Interrupt
Chapter 10

The AVR microcontroller
and embedded
systems
using assembly and c

Contents

- Polling Vs. interrupt
- Interrupt unit
- Steps in executing an interrupt
- Edge trigger Vs. Level trigger in external interrupts
- Timer interrupt
- Interrupt priority
- Interrupt inside an interrupt
- Task switching and resource conflict

Polling Vs. Interrupt

- 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

Interrupt unit

Interrupt Vectors in ATmega128

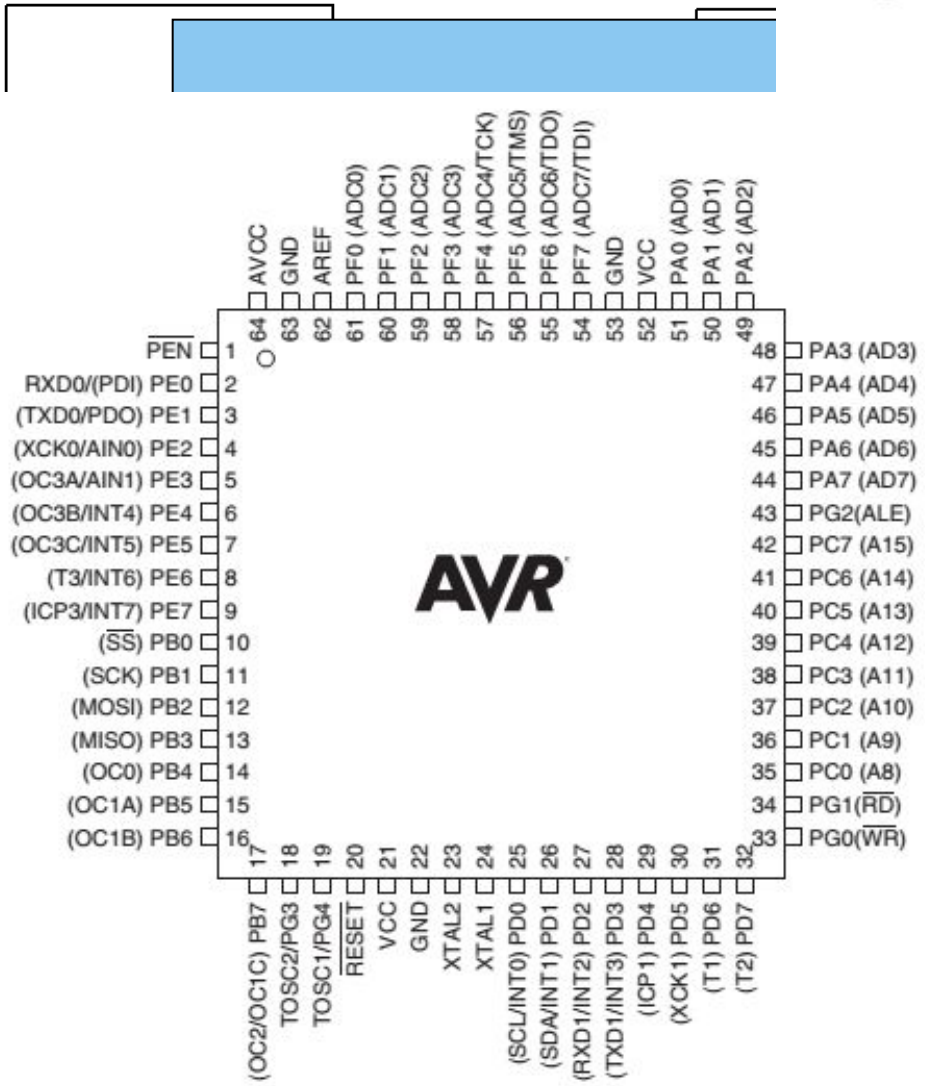
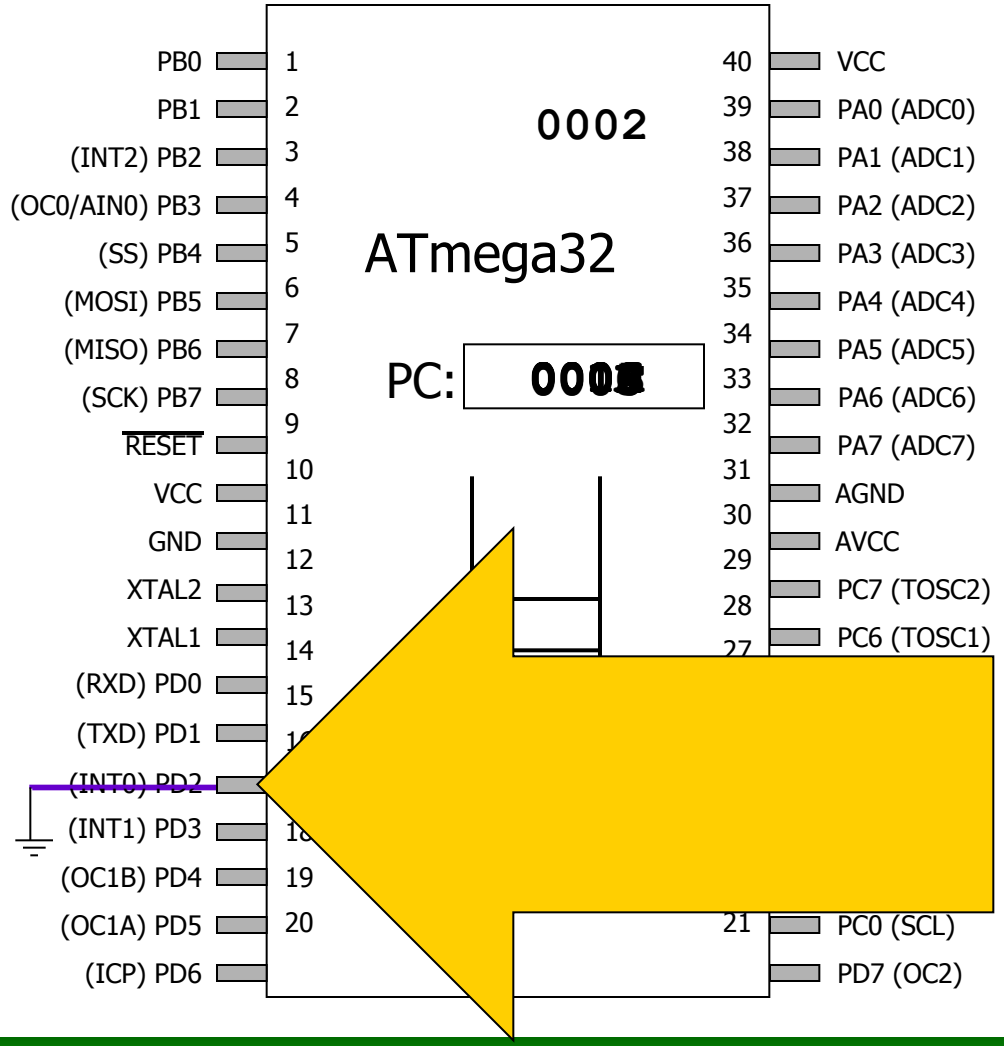


Table 23. Reset and Interrupt Vectors

Vector No.	Program Address(2)	Source	Interrupt Definition
1	$0000(1)	RESET	External Pin, Power-on Reset, Brown-out Reset, Watchdog Reset, and JTAG AVR Reset
2	$0002	INT0	External Interrupt Request 0
3	$0004	INT1	External Interrupt Request 1
4	$0006	INT2	External Interrupt Request 2
5	$0008	INT3	External Interrupt Request 3
6	$000A	INT4	External Interrupt Request 4
7	$000C	INT5	External Interrupt Request 5
8	$000E	INT6	External Interrupt Request 6
9	$0010	INT7	External Interrupt Request 7
10	$0012	TIMER2 COMP	Timer/Counter2 Compare Match
11	$0014	TIMER2 OVF	Timer/Counter2 Overflow
12	$0016	TIMER1 CAPT	Timer/Counter1 Capture Event
13	$0018	TIMER1 COMPA	Timer/Counter1 Compare Match A
14	$001A	TIMER1 COMPB	Timer/Counter1 Compare Match B
15	$001C	TIMER1 OVF	Timer/Counter1 Overflow
16	$001E	TIMER0 COMP	Timer/Counter0 Compare Match
17	$0020	TIMER0 OVF	Timer/Counter0 Overflow
18	$0022	SPI, STC	SPI Serial Transfer Complete
19	$0024	USART0, RX	USART0, Rx Complete
20	$0026	USART0, UDRE	USART0 Data Register Empty
21	$0028	USART0, TX	USART0, Tx Complete
22	$002A	ADC	ADC Conversion Complete
23	$002C	EE READY	EEPROM Ready
24	$002E	ANALOG COMP	Analog Comparator
25	$0030(3)	TIMER1 COMPC	Timer/Countre1 Compare Match C
26	$0032(3)	TIMER3 CAPT	Timer/Counter3 Capture Event
27	$0034(3)	TIMER3 COMPA	Timer/Counter3 Compare Match A
28	$0036(3)	TIMER3 COMPB	Timer/Counter3 Compare Match B
29	$0038(3)	TIMER3 COMPC	Timer/Counter3 Compare Match C
30	$003A(3)	TIMER3 OVF	Timer/Counter3 Overflow

Steps in executing an interrupt



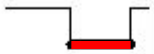
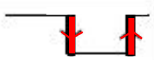
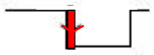
Address	Code
	.INCLUDE "M32DEF.INC"
	.ORG 0 ;location for reset
0000	JMP MAIN
0002	.ORG 0x02 ;location for external INT0
0002	JMP EX0_ISR
0004	MAIN: LDI R20,HIGH(RAMEND)
0005	OUT SPH,R20
0006	LDI R20,LOW(RAMEND)
0007	OUT SPL,R20
0008	SBI DDRC,3 ;PC.3 = output
0009	SBI PORTD,2 ;pull-up activated
000A	LDI R20,1<<INT0 ;Enable INT0
000B	OUT GICR,R20
000C	SEI ;Set I (Enable Interrupts)
000D	LDI R30, 3
000E	LDI R31, 4
000F	ADD R30, R31
0010	HERE:JMP HERE
0012	EX0_ISR:IN R21,PORTC
0013	LDI R22,0x08
0014	EOR R21,R22
0015	OUT PORTC,R21
0016	RETI

Edge trigger Vs. Level trigger in external interrupts

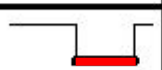
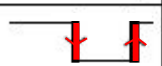
MCUCR	SE	SM2	SM1	SM0	ISC11	ISC10	ISC01	ISC00

ISC01, ISC00 (Interrupt Sense Control bits) These b
external INT0 pin that activates the interrupt, as shown i

```
LDI   R20,0x02     ;falling
OUT   MCUCR,R20
```

ISC01	ISC00		
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.

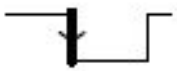
ISC11, ISC10 These bits define the level or edge that activates the INT1 pin.

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.

Edge trigger Vs. Level trigger (Cont.)

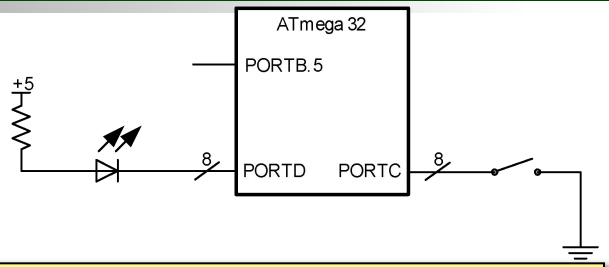
MCUCSR	JTD	ISC2	-	JTRF	WDRF	BORF	EXTRF	PORF

ISC2 This bit defines whether the INT2 interrupt activates on the falling edge or the rising edge.

ISC2		Description
0		The falling edge of INT2 generates an interrupt request.
1		The rising edge of INT2 generates an interrupt request.

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  ;Program 10-1
2  .INCLUDE "M32DEF.INC"
3  .ORG     0x0  ;location for reset
4       JMP MAIN
5  .ORG     0x16;loc. for Timer0 over.
6       JMP T0_OV_ISR
7  ;----main program for initialization
8  .ORG     0x100
9  MAIN:    LDI  R20,HIGH(RAMEND)
10      OUT SPH,R20
11      LDI  R20,LOW(RAMEND)
12      OUT SPL,R20
13      SBI  DDRB,5   ;output
14      LDI  R20,0
15      OUT DDRC, R20
16      LDI  R20,0xFF
17      OUT DDRD, R20
```

Timer int. init.

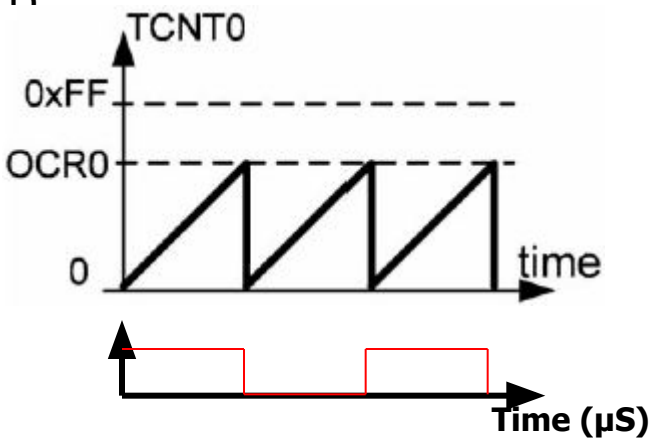
```
20      LDI  R20,(1<<TOIE0)
        OUT TIMSK,R20
        SEI
```

Timer init.

```
21      LDI  R20,-32  ;value for 4µs
        OUT TCNT0,R20
        LDI  R20,0x01
24      OUT TCCR0,R20
25 HERE:   IN   R20,PINC
26      OUT      PORTD,R20
27      JMP HERE
28 ;--------ISR for Timer 0
29 T0_OV_ISR:
30      IN   R16,PORTB
31      LDI  R17,0x20
32      EOR R16,R17
33      OUT      PORTB,R16
34      RETI
```

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




```
.INCLUDE "M32DEF.INC"
.ORG 0x0  ;location for reset
     JMP  MAIN
.ORG 0x14 ;location for Timer0 compare match
     JMP  T0_CM_ISR
;-main program for initialization and keeping CPU busy
.ORG 0x100
MAIN:    LDI  R20,HIGH(RAMEND)
     OUT  SPH,R20
     LDI  R20,LOW(RAMEND)
     OUT  SPL,R20
     LDI  R20,39
     OUT  OCR0,R20      ;OCR0 = 39
     LDI  R20,0x09
     OUT  TCCR0,R20     ;Start Timer0
     SBI  DDRB,5        ;PB5 as an output
     LDI  R20,(1<<OCIE0) ;Timer0 compare match
     OUT  TIMSK,R20
     SEI         ;Set I
     LDI  R20,0x00
     OUT  DDRC,R20    ;make PORTC input
     LDI  R20,0xFF
     OUT  DDRD,R20    ;make PORTD output
```

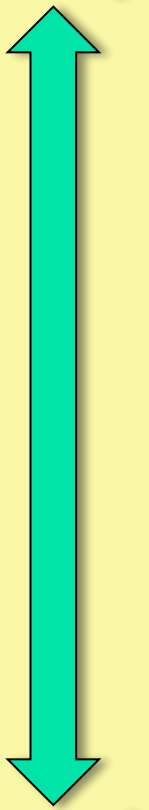
Timer init.

Timer int. init.

```
HERE:    IN   R20,PINC
     OUT  PORTD,R20
     JMP  HERE
;--------ISR for Timer 0
T0_CM_ISR:
     IN   R16,PORTB
     LDI  R17,0x20
     EOR  R16,R17
     OUT  PORTB,R16
     RETI
```

Interrupt priority

Vector No.	Program Address[2]	Source	Interrupt Definition
1	$0000[1]	RESET	External Pin, Power-on Reset, Brown-out Reset, Watchdog Reset, and JTAG AVR Reset
2	$0002	INT0	External Interrupt Request 0
3	$0004	INT1	External Interrupt Request 1
4	$0006	INT2	External Interrupt Request 2
5	$0008	INT3	External Interrupt Request 3
6	$000A	INT4	External Interrupt Request 4
7	$000C	INT5	External Interrupt Request 5
8	$000E	INT6	External Interrupt Request 6
9	$0010	INT7	External Interrupt Request 7
10	$0012	TIMER2 COMP	Timer/Counter2 Compare Match
11	$0014	TIMER2 OVF	Timer/Counter2 Overflow
12	$0016	TIMER1 CAPT	Timer/Counter1 Capture Event
13	$0018	TIMER1 COMPA	Timer/Counter1 Compare Match A
14	$001A	TIMER1 COMPB	Timer/Counter1 Compare Match B
15	$001C	TIMER1 OVF	Timer/Counter1 Overflow
16	$001E	TIMER0 COMP	Timer/Counter0 Compare Match
17	$0020	TIMER0 OVF	Timer/Counter0 Overflow
18	$0022	SPI, STC	SPI Serial Transfer Complete
19	$0024	USART0, RX	USART0, Rx Complete
20	$0026	USART0, UDRE	USART0 Data Register Empty
21	$0028	USART0, TX	USART0, Tx Complete
22	$002A	ADC	ADC Conversion Complete
23	$002C	EE READY	EEPROM Ready
24	$002E	ANALOG COMP	Analog Comparator
25	$0030[3]	TIMER1 COMPC	Timer/Countre1 Compare Match C
26	$0032[3]	TIMER3 CAPT	Timer/Counter3 Capture Event
27	$0034[3]	TIMER3 COMPA	Timer/Counter3 Compare Match A
28	$0036[3]	TIMER3 COMPB	Timer/Counter3 Compare Match B
29	$0038[3]	TIMER3 COMPC	Timer/Counter3 Compare Match C
30	$003A[3]	TIMER3 OVF	Timer/Counter3 Overflow



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.

Task switching and resource conflict

- Does the following program work?

```
1  .INCLUDE "M32DEF.INC"
2  .ORG     0x0  ;location for reset
3        JMP MAIN
4  .ORG     0x14;Timer0 compare match
5        JMP T0_CM_ISR
6  ;---------main program----------------------------
7  .ORG     0x100
8  MAIN:    LDI  R20,HIGH(RAMEND)
9        OUT SPH,R20
10       LDI  R20,LOW(RAMEND)
11       OUT SPL,R20  ;set up stack
12       SBI  DDRB,5  ;PB5 = output
13       LDI  R20,160
14       OUT OCR0,R20
15       LDI  R20,0x09
16       OUT TCCR0,R20
17       LDI  R20,(1<<OCIE0)
18       OUT TIMSK,R20
19       SEI
20       LDI  R20,0xFF
21       OUT DDRC,R20
22       OUT DDRD,R20
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   R20,PIND
30       INC  R20
31       OUT      PORTD,R20
32       RETI
```

Solution 1: different registers

- Use different registers for different tasks.

```
1  .INCLUDE "M32DEF.INC"
2  .ORG     0x0  ;location for reset
3        JMP MAIN
4  .ORG     0x14;Timer0 compare match
5        JMP T0_CM_ISR
6  ;---------main program----------------------------
7  .ORG     0x100
8  MAIN:    LDI  R20,HIGH(RAMEND)
9        OUT SPH,R20
10       LDI  R20,LOW(RAMEND)
11       OUT SPL,R20  ;set up stack
12       SBI  DDRB,5  ;PB5 = output
13       LDI  R20,160
14       OUT OCR0,R20
15       LDI  R20,0x09
16       OUT TCCR0,R20
17       LDI  R20,(1<<OCIE0)
18       OUT TIMSK,R20
19       SEI
20       LDI  R20,0xFF
21       OUT DDRC,R20
22       OUT DDRD,R20
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
```

Solution 2: Context saving

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

```
1  .INCLUDE "M32DEF.INC"
2  .ORG     0x0  ;location for reset
3        JMP MAIN
4  .ORG     0x14;Timer0 compare match
5        JMP T0_CM_ISR
6  ;---------main program----------------------------
7  .ORG     0x100
8  MAIN:    LDI  R20,HIGH(RAMEND)
9        OUT SPH,R20
10       LDI  R20,LOW(RAMEND)
11       OUT SPL,R20  ;set up stack
12       SBI  DDRB,5  ;PB5 = output
13       LDI  R20,160
14       OUT OCR0,R20
15       LDI  R20,0x09
16       OUT TCCR0,R20
17       LDI  R20,(1<<OCIE0)
18       OUT TIMSK,R20     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
```

Saving SREG

- 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
```