COMPUTER ENGINEERING
MAHIDOL UNIVERSITY

EGCI330: Microprocessor and Interfacing

Interrupt Handler AVR328P

COMPUTER ENGINEERING
MAHIDOL UNIVERSITY

Outline

- I/O Operation
- AVR 328P Interrupts
  - External Interrupt
  - Timer Interrupt
  - Pin Change Interrupt
- Interrupt in C language

COMPUTER ENGINEERING
MAHIDOL UNIVERSITY

I/O Operation

I/O operation are accomplished through exchanging the data between the external environment and the processor.

There are two types of I/O operation

- Asynchronous I/O Operation
- Synchronous I/O Operation

COMPUTER ENGINEERING
MAHIDOL UNIVERSITY

I/O Operation

- Synchronous I/O Operation
  - Synchronous operations require that one operation must wait for another to complete before it can begin.
  - Eg. Polling
- Asynchronous I/O Operation
  - It is the operation that starts the communication and then perform processing which does not require that the I/O be completed.
  - Eg. Interrupt

COMPUTER ENGINEERING
MAHIDOL UNIVERSITY

I/O Operation

Polling

- Ties down the CPU

```
while (true)
{
  if(PIND.2 == 0)
  //do something;
}
```

Interrupt

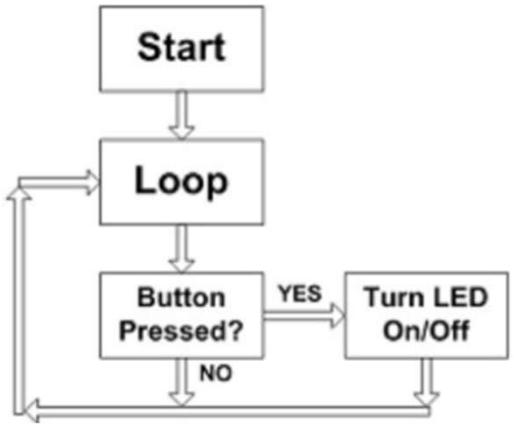
- Efficient CPU use
- Has priority
- Can be masked

```
main( )
{
  Do your common task
}
whenever PIND.2 is 0 then
   do something
```

I/O Operation

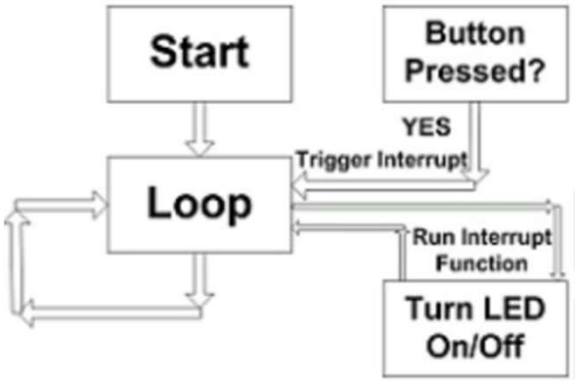
Polling

- Ties down the CPU



Interrupt

- Efficient CPU use
- Has priority
- Can be masked

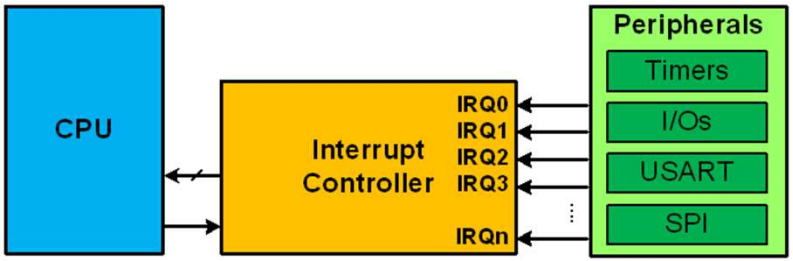


Interrupts

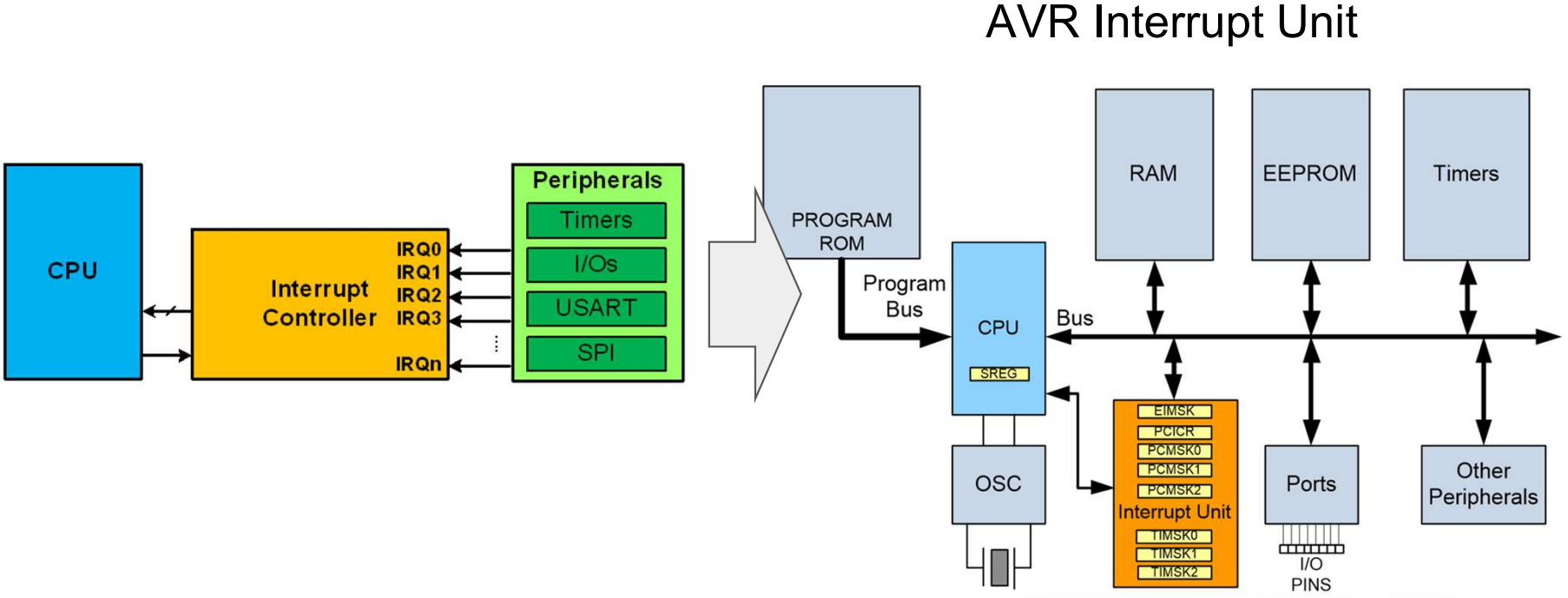
- Allow program to respond to events when they occur
- Allow program to ignore events until the occur
- External events e.g.:
  - UART ready with/for next character
  - Signal change on pin
- Internal events e.g.:
  - Power failure
  - Arithmetic exception
  - Timer “tick

COMPUTER ENGINEERING
MAHIDOL UNIVERSITY

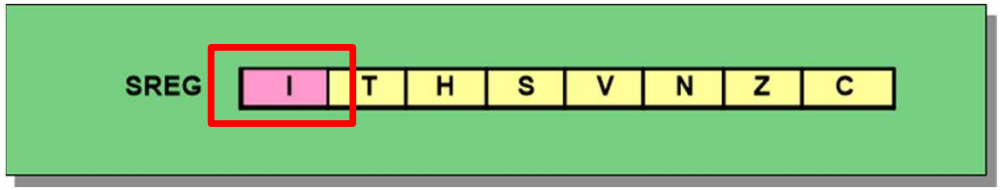
Interrupt Controllers

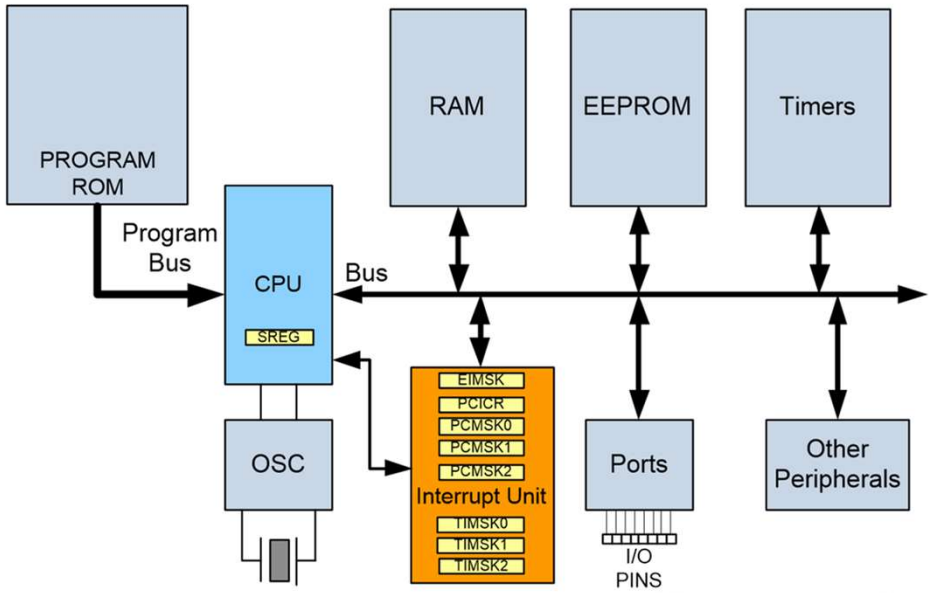


Interrupt Controllers



Interrupt Controllers





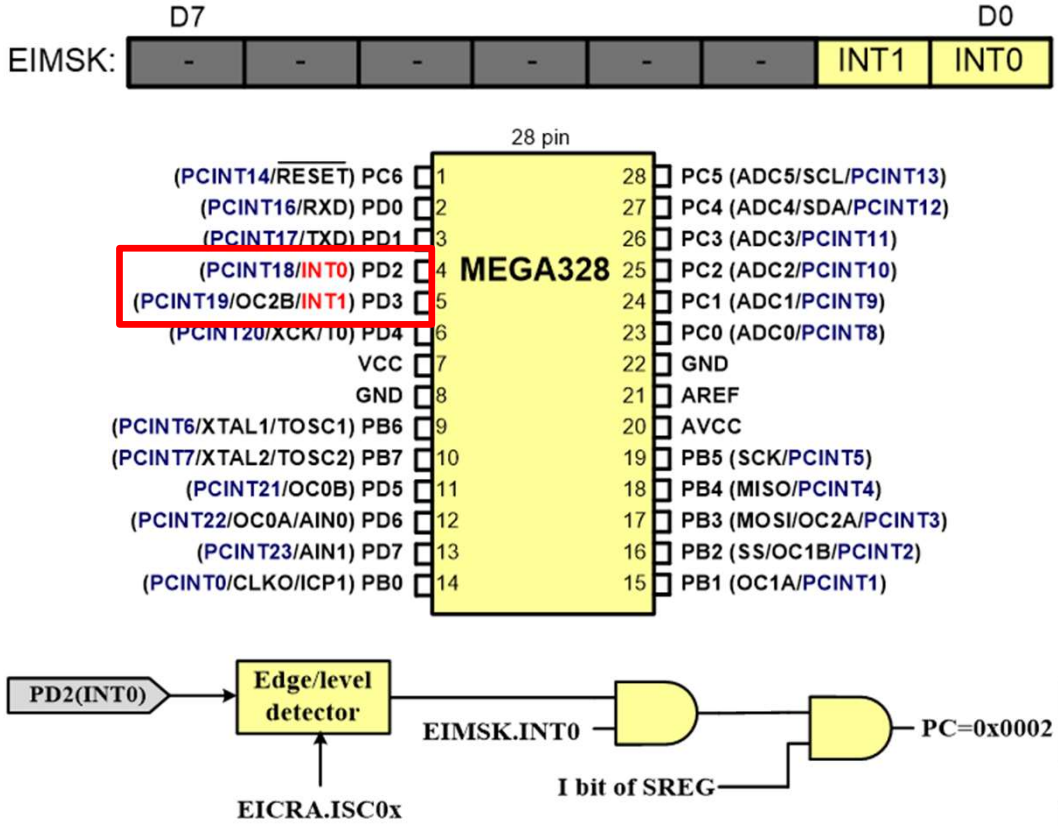
COMPUTER ENGINEERING MAHIDOL UNIVERSITY

AVR328P Interrupt Vector

Interrupt	Address (hex)
Reset	0000
External Interrupt Request 0	0002
External Interrupt Request 1	0004
Pin Change Interrupt Request 0	0006
Pin Change Interrupt Request 1	0008
Pin Change Interrupt Request 2	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

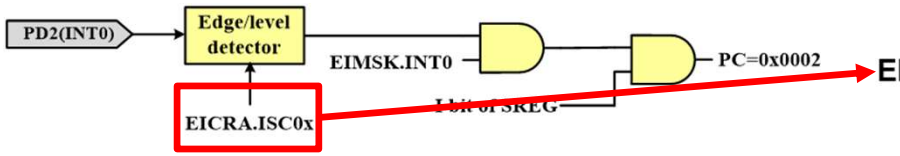
Interrupt	Address (hex)
Timer/Counter0 Overflow	0020
SPI Serial Transfer Complete	0022
USART Rx Complete	0024
USART Data Register Empty	0026
USART Tx Complete	0028
ADC Conversion Complete	002A
EEPROM ready	002C
Analog Comparator	002E

AVR328P External Interrupt

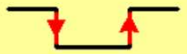


AVR328P External Interrupt

Edge trigger Vs. Level trigger in external interrupts




```
EICRA = 0x02;  //INT0 on falling edges
```

ISCx1	ISCx0	
0	0	
0	1	
1	0	
1	1	

AVR328P External Interrupt

External Interrupt Sample Code

Write a program that whenever INT0 goes low, it toggles PB5 once.

```
1  .ORG 0 ;location for reset
2    JMP MAIN
3  .ORG 0x02 ;loc. for external INT0
4    JMP EX0_ISR
5  MAIN:
6    LDI R20,HIGH(RAMEND)
7    OUT SPH,R20
8    LDI R20,LOW(RAMEND)
9    OUT SPL,R20 ;initialize stack
10   LDI R20,0x2 ;make INT0 falling
11   STS EICRA,R20
12   SBI DDRB,5 ;PORTB.5 = output
13   SBI PORTD,2 ;pull-up activated
14   LDI R20,1<<INT0 ;enable INT0
15   OUT EIMSK,R20
16   SEI ;enable interrupts
17 HERE:JMP HERE
```

```
18 EX0_ISR:
19   IN R21,PORTB
20   LDI R22,(1<<5) ;for toggling PB5
21   EOR R21,R22
22   OUT PORTB,R21
23   RETI
24
25
```

AVR328P External Interrupt

External Interrupt Sample Code

Write a program that whenever INT0 goes low, it toggles PB5 once.

```
1  .ORG 0 ;location for reset
2    JMP MAIN
3  .ORG 0x02 ;loc. for external INT0
4    JMP EX0_ISR
5  MAIN:
6    LDI R20,HIGH(RAMEND)
7    OUT SPH,R20
8    LDI R20,LOW(RAMEND)
9    OUT SPL,R20 ;initialize stack
10   LDI R20,0x2 ;make INT0 falling
11   STS EICRA,R20
12   SBI DDRB,5 ;PORTB.5 = output
13   SBI PORTD,2 ;pull-up activated
14   LDI R20,1<<INT0 ;enable INT0
15   OUT EIMSK,R20
16   SEI ;enable interrupts
17 HERE:JMP HERE
```

```
18 EX0_ISR:
19   IN R21,PORTB
20   LDI R22,(1<<5) ;for toggling PB5
21   EOR R21,R22
22   OUT PORTB,R21
23   RETI
24
25
```

AVR328P External Interrupt

External Interrupt Sample Code

Write a program that whenever INT0 goes low, it toggles PB5 once.

```
1  .ORG 0 ;location for reset
2    JMP MAIN
3  .ORG 0x02 ;loc. for external INT0
4    JMP EX0_ISR
5  MAIN:
6    LDI R20,HIGH(RAMEND)
7    OUT SPH,R20
8    LDI R20,LOW(RAMEND)
9    OUT SPL,R20 ;initialize stack
10   LDI R20,0x2 ;make INT0 falling
11   STS EICRA,R20
12   SBI DDRB,5 ;PORTB.5 = output
13   SBI PORTD,2 ;pull-up activated
14   LDI R20,1<<INT0 ;enable INT0
15   OUT EIMSK,R20
16   SEI ;enable interrupts
17 HERE:JMP HERE
18 EX0_ISR:
19   IN R21,PORTB
20   LDI R22,(1<<5) ;for toggling PB5
21   EOR R21,R22
22   OUT PORTB,R21
23   RETI
24
25
```

Ex. INT0. enable

AVR328P External Interrupt

External Interrupt Sample Code

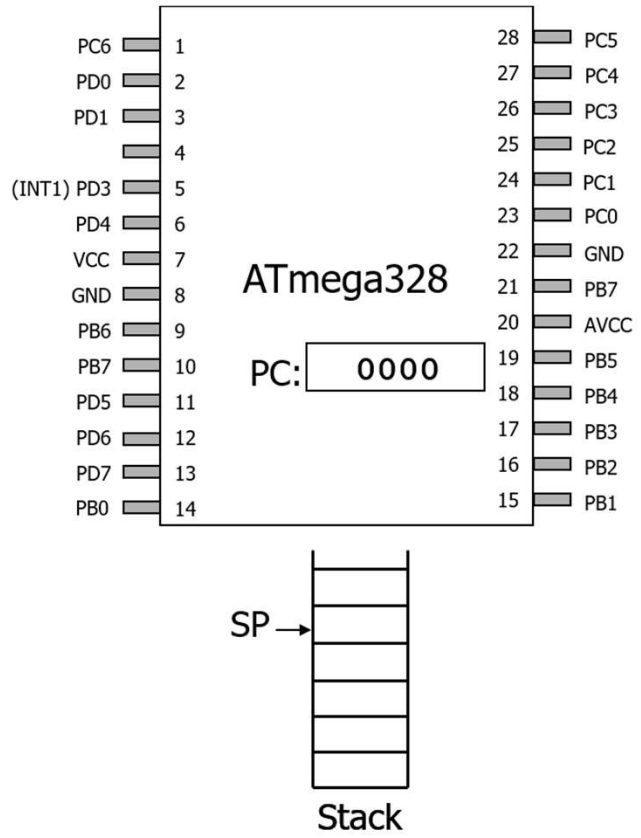
Write a program that whenever INT0 goes low, it toggles PB5 once.

```
1  .ORG 0 ;location for reset
2    JMP MAIN
3  .ORG 0x02 ;loc. for external INT0
4    JMP EX0_ISR
5  MAIN:
6    LDI R20,HIGH(RAMEND)
7    OUT SPH,R20
8    LDI R20,LOW(RAMEND)
9    OUT SPL,R20 ;initialize stack
10   LDI R20,0x2 ;make INT0 falling
11   STS EICRA,R20
12   SBI DDRB,5 ;PORTB.5 = output
13   SBI PORTD,2 ;pull-up activated
14   LDI R20,1<<INT0 ;enable INT0
15   OUT EIMSK,R20
16   SEI ;enable interrupts
17 HERE:JMP HERE
```

INT0 ISR

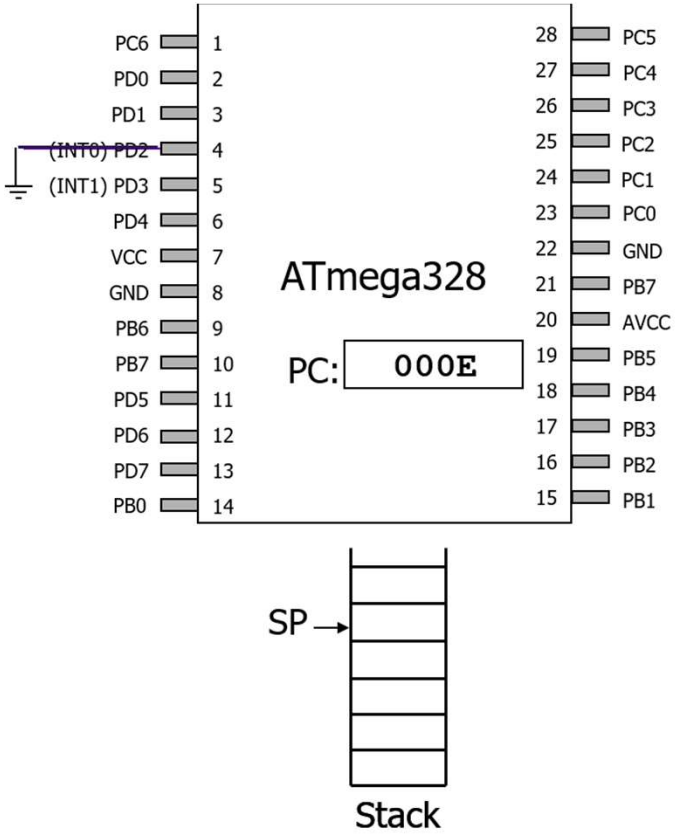
```
18 EX0_ISR:
19   IN R21,PORTB
20   LDI R22,(1<<5) ;for toggling PB5
21   EOR R21,R22
22   OUT PORTB,R21
23   RETI
24
25
```

AVR328P External Interrupt



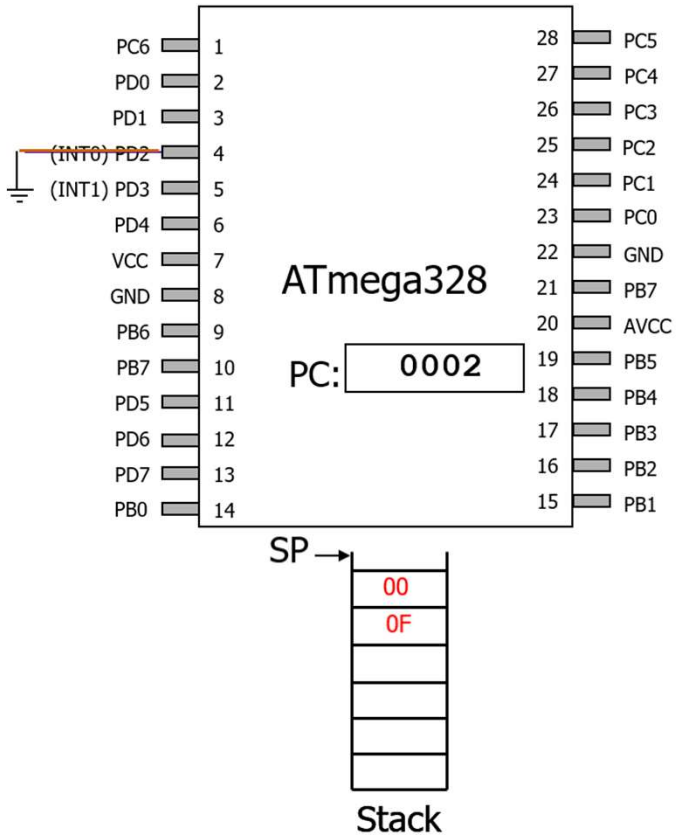
Address	Code
0000	`.ORG 0`
	`  JMP MAIN`
0002	`.ORG 0x02`
0002	`  JMP EX0_ISR`
0004	`MAIN: LDI R20,HIGH(RAMEND)`
0005	`  OUT SPH,R20`
0006	`  LDI R20,LOW(RAMEND)`
0007	`  OUT SPL,R20`
0008	`  LDI R20,0x2 ;make INT0 falling`
0009	`  STS EICRA,R20`
000A	`  SBI PORTD,2 ;pull-up activated`
000B	`  LDI R20,1<<INT0 ;enable INT0`
000C	`  OUT EIMSK,R20`
000D	`  SEI ;enable interrupts`
000E	`  SBI DDRB,5 ;PORTB.5 = output`
000F	`HERE: INC R25`
0010	`  JMP HERE`
	`EX0_ISR:`
0012	`  IN R21,PORTB`
0013	`  LDI R22,(1<<5) ;for toggling PB5`
0014	`  EOR R21,R22`
0015	`  OUT PORTB,R21`
0016	`  RETI`

AVR328P External Interrupt



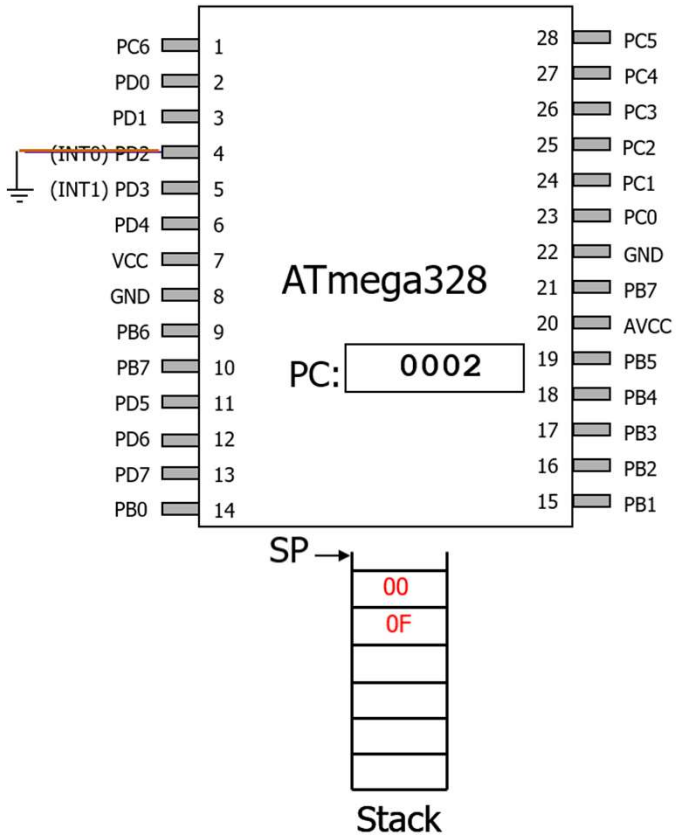
Address	Code
0000	`.ORG 0`
	`  JMP MAIN`
0002	`.ORG 0x02`
0002	`  JMP EX0_ISR`
0004	`MAIN: LDI R20,HIGH(RAMEND)`
0005	`  OUT SPH,R20`
0006	`  LDI R20,LOW(RAMEND)`
0007	`  OUT SPL,R20`
0008	`  LDI R20,0x2 ;make INT0 falling`
0009	`  STS EICRA,R20`
000A	`  SBI PORTD,2 ;pull-up activated`
000B	`  LDI R20,1<<INT0 ;enable INT0`
000C	`  OUT EIMSK,R20`
000D	`  SEI ;enable interrupts`
000E	`  SBI DDRB,5 ;PORTB.5 = output`
000F	`HERE: INC R25`
0010	`  JMP HERE`
	`EX0_ISR:`
0012	`  IN R21,PORTB`
0013	`  LDI R22,(1<<5) ;for toggling PB5`
0014	`  EOR R21,R22`
0015	`  OUT PORTB,R21`
0016	`  RETI`

AVR328P External Interrupt



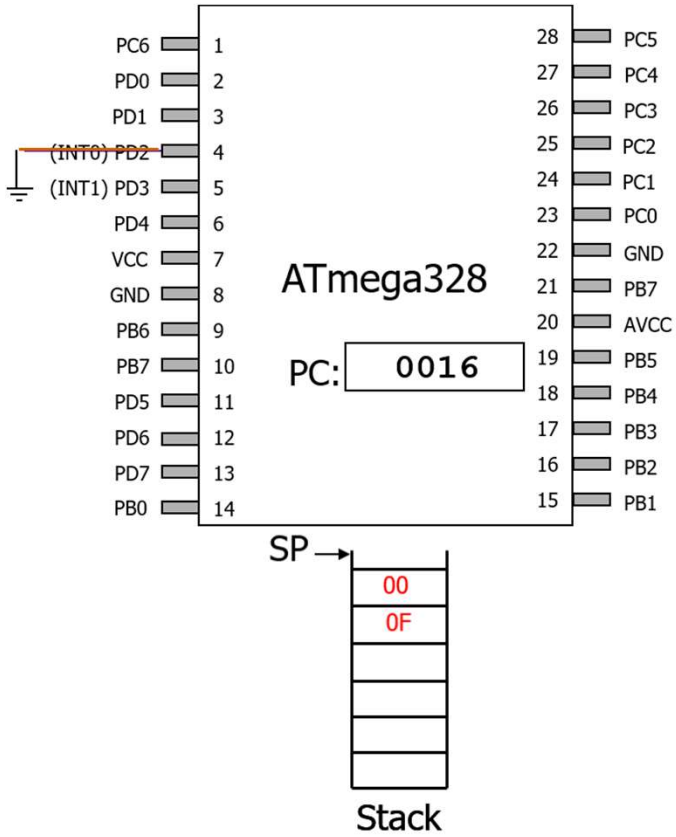
Address	Code
	`.ORG 0`
0000	`  JMP MAIN`
0002	`.ORG 0x02`
0002	`  JMP EX0_ISR`
0004	`MAIN: LDI R20,HIGH(RAMEND)`
0005	`  OUT SPH,R20`
0006	`  LDI R20,LOW(RAMEND)`
0007	`  OUT SPL,R20`
0008	`  LDI R20,0x2 ;make INT0 falling`
0009	`  STS EICRA,R20`
000A	`  SBI PORTD,2 ;pull-up activated`
000B	`  LDI R20,1<<INT0 ;enable INT0`
000C	`  OUT EIMSK,R20`
000D	`  SEI ;enable interrupts`
000E	`  SBI DDRB,5 ;PORTB.5 = output`
000F	`HERE: INC R25`
0010	`  JMP HERE`
	`EX0_ISR:`
0012	`  IN R21,PORTB`
0013	`  LDI R22,(1<<5) ;for toggling PB5`
0014	`  EOR R21,R22`
0015	`  OUT PORTB,R21`
0016	`  RETI`

AVR328P External Interrupt



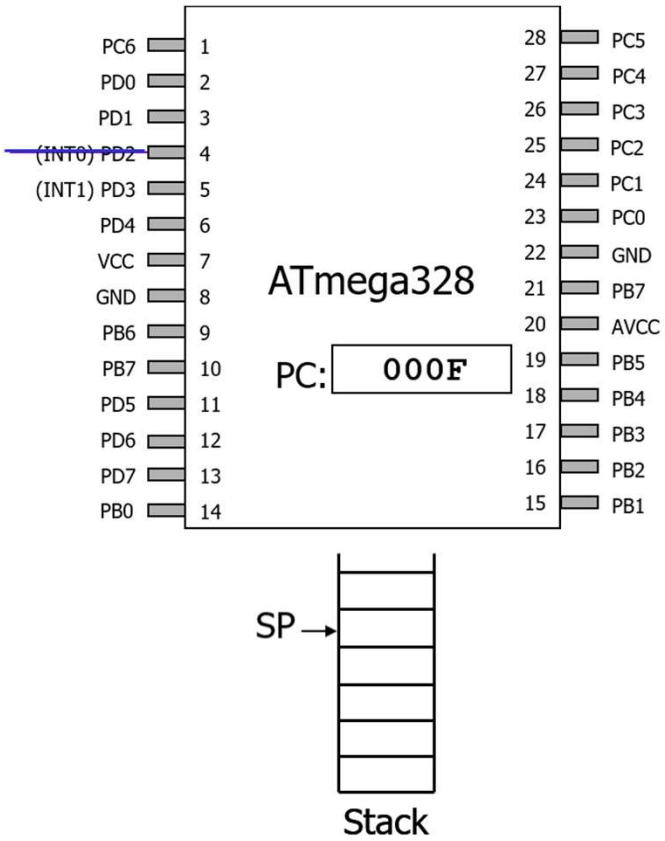
Address	Code
0000	`.ORG 0`
0002	`  JMP MAIN`
0002	`.ORG 0x02`
0002	`  JMP EX0_ISR`
0004	`MAIN: LDI R20,HIGH(RAMEND)`
0005	`  OUT SPH,R20`
0006	`  LDI R20,LOW(RAMEND)`
0007	`  OUT SPL,R20`
0008	`  LDI R20,0x2 ;make INT0 falling`
0009	`  STS EICRA,R20`
000A	`  SBI PORTD,2 ;pull-up activated`
000B	`  LDI R20,1<<INT0 ;enable INT0`
000C	`  OUT EIMSK,R20`
000D	`  SEI ;enable interrupts`
000E	`  SBI DDRB,5 ;PORTB.5 = output`
000F	`HERE: INC R25`
0010	`  JMP HERE`
	`EX0_ISR:`
0012	`  IN R21,PORTB`
0013	`  LDI R22,(1<<5) ;for toggling PB5`
0014	`  EOR R21,R22`
0015	`  OUT PORTB,R21`
0016	`  RETI`

AVR328P External Interrupt



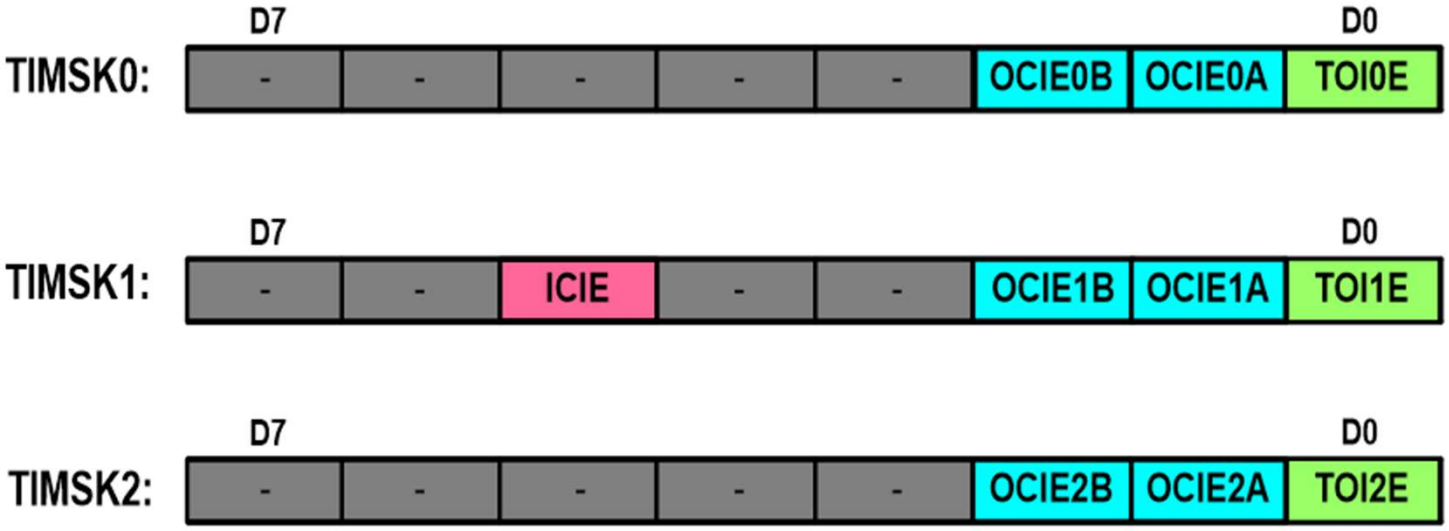
Address	Code
	`.ORG 0`
0000	`  JMP MAIN`
0002	`.ORG 0x02`
0002	`  JMP EX0_ISR`
0004	`MAIN: LDI R20,HIGH(RAMEND)`
0005	`  OUT SPH,R20`
0006	`  LDI R20,LOW(RAMEND)`
0007	`  OUT SPL,R20`
0008	`  LDI R20,0x2 ;make INT0 falling`
0009	`  STS EICRA,R20`
000A	`  SBI PORTD,2 ;pull-up activated`
000B	`  LDI R20,1<<INT0 ;enable INT0`
000C	`  OUT EIMSK,R20`
000D	`  SEI ;enable interrupts`
000E	`  SBI DDRB,5 ;PORTB.5 = output`
000F	`HERE: INC R25`
0010	`  JMP HERE`
	`EX0_ISR:`
0012	`  IN R21,PORTB`
0013	`  LDI R22,(1<<5) ;for toggling PB5`
0014	`  EOR R21,R22`
0015	`  OUT PORTB,R21`
0016	`  RETI`

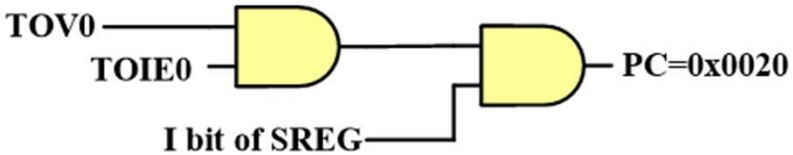
AVR328P External Interrupt



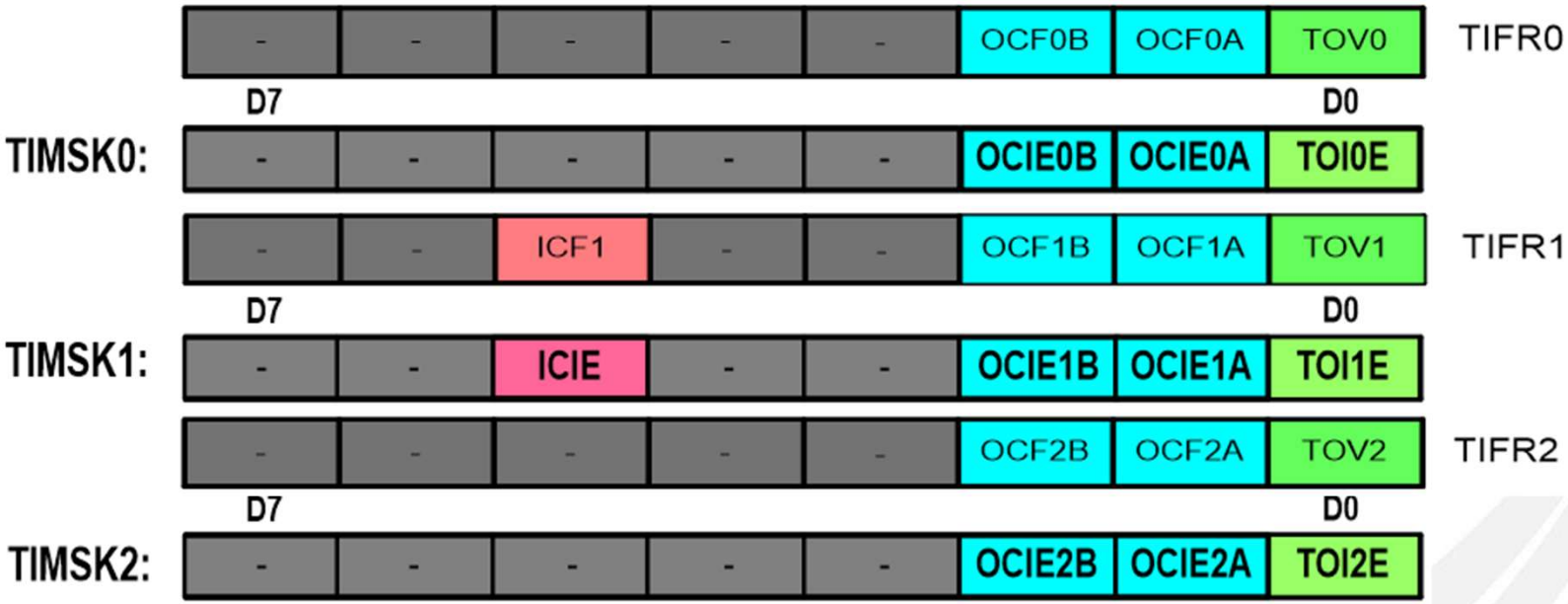
Address	Code
0000	`.ORG 0`
0000	`  JMP MAIN`
0002	`.ORG 0x02`
0002	`  JMP EX0_ISR`
0004	`MAIN: LDI R20,HIGH(RAMEND)`
0005	`  OUT SPH,R20`
0006	`  LDI R20,LOW(RAMEND)`
0007	`  OUT SPL,R20`
0008	`  LDI R20,0x2 ;make INT0 falling`
0009	`  STS EICRA,R20`
000A	`  SBI PORTD,2 ;pull-up activated`
000B	`  LDI R20,1<<INT0 ;enable INT0`
000C	`  OUT EIMSK,R20`
000D	`  SEI ;enable interrupts`
000E	`  SBI DDRB,5 ;PORTB.5 = output`
000F	`HERE: INC R25`
0010	`  JMP HERE`
	`EX0_ISR:`
0012	`  IN R21,PORTB`
0013	`  LDI R22,(1<<5) ;for toggling PB5`
0014	`  EOR R21,R22`
0015	`  OUT PORTB,R21`
0016	`  RETI`

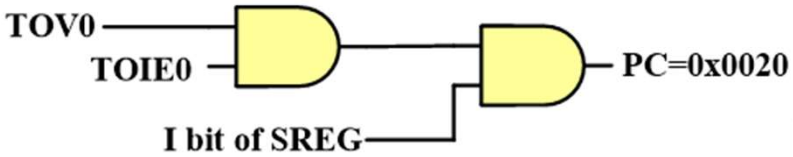
AVR328P Timer Interrupt*





AVR328P Timer Interrupt*





COMPUTER ENGINEERING MAHIDOL UNIVERSITY

AVR328P Timer Interrupt

Using Timer0 overflow interrupt

This program uses Timer0 to generate a square wave on pin PORTB.5, while at the same time data is being transferred from PORTC to PORTD

```
1  .ORG 0x0 ;location for reset
2  JMP MAIN
3  .ORG 0x20 ;location for Timer0 ov.
4    JMP T0_OV_ISR ;jump to ISR for T0
5  ;-main program for initialization
6  MAIN: LDI R20,HIGH(RAMEND)
7    OUT SPH,R20
8    LDI R20,LOW(RAMEND)
9    OUT SPL,R20 ;initialize stack
10   LDI R20,-32 ;value for 2 us
11   OUT TCNT0,R20 ;load Timer0
12   LDI R20,0x00
13   OUT TCCR0A,R20
14   LDI R20,0x01
15   OUT TCCR0B,R20 ;Normal, no scaler
16   LDI R20,(1<<TOIE0)
17   STS TIMSK0,R20 ;enable Timer0 Ov.
18   SEI ;set I (enable Ints)
19   SBI DDRB,5 ;PB5 as an output
20   LDI R20,0x00
21   OUT DDRC,R20 ;make PORTC input
22   LDI R20,0xFF
23   OUT PORTC,R20 ;enable pull-up
24   OUT DDRD,R20 ;make PORTD output
25 HERE:IN R20,PINC ;read from PORTC
26   OUT PORTD,R20 ;give it to PORTD
27   JMP HERE
28 T0_OV_ISR:
29   IN R16,PORTB ;read PORTB
30   LDI R17,(1<<5) ;for toggling PB5
31   EOR R16,R17
32   OUT PORTB,R16 ;toggle PB5
33   LDI R16,-32 ;timer value for 2 us
34   OUT TCNT0,R16;load Timer0 with -32
35   RETI  ;return from interrupt
```

AVR328P Timer Interrupt

Using Timer0 overflow interrupt

This program uses Timer0 to generate a square wave on pin PORTB.5, while at the same time data is being transferred from PORTC to PORTD

```
1  .ORG 0x0 ;location for reset
2  JMP MAIN
3  .ORG 0x20 ;location for Timer0 ov.
4    JMP T0_OV_ISR ;jump to ISR for T0
5  ;-main program for initialization
6  MAIN: LDI R20,HIGH(RAMEND)
7    OUT SPH,R20
8    LDI R20,LOW(RAMEND)
9    OUT SPL,R20 ;initialize stack
10   LDI R20,-32 ;value for 2 us
11   OUT TCNT0,R20 ;load Timer0
12   LDI R20,0x00
13   OUT TCCR0A,R20
14   LDI R20,0x01
15   OUT TCCR0B,R20 ;Normal, no scaler
16   LDI R20,(1<<TOIE0)
17   STS TIMSK0,R20 ;enable Timer0 Ov.
18   SEI ;set I (enable Ints)
19   SBI DDRB,5 ;PB5 as an output
20   LDI R20,0x00
21   OUT DDRC,R20 ;make PORTC input
22   LDI R20,0xFF
23   OUT PORTC,R20 ;enable pull-up
24   OUT DDRD,R20 ;make PORTD output
25 HERE:IN R20,PINC ;read from PORTC
26   OUT PORTD,R20 ;give it to PORTD
27   JMP HERE
28 T0_OV_ISR:
     N R16,PORTB ;read PORTB
     DI R17,(1<<5) ;for toggling PB5
     OR R16,R17
32   OUT PORTB,R16 ;toggle PB5
33   LDI R16,-32 ;timer value for 2 us
34   OUT TCNT0,R16;load Timer0 with -32
35   RETI  ;return from interrupt
```

Timer Init.

AVR328P Timer Interrupt

Using Timer0 overflow interrupt

This program uses Timer0 to generate a square wave on pin PORTB.5, while at the same time data is being transferred from PORTC to PORTD

```
1  .ORG 0x0 ;location for reset
2  JMP MAIN
3  .ORG 0x20 ;location for Timer0 ov.
4    JMP T0_OV_ISR ;jump to ISR for T0
5  ;-main program for initialization
6  MAIN: LDI R20,HIGH(RAMEND)
7    OUT SPH,R20
8    LDI R20,LOW(RAMEND)
9    OUT SPL,R20 ;initialize stack
10   LDI R20,-32 ;value for 2 us
11   OUT TCNT0,R20 ;load Timer0
12   LDI R20,0x00
13   OUT TCCR0A,R20
14   LDI R20,0x01
15   OUT TCCR0B,R20 ;Normal, no scaler
16   LDI R20,(1<<TOIE0)
17   STS TIMSK0,R20 ;enable Timer0 Ov.
18   SEI ;set I (enable Ints)
19   SBI DDRB,5 ;PB5 as an output
20   LDI R20,0x00
21   OUT DDRC,R20 ;make PORTC input
22   LDI R20,0xFF
23   OUT PORTC,R20 ;enable pull-up
24   OUT DDRD,R20 ;make PORTD output
25 HERE:IN R20,PINC ;read from PORTC
26   OUT PORTD,R20 ;give it to PORTD
27   JMP HERE
28 T0_OV_ISR:
29   IN R16,PORTB ;read PORTB
30   LDI R17,(1<<5) ;for toggling PB5
31   EOR R16,R17
32   OUT PORTB,R16 ;toggle PB5
33   LDI R16,-32 ;timer value for 2 us
       CNT0,R16;load Timer0 with -32
       ;return from interrupt
```

Timer Int. enable

COMPUTER ENGINEERING MAHIDOL UNIVERSITY

AVR328P Timer Interrupt

Using Timer0 overflow interrupt

This program uses Timer0 to generate a square wave on pin PORTB.5, while at the same time data is being transferred from PORTC to PORTD

```
1  .ORG 0x0 ;location for reset
2  JMP MAIN
3  .ORG 0x20 ;location for Timer0 ov.
4    JMP T0_OV_ISR ;jump to ISR for T0
5  ;-main program for initialization
6  MAIN: LDI R20,HIGH(RAMEND)
7    OUT SPH,R20
8    LDI R20,LOW(RAMEND)
9    OUT SPL,R20 ;initialize stack
10   LDI R20,-32 ;value for 2 us
11   OUT TCNT0,R20 ;load Timer0
12   LDI R20,0x00
13   OUT TCCR0A,R20
14   LDI R20,0x01
15   OUT TCCR0B,R20 ;Normal, no scaler
16   LDI R20,(1<<TOIE0)
17   STS TIMSK0,R20 ;enable Timer0 Ov.
18   SEI ;set I (enable Ints)
19   SBI DDRB,5 ;PB5 as an output
20   LDI R20,0x00
21   OUT DDRC,R20 ;make PORTC input
22   LDI R20,0xFF
23   OUT PORTC,R20 ;enable pull-up
24   OUT DDRD,R20 ;make PORTD output
25 HERE:IN R20,PINC ;read from PORTC
26   OUT PORTD,R20 ;give it to PORTD
27   JMP HERE
28 T0_OV_ISR:
     IN R16,PORTB ;read PORTB
     LDI R17,(1<<5) ;for toggling PB5
     EOR R16,R17
     OUT PORTB,R16 ;toggle PB5
     LDI R16,-32 ;timer value for 2 us
34   OUT TCNT0,R16;load Timer0 with -32
35   RETI  ;return from interrupt
```

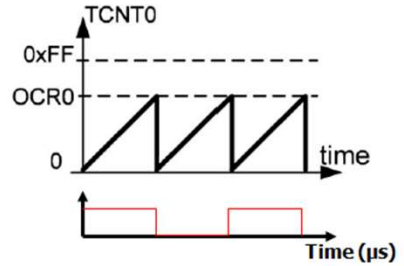
Timer0 ISR

AVR328P Timer Interrupt

Timer0 Compare Match Interrupt

using Timer0 and CTC mode generate a square wave on pin PORTB.5, while at the same time data is being transferred from PORTC to PORTD.

```
.ORG 0x0 ;location for reset
  JMP MAIN
.ORG 0x1C ;ISR location for Timer0 compare match A
  JMP T0_CM_ISR
;main program
MAIN:
  LDI R20,HIGH(RAMEND)
  OUT SPH,R20
  LDI R20,LOW(RAMEND)
  OUT SPL,R20;set up stack
  SBI DDRB,5;PB5 as an output
  LDI R20,239
  OUT OCR0A,R20 ;load Timer0 with 239
  LDI R20,(1<<WGM01)
  OUT TCCR0A,R20
  LDI R20,0x01
  OUT TCCR0B,R20 ;start Timer0, CTC mode, no scaler
  LDI R20,(1<<OCIE0A)
  STS TIMSK0,R20 ;enable Timer0 compare match interrupt
  SEI ;set I (enable interrupts globally)
  LDI R20,0x00
  OUT DDRC,R20 ;make PORTC input
  LDI R20,0xFF
  OUT DDRD,R20 ;make PORTD output
```




```
;--------------- Infinite loop
HERE:
  IN R20,PINC   ;read from PORTC
  OUT PORTD,R20 ;and send it to PORTD
  JMP HERE
;--ISR for Timer0 (executed every 40 µs)
T0_CM_ISR:
  IN R16,PORTB ;read PORTB
  LDI R17,1<<5 ;00100000 for toggling PB5
  EOR R16,R17
  OUT PORTB,R16 ;toggle PB5
  RETI ;return from interrupt
```

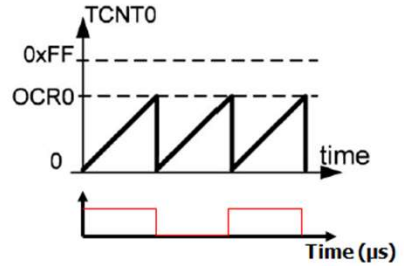
AVR328P Timer Interrupt

Timer0 Compare Match Interrupt

using Timer0 and CTC mode generate a square wave on pin PORTB.5, while at the same time data is being transferred from PORTC to PORTD.

```
.ORG 0x0 ;location for reset
  JMP MAIN
.ORG 0x1C ;ISR location for Timer0 compare match A
  JMP T0_CM_ISR
;main program
MAIN:
  LDI R20,HIGH(RAMEND)
  OUT SPH,R20
  LDI R20,LOW(RAMEND)
  OUT SPL,R20;set up stack
  SBI DDRB,5;PB5 as an output
  LDI R20,239
  OUT OCR0A,R20 ;load Timer0 with 239
  LDI R20,(1<<WGM01)
  OUT TCCR0A,R20
  LDI R20,0x01
  OUT TCCR0B,R20 ;start Timer0, CTC mode, no scaler
  LDI R20,(1<<OCIE0A)
  STS TIMSK0,R20 ;enable Timer0 compare match interrupt
  SEI ;set I (enable interrupts globally)
  LDI R20,0x00
  OUT DDRC,R20 ;make PORTC input
  LDI R20,0xFF
  OUT DDRD,R20 ;make PORTD output
```

Timer Init.




```
;--------------- Infinite loop
HERE:
  IN R20,PINC   ;read from PORTC
  OUT PORTD,R20 ;and send it to PORTD
  JMP HERE
;--ISR for Timer0 (executed every 40 µs)
T0_CM_ISR:
  IN R16,PORTB ;read PORTB
  LDI R17,1<<5 ;00100000 for toggling PB5
  EOR R16,R17
  OUT PORTB,R16 ;toggle PB5
  RETI ;return from interrupt
```

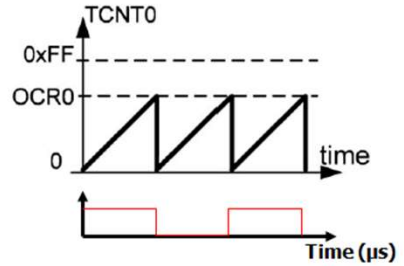
AVR328P Timer Interrupt

Timer0 Compare Match Interrupt

using Timer0 and CTC mode generate a square wave on pin PORTB.5, while at the same time data is being transferred from PORTC to PORTD.

```
.ORG 0x0 ;location for reset
  JMP MAIN
.ORG 0x1C ;ISR location for Timer0 compare match A
  JMP T0_CM_ISR
;main program
MAIN:
  LDI R20,HIGH(RAMEND)
  OUT SPH,R20
  LDI R20,LOW(RAMEND)
  OUT SPL,R20;set up stack
  SBI DDRB,5;PB5 as an output
  LDI R20,239
  OUT OCR0A,R20 ;load Timer0 with 239
  LDI R20,(1<<WGM01)
  OUT TCCR0A,R20
  LDI R20,0x01
  OUT TCCR0B,R20 ;start Timer0, CTC mode, no scaler
  LDI R20,(1<<OCIE0A)
  STS TIMSK0,R20 ;enable Timer0 compare match interrupt
  SEI ;set I (enable interrupts globally)
  LDI R20,0x00
  OUT DDRC,R20 ;make PORTC input
  LDI R20,0xFF
  OUT DDRD,R20 ;make PORTD output
```

Timer Int. enable



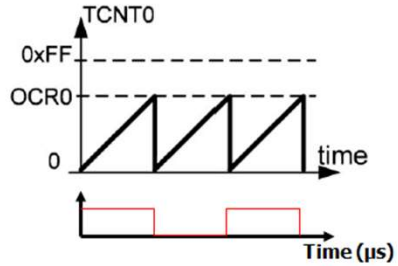

```
;--------------- Infinite loop
HERE:
  IN R20,PINC   ;read from PORTC
  OUT PORTD,R20 ;and send it to PORTD
  JMP HERE
;--ISR for Timer0 (executed every 40 µs)

  ORTB ;read PORTB
  L<<5 ;00100000 for toggling PB5
  R17
  OUT PORTB,R16 ;toggle PB5
  RETI ;return from interrupt
```

AVR328P Timer Interrupt

Timer0 Compare Match Interrupt

using Timer0 and CTC mode generate a square wave on pin PORTB.5, while at the same time data is being transferred from PORTC to PORTD.

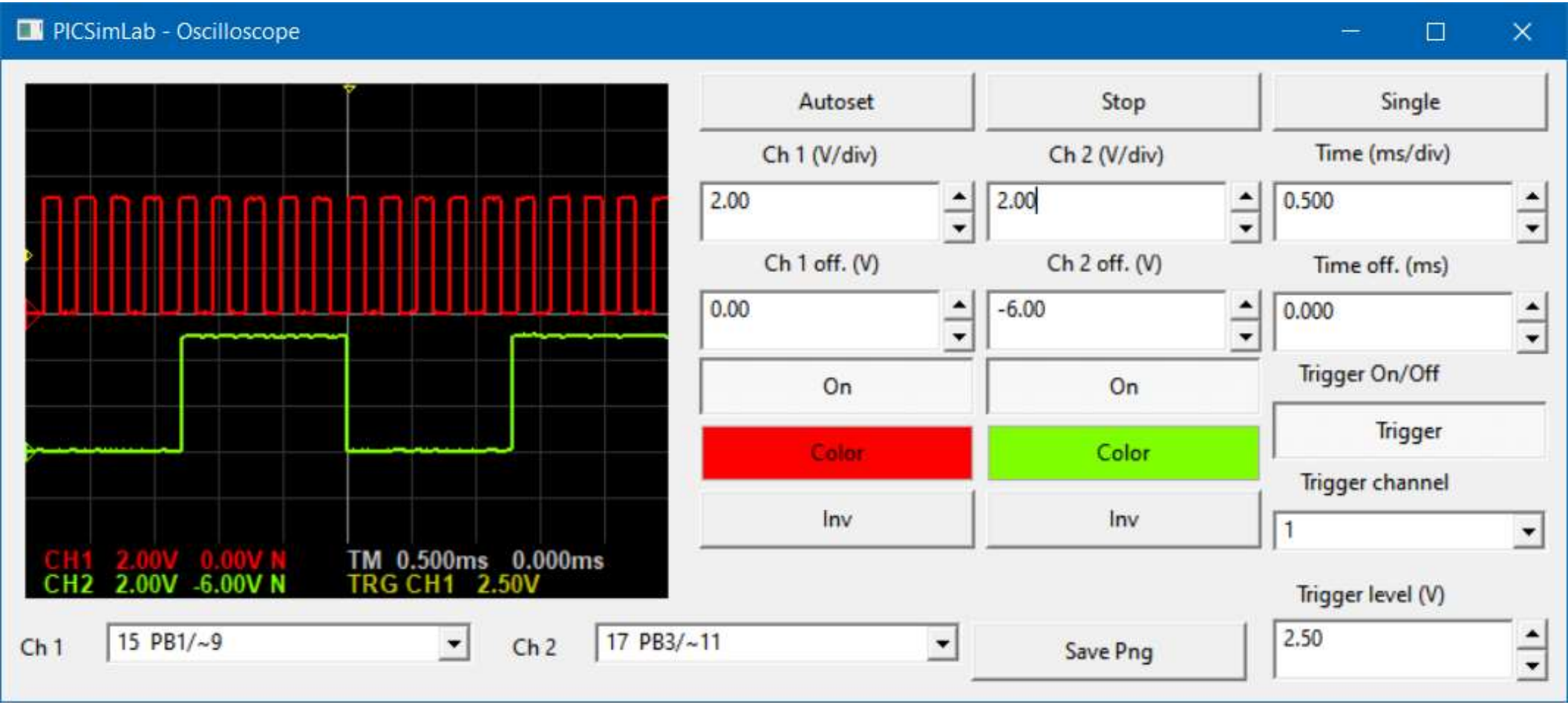
```
.ORG 0x0 ;location for reset
  JMP MAIN
.ORG 0x1C ;ISR location for Timer0 compare match A
  JMP T0_CM_ISR
;main program
MAIN:
  LDI R20,HIGH(RAMEND)
  OUT SPH,R20
  LDI R20,LOW(RAMEND)
  OUT SPL,R20;set up stack
  SBI DDRB,5;PB5 as an output
  LDI R20,239
  OUT OCR0A,R20 ;load Timer0 with 239
  LDI R20,(1<<WGM01)
  OUT TCCR0A,R20
  LDI R20,0x01
  OUT TCCR0B,R20 ;start Timer0, CTC mode, no
  LDI R20,(1<<OCIE0A)
  STS TIMSK0,R20 ;enable Timer0 compare matc
  SEI ;set I (enable interrupts globally)
  LDI R20,0x00
  OUT DDRC,R20 ;make PORTC input
  LDI R20,0xFF
  OUT DDRD,R20 ;make PORTD output
```




```
;--------------- Infinite loop
HERE:
  IN R20,PINC   ;read from PORTC
  OUT PORTD,R20 ;and send it to PORTD
  JMP HERE
```

Timer0 ISR

```
;--ISR for Timer0 (executed every 40 µs)
T0_CM_ISR:
  IN R16,PORTB ;read PORTB
  LDI R17,1<<5 ;00100000 for toggling PB5
  EOR R16,R17
  OUT PORTB,R16 ;toggle PB5
  RETI ;return from interrupt
```


PICSimLab - Oscilloscope
CH1 2.00V 0.00V N
CH2 2.00V -6.00V N
TM 0.500ms 0.000ms
TRG CH1 2.50V
Autoset
Stop
Single
Ch 1 (V/div)
Ch 2 (V/div)
Time (ms/div)
2.00
2.00
0.500
Ch 1 off. (V)
Ch 2 off. (V)
Time off. (ms)
0.00
-6.00
0.000
On
On
Trigger On/Off
Color
Color
Trigger
Inv
Inv
Trigger channel
1
Ch 1
15 PB1/~9
Ch 2
17 PB3/~11
Save Png
Trigger level (V)
2.50

COMPUTER ENGINEERING
MAHIDOL UNIVERSITY

Pin Change Interrupts

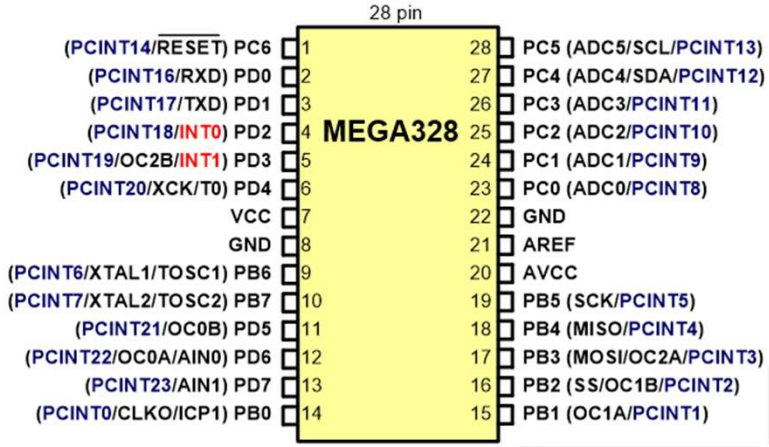
External interrupts Vs. Pin Change Interrupts

- Both are interrupts signal which is received from external to the chip
- External Interrupts are limited to only a couple pins (INT0, INT1)
- Pin Change interrupts can occur on all input pins
- On the ATMEGA328P, there are 2 External Interrupts but 24 Pin Change Interrupts.

Pin Change Interrupts

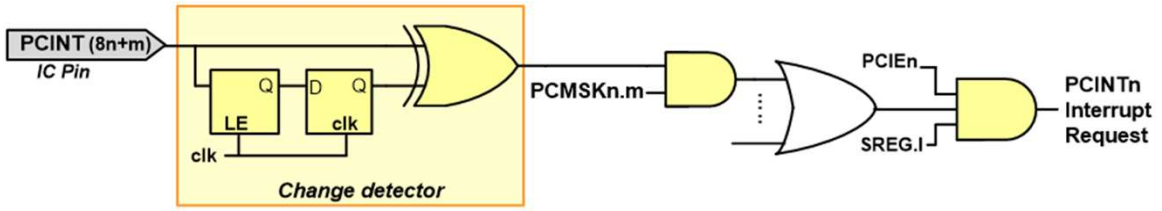
Pin Change Interrupt Registers

Bit	D7						D0	
PCMSK0: (for PORTB)	PCINT7	PCINT6	PCINT5	PCINT4	PCINT3	PCINT2	PCINT1	PCINT0	0x6B
PCMSK1: (for PORTC)	PCINT15	PCINT14	PCINT13	PCINT12	PCINT11	PCINT10	PCINT9	PCINT8	0x6C
PCMSK2: (for PORTD)	PCINT23	PCINT22	PCINT21	PCINT20	PCINT19	PCINT18	PCINT17	PCINT16	0x6D



Pin Change Interrupts

Pin Change Interrupt Registers



Bit	D7							D0	
PCICR:	-	-	-	-	-	PCIE2	PCIE1	PCIE0	0x68

PCIE0: Pin Change Interrupt Enable bit for PORTB
PCIE1: Pin Change Interrupt Enable bit for PORTC (0: disabled, 1: enabled)
PCIE2: Pin Change Interrupt Enable bit for PORTD (0: disabled, 1: enabled)

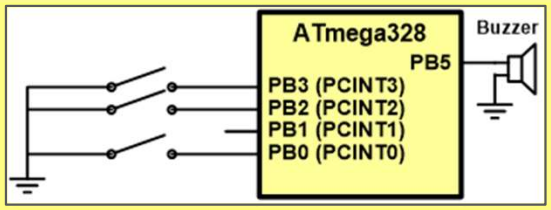
Bit	D7							D0	
PCMSK0: (for PORTB)	PCINT7	PCINT6	PCINT5	PCINT4	PCINT3	PCINT2	PCINT1	PCINT0	0x6B
PCMSK1: (for PORTC)	PCINT15	PCINT14	PCINT13	PCINT12	PCINT11	PCINT10	PCINT9	PCINT8	0x6C
PCMSK2: (for PORTD)	PCINT23	PCINT22	PCINT21	PCINT20	PCINT19	PCINT18	PCINT17	PCINT16	0x6D

Pin Change Interrupts

Pin Change Int. Sample Code

The PB0, PB2, and PB3 pins are connected to switches. Write a program that makes PORTB.5 high whenever any of the switches changes state.

```
1  .ORG 0  ;location for reset
2          JMP     MAIN
3  .ORG 0x06 ;loc. for PCINT0
4          JMP     PCINT0_ISR
5  MAIN:   LDI     R20,HIGH(RAMEND)
6          OUT     SPH,R20
7          LDI     R20,LOW(RAMEND)
8          OUT     SPL,R20
9          SBI     DDRB,5
10         CBI     PORTB,5
11   ;enable pull-up resistors
12         OUT     PORTB,R20
13 ;enable PB0, PB2, and PB3 PCINTs
14         LDI     R20,0b00001101
15         STS     PCMSK0,R20
16   ;enable PORTB change interrupt
17         LDI     R20,(1<<PCIE0)
18         STS     PCICR,R20
19         SEI
20 HERE:   JMP     HERE
21
22 PCINT0_ISR:
23         SBI     PORTB,5
24         RETI
25
26
27
28
29
30
```



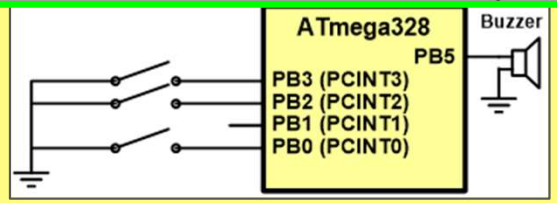
Pin Change Interrupts

Pin Change Int. Sample Code

The PB0, PB2, and PB3 pins are connected to switches. Write a program that makes PORTB.5 high whenever any of the switches changes state.

```
1  .ORG 0  ;location for reset
2          JMP     MAIN
3  .ORG 0x06 ;loc. for PCINT0
4          JMP     PCINT0_ISR
5  MAIN:   LDI     R20,HIGH(RAMEND)
6          OUT     SPH,R20
7          LDI     R20,LOW(RAMEND)
8          OUT     SPL,R2
9          SBI     DDRB,5
10         CBI     PORTB,
11  ;enable pull-up resistors
12         OUT     PORTB,R20
13 ;enable PB0, PB2, and PB3 PCINTs
14         LDI     R20,0b00001101
15         STS     PCMSK0,R20
16 ;enable PORTB change interrupt
17         LDI     R20,(1<<PCIE0)
18         STS     PCICR,R20
19         SEI
20 HERE:   JMP     HERE
21
22 PCINT0_ISR:
27
28
29
30
```

Pin Change Interrupt Request 0	0006
Pin Change Interrupt Request 1	0008
Pin Change Interrupt Request 2	000A



COMPUTER ENGINEERING MAHIDOL UNIVERSITY

Interrupts Priority

Interrupt	Address (hex)
Reset	0000
External Interrupt Request 0	0002
External Interrupt Request 1	0004
Pin Change Interrupt Request 0	0006
Pin Change Interrupt Request 1	0008
Pin Change Interrupt Request 2	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

Highest priority

Lowest priority

Interrupt inside an interrupt

- The I flag is cleared when the AVR begins to execute an ISR. So, interrupts are disabled.
- The I flag is set when RETI is executed.

COMPUTER ENGINEERING
MAHIDOL UNIVERSITY

Resource conflict

Does this program correctly work?

```
1  .ORG     0x0          ;location for reset
2           JMP          MAIN
3  .ORG     0x1C         ;Timer0 compare match
4           JMP          T0_CM_ISR
5  ;---------main program----------------------------
6  MAIN:    LDI          R20,HIGH(RAMEND)
7           OUT          SPH,R20
8           LDI          R20,LOW(RAMEND)
9           OUT          SPL,R20  ;set up stack
10          LDI          R20,159
11          OUT          OCR0A,R20
12          LDI          R20,(1<<WGM01)
13          OUT          TCCR0A,R20
14          LDI          R20,0x01
15          OUT          TCCR0B,R20
16          LDI          R20,(1<<OCIE0A)
17          STS          TIMSK0,R20
18          SEI
19          LDI          R20,0xFF
20          OUT          DDRC,R20
21          OUT          DDRD,R20
22 HERE:    OUT          PORTC,R20
23          INC          R20
24          JMP          HERE
25 ;-------------------------ISR  for Timer0
26 T0_CM_ISR:
27          IN           R20,PIND
28          INC          R20
29          OUT          PORTD,R20
30          RETI
```

Resource conflict

Does this program correctly work?

```
1  .ORG     0x0          ;location for reset
2           JMP          MAIN
3  .ORG     0x1C         ;Timer0 compare match
4           JMP          T0_CM_ISR
5  ;---------main program-----------------------------
6  MAIN:    LDI          R20,HIGH(RAMEND)
7           OUT          SPH,R20
8           LDI          R20,LOW(RAMEND)
9           OUT          SPL,R20  ;set up stack
10          LDI          R20,159
11          OUT          OCR0A,R20
12          LDI          R20,(1<<WGM01)
13          OUT          TCCR0A,R20
14          LDI          R20,0x01
15          OUT          TCCR0B,R20
16          LDI          R20,(1<<OCIE0A)
17          STS          TIMSK0,R20
18          SEI
19          LDI          R20,0xFF
20          OUT          DDRC,R20
21          OUT          DDRD,R20
22 HERE:    OUT          PORTC,R20
23          INC          R20
24          JMP          HERE
25 ;-------------------------ISR for Timer0
26 T0_CM_ISR:
27          IN           R20,PIND
28          INC          R20
29          OUT          PORTD,R20
30          RETI
```

Resource conflict

Solution 1: Use different registers for different tasks.

```
1  .ORG    0x0       ;location for reset
2          JMP       MAIN
3  .ORG    0x1C      ;Timer0 compare match
4          JMP       T0_CM_ISR
5  ;---------main program-----------------------------
6  MAIN:   LDI       R20,HIGH(RAMEND)
7          OUT       SPH,R20
8          LDI       R20,LOW(RAMEND)
9          OUT       SPL,R20  ;set up stack
10         LDI       R20,159
11         OUT       OCR0A,R20
12         LDI       R20,(1<<WGM01)
13         OUT       TCCR0A,R20
14         LDI       R20,0x01
15         OUT       TCCR0B,R20
16         LDI       R20,(1<<OCIE0A)
17         STS       TIMSK0,R20
```

```
18         SEI
19         LDI       R20,0xFF
20         OUT       DDRC,R20
21         OUT       DDRD,R20
22
23         LDI       R20, 0
24 HERE:   OUT       PORTC,R20
25         INC       R20
26         JMP       HERE
27 ;-------------------------ISR  for Timer0
28 T0_CM_ISR:
29         IN        R21,PIND
30         INC       R21
31         OUT       PORTD,R21
32         RETI
```

Resource conflict

Solution 2: Save the contents of registers on the stack before execution of each task, and reload the registers at the end of the task.

```
1  .ORG    0x0         ;location for reset
2          JMP         MAIN
3  .ORG    0x1C        ;Timer0 compare match
4          JMP         T0_CM_ISR
5  ;---------main program----------------------------
6  MAIN:   LDI         R20,HIGH(RAMEND)
7          OUT         SPH,R20
8          LDI         R20,LOW(RAMEND)
9          OUT         SPL,R20  ;set up stack
10         LDI         R20,159
11         OUT         OCR0A,R20
12         LDI         R20,(1<<WGM01)
13         OUT         TCCR0A,R20
14         LDI         R20,0x01
15         OUT         TCCR0B,R20
16         LDI         R20,(1<<OCIE0A)
17         STS         TIMSK0,R20
18         SEI
19         LDI         R20,0xFF
20         OUT         DDRC,R20
21         OUT         DDRD,R20
22         LDI         R20, 0
23 HERE:   OUT         PORTC,R20
24         INC         R20
25         JMP         HERE
26 ;-------------------------ISR  for Timer0
27 T0_CM_ISR:
28         PUSH        R20      ;save R20
29         IN          R20,PIND
30         INC         R20
31         OUT         PORTD,R20
32         POP         R20      ;restore R20
33         RETI
34
```

Resource conflict

Solution 3: Make a stack for yourself and use it to save the contents of registers

```
1  .ORG    0x0         ;location for reset
2          JMP         MAIN
3  .ORG    0x1C        ;Timer0 compare match
4          JMP         T0_CM_ISR
5  ;---------main program-----------------------------
6  MAIN:   LDI         R20,HIGH(RAMEND)
7          OUT         SPH,R20
8          LDI         R20,LOW(RAMEND)
9          OUT         SPL,R20  ;set up stack
10         LDI         YH, HIGH($100)
11         LDI         YL, LOW($100)
12
13         LDI         R20,159
14         OUT         OCR0A,R20
15         LDI         R20,(1<<WGM01)
16         OUT         TCCR0A,R20
17         LDI         R20,0x01
18         OUT         TCCR0B,R20
19         LDI         R20,(1<<OCIE0A)
20         STS         TIMSK0,R20
21         SEI
22         LDI         R20,0xFF
23         OUT         DDRC,R20
24         OUT         DDRD,R20
25         LDI         R20, 0
26 HERE:   OUT         PORTC,R20
27         INC         R20
28         JMP         HERE
29 ;-------------------------ISR  for Timer0
30 T0_CM_ISR:
31         ST          Y+,R20   ;save R20
32         IN          R20,PIND
33         INC         R20
34         OUT         PORTD,R20
35         LD          R20, -Y  ;restore R20
36         RETI
```

Resource conflict

***** We should save SREG, when we change flags in the ISR. *****

```
PUSH    R20
IN      R20,SREG
PUSH    R20
...
POP     R20
OUT     SREG,R20
POP     R20
```

COMPUTER ENGINEERING MAHIDOL UNIVERSITY

Interrupt in C

Using Timer0 generate a square wave on pin PORTB.5, while at the same time transferring data from PORTC to PORTD.

```
#include "avr/io.h"
#include "avr/interrupt.h"
int main ()
{
    DDRB |= (1<<5);     //DDRB.5 = output
    TCNT0 = -32;        //timer value for 2 µs
    TCCR0A = 0x00;
    TCCR0B = 0x01;              //Normal mode, int clk, no prescaler
    TIMSK0 = (1<<TOIE0);        //enable Timer0 overflow interrupt
    sei ();                     //enable interrupts
    DDRC = 0x00;                //make PORTC input
    DDRD = 0xFF;                //make PORTD output
    while (1)           //wait here
        PORTD = PINC;
}
ISR (TIMER0_OVF_vect)  //ISR for Timer0 overflow
{
    TCNT0 = -32;
    PORTB ^= 0x20;      //toggle PORTB.5
}
```

COMPUTER ENGINEERING
MAHIDOL UNIVERSITY

Interrupt in C

Using Timer0 generate a square wave on pin PORTB.5, while at the same time transferring data from PORTC to PORTD.

```
#include "avr/io.h"
#include "avr/interrupt.h"
int main ()
{
    DDRB |= (1<<5);     //DDRB.5 = output
    TCNT0 = -32;        //timer value for 2 µs
    TCCR0A = 0x00;
    TCCR0B = 0x01;              //Normal mode, int clk, no prescaler
    TIMSK0 = (1<<TOIE0);        //enable Timer0 overflow interrupt
    sei ();
    DDRC = 0x00;
    DDRD = 0xFF;
    while (1)           //wait h
        PORTD = PINC;
}
ISR (TIMER0_OVF_vect)  //ISR for Timer0 overflow
{
    TCNT0 = -32;
    PORTB ^= 0x20;      //toggle PORTB.5
}
```

```
sei ( );      //set I
cli  ( );     //clear I
```

COMPUTER ENGINEERING MAHIDOL UNIVERSITY

Interrupt in C

Example 1 Using Timer1 and CTC mode write a program that toggles pin PORTB.5 every second, while at the same time transferring data from PORTC to PORTD. Assume XTAL = 16 MHz.

```
#include <avr/io.h>
#include <avr/interrupt.h>
int main () {
        DDRB |= (1<<5);         //make DDRB.5 output

        OCR1A = 15624;
        TCCR1A = 0x00; //CTC mode, internal clk, prescaler=1024
        TCCR1B = 0x0D;
        TIMSK1 = (1<<OCIE1A); //enable Timer1 compare match A int.
        sei ();               //enable interrupts

        DDRC = 0x00;          //make PORTC input
        DDRD = 0xFF;          //make PORTD output
        while (1)             //wait here
                PORTD = PINC;
}
ISR (TIMER1_COMPA_vect) {     //ISR for Timer1 compare match A
        PORTB ^= (1<<5);      //toggle PORTB.5
}
```

COMPUTER ENGINEERING MAHIDOL UNIVERSITY

Interrupt in C

Example 2 Assume that the INT0 pin is connected to a switch that is normally high. Write a program that toggles PORTB.5, whenever INT0 pin goes low.

```
#include <avr/io.h>
#include <avr/interrupt.h>
int main ()
{
        DDRB = 1<<5;            //PB5 as an output
        PORTD = 1<<2;           //pull-up activated
        EICRA = 0x2;            //make INT0 falling edge triggered

        EIMSK = (1<<INT0);      //enable external interrupt 0
        sei ();                 //enable interrupts

        while (1);                      //wait here
}

ISR (INT0_vect)                         //ISR for external interrupt 0
{
        PORTB ^= (1<<5);                //toggle PORTB.5
}
```

ATmega328P

Vector Number	Interrupt definition	Vector name
2	External Interrupt Request 0	`INT0_vect`
3	External Interrupt Request 1	`INT1_vect`
4	Pin Change Interrupt Request 0	`PCINT0_vect`
5	Pin Change Interrupt Request 1	`PCINT1_vect`
6	Pin Change Interrupt Request 2	`PCINT2_vect`
7	Watchdog Time-out Interrupt	`WDT_vect`
8	Timer/Counter2 Compare Match A	`TIMER2_COMPA_vect`
9	Timer/Counter2 Compare Match B	`TIMER2_COMPB_vect`
10	Timer/Counter2 Overflow	`TIMER2_OVF_vect`
11	Timer/Counter1 Capture Event	`TIMER1_CAPT_vect`
12	Timer/Counter1 Compare Match A	`TIMER1_COMPA_vect`
13	Timer/Counter1 Compare Match B	`TIMER1_COMPB_vect`
14	Timer/Counter1 Overflow	`TIMER1_OVF_vect`
15	Timer/Counter0 Compare Match A	`TIMER0_COMPA_vect`
16	Timer/Counter0 Compare Match B	`TIMER0_COMPB_vect`
17	Timer/Counter0 Overflow	`TIMER0_OVF_vect`
18	SPI Serial Transfer Complete	`SPI_STC_vect`
19	USART Rx Complete	`USART_RX_vect`
20	USART Data Register Empty	`USART_UDRE_vect`
21	USART Tx Complete	`USART_TX_vect`
22	ADC Conversion Complete	`ADC_vect`
23	EEPROM Ready	`EE_READY_vect`
24	Analog Comparator	`ANALOG_COMP_vect`
25	Two-wire Serial Interface	`TWI_vect`
26	Store Program Memory Read	`SPM_READY_vect`

Exercise 10

- Using both TIMER0 and TIMER1, write a program that generate two square waveforms simultaneously. One waveform is at pin PB1 and the other is at pin PB3.
- Assume a clock speed of 16 MHz.
- Use normal mode for both timers. This means they count from -x to 0.
- For the waveform at PB1, use TIMER0. The waveform should toggles from low to high or high to low every 10 microseconds. Calculate the starting value for TIMER0.
- For the waveform at PB3, use TIMER1. The waveform should toggles from low to high or high to low every 100 microseconds. Calculate the starting value for TIMER1.
- Starter code is given in *exercise10-1-template.asm*. Complete every line with the comment ;TODO
- Show the output of both waveforms at the same time in PicSimLab's oscilloscope.
- Expected output on the oscilloscope on the next slide. The green waveform is actually moving, this is not a problem.