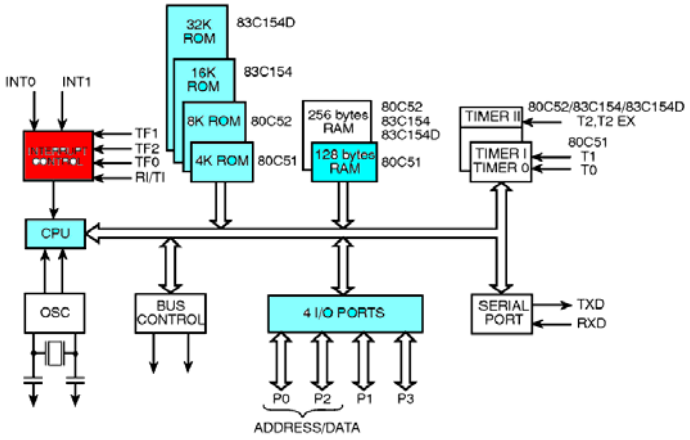
11.ระบบอินเตอร์รัพท์ของ 8051

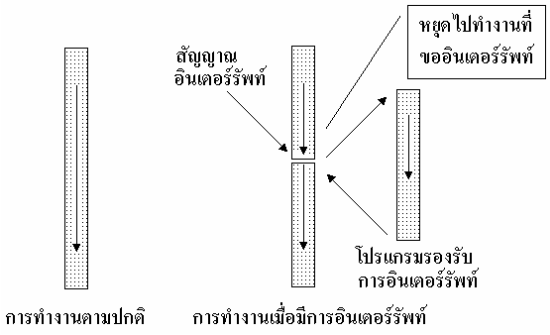
- อินเตอร์รัพท์คืออะไร
- ระบบอินเตอร์รัพท์ของ 8051
- โครงสร้างการอินเตอร์รัพท์
- ตำแหน่งโปรแกรมรองรับการอินเตอร์รัพท์
- การกำหนดสถานะการอินเตอร์รัพท์
- ลำดับการตรวจสอบสัญญาณอินเตอร์รัพท์
- ระดับความสำคัญของการอินเตอร์รัพท์
- เกิดอะไรขึ้นเมื่อมีสัญญาณอินเตอร์รัพท์
- เกิดอะไรขึ้นเมื่อเสร็จสิ้นการอินเตอร์รัพท์
- Common Bugs in Interrupts

ระบบอินเตอร์รัพท์ของ 8051



การอินเตอร์รัพท์คืออะไร ?

เมื่อมีเหตุการณ์ใดเหตุการณ์หนึ่งเกิดขึ้นแล้วไปขัดจังหวะการทำงานปกติของซีพียูเรา เรียกว่าการอินเตอร์รัพท์ การขัดจังหวะของเหตุการณ์นั้นอาจทำให้ซีพียูต้องทำงานอื่นก่อนเมื่อเสร็จแล้วจึงจะกลับไปทำงานเดิมหรือไม่มีก็ได้



ระบบอินเตอร์รัพท์ของ 8051

ประเภทของการอินเตอร์รัพท์

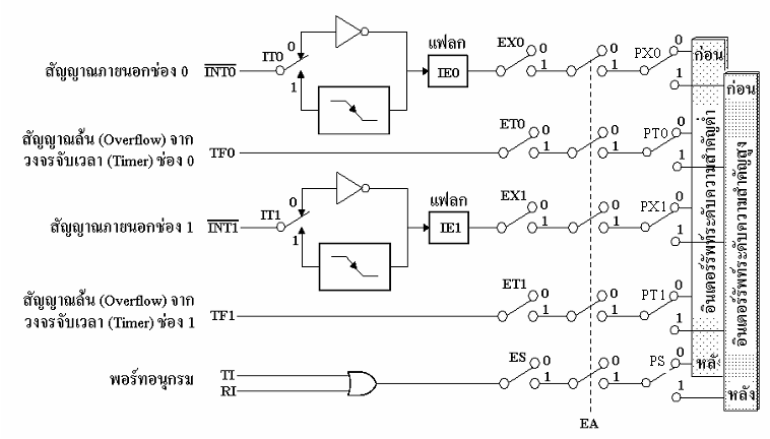
- **สัญญาณอินเตอร์รัพท์จากภายนอก** การตรวจสอบสัญญาณที่มา อินเตอร์รัพท์นี้ จะสามารถกำหนดให้มีการตรวจสอบในลักษณะ เมื่อได้มีการเปลี่ยนแปลงระดับสัญญาณ (Level-sensitive) ไปแล้ว หรือในช่วงเวลาขณะเริ่มมี การเปลี่ยนแปลงสัญญาณจาก logic สูงไปต่ำ (Edge-sensitive)
- **สัญญาณอินเตอร์รัพท์จากภายใน** แหล่งกำเนิดสัญญาณนี้จะเป็นวงจรภายใน Microcontroller เอง เช่น วงจรนับ/จับเวลา วงจรเชื่อมต่อสัญญาณอนุกรม เป็นต้น

โครงสร้างการอินเตอร์รัพท์

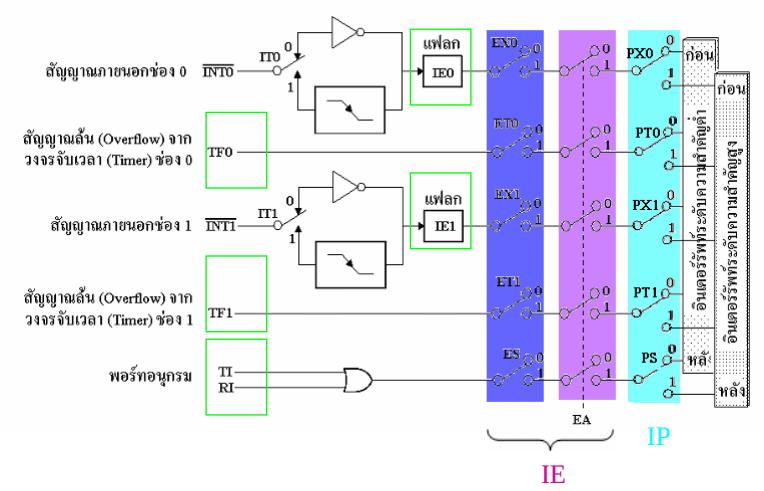
เกิดได้ 5 ลักษณะ คือ

1. **INT0** สัญญาณอินเตอร์รัพท์ จากภายนอกทางขาสัญญาณ P3.2 โดย 8051 จะทำการสุ่มตัวอย่าง สัญญาณเมื่อสิ้นสุดทุก Machine Cycle (/INT0)
2. **INT1** สัญญาณอินเตอร์รัพท์จากภายนอกทางขาสัญญาณ P3.3 โดย 8051 จะทำการสุ่มตัวอย่าง สัญญาณเมื่อสิ้นสุดทุก Machine Cycle (/INT1)
3. **Timer0** สัญญาณการเกิด Overflow ของ Timer 0 (TF0)
4. **Timer1** สัญญาณการเกิด Overflow ของ Timer 1 (TF1)
5. **Serial Port** การเกิด interrupt ที่เกิดขึ้นจากการรับ/ส่งข้อมูลอนุกรม ทำให้มีผลต่อแฟลกอินเตอร์รัพท์ RI และ TI ตามลำดับ

โครงสร้างระบบอินเตอร์รัพท์ของ 8051



โครงสร้างระบบอินเตอร์รัพท์ของ 8051



ตำแหน่งโปรแกรมรองรับการอินเตอร์รัพท์

Interrupt	Flag	ตำแหน่งเริ่มต้น
External 0	**IE0**	**0003h**
Timer 0	**TF0**	**000Bh**
External 1	**IE1**	**0013h**
Timer 1	**TF1**	**001Bh**
Serial	**RI/TI**	**0023h**

เช่นถ้าเกิดการอินเตอร์รัพท์จาก Timer 0 แล้วให้ทำการคอมพลีเมนท์ค่าที่พอร์ท 1 บิต 0 ก็สามารถเขียนโปรแกรมรองรับการอินเตอร์รัพท์ได้ดังนี้

```
                ORG     0000H
                :
                :
                :
                ORG     000BH
Timer0_Ser:     CPL     P1.0
                RET
```

การกำหนดสถานะการอินเตอร์รัพท์

เมื่อเริ่มต้นการทำงาน ซีพียูจะถูกรีเซ็ทสถานะการอินเตอร์รัพท์ของทุกสัญญาณเป็น Disable คือไม่สามารถขัดจังหวะการทำงานของซีพียูได้ ดังนั้นถ้าต้องการให้สัญญาณใดสามารถขัดจังหวะการทำงานของซีพียูได้ ต้องกำหนดสถานะการอินเตอร์รัพท์ของสัญญาณนั้นให้เป็น Enable และต้องกำหนดสถานะการอินเตอร์รัพท์โดยรวมให้เป็น Enable ด้วย สถานะเหล่านี้กำหนดได้ที่รีจิสเตอร์ IE (Interrupt Enable Register) หรือ รีจิสเตอร์ตำแหน่งที่ A8H

IE (Interrupt Enable Register) แอดเดรส A8H

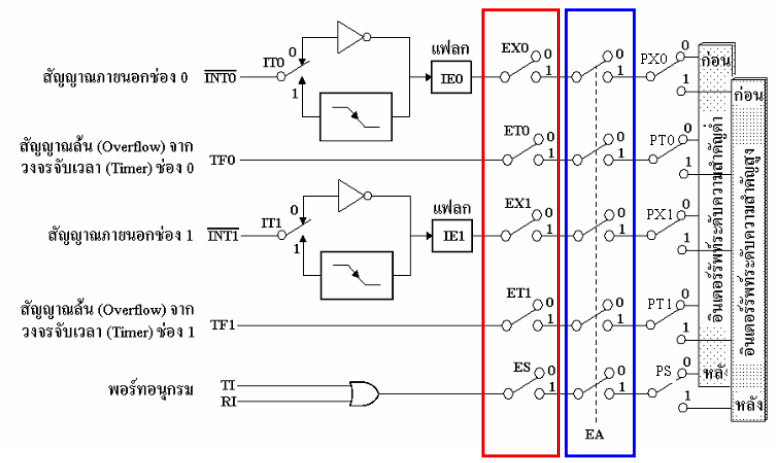
7	6	5	4	3	2	1	0
EA	-	-	ES	ET1	EX1	ET0	EX0

0 = Disable
1 = Enable

บิต	ชื่อ	ตำแหน่งบิต	ความหมายของฟังก์ชั่น
7	EA	AFh	ใช้กำหนดสถานะการอินเตอร์รัพท์โดยรวม
6	-	AEh	สำรอง
5	-	ADh	สำรอง
4	ES	ACh	ใช้กำหนดสถานะการอินเตอร์รัพท์ของพอร์ทอนุกรม
3	ET1	ABh	ใช้กำหนดสถานะการอินเตอร์รัพท์ของวงจรจับเวลาช่อง 1 (Timer 1)
2	EX1	AAh	ใช้กำหนดสถานะการอินเตอร์รัพท์ของสัญญาณภายนอกช่อง 1
1	ET0	A9h	ใช้กำหนดสถานะการอินเตอร์รัพท์ของวงจรจับเวลาช่อง 0 (Timer 0)
0	EX0	A8h	ใช้กำหนดสถานะการอินเตอร์รัพท์ของสัญญาณภายนอกช่อง 0

โครงสร้างระบบอินเตอร์รัพท์ของ 8051

7	6	5	4	3	2	1	0
EA	-	-	ES	ET1	EX1	ET0	EX0



เช่นถ้าต้องการให้ Tiomer 0 สามารถอินเตอร์รัพท์ได้เพียงสัญญาณเดียว รีจิสเตอร์ IE ต้องมีค่าดังนี้

IE =

7	6	5	4	3	2	1	0
EA	-	-	ES	ET1	EX1	ET0	EX0
1	0	0	0	0	0	1	0

= 82H

และเขียนโปรแกรมดังนี้

```
MOV     IE,#82H
```

หรือ

```
SETB    EA
SETB    ET0
```

TCON (Timer Control Register) แอดเดรส 88H

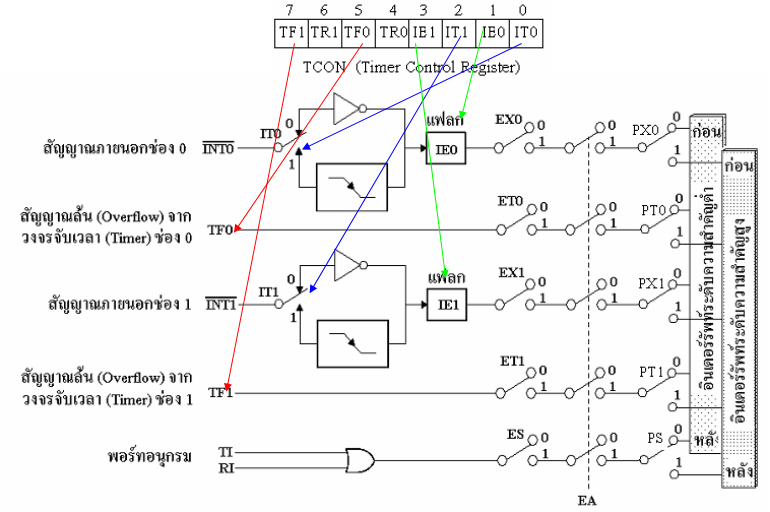
7	6	5	4	3	2	1	0
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

TCON (Timer Control Register)

การกำหนดลักษณะของสัญญาณภายนอกเพื่ออินเตอร์รัพท์ซีพียู

บิต	ชื่อ	ตำแหน่งบิต	ความหมายของฟังก์ชั่น
7	TF1	8Fh	Timer 1 Overflow. บิตนี้จะถูกเซ็ตเมื่อ Timer 1 Overflow
6	TR1	8Eh	Timer 1 Run เมื่อบิตนี้เซ็ต Timer 1 จะทำงาน แต่ถ้าบิตเป็น 0 จะหยุด
5	TF0	8Dh	Timer 0 Overflow. บิตนี้จะถูกเซ็ตเมื่อ Timer 0 Overflow
4	TR0	8Ch	Timer 0 Run เมื่อบิตนี้เซ็ต Timer 0 จะทำงาน แต่ถ้าบิตเป็น 0 จะหยุด
3	IE1	8Bh	แฟลกแสดงการอินเตอร์รัพท์ของ INT1
2	IT1	8Ah	บิตเลือกประเภทสัญญาณอินเตอร์รัพท์ เป็น 0 = ระดับ ถ้าเป็น 1 = ขอบ
1	IE0	89h	แฟลกแสดงการอินเตอร์รัพท์ของ INT0
0	IT0	88h	บิตเลือกประเภทสัญญาณอินเตอร์รัพท์ เป็น 0 = ระดับ ถ้าเป็น 1 = ขอบ

โครงสร้างระบบอินเตอร์รัพท์ของ 8051



ลำดับการตรวจสอบสัญญาณอินเตอร์รัพท์

8051 จะทำการตรวจสอบสัญญาณอินเตอร์รัพท์ต่างๆโดยอัตโนมัตทุกๆครั้งที่สิ้นสุดการทำคำสั่งแต่ละคำสั่ง ลำดับการตรวจสอบสัญญาณเป็นดังนี้:

1. External 0 Interrupt
2. Timer 0 Interrupt
3. External 1 Interrupt
4. Timer 1 Interrupt
5. Serial Interrupt

นั้นมีความหมายว่า ถ้าสัญญาณอินเตอร์รัพท์จากพอร์ทอนุกรมเกิดขึ้นพร้อมๆกับสัญญาณอินเตอร์รัพท์จากภายนอกช่องที่ 0 งานของสัญญาณอินเตอร์รัพท์ภายนอกช่องที่ 0 จะได้รับการตอบสนองก่อนและเมื่อเสร็จงานจึงจะไปทำงานของพอร์ทอนุกรม

ระดับความสำคัญของการอินเตอร์รัพท์

นอกจากการตรวจสอบสัญญาณอินเตอร์รัพท์ตามลำดับแล้ว 8051 ยังให้เราสามารถเลือกกำหนดระดับความสำคัญของสัญญาณอินเตอร์รัพท์ได้ โดยให้จัดเป็นกลุ่มที่มีความสำคัญสูงกับกลุ่มที่มีความสำคัญต่ำ เช่นถ้าเราจัดให้สัญญาณอินเตอร์รัพท์จากพอร์ทอนุกรมมีความสำคัญสูงกว่าตัวอื่น เมื่อเกิดการอินเตอร์รัพท์ขึ้นพร้อมกับสัญญาณอื่นๆ เช่น สัญญาณอินเตอร์รัพท์จาก Timer 0 เกิดพร้อมกับสัญญาณจากพอร์ทอนุกรม ลักษณะเช่นนี้ ซีพียูจะตอบสนองกับสัญญาณที่มีระดับความสำคัญสูงก่อนคือสัญญาณจากพอร์ทอนุกรม เมื่อทำงานเสร็จแล้วจึงจะมาทำของสัญญาณที่มีระดับความสำคัญต่ำกว่า แต่ถ้าเป็นสัญญาณที่มีระดับความสำคัญเท่ากัน ซีพียูก็จะพิจารณาเรียงตามลำดับการตรวจสอบก่อนหลัง รีจิสเตอร์ที่ใชกำหนดระดับความสำคัญคือ IP (Interrupt priorities) มีตำแหน่งเป็น B8h และมีรูปแบบดังนี้

IP Interrupt Priorities แอดเดรส B8H

7	6	5	4	3	2	1	0
-	-	-	PS	PT1	PX1	PT0	PX0

IP Interrupt Priority

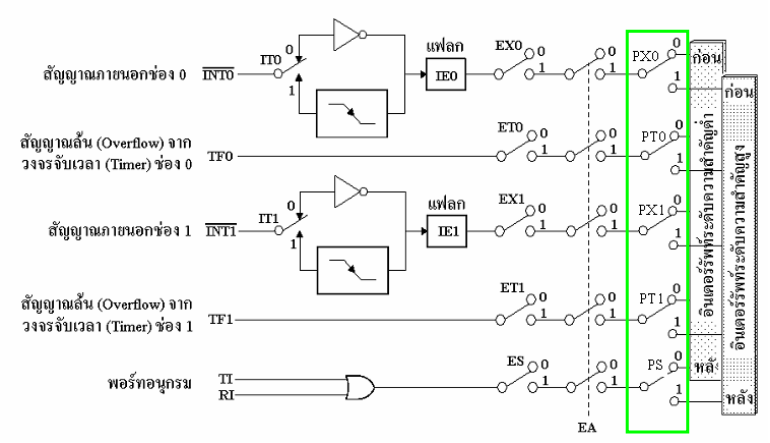
บิต	ชื่อ	ตำแหน่งบิต	ความหมายของฟังก์ชั่น
7	-	-	ไม่กำหนด
6	-	-	ไม่กำหนด
5	-	-	ไม่กำหนด
4	PS	BCh	Serial Interrupt Priority
3	PT1	BBh	Timer 1 Interrupt Priority
2	PX1	BAh	External 1 Interrupt Priority
1	PT0	B9h	Timer 0 Interrupt Priority
0	PX0	B8h	External 0 Interrupt Priority

ถ้าบิตเป็นโลจิก 0 แสดงว่ามีระดับความสำคัญต่ำ
ถ้าบิตเป็นโลจิก 1 แสดงว่ามีระดับความสำคัญสูง

โครงสร้างระบบอินเตอร์รัพท์ของ 8051

7	6	5	4	3	2	1	0
-	-	-	PS	PT1	PX1	PT0	PX0

IP Interrupt Priority



เกิดอะไรขึ้นเมื่อมีสัญญาณอินเตอร์รัพท์

เมื่อมีสัญญาณอินเตอร์รัพท์ปรากฏขึ้นที่ซีพียู ซีพียูจะมีการตอบสนองดังนี้

1. ค่าตัวนับโปรแกรม (PC) ถูกเก็บลงสแตคโดยเก็บไบต์ต่ำก่อน
2. สำหรับอินเตอร์รัพท์อื่นที่เกิดขึ้นพร้อมกันแต่มีระดับความสำคัญต่ำกว่าจะถูกบล็อก
3. ในกรณีของอินเตอร์รัพท์ที่มาจาก Timer และ อินเตอร์รัพท์จากภายนอก แฟลกของการอินเตอร์รัพท์ถูกทำให้เป็น 0 ก่อนที่จะไปขั้นตอนที่ 4 นั้น หมายความว่าผู้ใช้ไปจำเป็นต้อง เคลียร์แฟลกนี้ก่อน
4. ซีพียูจะทำงานโปรแกรมรองรับการอินเตอร์รัพท์

เกิดอะไรขึ้นเมื่อเสร็จสิ้นการอินเตอร์รัพท์

เมื่อซีพียูทำโปรแกรมรองรับอินเตอร์รัพท์จนถึงคำสั่ง RETI (Return from Interrupt) ซึ่งเป็นคำสั่งสิ้นสุดของโปรแกรมรองรับอินเตอร์รัพท์ ซีพียูจะทำงานดังต่อไปนี้

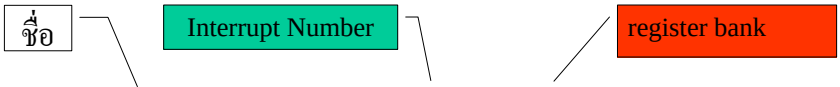
1. ซีพียูจะคืนค่า 2 ไบต์จากสแตคให้กับตัวนับโปรแกรม
2. สถานะการอินเตอร์รัพท์จะกลับคืนเหมือนเดิม

ตัวอย่างคำสั่งการรองรับอินเตอร์รัพท์จาก INT0

```
                ORG     0000H
                SJMP    START
                ORG     03H         ;Start address of external interrupt 0
                LJMP    INT0_SERV
START:          SETB    EA          ;Set enable global interrupt
                SETB    IT0         ;Set falling edge type for INT0
                SETB    EX0         ;Enable External INT0
                :
                :
INT0_SERV:      PUSH    ACC
                PUSH    PSW
                MOV     A,30H
                :
                :
                POP     PSW
                POP     ACC
                RETI

                END
```

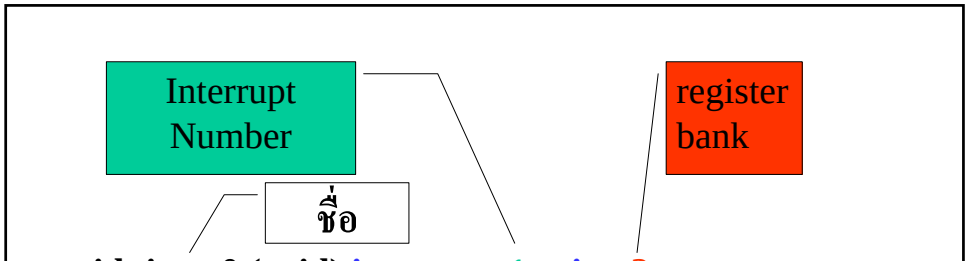
การเขียนโปรแกรมรองรับการอินเตอร์รัพท์ด้วยภาษาซี




```
void timer0 (void) interrupt 1 using 2
{
}
```

Interrupt Number	Interrupt Description	Address
0	EXTERNAL INT 0	0003h
1	TIMER/COUNTER 0	000Bh
2	EXTERNAL INT 1	0013h
3	TIMER/COUNTER 1	001Bh
4	SERIAL PORT	0023h

รีจิสแตอร์แบงค์นี้ถ้าไม่มีการเรียกใช้ก็จะไม่มีการเลือกแบงค์เพื่อให้ขนาดเล็ก




```
void timer0 (void) interrupt 1 using 2
{
        if (++interruptcnt == 4000)
        { /* count to 4000 */
                second++; /* second counter */
                interruptcnt = 0; /* clear int counter */
        }
}
```