Co-funded by the Erasmus+ Programme of the European Union

Lecture A5: Microcontroller Unit 2

Representation of number in processor

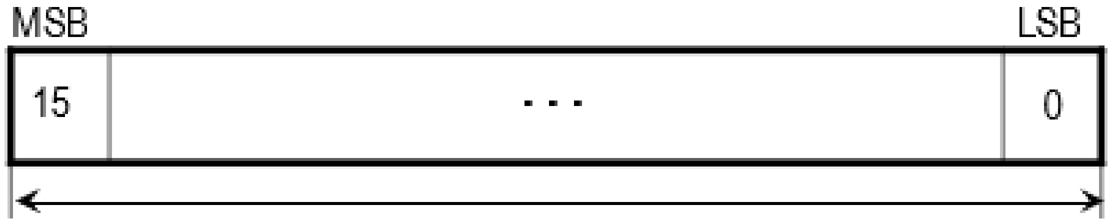
A/D convertor

Real time applications

Representation of number in CPU

- Basic integer data type: integer ... 16bit
  - ✓ Integer (int)
    - unsigned int – integer without sign (only positive numbers)
    - signed int – integer with sign

- Unsigned int:

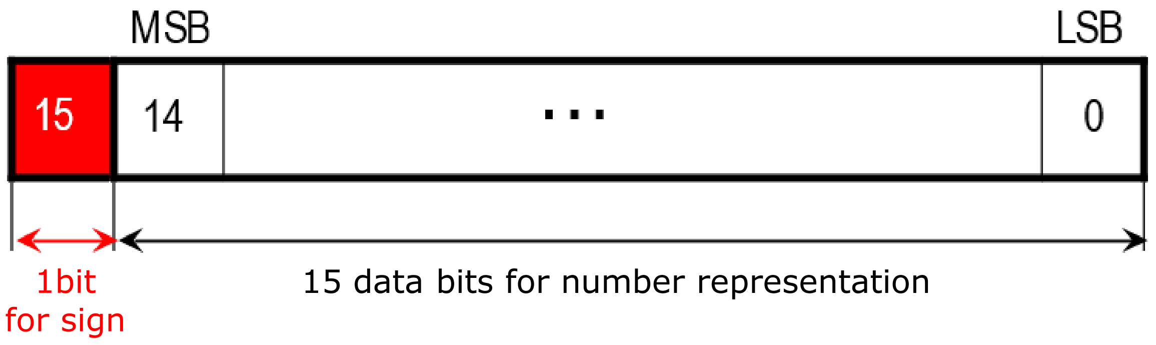


16 data bits for number representation:

Range	Binary	Hexadecimal	Decadic
	0000 0000 0000 0000b	0x0000	0
	1111 1111 1111 1111b	0xFFFF	2^{16} – 1=65535

Representation of number in CPU

- Signed integer:



Range	Binary	Hexadecimal	Decadic
	1000 0000 0000 0000b	0x8000	-32768
	0111 1111 1111 1111b	0x7FFF	32767

Signed number representations

- Sign-magnitude
- One’s complement
- Two’s complement

4-bit signed integer data type

Decadic	Sign-magnitude	One's complement	Two's complement
+7	0111	0111	0111
+6	0110	0110	0110
+5	0101	0101	0101
+4	0100	0100	0100
+3	0011	0011	0011
+2	0010	0010	0010
+1	0001	0001	0001
+0	0000	0000	0000
-0	1000	1111	-
-1	1001	1110	1111
-2	1010	1101	1110
-3	1011	1100	1101
-4	1100	1011	1100
-5	1101	1010	1011
-6	1110	1001	1010
-7	1111	1000	1001
-8	-	-	1000

One’s complement

- Positive number
  - ✓ Represented by value in data bits + sign bit is equal to 0
- Negative number
  - ✓ One’s complement of number X… X`
  - ✓ X` = ~X … bit negation of number X
- For example, number -7
  - ✓ X = 7 = 0111b
  - ✓ X`=~7=~0111b=1000b=-7

One’s complement

Decadic	Sign-magnitude	One’s complement	Two’s complement
+7	0111	0111	0111
+6	0110	0110	0110
+5	0101	0101	0101
+4	0100	0100	0100
+3	0011	0011	0011
+2	0010	0010	0010
+1	0001	0001	0001
+0	0000	0000	0000
-0	1000	1111	-
-1	1001	1110	1111
-2	1010	1101	1110
-3	1011	1100	1101
-4	1100	1011	1100
-5	1101	1010	1011
-6	1110	1001	1010
-7	1111	$\boxed{1000}$	1001
-8	-	-	1000

Two’s complement

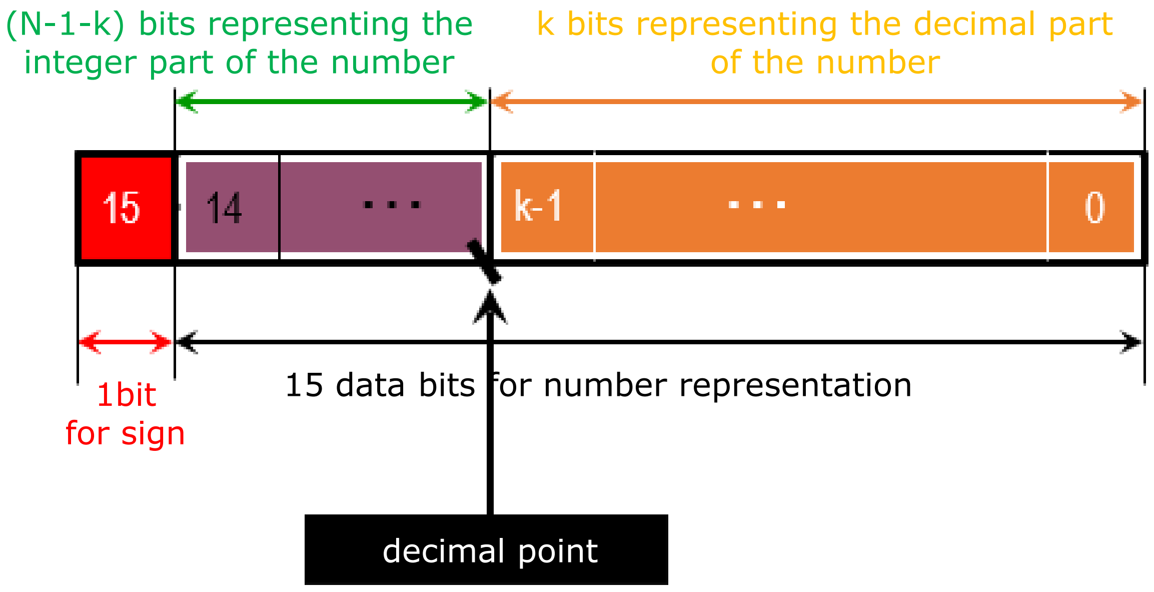
- Positive number
  - ✓ Represented by value in data bits + sign bit is equal to 0
- Negative number
  - ✓ Two’s complement of number X… X`
  - ✓ X` = ~X + 1… bit negation of number X + 1
- For example, number -7
  - ✓ X = 7 = 0111b
  - ✓ X`=~7+1=~0111b+1=1000b+1=1001b=-7
- Number -8
  - ✓ X=0=1000b
  - ✓ X`=~8+1=~1000b+1=0111b+1=1000b=-8

Two’s complement

Decadic	Sign-magnitude	One’s complement	Two’s complement
+7	0111	0111	0111
+6	0110	0110	0110
+5	0101	0101	0101
+4	0100	0100	0100
+3	0011	0011	0011
+2	0010	0010	0010
+1	0001	0001	0001
+0	0000	0000	0000
-0	1000	1111	-
-1	1001	1110	1111
-2	1010	1101	1110
-3	1011	1100	1101
-4	1100	1011	1100
-5	1101	1010	1011
-6	1110	1001	1010
-7	1111	1000	1001
-8	-	-	1000

Fixed point arithmetics

- The data bits are divided into 2 parts:
  - ✓ part that represents the integer part of the number
  - ✓ part that represents the decimal part of the number


(N-1-k) bits representing the integer part of the number
k bits representing the decimal part of the number
15
14
. . .
k-1
. . .
0
1bit for sign
15 data bits for number representation
decimal point

Fixed point arithmetics

- Let the variable X ∊ (-a; a).
- We typically have "int" with the range $\langle -32768; 32767\rangle$
- Objective is to project interval (-a; a) into the range $\langle -32768; 32767\rangle$

$$\boxed{X_f = k \,.\, X}$$

X_f … is a fixed-point representation of X,
K … is appropriately chosen scaling factor.

Fixed point arithmetics

- The constant "k" can be chosen arbitrarily (of course condition $k*X \leq 2^N$ must be true, where N is the number of data bits).
- "k" (scaling factor) should be chosen as a power of 2, because division by 2 (or its power) corresponds to a bit shift to the right.
- For example, representation of the number 0.5 in a fixed point:

$X = 0.5$

Scaling factor can be chosen e.g. $k = 2^{15} = 32768$.

$X_f = k * X = 30768 * 0.5 = 16384$

Check - reverse conversion to decimal (original) number:

$X = X_f / k = 16384/32768 = 0.5$

Fixed point data types

Data format: signed type	scaling factor	Representable range
Q0.15	$2^{15} = 32768$	⟨-1; 0,99997⟩
Q1.14	$2^{14} = 16384$	⟨-2; 1,99994⟩
Q2.13	$2^{13} = 8192$	⟨-4; 3,9999⟩
Q3.12	$2^{12} = 4096$	⟨-8; 7,9998⟩
Q4.11	$2^{11} = 2048$	⟨-16; 15,9995⟩
Q5.10	$2^{10} = 1024$	⟨-32; 31,999⟩
Q6.9	$2^{9} = 512$	⟨-64; 63,998⟩
Q7.8	$2^{8} = 256$	⟨-128; 127,996⟩
Q8.7	$2^{7} = 128$	⟨-256; 255,992⟩
…		
Q15.0	$2^{0} = 1$	⟨-32768; 32767⟩

Fixed point data types

- Suitable format for the number 2,5 and conversion of the number to a fixed point

Data format: signed type	scaling factor	Representable range
Q0.15	$2^{15} = 32768$	⟨-1; 0,99997⟩
Q1.14	$2^{14} = 16384$	⟨-2; 1,99994⟩
Q2.13	$2^{13} = 8192$	⟨-4; 3,9999⟩
Q3.12	$2^{12} = 4096$	⟨-8; 7,9998⟩
Q4.11	$2^{11} = 2048$	⟨-16; 15,9995⟩
Q5.10	$2^{10} = 1024$	⟨-32; 31,999⟩
Q6.9	$2^{9} = 512$	⟨-64; 63,998⟩
Q7.8	$2^{8} = 256$	⟨-128; 127,996⟩
Q8.7	$2^{7} = 128$	⟨-256; 255,992⟩
…		
Q15.0	$2^{0} = 1$	⟨-32768; 32767⟩

Fixed point data types

- Suitable format for the number 2,5 and conversion of the number to a fixed point

Data format: signed type	scaling factor	Representable range
Q0.15	$2^{15} = 32768$	⟨-1; 0,99997⟩
Q1.14	$2^{14} = 16384$	⟨-2; 1,99994⟩
Q2.13	$2^{13} = 8192$	⟨-4; 3,9999⟩

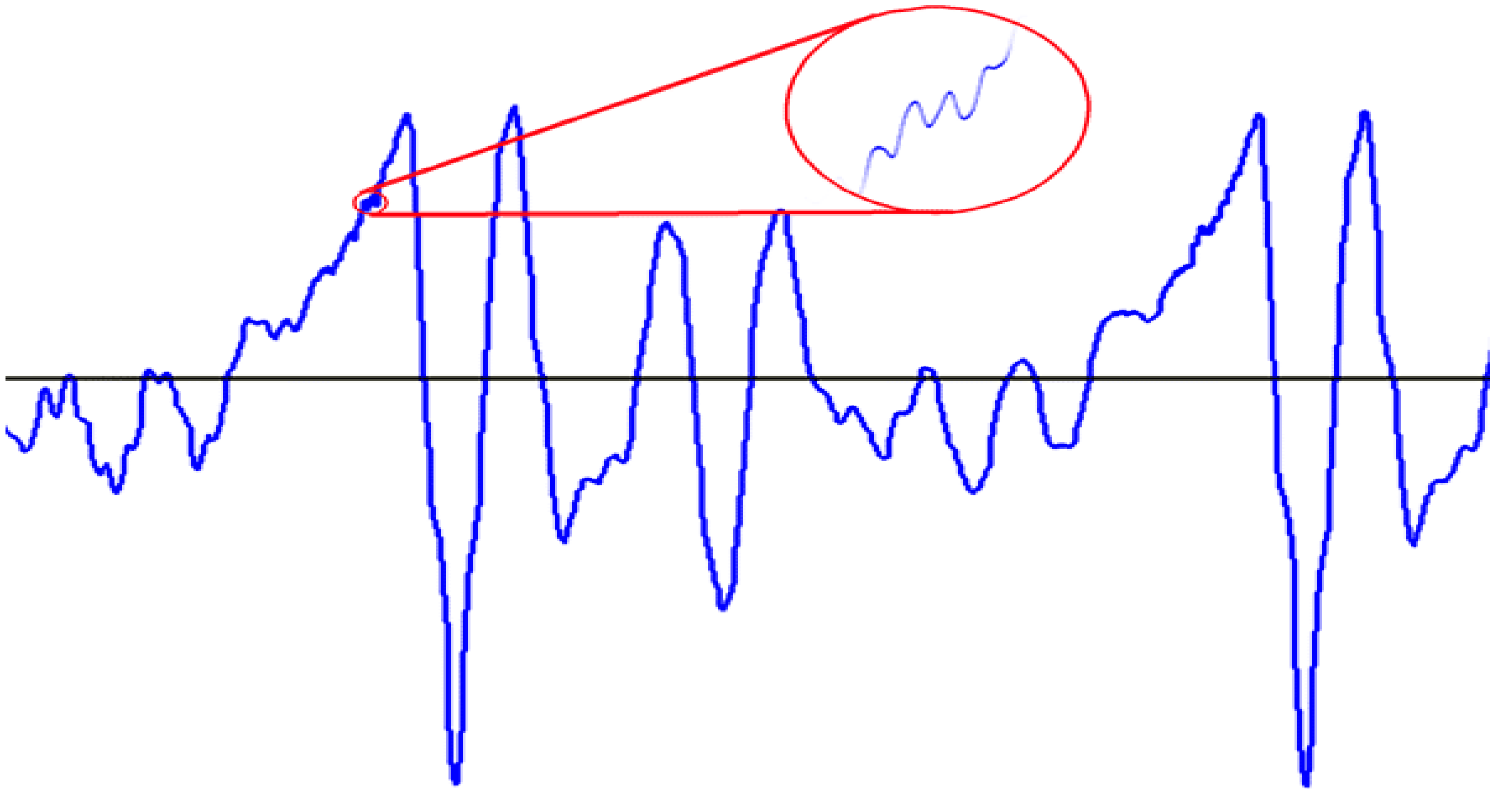
- Format Q2.13 was chosen => $k = 2^{13} = 8192$

- $X^f = k * X = 8192 * 2.5 = 20480$

- Check: $X = X_f / k = 20480 / 8192 = 2.5$

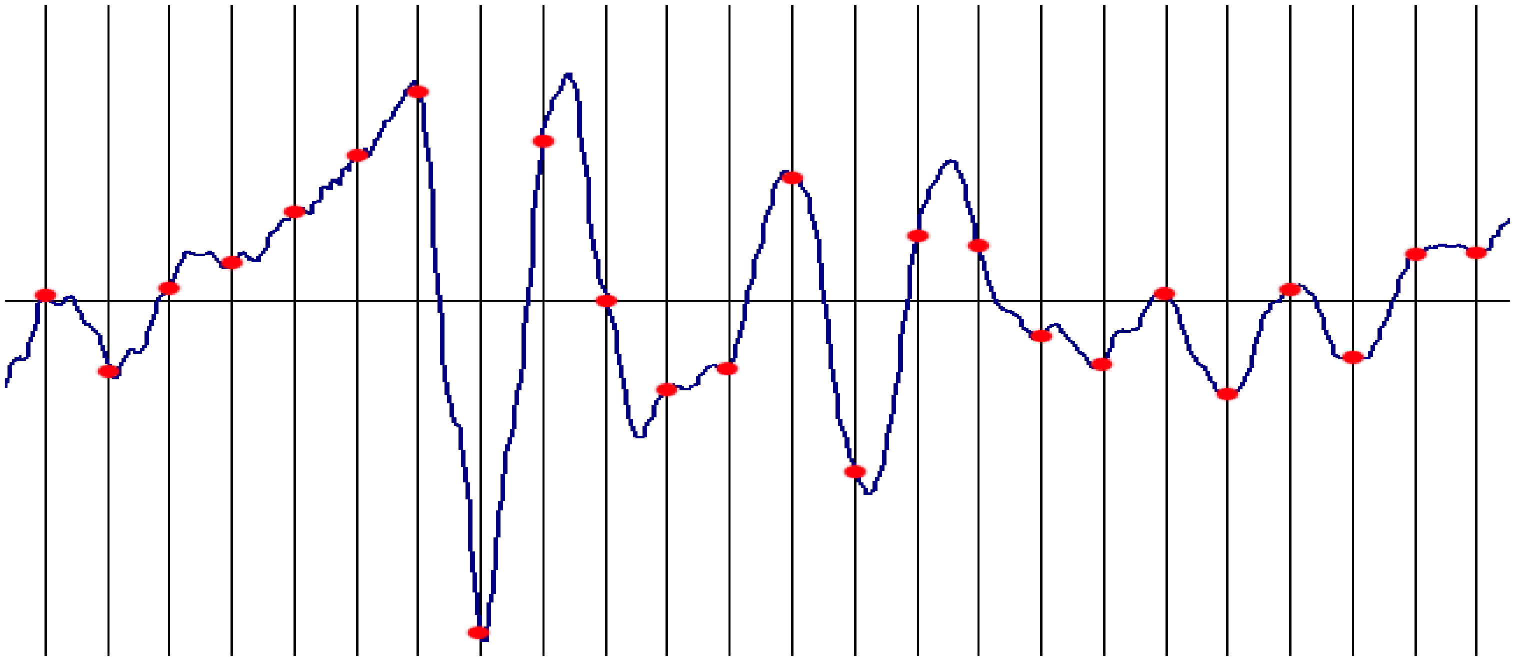
Analog to Digital (AD) conversion

- Sample of real analog signal



AD conversion - principle

- The conversion of a continuous signal to a discrete signal consists of two phases:
- 1. Sampling
  - ✓ We don't have an infinite amount of memory or an infinite ADC speed to record complete waveform
  - ✓ Scanning and recording of monitored values is necessary/possible only after certain time periods - SAMPLING



AD conversion - principle

- The conversion of a continuous signal to a discrete signal consists of two phases:

- 2. Quantization

  - ✓ We have limited converter accuracy available for sample storage.
  - ✓ The value of a sample can only be expressed by specific quantums - QUANTIFICATION

AD conversion - principle

- The conversion of a continuous signal to a discrete signal consists of two phases:

- 2. Quantization

  - ✓ We have limited converter accuracy available for sample storage.
  - ✓ The value of a sample can only be expressed by specific quantums - QUANTIFICATION

AD conversion - principle

- The conversion of a continuous signal to a discrete signal consists of two phases:

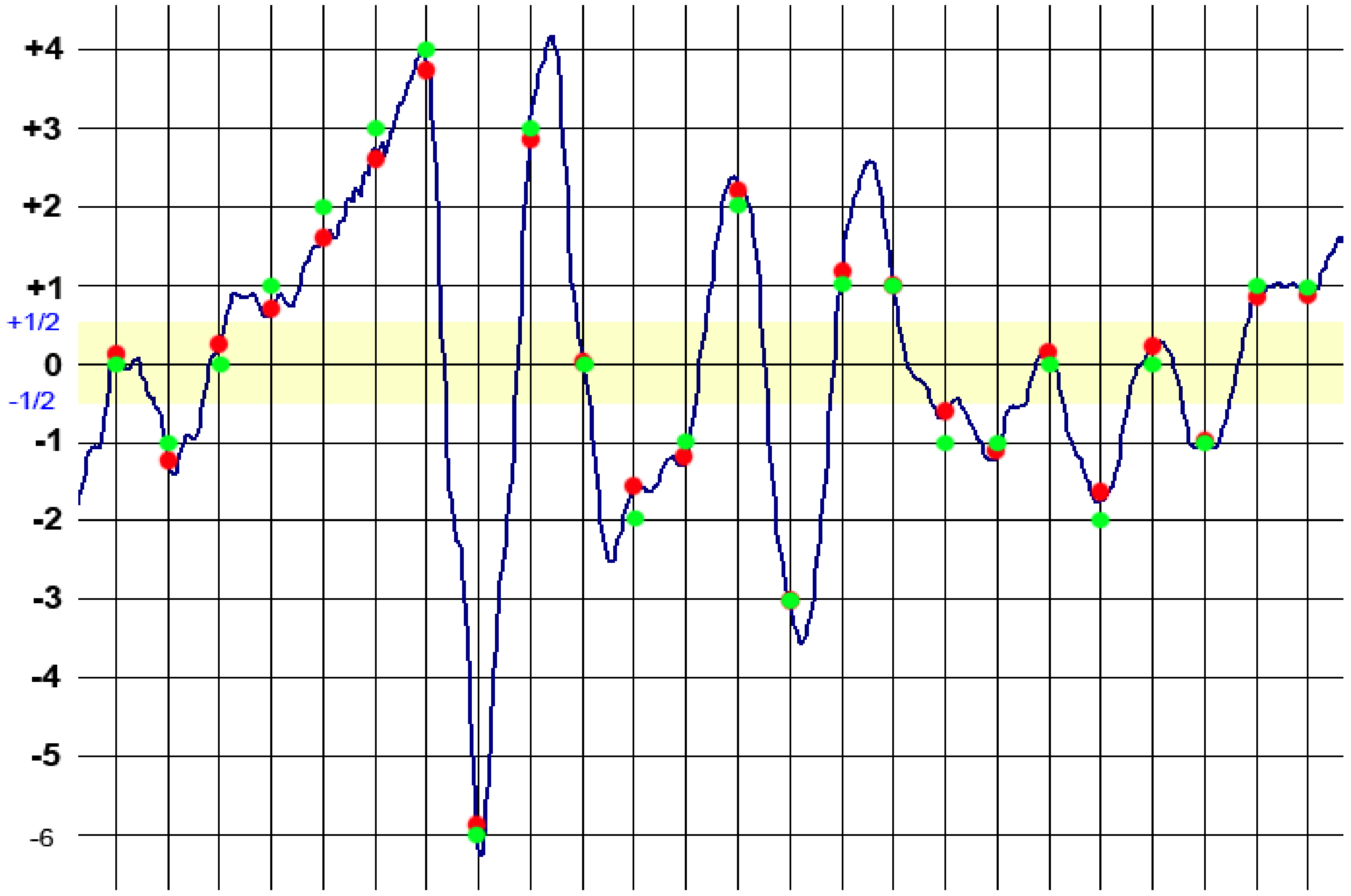
- 2. Quantization
  - ✓ To determine the value of a particular sample for quantization, the space around each value must be divided into tolerance bands of size (±1/2) quantization level.
  - ✓ Any sample that falls within a given tolerance band is assigned a value in the quantization process.
  - ✓ The number of quantization levels is given by the relation:

$$Q = 2^N$$

where N is the number of bits of the converter.

  - ✓ The magnitude of the quantization error is then given by distance between the quantized (digitized) and the original value of the sampled point.

AD conversion - principle



AD conversion - principle

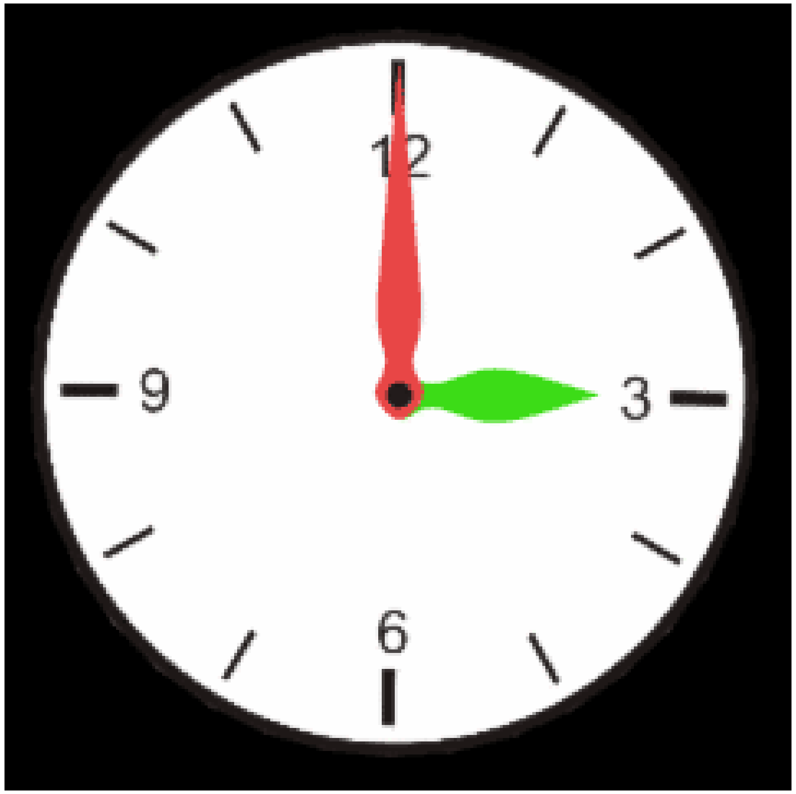
- The condition for the correct sampling rate is given by the theorem (Shannon-Kotelnikov, Nyquist–Shannon theorem):

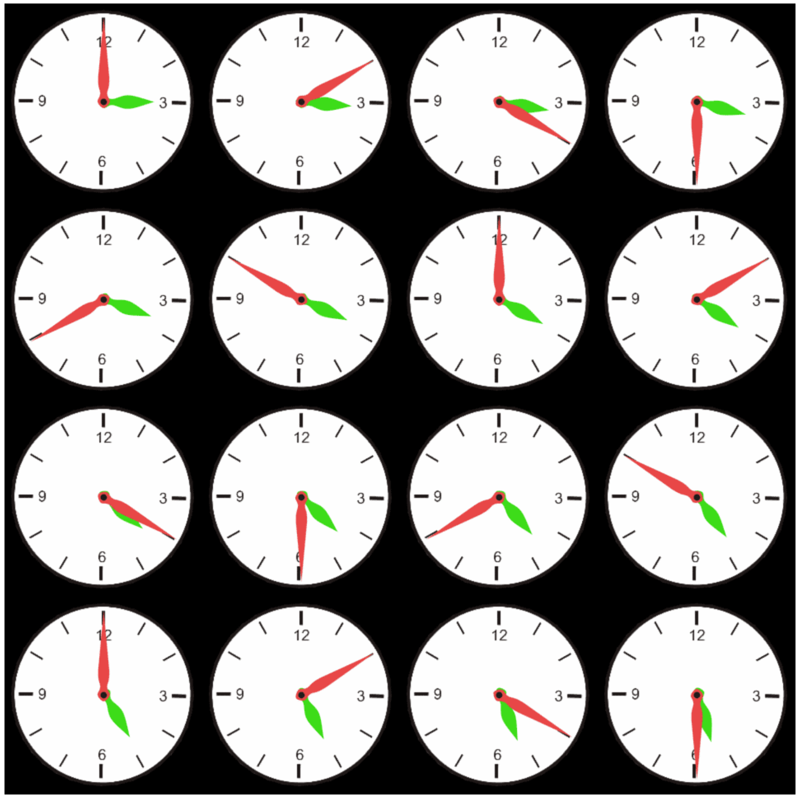
$$f_s \geq 2f_{max} \qquad \text{or} \qquad T_s = \leq \frac{1}{2} T_{max}$$

- Definition:
  - ✓ If we sample at least twice as fast as the highest frequency in the spectrum of the sampled signal, then no information about the frequency of the original signal is lost.
  - ✓ Only under this assumption can the continuous signal be reconstructed from the sampled signal without loss of information.
  - ✓ When this condition is not met, aliasing occurs - complete and irreversible distortion of the signal - especially loss of information about the frequency of the original signal.

AD conversion - aliasing

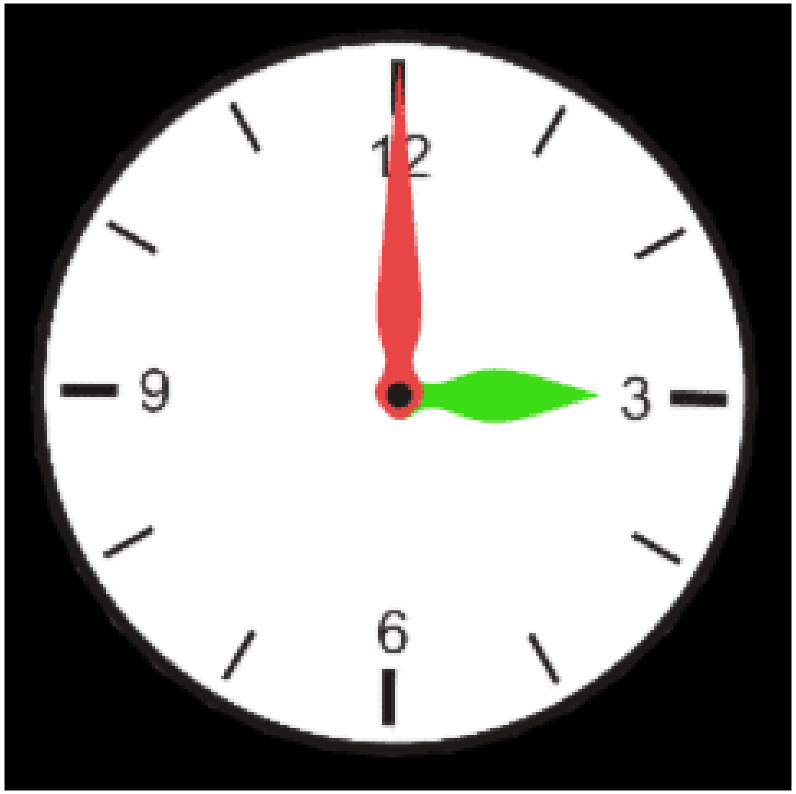
- Capturing rotational motion on film - clock capture
  - ✓ T_s=10 min, T_{max}=60 min, condition $T_s < 0.5.T_{max}$ is FULFILLED

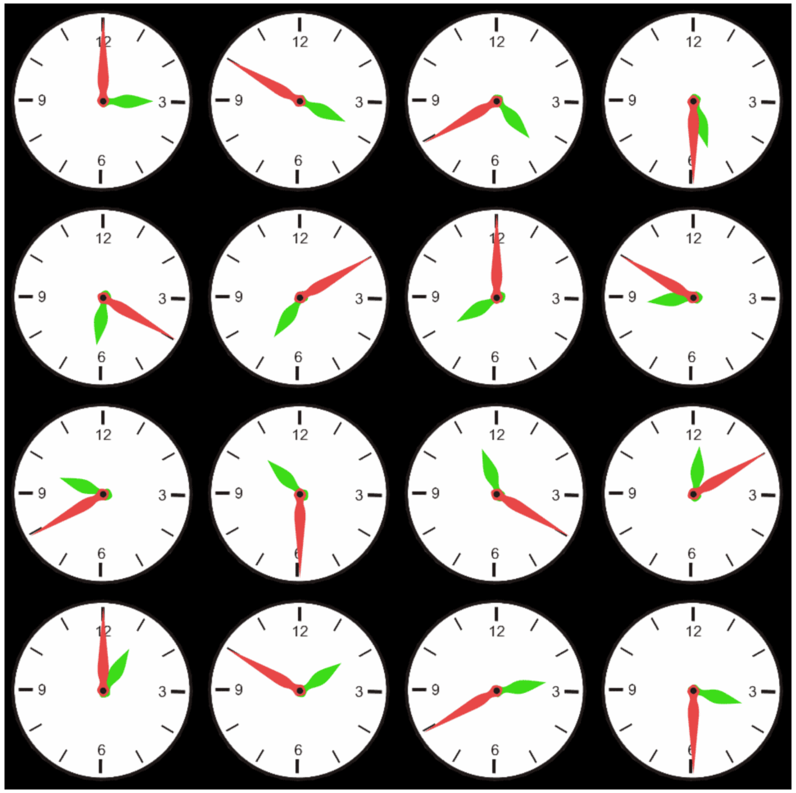




AD conversion - aliasing

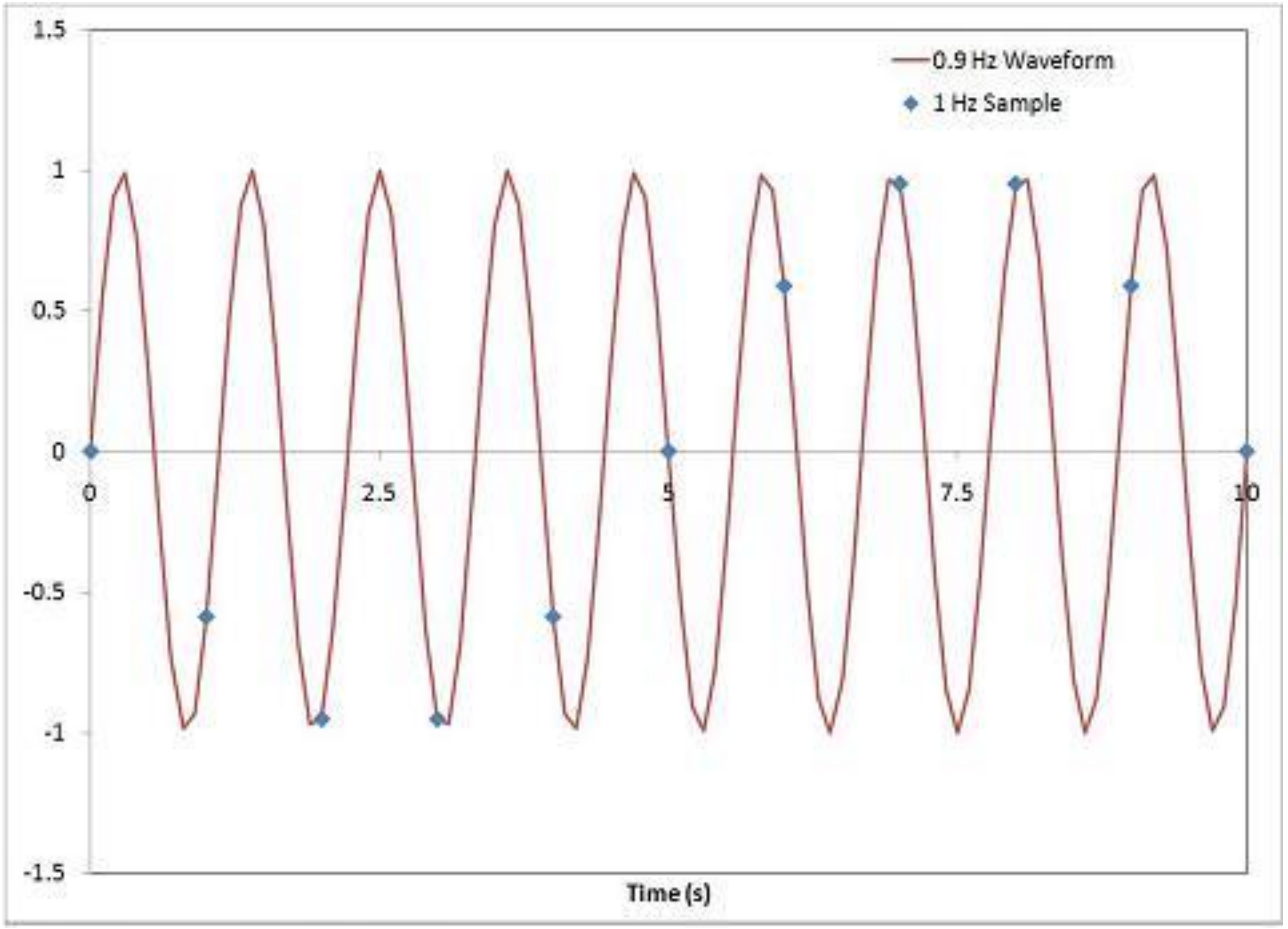
- Capturing rotational motion on film - clock capture
  - ✓ T_s=50 min, T_{max}=60 min, condition $T_s < 0.5.T_{max}$ is NOT FULFILLED





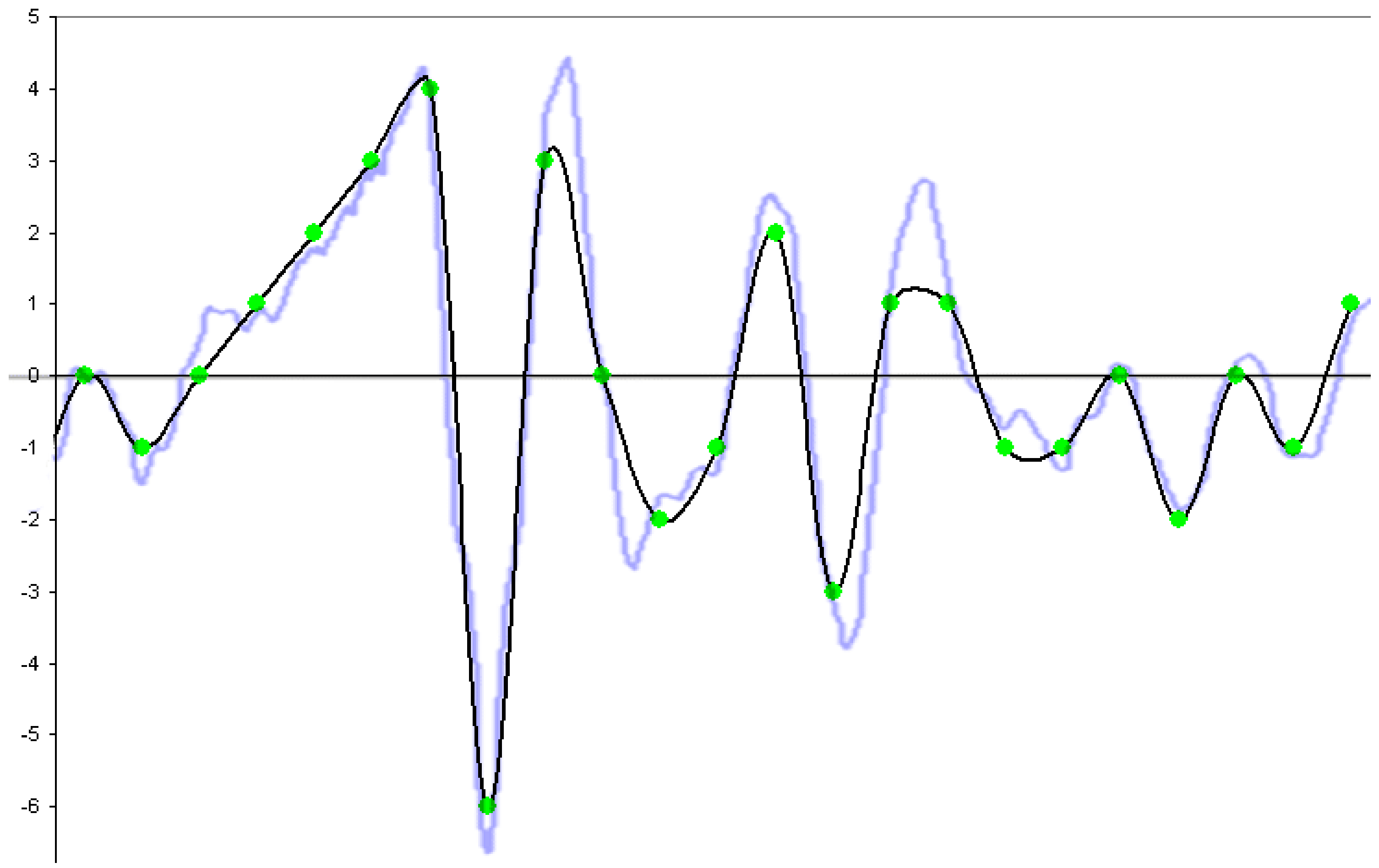
AD conversion - aliasing

- Digitalization of sine signal
  - ✓ 1Hz signal sampled by 0.9 Hz frequency



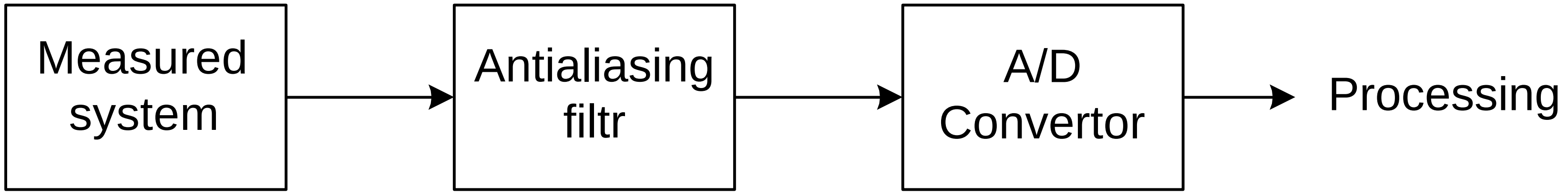
AD conversion - principle

- Reconstruction of the sampled signal, comparison with the original



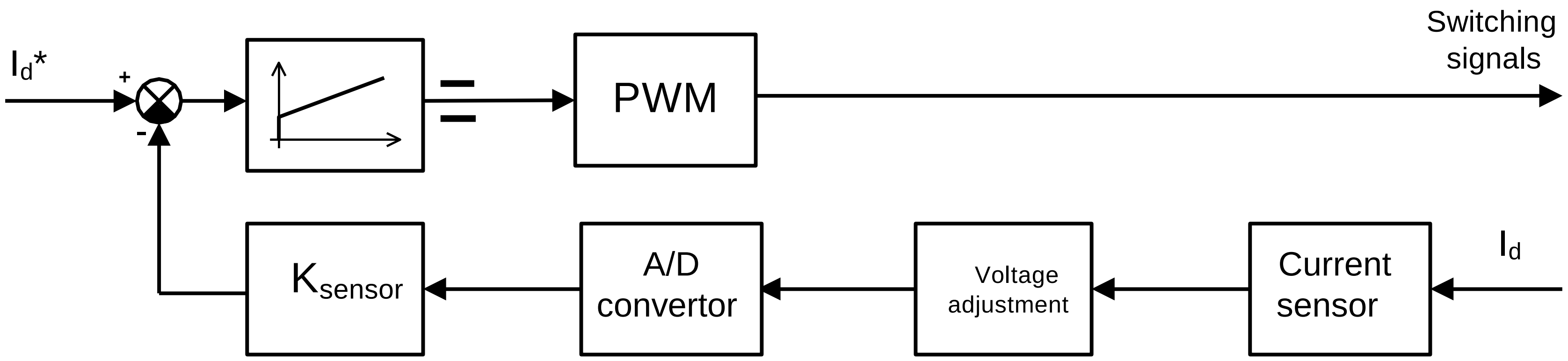
AD conversion – protection against aliasing

- Aliasing must be prevented because, if it occurs, its consequences are very difficult to eliminate.
- For this reason, an anti-aliasing filter is included before the A/D converter in most cases to filter out frequencies higher than the corresponding Shannon-Kotelnikov theorem (f > fmax).
- Filter is implemented in case of conventional A/D converters as an analogue low-pass filter (e.g. as RC circuit):



Integration into the control loop

- Principle diagram of a real loop current controller:



- The real measurement of system variables using the A/D converter is always supplemented by additional components.
- The measurement is most often performed by a voltage or current sensor.
- After the sensor output, there must be a voltage adjustment to the levels used by the converter.
- The value measured by the A/D converter must be converted to the system used by the microcontroller.

Parameters of A/D converters

- Number of bits of A/D convertor- resolution (accuracy increases with bit count):
  - ✓ 10b - 1024 values
  - ✓ 12b - 4096 values

- Transfer speed (transfer duration) - typically tens of ns to tens of μs
- Number of simultaneously converted channels
- Number of A/D converter channels on chip - typically 1-16
- Number of A/D converters on the DSP chip - typically 1-2
- Possibility of further configuration of the converter (triggering - synchronization, conversion speed ...)

A/D conversion synchronization

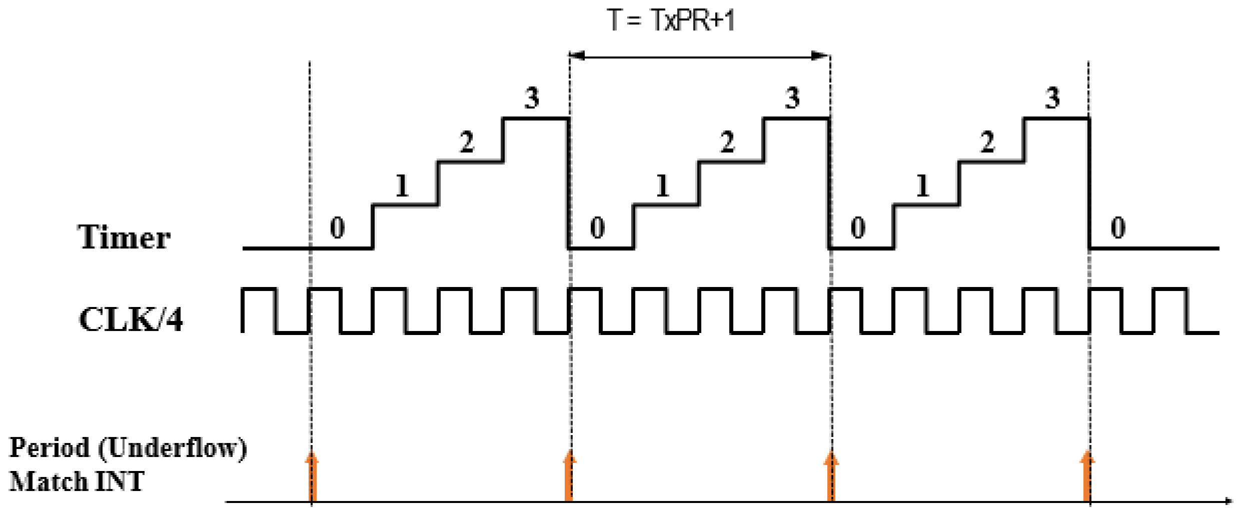
- In real applications, it is always necessary to synchronize the A/D conversion with the controlled system.

- The synchronization is related to the options for triggering the conversion:
  - ✓ By software
  - ✓ By hardware
  - ✓ By external signal

- In most practical applications it is advantageous to select the HW signal from the timer as the trigger source.

A/D conversion synchronization

- Interrupt generation synchronously with timer overflow:

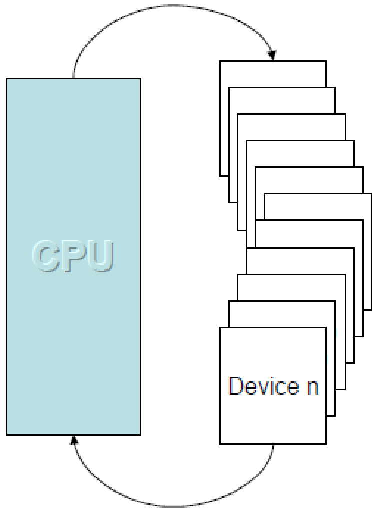


Real Time Operation System (RTOS)

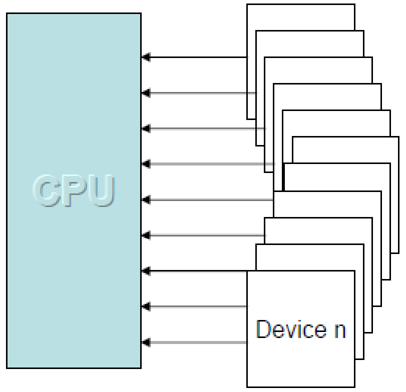
- Some definitions of RTOS - Real Time Operating System:
- A system in which the correctness of the output depends not only on the correctness of the result of the calculation, but also on the time at which the result is calculated.
- The time frame is given and limited (calculations must be performed within the defined time).
- A system that responds in a predictable manner to unpredictable external events.
- Interrupts are one of the most important building blocks of real-time applications!

Real Time Operation System (RTOS)

- The real-time work system can be of two types:
  - ✓ Device polling is a technique in which the CPU periodically determines whether a device requires servicing. This slows down the entire system, but some lower-end microprocessor series lack interrupts mechanism entirely and this is a way to partially replace it.
  - ✓ Device Interrupt request is a technique where each device that requires service sends an interrupt request to the CPU. This is where the issues of interrupt handling, priorities, etc. are then addressed.



Device Polling



Device Interrupt request

Interrupt

- Types of interrupts

- External (external source connected to processor pins, I/O)
- Internal (peripherals - timers, ADC, etc.)

- Maskable (can be enabled or disabled)
- Non-maskable (NMI - are always enabled, e.g. RESET)

- HW
- SW

Interrupt

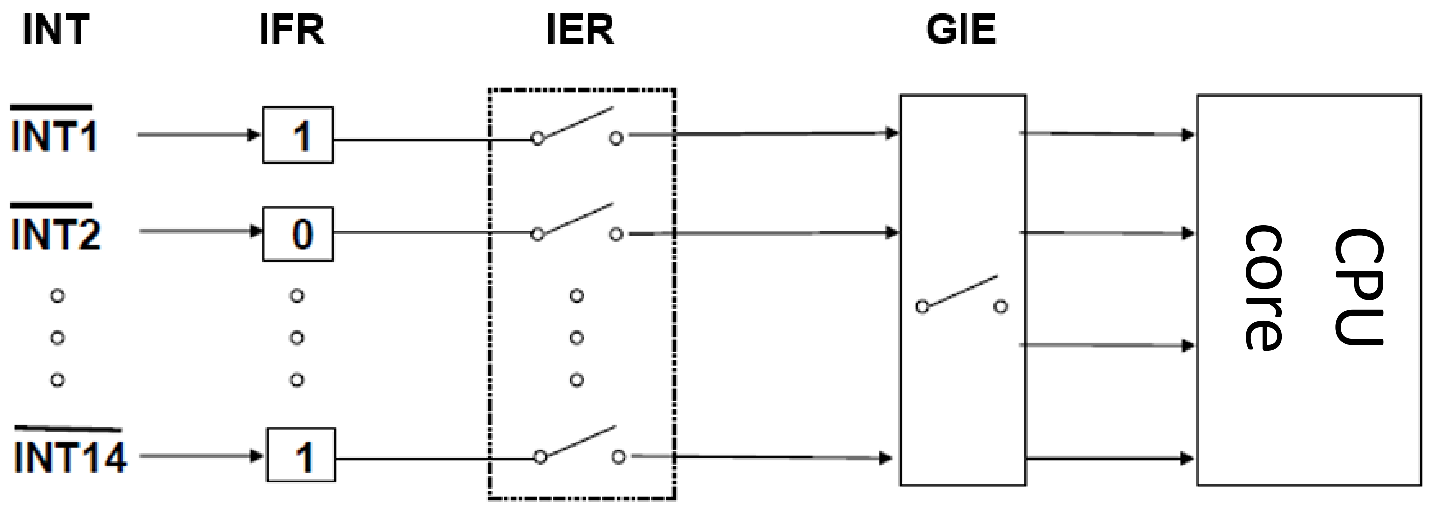
- Definition of basic terms:
  - ✓ **An interrupt** is a method for asynchronous event handling where the processor interrupts the execution of the main program, executes the interrupt handler, and then resumes the previous operation (returns to the main program where it left off).
  - ✓ **Interrupt handler** (abbreviated ISR, Interrupt Service Routine) is a special function that is triggered by an interrupt.
  - ✓ **A Programmable Interrupt Controller** (PIC) is a circuit that allows incoming interrupts to be handled according to their set priority. In microprocessors, this is either fixed according to the source of the interrupt or user adjustable.

Interrupt

- **Interrupt Latency** - the time between the interrupt request and the start of the first interrupt handler instruction. This time takes min. units of machine cycles, but is generally affected by the following operations:
  - ✓ Evaluation of priorities, masks, interrupt controller request processing.
  - ✓ The processor processes the ISR with the same or higher priority, the latency then increases depending on the length of this processing.
  - ✓ Storing the context (important registers) and the return address, possibly optional registers or the whole memory section (by the user in the program).
  - ✓ Time to jump to the interrupt handler (execute commands at the interrupt address).

Interrupt

- Typical HW interrupt processing flow:
  - ✓ Interrupts are handled according to their priority. When multiple interrupts of the same priority are serviced, they are serviced in the order in which they were generated.

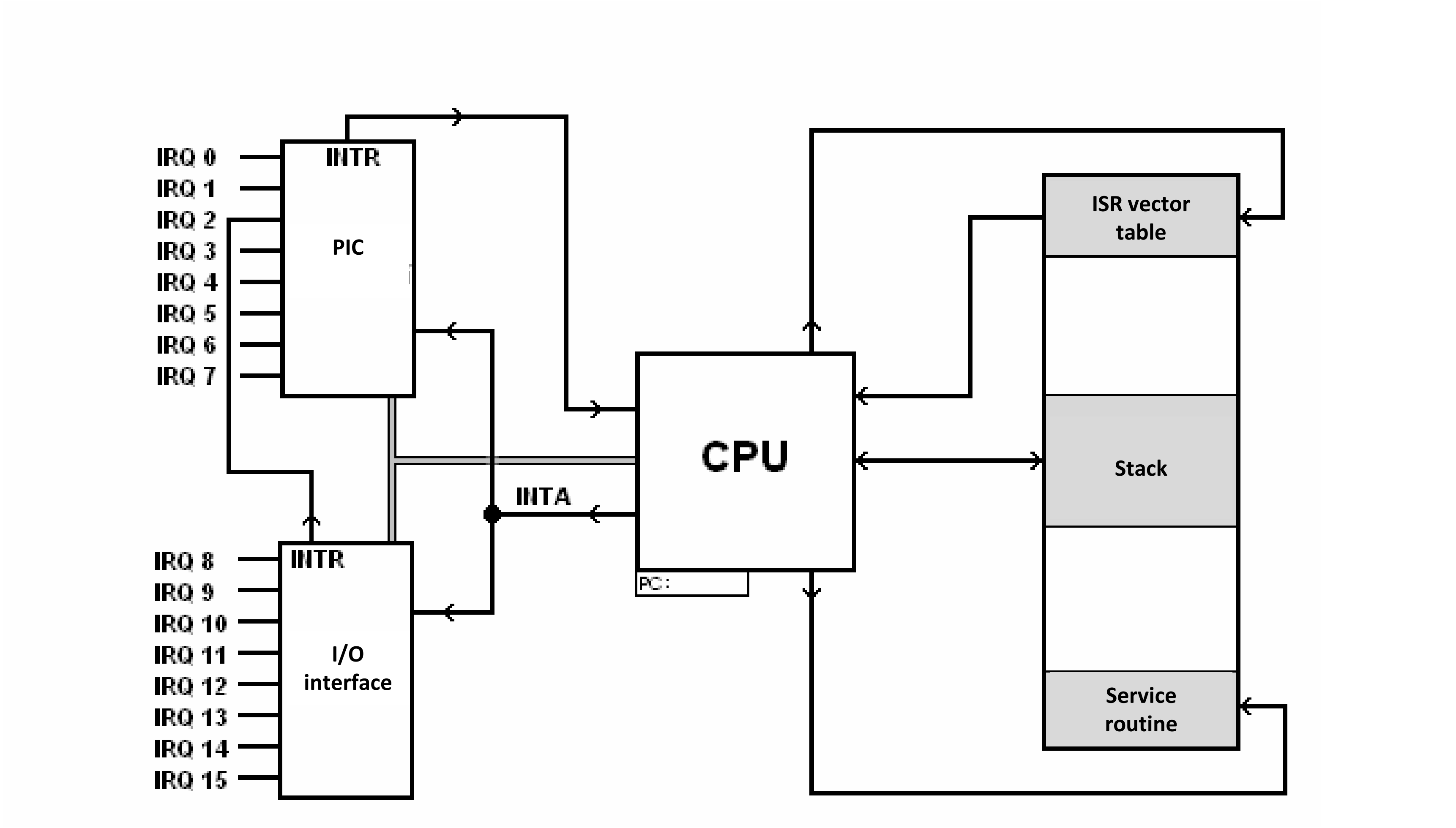


Interrupt

- Interrupt service flow:
  - ✓ Saving the context and return address into a stack or a reserved part of the memory - significant system registers are saved; it is possible to save any registers or even a part of the memory by program (solved by the programmer).
  - ✓ Actual ISR program - program code (solved by the programmer).
  - ✓ Clearing the interrupt flag - by writing log.1 to the corresponding bit (solved by the programmer).
  - ✓ Restore context and PC from stack, exit ISR, return to main program.

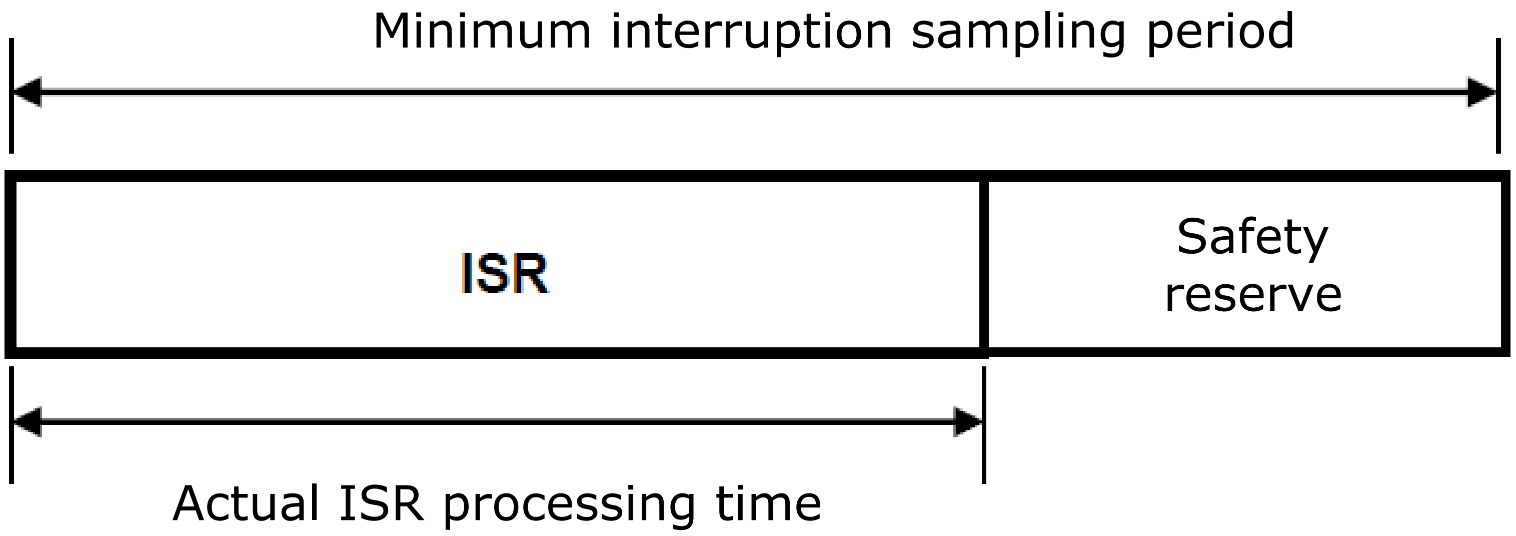
Interrupt

- Interrupt service flow



Interrupt

- Number of interruptions and their coordination:
  - ✓ It is necessary to minimize the number of interruptions and the duration of each ISR
  - ✓ When using multiple interrupts, it is necessary to prioritize them correctly!
- How to choose the correct interrupt duration?


Minimum interruption sampling period
ISR
Safety reserve
Actual ISR processing time