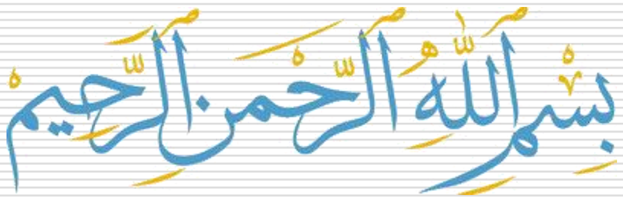


میکرو کنترلرهای AVR

Analog to Digital Converter

دانشکده برق و رباتیک

دانشگاه صنعتی شاهرود

حسین خسروی

Introduction to A-to-D conversion

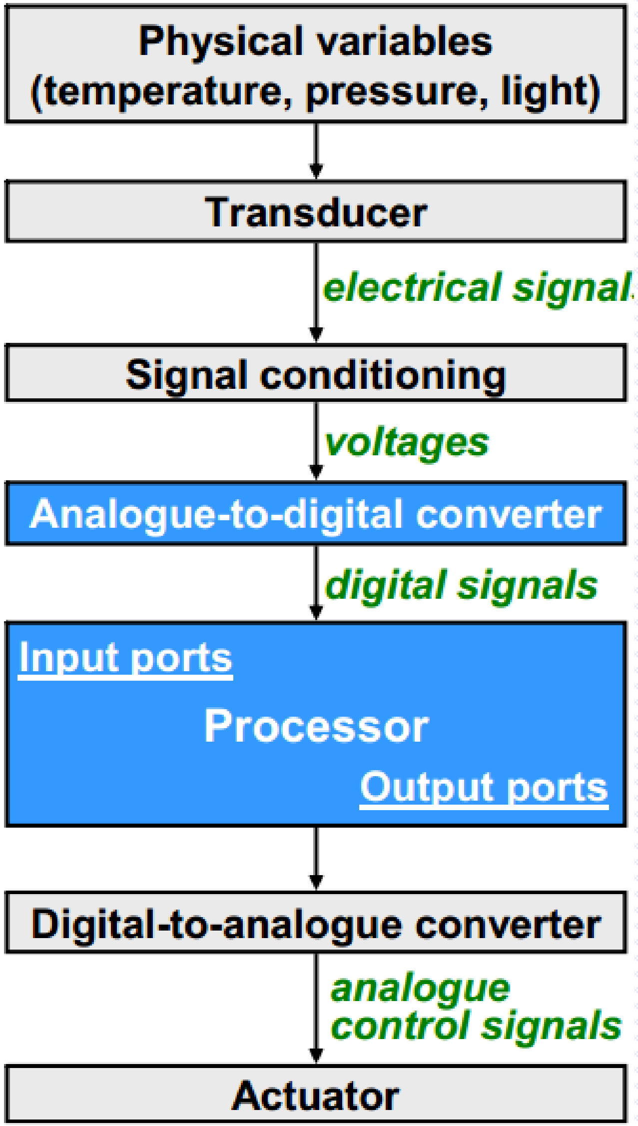
- An ADC samples an analogue signal at discrete times, and converts the sampled signal to digital form.

- Used with transducers, ADCs allow us to monitor real-world inputs and perform control operations based on these inputs.

- Many dedicated ICs are made for ADC.
  - ADC0804: 8-bit, successive approximation.
  - Maxim104: 8-bit, flash type.

A-to-D conversion: Typical embedded application

- Because of ADC is commonly needed, most modern microcontrollers has an in-built ADC unit.



A-to-D conversion: Example applications

- **Local positioning sensor for object tracking**
  - Measure the distance between FM transmitter/receiver.
  - The receiver has an RSSI output (Receiver Signal Strength Indicator).
  - The RSSI voltage is inversely proportional to the squared distance.

- **Temperature sensor for shower water**
  - Measure the temperature of shower water and control hot/cold water valves.
  - Use a thermistor as sensor.

- **Electric fence monitoring**
  - Determine if a electric fence is being tampered.
  - Measure the voltage level of an electric fence.

A-to-D conversion: Example applications

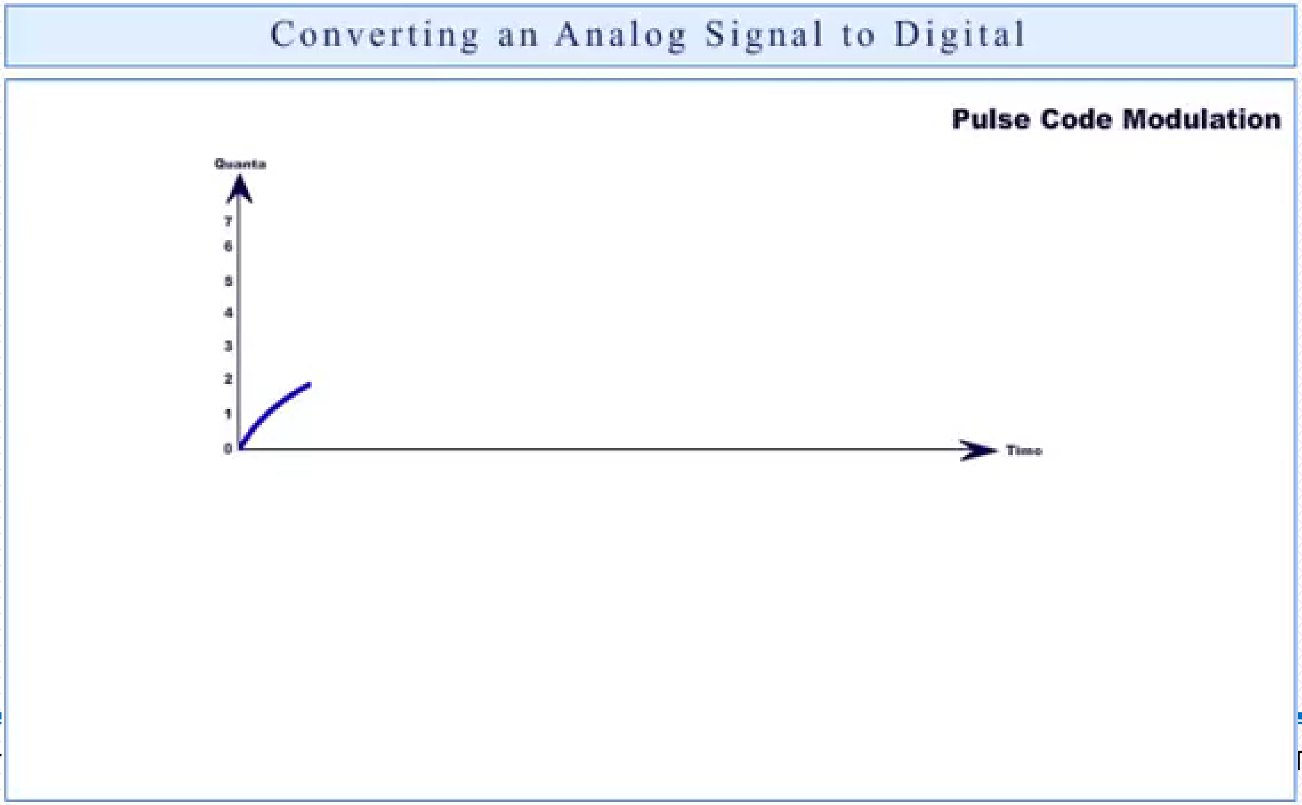
- **Obstacle sensor for the blind**
  - **Measure distance to nearest object with an ultrasonic sensor.**
  - **The sensor output is digitized using the ADC.**

- **Wireless irrigation system**
  - **Measure the moisture of the soil with resistor & ADC.**
  - **Transmit data wirelessly to base station & turn on/off sprinkler.**

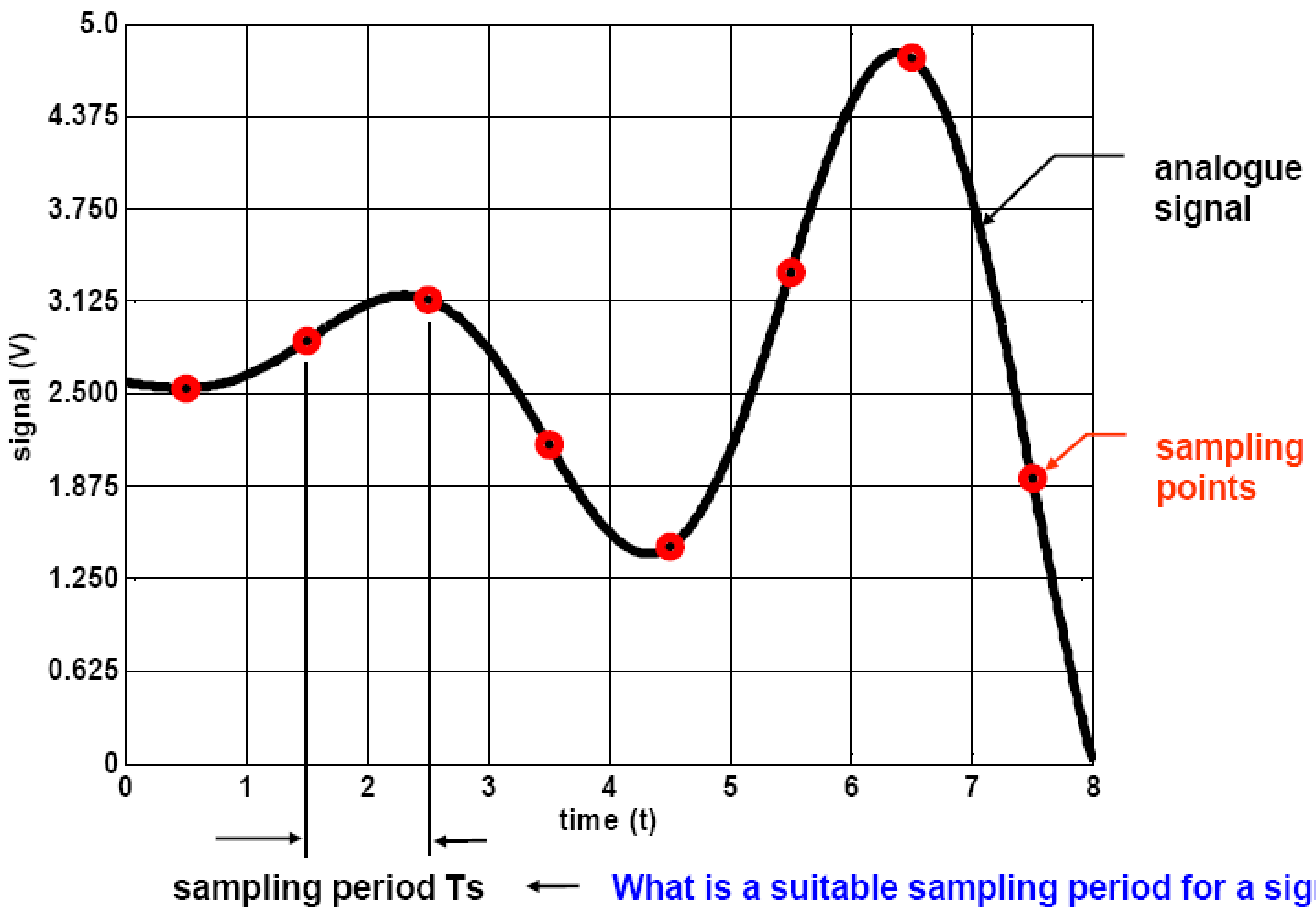
- **Intelligent clothes line**
  - **Use a set of sensors to measure humidity, temperature, wind speed.**
  - **Open/close the cover of the clothesline to protect against rain.**

A-to-D conversion: The process

- There are two related steps in A-to-D conversion:
- **Sampling**:
  - the analogue signal is extracted, usually at regularly spaced time instants.
  - the samples have real values.
- **Quantization**:
  - the samples are quantized to discrete levels.
  - each sample is represented as a digital value.



Sampling an analogue signal



What is a suitable sampling period for a signal?

The sampling theorem

- An analogue signal $x(t)$ with frequencies of no more than F_{max} can be reconstructed exactly from its samples if the sampling rate satisfies: $F_s \geq 2F_{\text{max}}$.

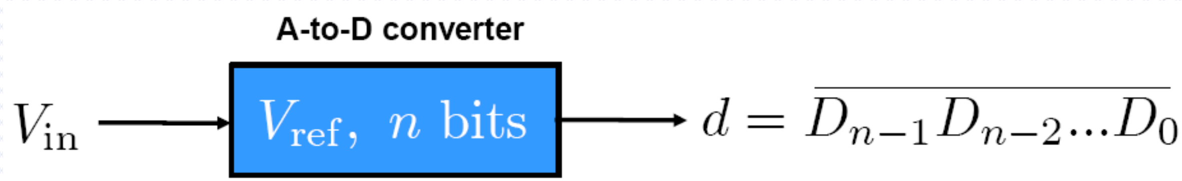
- If maximum frequency of the signal is known to be F_{max}, the sampling rate we use should be at least:

$$Nyquist\ rate = 2 \times F_{max}$$

- If the sampling rate is known to be F_s, the maximum frequency in the signal must not exceed:

$$Nyquist\ frequency = ½\, \boldsymbol{F_s}$$

Quantizing the sampled signal



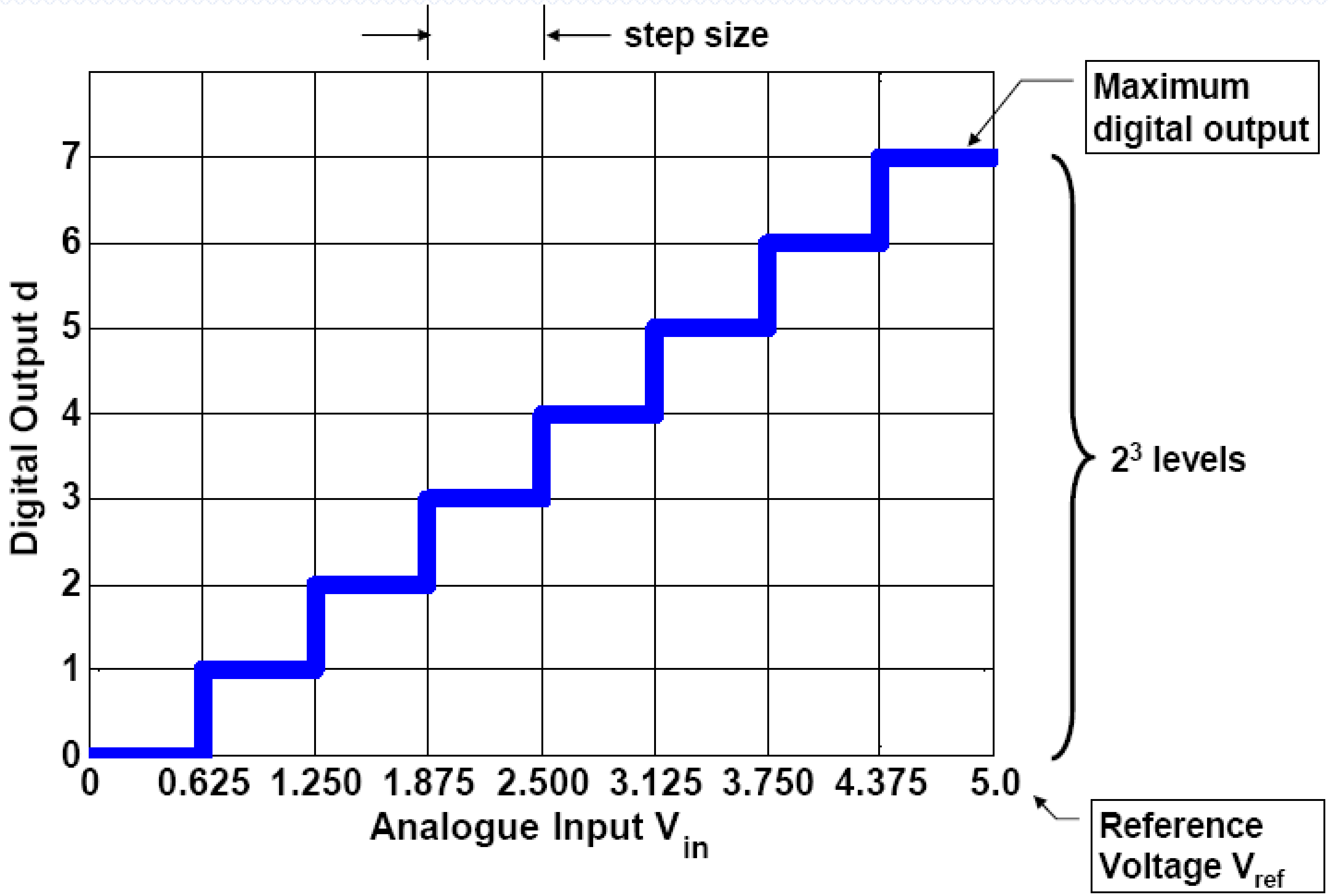
- Let's consider an n-bit ADC.
- Let V_{ref} be the reference voltage.
- Let V_{in} be the analogue input voltage.
- Let V_{min} be the minimum allowable input voltage, usually = 0.
- The ADC's digital output, $d = D_{n-1}D_{n-2} ...D_0$, is given as

$$d \;=\; round\ down\ [\frac{V_{in} - V_{min}}{step\ size}]$$

- The step size (resolution) is the smallest change in input that can be discerned by the ADC:

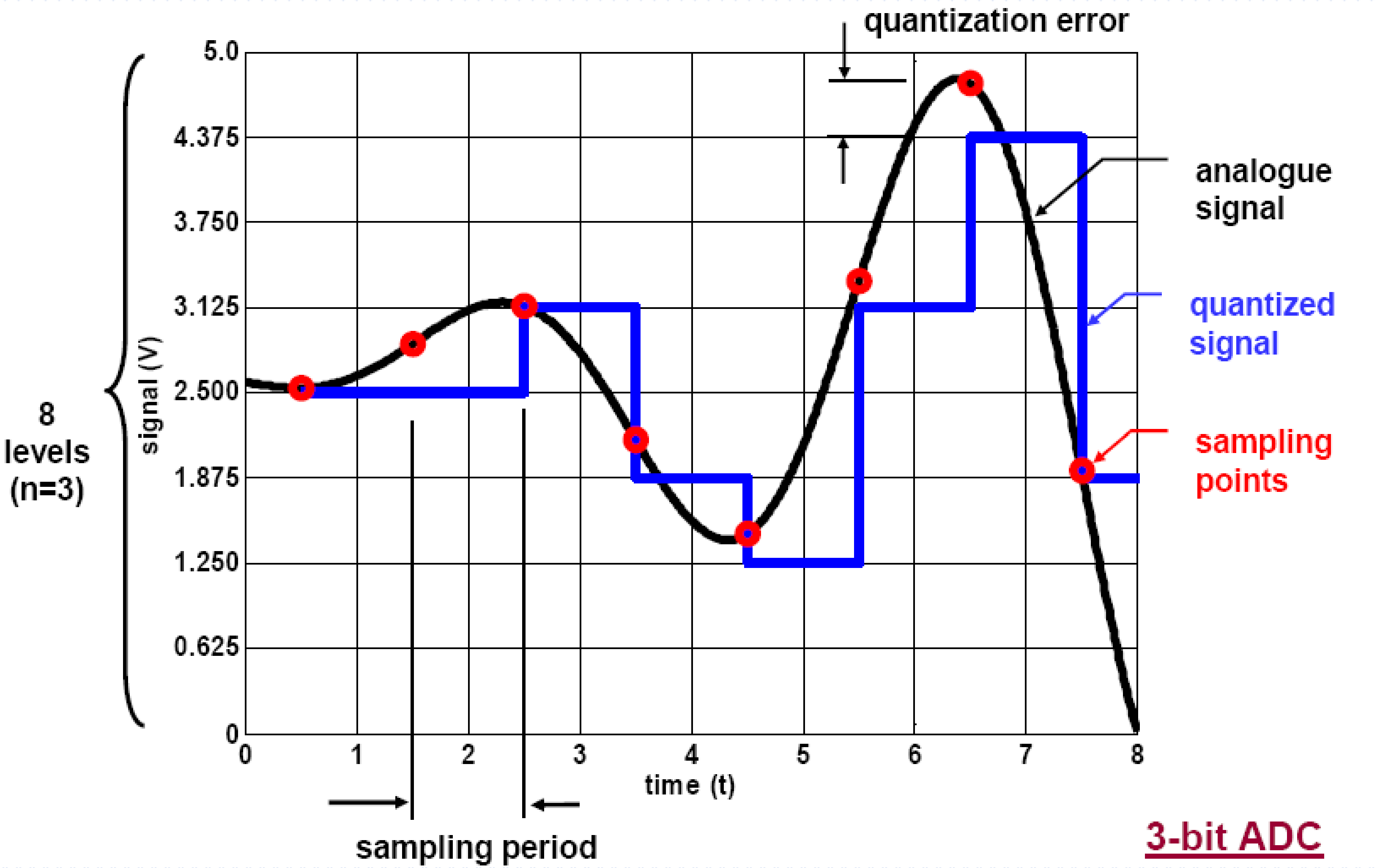
$$step\ size \;=\frac{V_{ref} - V_{min}}{2^n}$$

Quantizing the sampled signal



A 3-bit A-to-D converter

Quantizing the sampled signal



A-to-D converter: Parameters

- **Number of bits n: The higher the number of bits, the more precise the digital output.**

- **Quantisation error E_q: the average difference between the analogue input and the quantized value. The quantization error of an ideal ADC is half of the step size**

- **Sample time T_{sample}: a sampling capacitor must be charged for a duration of t_{sample} before conversion taking place.**

- **Conversion time T_{conv}: time taken to convert the voltage on the sampling capacitor to a digital output.**

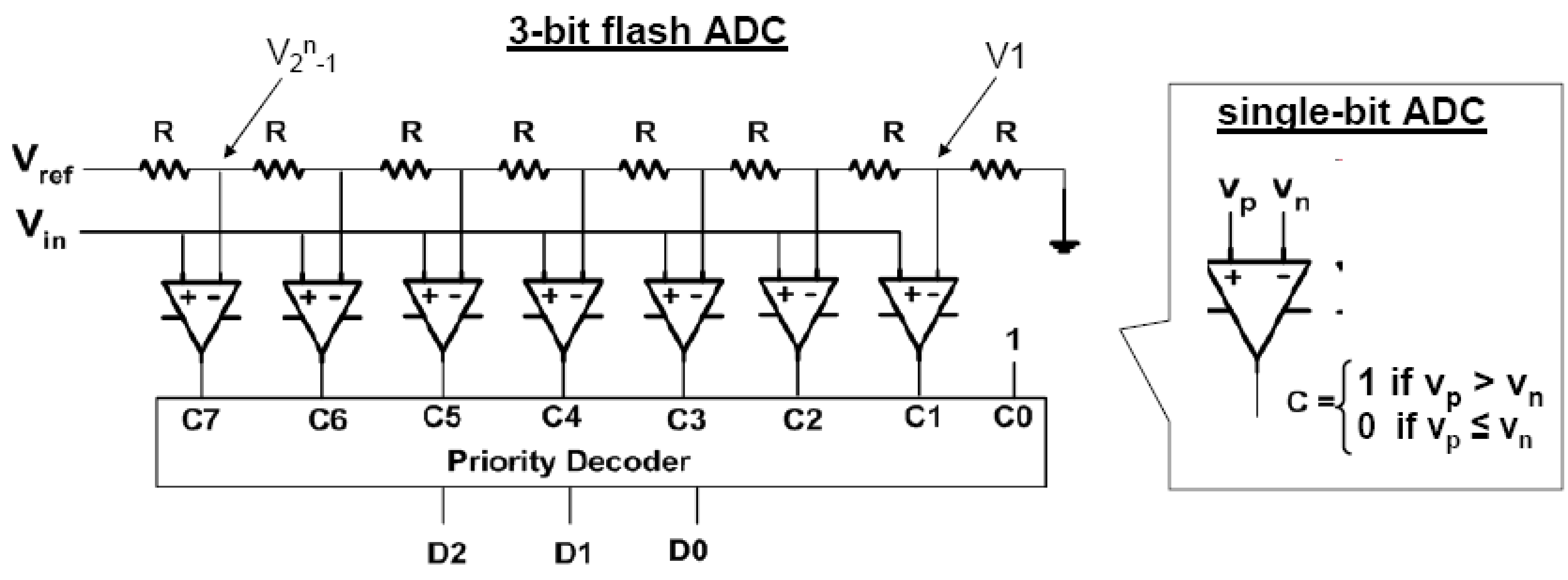
A-to-D converter: Designs

- **There are many designs for analogue-to-digital converters.**

- **We'll consider two common designs.**
  - **Flash ADC**
  - **Successive-approximation ADC (SAR)**

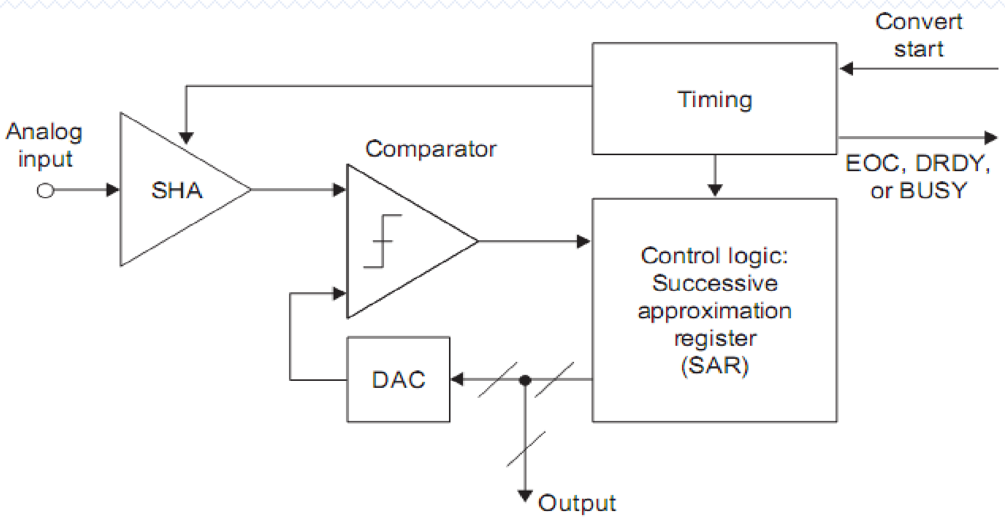
Flash ADC

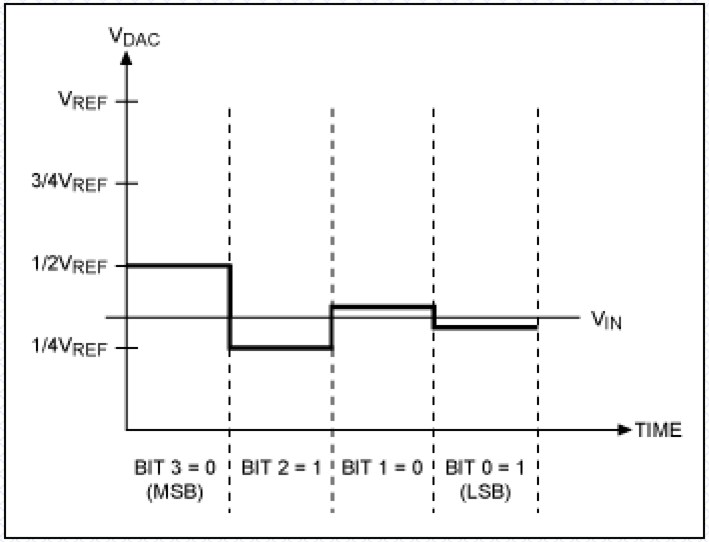
- A n-bit flash ADC uses 2^n-1 comparators and a priority decoder.
- **Advantage**: the fastest type of ADC.
- **Disadvantages**: limited resolution, expensive, and large power consumption.



Successive-approximation ADC

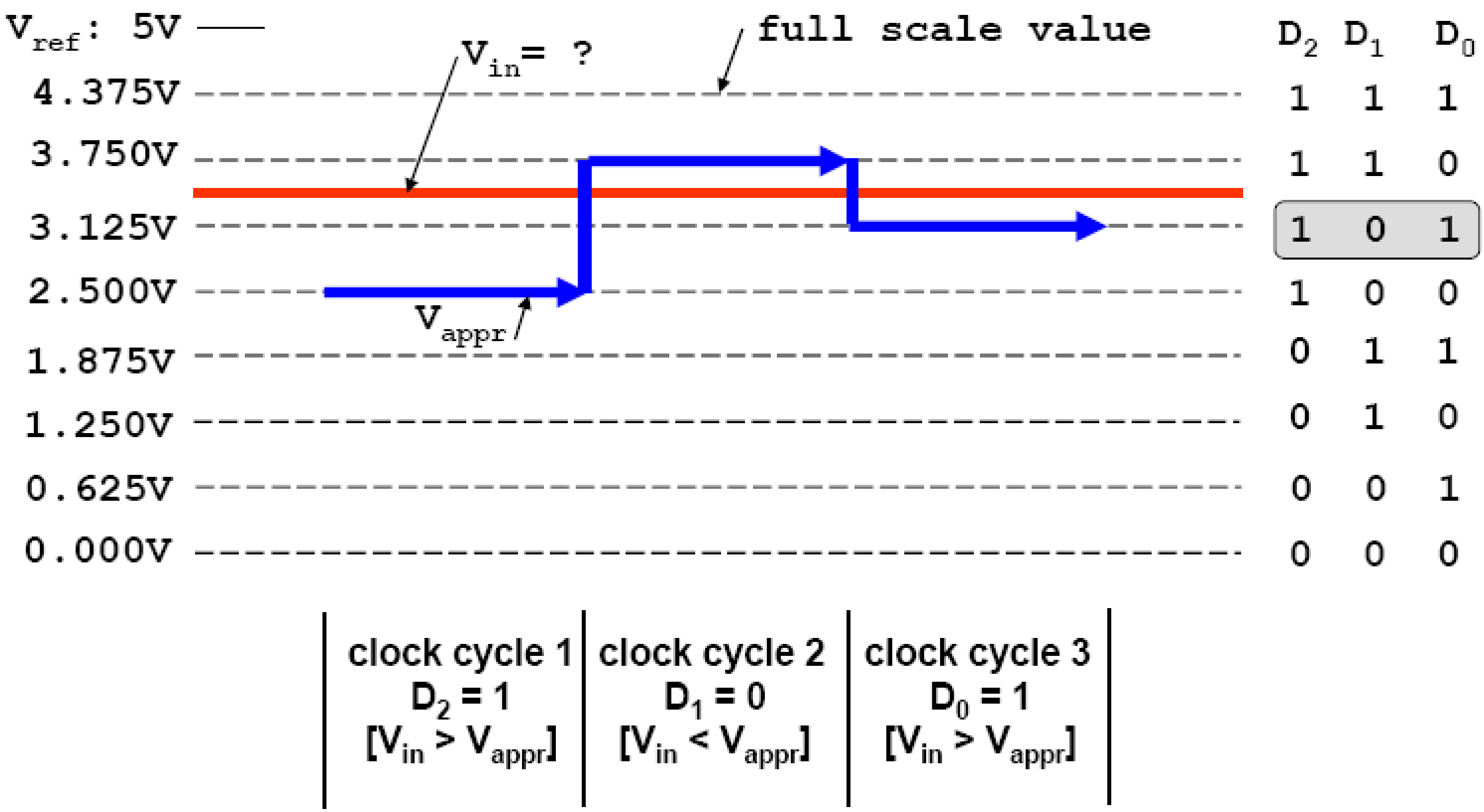
- The mainstay of data acquisition for many years.
- Also known as Feedback Subtraction ADC
- On CONVERT START command, the SHA is placed in the *hold* mode, and all the bits of the successive approximation register (SAR) are reset to “ **0** ” except the MSB which is “ **1** ”.





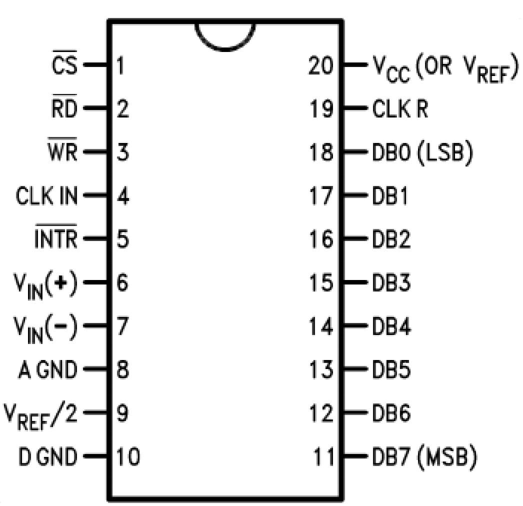
Successive-approximation ADC

Binary search for a 3-bit ADC



Example IC for ADC

- VCC: reference voltage
- RD: to read digital output
- WR: to start a new conversion
- INTR: when conversion completes
- VIN(+), VIN(-): analogue input
- DB0-DB7: 8-bit output
- CLK IN, CLK R: clock signal



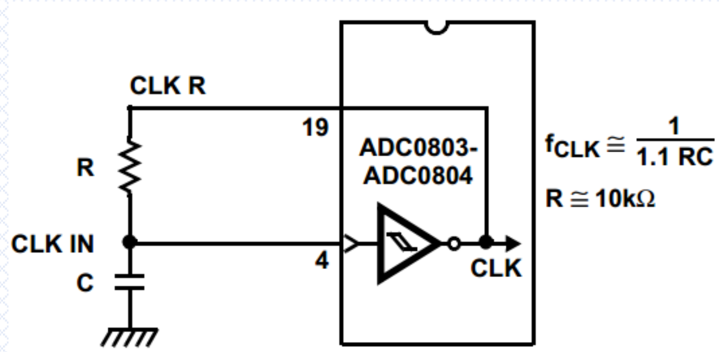
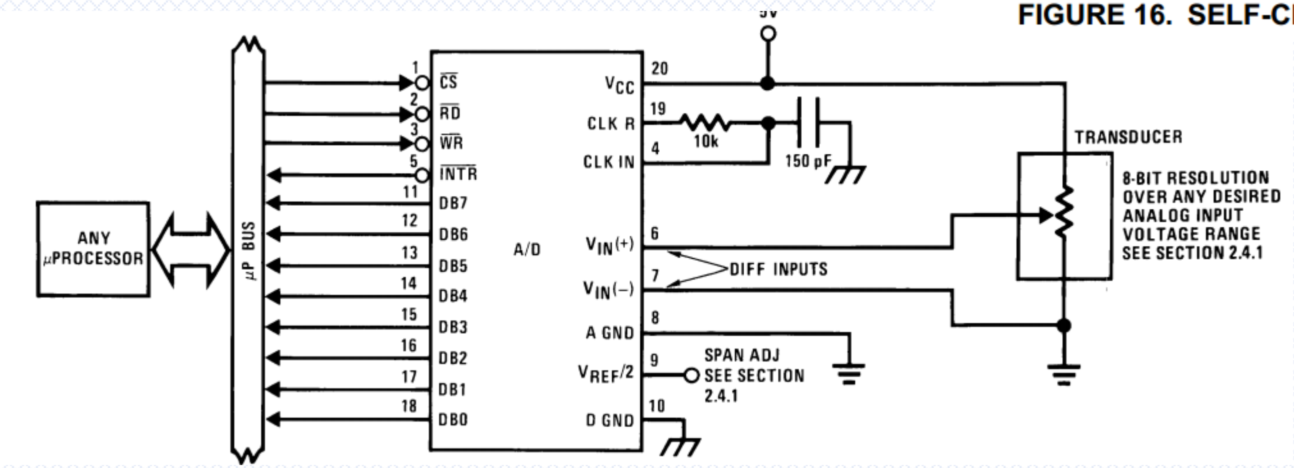


FIGURE 16. SELF-CLOCKING THE A/D



The ADC in ATmega32

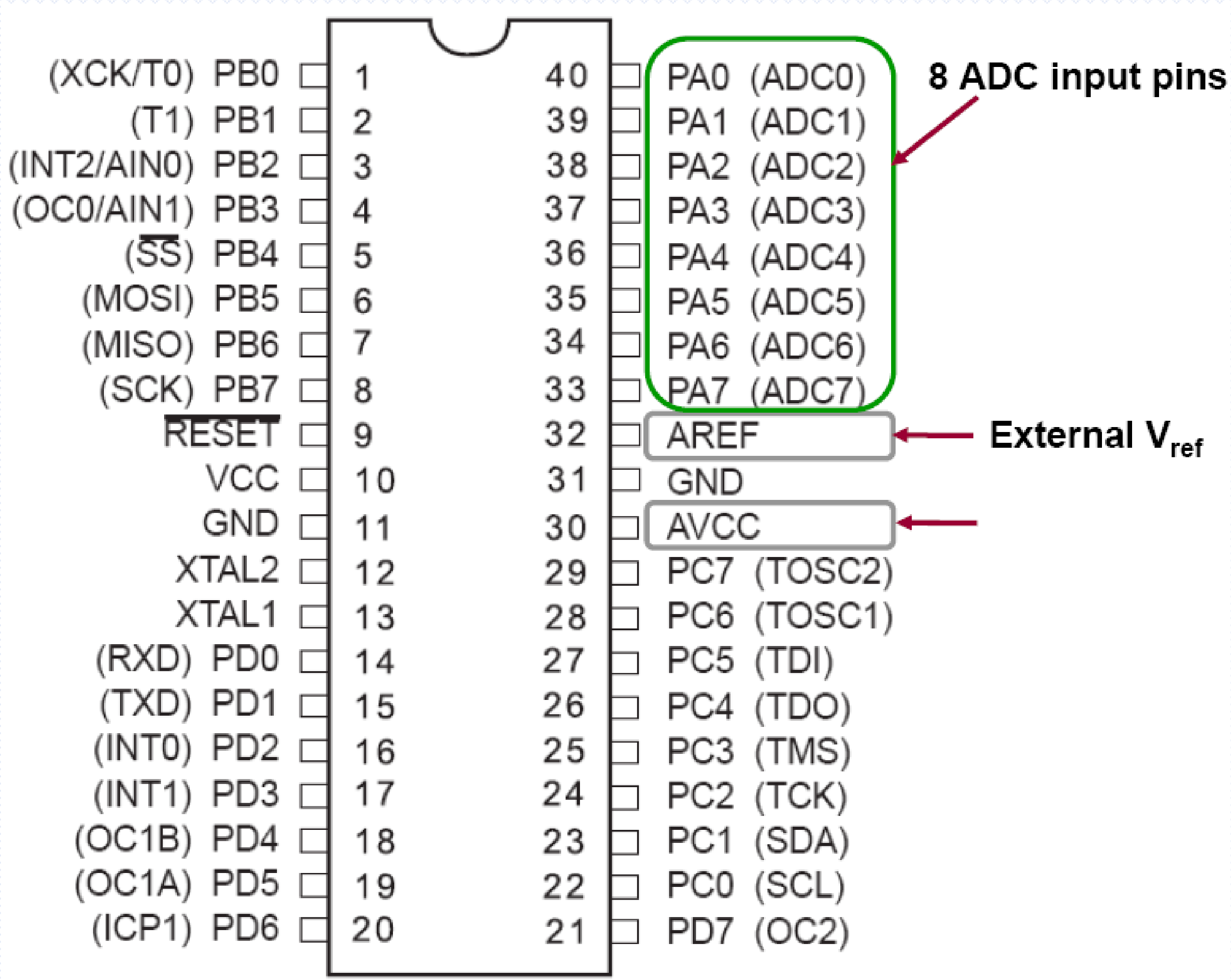
- **The ADC in ATmega32 has a 10-bit resolution.**
  - **The digital output has n = 10 bits.**

- **The ADC has 8 input channels.**
  - **Analogue input can come from 8 different sources.**
  - **However, it performs conversion on only one channel at a time.**

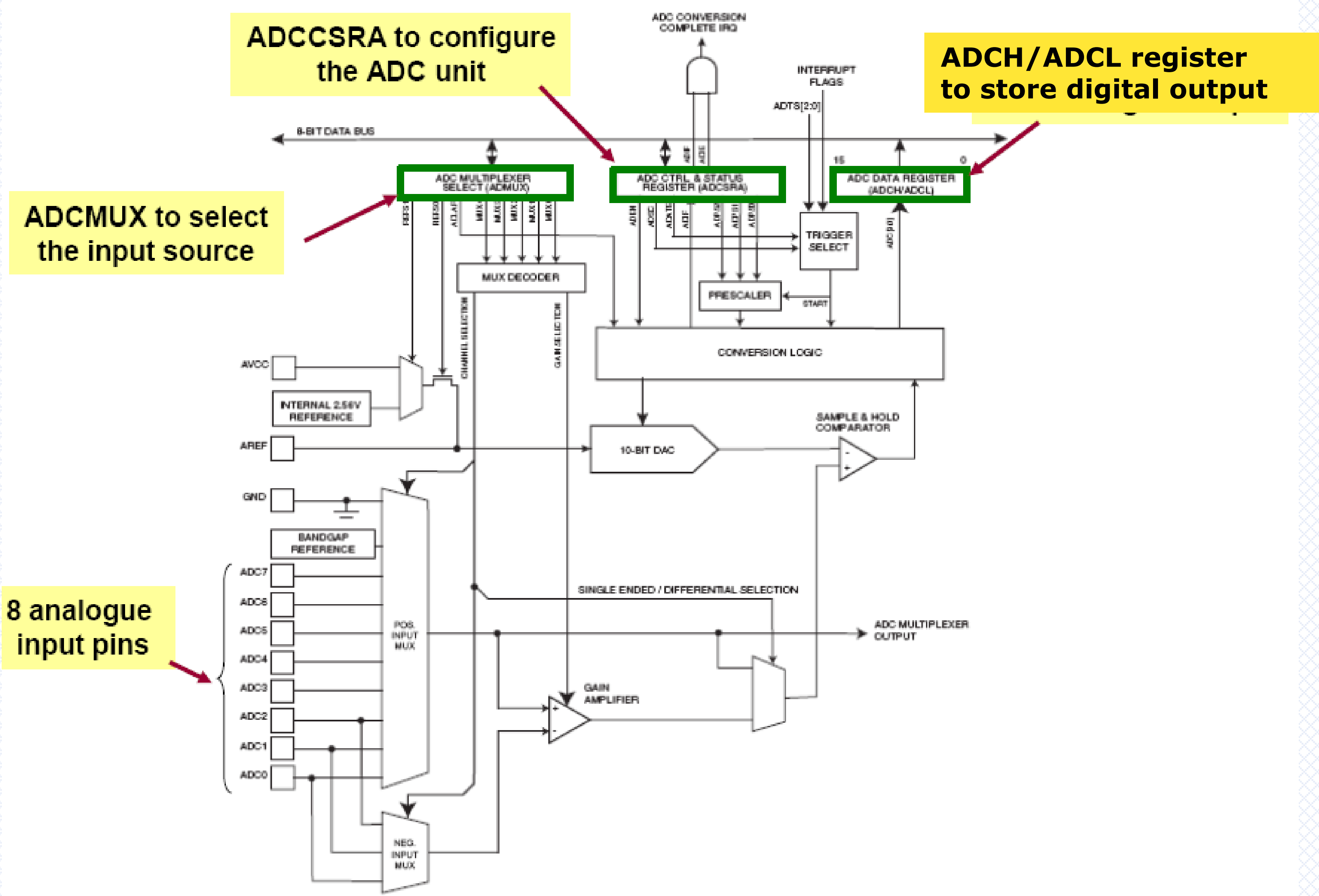
- **If default reference voltage V_{ref} = 5V is used.**
  - **step size: 5(V)/1024 (steps) = 4.88mV.**
  - **accuracy: $\pm$ 2 LSB = $\pm$ 9.76mV**

- **The clock rate of the ADC can be different from the CPU clock rate.**
  - **One ADC conversion takes 13 ADC cycles.**
  - **An ADC prescaler will decide the ADC clock rate.**

ADC unit – Relevant pins

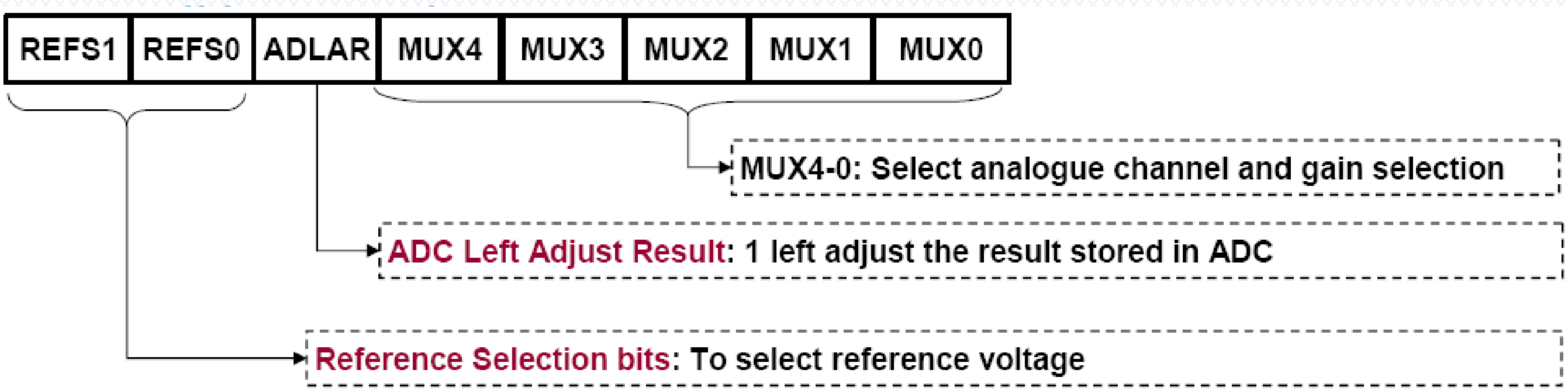


ADC unit – Block diagram



ADC Multiplexer Selection Register (ADMUX)

- Reference voltage Vref can be selected among 3 choices.
- Analogue input voltage can be selected as among different pins.
- Differential input and custom gain factor can also be chosen.
- [10101010][10xxxxxx]
- ADLAR flag will determine how the 10-bit digital output



Selecting reference voltage Vref

- Usually, mode 01 is used: AVCC = 5V as reference voltage.

- However, if the input voltage has a different dynamic range, we can use mode 00 to select an external reference voltage on AREF.

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

Selecting input source and gain factor

- Analogue input voltage can be selected as
  - 8 ADC pins ADC7 to ADC0,
  - the differential input between two of ADC pins.
- A gain factor of 1, 10 or 200 can be selected for differential input.

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	
01010		ADC0	ADC0	200x
01011		ADC1	ADC0	
01100		ADC2	ADC2	10x
01101		ADC3	ADC2	
01110		ADC2	ADC2	200x
01111		ADC3	ADC2	
10000		ADC0	ADC1	1x
10001		ADC1	ADC1	
10010		ADC2	ADC1	
10011		ADC3	ADC1	
10100		ADC4	ADC1	
			ADC1	
			ADC1	
			ADC1	
			ADC2	
			ADC2	
			ADC2	
			ADC2	
			ADC2	

Table 84. Input Channel and Gain Selections (Continued)

MUX4..0	Single Ended Input	Positive Differential Input	Negative Differential Input	Gain
11101		ADC5	ADC2	1x
11110	1.22V (V_{BG})	N/A		
11111	0V (GND)			

ADC Left Adjust flag and ADCH/L registers

- Digital output of A-to-D conversion is stored in two 8-bit registers ADCH and ADCL.
- The format of ADCH and ADCL are interpreted differently depending on bit ADLAR.
- **Important:** When retrieving digital output, register ADCL must be read first, before register ADCH.

When bit ADLAR = 0 (Right Aligned)

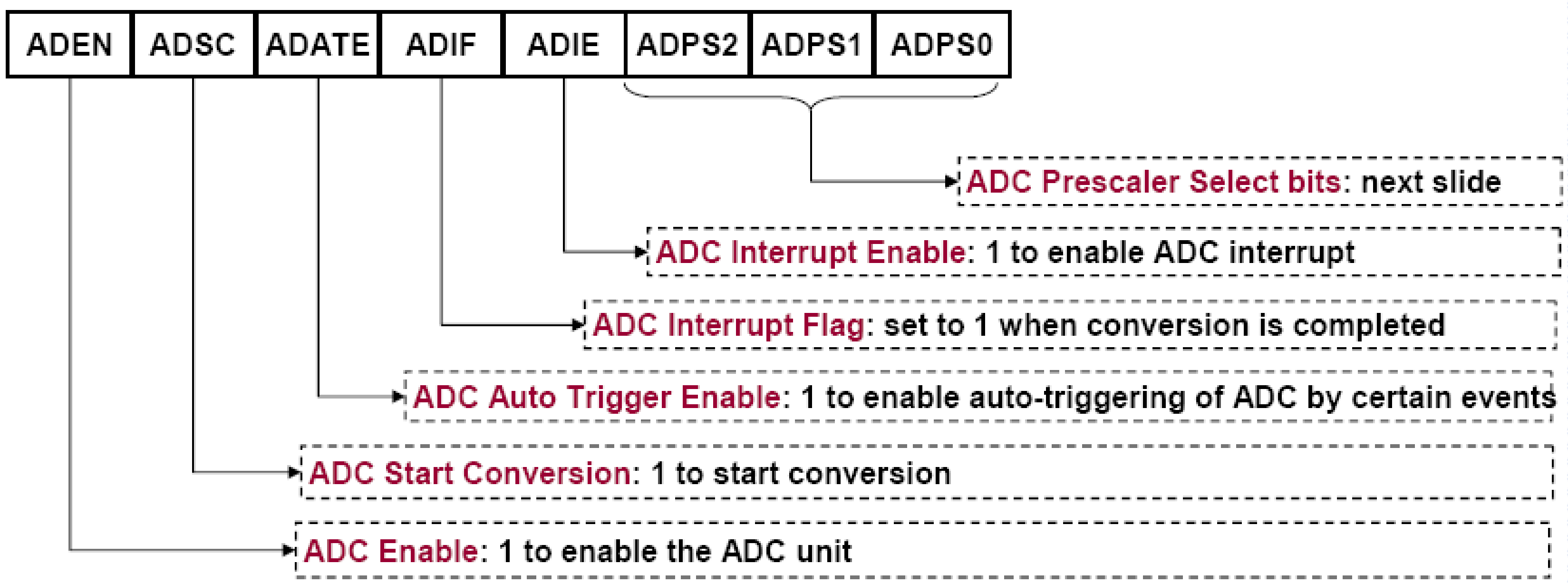
ADCH	#	#	#	#	#	#	ADC9	ADC8
ADCL	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0

When bit ADLAR = 1 (Left Aligned)

ADCH	ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2
ADCL	ADC1	ADC0	#	#	#	#	#	#

ADC Control and Status Register (ADCSRA)

- ADC unit can operate in two modes: manual or auto-trigger.
- In manual mode, set bit ADSC will start conversion.
- In auto-trigger mode, an predefined event will start conversion.



ADC clock

- The clock of the ADC is obtained by dividing the CPU clock and a division factor.
- There are 8 possible division factors, decided by the three bits {ADPS2, ADPS1, ADPS0}
- **Example**: Using internal clock of 1Mz and a ADC prescaler bits of ‘010’, the clock rate of ADC is: 1MHz/4 = 250KHz.

Table 85. ADC Prescaler Selections

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

Special Function IO Register (SFIOR)

- Three bits in register SFIOR specify the event that will auto-trigger an A-to-D conversion.

ADC Auto Trigger Source bits

ADTS2	ADTS1	ADTS0	-	ACME	PUD	PSR2	PSR10

Table 11.4: ADC Auto Trigger Source

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/Counter1 Compare Match B
1	1	0	Timer/Counter1 Overflow
1	1	1	Timer/Counter1 Capture Event

Steps to use the ADC

- **Step 1:** Configure the ADC using registers ADMUX, ADCSRA, and SFIOR.
  - What is the ADC source?
  - What reference voltage to use?
  - Align left or right the result in ADCH, ADCL?
  - Enable or disable ADC auto-trigger?
  - Enable or disable ADC interrupt?
  - What is the prescaler?
- **Step 2:** Start ADC operation
  - Write 1 to flag ADSC of register ADCCSRA.
- **Step 3:** Extract ADC result
  - Wait until flag ADSC becomes 0.
  - Read result from registers ADCL and then ADCH.

Example: Performing ADC

- Write C program that repeatedly performs ADC on a sinusoidal signal and displays the result on LEDs.
- Step 1: Configure the ADC
  - What is the ADC source? ADC0 (pin A.0)
  - What reference voltage to use? AVCC = 5V
  - Align left or right? Left, top 8-bit in ADCH
  - Enable or disable ADC auto-trigger? Disable
  - Enable or disable ADC interrupt? Disable
  - What is the prescaler? 2 (001)

0	1	1	0	0	0	0	0	
REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0	ADCMUX

1	0	0	0	0	0	0	1	
ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCCSRA

Example: adc.c (demo: adc.mp4)

```
#include<avr/io.h>
int main (void){
   unsigned char result;

   DDRB = 0xFF;  // set port B for output

   // Configure the ADC module of the ATmega16
   ADMUX = 0b01100000;  // REFS1:0 = 01     -> AVCC as reference,
                        // ADLAR   =  1     -> Left adjust
                        // MUX4:0  = 00000 -> ADC0 as input

   ADCSRA = 0b10000001; // ADEN = 1: enable ADC,
                        // ADSC = 0: don't start conversion yet
                        // ADATE = 0: disable auto trigger,
                        // ADIE  = 0: disable ADC interrupt
                        // ASPS2:0 = 001: prescaler = 2

   while(1){            // main loop
       // Start conversion by setting flag ADSC
       ADCSRA |= (1 << ADSC);

       // Wait until conversion is completed
       while (ADCSRA & (1 << ADSC)){;}

       // Read the top 8 bits, output to PORTB
       result = ADCH;

       PORTB = ~result;
   }
   return 0;
}
```

1

2

3

Using ADC interrupt

- In polling approach shown previously, we must check ADSC flag to know when result of an ADC operation is ready.
- The ADC unit can trigger an interrupt when ADC operation is completed.
- We need to enable ADC interrupt through ADIE flag in register ADCSRA.
- In the ISR, we can write code to read the ADC result from ADCL and then ADCH register.
- ADC interrupt is usually combined with auto-trigger mode

Example 2: ADC interrupt

- **Write interrupt-driven program to digitize a sinusoidal signal and display the result on LEDs.**

- **Step 1: Configure the ADC**
  - **What is the ADC source? ADC0**
  - **What reference voltage to use? AVCC = 5V**
  - **Align left or right? Left, top 8-bit in ADCH**
  - **Enable or disable ADC auto-trigger? Disable**
  - **Enable or disable ADC interrupt? Enable**
  - **What is the prescaler? 128 (slowest conversion)**
- **Step 2: Start ADC operation**
- **Step 3: In ISR, read and store ADC result.**

adc_int.c

```c
#include<avr/io.h>
#include<avr/interrupt.h>

volatile unsigned char result;

ISR(ADC_vect){
    result = ADCH; // Read the top 8 bits, and store in variable result
}

int main (void){
    DDRB = 0xFF;  // set port B for output

    // Configure the ADC module of the ATmega16
    ADMUX = 0b01100000;  // REFS1:0 = 01    -> AVCC as reference,
                         // ADLAR   =  1    -> Left adjust
                         // MUX4:0  = 00000 -> ADC0 as input

    ADCSRA = 0b10001111; // ADEN = 1: enable ADC,
                         // ADSC = 0: don't start conversion yet
                         // ADATE = 0: diable auto trigger,
                         // ADIE  = 1: enable ADC interrupt
                         // ASPS2:0 = 002: prescaler = 2

    sei();               // enable interrupt system globally
    while(1){            // main loop
        ADCSRA |= (1 << ADSC); // start a conversion

        PORTB = ~result;       // display on port B
    }
    return 0;
}
```

3

1

2

Example application of the ADC