

Department of Computer Engineering
Faculty of Engineering
Prince of Songkla University

240-319

Embedded System Developer Module



March 23, 2023

Associate Prof. Dr. Panyayot Chaikan
panyayot@coe.psu.ac.th



Chapter 3

การอินเทอร์เน็ต





เนื้อหา

ทฤษฎีพื้นฐานของการขัดจังหวะ

แหล่งสัญญาณขัดจังหวะจากภายนอกและภายในตัวประมวลผล

โปรแกรมบริการการขัดจังหวะ (Interrupt Service Routine : ISR)

เวกเตอร์การขัดจังหวะ

ลำดับความสำคัญของการขัดจังหวะ

เรจิสเตอร์สำหรับควบคุมการขัดจังหวะ

การขัดจังหวะจากขา INT0 และ INT1

การขัดจังหวะแบบพินเชนจ์ (Pin-change Interrupt)



คำถามก่อนเรียน

- ◆ เครื่องคอมพิวเตอร์รู้ได้อย่างไรว่ามีการกดคีย์บอร์ด
- ◆ หากต่อคีย์บอร์ดเข้ากับซีพียู AVR จะทำให้ซีพียูรู้ได้อย่างไรว่ามีการกดคีย์



วิธีการติดต่ออุปกรณ์ I/O

- ◆ การหยั่งสัญญาณ (Polling)
 - ◆ ซีพียูวนำตรวจสอบการทำงานของอุปกรณ์ I/O ตลอดเวลา
 - ◆ เสียเวลาการทำงานของซีพียูโดยเปล่าประโยชน์
- ◆ การขัดจังหวะ (Interrupt)
 - ◆ ซีพียูทำงานหลักของตนเองโดยไม่ต้องสนใจอุปกรณ์ I/O
 - ◆ เมื่อ I/O ต้องการติดต่อซีพียู จะส่งสัญญาณ Interrupt มาให้ซีพียูรับทราบ



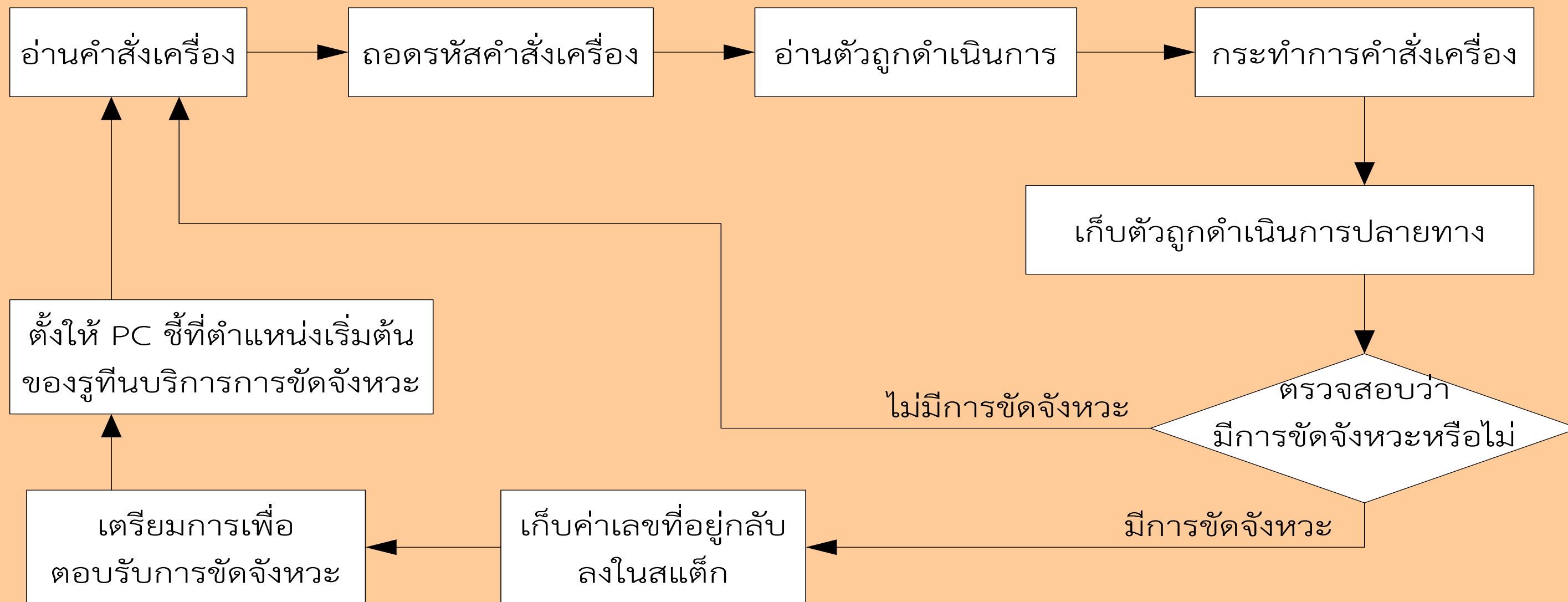
การขัดจังหวะคืออะไร?

- ◆ การขัดจังหวะ (Interrupts) คือ การทำให้ Microprocessor หรือ Microcontroller หยุดพักจากงานที่กระทำอยู่ในปัจจุบัน แล้วกระโดดไปทำงานอื่นงานหนึ่งจนเสร็จ จากนั้นจึงกระโดดกลับมาทำงานชิ้นเดิมที่หยุดพักไว้ต่อไป

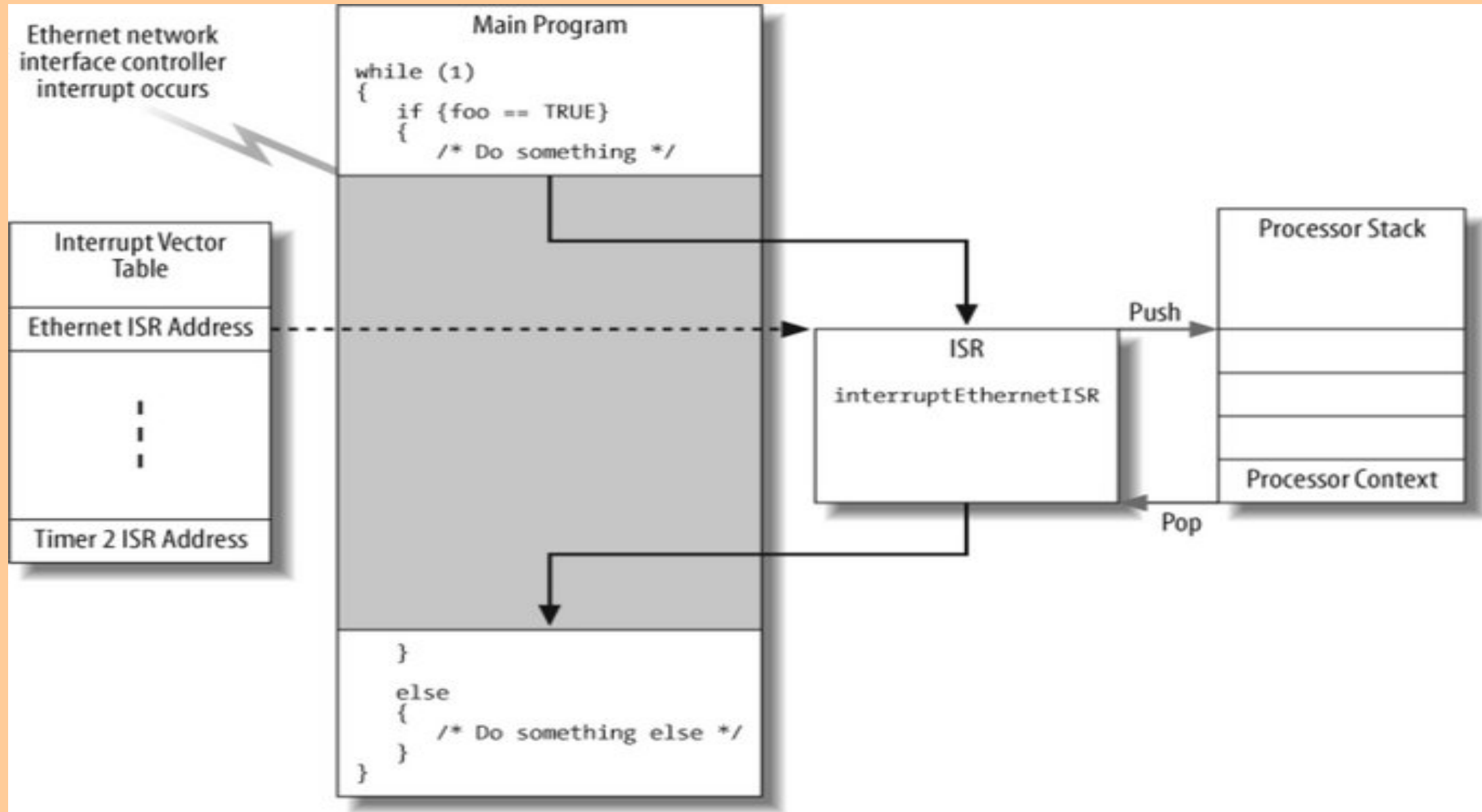




ลำดับขั้นตอนการทำงานของตัวประมวลผล



ตัวอย่างการขัดจังหวะ

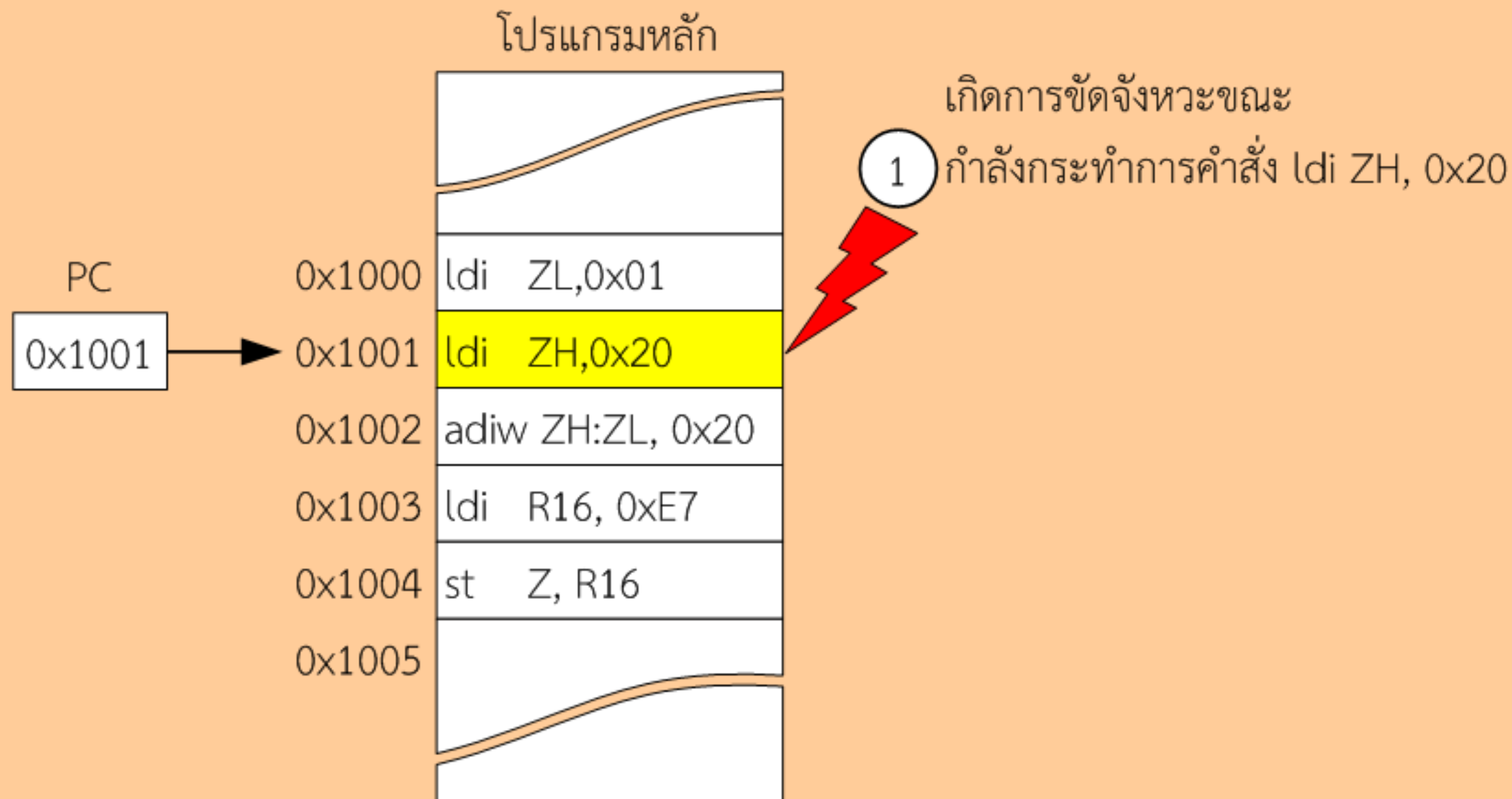


รูปจาก http://book.opensourceproject.org.cn/embedded/oreillyembed/opensource/0596009836/id-i_0596009836_chp_8_sect_3.html



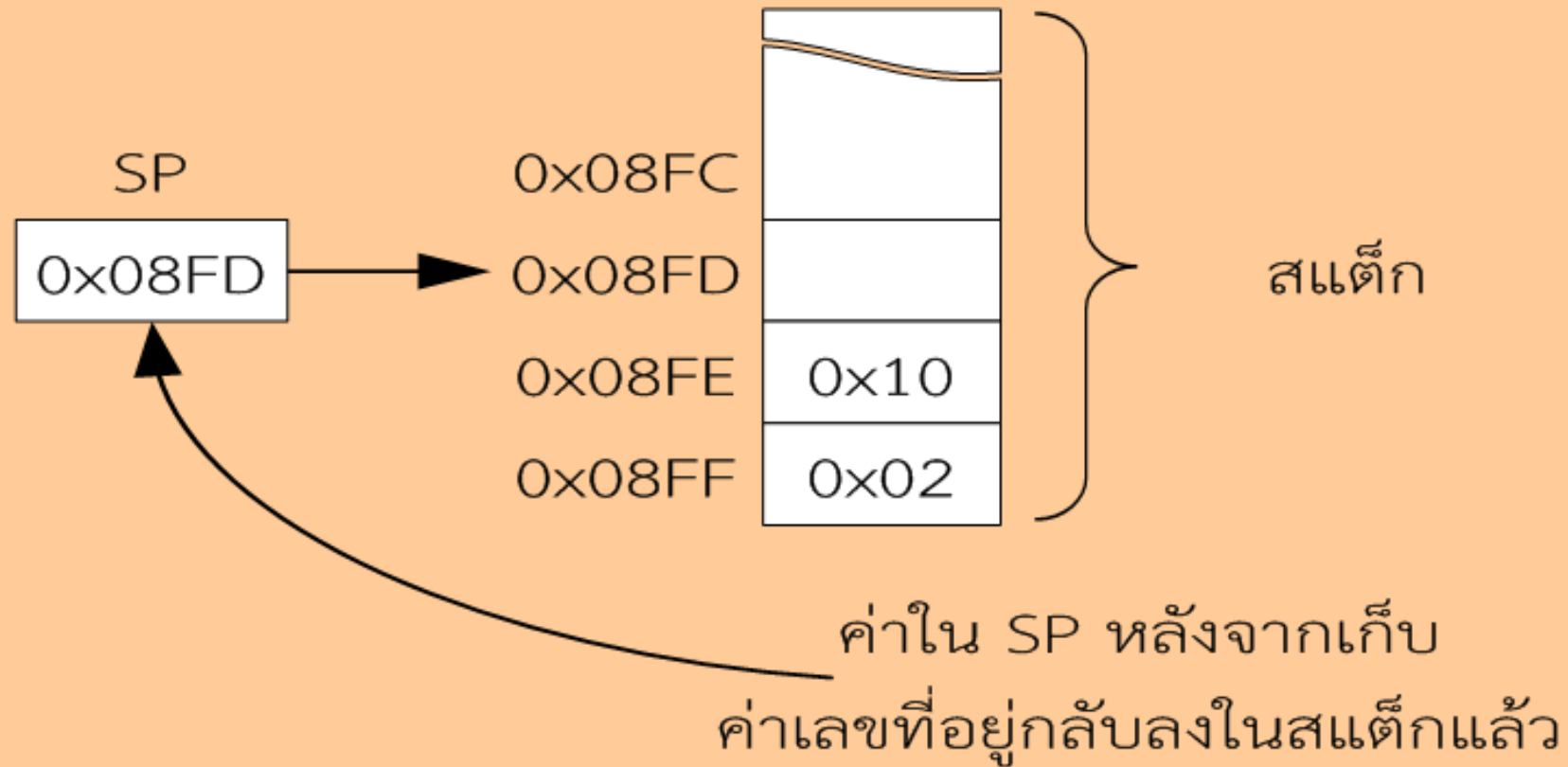


ตัวอย่างการขัดจังหวะในสถาปัตยกรรมเอวี่อาร์



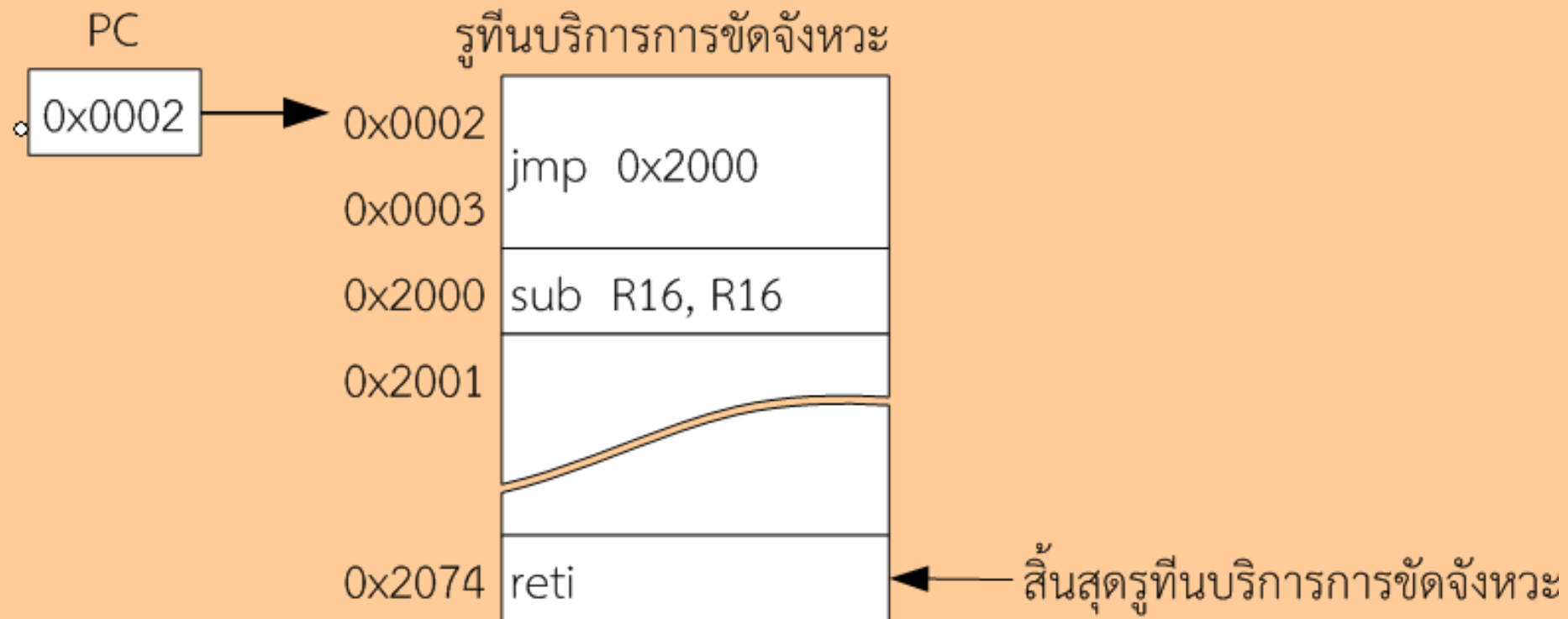
ตัวอย่างการขัดจังหวะในสถาปัตยกรรมเอวี่อาร์

- 2) เก็บค่าเลขที่อยู่กลับ (ค่า 0x1002)
ไว้ในสแต็ก จากนั้นเรจิสเตอร์ SP
ซึ่งเดิมชี้อยู่ที่ตำแหน่ง 0x08FF
เปลี่ยนมาชี้ที่ตำแหน่ง 0x08FD แทน

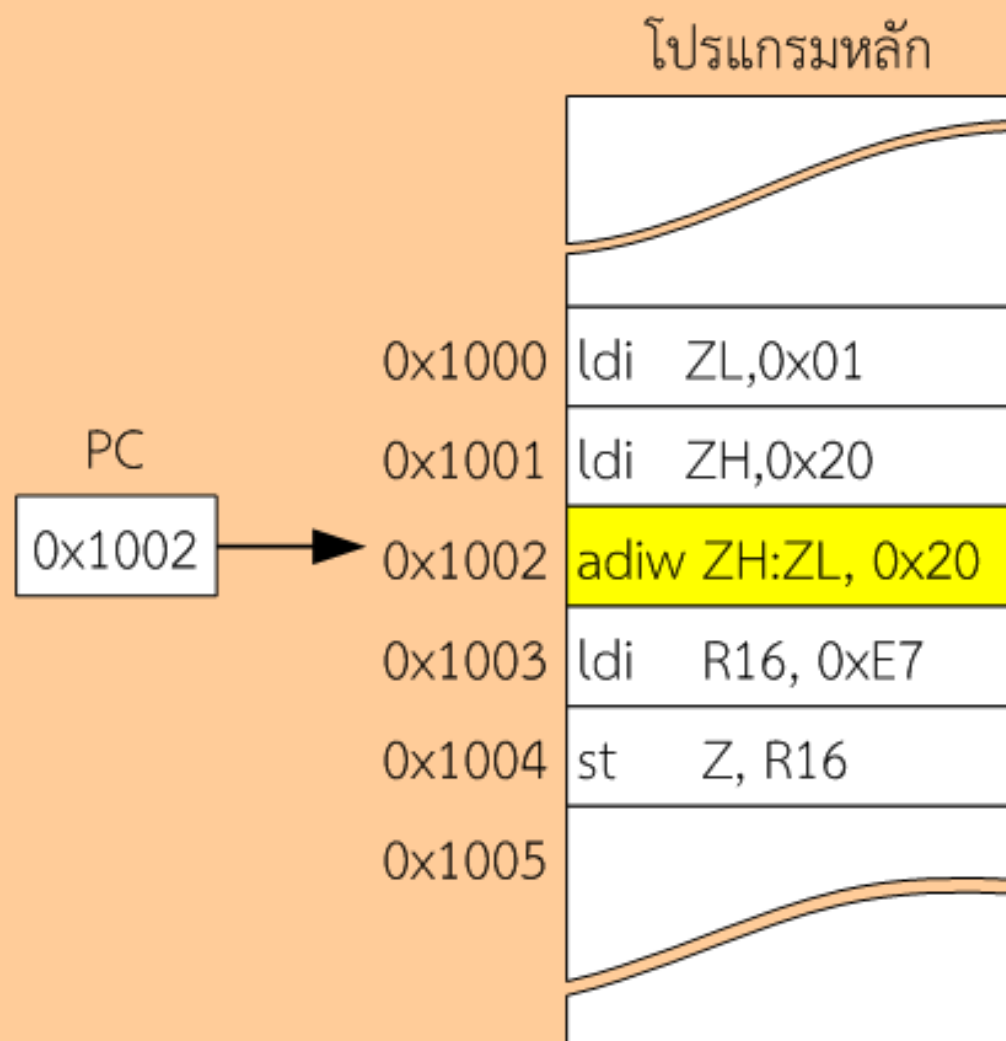


ตัวอย่างการขัดจังหวะในสถาปัตยกรรมเอวียาร์

- 3 จากนั้นตัวประมวลผลจะตั้งให้ PC ขึ้นมาที่ตำแหน่งเริ่มต้นของ รوتينบริการการขัดจังหวะ (ในที่นี้คือตำแหน่ง 0x0002) และอ่านคำสั่งของรูทีนนี้ขึ้นมาทำงาน



ตัวอย่างการขัดจังหวะในสถาปัตยกรรมเอวียาร์

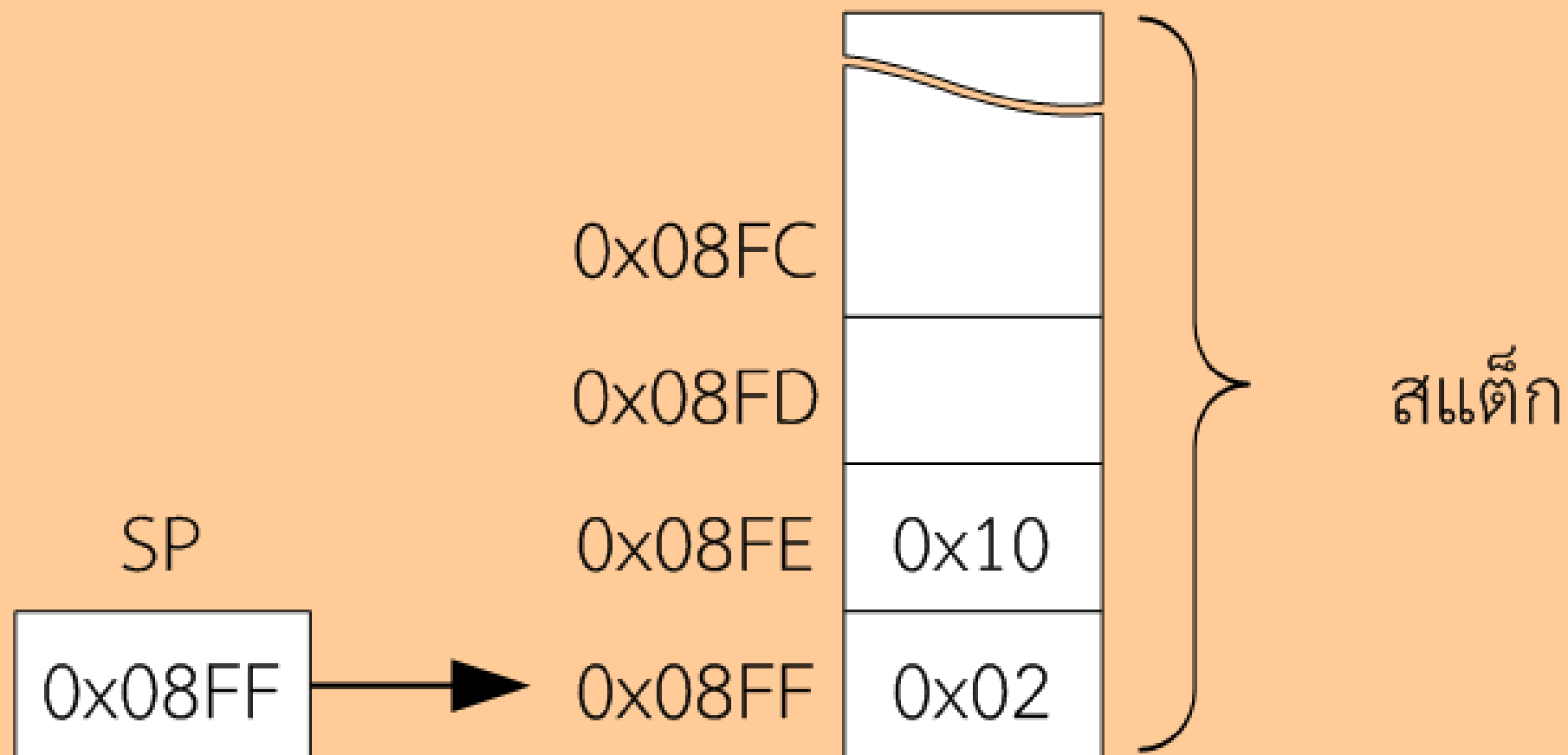


4

หลังจากสิ้นสุดการทำงานของรูทีนบริการการขัดจังหวะ ตัวประมวลผลจะอ่านค่าเลขที่อยู่กลับจากสแต็กขึ้นมา ใส่กลับในเรจิสเตอร์ PC ส่งผลให้คำสั่ง adi w ที่อยู่ในหน่วยความจำตำแหน่ง 0x1002 ได้รับการกระทำการต่อไป



ตัวอย่างการขัดจังหวะในสถาปัตยกรรมเอวี่อาร์



ค่าใน SP และในสแต็กหลังจาก

ตัวประมวลผลกลับมากระทำการโปรแกรมหลัก





ประเภทของการขัดจังหวะ

- ◆ การขัดจังหวะจากภายนอก (External Interrupt) เป็นการตรวจสอบสัญญาณที่รับมาจากภายนอกตัวไมโครคอนโทรลเลอร์
- ◆ การขัดจังหวะจากภายใน (Internal Interrupt) เป็นการตรวจสอบสัญญาณที่แหล่งกำเนิดสัญญาณเกิดจากวงจรภายในไมโครคอนโทรลเลอร์เอง



เวกเตอร์การขัดจังหวะของ ATMEGA328P

V_N	S_A	ชนิดของ การขัดจังหวะ	คำอธิบาย
1	0x0000	RESET	เกิดการรีเซ็ตตัวประมวลผล
2	0x0002	INT0	การขัดจังหวะจากภายนอกผ่านทางขาสัญญาณ INT0
3	0x0004	INT1	การขัดจังหวะจากภายนอกผ่านทางขาสัญญาณ INT1
4	0x0006	PCINT0	การขัดจังหวะแบบพินเซนจ์ที่ขาสัญญาณ PCINT[7:0]
5	0x0008	PCINT1	การขัดจังหวะแบบพินเซนจ์ที่ขาสัญญาณ PCINT[14:8]
6	0x000A	PCINT2	การขัดจังหวะแบบพินเซนจ์ที่ขาสัญญาณ PCINT[23:16]
7	0x000C	WDT	การขัดจังหวะจากวงจรจับเวลาวิอตซ์ด็อก
8	0x000E	TIMER2 COMPA	การขัดจังหวะจากวงจรนับ/จับเวลาหมายเลข 2 เมื่อค่าในเรจิสเตอร์ TCNT2 เท่ากับค่าในเรจิสเตอร์ OCR2A
9	0x0010	TIMER2 COMPB	การขัดจังหวะจากวงจรนับ/จับเวลาหมายเลข 2 เมื่อค่าใน TCNT2 เท่ากับค่าในเรจิสเตอร์ OCR2B
10	0x0012	TIMER2 OVF	การขัดจังหวะจากวงจรนับ/จับเวลาหมายเลข 2 เมื่อเกิดการล้นของค่าในเรจิสเตอร์ TCNT2



เวกเตอร์การขัดจังหวะของ ATMEGA328P

V_N	S_A	ชนิดของ การขัดจังหวะ	คำอธิบาย
11	0x0014	TIMER1 CAPT	การขัดจังหวะจากหน่วยดักจับ (Input Capture Unit)
12	0x0016	TIMER1 COMPA	การขัดจังหวะจากวงจรนับ/จับเวลาหมายเลข 1 เมื่อค่าใน TCNT1 เท่ากับ ค่าในเรจิสเตอร์ OCR1A
13	0x0018	TIMER1 COMPB	การขัดจังหวะจากวงจรนับ/จับเวลาหมายเลข 1 เมื่อค่าใน TCNT1 เท่ากับ ค่าในเรจิสเตอร์ OCR1B
14	0x001A	TIMER1 OVF	การขัดจังหวะจากวงจรนับ/จับเวลาหมายเลข 1 เมื่อเกิดการล้นของค่าในเรจิสเตอร์ TCNT1
15	0x001C	TIMER0 COMPA	การขัดจังหวะจากวงจรนับ/จับเวลาหมายเลข 0 เมื่อค่าใน เรจิสเตอร์ TCNT0 เท่ากับค่าในเรจิสเตอร์ OCR0A
16	0x001E	TIMER0 COMPB	การขัดจังหวะจากวงจรนับ/จับเวลาหมายเลข 0 เมื่อค่าใน เรจิสเตอร์ TCNT0 เท่ากับค่าในเรจิสเตอร์ OCR0B
17	0x0020	TIMER0 OVF	การขัดจังหวะจากวงจรนับ/จับเวลาหมายเลข 0 เมื่อเกิดการล้นของค่าใน เรจิสเตอร์ TCNT0
18	0x0022	SPI, STC	การขัดจังหวะเมื่อวงจรรูสาร์ท (USART) ส่งข้อมูลอนุกรมแบบแอสซิงโครนัสเสร็จสิ้นลง
19	0x0024	USART, RX	การขัดจังหวะเมื่อวงจรรูสาร์ทรับข้อมูลเสร็จสิ้นจำนวน 1 เฟรม



เวกเตอร์การขัดจังหวะของ ATMEGA328P

V_N	S_A	ชนิดของ การขัดจังหวะ	คำอธิบาย
20	0x0026	USART, UDRE	การขัดจังหวะเมื่อค่าในเรจิสเตอร์ UDR ถูกส่งออกจนจรรยาสำหรับข้อมูลเสร็จสิ้นจำนวน 1 เฟรม
21	0x0028	USART, TX	การขัดจังหวะเมื่อวงจรรายส่งข้อมูลเสร็จสิ้นจำนวน 1 เฟรม
22	0x002A	ADC	การขัดจังหวะเมื่อวงจรแปลงแอนะล็อกเป็นดิจิทัลแปลงข้อมูลเสร็จ
23	0x002C	EE READY	การขัดจังหวะเมื่ออีอีพรอมพร้อมใช้งาน
24	0x002E	ANALOG COMP	การขัดจังหวะจากวงจรเปรียบเทียบสัญญาณแอนะล็อก
25	0x0030	TWI	การขัดจังหวะจากวงจรสื่อสารอนุกรมแบบทูไวร์ (2-wire)

หมายเหตุ

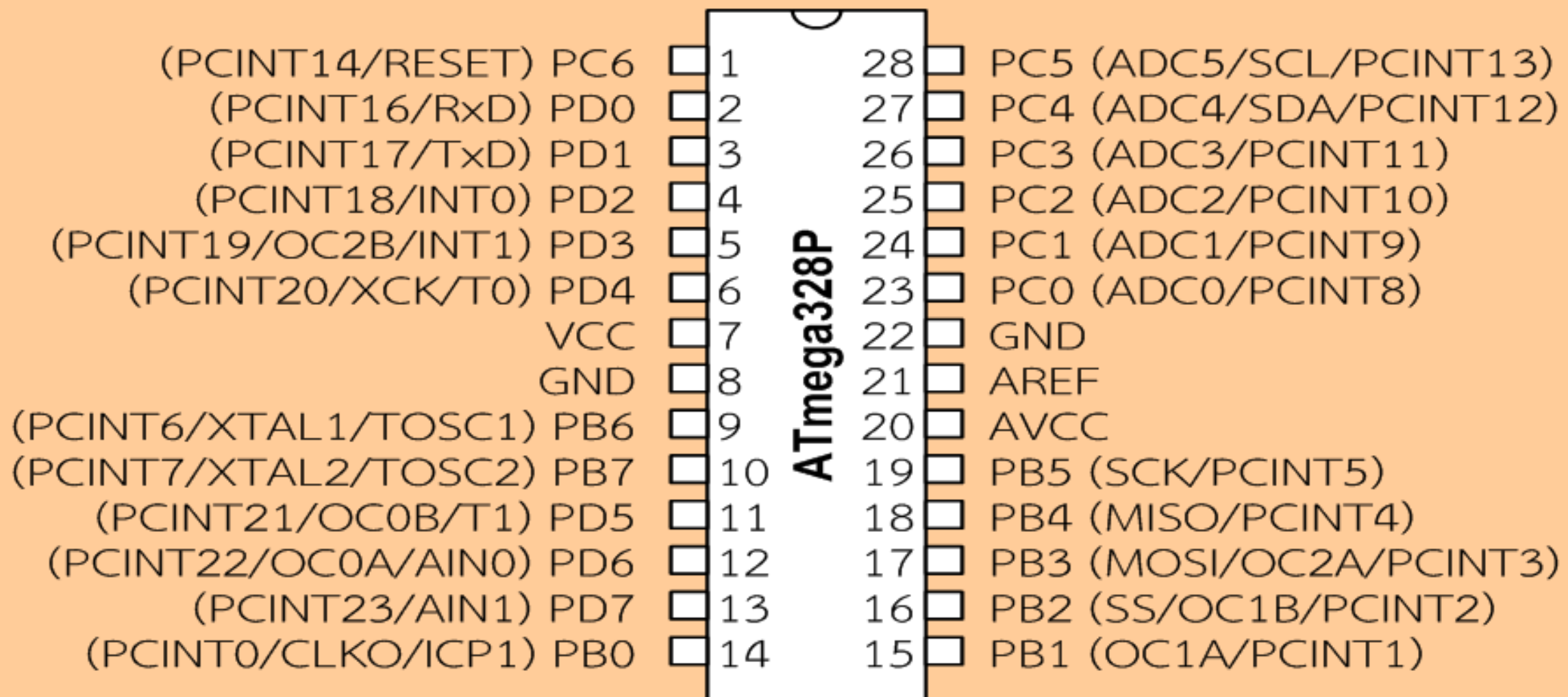
- V_N คือ หมายเลขเวกเตอร์การขัดจังหวะ (Interrupt vector number)
- S_A คือ ค่าหมายเลขที่อยู่เริ่มต้นของรูทีนบริการการขัดจังหวะ





การขัดจังหวะภายนอกของ ATMEGA328P

❖ คิว Reset และ INT0-INT1, และ PCINT0-PCINT2





การขัดจังหวะจากภายในตัวประมวลผล

◆ แหล่งกำเนิดสัญญาณขัดจังหวะภายใน

- ◆ Watch dog timer
- ◆ Timer / Counter
- ◆ Serial Communication
- ◆ วงจรเขียน EEPROM
- ◆ วงจรแปลงแอนะล็อกเป็นดิจิทัล

ลำดับความสำคัญของการขัดจังหวะ

- ◆ หากเกิดการขัดจังหวะจากแหล่งกำเนิดมากกว่า
1 แหล่งขึ้นไป ซีพียูจะให้บริการการขัดจังหวะใดก่อน

V _N	S _A	ชนิดของ การขัดจังหวะ	คำอธิบาย
1	0x0000	RESET	เกิดการรีเซ็ตตัวประมวลผล
2	0x0002	INT0	การขัดจังหวะจากภายนอกผ่านทางขาสัญญาณ INT0
3	0x0004	INT1	การขัดจังหวะจากภายนอกผ่านทางขาสัญญาณ INT1
4	0x0006	PCINT0	การขัดจังหวะแบบพินเซนจ์ที่ขาสัญญาณ PCINT[7:0]
5	0x0008	PCINT1	การขัดจังหวะแบบพินเซนจ์ที่ขาสัญญาณ PCINT[14:8]
6	0x000A	PCINT2	การขัดจังหวะแบบพินเซนจ์ที่ขาสัญญาณ PCINT[23:16]
7	0x000C	WDT	การขัดจังหวะจากวงจรจับเวลาวิอตซ์ด็อก
8	0x000E	TIMER2 COMPA	การขัดจังหวะจากวงจรนับ/จับเวลาหมายเลข 2 เมื่อค่าในเรจิสเตอร์ TCNT2 เท่ากับค่าในเรจิสเตอร์ OCR2A
9	0x0010	TIMER2 COMPB	การขัดจังหวะจากวงจรนับ/จับเวลาหมายเลข 2 เมื่อค่าใน TCNT2 เท่ากับค่าในเรจิสเตอร์ OCR2B
10	0x0012	TIMER2 OVF	การขัดจังหวะจากวงจรนับ/จับเวลาหมายเลข 2 เมื่อเกิดการล้นของค่าในเรจิสเตอร์ TCNT2

ความสำคัญสูง

ความสำคัญต่ำ





คำสั่งเปิดทาง/ปิดทางการขัดจังหวะส่วนกลาง

ภาษาแอสเซมบลี

◆ Interrupt Enable

```
sei
```

◆ Interrupt Disable

```
cli
```

ภาษาซี

◆ Interrupt Enable

```
sei();
```

◆ Interrupt Disable


```
cli();
```

◆ อย่าลืมสั่ง

```
#include <avr/interrupt.h>
```

Global interrupt (บิต I ใน SREG)

Status Register :SREG



บิตที่ 7	บิตที่ 6	บิตที่ 5	บิตที่ 4	บิตที่ 3	บิตที่ 2	บิตที่ 1	บิตที่ 0
I	T	H	S	V	N	Z	C

- ◆ Bit 7 : Global Interrupt Enable
- ◆ Bit 6 : Bit Copy Storage
- ◆ Bit 5 : Half Carry Flag
- ◆ Bit 4 : Sign Bit



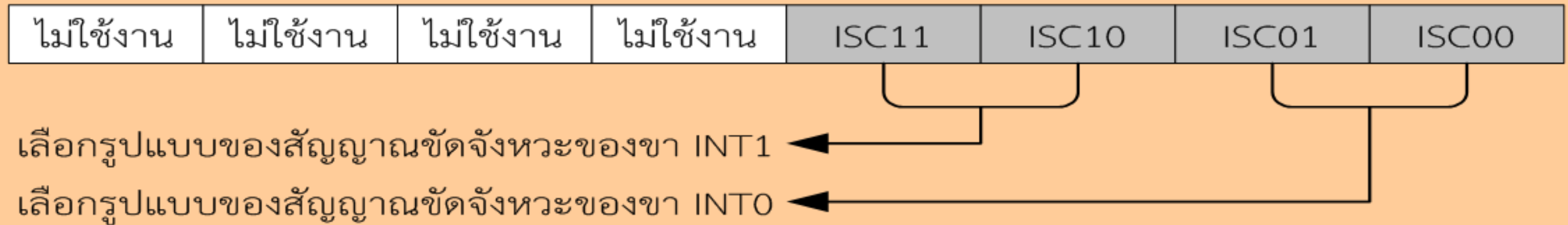
สิ่งที่ซีพียูทำเมื่อเกิดการขัดจังหวะ



- ◆ เก็บค่าในเรจิสเตอร์ PC (ซึ่งเป็นตำแหน่งแอดเดรสของคำสั่งที่ซีพียูจะกลับมาทำงานหลัง ISR เสร็จ) เอาไว้ใน Stack
- ◆ ปิดทางการขัดจังหวะ (ป้องกันการขัดจังหวะซ้อน)
- ◆ โหลด Interrupt Vector Address ของการขัดจังหวะที่ร้องขอขึ้นมาใส่ในเรจิสเตอร์ PC
- ◆ กระทำการคำสั่งใน ISR (interrupt service routine)



EICRA – External Interrupt Control Register A

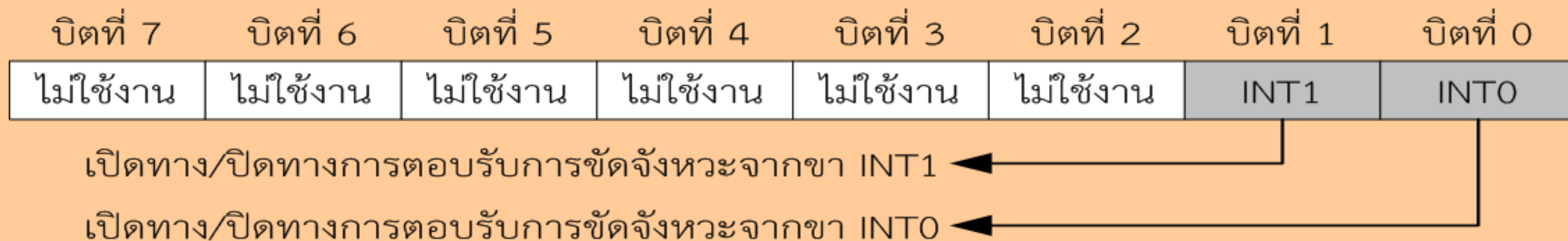


ค่าของบิตควบคุม		ความหมาย
ISCx1	ISCx0	
0	0	เมื่อขา INTx มีค่าเป็นตรรกะต่ำ จะถือว่าเป็นการขัดจังหวะเกิดขึ้น
0	1	เมื่อเกิดการเปลี่ยนแปลงใด ๆ ในทางตรรกะที่ขา INTx จะถือว่าเป็นการขัดจังหวะเกิดขึ้น
1	0	เมื่อเกิดขอบขาลงของสัญญาณ INTx ตัวประมวลผลจะถือว่าเป็นการขัดจังหวะ
1	1	เมื่อเกิดขอบขาขึ้นของสัญญาณ INTx ตัวประมวลผลจะถือว่าเป็นการขัดจังหวะ

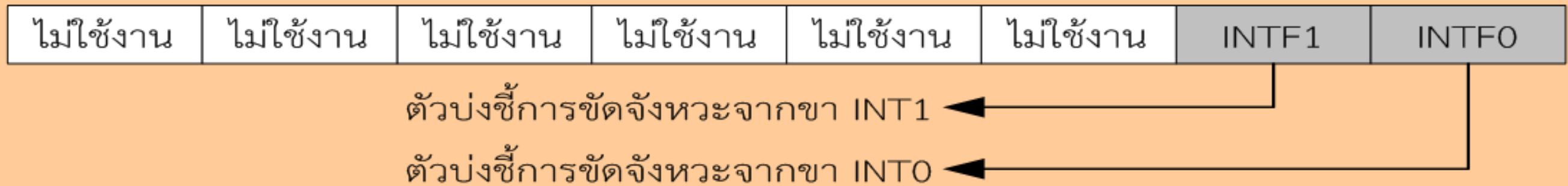




EIMSK – External Interrupt Mask Register

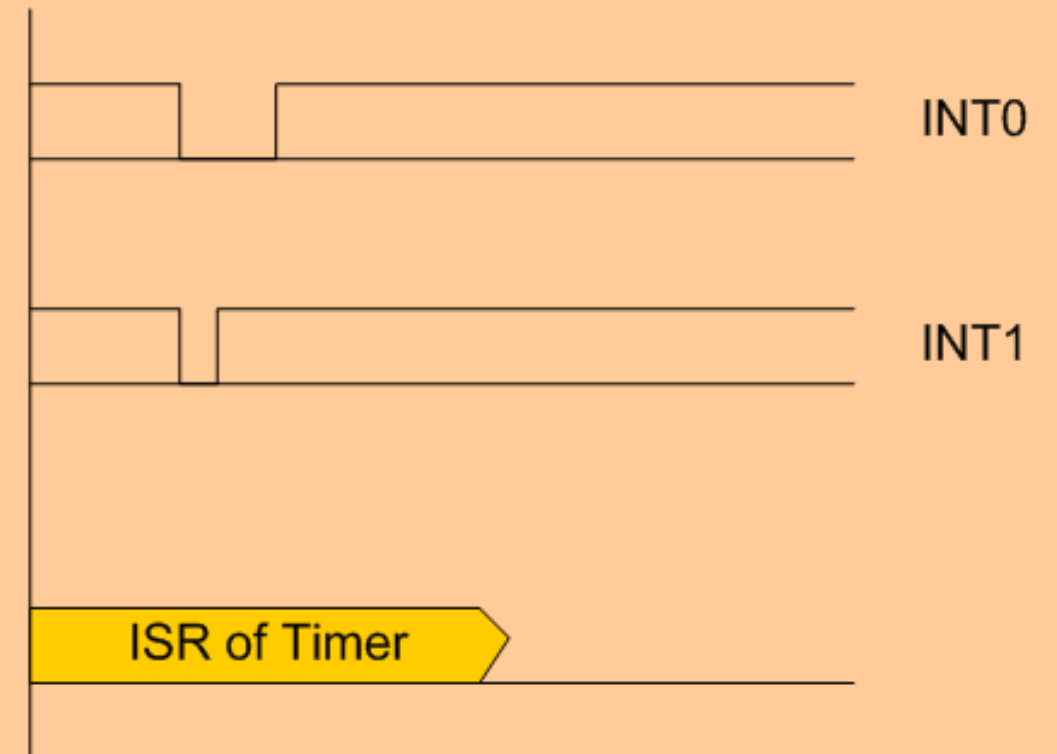
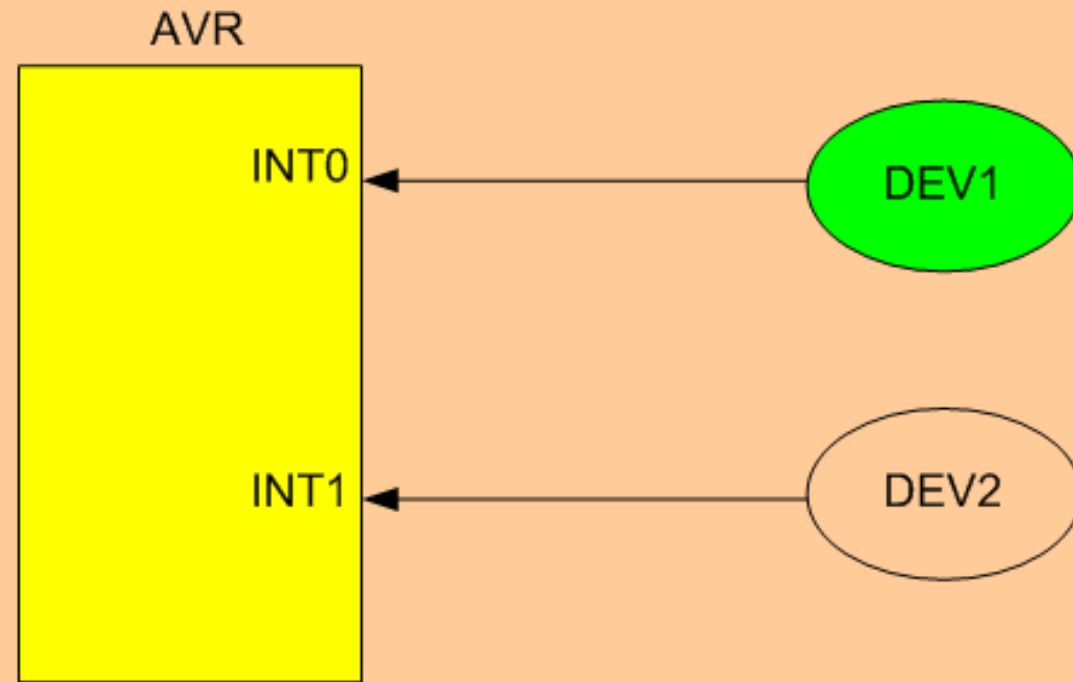


EIFR – External Interrupt Flag Register



- ◆ ใช้เก็บตัวบ่งชี้ว่ามีการอินเทอร์รัพต์จากขา INT0 หรือ INT1 หรือไม่
- ◆ จะไม่ทำงานกรณีที่ INT0 หรือ INT1 ถูกตั้งให้แอสคตีฟที่ตรรกะต่ำ

การขัดจังหวะใดได้รับการตอบสนอง ?



- ◆ สมมุติให้ INT0 เป็นชนิด low level interrupt
- ◆ และ INT1 เป็นชนิด rising edge interrupt
- ◆ สมมุติให้ ISR ของ Timer เสร็จสิ้นที่เวลา ดังรูปขวา





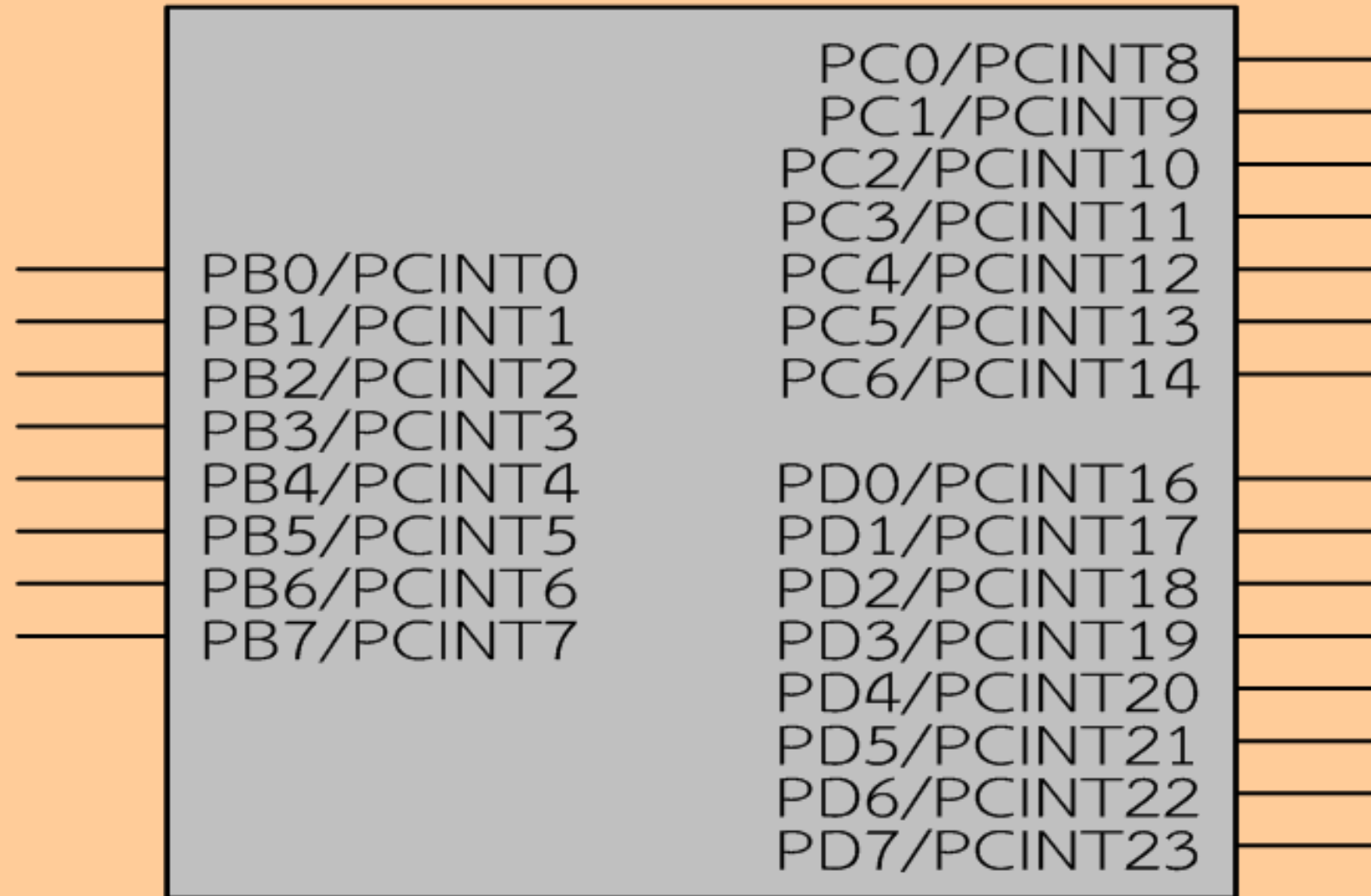
PIN Change interrupt

- ◆ แบ่งการตรวจสอบออกเป็น 3 กลุ่ม
 - ◆ การเปลี่ยนแปลงที่ขาใด ๆ ของ PCINT[7:0] → PCI0
 - ◆ การเปลี่ยนแปลงที่ขาใด ๆ ของ PCINT[14:8] → PCI1
 - ◆ การเปลี่ยนแปลงที่ขาใด ๆ ของ PCINT[23:16] → PCI2
- ◆ จะมีตัวบ่งชี้ PCIF0-PCIF2 เอาไว้เก็บสถานะการเปลี่ยนแปลงของขาสัญญาณ PCINT



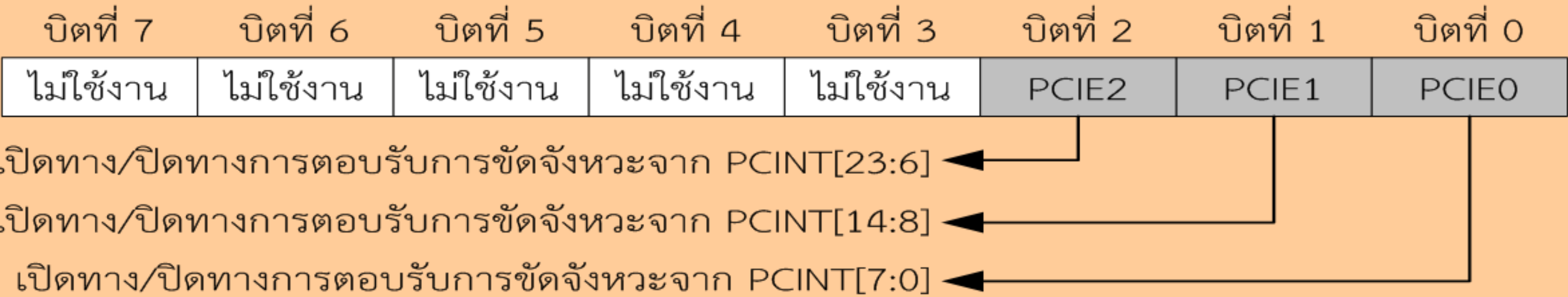
การับสัญญาณขัดจังหวะแบบพินเซนจ์

ATmega328P



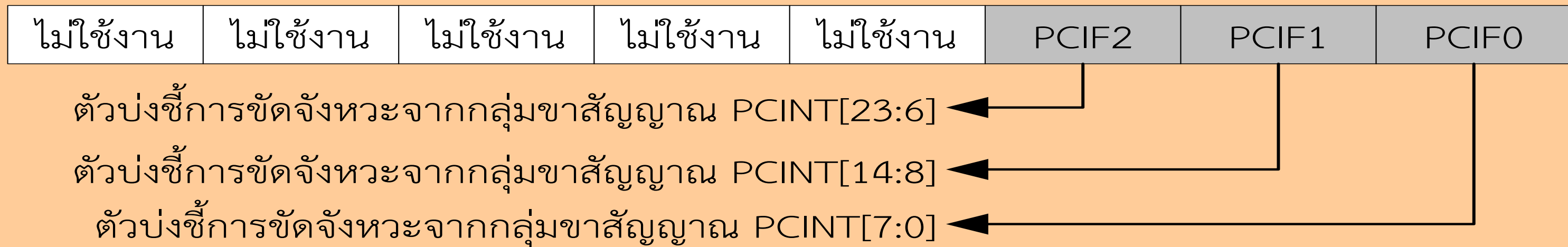


PCICR – Pin Change Interrupt Control Register





PCIFR – Pin Change Interrupt Flag Register





PCMSK

PCINT7	PCINT6	PCINT5	PCINT4	PCINT3	PCINT2	PCINT1	PCINT0	เรจิสเตอร์ PCMSK0
ไม่ใช้งาน	PCINT15	PCINT13	PCINT12	PCINT11	PCINT10	PCINT9	PCINT8	เรจิสเตอร์ PCMSK1
PCINT23	PCINT22	PCINT21	PCINT20	PCINT19	PCINT18	PCINT17	PCINT16	เรจิสเตอร์ PCMSK2



การเขียนโปรแกรมบริการการขัดจังหวะด้วย C



```
#include <avr/io.h>
#include <avr/interrupt.h>
int main(void)
{
    // กำหนดค่าเริ่มต้นเพื่อให้ชิพตอบสนองต่อการอินเทอร์รัพต์นั้นๆ
    ...
    while(1)
    {
        // ส่วนของโค้ดโปรแกรมหลัก
        ...
    }
}
```

```
ISR (interrupt_vector)
{
    // โค้ดที่ต้องการให้ชิพทำงานเมื่อเกิดอินเทอร์รัพต์
    ....
}
```



ค่าเวกเตอร์การขัดจังหวะในภาษาซี

```
#define INT0_vect      _VECTOR(1)    /* External Interrupt Request 0 */
#define INT1_vect      _VECTOR(2)    /* External Interrupt Request 1 */
#define PCINT0_vect    _VECTOR(3)    /* Pin Change Interrupt Request 0 */
#define PCINT1_vect    _VECTOR(4)    /* Pin Change Interrupt Request 1 */
#define PCINT2_vect    _VECTOR(5)    /* Pin Change Interrupt Request 2 */
#define WDT_vect       _VECTOR(6)    /* Watchdog Time-out Interrupt */
#define TIMER2_COMPA_vect _VECTOR(7) /* Timer/Counter2 Compare Match A */
#define TIMER2_COMPB_vect _VECTOR(8) /* Timer/Counter2 Compare Match A */
#define TIMER2_OVF_vect _VECTOR(9)   /* Timer/Counter2 Overflow */
#define TIMER1_CAPT_vect _VECTOR(10) /* Timer/Counter1 Capture Event */
#define TIMER1_COMPA_vect _VECTOR(11) /* Timer/Counter1 Compare Match A */
#define TIMER1_COMPB_vect _VECTOR(12) /* Timer/Counter1 Compare Match B */
#define TIMER1_OVF_vect  _VECTOR(13) /* Timer/Counter1 Overflow */
#define TIMER0_COMPA_vect _VECTOR(14) /* TimerCounter0 Compare Match A */
#define TIMER0_COMPB_vect _VECTOR(15) /* TimerCounter0 Compare Match B */
#define TIMER0_OVF_vect  _VECTOR(16) /* Timer/Couner0 Overflow */
#define SPI_STC_vect     _VECTOR(17) /* SPI Serial Transfer Complete */
#define USART_RX_vect    _VECTOR(18) /* USART Rx Complete */
#define USART_UDRE_vect  _VECTOR(19) /* USART, Data Register Empty */
#define USART_TX_vect    _VECTOR(20) /* USART Tx Complete */
#define ADC_vect         _VECTOR(21) /* ADC Conversion Complete */
#define EE_READY_vect    _VECTOR(22) /* EEPROM Ready */
#define ANALOG_COMP_vect _VECTOR(23) /* Analog Comparator */
#define TWI_vect         _VECTOR(24) /* Two-wire Serial Interface */
#define SPM_READY_vect   _VECTOR(25) /* Store Program Memory Read */
```



ตัวอย่างที่ 3.1

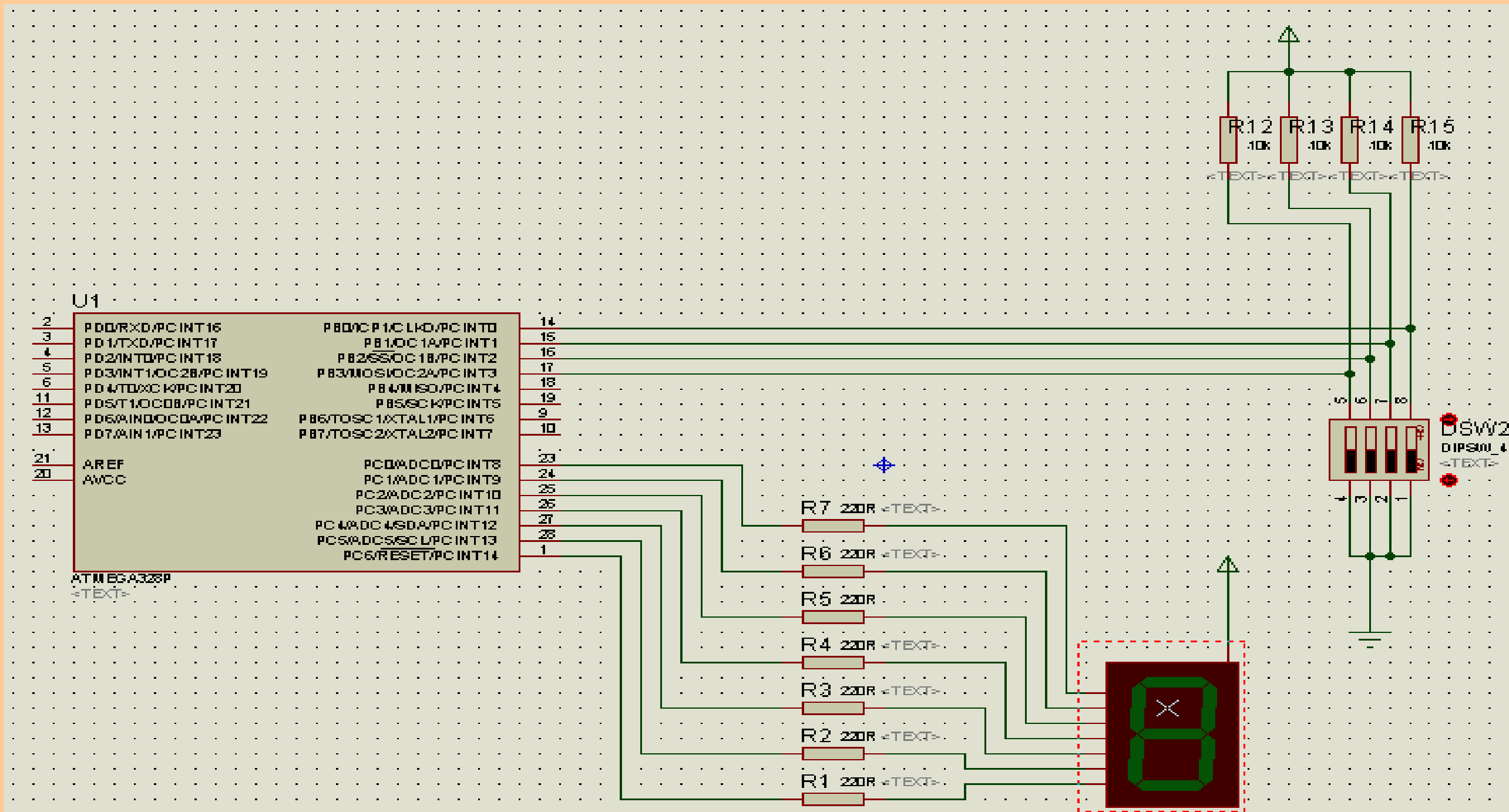
ตัว

- ◆ จงออกแบบวงจรอ่านคิปลวิตซ์ขนาด 4 บิตและแสดงผลเลขฐานสองที่ตั้งบนตัวสวิตซ์ออกแสดงผลที่แอลอีดีชนิด 7 ส่วน
- ◆ สวิตซ์จำนวน 4 บิต ต่อที่ พอร์ต B ที่ขา PB0-PB3
- ◆ แอลอีดีชนิด 7 ส่วน ต่อที่พอร์ต C
- ◆ ให้เขียนโปรแกรมควบคุมด้วยวิธีการขัดจังหวะโดยซีพียูไม่ต้องคอยวนซ้ำตรวจสอบการเลื่อนสวิตซ์





ตัวอย่างที่ 3.1



ตัวอย่างที่ 3.1

```
#include <avr/io.h>
#include <avr/interrupt.h>

unsigned char TB7SEG[] = {0b00111111,
                          0b000000110,
                          0b01011011,
                          0b01001111,
                          0b01100110,
                          0b01101101,
                          0b01111101,
                          0b000000111,
                          0b01111111,
                          0b01101111,
                          0b01110111,
                          0b01111100,
                          0b00111001,
                          0b01011110,
                          0b01111001,
                          0b01110001 };

void initial_read_sw_and_display(void)
{
    unsigned char a;
    DDRB = 0xF0;    //lower 4 bits are connected to switch
    a = PINB;
    a &= 0x0F;      //bit 7:4 are cleared
    PORTC = ~ (TB7SEG[a]);
}
```





ตัวอย่างที่ 3.1

```
int main(void)
{
    sei();           //enable global interrupt
    PCICR = 0x01;    //enable pinchange interrupt control register bit 0
    PCMSK0= 0x0F;    //enable PCINT0-PCINT3

    initial_read_sw_and_display();

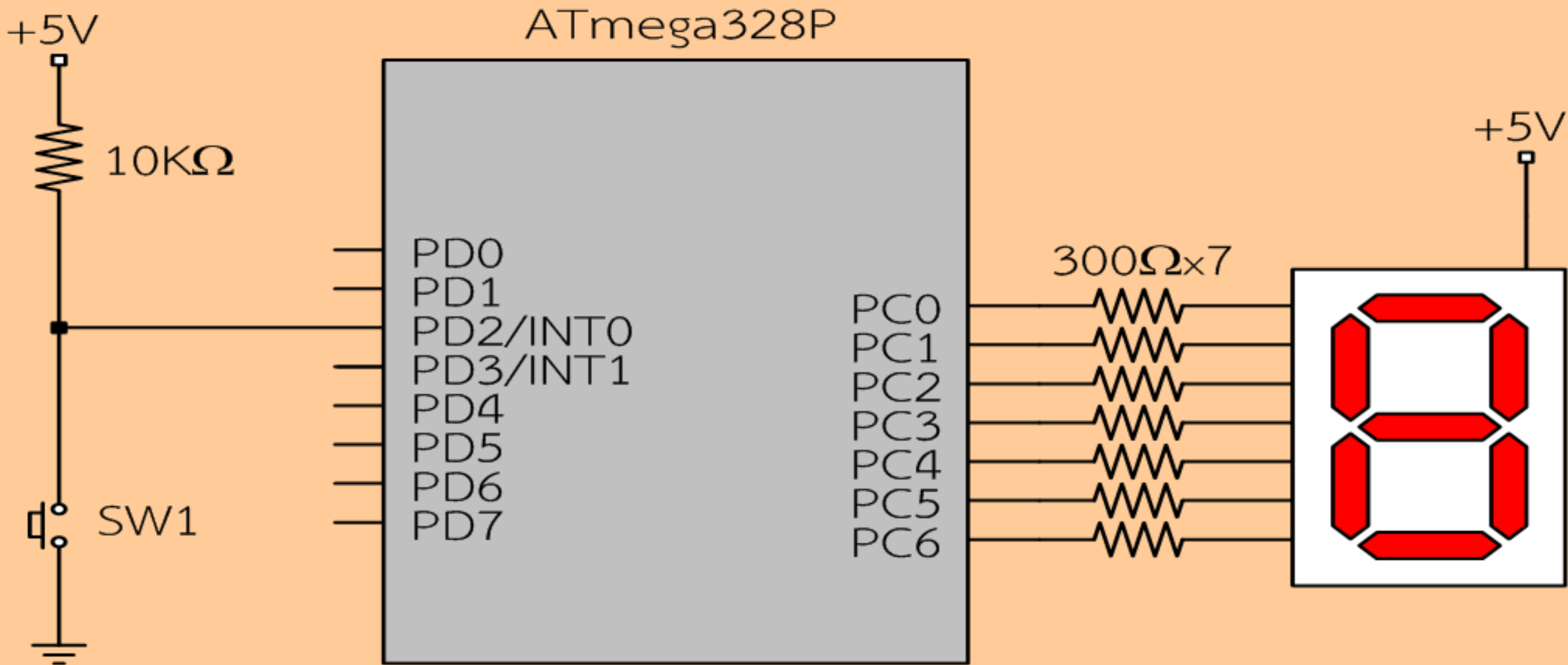
    while(1)        //---run forever
    {
        DDRC = 0xFF;
    }
}

ISR(PCINT0_vect)
{
    unsigned char a;
    DDRB = 0xF0;
    a = PINB;
    a &= 0x0F;       //bit 7:4 are cleared
    PORTC = ~ (TB7SEG[a]);
}
```





ตัวอย่างที่ 3.2





ตัวอย่างที่ 3.2

1	.INCLUDE "m328pdef.inc"	;เรียกใช้คลังโปรแกรมชื่อ m328pdef.inc
2	.DEF TEMP = R16	;กำหนดชื่อสัญลักษณ์ตัวแปร TEMP ให้ R16
3	.DEF COUNT = R17	;กำหนดชื่อสัญลักษณ์ตัวแปร COUNT ให้ R17
4	.DEF ZERO = R18	;กำหนดชื่อสัญลักษณ์ตัวแปร ZERO ให้ R18
5	.ORG 0x0000	;เริ่มต้นโปรแกรมที่ตำแหน่ง 0x0000
6	jmp MAIN	;กระโดดไปยังป้าย MAIN
7	.ORG 0x0002	;---รูทีนบริการการขัดจังหวะของ INT0 ใน ATmega328P ต้องเริ่มที่ตำแหน่ง 0x02 เสมอ
8	jmp ISR_INT0	;กระโดดไปยังรูทีนบริการการขัดจังหวะ INT0
9	MAIN:	;ป้ายบอกส่วนหลักของโปรแกรม
10	clr COUNT	;ลบล้างให้ตัวแปร COUNT มีค่าเป็นศูนย์
11	clr ZERO	;ลบล้างให้ตัวแปร ZERO มีค่าเป็นศูนย์
12	;---ส่งค่า 0x01 ให้ EIMSK เพื่ออนุญาตให้ตัวประมวลผลตอบรับการขัดจังหวะจาก INT0	
13	ldi TEMP, 0x01	;อนุญาตให้เฉพาะ INT0 ที่ส่งสัญญาณขัดจังหวะได้
14	out EIMSK, TEMP	;ส่งค่า 0x01 ให้ EIMSK





ตัวอย่างที่ 3.2

15		;---ส่งค่า 0x02 ให้เรจิสเตอร์ EICRA เพื่อให้ตัวประมวลผลตอบรับ INTO ที่ขอบขาลง	
16	ldi	ZL, low (EICRA)	;ส่งค่าตำแหน่งไบต์ต่ำของ EICRA ให้ ZL
17	ldi	ZH, high (EICRA)	;ส่งค่าตำแหน่งไบต์สูงของ EICRA ให้ ZH
18	ldi	TEMP, 0x02	;ตั้งให้ตัวประมวลผลตอบรับ INTO ที่ขอบขาลง
19	st	Z, TEMP	;ส่งค่า 0x02 ให้ EICRA
20	ldi	TEMP, 0xFF	;ตั้งทิศทางของพอร์ต C ให้เป็นพอร์ตเอาต์พุต
21	out	DDRC, TEMP	;ส่งค่า 0xFF ให้เรจิสเตอร์ DDRC
22	ldi	TEMP, 0b11111011	;ตั้งทิศทางของพอร์ต B ให้เป็นอินพุตเฉพาะบิต 2
23	out	DDRD, TEMP	;ส่งค่า 0b11111011 ให้เรจิสเตอร์ DDRD
24	sei		;เปิดทางการตอบรับการขัดจังหวะส่วนกลาง
25	ldi	TEMP, 0b11000000	;ค่าเริ่มต้นที่แอลอีดีให้ติดเลขศูนย์
26	out	PORTC, TEMP	;เริ่มต้นให้แสดงผลค่าศูนย์ออกที่แอลอีดี
27	LOOP:	rjmp	LOOP ;วนซ้ำการทำงานที่เดิม
28			





ตัวอย่างที่ 3.2

```
29      ;---ส่วนนี้ของโปรแกรมเป็นการเก็บตารางค้นหาของค่ารหัสการติดดับของแอลอีดี
30      ;   ชนิด 7-Segment ของรหัสปีซีดี ค่า 0-15
31  TB_7SEG:      .DB      0b00111111, 0b00000110      ;0 และ 1      ----a----
32              .DB      0b01011011, 0b01001111      ;2 และ 3      f      b
33              .DB      0b01100110, 0b01101101      ;4 และ 5      ----g----
34              .DB      0b01111101, 0b00000111      ;6 และ 7      e      c
35              .DB      0b01111111, 0b01101111      ;8 และ 9      ----d----
36              .DB      0b01110111, 0b011111100      ;10 และ 11 (แสดง A และ B)
37              .DB      0b00111001, 0b010111110      ;12 และ 13 (แสดง C และ D)
38              .DB      0b01111001, 0b011110001      ;14 และ 15 (แสดง E และ F)
39  ISR_INT0:      rcall    DELAY10MS      ;ช่วงเวลา 10 มิลลิวินาทีด้วยการเรียกใช้ซับรูทีน
40              in        TEMP, PIND      ;อ่านค่าจากขาสัญญาณของพอร์ต D
41              andi      TEMP, 0x04      ;กรองค่าบิตอื่นทิ้งไปให้เหลือเฉพาะบิตที่ 2
42              lsr        TEMP      ;เลื่อนบิตที่สองจำนวน 2 ครั้งไปทางขวาเพื่อให้บิต
43              lsr        TEMP      ;-ดังกล่าวมาตรงกับตำแหน่งบิตที่ 0 พอดี
```





ตัวอย่างที่ 3.2

44		cpi	TEMP, 0x01	;ค่าที่อ่านจากสวิตช์มีค่าเท่ากับตรรกะสูงหรือไม่
45		breq	CLR_FLAG	;หากเป็นตรรกะสูงให้ถือว่าไม่มีการกดสวิตช์ให้กระโดด
46				;ห้ามการเพิ่มค่า COUNT ไปยังป้าย CLR_FLAG
47		inc	COUNT	;เพิ่มค่า COUNT ขึ้น 1 ค่าเมื่อผ่านไป 10 มิลลิวินาที
48		cpi	COUNT, 16	;เปรียบเทียบค่า COUNT กับค่า 16
49		brlo	NEXT	;หาก COUNT ยังน้อยกว่า 16 ให้ข้ามไปยังป้าย NEXT
50		clr	COUNT	;หาก $COUNT \geq 16$ ให้ลบล้างค่า COUNT เป็นศูนย์
51	NEXT:	ldi	ZL, low(TB_7SEG*2)	;บรรจุค่าตำแหน่งไบต์ต่ำของ TB_7SEG ใส่ ZL
52		ldi	ZH, high(TB_7SEG*2)	;บรรจุค่าตำแหน่งไบต์สูงของ TB_7SEG ใส่ ZH
53		add	ZL, COUNT	;บวกค่า ZL ด้วยค่ารหัสปีซีดี อินพุต
54		adc	ZH, ZERO	;บวกค่าที่อาจมีการทดใน Carry ใส่ใน ZH
55		lpm		;อ่านหน่วยความจำโปรแกรมที่ Z ซี่งอยู่ใน R0
56		com	R0	;กลับค่าเป็นตรงข้าม เพราะใช้ LED แบบแอโนดร่วม
57		out	PORTC, R0	;แสดงผลค่าผลลัพธ์ออกสู่พอร์ต C
58	CLR_FLAG:	ldi	TEMP, 1<<INTF0	;เตรียมค่าสำหรับการลบล้างค่าในตัวบ่งชี้ INTF0
59		out	EIFR, TEMP	;ลบล้างค่าในตัวบ่งชี้ INTF0 ของเรจิสเตอร์ EIFR
60		reti		;สิ้นสุดรoutinesบริการการขัดจังหวะ



ตัวอย่างที่ 3.2

61	DELAY10MS:		;---จับรูทีนสำหรับหน่วงเวลา 10 มิลลิวินาที
62		push R16	;ทำงานครั้งเดียว ใช้สัญญาณนาฬิกา 2 รอบ
63		push R17	;ทำงานครั้งเดียว ใช้สัญญาณนาฬิกา 2 รอบ
64		ldi R16, 0X00	;ทำงานครั้งเดียว ใช้สัญญาณนาฬิกา 1 รอบ
65	LOOP2:	inc R16	;ทำงาน 78 ครั้ง แต่ละครั้งใช้สัญญาณนาฬิกา 1 รอบ
66		ldi R17, 0X00	;ใช้สัญญาณนาฬิกา 78 รอบ
67	LOOP1:	inc R17	;ใช้สัญญาณนาฬิกา 19,890 รอบ
68		cpi R17, 0XFF	;ใช้สัญญาณนาฬิกา 19,890 รอบ
69		brlo LOOP1	;ใช้สัญญาณนาฬิกา 39,702 รอบ
70		nop	;ใช้สัญญาณนาฬิกา 78 รอบ
71		cpi R16, 155	;ใช้สัญญาณนาฬิกา 78 รอบ
72		brlo LOOP2	;ใช้สัญญาณนาฬิกา 155 รอบ
73		pop R17	;ทำงานครั้งเดียว ใช้สัญญาณนาฬิกา 2 รอบ
74		pop R16	;ทำงานครั้งเดียว ใช้สัญญาณนาฬิกา 2 รอบ
75		ret	;ใช้สัญญาณนาฬิกา 4 รอบ
76			;---รวมใช้สัญญาณนาฬิกา 79,962 รอบ



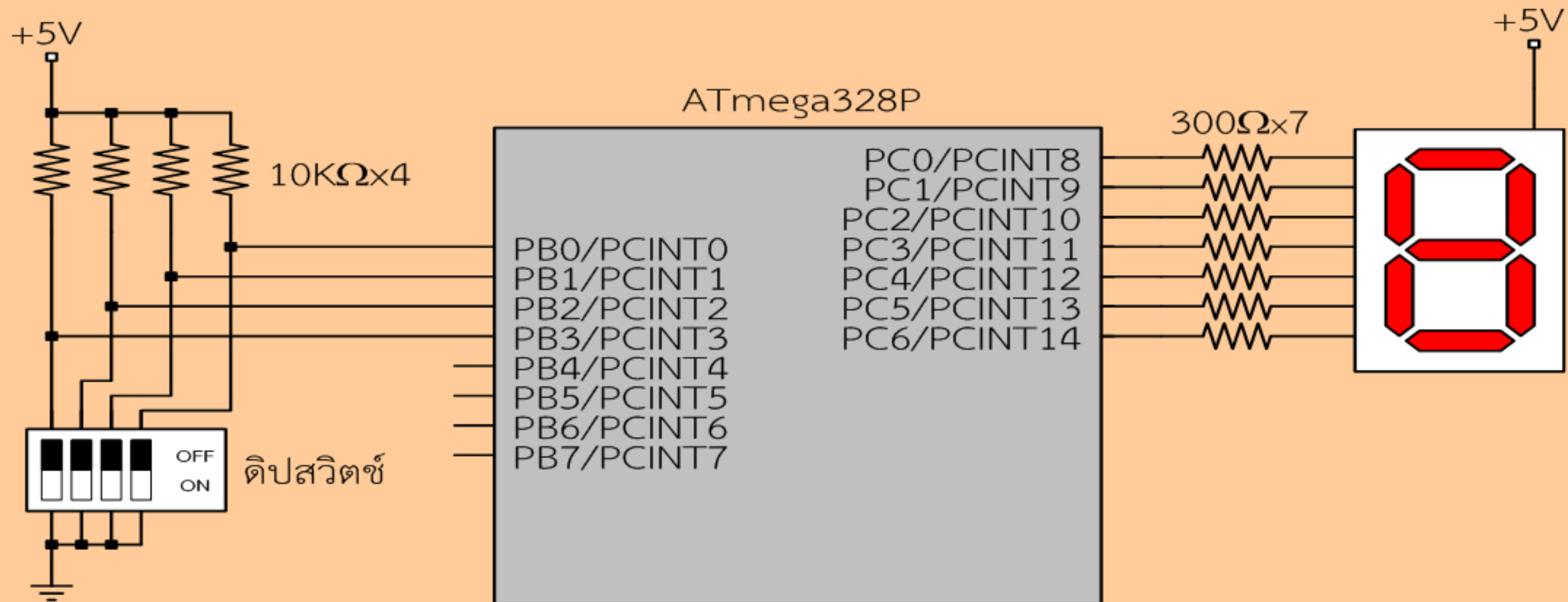
ฝึกทำโจทย์

- ◆ จงเขียนโปรแกรมในตัวอย่างที่ 3.2 ใหม่ โดยใช้ภาษาซี
- ◆ คำแนะนำ ให้ดูภาษาแอสเซมบลีของตัวอย่างที่ 3.2 ประกอบ





ตัวอย่างที่ 3.3





ตัวอย่างที่ 3.3

```
1  .INCLUDE "m328pdef.inc"
2  .DEF TEMP = R16
3  .DEF ZERO = R17
4  .ORG 0x0000
5          rjmp    MAIN          ;กระโดดไปยังป้าย MAIN
6  .ORG 0x0006          ;---รูทีนบริการการขัดจังหวะของ PCINT0 จะต้องเริ่มที่ตำแหน่ง 0x0006 เสมอ
7          rjmp    ISR_PCINT0    ;กระโดดไปยังรูทีนบริการการขัดจังหวะ PCINT0
8  MAIN:                ;ป้ายบอกส่วนหลักของโปรแกรม
9          clr     ZERO          ;ลบล้างค่าในตัวแปร ZERO ให้มีค่าเป็นศูนย์
10         ;---ส่งค่า 0x01 ให้ PCICR เพื่ออนุญาตให้ตัวประมวลผลตอบรับการขัดจังหวะจาก PCINT0
11         ldi     ZL, low (PCICR) ;ส่งค่าตำแหน่งไบต์ต่ำของ PCICR ให้ ZL
12         ldi     ZH, high (PCICR) ;ส่งค่าตำแหน่งไบต์สูงของ PCICR ให้ ZH
13         ldi     TEMP, 0x01     ;อนุญาตให้เฉพาะ PCINT0 ส่งสัญญาณขัดจังหวะได้
14         st      Z, TEMP        ;ส่งค่า 0x01 ให้ PCICR
```





ตัวอย่างที่ 3.3

15		;---ส่งค่า 0x0F ให้เรจิสเตอร์ PCMSK0 เพื่อให้ตอบรับการขัดจังหวะจาก	
16	ldi	ZL, low (PCMSK0)	;ส่งค่าตำแหน่งไบต์ต่ำของ PCMSK0 ให้ ZL
17	ldi	ZH, high (PCMSK0)	;ส่งค่าตำแหน่งไบต์สูงของ PCMSK0 ให้ ZH
18	ldi	TEMP, 0x0F	;อนุญาตให้เฉพาะขา PCINT0-3 ที่สามารถขัดจังหวะได้
19	st	Z, TEMP	;ส่งค่า 0x0F ให้ PCMSK0
20	ldi	TEMP, 0xFF	;ตั้งทิศทางของพอร์ต C ให้เป็นพอร์ตเอาต์พุต
21	out	DDRC, TEMP	;ส่งค่า 0xFF ให้เรจิสเตอร์ DDRC
22	ldi	TEMP, 0xF0	;ตั้งทิศทางของพอร์ต B ให้เป็นอินพุตเฉพาะบิต 0-3
23	out	DDRB, TEMP	;ส่งค่า 0xF0 ให้เรจิสเตอร์ DDRB
24	sei		;เปิดทางการตอบรับการขัดจังหวะส่วนกลางของเอวีอาร์
25	rcall	READ_DISPLAY	;เรียกใช้ซับรูทีน READ_DISPLAY
26	LOOP:	rjmp	LOOP ;วนซ้ำการทำงานที่เดิม





ตัวอย่างที่ 3.3

27	;---ส่วนนี้ของโปรแกรมเป็นการเก็บตารางค้นหาของค่ารหัสการติดดับของ LED			
28	; ชนิด 7-Segment ของรหัสปืซีดี ค่า 0-15			
29	TB_7SEG:	.DB	0b00111111, 0b00000110	;0 และ 1 ----a----
30		.DB	0b01011011, 0b01001111	;2 และ 3 f b
31		.DB	0b01100110, 0b01101101	;4 และ 5 ----g----
32		.DB	0b01111101, 0b00000111	;6 และ 7 e c
33		.DB	0b01111111, 0b01101111	;8 และ 9 ----d----
34		.DB	0b01110111, 0b01111100	;10 และ 11 (แสดง A และ B)
35		.DB	0b00111001, 0b01011110	;12 และ 13 (แสดง C และ D)
36		.DB	0b01111001, 0b01110001	;14 และ 15 (แสดง E และ F)
37	ISR_PCINT0:	rcall	READ_DISPLAY	;เรียกใช้ซับรูทีน READ_DISPLAY
38		reti		;ออกจากรูทีนบริการการขัดจังหวะ





ตัวอย่างที่ 3.3

39	READ_DISPLAY:	;	---ซักรูทึนสำหรัทำการอ่านค่าจากพอร์ต B และแสดงผลออกทางพอร์ต C
40	in	TEMP, PINB	;อ่านค่าจากพอร์ต B มาใส่ในตัวแปร TEMP
41	andi	TEMP, 0x0F	;พราง 4 บิตบนของพอร์ต B ที่มีได้ใช้งานทิ้งไป
42	LOOK_TABLE:	ldi	ZL, low(TB_7SEG*2) ;บรรจุค่าตำแหน่งไบต์ต่ำของ TB_7SEG ใส่ ZL
43		ldi	ZH, high(TB_7SEG*2) ;บรรจุค่าตำแหน่งไบต์สูงของ TB_7SEG ใส่ ZH
44		add	ZL, TEMP ;บวกค่า ZL ด้วยค่ารหัสปีซีดี อินพุต
45		adc	ZH, ZERO ;บวกค่าที่อาจมีการทดใน Carry ใส่ใน ZH
46		lpm	;อ่านหน่วยความจำโปรแกรมที่ Z ซึ่อยู่ใส่ใน R0
47		com	R0 ;กลับค่าเป็นตรงข้าม เพราะใช้ LED แบบแอโนดร่วม
48		out	PORTC, R0 ;แสดงผลค่าผลลัพธ์ออกสู่พอร์ต C
49		ret	;กลับสู่โปรแกรมหรือฟังก์ชันผู้เรียก

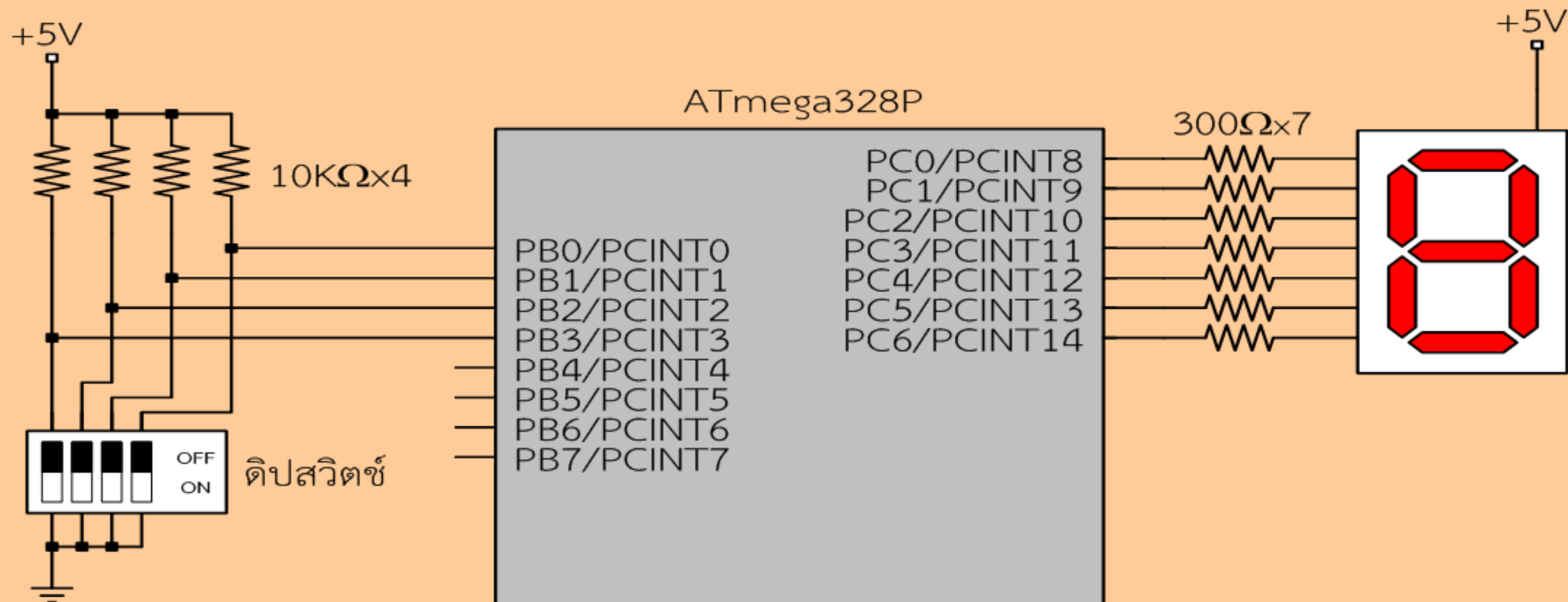
ฝึกทำโจทย์

- ◆ เขียนโปรแกรมให้ทำงานเช่นเดียวกับตัวอย่างที่ 3.3 แต่ให้ใช้ภาษาซีแทนแอสเซมบลี





ตัวอย่างที่ 3.4





ตัวอย่างที่ 3.4

```
1  #include <avr/io.h>                //เรียกใช้คลังโปรแกรมชื่อ io.h
2  #include <avr/interrupt.h>          //เรียกใช้คลังโปรแกรมชื่อ interrupt.h
3
4  //-----ตารางค้นหาสำหรับการแปลงค่ารหัสเลขฐานสองเป็นรหัสแสดงผลบนแอลอีดีชนิด 7 ส่วน
5  unsigned char TB7SEG[] = { 0b00111111, 0b00000110, 0b01011011, 0b01001111,
6                             0b01100110, 0b01101101, 0b01111101, 0b00000111,
7                             0b01111111, 0b01101111, 0b01110111, 0b01111100,
8                             0b00111001, 0b01011110, 0b01111001, 0b01110001 };
9  //-----ทำการอ่านค่าจากสวิตช์และแสดงผลที่แอลอีดีชนิด 7 ส่วน
10 void initial_read_sw_and_display(void)
11 {
12     unsigned char a;                //ประกาศตัวแปรเฉพาะที่
13     a = PINB;                       //อ่านค่าสถานะของสวิตช์จากพอร์ต B
14     a &= 0x0F;                      //พราง 4 บิตบนที่อ่านจากพอร์ต B ที่
15     PORTC = ~ (TB7SEG[a]);          //นำค่าเลขฐานสองที่ได้ไปเปิดตารางค้นหาเพื่อแปลงเป็นรหัส 7
16 }                                   // segment ที่ได้แสดงผลที่พอร์ต C
```



ตัวอย่างที่ 3.4

```
17 //-----ฟังก์ชัน main
18 int main(void)
19 {
20     DDRB = 0xF0;           //ตั้งค่าทิศทางให้พอร์ต B บิต 0-3 เป็นอินพุต
21     DDRC = 0xFF;           //ตั้งค่าทิศทางให้พอร์ต C ทุกบิตเป็นเอาต์พุต
22     PCICR = 0x01;          //เปิดทางการขัดจังหวะแบบพินเซนจ์จาก PCINT0
23     PCMSK0= 0x0F;          //อนุญาตให้เฉพาะ PCINT[3:0] เท่านั้นที่สามารถขัดจังหวะได้
24     initial_read_sw_and_display(); //อ่านค่าจากสวิตช์และแสดงผลที่ LED ในครั้งแรกที่ทำงาน
25     sei();                 //เปิดทางการขัดจังหวะส่วนกลาง
26     while(1)              //วนซ้ำแบบไม่รู้จบ
27     {                      //ไม่มีงานอะไรที่จะต้องทำในวงวนนี้
28     }
29 }
```





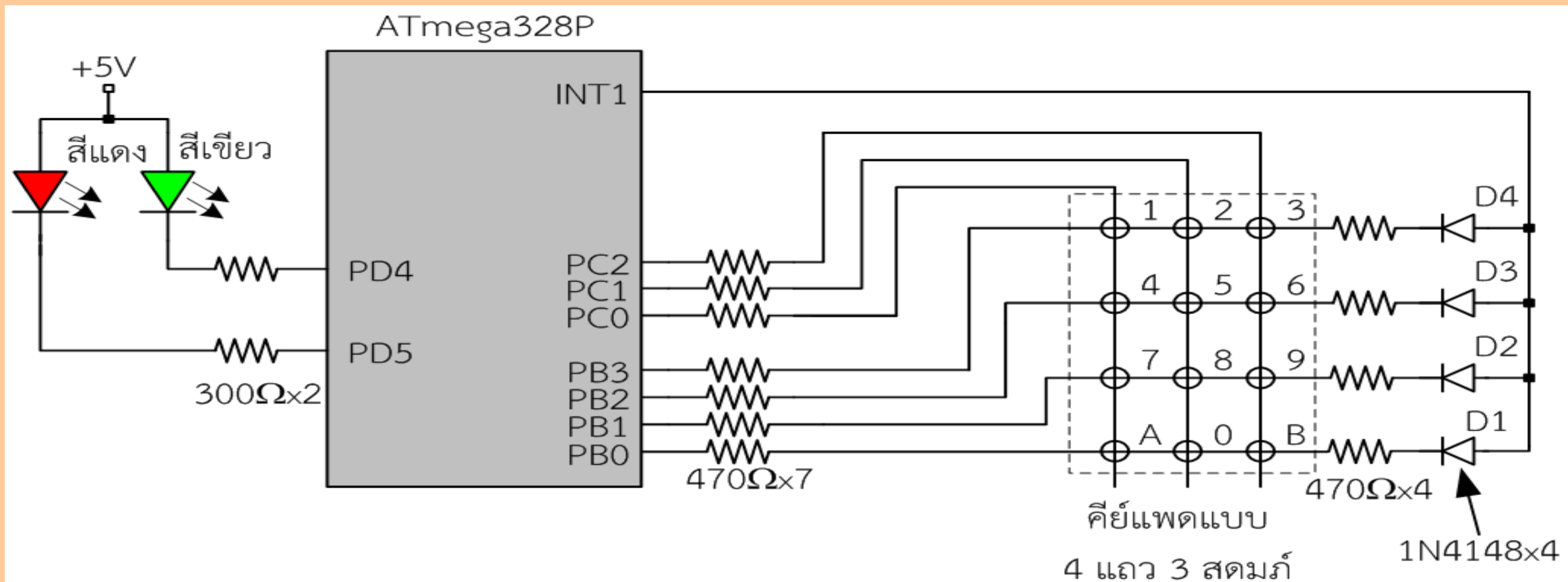
ตัวอย่างที่ 3.4

```
30 //-----ฟังก์ชันบริการการขัดจังหวะของ PCINT0
31 ISR(PCINT0_vect)                                //ระบุว่าฟังก์ชันบริการการขัดจังหวะนี้เป็นของ PCINT0
32 {
33     unsigned char a;                             //ประกาศตัวแปรเฉพาะที่
34     a = PINB;                                     //อ่านค่าจากพอร์ต B ซึ่งต่ออยู่กับสวิตช์ในตัวแปร a
35     a &= 0x0F;                                    //พราง 4 บิตบนที่อ่านได้จากพอร์ต B ทิ้งไป
36     PORTC = ~ (TB7SEG[a]);                       //
37 }
```





ตัวอย่างที่ 3.5





ตัวอย่างที่ 3.5

```
1  #include <avr/io.h>           //ใช้งานคลังโปรแกรมชื่อ io.h
2  #include <avr/interrupt.h>    //ใช้งานคลังโปรแกรมชื่อ interrupt.h
3  #define F_CPU 16000000UL      //ระบุความเร็วของตัวประมวลผลเพื่อใช้เป็นฐานในการหน่วงเวลา
4  #include <avr/delay.h>        //เปิดคลังโปรแกรม delay.h เพื่อใช้งานฟังก์ชันหน่วงเวลา
5
6  signed char PressedKey;       //ตัวแปรส่วนกลางสำหรับระบุว่าการกดคีย์แล้วหรือไม่
7  signed char n;               //ตัวแปรส่วนกลางสำหรับระบุหมายเลขสดมภ์ที่กำลังกราดตรวจ
8
9  int main(void)               //ฟังก์ชันหลักของโปรแกรม
10 {
11     DDRC = 0b111;             //สั่งพอร์ต C บิต 0-2 ซึ่งต่อกับแต่ละสดมภ์ของคีย์แพดเป็นเอาต์พุต
12     DDRB = 0b0000;           //สั่งพอร์ต B บิต 0-3 ซึ่งต่อกับแต่ละแถวของคีย์แพดเป็นอินพุต
13     DDRD = 0b11110111;       //สั่งให้ขา 3 ของพอร์ต D ซึ่งทำหน้าที่รับสัญญาณ INT1 เป็นอินพุต
14     EICRA = 0b1000;          //ตั้งให้รับสัญญาณขัดจังหวะ INT1 ในแบบขอบขาลง
15     EIMSK = 0x02;            //เปิดทางการขัดจังหวะของขา INT1
16     sei();                   //เปิดทางการตอบรับการขัดจังหวะส่วนกลางของตัวประมวลผล
17
```

ตัวอย่างที่ 3.5

```
18  PORTB = 0x0F;           //ตั้งขึ้นสัญญาณที่ทุกแถวของคีย์แพดให้มีค่าตรรกะสูง
19  PORTC = 0b000;         //ตั้งลงสัญญาณที่ทุกสดมภ์ของคีย์แพดให้มีค่าตรรกะต่ำ
20  PORTD |= (1<<3);        //ตั้งขึ้นที่ขาสัญญาณ INT1 ให้มีค่าตรรกะสูง
21  Switch_GREEN_LED_off(); //ปิดหลอด LED สีเขียวเมื่อเริ่มทำงานในครั้งแรก
22  Switch_RED_LED_off();   //ปิดหลอด LED สีแดงเมื่อเริ่มทำงานในครั้งแรก
23  while (1)               //วนซ้ำการทำงานแบบไม่รู้จบ
24  { }                     //ไม่ต้องมีการทำงานใด ๆ ในส่วนนี้
25  }                       //สิ้นสุดขอบเขตของฟังก์ชันหลักของโปรแกรม
26
27  void Switch_RED_LED_on(void) //ฟังก์ชันสำหรับใช้สั่งให้หลอด LED สีแดงติดสว่าง
28  {
29      PORTD &= 0b11011111; //ส่งค่าตรรกะต่ำออกที่บิต 5 ของพอร์ต D บิตส่วนบิตอื่น ๆ คงค่าเดิม
30  }
31
32  void Switch_GREEN_LED_on(void) //ฟังก์ชันสำหรับใช้สั่งให้หลอด LED สีเขียวติดสว่าง
33  {
34      PORTD &= 0b11101111; //ส่งค่าตรรกะต่ำออกที่บิต 4 ของพอร์ต D บิตส่วนบิตอื่น ๆ คงค่าเดิม
35  }
36
```

ตัวอย่างที่ 3.5

```
37 void Switch_RED_LED_off(void) //ฟังก์ชันสำหรับใช้สั่งให้หลอด LED สีแดงไม่สว่าง
38 {
39     PORTD |= 0b00100000; //ส่งค่าตรรกะสูงออกที่บิต 5 ของพอร์ต D บิตส่วนบิตอื่น ๆ คงค่าเดิม
40 }
41
42 void Switch_GREEN_LED_off(void) //ฟังก์ชันสำหรับใช้สั่งให้หลอด LED สีเขียวไม่สว่าง
43 {
44     PORTD |= 0b00010000; //ส่งค่าตรรกะสูงออกที่บิต 4 ของพอร์ต D บิตส่วนบิตอื่น ๆ คงค่าเดิม
45 }
46
47 void Blink_RED_LED(void) //ฟังก์ชันสำหรับใช้สั่งให้หลอด LED สีแดงกะพริบติดสลับดับ
48 {
49     Switch_RED_LED_on(); //สั่งให้ LED สีแดงติดสว่าง
50     _delay_ms(500); //หน่วงเวลา 500 มิลลิวินาที
51     Switch_RED_LED_off(); //สั่งให้ LED สีแดงดับ
52     _delay_ms(500); //หน่วงเวลา 500 มิลลิวินาที
53 }
54
```

ตัวอย่างที่ 3.5

55	<code>void Blink_GREEN_LED(void)</code>	<code>//ฟังก์ชันสำหรับใช้สั่งให้หลอด LED สีเขียวกระพริบ</code>
56	<code>{</code>	
57	<code> Switch_GREEN_LED_on();</code>	<code>//สั่งให้ LED สีเขียวติดสว่าง</code>
58	<code> _delay_ms(500);</code>	<code>//หน่วงเวลา 500 มิลลิวินาที</code>
59	<code> Switch_GREEN_LED_off();</code>	<code>//สั่งให้ LED สีเขียวดับ</code>
60	<code> _delay_ms(500);</code>	<code>//หน่วงเวลา 500 มิลลิวินาที</code>
61	<code>}</code>	
62		
63	<code>ISR(INT1_vect)</code>	<code>//รูทีนบริการการขัดจังหวะจากขา INT1</code>
64	<code>{</code>	
65	<code> PressedKey = -1;</code>	<code>//เริ่มต้นให้ตัวแปร PressedKey = -1 เพื่อบอกว่ายังไม่มีคีย์ใดถูกกด</code>
66	<code> n = 1;</code>	<code>//เริ่มต้นให้ตัวแปร n = 1 เพื่อเริ่มการตรวจสอบสแตมภ์แรก</code>
67	<code> do</code>	<code>//วงวนชนิด do-while</code>
68	<code> {</code>	<code>//เริ่มต้นขอบเขตของวงวนชนิด do-while</code>
69	<code> if (n==1)</code>	<code>//หาก n = 1 ให้ทำการตรวจสอบที่สแตมภ์ 1</code>
70	<code> {</code>	
71	<code> PORTC = 0b110;</code>	<code>//ตรวจสอบที่สแตมภ์ 1 (ต่อกับปุ่ม “1”, “4”, “7”, “A”)</code>
72	<code> _delay_ms(10);</code>	<code>//หน่วงเวลา 10 มิลลิวินาที</code>
73	<code> switch(PINB & 0x0F)</code>	<code>//อ่านผลการตรวจสอบสแตมภ์ 1</code>

ตัวอย่างที่ 3.5

```
74      {
75          case 0b0111:          //หากแถวที่ต่อกับ PB3 เท่ากับ 0 แสดงว่าปุ่ม “1” ถูกกด
76              PressedKey= 1;break;
77          case 0b1011:          //หากแถวที่ต่อกับ PB2 เท่ากับ 0 แสดงว่าปุ่ม “4” ถูกกด
78              PressedKey= 4;break;
79          case 0b1101:          //หากแถวที่ต่อกับ PB1 เท่ากับ 0 แสดงว่าปุ่ม “7” ถูกกด
80              PressedKey= 7;break;
81          case 0b1110:          //หากแถวที่ต่อกับ PB0 เท่ากับ 0 แสดงว่าปุ่ม “A” ถูกกด
82              PressedKey= 10;break; //หากปุ่ม “A” ถูกกดให้ถือว่ามียาของปุ่มเท่ากับ 10
83      }
84  }
85  else if (n==2)                //หาก n = 2 ให้ทำการกราดตรวจที่สดมภ์ 2
86  {
87      PORTC = 0b101;            //กราดตรวจที่สดมภ์ 2 (ต่อกับปุ่ม “2”, “5”, “8”, “0”)
88      _delay_ms(10);            //หน่วงเวลา 10 มิลลิวินาที
89      switch(PINB & 0x0F)        //อ่านผลการกราดตรวจสดมภ์ 2
```



ตัวอย่างที่ 3.5

```
90      {
91          case 0b0111:          //หากแถวที่ต่อกับ PB3 เท่ากับ 0 แสดงว่าปุ่ม “2” ถูกกด
92              PressedKey= 2;break;
93          case 0b1011:          //หากแถวที่ต่อกับ PB2 เท่ากับ 0 แสดงว่าปุ่ม “5” ถูกกด
94              PressedKey= 5;break;
95          case 0b1101:          //หากแถวที่ต่อกับ PB1 เท่ากับ 0 แสดงว่าปุ่ม “8” ถูกกด
96              PressedKey= 8;break;
97          case 0b1110:          //หากแถวที่ต่อกับ PB0 เท่ากับ 0 แสดงว่าปุ่ม “0” ถูกกด
98              PressedKey= 0;break;
99      }
100  }
101  else if (n==3)                //หาก n = 3 ให้ทำการกราดตรวจที่สดมภ์ 3
102  {
103      PORTC = 0b011;            //กราดตรวจที่สดมภ์ 3 (สดมภ์ที่ต่อกับปุ่ม 3, 6, 9, “B”)
104      _delay_ms(10);            //หน่วงเวลา 10 มิลลิวินาที
105      switch(PINB & 0x0F)        //อ่านผลการกราดตรวจสดมภ์ 3
106      {
107          case 0b0111:          //หากแถวที่ต่อกับ PB3 เท่ากับ 0 แสดงว่าปุ่ม “3” ถูกกด
108              PressedKey= 3;break;
```




ตัวอย่างที่ 3.5

```
109         case 0b1011:           //หากแถวที่ต่อกับ PB2 เท่ากับ 0 แสดงว่าปุ่ม “6” ถูกกด
110             PressedKey= 6;break;
111         case 0b1101:           //หากแถวที่ต่อกับ PB1 เท่ากับ 0 แสดงว่าปุ่ม “9” ถูกกด
112             PressedKey= 9;break;
113         case 0b1110:           //หากแถวที่ต่อกับ PB0 เท่ากับ 0 แสดงว่าปุ่ม “B” ถูกกด
114             PressedKey= 11;break; //หากปุ่ม “B” ถูกกดให้ถือว่ามียี่ห้อของปุ่มเท่ากับ 11
115     }
116 }
117     n++;                       //เพิ่มค่าตัวแปร n ขึ้นหนึ่งค่า
118 } while (n<= 3);              //วนซ้ำในวงวน do-while นี้ตราบใดที่  $n \leq 3$ 
119
```





ตัวอย่างที่ 3.5

120	if (PressedKey==0)	//หากพบว่าปุ่มที่ถูกกด คือ ปุ่ม “0”
121	{	//ให้ทำการกระพริบหลอด LED สีแดง จำนวน 5 ครั้ง
122	for (int i=0;i<5;i++)	//ซึ่งทำได้โดยการวนซ้ำเรียกฟังก์ชัน Blink_RED_LED 5 ครั้ง
123	Blink_RED_LED();	
124	}	
125	else if (PressedKey != -1)	//หากพบว่าปุ่มที่ถูกกด แต่ปุ่มที่ถูกกดไม่ใช่ปุ่ม “0”
126	{	//ทำการกระพริบหลอด LED สีเขียวเป็นจำนวนครั้งเท่ากับ-
127	for(int i=0;i<PressedKey;i++)	//-ค่าที่เก็บในตัวแปร PressedKey
128	Blink_GREEN_LED();	//เรียกใช้ฟังก์ชันกระพริบแอลอีดีสีเขียว
129	}	
130	PORTC = 0b000;	//สั่งให้พอร์ต C บิต 0-3 เป็นตรรกะต่ำ เพื่อเตรียมพร้อมสำหรับ
131	}	//รับการขัดจังหวะในครั้งต่อไป
132		





การบ้าน บทที่ 3

- ♦ จงตัดแปลงวงจรในตัวอย่างที่ 3.5 ให้เปลี่ยนวิธีการกราด
ตรวจจากการส่งออกจากแวนอนและรับเข้า ที่แนวตั้งมา
เป็นส่งออกในแนวตั้งและรับเข้าในแวนอน จะต้องมีการ
ตัดแปลงวงจรอย่างไร จงเขียนโปรแกรมสำหรับวงจรที่
ตัดแปลงแล้วให้มีการทำงานแบบเดียวกับโปรแกรมใน
ตัวอย่างที่ 3.5



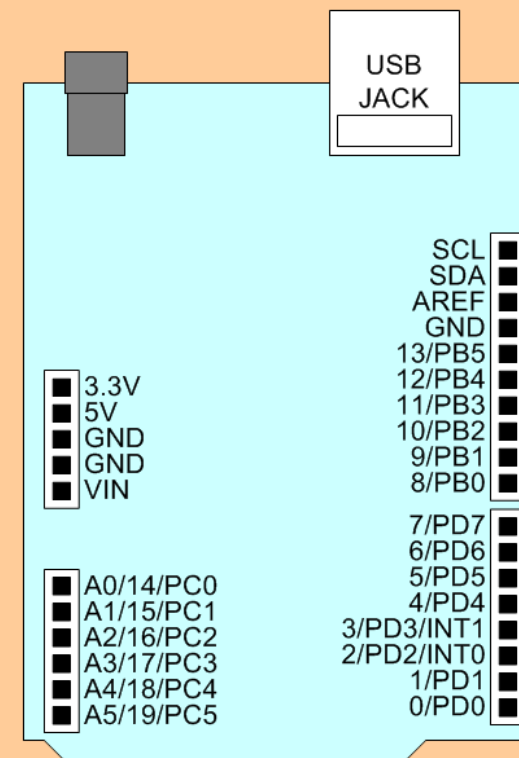


การตั้งค่า Interrupt ใน Arduino IDE

◆ ฟังก์ชันที่ใช้

```
attachInterrupt(digitalPinToInterrupt(pin), ISR, mode)
```

◆ ขา Digital Pin ที่ใช้ได้ คือ 2 และ 3 บนบอร์ด UNO





mode

- ◆ defines when the interrupt should be triggered.
- ◆ Four constants are predefined as valid values:
 - ◆ **LOW** to trigger the interrupt whenever the pin is low
 - ◆ **CHANGE** to trigger the interrupt whenever the pin changes value
 - ◆ **RISING** to trigger when the pin goes from low to high
 - ◆ **FALLING** for when the pin goes from high to low.





Example 3.6

```
const byte ledPin = 13;
const byte interruptPin = 2;
volatile byte state = LOW;

void setup() {
    pinMode(ledPin, OUTPUT);
    pinMode(interruptPin, INPUT_PULLUP);
    attachInterrupt(digitalPinToInterrupt(interruptPin), blink, CHANGE);
}

void loop() {
    digitalWrite(ledPin, state);
}

void blink() {
    state = !state;
}
```





ฝึกทำโจทย์

- ◆ ฝึกเขียนโปรแกรมในรูปแบบของตัวอย่างที่ 3.3 แต่ให้ใช้ภาษาซีในรูปแบบแพลตฟอร์ม Arduino IDE (ใช้วิธีการ `attachInterrupt`)



ฉบับที่ 3

