การอินเตอร์รัพท์ (Interrupt)

วัตถุประสงค์

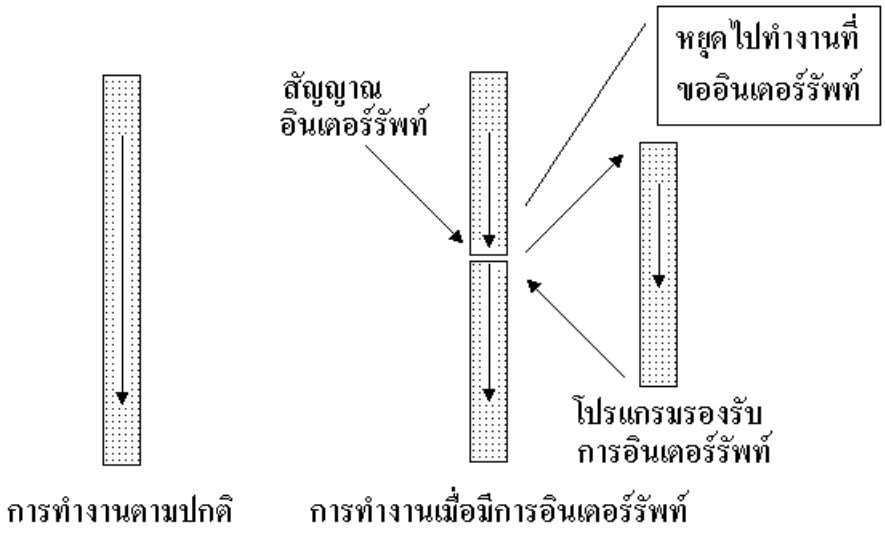
เพื่อให้มีความรู้เกี่ยวกับการอินเตอร์รัพท์

สามารถใช้งานระบบอินเตอร์รัพท์ AVR และ Arduino ได้

สามารถเขียนโปรแกรมตอบสนองสัญญาณอินเตอร์รัพท์ได้

การอินเตอร์รัพท์คืออะไร

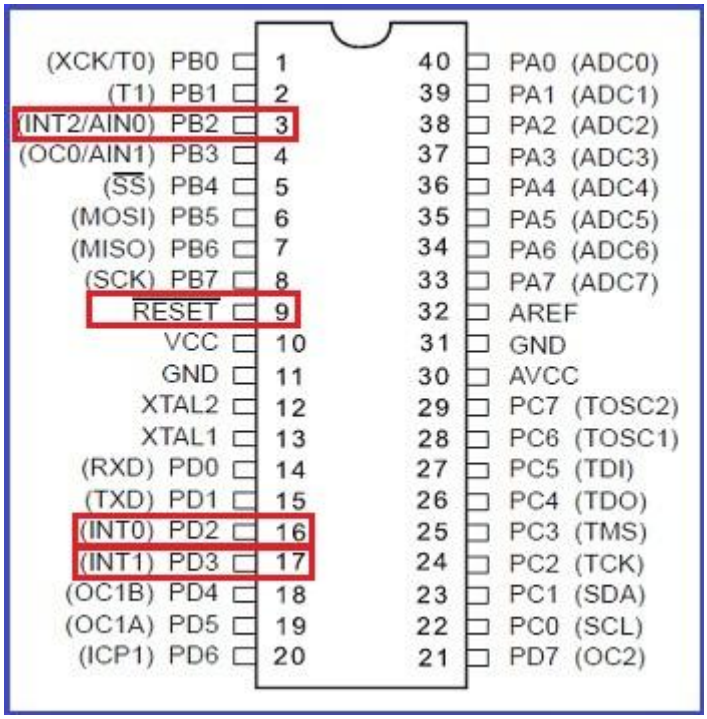
เมื่อมีเหตุการณ์ใดเหตุการณ์หนึ่งเกิดขึ้นแล้วไปขัดจังหวะการทำงานปกติของซีพียูเราเรียกว่าการอินเตอร์รัพท์ การขัดจังหวะของเหตุการณ์นั้นอาจทำให้ซีพียูต้องทำงานอื่นก่อนเมื่อเสร็จแล้วจึงจะกลับไปทำงานเดิมหรือไม่ก็ได้



ระบบอินเตอร์รัพท์ของ AVR

ประเภทของการอินเตอร์รัพท์

1. **สัญญาณอินเตอร์รัพท์จากภายนอก** เป็นการอินเตอร์รัพท์เนื่องจากสัญญาณภายนอกตัวไมโครคอนโทรเลอร์ ซึ่งได้แก่สัญญาณที่ขา
   - Reset
   - External Interrupt 0 (INT0) ขาสัญญาณ PD2
   - External Interrupt 1 (INT1) ขาสัญญาณ PD3
   - External Interrupt 2 (INT2) ขาสัญญาณ PB2



โดยสัญญาณที่ขาพอร์ท PD2 (INT0) PD3 (INT1) และ PB2 (INT2) สามารถกำหนดรูปแบบการเกิดอินเตอร์รัพท์ได้หลายแบบเช่น

- ขณะที่สัญญาณเปลี่ยนเป็นระดับต่ำ
- ขณะที่มีการเปลี่ยนแปลงระดับสัญญาณ
- ที่ขอบขาลงของสัญญาณ
- ที่ขอบขาขึ้นของสัญญาณ

2. **สัญญาณอินเตอร์รัพท์จากภายใน** แหล่งกำเนิดสัญญาณนี้จะเป็นวงจรภายใน Microcontroller เอง เช่น

- วงจรนับ/จับเวลา
- การสื่อสารสัญญาณอนุกรม
- การแปลงสัญญาณอนาลอก
- การเขียนข้อมูลลงหน่วยความจำ

ตำแหน่งโปรแกรมรองรับการอินเตอร์รัพท์

เมื่อมีสัญญาณอินเตอร์เข้ามาที่ซีพียู ถ้าซีพียูยอมรับการอินเตอร์รัพท์นั้น ก็จะไปทำงานโปรแกรมที่กำหนดไว้ โปรแกรมนี้เรียกว่า "โปรแกรมรองรับการอินเตอร์รัพท์" โดยแอดเดรสเริ่มต้นของโปรแกรมรองรับการอินเตอร์รัพท์ จะถูกกำหนดโดยผู้ออกแบบซีพียู สำหรับ AVR นั้นแอดเดรสเริ่มต้นถูกกำหนดไว้ตามตารางที่ 1 เช่นถ้าเกิดการอินเตอร์รัพท์จาก INT 0 แล้วต้องการให้ซีพียูทำการงานรองการ

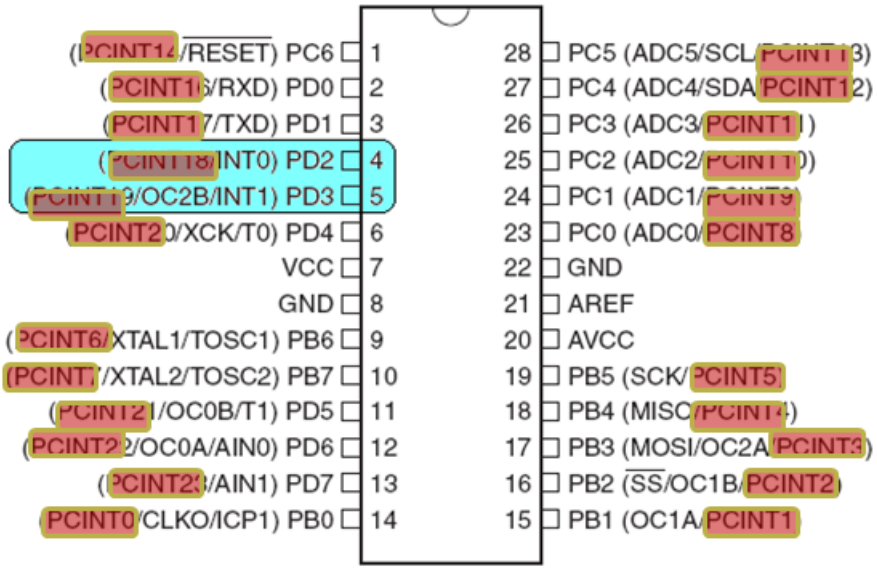
อินเตอร์รัพท์นี้ โปรแกรมรองรับการอินเตอร์รัพท์นี้ต้องเริ่มต้นที่ตำแหน่ง 0x0002

12.4 Interrupt vectors in Atmel ATmega168

Table 12-4. Reset and interrupt vectors in ATmega168.

Vector no.	Program address(2)	Source	Interrupt definition
1	0x0000(1)	RESET	External pin, power-on reset, brown-out reset and watchdog system reset
2	0x0002	INT0	External interrupt request 0
3	0x0004	INT1	External interrupt request 1
4	0x0006	PCINT0	Pin change interrupt request 0
5	0x0008	PCINT1	Pin change interrupt request 1
6	0x000A	PCINT2	Pin change interrupt request 2
7	0x000C	WDT	Watchdog time-out interrupt
8	0x000E	TIMER2 COMPA	Timer/Counter2 compare match A
9	0x0010	TIMER2 COMPB	Timer/Counter2 compare match B
10	0x0012	TIMER2 OVF	Timer/Counter2 overflow
11	0x0014	TIMER1 CAPT	Timer/Counter1 capture event
12	0x0016	TIMER1 COMPA	Timer/Counter1 compare match A
13	0x0018	TIMER1 COMPB	Timer/Coutner1 compare match B
14	0x001A	TIMER1 OVF	Timer/Counter1 overflow
15	0x001C	TIMER0 COMPA	Timer/Counter0 compare match A
16	0x001E	TIMER0 COMPB	Timer/Counter0 compare match B
17	0x0020	TIMER0 OVF	Timer/Counter0 overflow
18	0x0022	SPI, STC	SPI serial transfer complete
19	0x0024	USART, RX	USART Rx complete
20	0x0026	USART, UDRE	USART, data register empty
21	0x0028	USART, TX	USART, Tx complete
22	0x002A	ADC	ADC conversion complete
23	0x002C	EE READY	EEPROM ready
24	0x002E	ANALOG COMP	Analog comparator
25	0x0030	TWI	2-wire serial interface
26	0x0032	SPM READY	Store program memory ready

External Interrupt and Pin Change Interrupt



Pin Change Interrupt Vector

Vector No	Program Address	Source	Description
4	0×006	PCINT0 (PB0 to PB7)	Pin Change Interrupt Request 0
5	0×008	PCINT1 (PC0 to PC6)	Pin Change Interrupt Request 1
6	0x00A	PCINT2 (PD0 to PD7)	Pin Change Interrupt Request 2

detect a change on any pin. Because each interrupt is for a group of pin, you will also need to do extra work to determine which pin changed and how it changed.

Reset and Interrupt Vectors Placement

BOOTRST[1]	IVSEL	Reset Address	Interrupt Vectors Start Address
1	0	0x000	0x001
1	1	0x000	Boot Reset Address + 0x001
0	0	Boot Reset Address	0x001
0	1	Boot Reset Address	Boot Reset Address + 0x001

ทั้ง 2 บิตอยู่ใน MCUCR – MCU control register

Bit 0 – IVCE: Interrupt vector change enable

Bit 1 – IVSEL: Interrupt vector select

0 -> the interrupt vectors are placed at the start of the flash memory

1 -> the interrupt vectors are moved to the beginning of the boot loader section of the flash.

การกำหนดสถานะการอินเตอร์รัพท์

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

รีจิสเตอร์ SREG (Status Register)

Bit	7	6	5	4	3	2	1	0	
	I	T	H	S	V	N	Z	C	SREG
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – I: Global Interrupt Enable**

The Global Interrupt Enable bit must be set for the interrupts to be enabled. The individual interrupt enable control is then performed in separate control registers. If the Global Interrupt Enable Register is cleared, none of the interrupts are enabled independent of the individual interrupt enable settings. The I-bit is cleared by hardware after an interrupt has occurred, and is set by the RETI instruction to enable subsequent interrupts. The I-bit can also be set and cleared by the application with the SEI and CLI instructions, as described in the Instruction Set Reference.

รีจิสเตอร์ MCUCR (MCU Control Register)

Bit	7	6	5	4	3	2	1	0	
	SE	SM2	SM1	SM0	ISC11	ISC10	ISC01	ISC00	MCUCR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 3, 2 – ISC11, ISC10: Interrupt Sense Control 1 Bit 1 and Bit 0**

The External Interrupt 1 is activated by the external pin INT1 if the SREG I-bit and the corresponding interrupt mask in the GICR are set. The level and edges on the external INT1 pin that activate the interrupt are defined in Table 13-1. The value on the INT1 pin is sampled before detecting edges. If edge or toggle interrupt is selected, pulses that last longer than one clock period will generate an interrupt. Shorter pulses are not guaranteed to generate an interrupt. If low level interrupt is selected, the low level must be held until the completion of the currently executing instruction to generate an interrupt.

Interrupt Sense Control

Table 13-1. Interrupt 1 Sense Control

ISC11	ISC10	Description
0	0	The low level of INT1 generates an interrupt request.
0	1	Any logical change on INT1 generates an interrupt request.
1	0	The falling edge of INT1 generates an interrupt request.
1	1	The rising edge of INT1 generates an interrupt request.

- **Bit 1, 0 – ISC01, ISC00: Interrupt Sense Control 0 Bit 1 and Bit 0**

The External Interrupt 0 is activated by the external pin INT0 if the SREG I-flag and the corresponding interrupt mask are set. The level and edges on the external INT0 pin that activate the interrupt are defined in Table 13-2. The value on the INT0 pin is sampled before detecting edges. If edge or toggle interrupt is selected, pulses that last longer than one clock period will generate an interrupt. Shorter pulses are not guaranteed to generate an interrupt. If low level interrupt is selected, the low level must be held until the completion of the currently executing instruction to generate an interrupt.

Table 13-2. Interrupt 0 Sense Control

ISC01	ISC00	Description
0	0	The low level of INT0 generates an interrupt request.
0	1	Any logical change on INT0 generates an interrupt request.
1	0	The falling edge of INT0 generates an interrupt request.
1	1	The rising edge of INT0 generates an interrupt request.

รีจิสเตอร์ MCUCSR (MCU Control and status Register)

Bit	7	6	5	4	3	2	1	0	
	JTD	ISC2	–	JTRF	WDRF	BORF	EXTRF	PORF	MCUCSR
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	See Bit Description					

- **Bit 6 – ISC2: Interrupt Sense Control 2**

The Asynchronous External Interrupt 2 is activated by the external pin INT2 if the SREG I-bit and the corresponding interrupt mask in GICR are set. If ISC2 is written to zero, a falling edge on INT2 activates the interrupt. If ISC2 is written to one, a rising edge on INT2 activates the interrupt. Edges on INT2 are registered asynchronously. Pulses on INT2 wider than the minimum pulse width given in Table 36 will generate an interrupt. Shorter pulses are not guaranteed to generate an interrupt. When changing the ISC2 bit, an interrupt can occur. Therefore, it is recommended to first disable INT2 by clearing its Interrupt Enable bit in the GICR Register. Then, the ISC2 bit can be changed. Finally, the INT2 Interrupt Flag should be cleared by writing a logical one to its Interrupt Flag bit (INTF2) in the GIFR Register before the interrupt is re-enabled.

รีจิสเตอร์ GICR (General Interrupt Control)

Bit	7	6	5	4	3	2	1	0	
	INT1	INT0	INT2	–	–	–	IVSEL	IVCE	GICR
Read/Write	R/W	R/W	R/W	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – INT1: External Interrupt Request 1 Enable**

When the INT1 bit is set (one) and the I-bit in the Status Register (SREG) is set (one), the external pin interrupt is enabled. The Interrupt Sense Control1 bits 1/0 (ISC11 and ISC10) in the MCU General Control Register (MCUCR) define whether the External Interrupt is activated on rising and/or falling edge of the INT1 pin or level sensed. Activity on the pin will cause an interrupt request even if INT1 is configured as an output. The corresponding interrupt of External Interrupt Request 1 is executed from the INT1 interrupt Vector.

- **Bit 6 – INT0: External Interrupt Request 0 Enable**

When the INT0 bit is set (one) and the I-bit in the Status Register (SREG) is set (one), the external pin interrupt is enabled. The Interrupt Sense Control0 bits 1/0 (ISC01 and ISC00) in the MCU General Control Register (MCUCR) define whether the External Interrupt is activated on rising and/or falling edge of the INT0 pin or level sensed. Activity on the pin will cause an interrupt request even if INT0 is configured as an output. The corresponding interrupt of External Interrupt Request 0 is executed from the INT0 interrupt vector.

- **Bit 5 – INT2: External Interrupt Request 2 Enable**

When the INT2 bit is set (one) and the I-bit in the Status Register (SREG) is set (one), the external pin interrupt is enabled. The Interrupt Sense Control2 bit (ISC2) in the MCU Control and Status Register (MCUCSR) defines whether the External Interrupt is activated on rising or falling edge of the INT2 pin. Activity on the pin will cause an interrupt request even if INT2 is configured as an output. The corresponding interrupt of External Interrupt Request 2 is executed from the INT2 Interrupt Vector.

รีจิสเตอร์ GIFR (General Interrupt Flag)

Bit	7	6	5	4	3	2	1	0	
	INTF1	INTF0	INTF2	–	–	–	–	–	GIFR
Read/Write	R/W	R/W	R/W	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – INTF1: External Interrupt Flag 1**

When an edge or logic change on the INT1 pin triggers an interrupt request, INTF1 becomes set (one). If the I-bit in SREG and the INT1 bit in GICR are set (one), the MCU will jump to the corresponding Interrupt Vector. The flag is cleared when the interrupt routine is executed. Alternatively, the flag can be cleared by writing a logical one to it. This flag is always cleared when INT1 is configured as a level interrupt.

- **Bit 6 – INTF0: External Interrupt Flag 0**

When an edge or logic change on the INT0 pin triggers an interrupt request, INTF0 becomes set (one). If the I-bit in SREG and the INT0 bit in GICR are set (one), the MCU will jump to the corresponding interrupt vector. The flag is cleared when the interrupt routine is executed. Alternatively, the flag can be cleared by writing a logical one to it. This flag is always cleared when INT0 is configured as a level interrupt.

- **Bit 5 – INTF2: External Interrupt Flag 2**

When an event on the INT2 pin triggers an interrupt request, INTF2 becomes set (one). If the I-bit in SREG and the INT2 bit in GICR are set (one), the MCU will jump to the corresponding Interrupt Vector. The flag is cleared when the interrupt routine is executed. Alternatively, the flag can be cleared by writing a logical one to it. Note that when entering some sleep modes with the INT2 interrupt disabled, the input buffer on this pin will be disabled. This may cause a logic change in internal signals which will set the INTF2 Flag. See “Digital Input Enable and Sleep Modes” on page 54 for more information.

****** คำสั่งที่น่าสนใจของ GCC ที่เขียนอยู่ในตัวอย่างต่างๆ ******

PORTD |=(1<<1)|(1<<3)|(1<<5)|(1<<7);

ทำให้บิตที่ 1,3,5,7 เป็น 1 บิตอื่นเป็น 0 หมด แล้วเอาไป OR กับ PORTD ผลลัพธ์เก็บที่ PORTD

PORTD &=(1<<2)|(1<<5)|(1<<7);

ทำให้บิตที่ 2,5,7 เป็น 1 บิตอื่นเป็น 0 หมด แล้วเอาไป AND กับ PORTD ผลลัพธ์เก็บที่PORTD

PORTD = (1<<PD0)|(1<<PD1)|(1<<PD2);

PORTD บิตที่ 0 , 1, 2 เป็น 1 บิตอื่นเป็น 0

การใช้งานอินเตอร์รัพท์เนื่องจากสัญญาณภายนอก

```
//------------ Includes --------//
#include <avr/io.h>                     // AVR device-specific IO definitions
#include <avr/interrupt.h>              // Interrupt Service routine
#include <compat/deprecated.h>          // Use sbi(), cbi() function

#define F_CPU 8000000UL                 // 8 MHz
#include <util/delay.h>                 // header file implement simple delay loops

unsigned char i=0;

//-------- delay_ms  //
void delay_ms(unsigned int i)
{
   for (; i>0; i--)
       _delay_ms(1);
}
//------------- Main Functions ------------//
int main(void)
{
   // The rising edge of INT0 generates an interrupt request
   MCUCR = (1<<ISC01)|(1<<ISC00);

   GICR  = (1<<INT0);           // Enable INT0 interrupt
   SREG  = 0x80;                // Set I-bit Global interrupt

   DDRA  = (1<<DDA0);           // PORT A0 Output
   PORTA = (0<<PA0);            // Clear port

   while (1) {                  // Loop nothing
                ;
   }

   return 0;
}

// ------------ External Interrupt Request 0 ------//
// Interrupt INT0 (Port PD2)
ISR (INT0_vect)
{
   if (i)
       sbi(PORTA,0);       // Set bit
   else
       cbi(PORTA,0);       // Clear bit

   i = !i;                 // Toggle bit
   delay_ms(200);          // Delay debound
}
```

ในกรณีต้องการใช้งานอินเตอร์รัพท์1 (INT1)

```
int main(void)
{
   MCUCR |= (1<<ISC11)|(0<<ISC10);     // The falling edge of INT1

   GICR  = (1<<INT1);        // Enable INT1 interrupt
   sei();                    // Set I-bit Global interrupt

   DDRA  = (1<<DDA7)|(1<<DDA1)|(1<<DDA0);  // PORT A0,A1,A7 Output
   PORTA = (0<<PA7)|(0<<PA1)|(0<<PA0);     // Clear port

   while (1) {                       // Loop Forever
      sbi(PORTA,7);              // Set bit PA7
      delay_ms(500);              // Delay 0.5s
      cbi(PORTA,7);              // Clear bit PA7
      delay_ms(500);              // Delay 0.5s
   }
   return 0;
}
//--------- External Interrupt Request 1 ----------//
// Interrupt INT1 (PD3)
ISR (INT1_vect)
{
   if (bit1) {
      sbi(PORTA,1);                // Set bit PA1
   } else {
      cbi(PORTA,1);                // Clear bit PA1
   }

   bit1 = !bit1;                    // Toggle bit
   delay_ms(200);                   // delay debound
}
```

การใช้อินเตอร์รัพท์ภายนอกใน Arduino

- attachInterrupt()
- detachInterrupt()

attachInterrupt()

Specifies a function to call when an external interrupt occurs. Replaces any previous function that was attached to the interrupt. Most Arduino boards have two external interrupts: numbers 0 (on digital pin 2) and 1 (on digital pin 3). The table below shows the available interrupt pins on various boards.

Board	int.0	int.1	int.2	int.3	int.4	int.5
Uno, Nano	2	3				
Mega2560	2	3	18	19	20	21

หมายเหตุ เนื่องจาก บอร์ด Arduino มีผู้ผลิตหลายเจ้า ดังนั้นควรดูคู่มือของบอร์ดนั้นๆเป็นหลัก

Syntax

attachInterrupt(interrupt, function, mode)

สำหรับ ***Arduino Due*** ใช้

attachInterrupt(pin, function, mode)

interrupt: the number of the interrupt (int)

pin: the pin number (Arduino Due only)

function: the function to call when the interrupt occurs; this function must take no parameters and return nothing. This function is sometimes referred to as an interrupt service routine.mode: defines when the interrupt should be triggered. Four contstants are predefined as valid values:

mode:

- 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.
- HIGH to trigger the interrupt whenever the pin is high. (Arduino Due only)

detachInterrupt()

Description Turns off the given interrupt.

Syntax

detachInterrupt(interrupt)

detachInterrupt(pin) (Arduino Due only)

Parameters

interrupt: the number of the interrupt to disable

pin: the pin number of the interrupt to disable (Arduino Due only)

ตัวอย่างที่ 1

```
int pin = 13;
volatile int state = LOW;
void setup()
{
  pinMode(pin, OUTPUT);
  attachInterrupt(0, blink, CHANGE);
}

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

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

ตัวอย่างที่ 2

```
int pbIn = 0;     // Interrupt 0 is on DIGITAL PIN 2!
int ledOut = 4;                // The output LED pin
volatile int state = LOW;      // The input state toggle
void setup()
{
  // Set up the digital pin 2 to an Interrupt and Pin 4 to an Output
  pinMode(ledOut, OUTPUT);

 //Attach the interrupt to the input pin and monitor for ANY Change
 attachInterrupt(pbIn, stateChange, CHANGE);
}
void loop()
{

//Simulate a long running process or complex task

for (int i = 0; i < 100; i++){
    delay(10); // do nothing but waste some time
  }
}
void stateChange()

{

  state = !state;

  digitalWrite(ledOut, state);

}
```

ตัวอย่างที่ 3

```
#include < avr/io.h >              // Definition of interrupt names
#include < avr/interrupt.h >      // ISR interrupt service routine
int ledPin = 13;                  // LED connected to digital pin 13
int sensePin = 2;                 // This is the INT0 Pin of the ATMega8. We need to declare the data
                                  //exchange variable to be volatile - the value is read from memory.
volatile int value = 0;
// Install the interrupt routine.
ISR(INT0_vect) {
  // check the value again - since it takes some time to
  // activate the interrupt routine, we get a clear signal.
  value = digitalRead(sensePin);
}

void setup() {
  Serial.begin(9600);
  Serial.println("Initializing ihandler");
  pinMode(ledPin, OUTPUT);        // sets the digital pin as output
  pinMode(sensePin, INPUT);       // read from the sense pin
  Serial.println("Processing initialization");
 GICR |= ( 1 < < INT0);           // Global Enable INT0 interrupt
 MCUCR |= ( 1 << ISC00);          // Signal change triggers interrupt
  MCUCR |= ( 0 << ISC01);
  Serial.println("Finished initialization");
}
void loop() {
  if (value) {
    Serial.println("Value high!");
    digitalWrite(ledPin, HIGH);
  } else {
    Serial.println("Value low!");
    digitalWrite(ledPin, LOW);
  }
  delay(100);
}
```