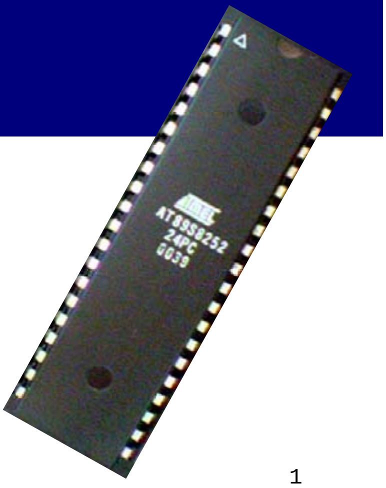
ATmega16A Microcontroller



Interrupts

Interrupts

- Interruption mechanism allows the microcontroller to respond to external events, or to events generated by chip peripherals.
- If no event, the processor
    - □ can run the main program,
    - □ or may enter into a state of inactivity (sleep) to conserve energy.

Interrupts Enable / Disable

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	

SREG Bit 7 (I): Global Interrupt Enable

1 = enable, 0 = disable all interrupts

- Instructions
  - □ **SEI** – interrupt system enabled SREG(7) = 1
  - □ **CLI** – interrupt system disabled SREG(7) = 0

Interrupts Enable

- Assembly Code Example

```
sei ; set global interrupt enable
sleep ; enter sleep, waiting for interrupt
```

- C Code Example

```
_SEI(); /* set global interrupt enable */
_SLEEP(); /* enter sleep, waiting for
  interrupt */
```

Interrupts Disable

- C Code Example

```
char cSREG;
cSREG = SREG; /* store SREG value */
_CLI(); /* disable interrupts */
EECR |= (1<<EEMWE); /* start EEPROM write */
EECR |= (1<<EEWE);
SREG = cSREG; /* restore SREG value (I-bit)*/
```

Interrupts sources

- *internal* interrupts:
  - □ an event given by internal timer (reaches a certain value)
  - □ end of a A/D conversion, etc.
- *external* interrupt triggered by:
  - □ the receiving a front / level on external pin
  - □ reception of a serial character, etc

Sources and Interrupt Vectors

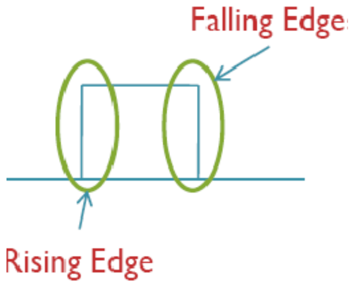
Vector No.	Program Address	Source	Interrupt Definition
1	$000	RESET	External Pin, Power-on Reset, Brown-out Reset, Watchdog Reset, and JTAG Reset
2	$002	INT0	External Interrupt Request 0
3	$004	INT1	External Interrupt Request 1
4	$006	TIMER2 COMP	Timer/Counter2 Compare Match
5	$008	TIMER2 OVF	Timer/Counter2 Overflow
6	$00A	TIMER1 CAPT	Timer/Counter1 Capture Event
7	$00C	TIMER1 COMPA	Timer/Counter1 Compare Match A
8	$00E	TIMER1 COMPB	Timer/Counter1 Compare Match B
9	$010	TIMER1 OVF	Timer/Counter1 Overflow
10	$012	TIMER0 OVF	Timer/Counter0 Overflow

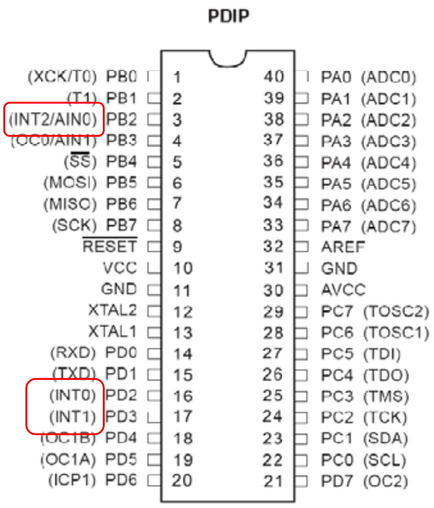
Sources and Interrupt Vectors

Vector No.	Program Address	Source	Interrupt Definition
11	$014	SPI, STC	Serial Transfer Complete
12	$016	USART, RXC	USART, Rx Complete
13	$018	USART, UDRE	USART Data Register Empty
14	$01A	USART, TXC	USART, Tx Complete
15	$01C	ADC	ADC Conversion Complete
16	$01E	EE_RDY	EEPROM Ready
17	$020	ANA_COMP	Analog Comparator
18	$022	TWI	Two-wire Serial Interface
19	$024	INT2	External Interrupt Request 2
20	$026	TIMER0 COMP	Timer/Counter0 Compare Match
21	$028	SPM_RDY	Store Program Memory Ready

External interrupts

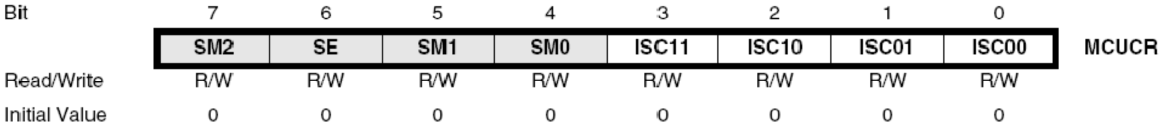
- Triggered by changes in voltage on external pins INT0, INT1 or INT2
  - □ rising edge
  - □ falling edge
  - □ low level





External interrupts

- External interrupt sense control – **MCUCR** (MCU Control and Status Register)

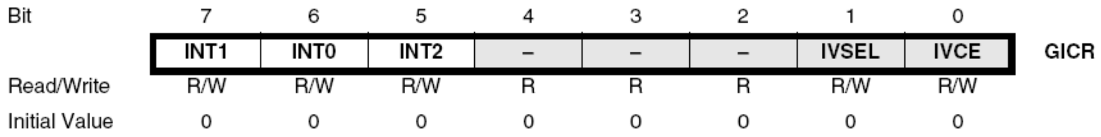


Bit	7	6	5	4	3	2	1	0	
	SM2	SE	SM1	SM0	ISC11	ISC10	ISC01	ISC00	MCUCR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

ISCn1	ISCn0	Descriere
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.

External interrupts

- External Interrupt are activated seting bits 7,6 or 5 from **GICR** (General Interrupt Control Register)



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	

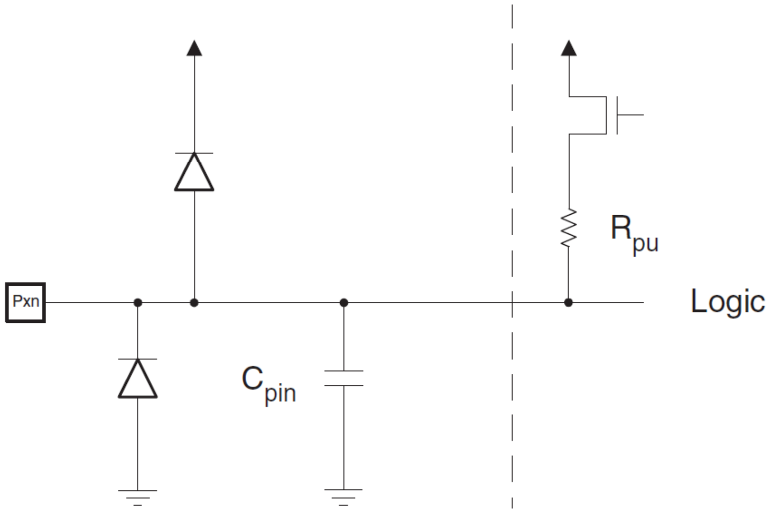
Interrupts

- In C code, interrupt is *serviced* by a C function called ISR (*Interrupt Service Routine*)
  - □ See the test program – Timer 1 interrupt ISR

```
/* Timer 1 Output Compare A is used to blink LED */
interrupt [TIM1_COMPA] void timer1_compa_isr(void)
{
LED1 = ~LED1; // invert LED
}
```

I/O Ports

- **Port A, Port B, Port C, Port D**
- Each bit of each port can be configured as input or output by writing direction register **DDRX**



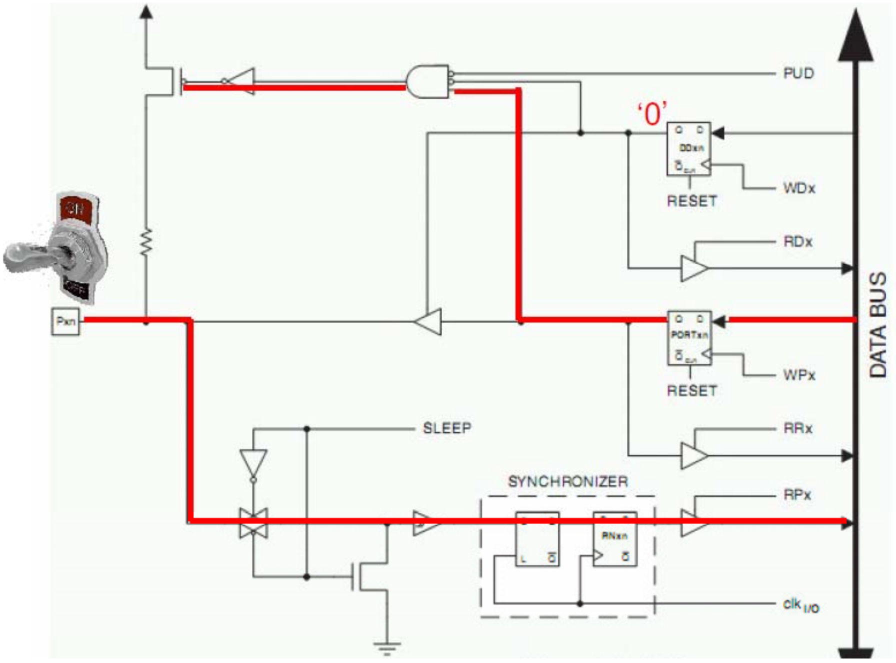
I/O Ports – Input pin

- **Input pins:** initialize with **DDRX.n** = 0
  - □ “Pull up”, resistance can be activated / deactivated
  - □ Set **PORTX.n** = 1 to enable the internal pull-up resistor
  - □ By default, set **PORTX.n** = 0 (no pull-up resistor)
- **Read** value using PINX.n

Example:

```
if(PIND.5 == 0)  // read switch connected on D.5
  LED = 1
```

I/O Ports – Input pin



I/O Ports – Output pin

- **Output pins:** initialize with **DDRx.n** = 1
- **Write value** using **PORTX.n**

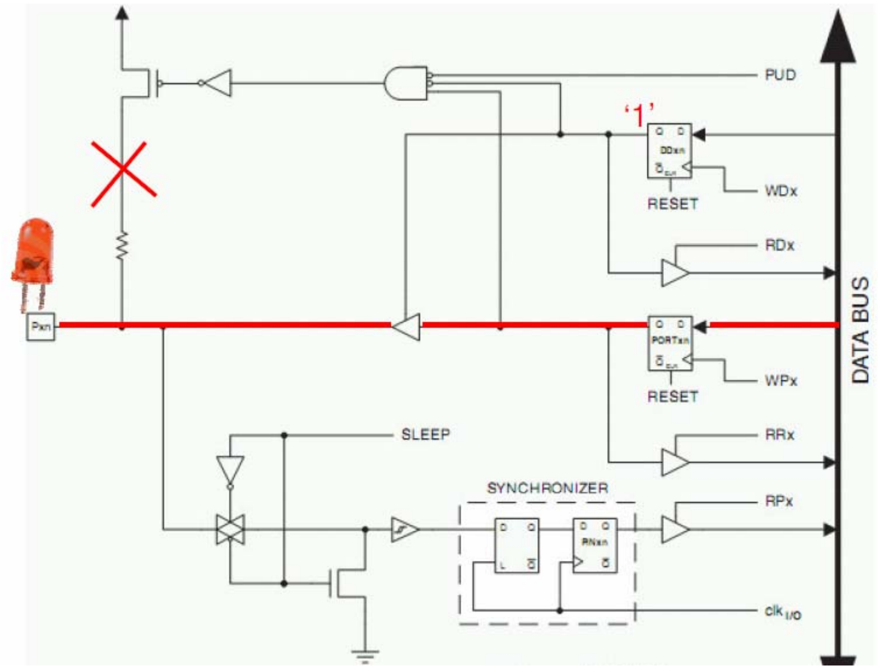
Example:

```
PORTD.6 = 1    // light up LED connected on D.6
```

- Note: you can access all 8 pins of a port at a time:

```
PORTD = 0b11101011
```

I/O Ports – Output pin



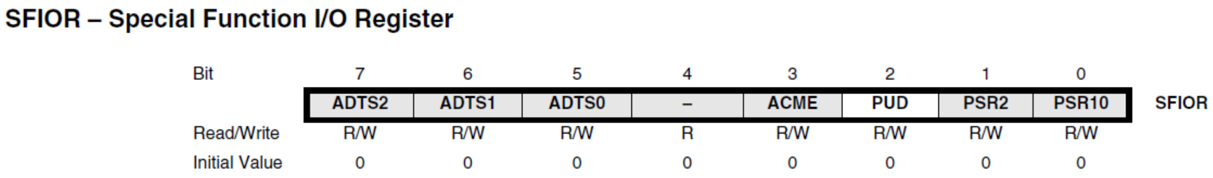
I/O Ports

- Port Pin Configurations

DDxn	PORTxn	PUD (in SFIOR)	I/O	Pull-up	Comment
0	0	X	Input	No	Tri-state (Hi-Z)
0	1	0	Input	Yes	Pxn will source current if ext. pulled low.
0	1	1	Input	No	Tri-state (Hi-Z)
1	0	X	Output	No	Output Low (Sink)
1	1	X	Output	No	Output High (Source)

I/O Ports

- Pull Up Disable – GLOBAL



SFIOR – Special Function I/O Register

Bit	7	6	5	4	3	2	1	0	
	ADTS2	ADTS1	ADTS0	–	ACME	PUD	PSR2	PSR10	SFIOR
Read/Write	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

  - □ **Bit 2 – PUD: Pull-up disable**
  - □ When this bit is written to one, the pull-ups in the I/O ports are disabled even if the DDxn and
  - □ PORTxn Registers are configured to enable the pull-ups

Sensors

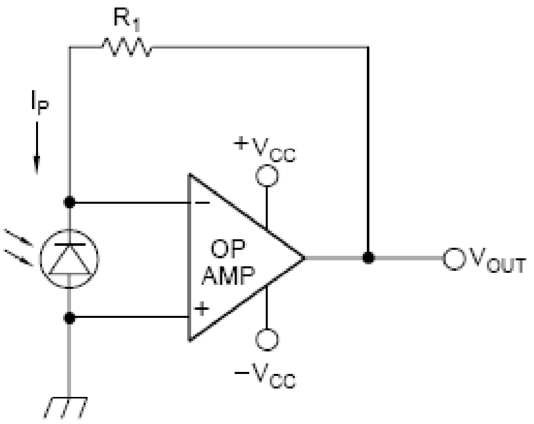
- Digital sensors (TTL)
  - □ examples: contact switches, magnetic switches, optical switches, etc
  - □ states: LO and HI (only 2 values)
  - □ read on an input pin (PINX.n, not PORTX.n)
  - □ you may user a pull-up resistor so the HI state is default; pull LO by connecting the pin to ground → see the first circuit
  - □ internal pull-up: activate using PORTX.n=1 when the direction is set to “input” (DDRX.n=0)
  - □ use the same for analog sensors, when you need to detect the crossing of a treshold

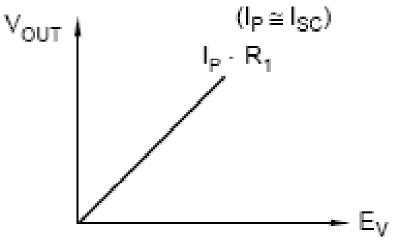
Sensors

- Analog sensors
  - □ many values (8 bits = 256 values; 10 bits = 1024 values)
  - □ use the internal A/D converter
  - □ 8 channels are built-in so you can read 8 separate inputs

Sensors

- Example: light sensor





- AO = 1/2 LM358 (-Vcc = 0V, +Vcc = +5V)
- The photodiode is reverse biased so we measure its dark current
- R1 = tens of KΩ up to 1M Ω