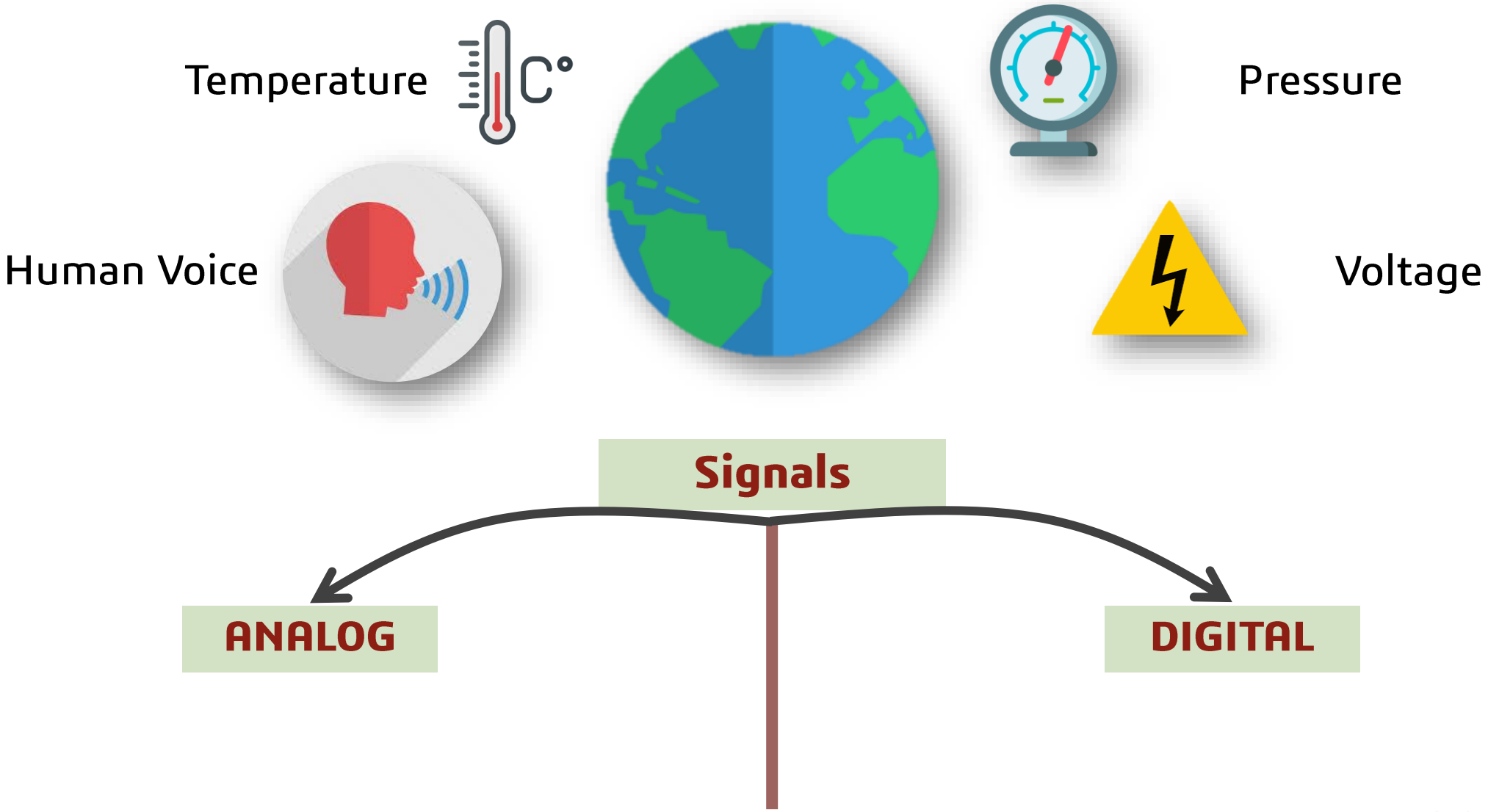
ADC

ANALOG TO DIGITAL CONVERTER

AMIT

Real world data



ANALOG SIGNALS

- An analog signal is a continuous signal that represents physical measurements
- It uses a continuous range of values that help you to represent information
- Analog signal doesn't offer any fixed range
- It always has a value along time

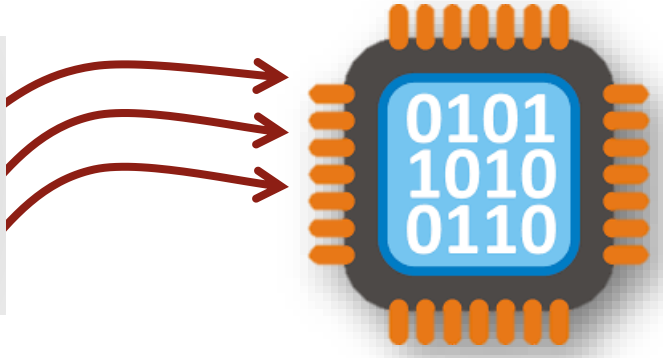
DIGITAL SIGNALS

- Digital signals are time separated signals which are generated using digital modulation
- Digital signal uses discrete 0 and 1 to represent information
- Digital signal has a finite number, i.e., 0 and 1
- Time concept in this signal only exists at start of cycle

AMIT

Introduction to ADC

Microcontroller will receive **analog signals**



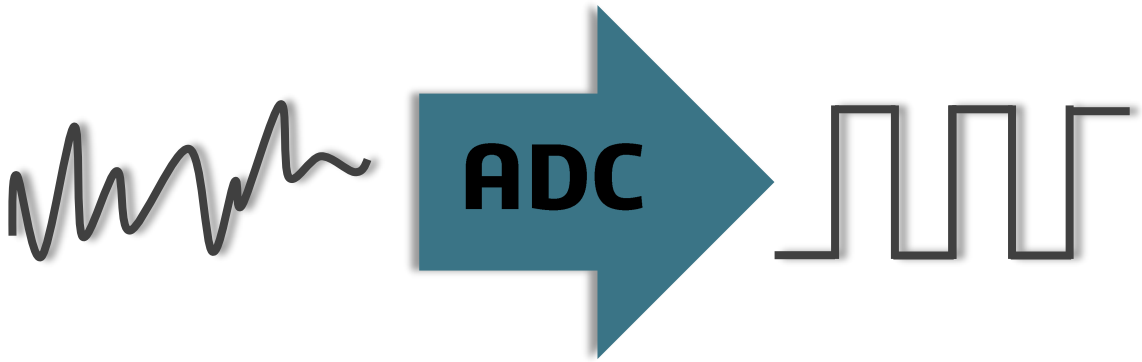
SO

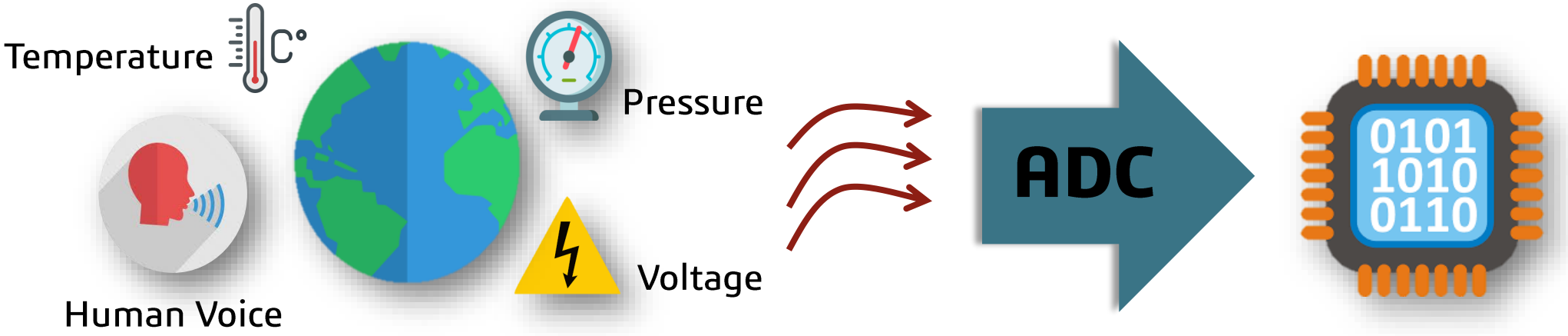
What if we need to get some non-digital data in the microcontroller

The Microcontroller have to deal with digital information “They only understand ‘0’ or ‘1’ values

AMIT

ANALOG TO DIGITAL CONVERTER

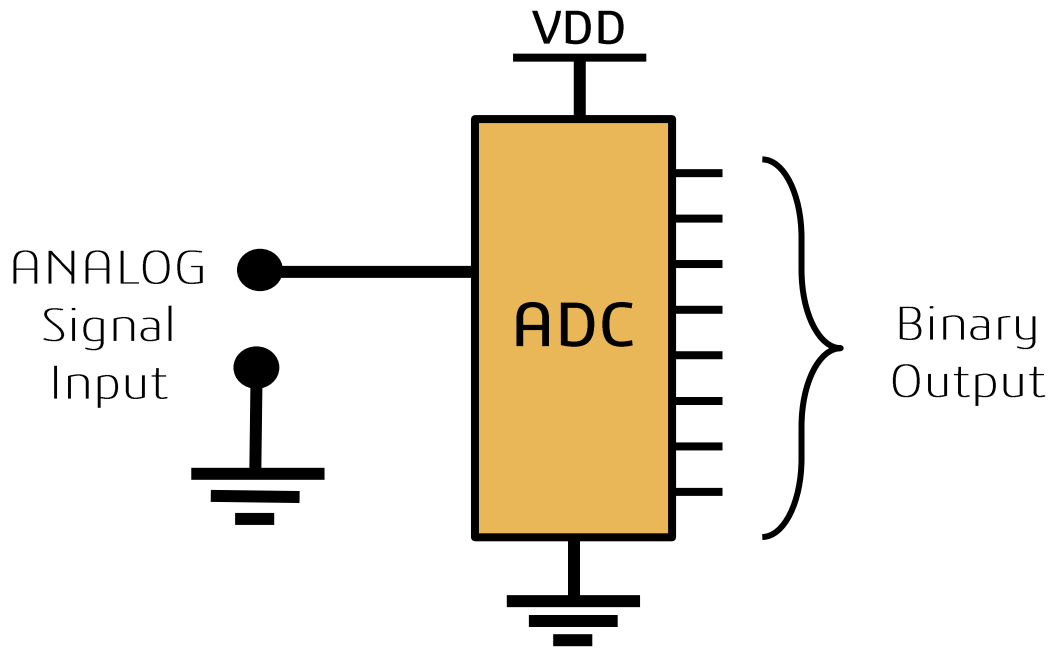




AMIT

Digitization progress

- To convert the sampled signal into a digital value
- **It depends on two mandatory parameters**
  - **Resolution**:
    It is the number of bits for each sample
  - **Maximum Voltage**:
    It is the maximum voltage can be sampled, and all bits' values of sample become all ones

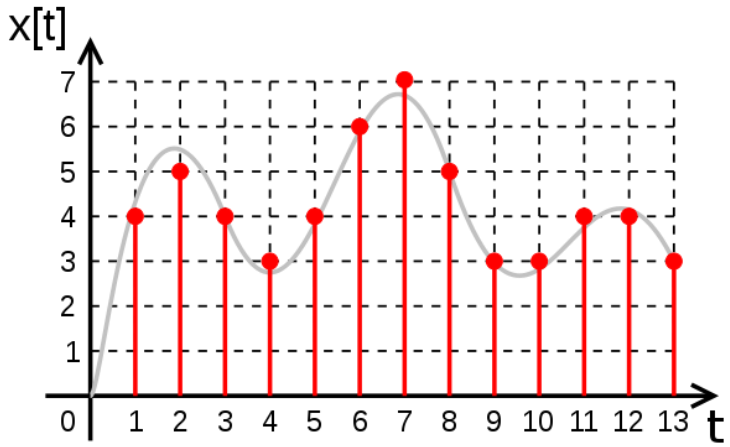


Sampling → **Quantization**

AMIT

Sampling

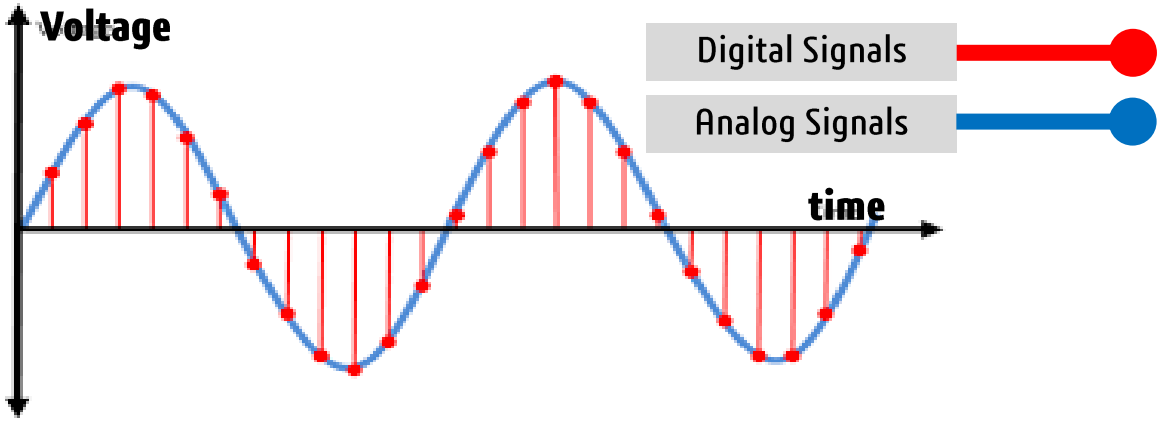
To convert a continuous signal to digital signal “analog to digital”, the analog signal must be sampled at specific rate



Nyquist – Shannon sampling theory:

It states that the sampling rate must be higher than or equal double of frequency of analog signal.

If the sampling rate equal the analog frequency, the reads will be appeared as if it is DC signal not a sine wave



AMIT

Quantization Laws

Step Voltage (Step Size) $= \dfrac{\text{Maximum Voltage}}{2^{\text{Resolution}}}$

Analog Voltage = **Digital Value** X **Step voltage**

= **Digital Value** X $\dfrac{\text{Maximum Voltage}}{2^{\text{Resolution}}}$

Quantization Error the difference between real analog volt and conversion result

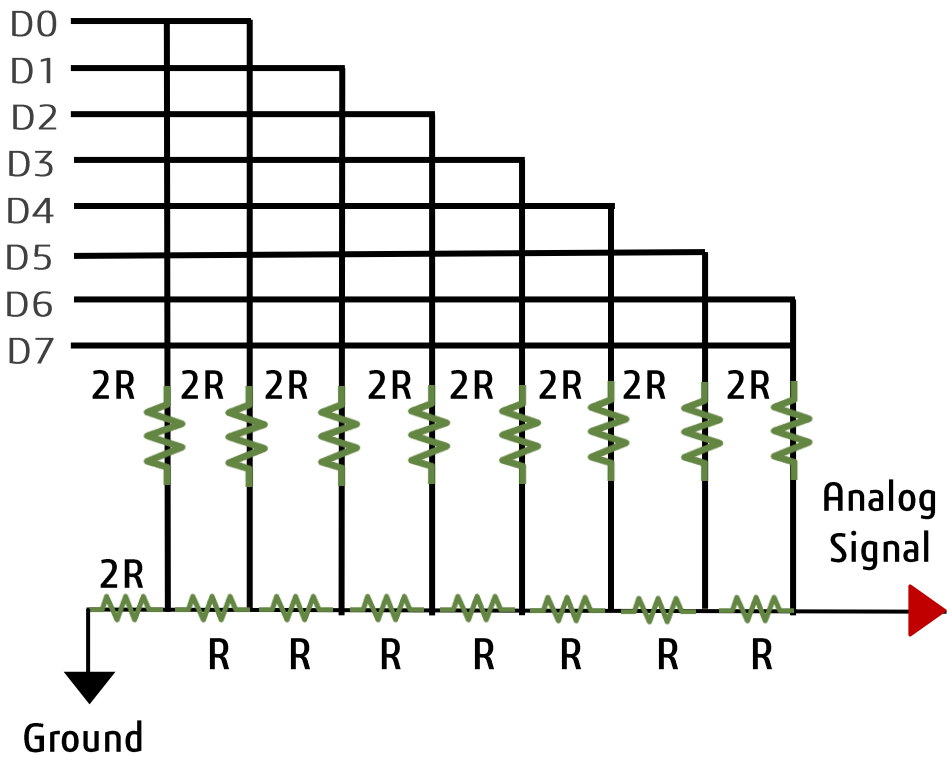
Quantization Error = **One Step Voltage**

AMIT

Digital to Analog Converter

R2R DAC

- It is a simple circuit which can convert the digital value to analog value, depending on Kirchhoff's Law
- The value of resistor 'R' does not care, but the value of '2R' must be doubled of 'R'



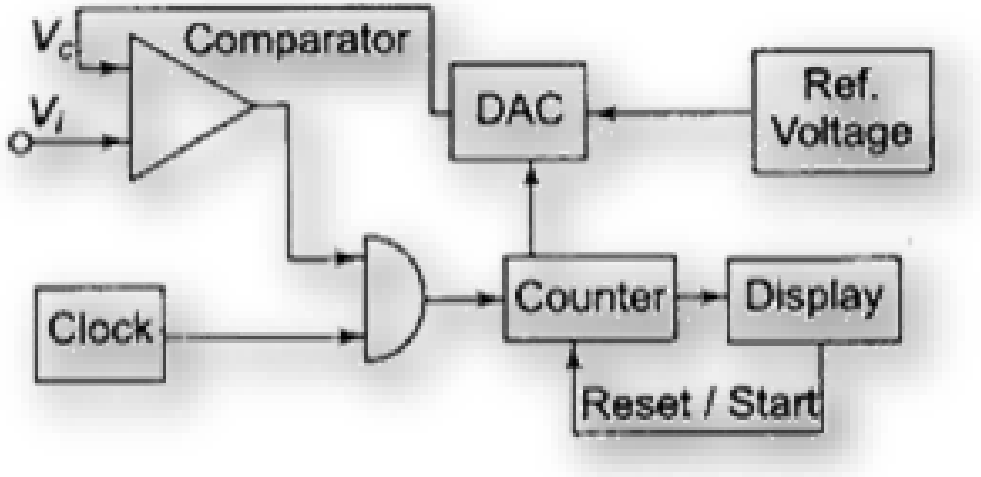
- This circuit can be used individually to convert digital values, or integrated into the Analog to Digital circuit to apply the quantization

AMIT

Analog to Digital Converter

RAMP ADC

- It depends on a counter register, increased by one every high-level clock



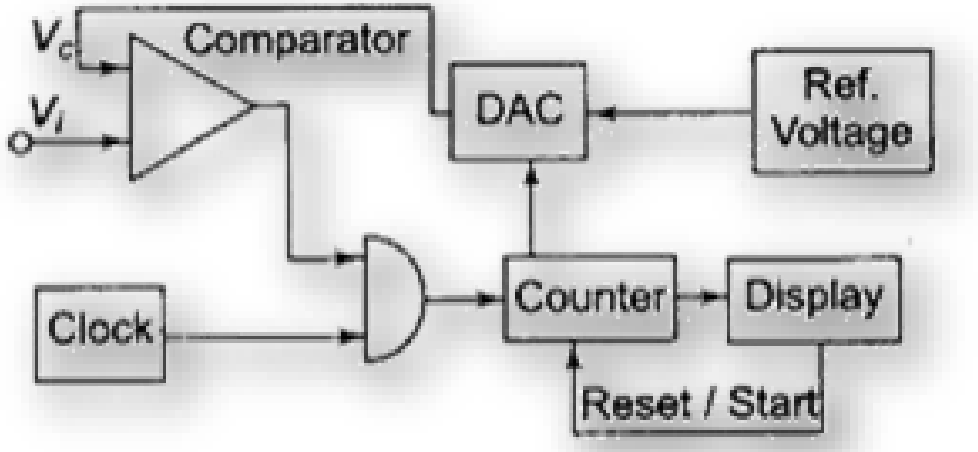
- Then, the value exists into this register will be converted to analog signal, then it will be compared with the input signal by **Comparator**, If the input signal is higher than the **DAC signal**, the **Comparator** output will be **ONE**, and it will be **ANDed** with the high level of clock, then the clock will trigger the register and increase it by **ONE**, and so on

AMIT

Analog to Digital Converter

RAMP ADC

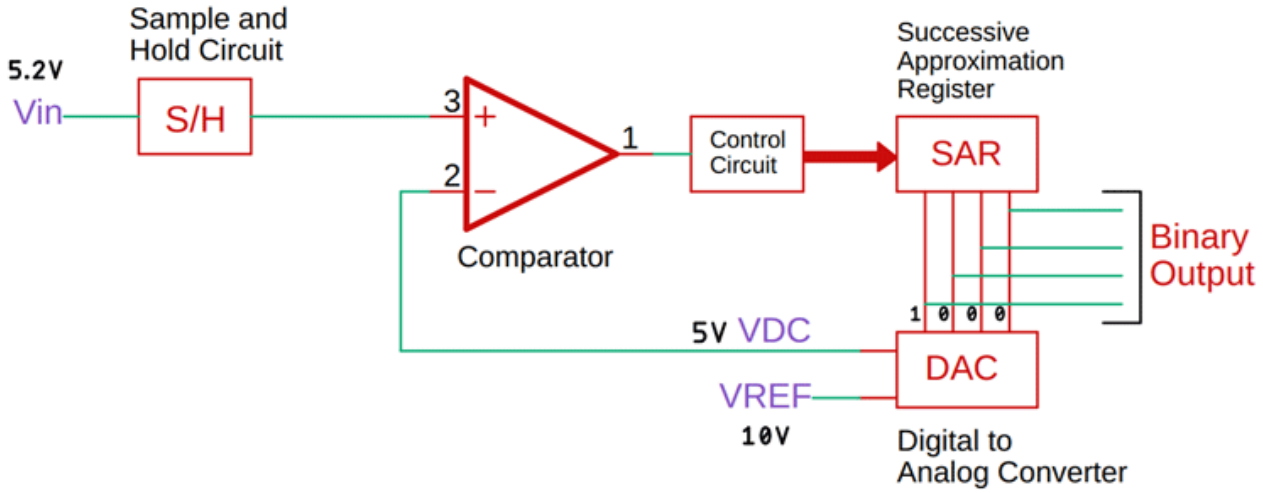
- Until the output of Comparator becomes ZERO, if the input signal becomes lower the DAC signal, the register will not be triggered again, and the value into it became the equivalent value of the input signal
- This ADC type takes many clock- cycles for conversion.
- In the best-case scenario, if the value is low, is will take little clock-cycles for conversion
- In the worst-case scenario, if value is near to MAX voltage, it will take many clock-cycles for conversion



Analog to Digital Converter

SAR ADC

- It is called a Successive Approximately Register ADC
- It depends on SAR register, which its reset value “default value” is ONE into the MSB into the register
- Like the previous, the value exists into this register will be converted to analog signal by DAC, then it will be compared with the input signal by Comparator, if the input signal is higher than the DAC signal, the comparator output will be ONE, and it will trigger the control unit of ADC to take an action.



Analog to Digital Converter

SAR ADC

The action depends on the output value of comparator:

ONE:
It means that the input signal voltage is higher than DAC Signal.
So, the action is : **RESET & SET** reset currently bit and set the next bit.

ZERO:
It means that the input signal voltage is higher than DAC Signal.

So, the action is : **KEEP & SET** keep currently bit and set the next bit

AMIT

Analog to Digital Converter

SAR ADC

Example If we have **SAR ADC** has 5 max-volt and has 8-bit resolution, and it converts 1.86 input-signal volt ??

Step Voltage $= \frac{5000 \text{ mv}}{2^8} =$ **19.53125** mv

128	64	32	16	8	4	2	1
1	0	0	0	0	0	0	0

At first, **SAR** = **0b10000000**, So **DAC Volt** = **19.53125** * **128** = **2500** mv

2500 mv is **higher** than input signal, so the currently bit will be reset, and the next bit will be set, **SAR** = **0b01000000**

AMIT

Analog to Digital Converter

SAR ADC

Example If we have **SAR ADC** has 5 max-volt and has 8-bit resolution, and it converts 1.86 input-signal volt ??

Step Voltage $= \frac{5000 \text{ mv}}{2^8} =$ **19.53125** mv

128	64	32	16	8	4	2	1
0	1	0	0	0	0	0	0

At first, **SAR** = **0b10000000**, So **DAC Volt** = **19.53125** * **128** = **2500** mv

2500 mv is **higher** than input signal, so the currently bit will be reset, and the next bit will be set, **SAR** = **0b01000000**

DAC Volt = **19.53125** * **64** = **1250** mv

AMIT

Analog to Digital Converter

SAR ADC

128	64	32	16	8	4	2	1
0	1	1	0	0	0	0	0

1250 mv is **lower** than the input signal, so the currently bit will be kept, and the next bit will be set, **SAR** = **0b01100000**

DAC Volt = **19.53125** * **96** = **1875** mv

AMIT

Analog to Digital Converter

SAR ADC

128	64	32	16	8	4	2	1
0	1	0	1	0	0	0	0

1250 mv is **lower** than the input signal, so the currently bit will be kept, and the next bit will be set, **SAR** = **0b01100000**

→ **DAC Volt** = **19.53125** * **96** = **1875** mv

1875 is higher than input signal, so the currently bit will be reset, and the next bit will be set, **SAR** = **0b01010000**

→ **DAC Volt** = **19.53125** * **80** = **1562.5** mv

AMIT

Analog to Digital Converter

SAR ADC

128	64	32	16	8	4	2	1
0	1	0	1	1	0	0	0

1250 mv is **lower** than the input signal, so the currently bit will be kept, and the next bit will be set, **SAR** = **0b01100000**

→ **DAC Volt** = **19.53125** * **96** = **1875** mv

1875 is higher than input signal, so the currently bit will be reset, and the next bit will be set, **SAR** = **0b01010000**

→ **DAC Volt** = **19.53125** * **80** = **1562.5** mv

1562.5 is lower than the input signal, so the currently bit will be kept, and the next bit will be set, **SAR** = **0b01011000**

→ **DAC Volt** = **19.53125** * **88** = **1718.5** mv

AMIT

Analog to Digital Converter

SAR ADC

128	64	32	16	8	4	2	1
0	1	0	1	1	1	0	0

1718.75 mv is **lower** than the input signal, so the currently bit will be kept, and the next bit will be set, **SAR** = **0b01011100**

DAC Volt = **19.53125** * **92** = **1769.875** mv

AMIT

Analog to Digital Converter

SAR ADC

128	64	32	16	8	4	2	1
0	1	0	1	1	1	1	0

1718.75 mv is **lower** than the input signal, so the currently bit will be kept, and the next bit will be set, **SAR** = **0b01011100**

→ **DAC Volt** = **19.53125** * **92** = **1769.875** mv

1769.875 mv is **Lower** than input signal, so the currently bit will be reset, and the next bit will be set, **SAR** = **0b01011110**

→ **DAC Volt** = **19.53125** * **94** = **1835.9375** mv

AMIT

Analog to Digital Converter

SAR ADC

128	64	32	16	8	4	2	1
0	1	0	1	1	1	1	1

1718.75 mv is **lower** than the input signal, so the currently bit will be kept, and the next bit will be set, **SAR** = **0b01011100**

→ **DAC Volt** = **19.53125** * **92** = **1769.875** mv

1769.875 mv is **Lower** than input signal, so the currently bit will be reset, and the next bit will be set, **SAR** = **0b01011110**

→ **DAC Volt** = **19.53125** * **94** = **1835.9375** mv

1835.9375 mv is **Lower** than input signal, so the currently bit will be reset, and the next bit will be set, **SAR** = **0b01011111**

→ **DAC Volt** = **19.53125** * **95** = **1855.4675** mv

AMIT

Analog to Digital Converter

SAR ADC

128	64	32	16	8	4	2	1
0	1	0	1	1	1	1	1

1855.4675 mv is lower than the input signal, so the currently bit will be kept, and the next bit will be set, but it is the end of register, so the Digital value will be **0b01011111** that is the equivalent of **1.86** input volt

SAR ADC always approximates to **lower** in contrast to **RAMP ADC** that it always approximates to **higher**

AMIT

Analog to Digital Converter Peripheral

- Features of ADC:
  - Our ADC is **SAR ADC** circuit (**10**-bit Resolution)
  - **0.5 LSB Integral Non-linearity:**
    - It is an error in ADC that affect on the value of step "LSB equals One Step", mean that the step value can be errored by 0.5 step
  - **±2 LSB Absolute Accuracy:**
    - Overall errors into the ADC equal + or – two steps
  - **65 - 260 µs Conversion Time:**
    - It depends on the ADC frequency itself
  - **Up to 15 kSPS at Maximum Resolution:**
    - It can take 15K sample/sec at the max speed

AMIT

Analog to Digital Converter Peripheral

Features of ADC:

- **8 Multiplexed Single Ended Input Channels:**
  - It has 8 channels, and it can be connected to 8 different sensors and the channels can be changed between of them “multiplexed”
- **7 Differential Input Channels:**
  - It has 7 channels for differential sensor, but it is not multiplexed so it is limited for sensors number
- **2 Differential Input Channels with Optional Gain of 10x and 200x(1):**
  - It has 2 channels for differential sensor with amplifying circuit “10x , 200x”, it is used to noise measuring
- **Optional Left adjustment for ADC Result Readout**

AMIT

Analog to Digital Converter Peripheral

Registers of ADC:

ADMUX (ADC Multiplexer Register)

This register is used to configure some of ADC features, Like:

1. Voltage reference
2. Left/Right Adjustment
3. Channel of ADC Single-ended or Differential inputs

ADC Multiplexer Selection Register – ADMUX

Bit	7	6	5	4	3	2	1	0
	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0
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

AMIT

Analog to Digital Converter Peripheral

ADMUX:

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 2.56 v Voltage Reference with external capacitor at AREF pin

Voltage reference configuration “REFS1 – REFS0”:

1. AREF pin is used to connect to any voltage between 0V to 5V as a voltage reference “Max Voltage” “ **00**”
2. AVCC pin is used to power on the ADC peripheral itself, because it is the only peripheral that is powered externally, so it can be also exploited as a voltage reference for ADC “**01**”
3. ADC can generate an internal voltage equals “**2.56 V**” and it can be used as voltage reference to ADC “**11**”

AMIT

Analog to Digital Converter Peripheral

ADMUX:

ADC Left Adjust Result "ADLAR":

ADLAR is used to configure the adjustment of converted-data result "ADC conversion value" from right to left "0", Like:

- The arrangement here is normal, from **0** to **9** and the reading of it needs two cycles and operations, because our ATMEGA32 is 8-bit **MC**
- Left adjustment allows you to read only the Most Significant Eight bits from high register only within one cycle and operation, but ADC becomes **8**-bit resolution, and it looks like a right shifting by two of all bits

ADLAR = 0

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	

ADLAR = 1

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	

AMIT

Analog to Digital Converter Peripheral

ADMUX:

Analog Channel and Gain Selection Bits"ADMUX0 – ADMUX4":

ADC has a multiplexer has 32 channel, first eight channels are used as a single-ended input, they are used for the sensor that has only one signal wire

- The next eight channels are used for differential-gained input"10x or 200x" which are used to amplify the input signal by 10x or 200x if the differential-input signal was smaller that the ADC step
- The remaining Channels are also used for differential-input signal but without amplifying
- The last two channels on the multiplexer have fixed values, which are used for noise measuring affecting on the ADC peripheral

AMIT

Analog to Digital Converter Peripheral

ADMUX:

Differential input mode:

- This mode is used for the sensors with have two wires for its analog signal.
- The sensor will send its signal on a differential wires “positive and negative”.
- If any noise affects on the wires, the noise value will be same on the two wires, so the differential-input signal will be constant.
- Gained channels may be also used for smaller noise measuring

AMIT

Analog to Digital Converter Peripheral

- Registers of ADC:
  - **ADCSRA**(ADC Control and Status Register)

This register is used to configure some of ADC features, Like:

- Enable ADC
- Start Conversion
- Enable Auto Triggering
- Flag of ADC
- Enable ADC interrupt
- Pre-scaler of ADC(adjust the ADC frequency)

ADC Control and Status Register A – ADCSRA

Bit	7	6	5	4	3	2	1	0
	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0
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

Analog to Digital Converter Peripheral

ADCSRA:

Enable ADC Bit "ADEN":

If it is set, ADC peripheral itself will be ready to run, if it is cleared the ADC peripheral will be disabled

ADC Start Conversion Bit "ADSC":

If it is **set**, ADC peripheral will convert the input signal, in **Single-Running mode** "it means that ADSC must be set every time you need ADC to convert the input signal"

In Free-Running mode "it will be declared later", it must be set at least once

AMIT

Analog to Digital Converter Peripheral

ADCSRA:

ADC Auto Trigger Enable "ADATE":

- When this bit is written to one, Auto Triggering of the ADC is enabled. The ADC will start a conversion on a positive edge of the selected trigger signal. The trigger source is selected by setting the ADC Trigger Select bits, ADTS in SFIOR.
- This bit is used to change the ADC mode from single mode to triggering source mode.
- Triggering mode means that **ADSC** bit will be set with the rising edge of the selected interrupt source "chosen from SFIOR" flag.
- In triggering mode, **ADSC** don't need to set manually every time to start ADC conversion, the conversion starts automatically within the rising edge of interrupt source flag.

AMIT

Analog to Digital Converter Peripheral

ADCSRA:

ADC Auto Trigger Enable "ADATE":

Special Function IO Register – SFIOR:

SFIOR register has most significant three bits which are used to select the triggering conversion 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

Special FunctionIO Register – SFIOR

Bit	7	6	5	4	3	2	1	0
	ADTS2	ADTS1	ADTS0	–	ACME	PUD	PSR2	PSR10
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

AMIT

Analog to Digital Converter Peripheral

ADCSRA:

ADC Auto Trigger Enable "ADATE":

Special FunctionIO Register – SFIOR

Bit	7	6	5	4	3	2	1	0
	ADTS2	ADTS1	ADTS0	–	ACME	PUD	PSR2	PSR10
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

Special Function IO Register – SFIOR:

- 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.

- Free running mode is the only mode that needs at least once to set the bit of starting conversion, because this mode depends on ADC flag itself, and the ADC flag is only set after complete conversion, so it needs one setting to run first conversion and after that every flag rising, **ADSC** bit will be automatically triggered every flag rising.

AMIT

Analog to Digital Converter Peripheral

ADCSRA:

ADC Auto Trigger Enable "ADATE":

Special FunctionIO Register – SFIOR

Bit	7	6	5	4	3	2	1	0
	ADTS2	ADTS1	ADTS0	–	ACME	PUD	PSR2	PSR10
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

Special Function IO Register – SFIOR:

SFIOR register has most significant three bits which are used to select the triggering conversion 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

AMIT

Analog to Digital Converter Peripheral

ADCSRA:

ADC Auto Trigger Enable "ADATE":

Special Function IO Register – SFIOR:

Special FunctionIO Register – SFIOR

Bit	7	6	5	4	3	2	1	0
	ADTS2	ADTS1	ADTS0	–	ACME	PUD	PSR2	PSR10
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

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

AMIT

Analog to Digital Converter Peripheral

ADCSRA:

ADC Interrupt Flag "ADIF":

- This bit is set when an ADC conversion completes, and the Data Registers are updated.
- The ADC Conversion Complete Interrupt is executed if the ADIE bit and the I-bit in SREG are set. ADIF is cleared by hardware when executing the corresponding interrupt handling vector.
- Alternatively, ADIF is cleared by writing a logical one to the flag.
- Beware that if doing a Read-Modify-Write on ADCSRA, a pending interrupt can be disabled.

AMIT

Analog to Digital Converter Peripheral

ADCSRA:

ADC Interrupt Enable "ADIE":

- When this bit is written to one and the I-bit in SREG is set, the ADC Conversion Complete Interrupt is activated

ADC Pre-scaler Select Bits "ADPS 2:0":

- These bits are used to configure the Pre-scaler circuit that is used to shrink the frequency of system freq. to decrease the speed of ADC peripheral

- There is no one-division factor because ADC must be slower than processor to manage to save the converted data into the memory

AMIT

THANK YOU!

AMIT