CS301
Embedded System and Microcomputer Principle

Lecture 12: ADC

2024 Fall

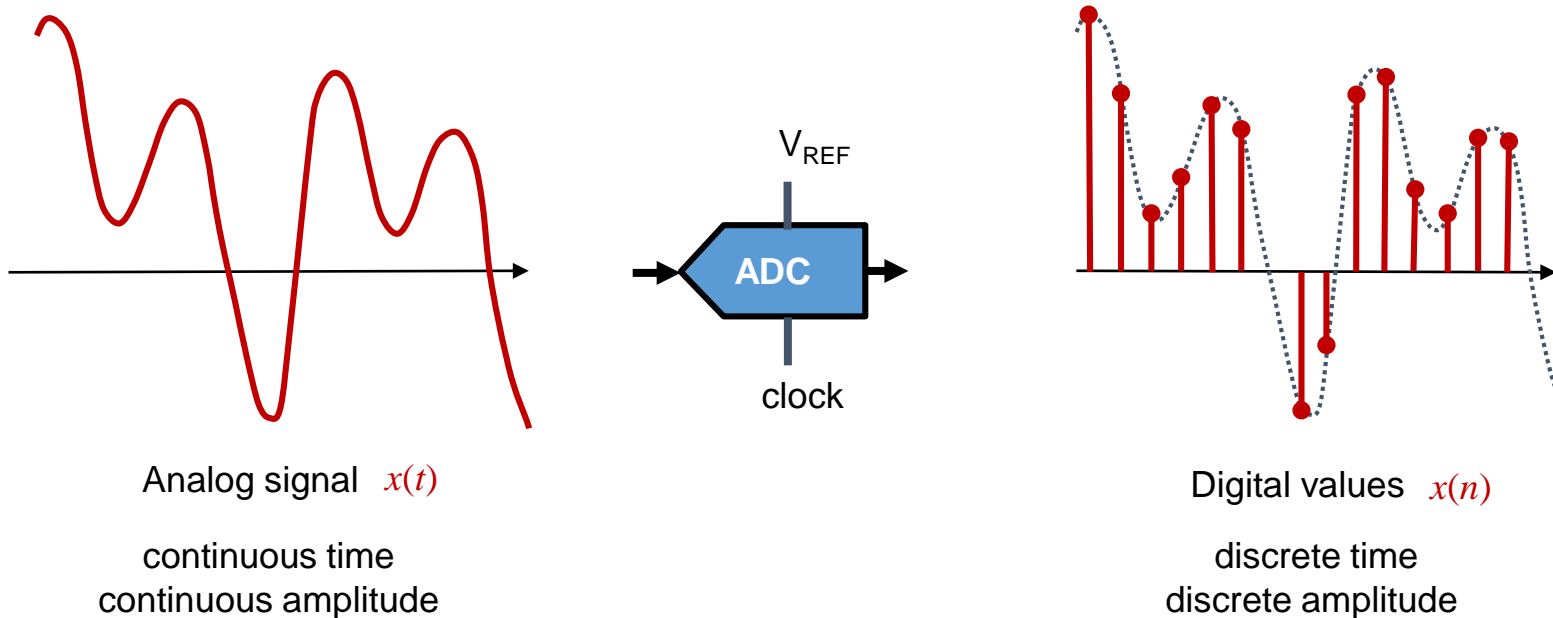# Outline
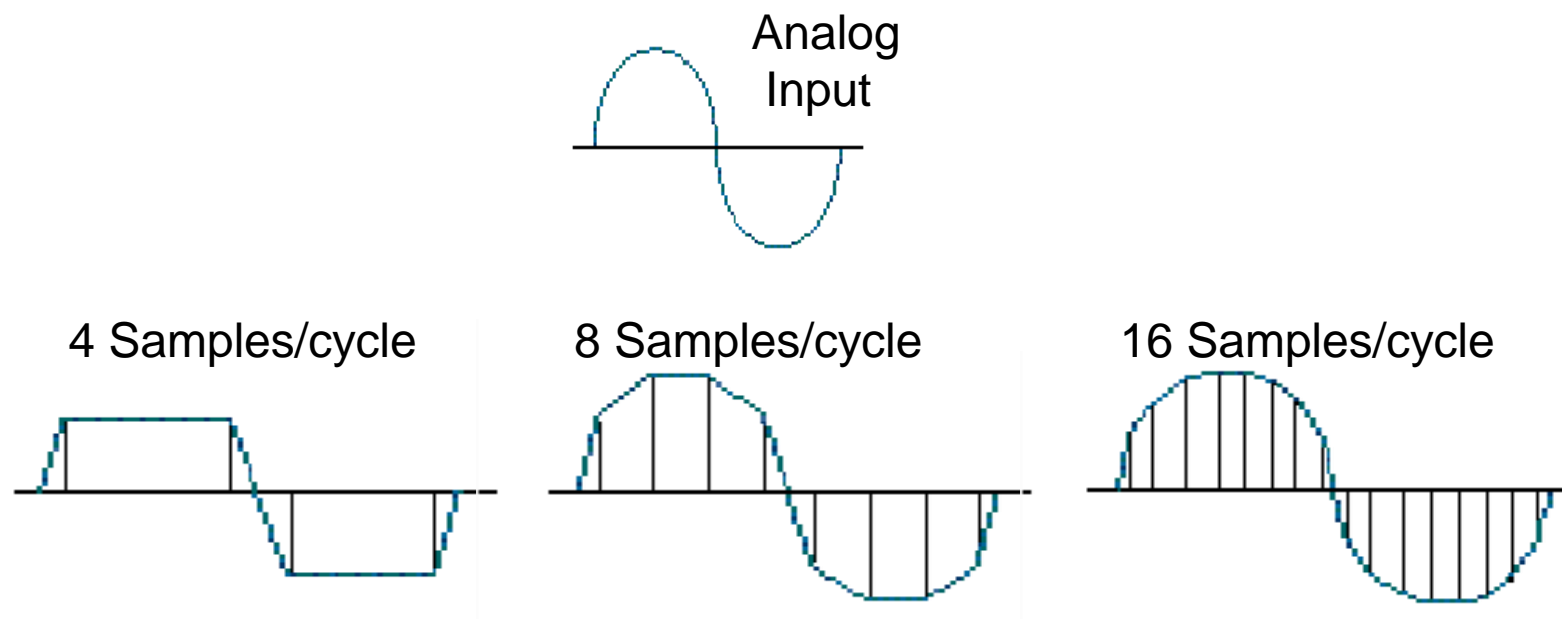
- **ADC introduction**
- STM32 ADC

Analog-to-Digital Converter (ADC)

- ADC: a circuit that takes in an analog voltage and produces a digital representation of its value
  - To know nature phenomena, which is analog, and make it feasible for computer to handle, we need to convert it into digital signals
  - To transform the analog, continuous signals into digital ones, the ADC samples the input at fixed interval and do the conversion


Analog signal $x(t)$

continuous time
continuous amplitude
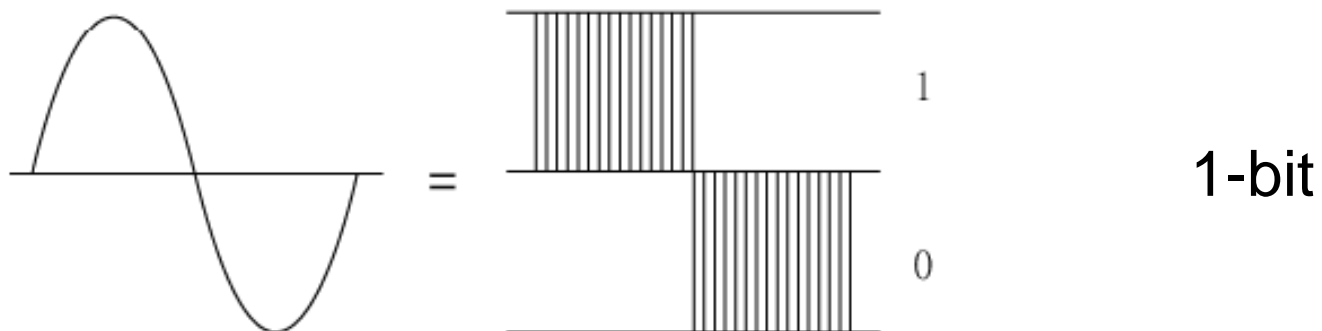
V_{REF}

ADC

clock

Digital values $x(n)$

discrete time
discrete amplitude

Sampling

- Sampling rate:
  - How often analog signal is measured (samples per second, Hz), e.g. 100 Hz

Analog Input

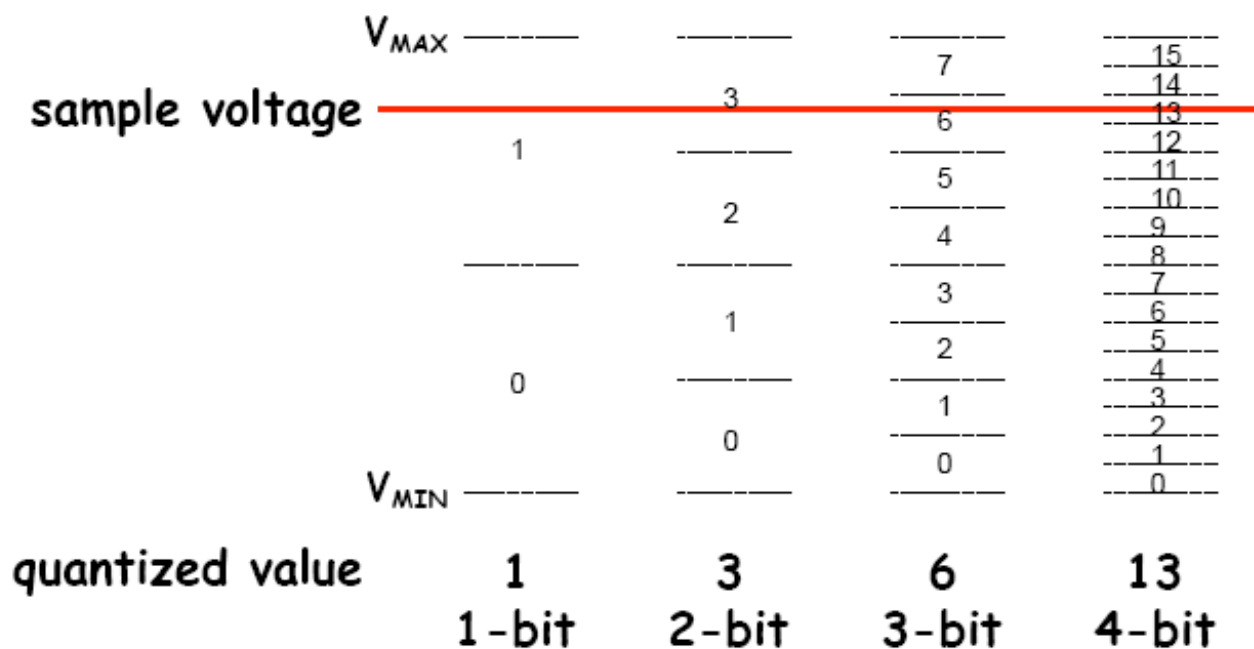4 Samples/cycle   8 Samples/cycle   16 Samples/cycle

Sampling

- Sampling resolution:
  - Number of bits to represent each sample (bit depth), e.g. 12 bit in STM32

1-bit

3-bit
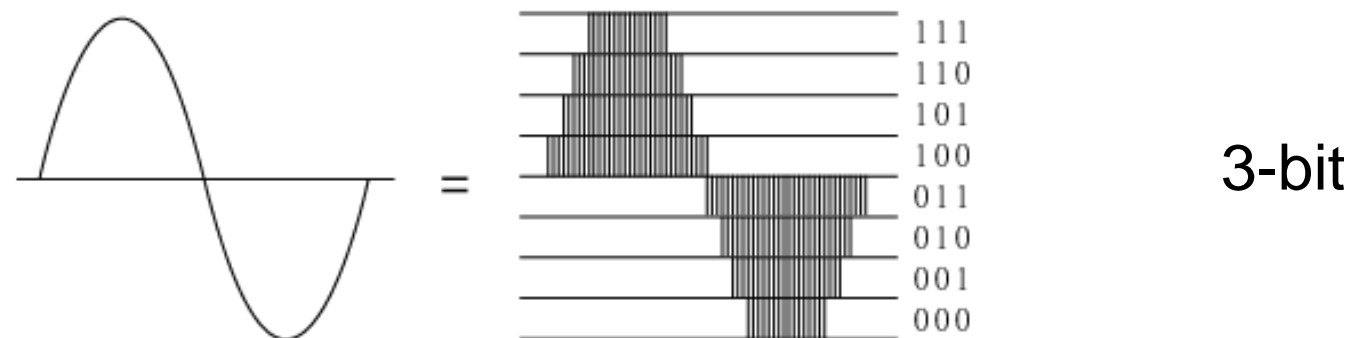
Resolution

- If we use N bits to encode the magnitude of one of the discrete-time samples, we can capture 2^N possible values
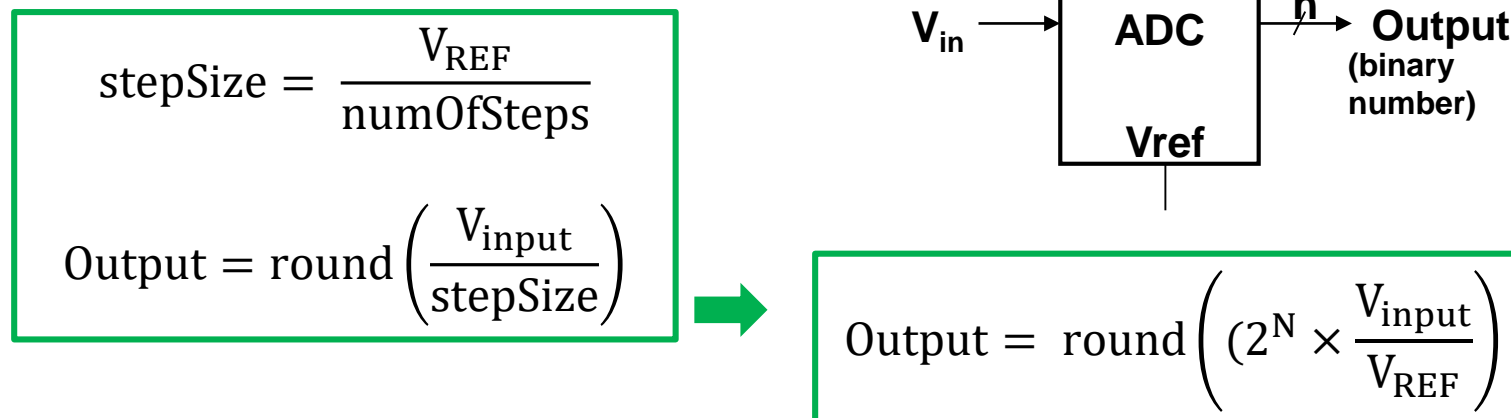
Resolution

- The ADC has n-bit resolution, where n can be 8, 10, 12, 16, or even 24 bits. Higher-resolution ADCs provide a smaller step size, where step size is the smallest change that can be discerned by an ADC.
- We can control the step size with the help of what is called Vref

n-bit	Number of steps	Step size
8	256	5V /256 = 19.53 mV
10	1024	5V /1024 = 4.88 mV
12	4096	5V /4096 = 1.2 mV
16	65,536	5V /65,536 = 0.076 mV
E.G. : V_{ref}= 5V		

Resolution

- Calculate digital data output

$$stepSize = \frac{V_{REF}}{numOfSteps}$$

$$Output = round\left(\frac{V_{input}}{stepSize}\right)$$

$V_{in} \longrightarrow$ **ADC** $\xrightarrow{\quad n \quad}$ **Output** (binary number)

Vref

$$Output = round\left(\left(2^N \times \frac{V_{input}}{V_{REF}}\right)\right)$$

- Example
  - For a given 8-bit ADC, we have Vref = 2.56 V. Calculate the Dout if the analog input is: (a)1.7 V, and (b) 2.1 V.
  - step size = 2.56/256 = 10 mV
  - a)D_{Out} = 1.7V/10 mV = 170_{dec} = 10101011_{bin} ,
  - b)D_{Out} = 2.1V/10 mV = 210_{dec} = 11010010_{bin}

ADC Overall Schematic

- If the input signal changes during the conversion, the final value Dout may be erroneous.
- We need a sample-and-hold circuit to sample the input voltage, and hold it to a constant value while the conversion is in progress.
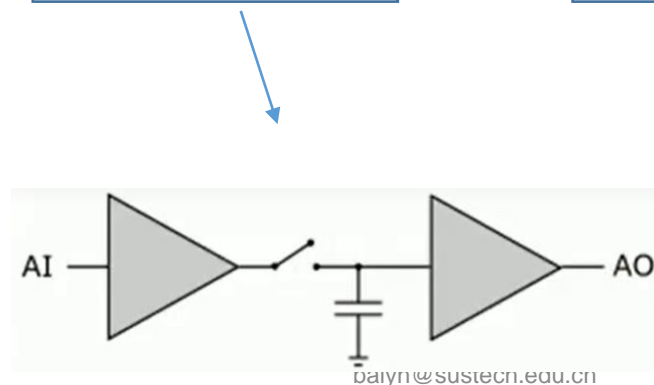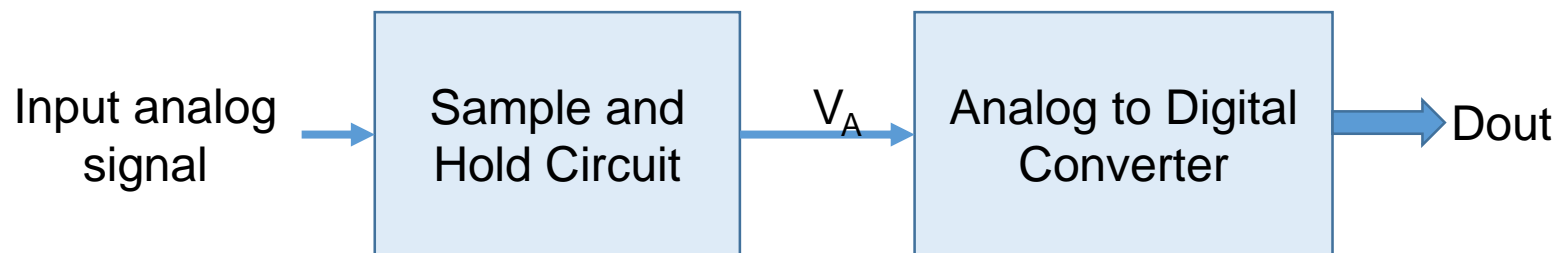
ADC Architecture

- Popular ADC Architectures
  - **Successive-approximation ADC(SAR):**
    - Use binary search to determine the closest digital representation of the input signal, e.g., ADC12 to give 12 bits of output
  - Flash ADC:
    - Flash ADC directly converts an analog input signal into a digital output by using a set of comparators to rapidly compare the input against multiple reference voltages, yielding a high-speed conversion with low resolution, such as in the case of a 4-bit flash ADC.
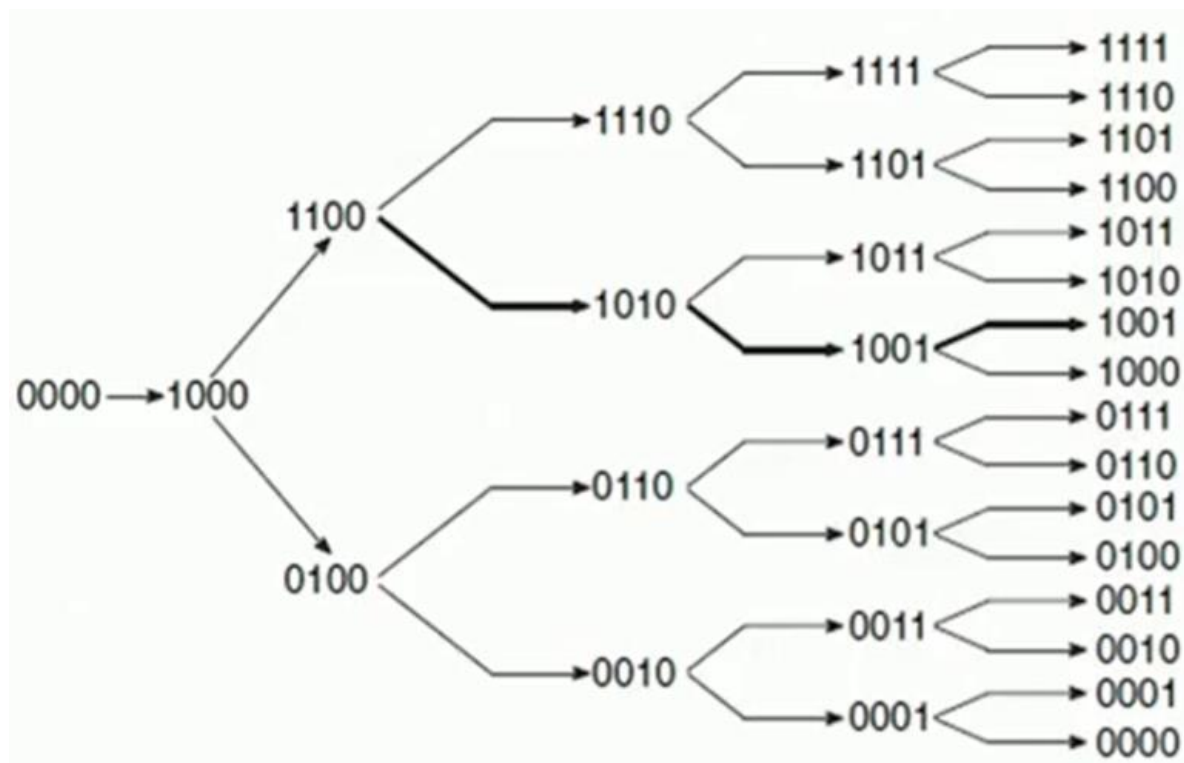 - etc

ADC Architecture

- Comparison of the speed/resolution characteristics of various ADCs
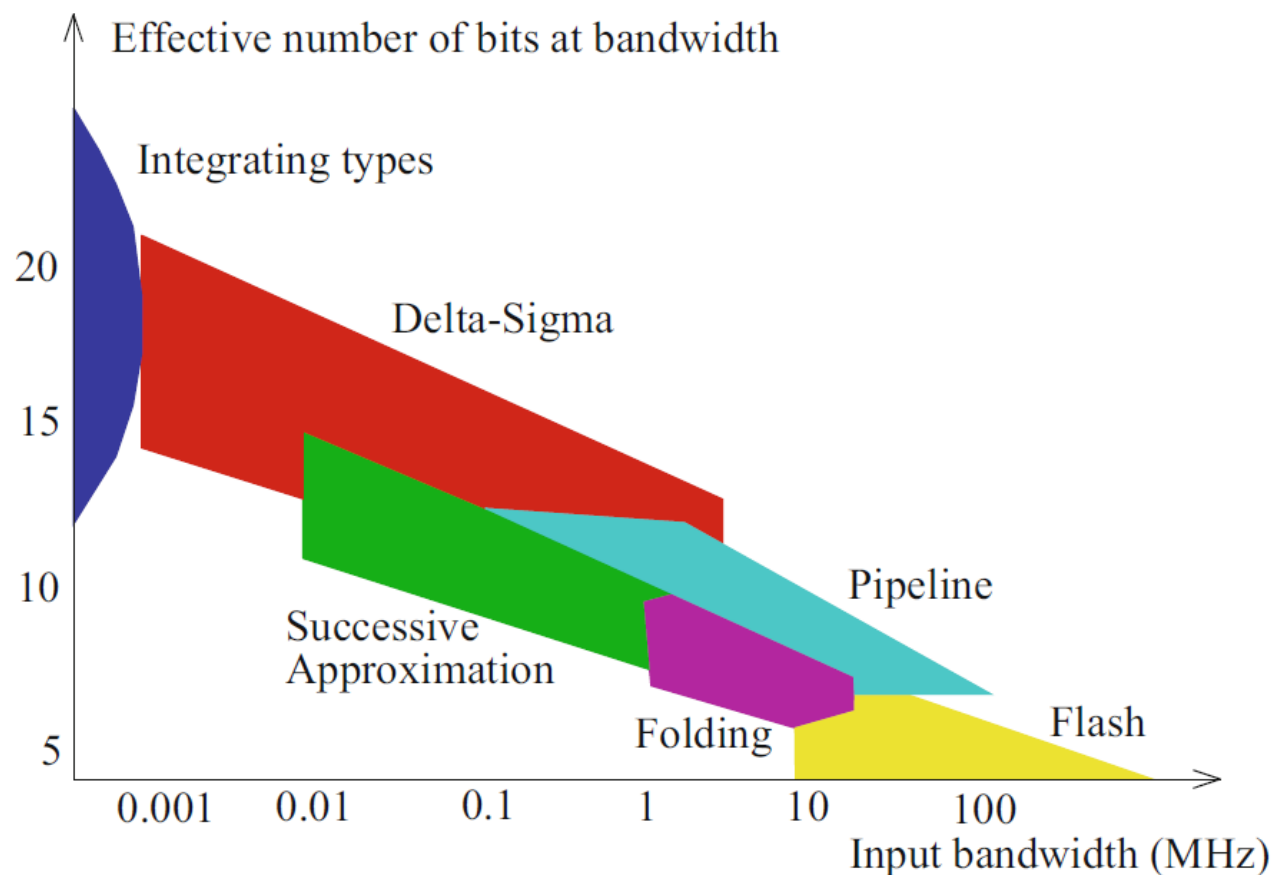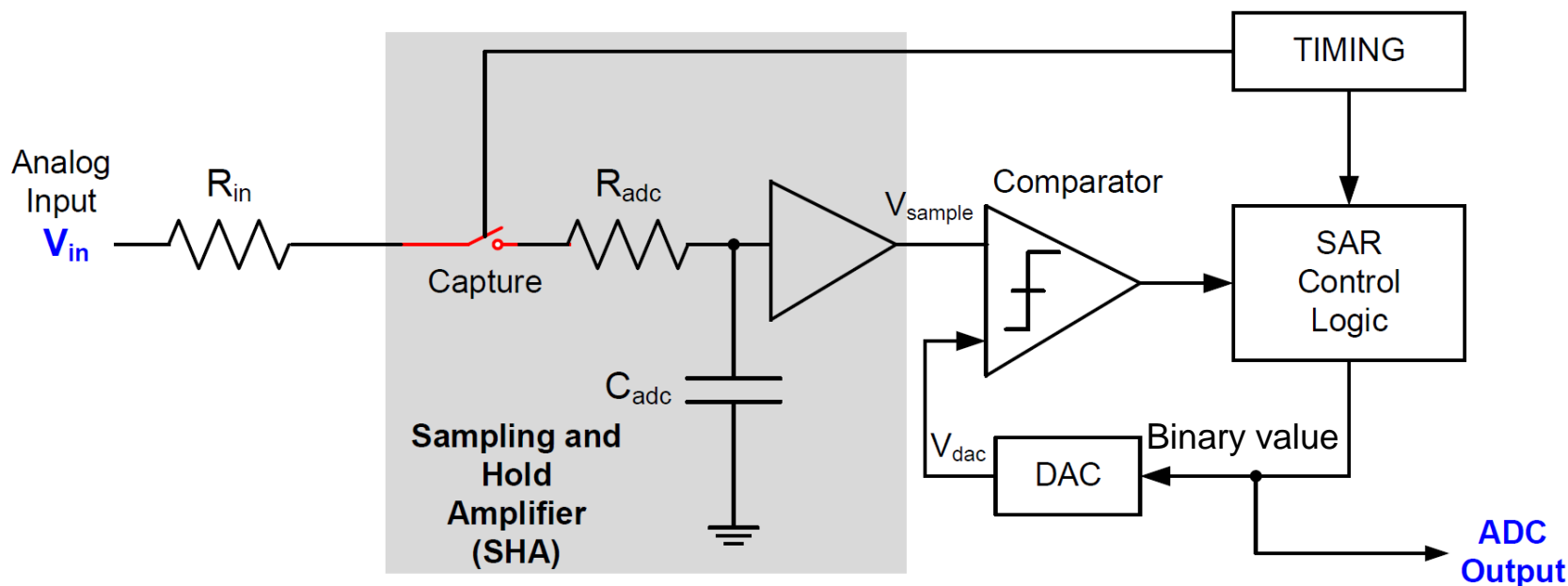
Successive-approximation (SAR)

- The principle of operation is analogous to binary search
- E.g. in 4 bits, it starts with 1000 (i.e. 8)
    - If the input is smaller, the next value is set as 0100 (i.e. 4)
   - If the input is larger, the next value is set at 1100 (i.e. 12).

Successive-approximation (SAR)

- SAR involves iteratively approximating the input analog voltage by comparing it with a series of binary-weighted voltage levels until a digital output is obtained.

Successive-approximation (SAR)

- Conversion time: number of iterations, usually N cycles for N-bit ADC. In STM32 12-bit ADC, T conversion = 12.5 cycles
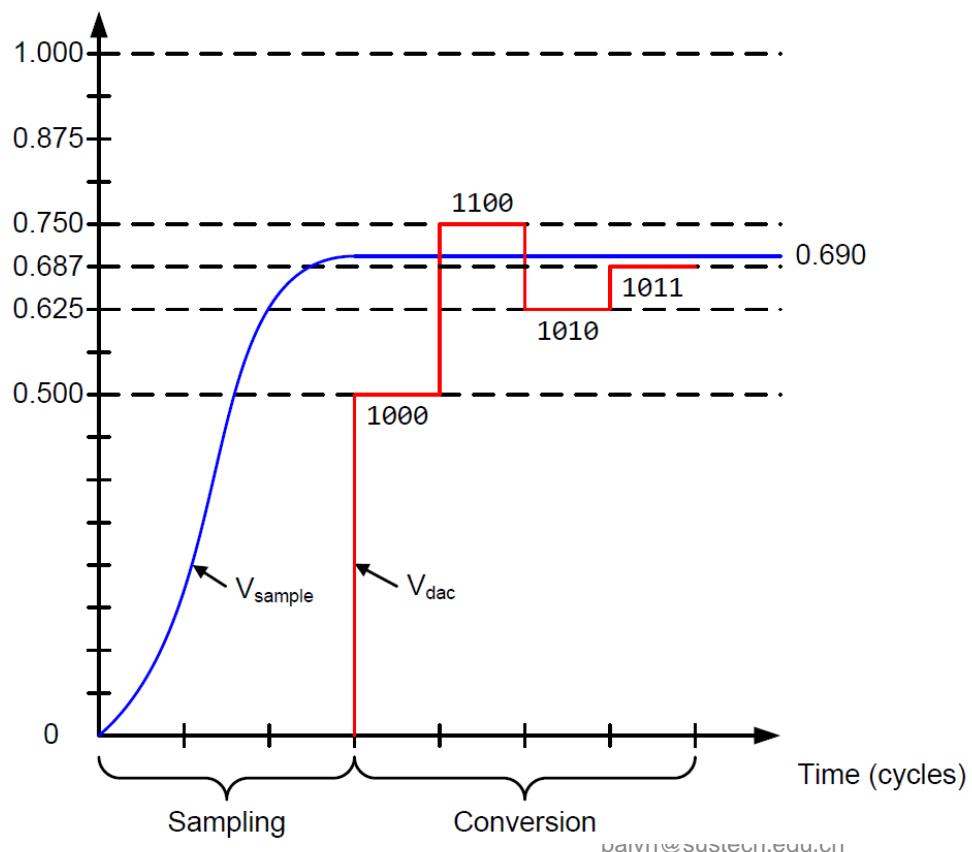

Range: 0 ~ 1111
1st ite: 1000? Too low
2nd ite: 1100? Too high
3rd ite: 1010? Too low
4th ite: 1011 → done!
For 4 bits ADC, at
most 4 iterations

Flash ADC   Parallel ADC

- Aka. Parallel ADC
- A series of comparators, each one comparing the input signal to a unique reference voltage. The comparator outputs connect to the inputs of a priority encoder circuit, which then produces a binary output.

	Resolution	Sampling rate	Cost
SAR ADC	High	Low	Low
Flash ADC	Low	High	High

baiyh@sustech.edu.cn
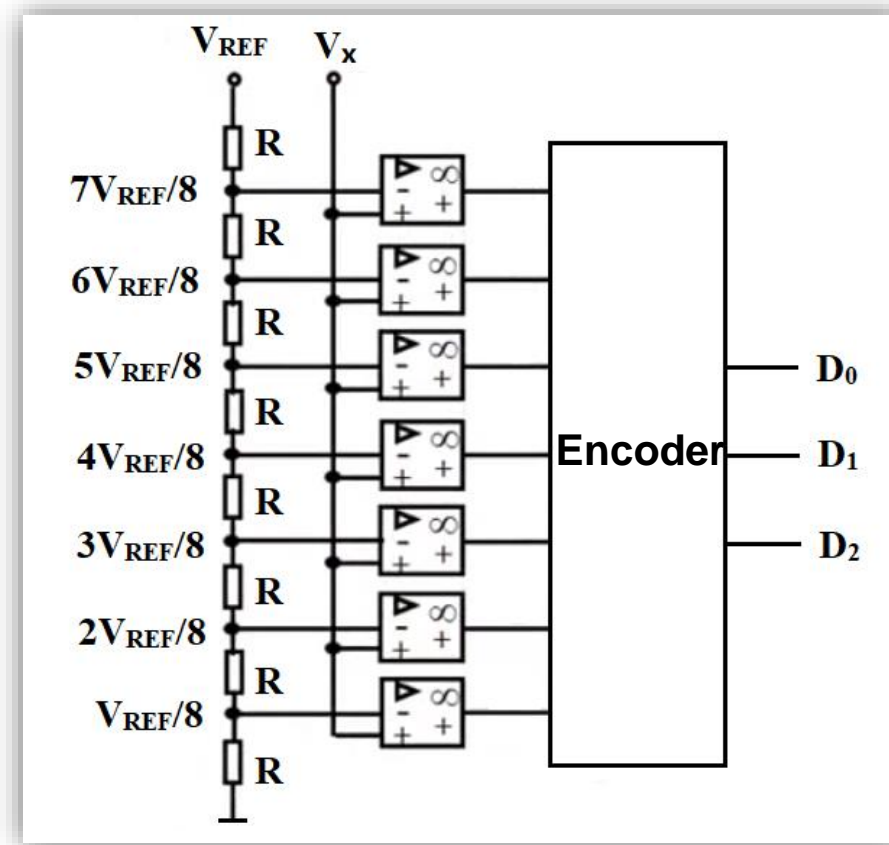
ADC working flow

- Provide continuous sampling of multiple analog inputs and store sampled data
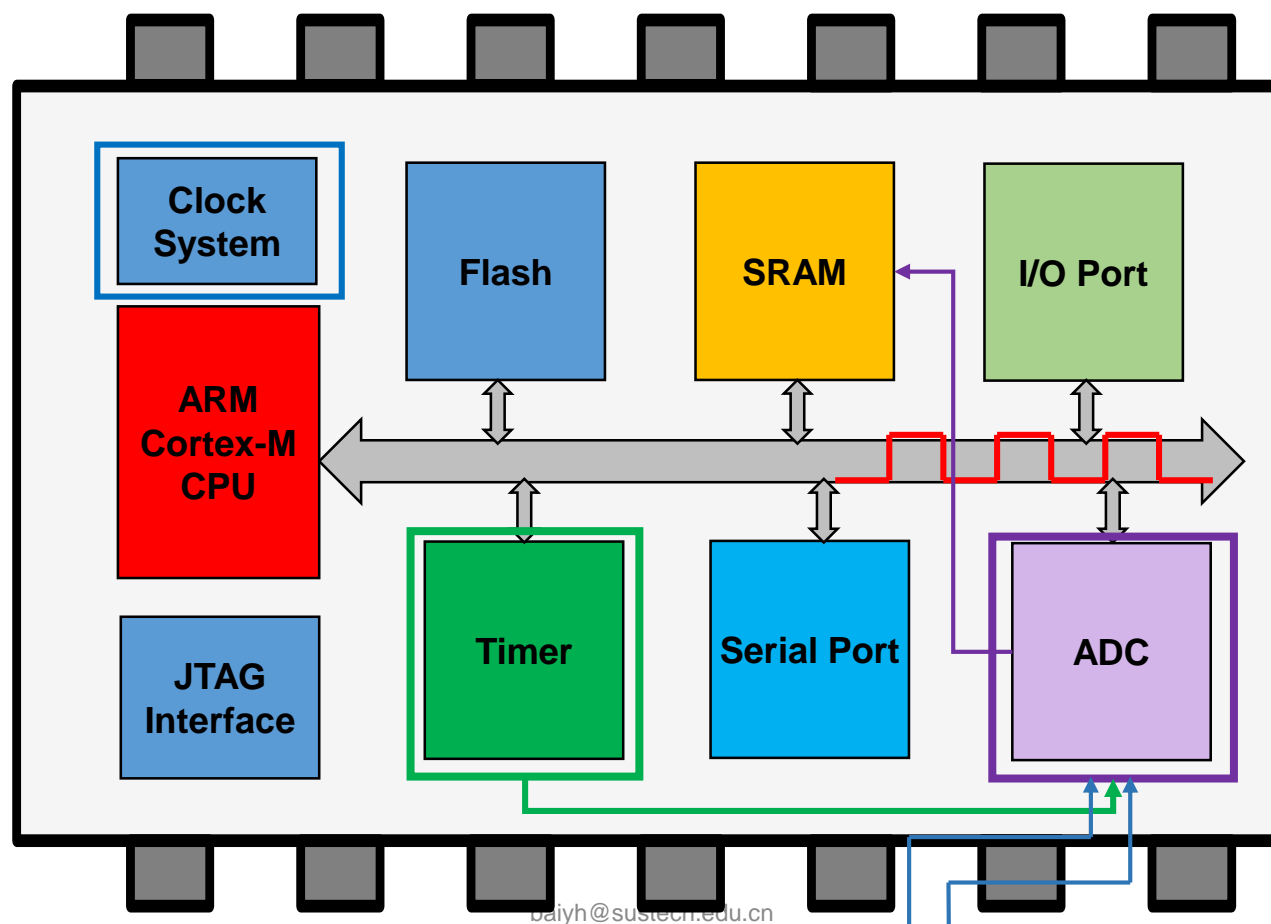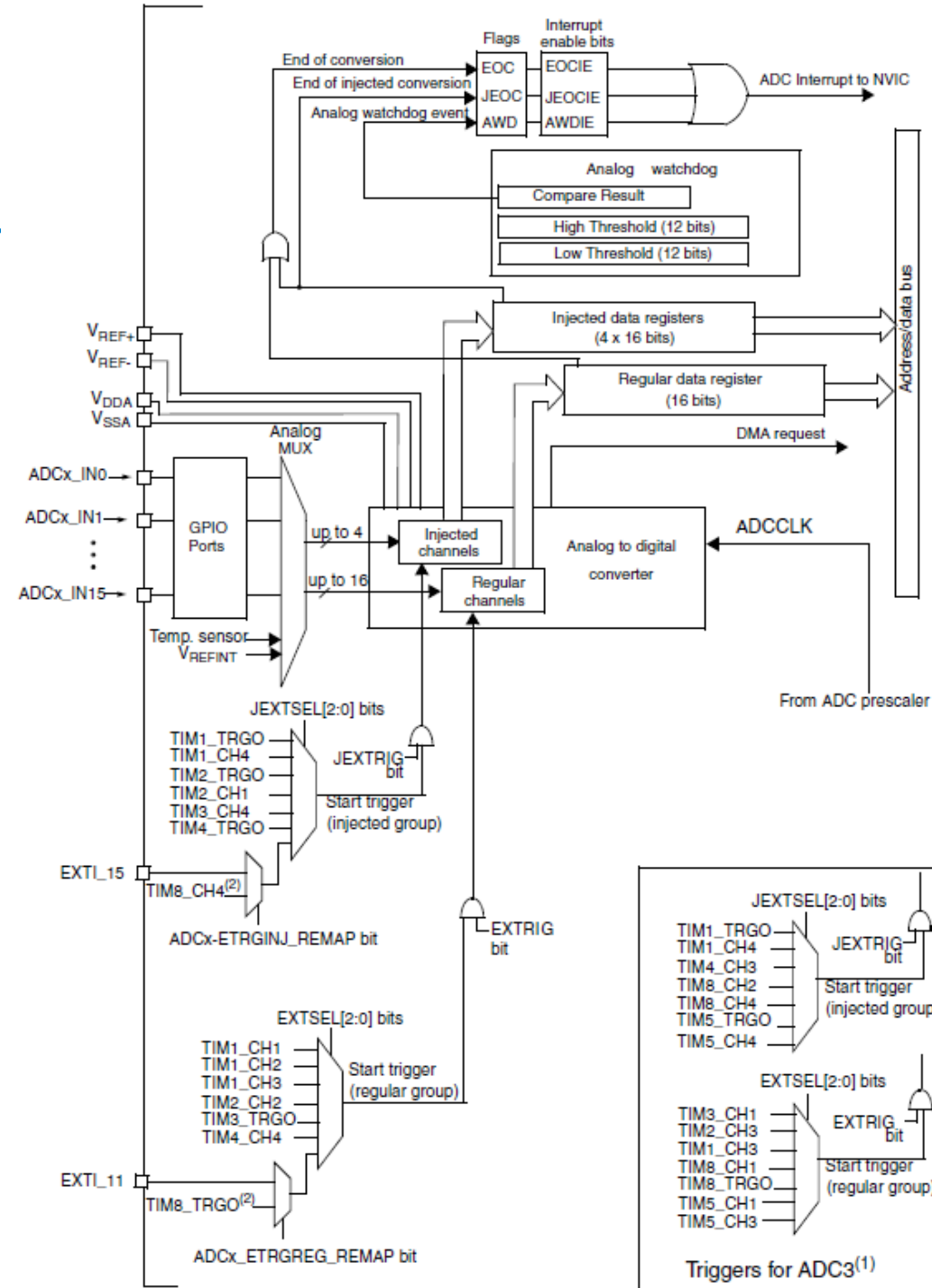
Outline

- ADC introduction
- **STM32 ADC**

STM32 ADC

- ADC clock, Max 14MHz
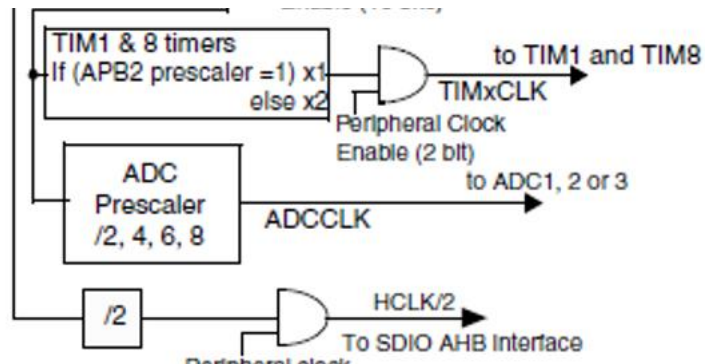
ADC Conversion Modes

• regular group: composed of up to 16 channels

• injected group: composed of up to 4 channels

• Injected group has higher priority
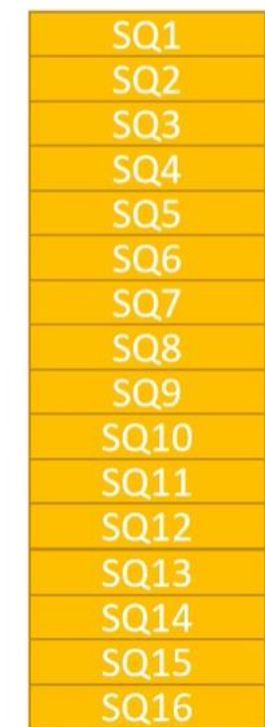
regular channel
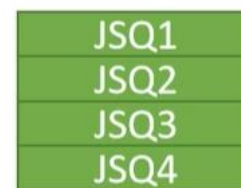
injected channel

Single conversion mode

Start   Finished   Start   Finished

5
SQ2
SQ3
SQ4
SQ5
SQ6
SQ7
SQ8
SQ9
SQ10
SQ11
SQ12
SQ13
SQ14
SQ15
SQ16

regular channel

5
JSQ2
JSQ3
JSQ4

injected channel
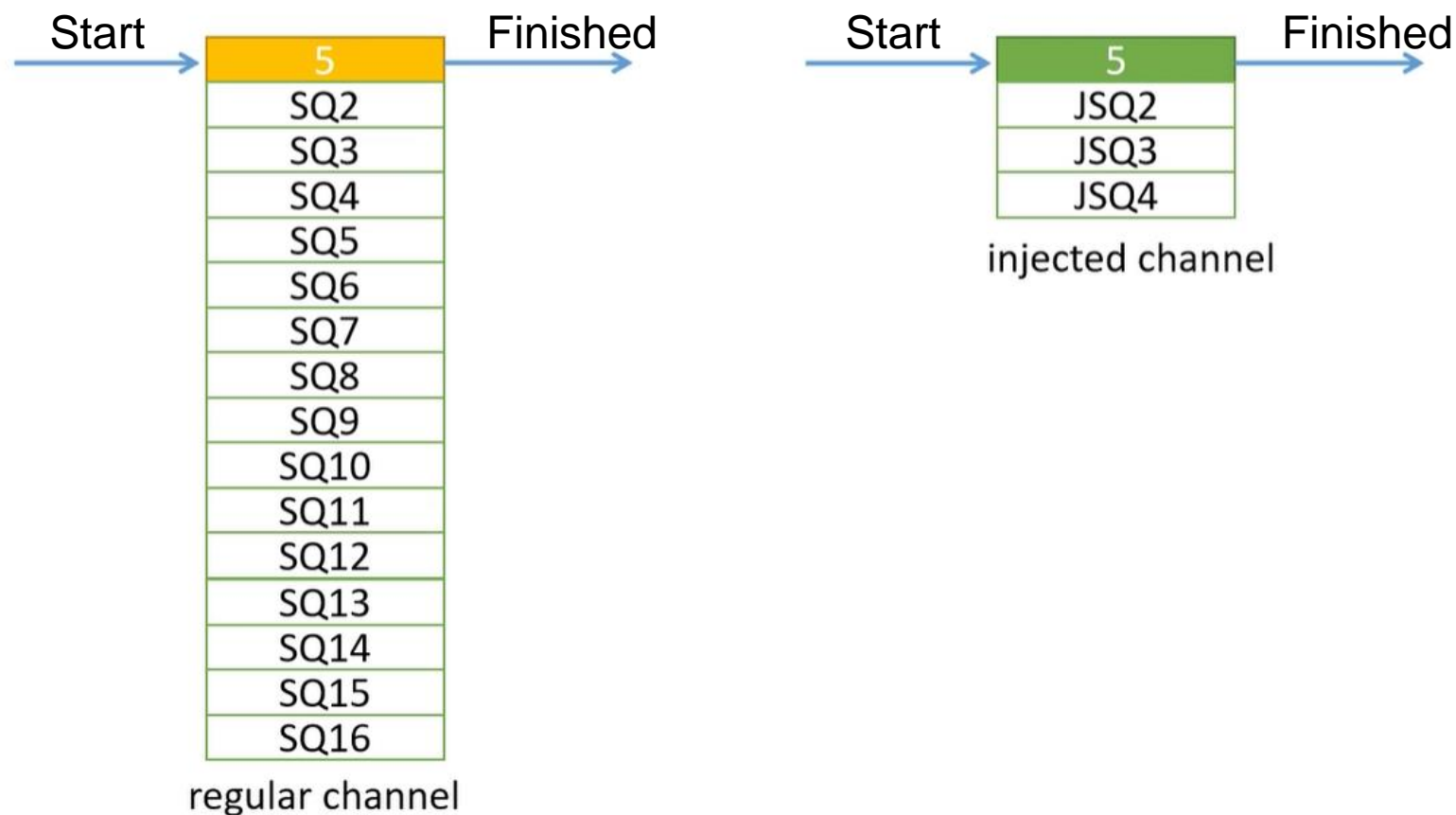
Continuous conversion mode

Start   Continue

5
SQ2
SQ3
SQ4
SQ5
SQ6
SQ7
SQ8
SQ9
SQ10
SQ11
SQ12
SQ13
SQ14
SQ15
SQ16

regular channel

Start   Continue

5
JSQ2
JSQ3
JSQ4

injected channel
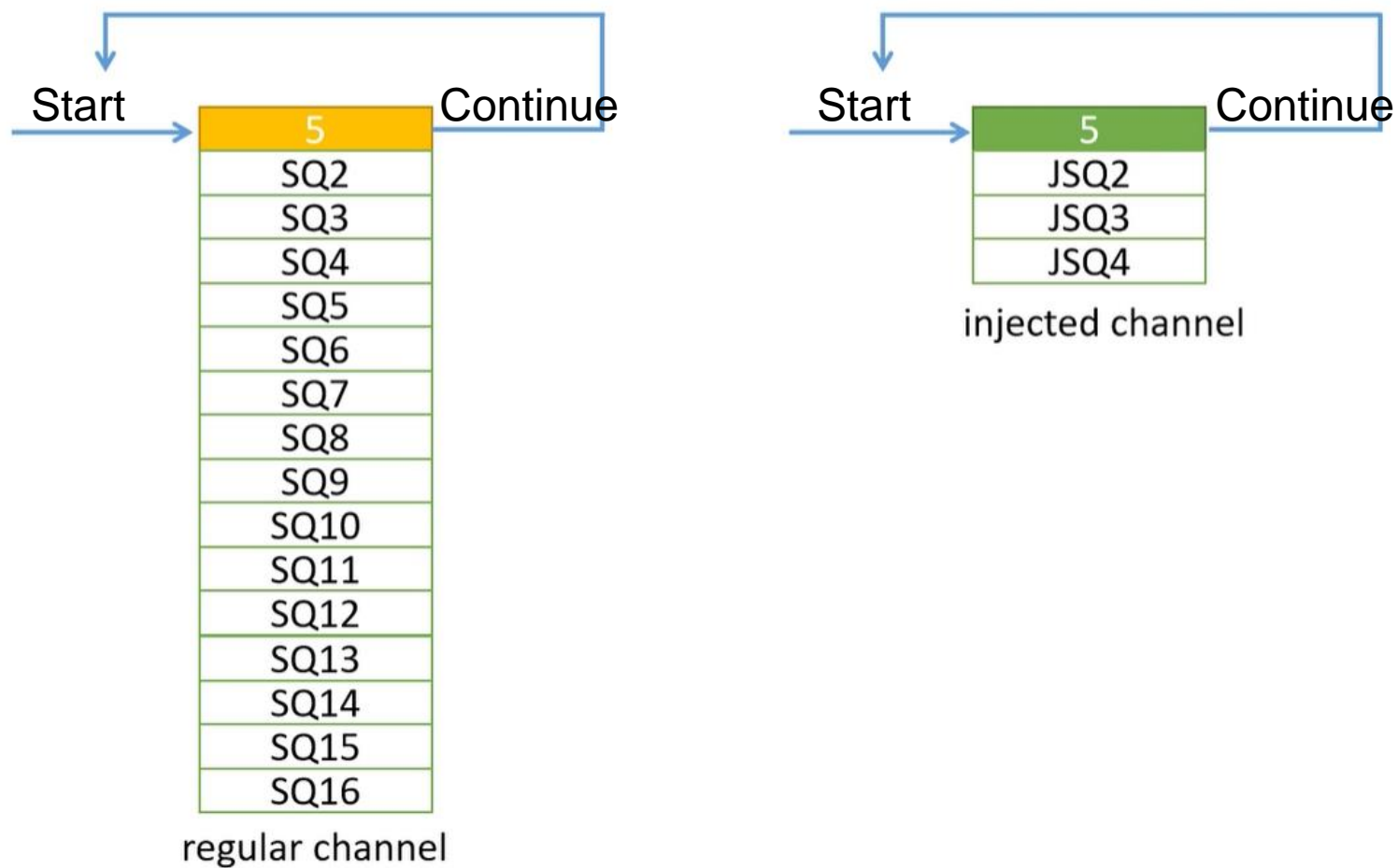
Scan mode

- E.g. continued scan mode

Discontinuous mode
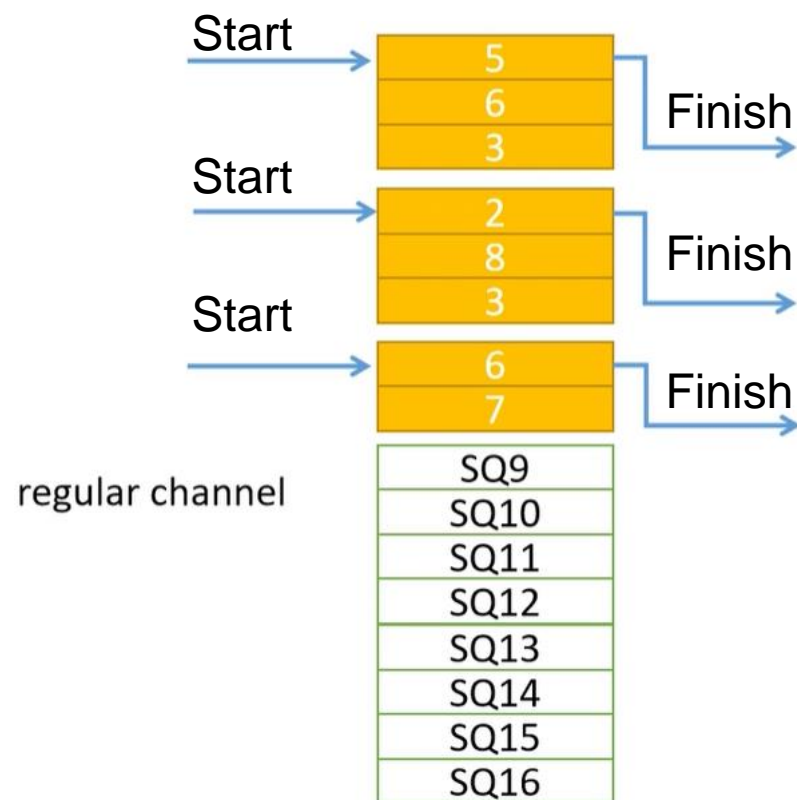
Sample time

- The total conversion time is calculated as follows:

$$T_{ADC} = T_{sampling} + T_{Conversion}$$

- Sampling time is defined in ADCx_SMPR register (1.5 ~ 239.5)
- Conversion time is 12.5 ADC clock cycles
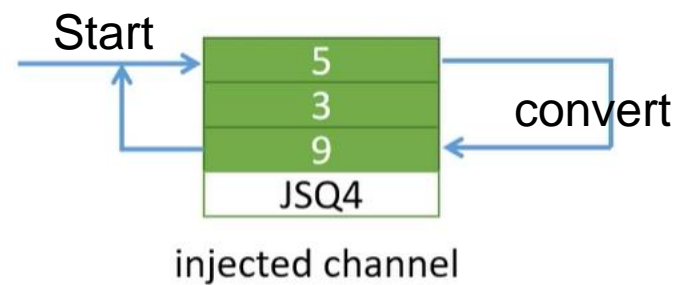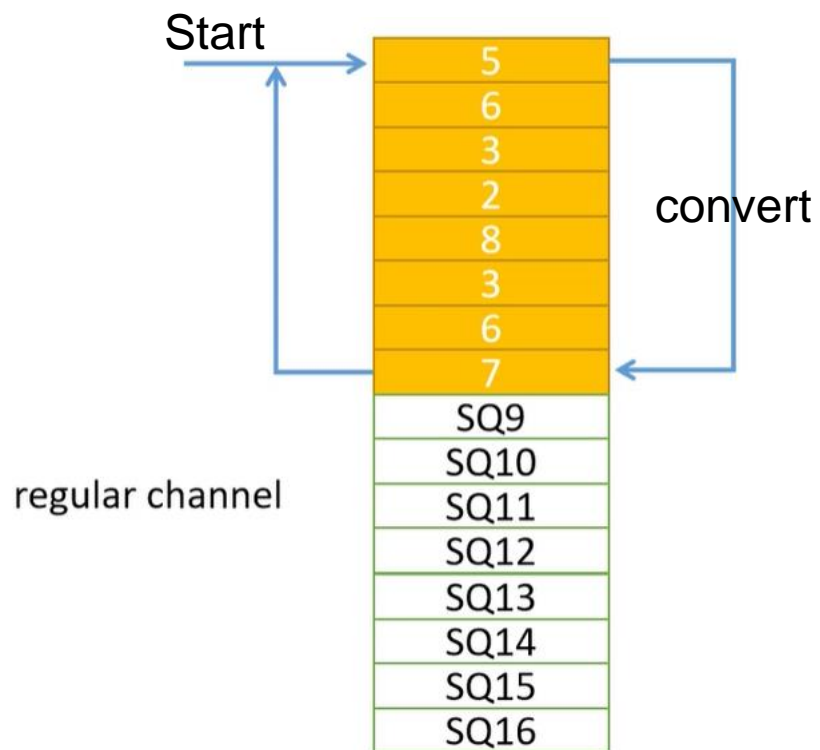
- Example:
  - With an ADCCLK =14 MHz and a sampling time of 1.5 cycles:
  - T_{conv} = 1.5 + 12.5 = 14 cycles = 1 μs

SMP	Sample time	SMP	Sample time
000	1.5 ADC clock cycles	100	41.5 ADC clock cycles
001	7.5 ADC clock cycles	101	55.5 ADC clock cycles
010	13.5 ADC clock cycles	110	71.5 ADC clock cycles
011	28.5 ADC clock cycles	111	239.5 ADC clock cycles

Example 1: ADC with Polling (Busy Waiting)

Main program
(ADC in single conversion mode)

```
ADC1->CR |= ADC_CR_ADSTART;
```

Wait until `EOC_MST` is set in ADC common status register
(CSR)

```
Result = ADC1->DR;
```

Repeat if more conversions are needed

Software starts one ADC conversion on a single channel.

ADC

Hardware sets `EOC_MST` if done.
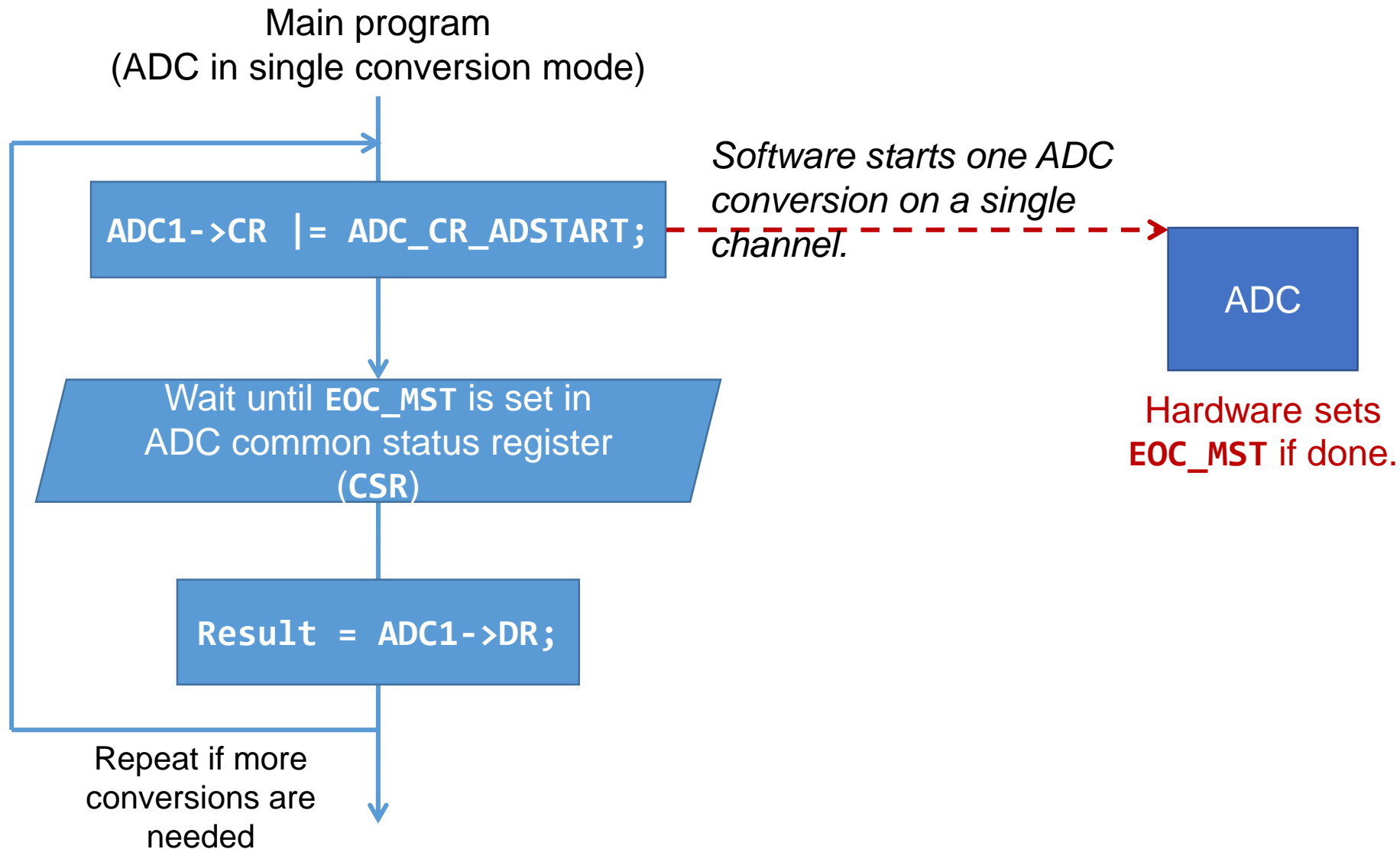
Example 2: ADC with Interrupts

Main program

Configure and
Starts ADC

An ADC interrupt
request arrives.

```c
void ADC1_2_IRQHandler(void){
    if (ADC1->ISR | ADC_ISR_EOC){
        buffer[i] = ADC1->DR;
        i++;
    }
    ...
}
```
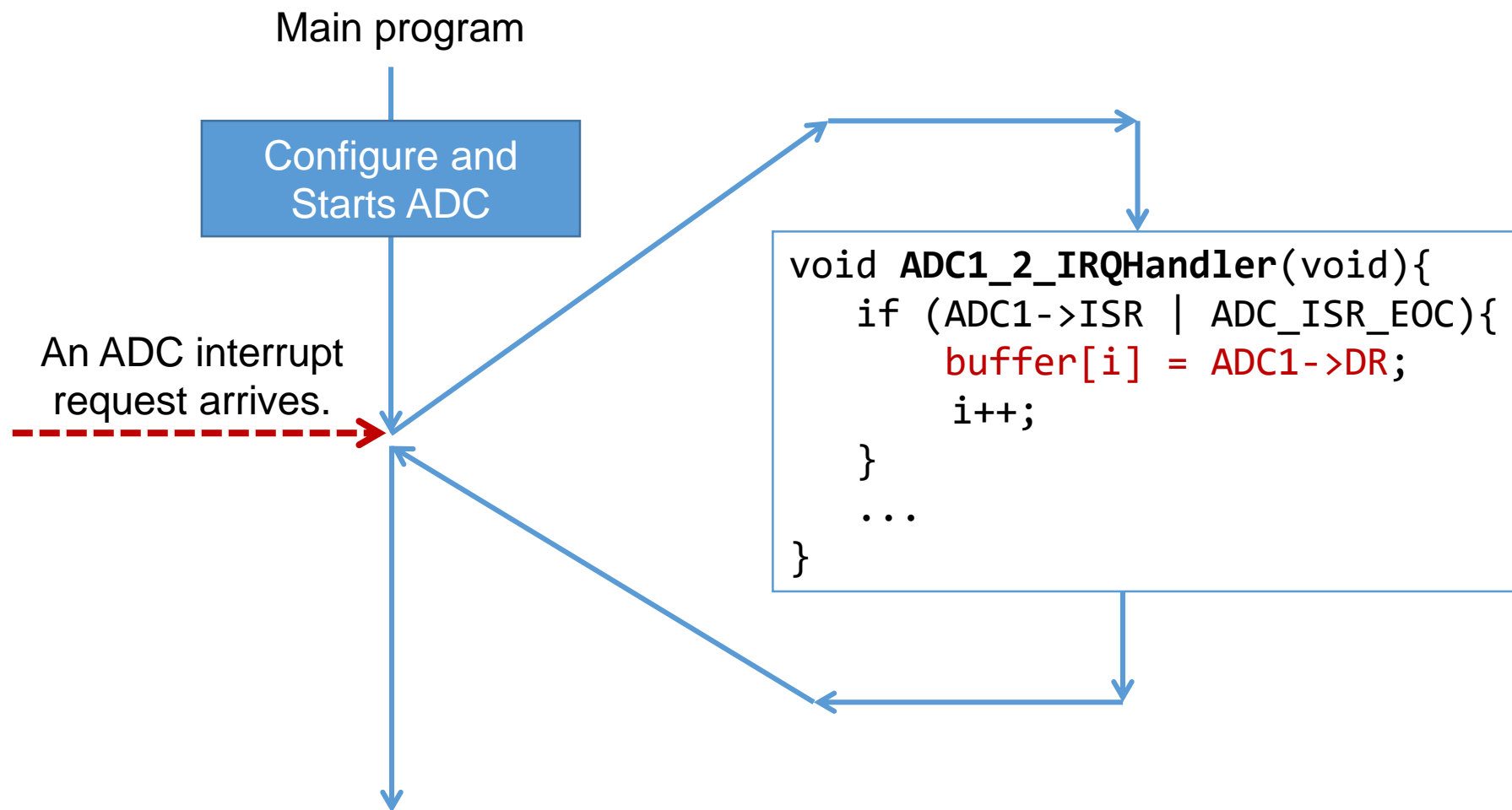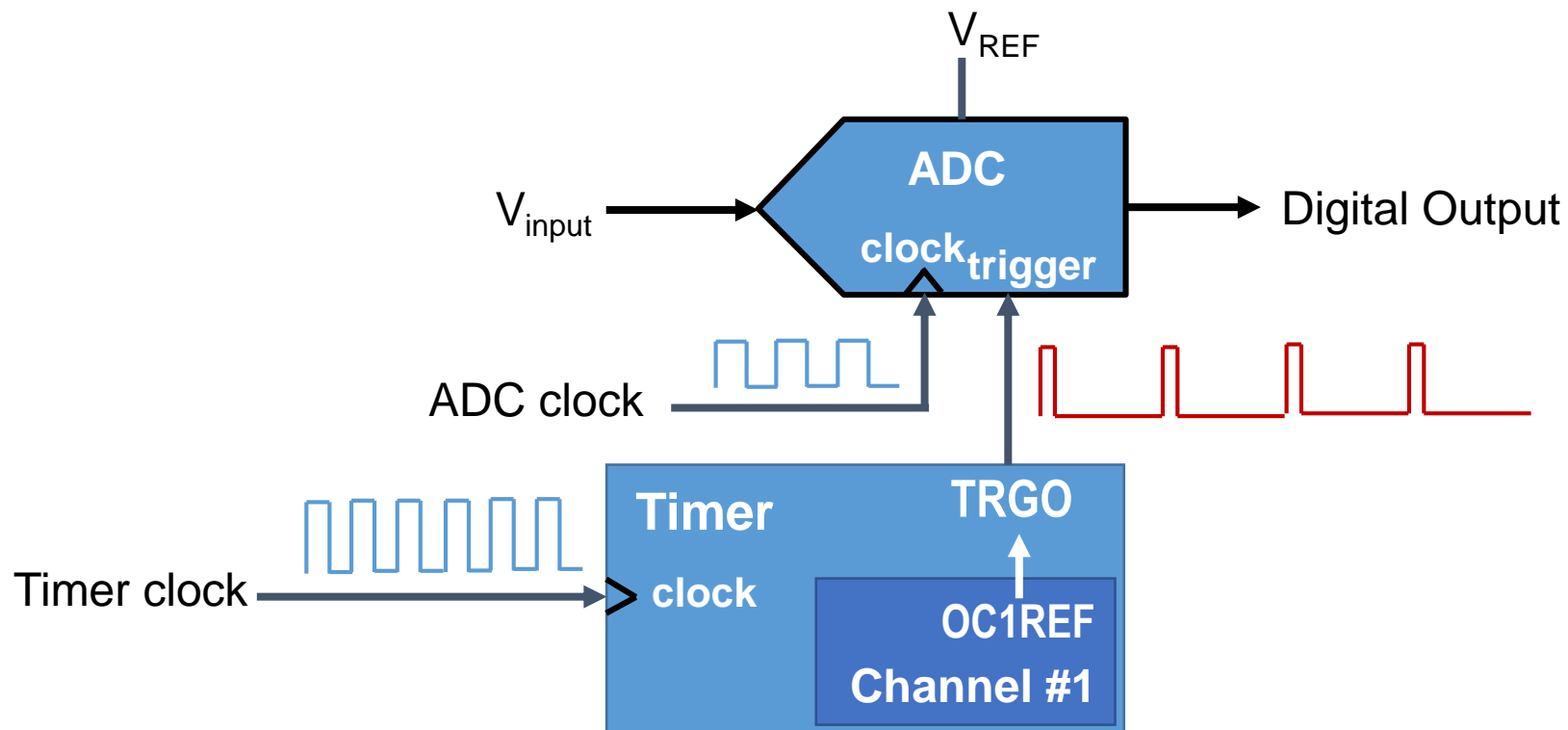
Example 3: ADC Triggered by a Timer

Example 3: ADC Triggered by a Timer

main program

start timer

Timer

TRGO
signals

ADC

interrupt request

ADC ISR

Wait for
ADC_Done = 1

Process Data

Repeat

```
volatile uint ADC_Done = 0;

void ADC1_2_IRQHandler(void){
    if (ADC1->ISR | ADC_ISR_EOC){
        result[i] = ADC1->DR;
        i++;
        if (i >= N)
            ADC_Done = 1;
    }
    ...
}
```

Make N sampling
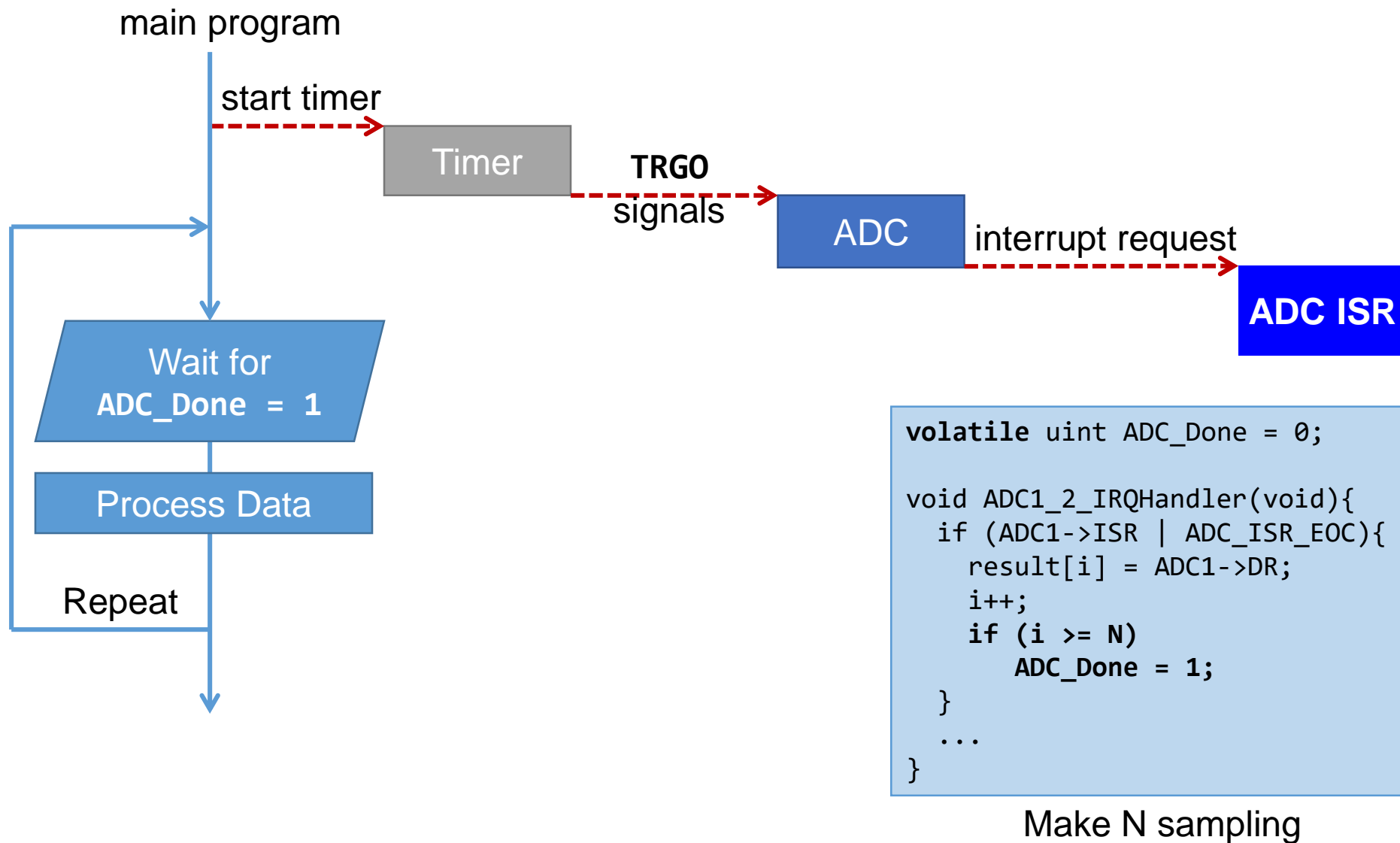
Example 4: ADC Triggered by a Timer with DMA

start timer

Timer

TRGO signals

ADC

DMA request

DMA controller

interrupt request

DMA ISR

DMA copies ADC DR register to memory after each conversion.

Wait for
`DMA_Done = 1`

Process Data

Repeat

```
volatile uint DMA_Done = 0;

void DMA1_Channel1_IRQHandler(void){
  ...
  DMA_Done = 1;
  ...
}
```

Make N sampling