Analog I/O Operations

CSE0420 – Embedded Systems

By

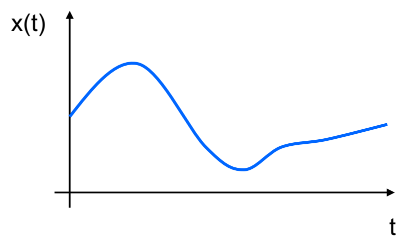
Z. Cihan TAYŞİ

Outline

- Analog I/O
  - **Analog !?**
  - Analog Input
    - What is ADC
    - Types of ADC
    - Real-life examples
  - Analog Output
    - What is DAC

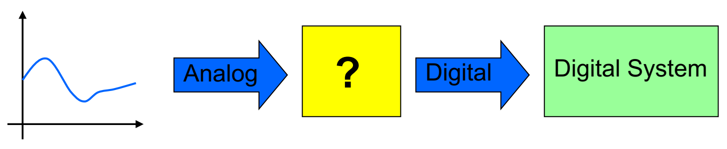
Analog

- Most signals we want to process are analog
  - they are continuous and can take an inifinity of values



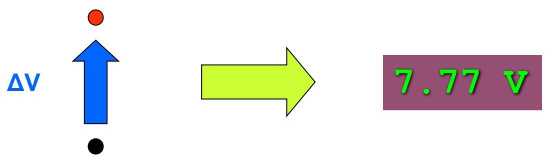
Definition of ADC

- Digital systems require discrete digital data
- ADC converts an analog information into a digital information

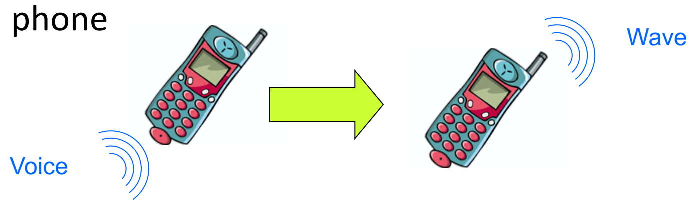


Examples of Use

- Voltmeter



- Cell phone

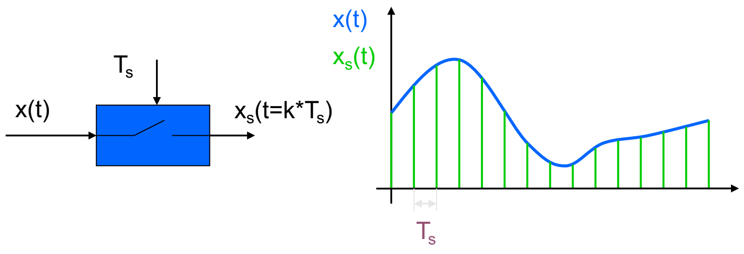


Conversion Process

- 3 steps of conversion
  - Sampling
  - Quantification
  - Coding
- These operations are all performed in a same element:
  - the Analog to Digitial Converter

Conversion Process : Sampling

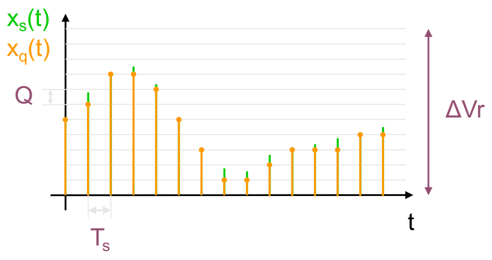
- Digital system works with discrete states
- The signal is only defined at determined times
- The sampling times are proportional to the sampling period (**T_s**)



Conversion Process : Quatification

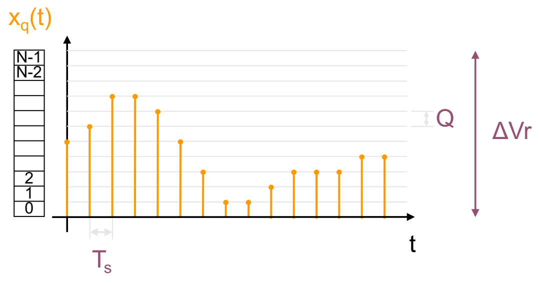
- The signal can only take determined values belonging to a range of conversion (**ΔV_r**)
- Based on number of bit combinations that the converter can output
- Number of possible states:

  $N=2^n$ where n is number of bits
- Resolution: **$Q= \Delta V_r/N$**

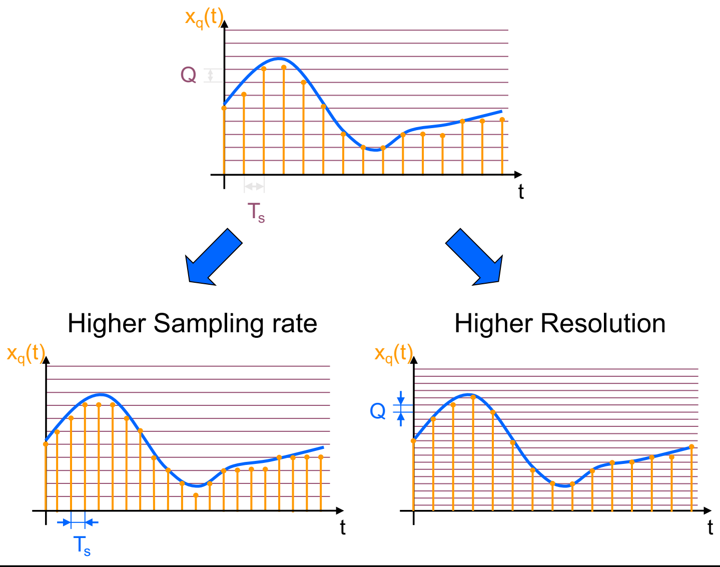


Conversion Process : Coding

- Assigning a unique digital word to each sample
- Matching the digital word to the input signal



Accuracy

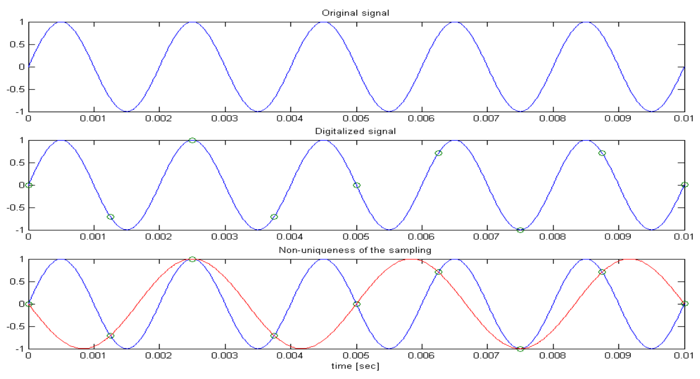


The accuracy of an ADC can be improved by increasing:

- The sampling rate (T_s)
- The resolution (Q)

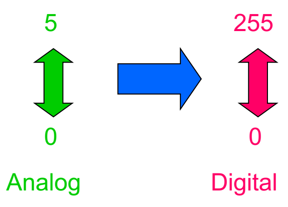
Sampling Rate

- **Nyquist-Shannon theorem**: Minimum sampling rate should be at least twice the highest data frequency of the analog signal
  $f_s > 2_* f_{max}$

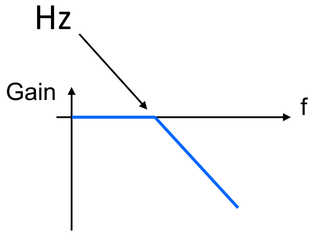


Example

- 8 bits converter: n=8
- Range of conversion: ΔVr=5V
- Sampling time: Ts=1ms



- Number of possible states: $N=2^8=256$
- Resolution: Q=ΔVr/N=19.5 mV
- Analog Filter: ffilter ≈ fs/5 = 200 Hz

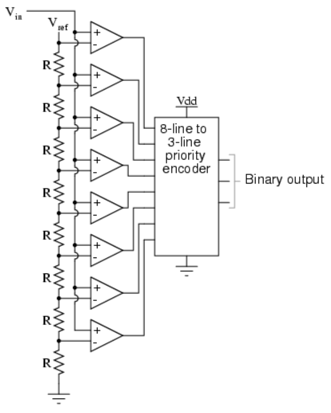


Types of ADC

- Flash ADC
- Sigma-delta ADC
- Dual slope converter
- Successive approximation converter

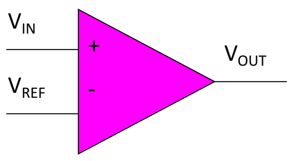
Flash ADC

- "parallel A/D"
- Uses a series of comparators
- Each comparator compares V_{in} to a different reference voltage, starting w/ V_{ref} = 1/2 lsb



Flash ADC

Comparator is one use of an Op-Amp



If	Output
$V_{IN} > V_{REF}$	High
$V_{IN} < V_{REF}$	Low

Flash ADC

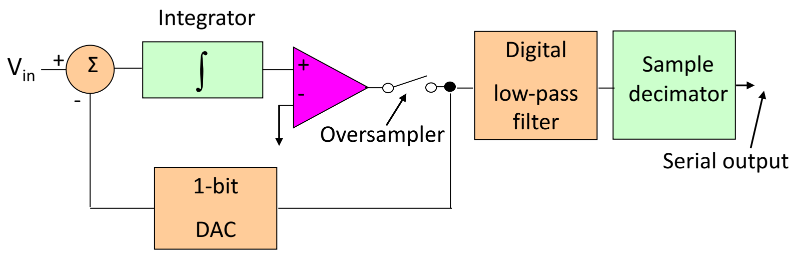
Advantages

- Very fast

Disadvantages

- Needs many parts (255 comparators for 8-bit ADC)
- Lower resolution
- Expensive
- Large power consumption

Sigma-Delta ADC



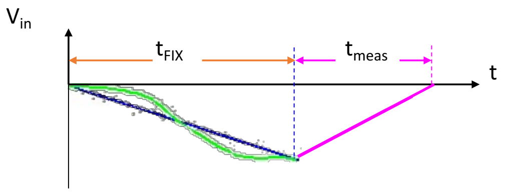
- Oversampled input signal goes in the integrator
- Output of integration is compared to GND
- Iterates to produce a serial bitstream
- Output is serial bit stream with # of 1's proportional to V_{in}

Sigma-Delta ADC

Advantages	Disadvantages
▶ High resolution ▶ No precision external components needed	▶ Slow due to oversampling

Dual Slope Converter

- The sampled signal charges a capacitor for a fixed amount of time
- By integrating over time, noise integrates out of the conversion.
- Then the ADC discharges the capacitor at a fixed rate while a counter counts the ADC's output bits. A longer discharge time results in a higher count.

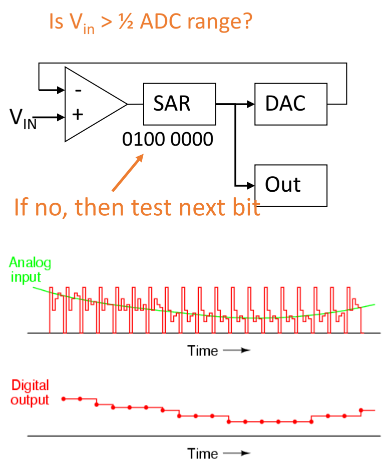


Dual Slope Converter

Advantages	Disadvantages
Input signal is averaged	Slow
Greater noise immunity than other ADC types	High precision external components required to achieve accuracy
High accuracy	

Successive Approximation

- Sets MSB
- Converts MSB to analog using DAC
- Compares guess to input
- Set bit
- Test next bit



Successive Approximation

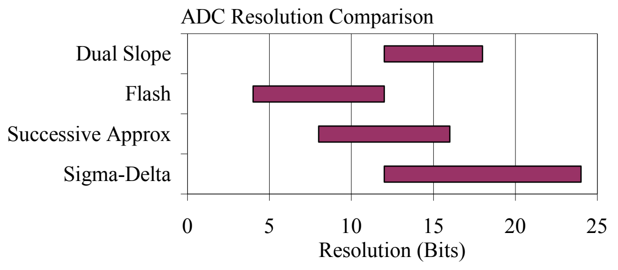
Advantages

- Capable of high speed
- Medium accuracy compared to other ADC types
- Good tradeoff between speed and cost

Disadvantages

- Higher resolution successive approximation ADCs will be slower
- Speed limited ~5Msps

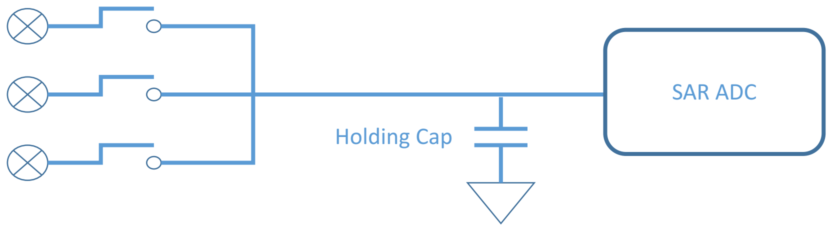
ADC Type Comparison



Type	Speed (relative)	Cost (relative)
Dual Slope	Slow	Med
Flash	Very Fast	High
Successive Appox	Medium – Fast	Low
Sigma-Delta	Slow	Low

Simplified ADC Module Diagram

- The analog input pins are connected to the inputs of an analog multiplexer which connects the selected channel to the holding capacitor.
- There is only one analog-to-digital converter on the microcontroller, and only one channel can be selected, and therefore converted at a time.
- When a conversion is initiated, the analog multiplexer disconnects all inputs from the holding capacitor, and the successive approximation converter performs the conversion on the voltage stored on the holding capacitor.



A/D Conversion Steps

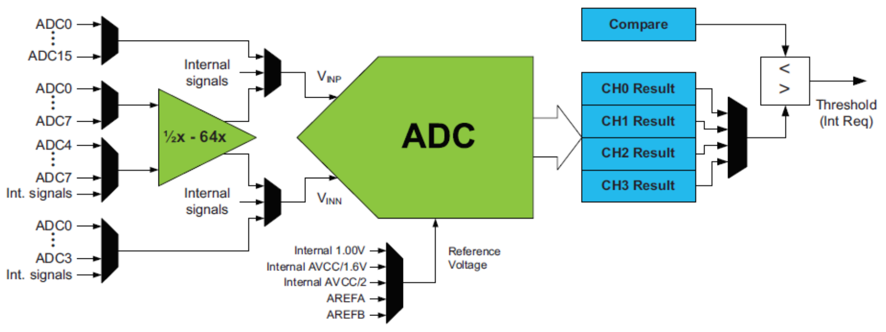
- Configure I/O Pins
- Select the Channel to Convert
- Configure and Enable the A/D
  - clock source, reference voltage
- Wait the Acquisition Time
- Initiate the Conversion
- Wait for the Conversion to Complete
- Read the Result
  - left alignment / right alignment

A Real-life Example

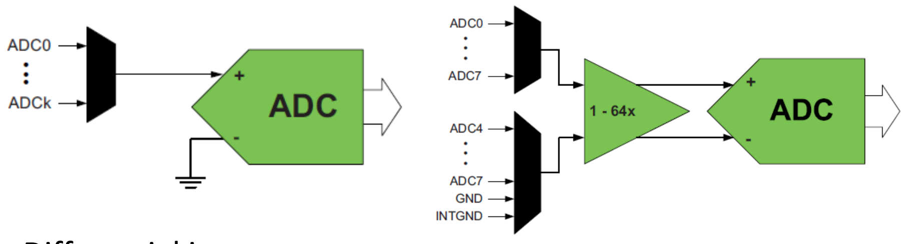
- Atmel ATXMEGA
  - 12-bit Analog to Digital Converter
- Features
  - 12-bit resolution
  - Up to two million samples per second
  - Differential and single-ended input
  - Built-in differential gain stage
  - Single, continuous and scan conversion options
  - Four internal inputs
  - Four conversion channels with individual input control and result registers
  - Internal and external reference options

Atmel ATXMEGA

Figure 29-1. ADC overview.



Input Types



- Differential input
- Differential input with gain
- Single ended input
- Internal input

Voltage Reference

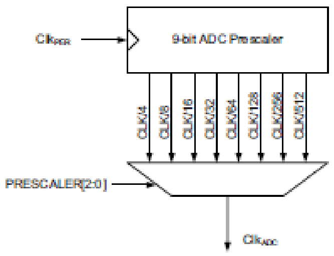
- Four types of voltage reference available
  - Accurate internal 1.00V voltage generated from the bandgap
  - Internal VCC/1.6V voltage
  - External voltage applied to AREF pin on PORTA
  - External voltage applied to AREF pin on PORTB

Starting a Conversion

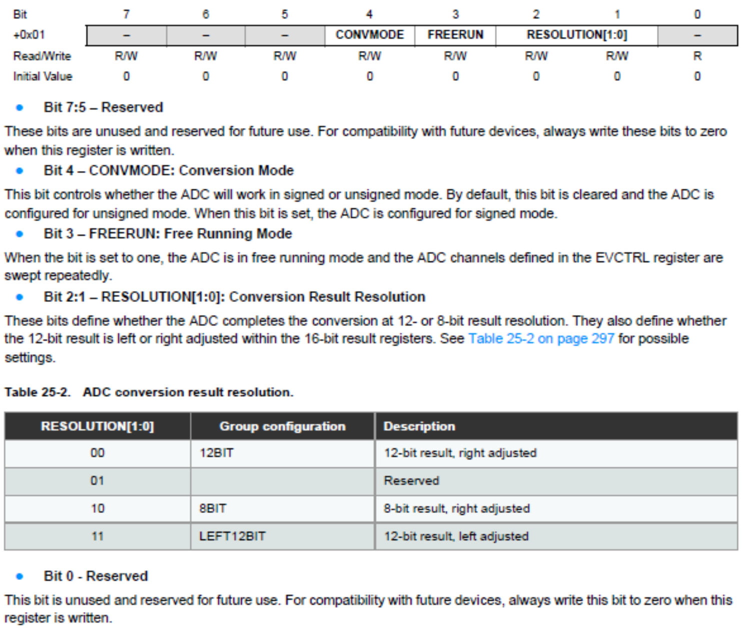
- Before a conversion is started,
  - the input source must be selected for one or more ADC channels.
- An ADC conversion for a channel can be started either by
  - the application software writing to the start conversion bit for the channel or
  - from any events in the event system.
- It is possible to write the start conversion bit for several channels at the same time, or use

ADC Clock and Conversion Timing

- The ADC is clocked from the peripheral clock.
  - The ADC can prescale the peripheral clock to provide an ADC Clock (clkADC) that matches the application requirements and is within the operating range of the ADC.
- The maximum ADC sample rate is given by the he ADC clock frequency (fADC).
  - The ADC can sample a new measurement on every ADC clock cycle.



$$\text{Propagation Delay} = \frac{1 + \frac{RESOLUTION}{2} + GAIN}{f_{ADC}}$$



Bit	7	6	5	4	3	2	1	0
+0x01	–	–	–	CONVMODE	FREERUN	RESOLUTION[1:0]		–
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R
Initial Value	0	0	0	0	0	0	0	0

- **Bit 7:5 – Reserved**

These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.

- **Bit 4 – CONVMODE: Conversion Mode**

This bit controls whether the ADC will work in signed or unsigned mode. By default, this bit is cleared and the ADC is configured for unsigned mode. When this bit is set, the ADC is configured for signed mode.

- **Bit 3 – FREERUN: Free Running Mode**

When the bit is set to one, the ADC is in free running mode and the ADC channels defined in the EVCTRL register are swept repeatedly.

- **Bit 2:1 – RESOLUTION[1:0]: Conversion Result Resolution**

These bits define whether the ADC completes the conversion at 12- or 8-bit result resolution. They also define whether the 12-bit result is left or right adjusted within the 16-bit result registers. See Table 25-2 on page 297 for possible settings.

Table 25-2. ADC conversion result resolution.

RESOLUTION[1:0]	Group configuration	Description
00	12BIT	12-bit result, right adjusted
01		Reserved
10	8BIT	8-bit result, right adjusted
11	LEFT12BIT	12-bit result, left adjusted

- **Bit 0 - Reserved**

This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written.

ADC Configuration

```
/* Configure the ADC module:
 * - signed, 12-bit results
 * - bandgap (1 V) voltage reference
 * - 200 kHz lock rate
 * - channel0 triggered by event
 * - use the ADC current limiter in high setting
mode
 * - enable the internal bandgap reference
 * - callback function
 */
```

```
adc_set_conversion_parameters(&adc_conf,
ADC_SIGN_ON, ADC_RES_12,
ADC_REF_BANDGAP);
adc_set_clock_rate(&adc_conf, 200000UL);
adc_set_conversion_trigger(&adc_conf,
ADC_TRIG_EVENT_SWEEP, 1, 0);
adc_set_current_limit(&adc_conf,
ADC_CURRENT_LIMIT_HIGH);
adc_set_gain_impedance_mode(&adc_conf,
ADC_GAIN_HIGHIMPEDANCE);
adc_enable_internal_input(&adc_conf,
ADC_INT_BANDGAP);
adc_set_callback(&ADCA, &adc_handler);
```

References

- http://www.atmel.com/images/atmel-8032-using-the-atmel-avr-xmega-adc_application-note_avr1300.pdf
- https://github.com/eewiki/asf/blob/master/xmega/drivers/adc/example3/adc_example3.c