						USB Start of Frame			PSoC Block DBB21
18	0048h	✓		USB Endpoint 0						USB Endpoint 0			PSoC Block DCB22
19	004Ch	✓		USB Endpoint 1						USB Endpoint 1			PSoC Block DCB23
20	0050h	✓		USB Endpoint 2						USB Endpoint 2			PSoC Block DBB30
21	0054h	✓		USB Endpoint 3						USB Endpoint 3			PSoC Block DBB31
22	0058h	✓		USB Endpoint 4						USB Endpoint 4			PSoC Block DCB32
23	005Ch	✓		USB Wakeup Interrupt						USB Wakeup Interrupt			PSoC Block DCB33
24	0060h	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	I2C
25 (Lowest)	0064h	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	Sleep Timer

ให้ทำงานอะไรได้บ้าง



ใครจะให้ทำ
อะไรบอกมา

ทดลองอินเตอร์รัพท์จาก GPIO



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

Pull down
ขอบขาลง

```
M8C_EnableGInt;                          //Enable global interrupts
M8C_EnableIntMask(INT_MSK0, INT_MSK0_GPIO);   //Enable GPIO interrupts
```

การกำหนดตำแหน่งของโปรแกรมรองรับอินเตอร์รัพท์

```
#pragma interrupt_handler GPIO_Interrupt
```

#pragma interrupt_handler บอกให้ตัวแปล C เก็บและคืนค่ารีจิสเตอร์และใช้ interrupt return ที่ตอนจบโปรแกรม

```
void GPIO_Interrupt(void)
{
    if(P1_0==0) i++;
    else if (P1_1==0) j--;
    else if (P1_2 == 0) k = k+2;
}
```

```
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
```

การกำหนดชื่อเทียมด้วย #define

```
#define P1_0 ((PRT1DR&1)==1)   //port1  xxxx xxx? and 0000 0001 เท่ากับ 1 ?
#define P1_1 ((PRT1DR&2)==2)   //port1 xxxx xx?x and 0000 0010 เท่ากับ 2 ?
#define P1_2 ((PRT1DR&4)==4)   //port1 xxxx x?xx and 0000 0100 เท่ากับ 4 ?
```

เวลาใช้งานเพื่อตรวจสอบ

```
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
```

การกำหนดให้ให้เป็นพอร์ทอินพุท

PRT1DR|=7;

หมายถึง PRT1DR = PRT1DR or 0000 0111

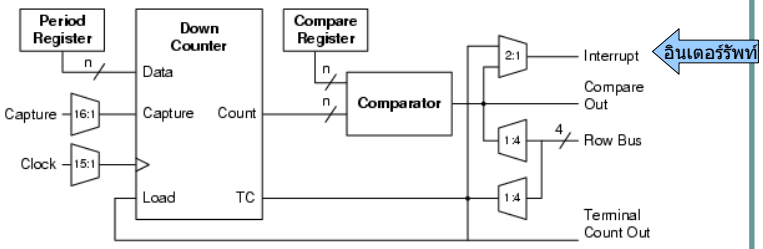
ทำให้ Port P1.0 - P1.2 เป็น input

Timer

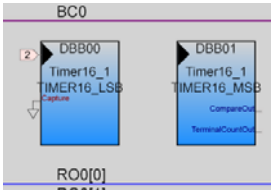
Features and Overview

- 8-, 16-, 24- or 32-bit general purpose timer uses one, two, three or four PSoC blocks, respectively
- Source clock rates up to 48 MHz
- Automatic reload of period on terminal count
- Capture for clocks up to 24 MHz.
- Terminal count output pulse may be used as input clock for other analog and digital functions
- Interrupt option on terminal count, capture, or when counter reaches a preset value

บล็อกไดอะแกรม Timer 16



- Timer Block Diagram (42xxx Family), Data Path width n = 8, 16, 24 or 32



การกำหนดพารามิเตอร์ของ Timer 16

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

ความหมายการกำหนดค่า

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 เมื่อต้องการให้
แสดงผลเป็น นาที วินาที และ เศษของวินาที จึงต้องทำการหารด้วย 100 และ 60
เพื่อให้ได้ค่า นาที และวินาทีตามลำดับ

API ของ Timer16

- Timer16_PERIOD
- Timer16_COMPARE_VALUE
- Timer16_EnableInt
- Timer16_DisableInt
- Timer16_Start
- Timer16_Stop

Timer16_PERIOD

Timer16_PERIOD

รายละเอียด

Represents the value chosen for the Period field of the Timer16 in the Device Editor. The value can have a range between 0 and 65535.

Timer16_COMPARE_VALUE

Timer16_COMPARE_VALUE

รายละเอียด

Represents the value chose for the PulseWidth field of the Timer16 in the Device Editor. The value can have a range between 0 and 65535.

Timer16_EnableInt

ฟังก์ชั่น Timer16_EnableInt

รายละเอียด

Enables the interrupt mode operation. Note, however, that global interrupts must also be enabled before interrupts will actually be serviced.

การใช้งานด้วยภาษา C

```
void  Timer16_EnableInt(void);
```

พารามิเตอร์

ไม่มี

ค่าส่งกลับ

ไม่มี

Timer16_DisableInt

ฟังก์ชั่น Timer16_DisableInt

รายละเอียด

Disables the interrupt mode operation.

การใช้งานด้วยภาษา C

```
void  Timer16_DisableInt(void);
```

พารามิเตอร์

ไม่มี

ค่าส่งกลับ

ไม่มี

Timer16_Start

ฟังก์ชั่น Timer16_Start

รายละเอียด

Starts the Timer16 operation. The Count register will be decremented on the next clock cycle.

การใช้งานด้วยภาษา C

```
void  Timer16_Start(void);
```

พารามิเตอร์

ไม่มี

ค่าส่งกลับ

ไม่มี

Timer16_Stop

ฟังก์ชั่น Timer16_Stop

รายละเอียด

Stops the Timer16 operation.

การใช้งานด้วยภาษา C

```
void  Timer16_Stop(void);
```

พารามิเตอร์

ไม่มี

ค่าส่งกลับ

ไม่มี

การกำหนดตำแหน่งของโปรแกรมรองรับอินเตอร์รัพท์

```
#pragma interrupt_handler GPIO_Interrupt    // Set interrupt handler for GPIO
```

```
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;
}
```

psocgpioint.asm

```
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
   ;---------------------------------------------------
```

การกำหนดตำแหน่งของโปรแกรมรองรับอินเตอร์รัพท์

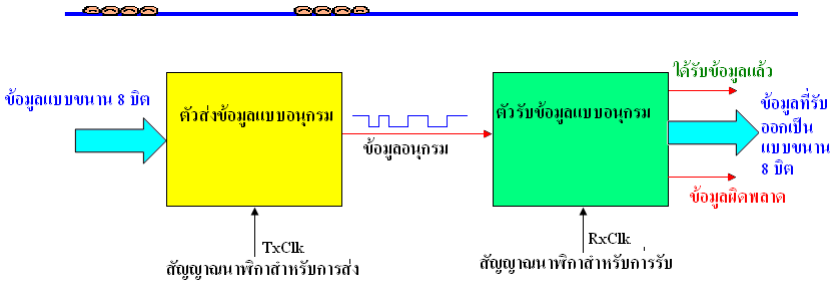
```
#pragma interrupt_handler Timer_Interrupt    // Set interrupt handler for timer
```

```
void Timer_Interrupt(void)
{
        Time++;
}
```

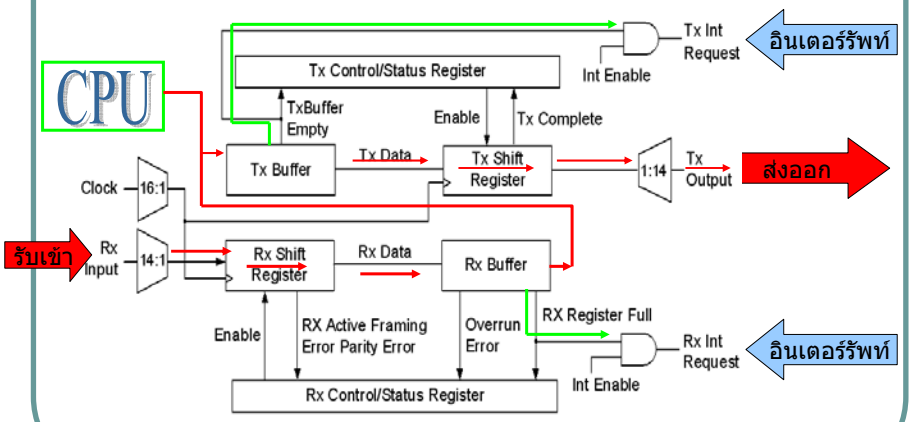
timer16_1int.asm

```
_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
   ;---------------------------------------------------
```

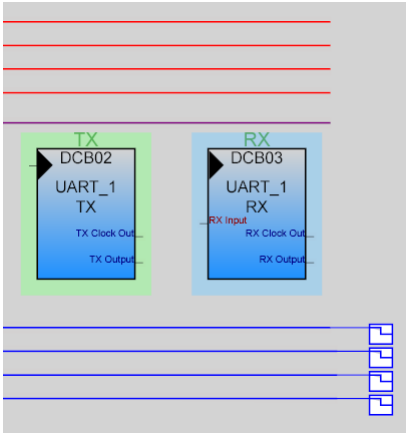
การสื่อสารแบบอนุกรม UART



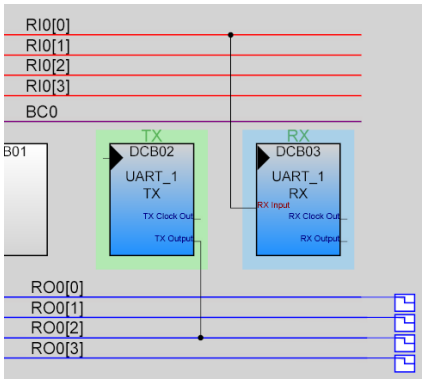
การสื่อสารแบบอนุกรม UART ของ PSoC

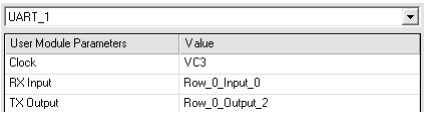


การจัดวาง UART



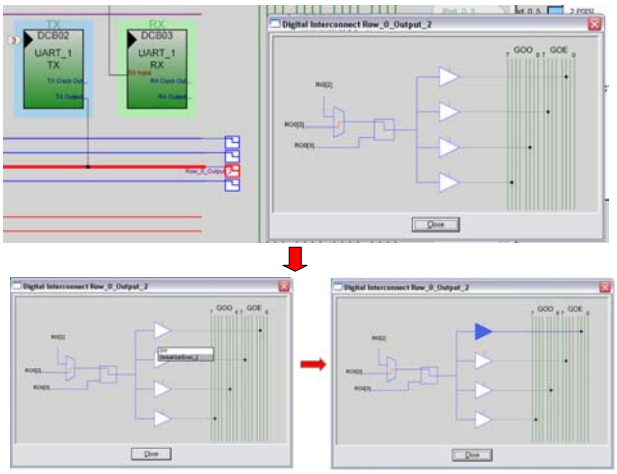
ตำแหน่งสัญญาณ Tx และ Rx



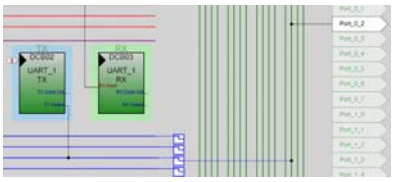


TX Output → Row_0_Output_2 (RO0[2])
RX Input → Row_0_Input_0 (RI0[0])

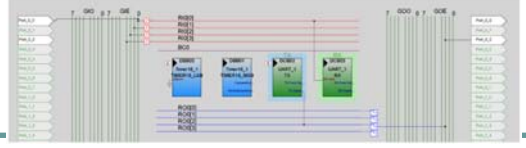
จัดตำแหน่งให้ขา Tx และ Rx



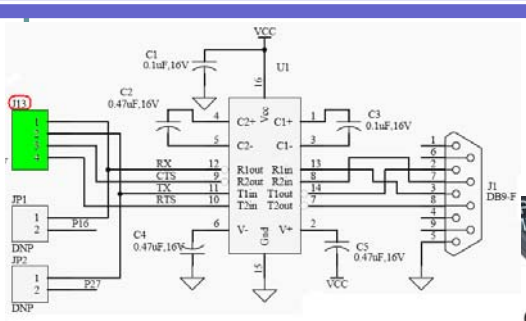
จัดตำแหน่งให้ขา Tx และ Rx



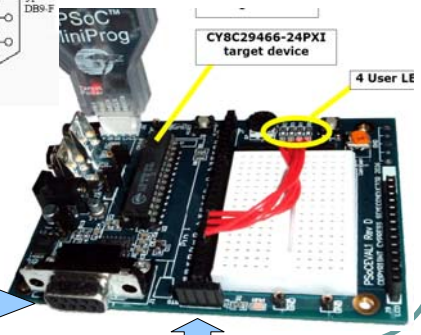
ส่วนการต่อสัญญาณ RX_Input ก็ทำในทำนองเดียวกัน ให้ต่อออกยังขา Port_0_0



ขาต่อ TX และ RX ของ UART



สรุป
P00 เข้ากับขา RX
P02 เข้ากับขา TX



DB9 UART Connector

J13

การกำหนดค่าพารามิเตอร์

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

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

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

VC3 source = VC1

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

API ของ UART

- UART_CmdReset
- UART_IntCntl
- UART_Start
- UART_PutString
- UART_CPutString
- UART_PutChar
- UART_PutCRLF
- UART_bCmdCheck

UART_CmdReset

ฟังก์ชั่น UART_CmdReset

รายละเอียด

Resets command buffer and flags. This allows characters for the next command to be accepted.

การใช้งานด้วยภาษา C

void UART_CmdReset(void)

พารามิเตอร์

ไม่มี

ค่าส่งกลับ

ไม่มี

UART_IntCntl

ฟังก์ชั่น UART_IntCntl

รายละเอียด

Enables/Disables the RX and TX interrupts independently.

Mask	Description
UART_ENABLE_RX_INT	Enable Receiver
UART_ENABLE_TX_INT	Enable Transmitter
UART_DISABLE_RX_INT	Disable Receiver
UART_DISABLE_TX_INT	Disable Transmitter

การใช้งานด้วยภาษา C

void UART_IntCntl(BYTE bMask)

พารามิเตอร์

BYTE bMask: Mask to enable or disable TX and RX blocks

ค่าส่งกลับ

ไม่มี

UART_Start

ฟังก์ชั่น UART_Start

รายละเอียด

Sets the parity and enables the UART receiver and transmitter. Once enabled, data can be received and transmitted

การใช้งานด้วยภาษา C

void UART_Start(BYTE bParitySetting)

พารามิเตอร์

bParitySetting: One byte that specifies the transmit parity. Symbolic names provided in C and assembly, and their associated values are given in the following table.

TX Parity	Value
UART_PARITY_NONE	0x00
UART_PARITY_EVEN	0x02
UART_PARITY_ODD	0x06

ค่าส่งกลับ

ไม่มี

UART_PutString

ฟังก์ชั่น UART_PutString

รายละเอียด

Sends a null terminated (RAM) string to the UART TX.

การใช้งานด้วยภาษา C

void UART_PutString(char * szRamString)

พารามิเตอร์

char * aRamString: Pointer to the string to be sent to the UART TX. MSB is passed in the Accumulator and LSB is passed in X register.

ค่าส่งกลับ

ไม่มี

UART_CPutString

ฟังก์ชั่น UART_CPutString

รายละเอียด

Sends constant (ROM), null terminated string out the UART TX port

การใช้งานด้วยภาษา C

void UART_CPutString(const char * szROMString)

พารามิเตอร์

const char * szROMString: Pointer to string to be sent to the UART. MSB of string pointer is passed in the Accumulator and LSB of the pointer is passed in the X register.

ค่าส่งกลับ

ไม่มี

UART_PutChar

ฟังก์ชั่น UART_PutChar

รายละเอียด

Writes single character to UART TX port when the port buffer is empty.

การใช้งานด้วยภาษา C

void UART_PutChar(CHAR cData)

พารามิเตอร์

CHAR cData: Character to be sent to the UART TX port. Data is passed in the Accumulator

ค่าส่งกลับ

ไม่มี

UART_PutCRLF

ฟังก์ชั่น UART_PutCRLF

รายละเอียด

Function to send out carriage return (0x0D) and line feed (0x0A) out of the UART TX port.

การใช้งานด้วยภาษา C

void UART_PutCRLF(void)

พารามิเตอร์

ไม่มี

ค่าส่งกลับ

ไม่มี

UART_bCmdCheck

ฟังก์ชั่น UART_bCmdCheck

รายละเอียด

Checks if command terminator has been received

การใช้งานด้วยภาษา C

BYTE UART_bCmdCheck(void)

พารามิเตอร์

ไม่มี

ค่าส่งกลับ

A non-zero value will be returned if a command terminator has been received

จบแล้ว

ถึงจะมืน
ครูสอนโลกก็หมุน
ดาวก็ยังเคลื่อน
เดือนก็ยังคล้อย

สู้ไว้ สู้ไว้ สู้ไว้