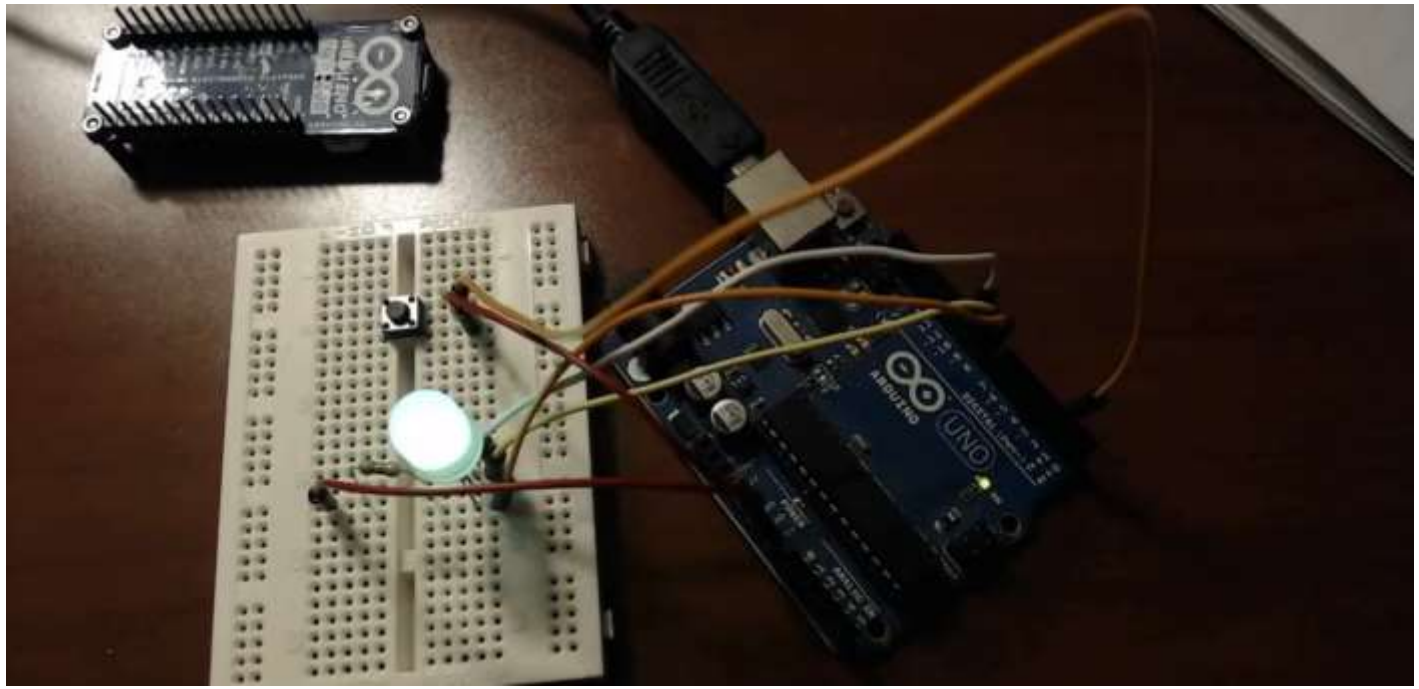
EMBEDDED SYSTEMS CMPE-453

Department of Computer Engineering

Interrupt Handling

Interrupt

- Device who needs service sends an interrupt signal to notify Microcontroller
- Microcontroller stops whatever it is doing and serves the device.
- The program associated with intterrupt is called interrupt service routine / interrupt handler
- Devices get service based on the priority assigned.
- Can ignore a device's request for service.



INTERRUPT SERVICE ROUTINE (ISR)

Invocation of Interrupt

ISR

Location in memory for each Interrupt

holds the address of ISR

Vector No.	Program Address	Source	Interrupt Definition
1	0x0000	RESET	External pin, power-on reset, brown-out reset and watchdog system reset
2	0x002	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/Counter1 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

STEPS FOR EXECUTING INTERRUPT

Activation of interrupt

1. It finishes the instruction it is currently executing and saves the address of the next instruction (program counter) on the stack.

2. It jumps to a fixed location in memory called the *interrupt vector table*. The interrupt vector table directs the microcontroller to the address of the interrupt service routine (ISR).

3. The microcontroller starts to execute the interrupt service subroutine until it reaches the last instruction of the subroutine, which is RETI (return from interrupt).

4. Upon executing the RETI instruction, the microcontroller returns to the place where it was interrupted. First, it gets the program counter (PC) address from the stack by popping the top bytes of the stack into the PC. Then it starts to execute from that address.

HOW INTERRUPT HANDLING WORKS?

- Specific flags are set when an interrupt occurs.
  - Interrupts are internal/external software/hardware events.

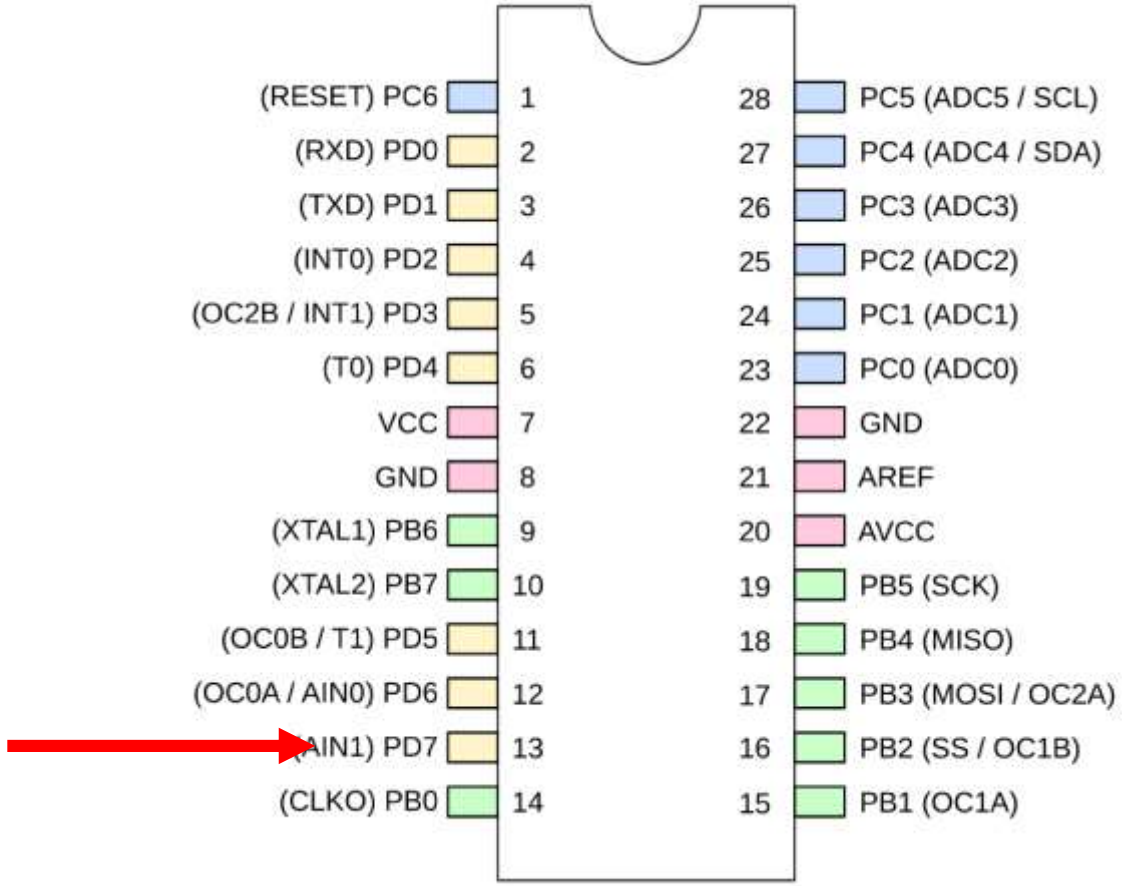
- Micrcontroller, upon finding that an interrupt flag is set,
  - Disables the global interrupt bit I in SREG
  - Goes to corresponding entry in the interrupt vector table.
  - The entry directs microcontroller to executes a special function (interrupt service routine) to take predefined actions.
  - After ISR is complete, I bit of SREG is set back to 1.
  - If there are any other interrupts waiting, they are served in order.
  - Otherwise, control return to main program (at the point where it left).

KIND OF INTERRUPTS

- Hardware Interrupts:
  - Triggered by events from outside of the microcontroller (INT0, INT1, PCINT)
  - Example: Pressing a push button connected to PD7 (PCINT).
- Software Interrupts:
  - Triggered by software running on the microcontroller.
  - Example: Transmit interrupt for USART.

Bit	7	6	5	4	3	2	1	0	
	RXCIEn	TXCIEn	UDRIEn	RXENn	TXENn	UCSZn2	RXB8n	TXB8n	UCSRnB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit	7	6	5	4	3	2	1	0	
	RXCn	TXCn	UDREn	FEn	DORn	UPEn	U2Xn	MPCMn	UCSRnA
Read/Write	R	R/W	R	R	R	R	R/W	R/W	
Initial Value	0	0	1	0	0	0	0	0	

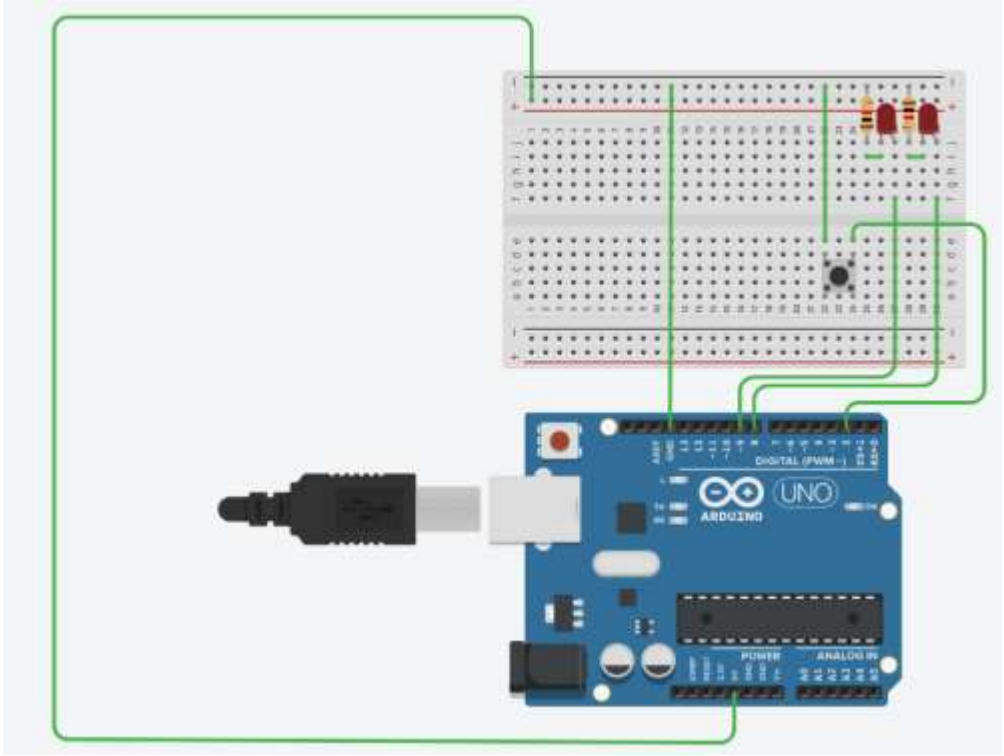



```
ISR(PCINT2_vect)
{

//body of ISR.



}
```

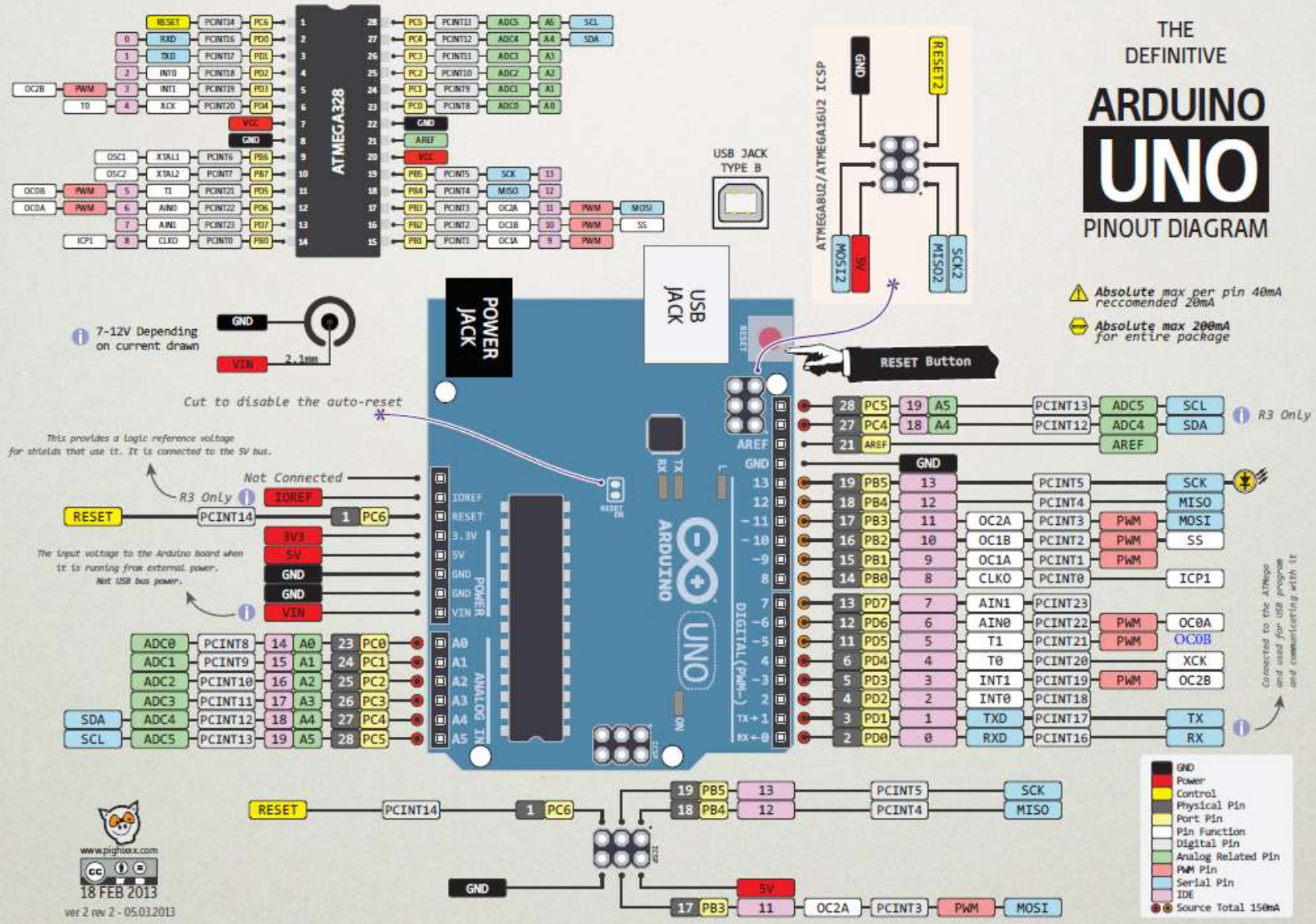
Interrupt	Address (hex)
Reset	0000
External Interrupt Request 0	0002
External Interrupt Request 1	0004
Pin Change Interrupt Request	0006
Pin Change Interrupt Request	0008
Pin Change Interrupt Request	000A
Watchdog Time-out Interrupt	000C
Timer/Counter2 Compare Match A	000E
Timer/Counter2 Compare Match B	0010
Timer/Counter2 Overflow	0012
Timer/Counter1 Capture Event	0014
Timer/Counter1 Compare Match A	0016
Timer/Counter1 Compare Match B	0018
Timer/Counter1 Overflow	001A
Timer/Counter0 Compare Match A	001C
Timer/Counter0 Compare Match B	001E
Timer/Counter0 Overflow	0020
SPI Serial Transfer Complete	0022
USART Rx Complete	0024
USART Data Register Empty	0026
USART Tx Complete	0028
ADC Conversion Complete	002A
EEPROM ready	002C
Analog Comparator	002E

EXAMPLE: INT0

- Program the microcontroller such that
  - LED0 blinks with a delay of 200 ms
  - LED1 is turned on when push button is pressed.
  - Push button must be handled using ISR for INT0



Arduino Connections



REGISTER DESCRIPTION

- AVR Status Register
  - Global Interrupt enable
  - 0: All interrupts are disabled
  - 1: Interrupts with their respective flag bits set to 1 are enabled.

Bit	7	6	5	4	3	2	1	0	
0x3F (0x5F)	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	

REGISTER DESCRIPTION

- EIMSK-External Interrupt Mask Register
- INT0/INT1
  - 0: disabled
  - 1: enabled (provided that bit I of SREG is also set to 1).

Bit	7	6	5	4	3	2	1	0	
0x1D (0x3D)	–	–	–	–	–	–	INT1	INT0	EIMSK
Read/Write	R	R	R	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

REGISTER DESCRIPTION

- EICRA: External Interrupt Control Register A

Bit	7	6	5	4	3	2	1	0	
(0x69)	–	–	–	–	ISC11	ISC10	ISC01	ISC00	EICRA
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

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.

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.

```
// ------- Preamble -------- //
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>
#include "pinDefines.h"

ISR(INT0_vect)
{
/* Run every time there is a change on button */
if (bit_is_clear(BUTTON_PIN, BUTTON))
      LED_PORT |= (1 << LED1);
else
      LED_PORT &= ~(1 << LED1);
}

void initInterrupt0(void)
{
EIMSK |= (1 << INT0); /* enable INT0 */
EICRA |= (1 << ISC00); /* trigger when button changes */
sei(); /* set (global) interrupt enable bit */
}
```

```
int main(void)
{
// -------- Inits --------- //
LED_DDR = 0xff; /* all LEDs active */
BUTTON_PORT |= (1 << BUTTON); /* pullup */
initInterrupt0();
// ------ Event loop ------ //
while (1)
      {
      _delay_ms(200);
      LED_PORT ^= (1 << LED0);
      } /* End event loop */

return 0; /* This line is never reached */
}
```

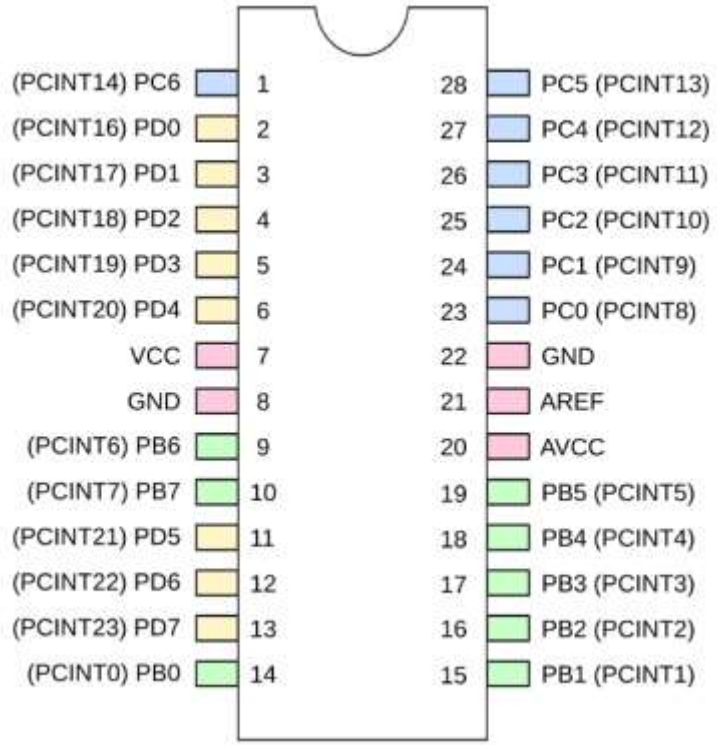
PCINT: PIN CHANGE INTERRUPT

- INT0 and INT1:
  - External Hardware interrupts
  - Dedicated registers for enabling them (EIMSK)
  - Dedicated register for defining the sensitivity (EICRA)
  - Vector name for ISRs: INT0_vect, INT1_vect
- PCINT:
  - Used to signal an interrupt on a change at a pin
  - Support interrupt handling for all 8x3=24 I/O pins.
  - Unlike INT0 and INT1, interupt can not be requested only for falling edge or rising edge, but for any change/transition.
  - Vector name for ISR: PCINTx_vect

PCINTX

- Three PCINTs (PCINT0, PCINT1, PCINT2).
- PCINT0: triggered by change on any of PB0-PB7 (Bank 0)
- PCINT1: triggered by change on any of PC0-PC7 (Bank 1)
- PCINT2: triggered by change on any of PD0-PD7 (Bank 2)

Name in datasheet	Vector name for ISRs	Which pins are covered
PCINT0	PCINT0_vect	PB0 .. PB7
PCINT1	PCINT1_vect	PC0 .. PC6
PCINT2	PCINT2_vect	PD0 .. PD7



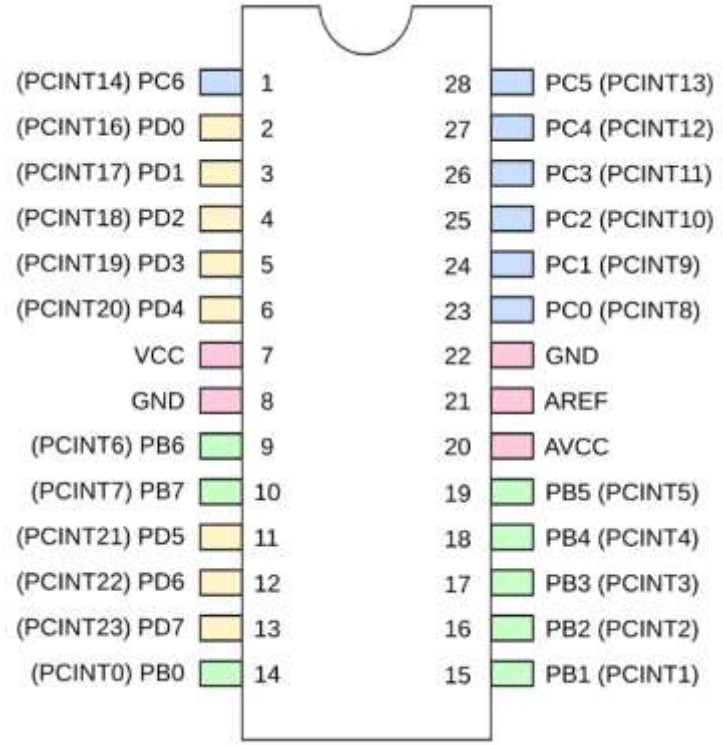
HOW TO ENABLE PCINTX?

- Three layered process.
  1. Set the bit corresponding to PCINTx in PCMSKn register.
  2. Set the bit PCIEn in PCICR correspnding to the bank to which PCINTx belongs.
  3. Set the I bit in SREG.

EXAMPLE-1

- Write C-code to enable PCINT10 (PC2) (C port: Bank1)

1. Set corresponding bit in PCMSK register.



Bit	7	6	5	4	3	2	1	0	
(0x6C)	–	PCINT14	PCINT13	PCINT12	PCINT11	PCINT10	PCINT9	PCINT8	PCMSK1
Read/Write	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

```
PCMSK1 |=(1<<PCINT10);
Or
PCMSK1 |=(1<<PC2);
```

Bit	7	6	5	4	3	2	1	0	
(0x6B)	PCINT7	PCINT6	PCINT5	PCINT4	PCINT3	PCINT2	PCINT1	PCINT0	PCMSK0
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	

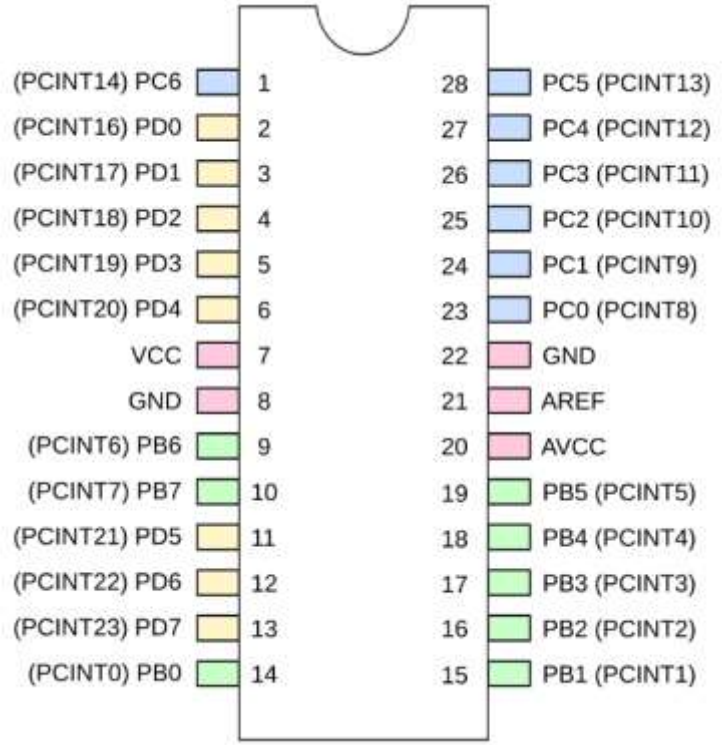
Bit	7	6	5	4	3	2	1	0	
(0x6D)	PCINT23	PCINT22	PCINT21	PCINT20	PCINT19	PCINT18	PCINT17	PCINT16	PCMSK2
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

EXAMPLE

- Write C-code to enable PCINT10.

2. Set the bit PCIEn in PCICR corresponding to the bank to which PCINTx belongs.(Port C: bank 1)

```
PCICR |=(1 << PCIE1);
```



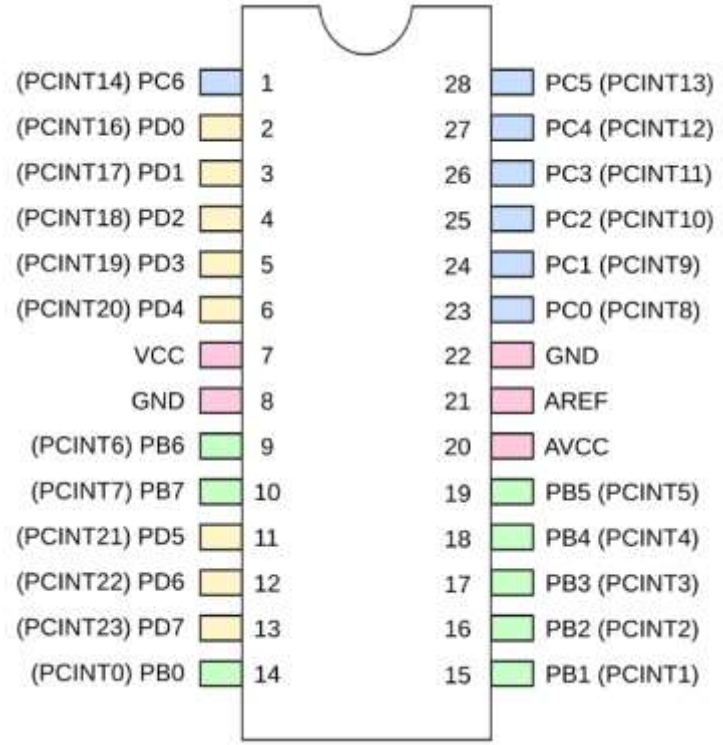
Bit	7	6	5	4	3	2	1	0	
(0x68)	–	–	–	–	–	PCIE2	PCIE1	PCIE0	PCICR
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

EXAMPLE

- Write C-code to enable PCINT10.

3. Set the I bit in SREG.

```
sei();
Or
SREG |=(1 << I);
```

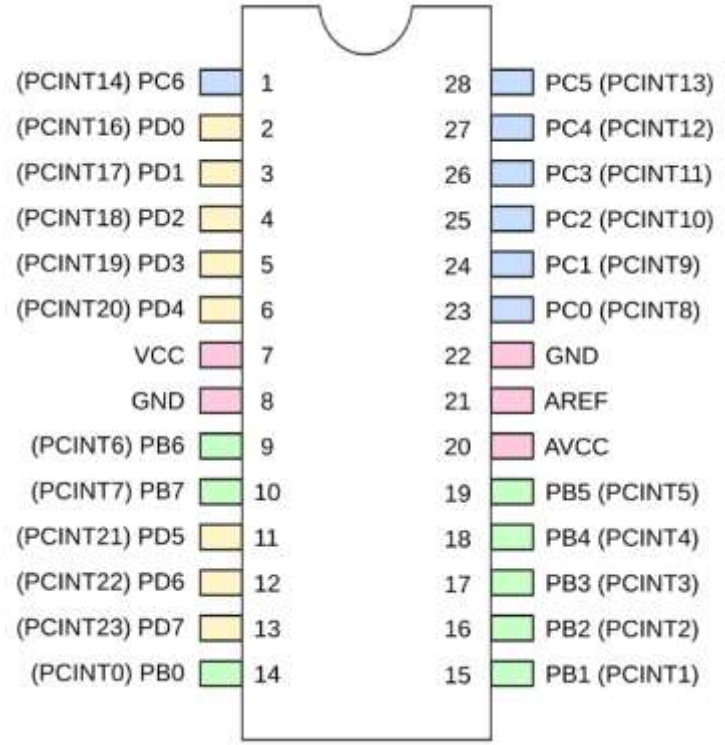


Bit	7	6	5	4	3	2	1	0	
0x3F (0x5F)	I	T	H	S	V	N	Z	C	SREG
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

EXAMPLE-2

- Write C-code to enable PCINT21 (PD5) (D port: Bank2)

1. Set corresponding bit in PCMSK register.



Bit	7	6	5	4	3	2	1	0	
(0x6C)	–	PCINT14	PCINT13	PCINT12	PCINT11	PCINT10	PCINT9	PCINT8	PCMSK1
Read/Write	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

```
PCMSK2 |=(1<<PCINT21);
Or
PCMSK1 |=(1<<PD5);
```

Bit	7	6	5	4	3	2	1	0	
(0x6B)	PCINT7	PCINT6	PCINT5	PCINT4	PCINT3	PCINT2	PCINT1	PCINT0	PCMSK0
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	

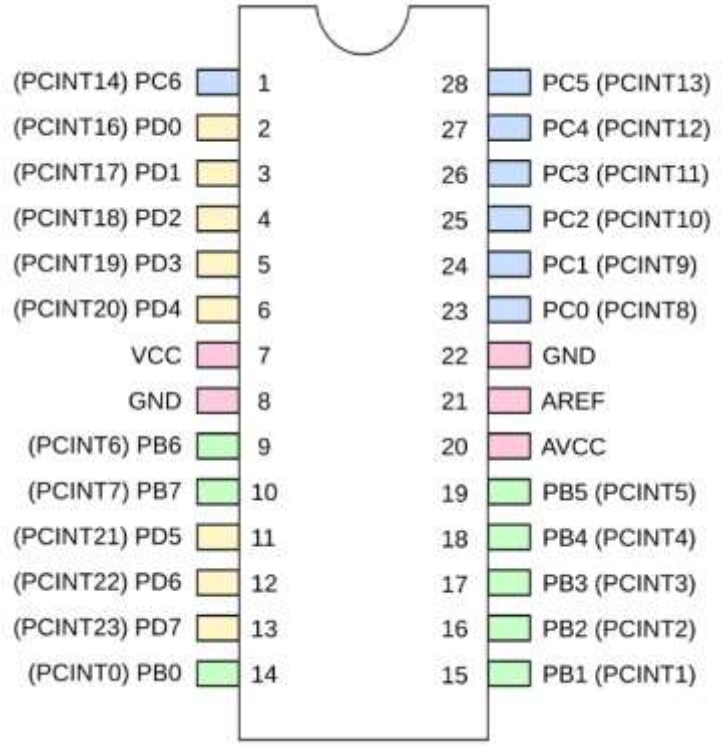
Bit	7	6	5	4	3	2	1	0	
(0x6D)	PCINT23	PCINT22	PCINT21	PCINT20	PCINT19	PCINT18	PCINT17	PCINT16	PCMSK2
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

EXAMPLE

- Write C-code to enable PCINT10.

2. Set the bit PCIEn in PCICR corresponding to the bank to which PCINTx belongs.(Port D: bank 2)

```
PCICR |=(1 << PCIE2);
```



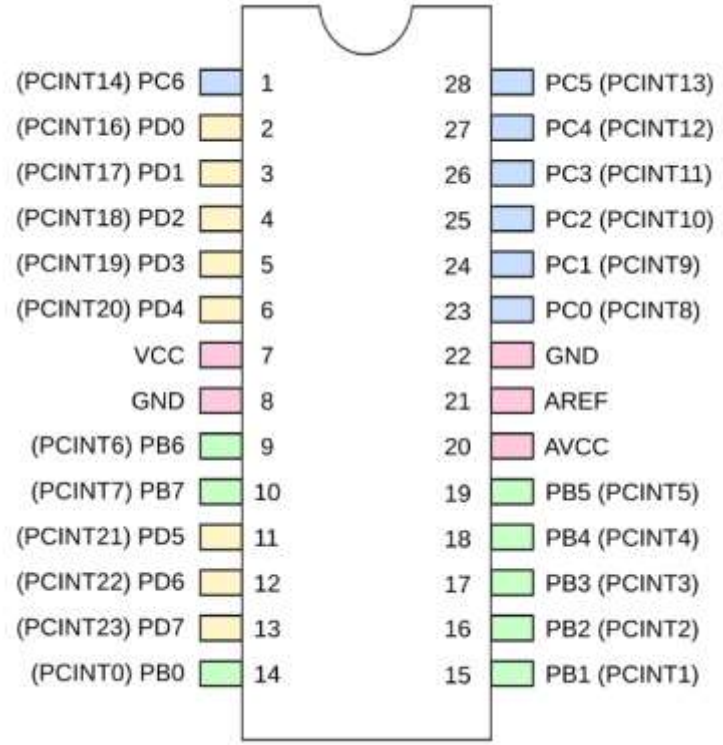
Bit	7	6	5	4	3	2	1	0	
(0x68)	–	–	–	–	–	PCIE2	PCIE1	PCIE0	PCICR
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

EXAMPLE

- Write C-code to enable PCINT10.

3. Set the I bit in SREG.

```
sei();
Or
SREG |=(1 << I);
```



Bit	7	6	5	4	3	2	1	0	
0x3F (0x5F)	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	