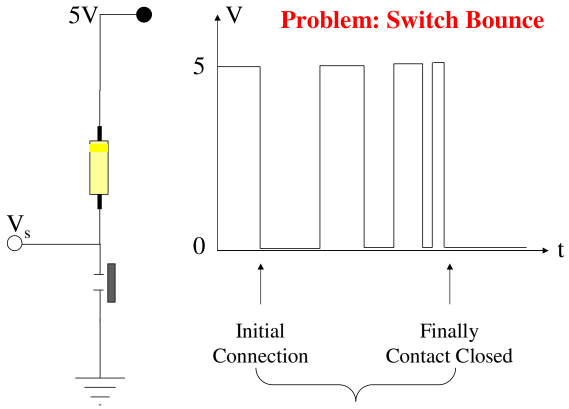
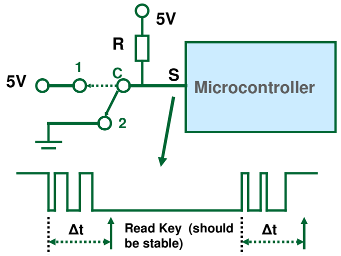
A Switch is Pressed, So What???

Problem: Switch Bounce



Debouncing

- When a switch (any type) changes state (on -> off or off -> on), it presents a mechanical bouncing which generates a signal similar to the one shown at the right.
- The resistor R is needed because the signal S can not be left "floating" in na undefined state when the switch changes from state 1 to 2.
- Without debouncing the signal can generate several interrupts (or status changes) corresponding to just one action.
- Debouncing consists in "Filtering" the signal S so that a proper operation of the switch action is sensed.
- Debouncing can be done in hardware of software



Techniques that can be used:

-If status loop: **after first status change, program timer and after elapsed time read key status.**

-If Interrupt: **on first interrupt program timer which will interrupt after elapsed time. Then read key status.**

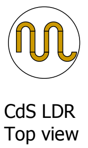
Light Dependent Resistors (LDRs)

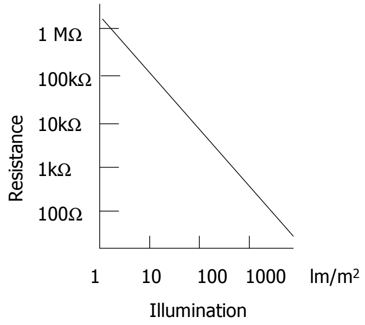
- Devices whose resistance changes (usually decreases) with light striking it
- (also called photocells, photoconductors)
- Light striking a semiconducting material can provide sufficient energy to cause electrons to break away from atoms.
- Free electrons and holes can be created which causes resistance to be reduced

Light Dependent Resistors (LDRs)

- Typical materials used are Cadmium Sulphide (CdS), Cadmium Selenide (CdSe), Lead Sulphide
- With no illumination, resistance can be greater than 1 MΩ (dark resistance).
- Resistance varies inversely proportional to light intensity.
- Reduces down to 10-100s ohms
- 100ms/10ms response time

Light Dependent Resistors (LDRs)



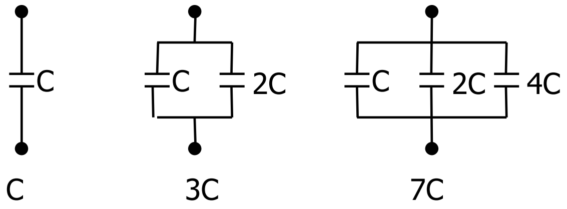


Light Dependent Resistors (LDRs)

- LDRs have a low energy gap
- Operate over a wide wavelengths (some, into infrared)
- Indium antimonide is good for IR. When cooled is very sensitive, used for thermal scanning of earth's surface

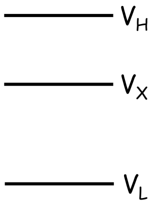
Analog to digital conversion

- Use charge-redistribution technique
  - no sample and hold circuitry needed
  - even with perfect circuits quantization error occurs
- Basic capacitors
  - sum parallel capacitance



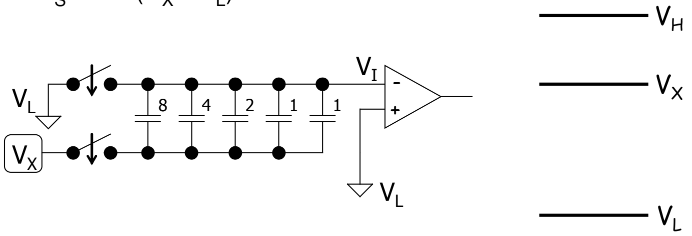
Analog to digital conversion (cont'd)

- Two reference voltage
  - mark bottom and top end of range of analog values that can be converted (V_L and V_H)
  - voltage to convert must be within these bounds (V_X)
- Successive approximation
  - most approaches to A/D conversion are based on this
  - 8 to 16 bits of accuracy
- Approach
  - sample value
  - hold it so it doesn't change
  - successively approximate
  - report closest match



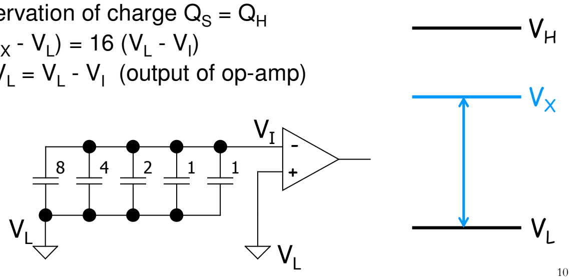
A-to-D – sample

- During the sample time the top plate of all capacitors is switched to reference low V_L
- Bottom plate is set to unknown analog input V_X
- Q = CV
- $Q_S = 16 (V_X - V_L)$



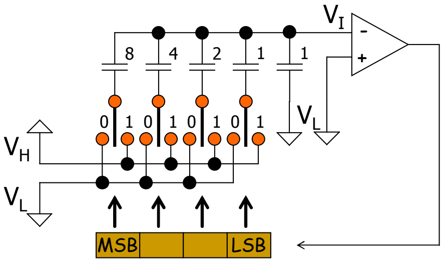
A-to-D – hold

- Hold state using logically controlled analog switches
  - Top plates disconnected from V_L
  - Bottom plates switched from V_X to V_L
- $Q_H = 16 (V_L - V_I)$
  - conservation of charge $Q_S = Q_H$
  - $16 (V_X - V_L) = 16 (V_L - V_I)$
  - $V_X - V_L = V_L - V_I$ (output of op-amp)



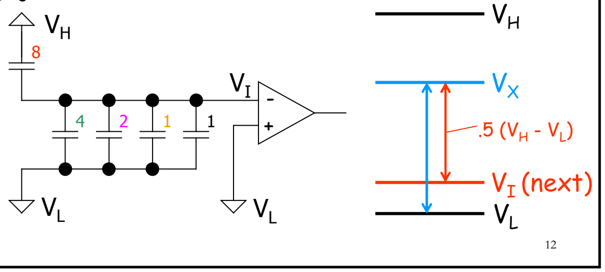
A-to-D – successive approximation

- Each capacitor successively switched from V_L to V_H
  - Largest capacitor corresponds to MSB
- Output of comparator determines bottom plate voltage of cap
  - > 0 : remain connected to V_H
  - < 0 : return to V_L



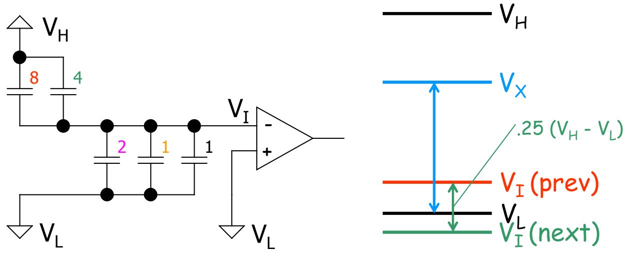
A-to-D example - MSB

- Suppose $V_X = 21/32 (V_H - V_L)$ and already sampled
- Compare after shifting half of capacitance to V_H
  - V_I goes up by $+ 8/16 (V_H-V_I) - 8/16 (V_L-V_I) = + 8/16 (V_H - V_L)$
  - original $V_L - V_I$ goes down and becomes
  - $V_L - (V_I + .5 (V_H - V_L)) = V_L - V_I - .5 (V_H - V_L)$
- Output > 0



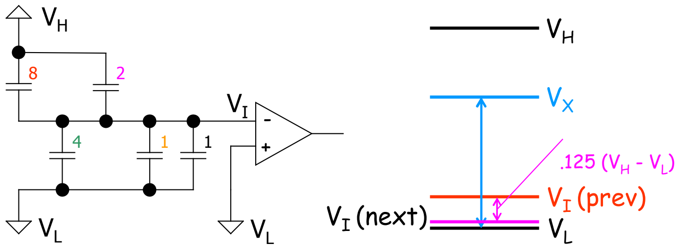
A-to-D example - (MSB-1)

- Compare after shifting another part of cap. to V_H
  - V_I goes up by + 4/16 (V_H-V_I) - 4/16 (V_L-V_I) = + 4/16 (V_H - V_L)
  - original V_L - V_I goes down and becomes
  - V_L - (V_I + .25 (V_H - V_L)) = V_L - V_I - .25 (V_H - V_L)
- Output < 0 (went too far)



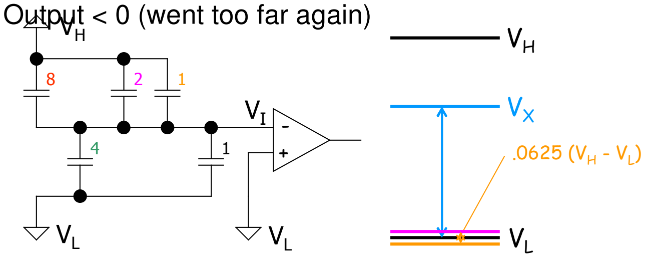
A-to-D example - (MSB-2)

- Compare after shifting another part of cap. to V_H
  - V_I goes up by + 2/16 (V_H-V_I) - 2/16 (V_L-V_I) = + 2/16 (V_H - V_L)
  - original V_L - V_I goes down and becomes
  - V_L - (V_I + .125 (V_H - V_L)) = V_L - V_I - .125 (V_H - V_L)
- Output > 0



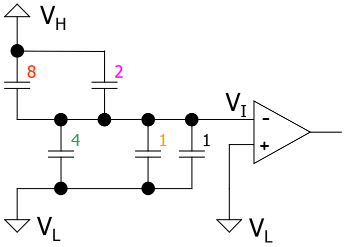
A-to-D example - LSB

- Compare after shifting another part of cap. to V_H
  - V_I goes up by + 1/16 (V_H-V_I) - 1/16 (V_L-V_I) = + 1/16 (V_H - V_L)
  - original V_L - V_I goes down and becomes
  - V_L - (V_I + .0625 (V_H - V_L)) = V_L - V_I - .0625 (V_H - V_L)
- Output < 0 (went too far again)

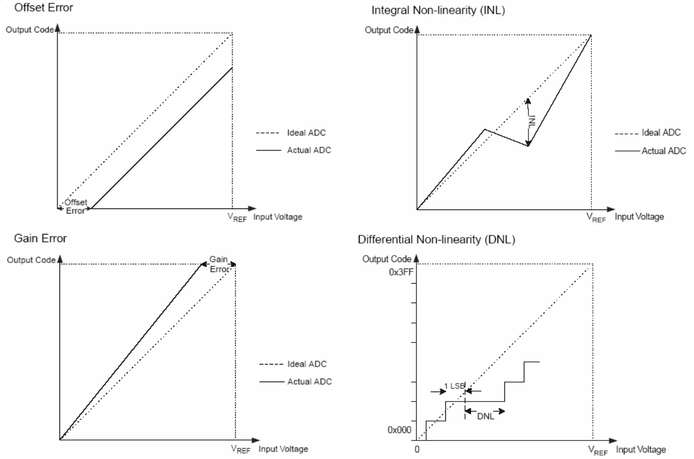


A-to-D example final result

- Input sample of 21/32
- Gives result of <u>1010</u> or 10/16 = 20/32
- 3% error

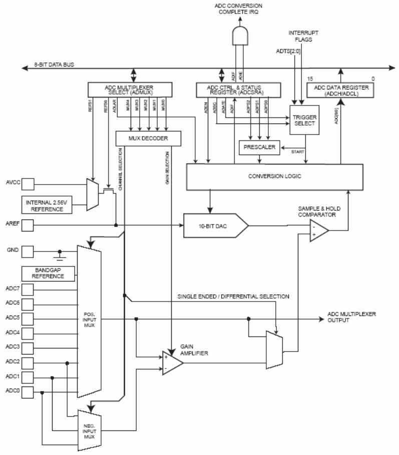


A-to-D Conversion Errors



Closer Look at A-to-D Conversion

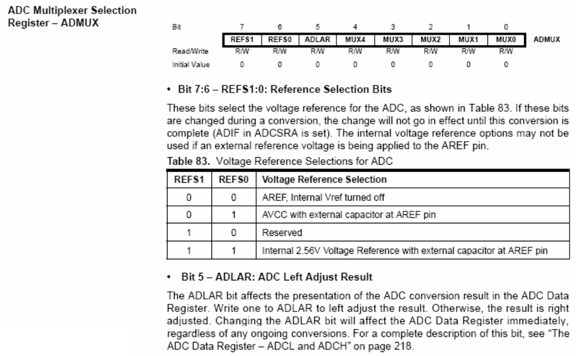
- Needs a comparator and a D-to-A converter
- Takes time to do successive approximation
- Interrupt generated when conversion is completed



A-to-D Conversion on the ATmega16

- 10-bit resolution (adjusted to 8 bits as needed)
- 65-260 usec conversion time
- 8 multiplexed input channels
- Capability to do differential conversion
  - Difference of two pins
  - Optional gain on differential signal (amplifies difference)
- Interrupt on completion of A-to-D conversion
- 0-V_{CC} input range
- 2*LSB accuracy (2 * 1/1024 = ~0.2%)
  - Susceptible to noise – special analog supply pin (AVCC) and capacitor connection for reference voltage (AREF)

A-to-D Conversion (cont'd)



ADC Multiplexer Selection Register – ADMUX

Bit	7	6	5	4	3	2	1	0	
	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0	ADMUX
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 – REFS1:0: Reference Selection Bits**

These bits select the voltage reference for the ADC, as shown in Table 83. If these bits are changed during a conversion, the change will not go in effect until this conversion is complete (ADIF in ADCSRA is set). The internal voltage reference options may not be used if an external reference voltage is being applied to the AREF pin.

Table 83. Voltage Reference Selections for ADC

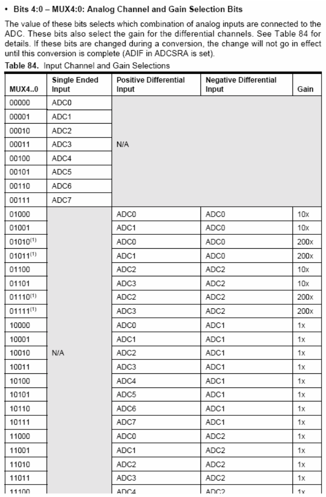
REFS1	REFS0	Voltage Reference Selection
0	0	AREF, Internal Vref turned off
0	1	AVCC with external capacitor at AREF pin
1	0	Reserved
1	1	Internal 2.56V Voltage Reference with external capacitor at AREF pin

- **Bit 5 – ADLAR: ADC Left Adjust Result**

The ADLAR bit affects the presentation of the ADC conversion result in the ADC Data Register. Write one to ADLAR to left adjust the result. Otherwise, the result is right adjusted. Changing the ADLAR bit will affect the ADC Data Register immediately, regardless of any ongoing conversions. For a complete description of this bit, see "The ADC Data Register – ADCL and ADCH" on page 218.

A-to-D Conversion (cont'd)

- Single-ended or differential
  - 1 of 8 single-ended
  - ADCx – ADC1 at 1x gain
  - ADC{0,1} – ADC0 at 10x
  - ADC{0,1} – ADC0 at 200x
  - ADC{2,3} – ADC2 at 10x
  - ADC{2,3} – ADC3 at 200x
  - ADC{0,1,2,3,4,5} – ADC2 at 1x



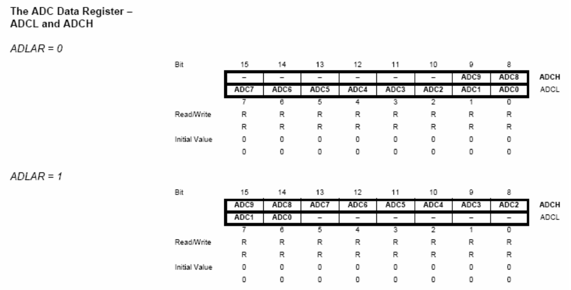
- **Bits 4:0 – MUX4:0: Analog Channel and Gain Selection Bits**

The value of these bits selects which combination of analog inputs are connected to the ADC. These bits also select the gain for the differential channels. See Table 84 for details. If these bits are changed during a conversion, the change will not go in effect until this conversion is complete (ADIF in ADCSRA is set).

Table 84. Input Channel and Gain Selections

MUX4..0	Single Ended Input	Positive Differential Input	Negative Differential Input	Gain
00000	ADC0	N/A		
00001	ADC1			
00010	ADC2			
00011	ADC3			
00100	ADC4			
00101	ADC5			
00110	ADC6			
00111	ADC7			
01000	N/A	ADC0	ADC0	10x
01001		ADC1	ADC0	10x
01010[1]		ADC0	ADC0	200x
01011[1]		ADC1	ADC0	200x
01100		ADC2	ADC2	10x
01101		ADC3	ADC2	10x
01110[1]		ADC2	ADC2	200x
01111[1]		ADC3	ADC2	200x
10000		ADC0	ADC1	1x
10001		ADC1	ADC1	1x
10010		ADC2	ADC1	1x
10011		ADC3	ADC1	1x
10100		ADC4	ADC1	1x
10101		ADC5	ADC1	1x
10110		ADC6	ADC1	1x
10111		ADC7	ADC1	1x
11000		ADC0	ADC2	1x
11001		ADC1	ADC2	1x
11010		ADC2	ADC2	1x
11011		ADC3	ADC2	1x
11100		ADC4	ADC2	1x

A-to-D Conversion (cont'd)



The ADC Data Register – ADCL and ADCH

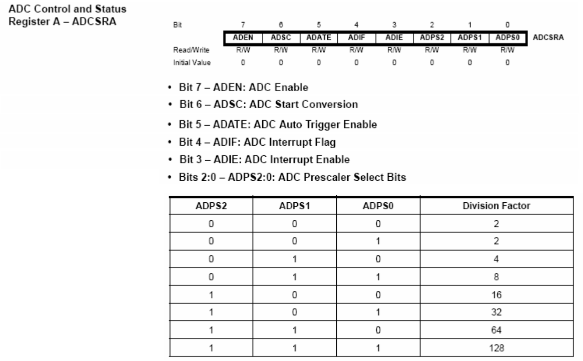
ADLAR = 0

Bit	15	14	13	12	11	10	9	8	
	–	–	–	–	–	–	ADC9	ADC8	ADCH
	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0	ADCL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	R	
	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

ADLAR = 1

Bit	15	14	13	12	11	10	9	8	
	ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADCH
	ADC1	ADC0	–	–	–	–	–	–	ADCL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	R	
	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

A-to-D Conversion (cont'd)



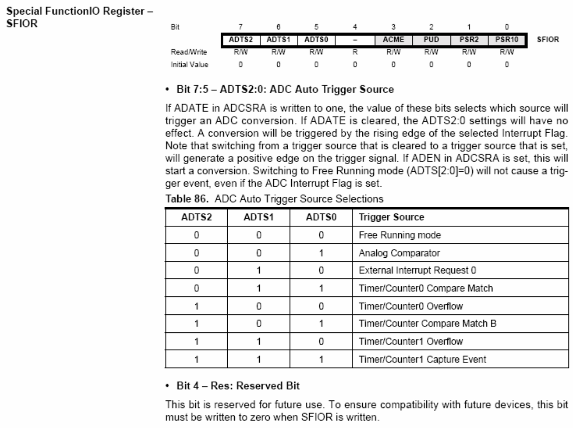
ADC Control and Status Register A – ADCSRA

Bit	7	6	5	4	3	2	1	0	
	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSRA
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 – ADEN: ADC Enable**
- **Bit 6 – ADSC: ADC Start Conversion**
- **Bit 5 – ADATE: ADC Auto Trigger Enable**
- **Bit 4 – ADIF: ADC Interrupt Flag**
- **Bit 3 – ADIE: ADC Interrupt Enable**
- **Bits 2:0 – ADPS2:0: ADC Prescaler Select Bits**

ADPS2	ADPS1	ADPS0	Division Factor
0	0	0	2
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

A-to-D Conversion (cont'd)



Special FunctionIO Register – SFIOR

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 7:5 – ADTS2:0: ADC Auto Trigger Source**

If ADATE in ADCSRA is written to one, the value of these bits selects which source will trigger an ADC conversion. If ADATE is cleared, the ADTS2:0 settings will have no effect. A conversion will be triggered by the rising edge of the selected Interrupt Flag. Note that switching from a trigger source that is cleared to a trigger source that is set, will generate a positive edge on the trigger signal. If ADEN in ADCSRA is set, this will start a conversion. Switching to Free Running mode (ADTS[2:0]=0) will not cause a trigger event, even if the ADC Interrupt Flag is set.

Table 86. ADC Auto Trigger Source Selections

ADTS2	ADTS1	ADTS0	Trigger Source
0	0	0	Free Running mode
0	0	1	Analog Comparator
0	1	0	External Interrupt Request 0
0	1	1	Timer/Counter0 Compare Match
1	0	0	Timer/Counter0 Overflow
1	0	1	Timer/Counter Compare Match B
1	1	0	Timer/Counter1 Overflow
1	1	1	Timer/Counter1 Capture Event

- **Bit 4 – Res: Reserved Bit**

This bit is reserved for future use. To ensure compatibility with future devices, this bit must be written to zero when SFIOR is written.

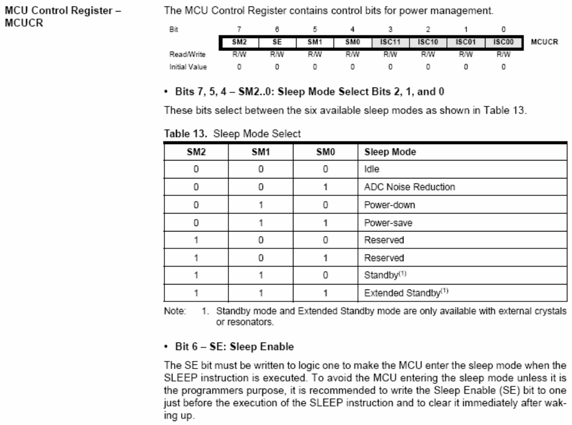
Writing an Interrupt Handler in C (again)

- Ensure main program sets up all registers
- Enable interrupts as needed
- Enable global interrupts (SEI)
- Write handler routine for each enabled interrupt
  - What if an interrupt occurs and a handler isn't defined?
- Make sure routine does not disrupt others
  - Data sharing problem
  - Save any state that might be changed (done by compiler)
- Re-enable interrupts upon return
  - done by compiler with RETI

Power modes

- Processor can go to "sleep" and save power
- Different modes put different sets of modules to sleep
  - Which one to use depends on which modules are needed to wake up processor
  - Timers, external interrupts, ADC, serial communication lines, etc.
- set_sleep_mode (mode);
- sleep_mode ();

Power modes (cont'd)



MCU Control Register – MCUCR

The MCU Control Register contains control bits for power management.

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	

- **Bits 7, 5, 4 – SM2..0: Sleep Mode Select Bits 2, 1, and 0**

These bits select between the six available sleep modes as shown in Table 13.

Table 13. Sleep Mode Select

SM2	SM1	SM0	Sleep Mode
0	0	0	Idle
0	0	1	ADC Noise Reduction
0	1	0	Power-down
0	1	1	Power-save
1	0	0	Reserved
1	0	1	Reserved
1	1	0	Standby[1]
1	1	1	Extended Standby[1]

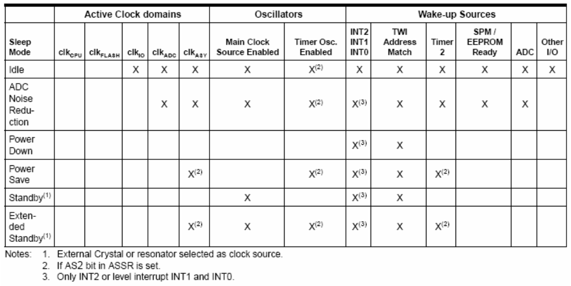
Note: 1. Standby mode and Extended Standby mode are only available with external crystals or resonators.

- **Bit 6 – SE: Sleep Enable**

The SE bit must be written to logic one to make the MCU enter the sleep mode when the SLEEP instruction is executed. To avoid the MCU entering the sleep mode unless it is the programmers purpose, it is recommended to write the Sleep Enable (SE) bit to one just before the execution of the SLEEP instruction and to clear it immediately after waking up.

Power modes (cont'd)

- Wake up sources and active clocks



	Active Clock domains					Oscillators		Wake-up Sources					
Sleep Mode	clk_{CPU}	clk_{FLASH}	clk_{IO}	clk_{ADC}	clk_{ASY}	Main Clock Source Enabled	Timer Osc. Enabled	INT2 INT1 INT0	TWI Address Match	Timer 2	SPM / EEPROM Ready	ADC	Other I/O
Idle			X	X	X	X	X[2]	X	X	X	X	X	X
ADC Noise Reduction				X	X	X	X[2]	X[3]	X	X	X	X	
Power Down								X[3]	X				
Power Save					X[2]		X[2]	X[3]	X	X[2]			
Standby[1]						X		X[3]	X				
Extended Standby[1]					X[2]	X	X[2]	X[3]	X	X[2]			

Notes: 1. External Crystal or resonator selected as clock source.
2. If AS2 bit in ASSR is set.
3. Only INT2 or level interrupt INT1 and INT0.