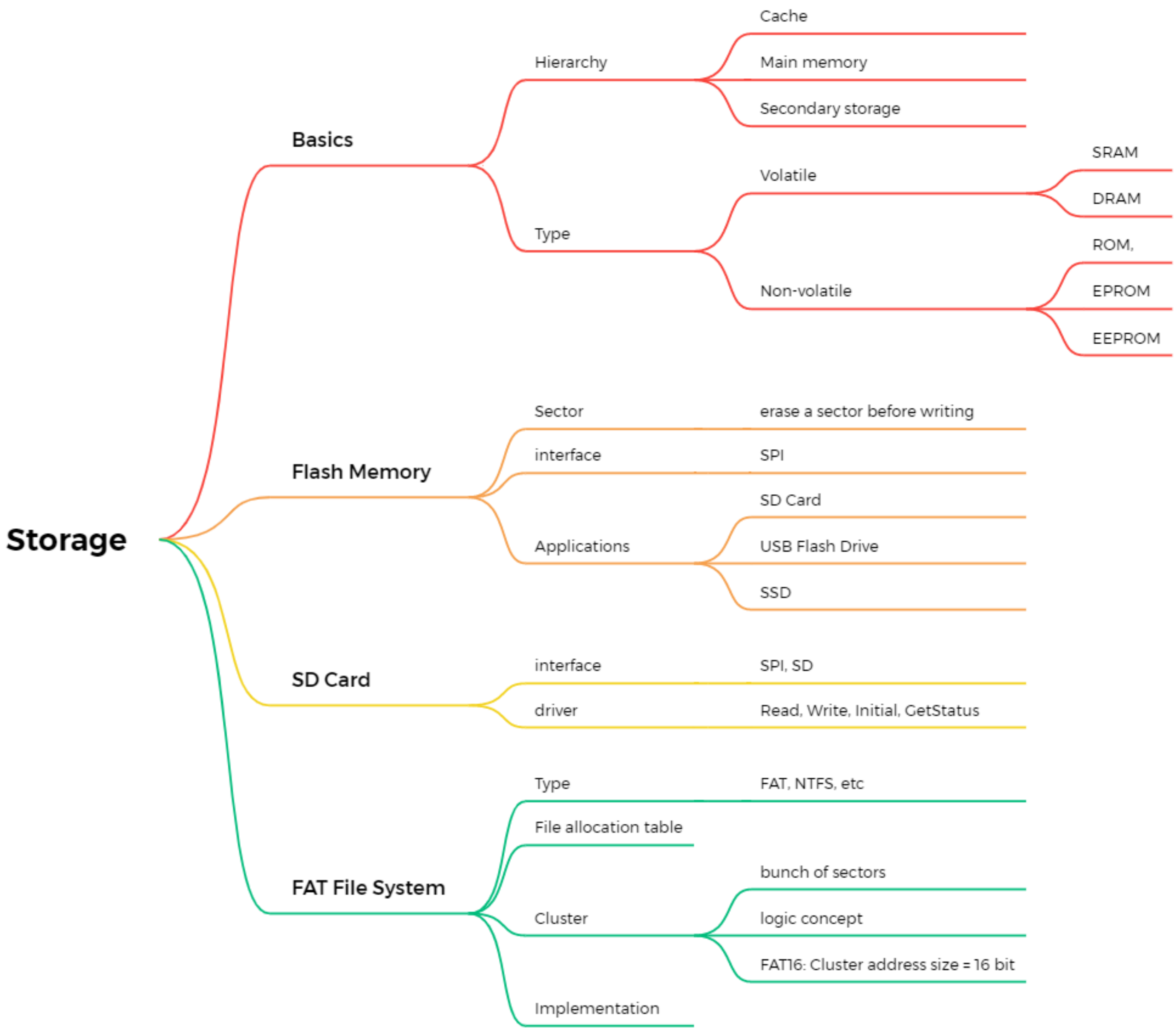
CS301
Embedded System and Microcomputer Principle

Lecture 13: ADC

2023 Fall

Recap

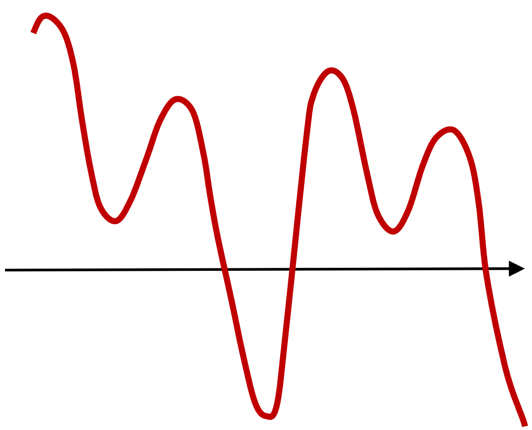

Storage
Basics
Hierarchy
Cache
Main memory
Secondary storage
Type
Volatile
SRAM
DRAM
Non-volatile
ROM,
EPROM
EEPROM
Flash Memory
Sector
erase a sector before writing
interface
SPI
Applications
SD Card
USB Flash Drive
SSD
SD Card
interface
SPI, SD
driver
Read, Write, Initial, GetStatus
FAT File System
Type
FAT, NTFS, etc
File allocation table
Cluster
bunch of sectors
logic concept
FAT16: Cluster address size = 16 bit
Implementation

Outline

- **ADC introduction**
- STM32 ADC

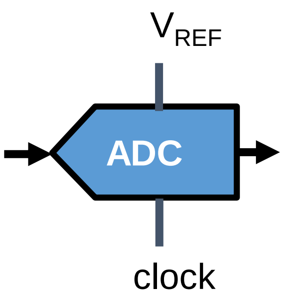
Analog-to-Digital Converter (ADC)

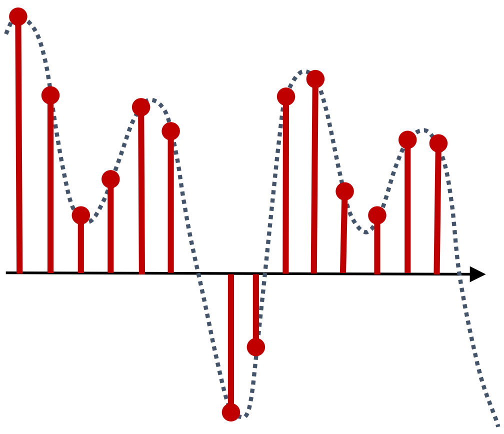
- ADC: a circuit that takes in an analog voltage and produces a digital representation of its value
  - To know nature phenomena, which is analog, and make it feasible for computer to handle, we need to convert it into digital signals
  - To transform the analog, continuous signals into digital ones,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



Analog signal $x(t)$

continuous time
continuous amplitude



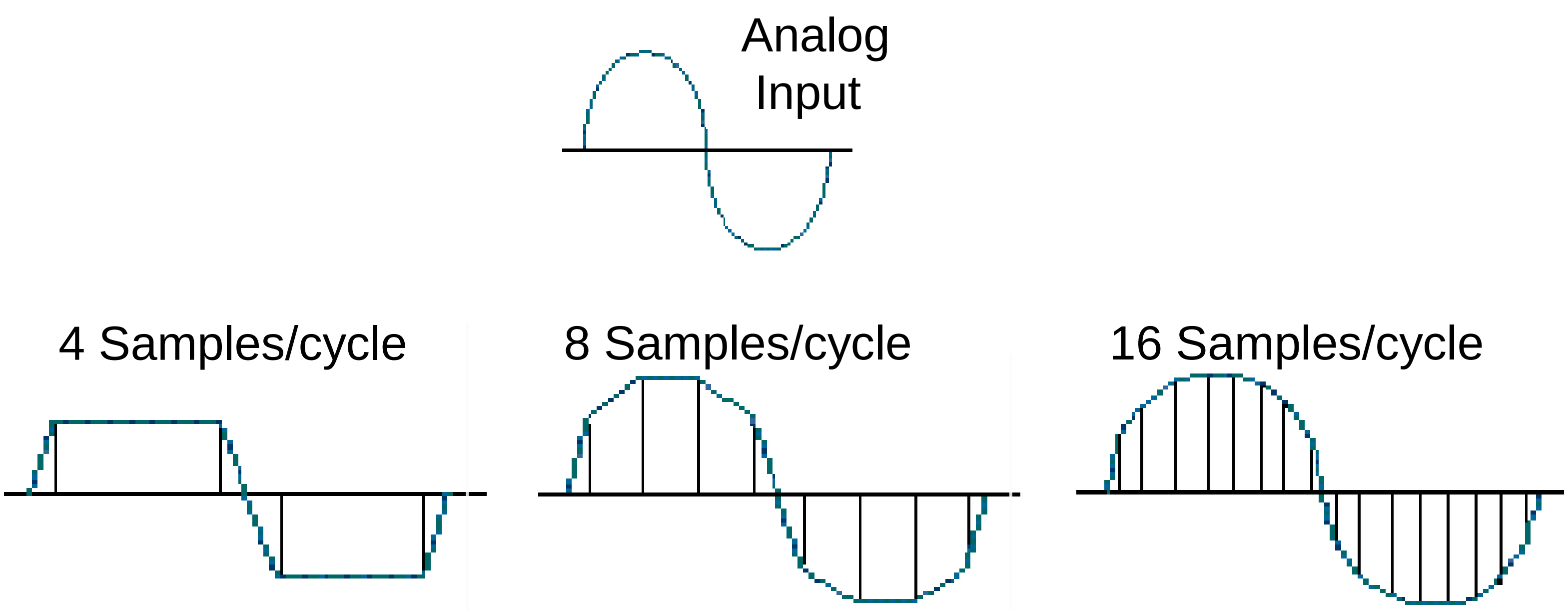


Digital values $x(n)$

discrete time
discrete amplitude

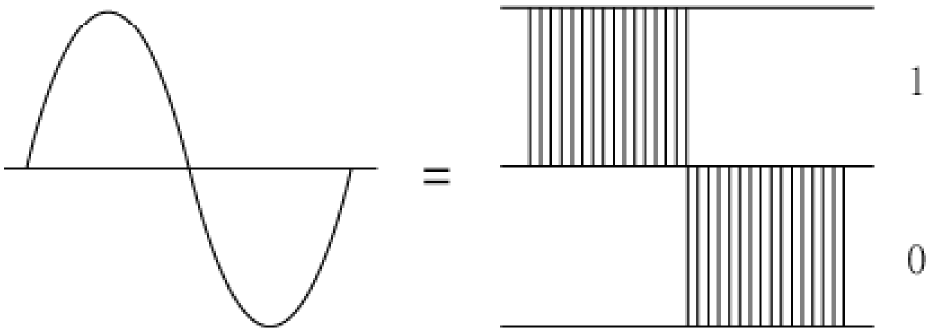
Sampling

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

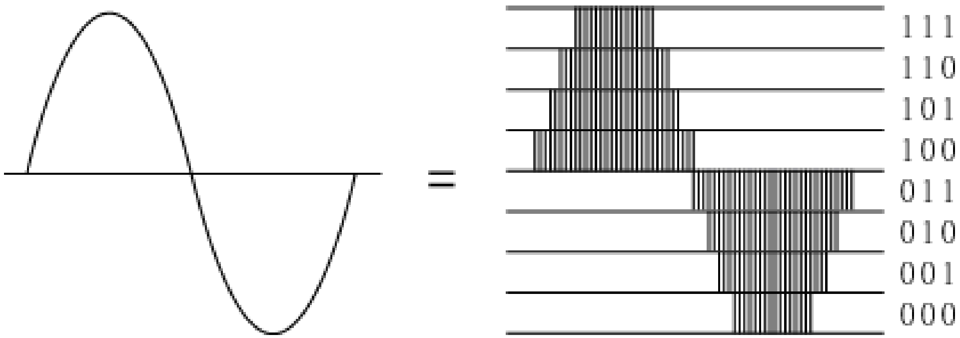


Sampling

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



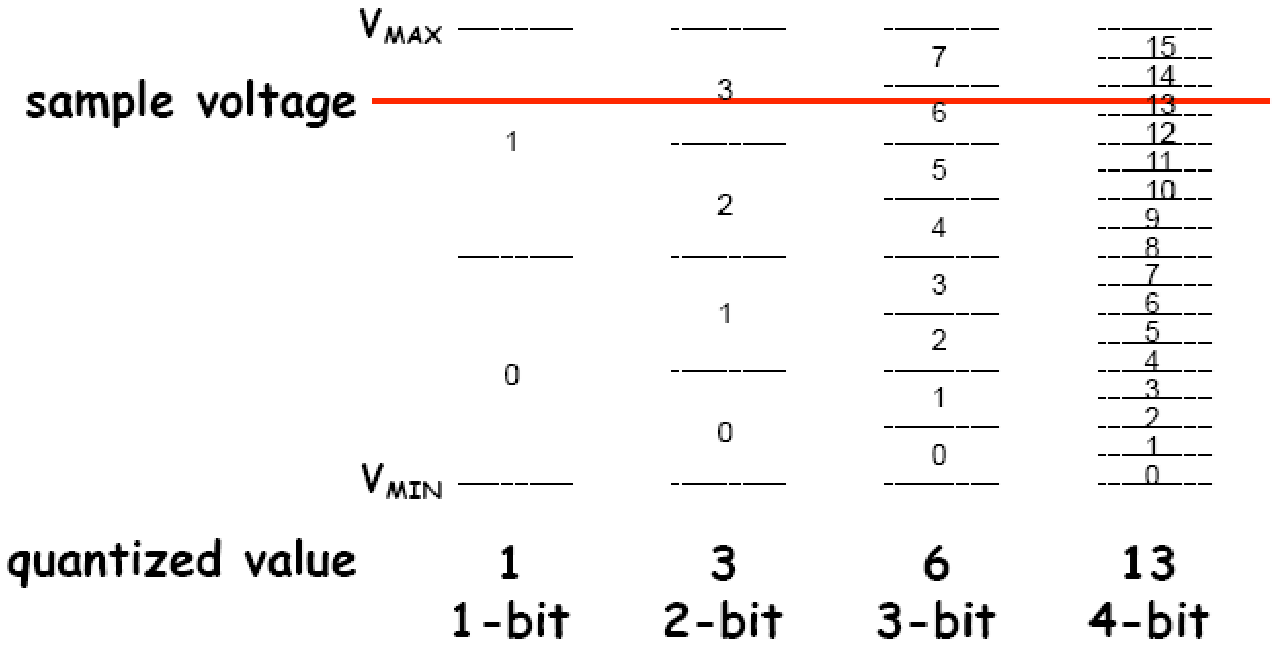
1-bit



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

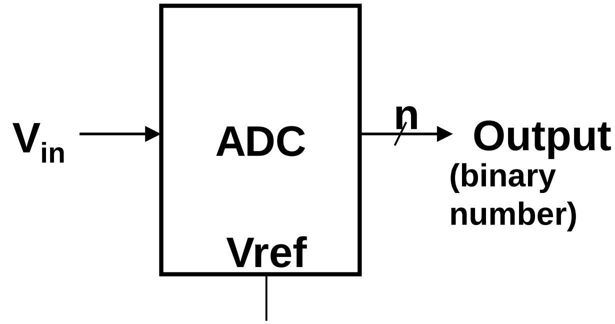
n-bit	Number of steps	Step size
8	256	5V /256 = 19.53 mV
10	1024	5V /1024 = 4.88 mV
12	4096	5V /4096 = 1.2 mV
16	65,536	5V /65,536 = 0.076 mV
E.G. : V_{ref} = 5V		

Resolution

- Calculate digital data output

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

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

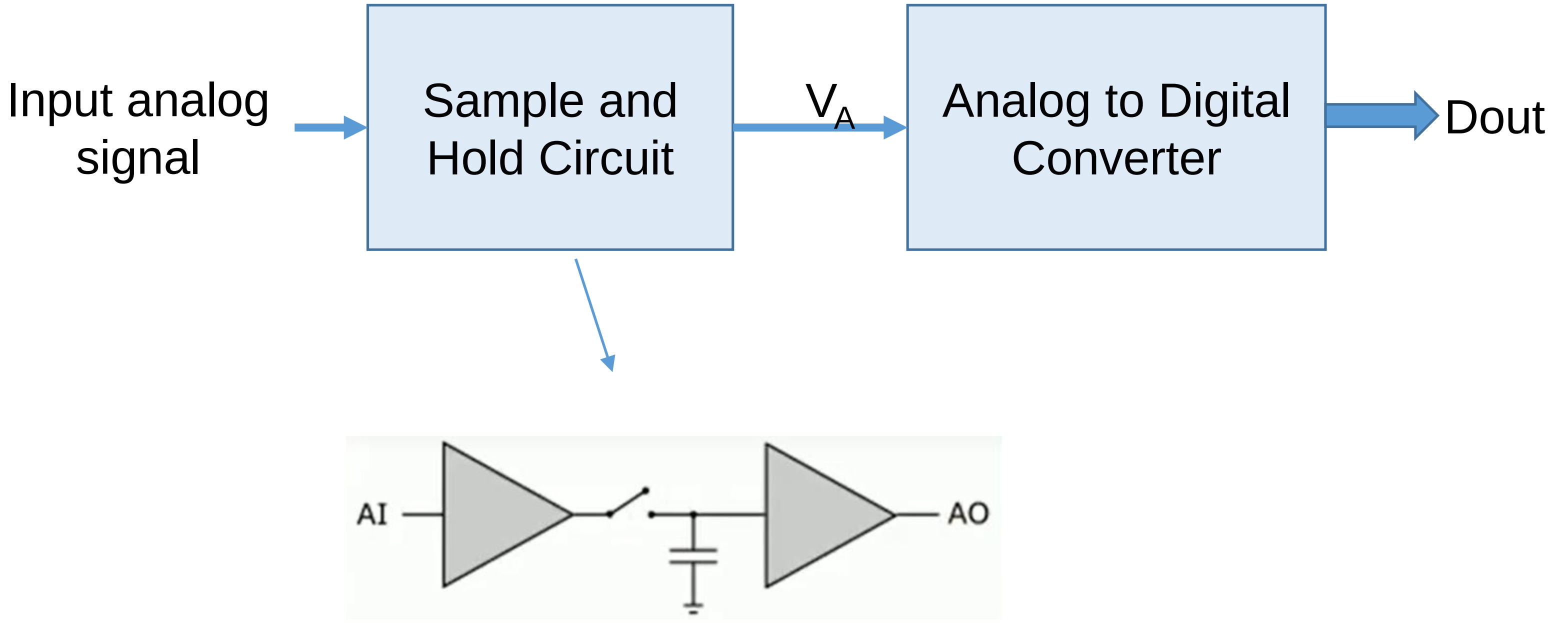


$$\text{Output} = \text{round}\left((2^N \times \frac{V_{input}}{V_{REF}} \right)$$

- 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.

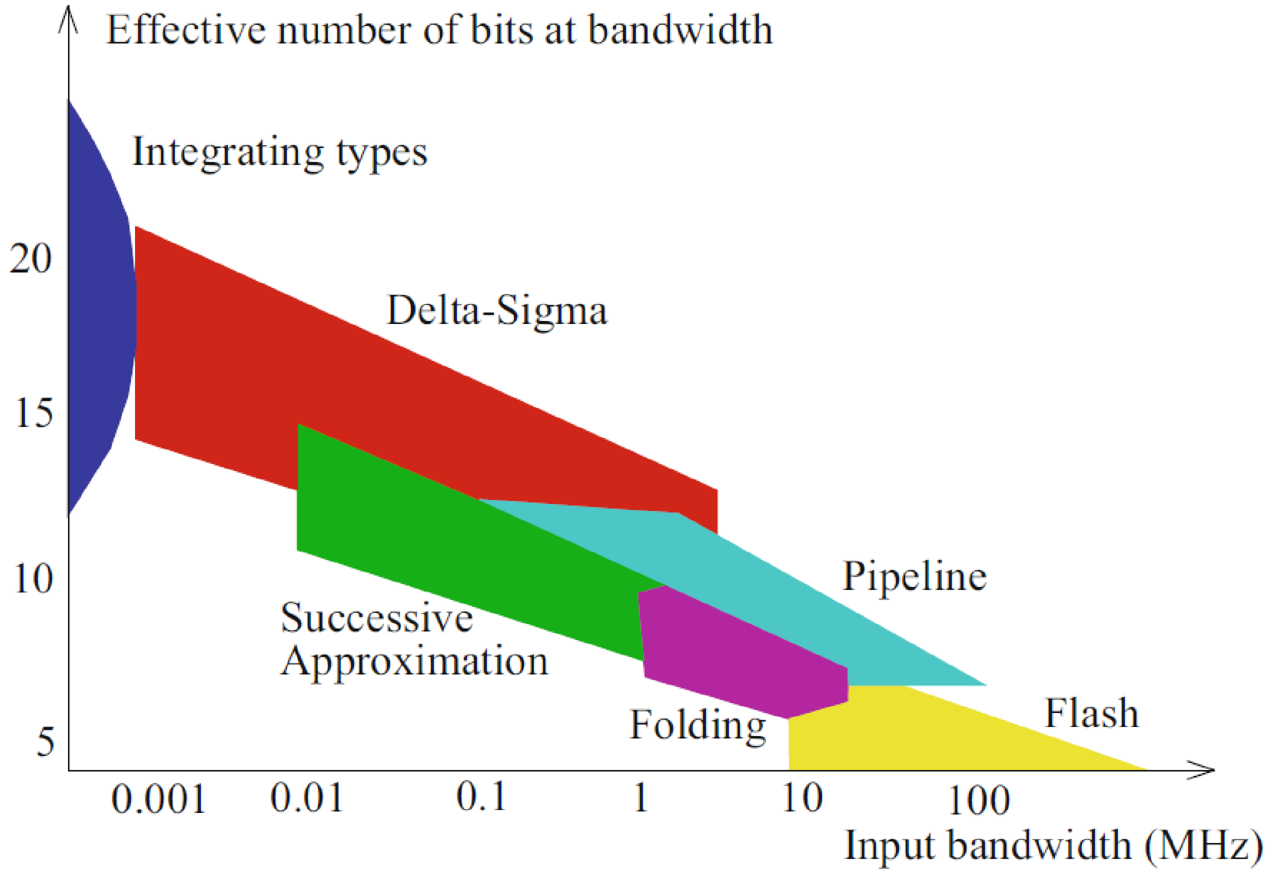


ADC Architecture

- 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

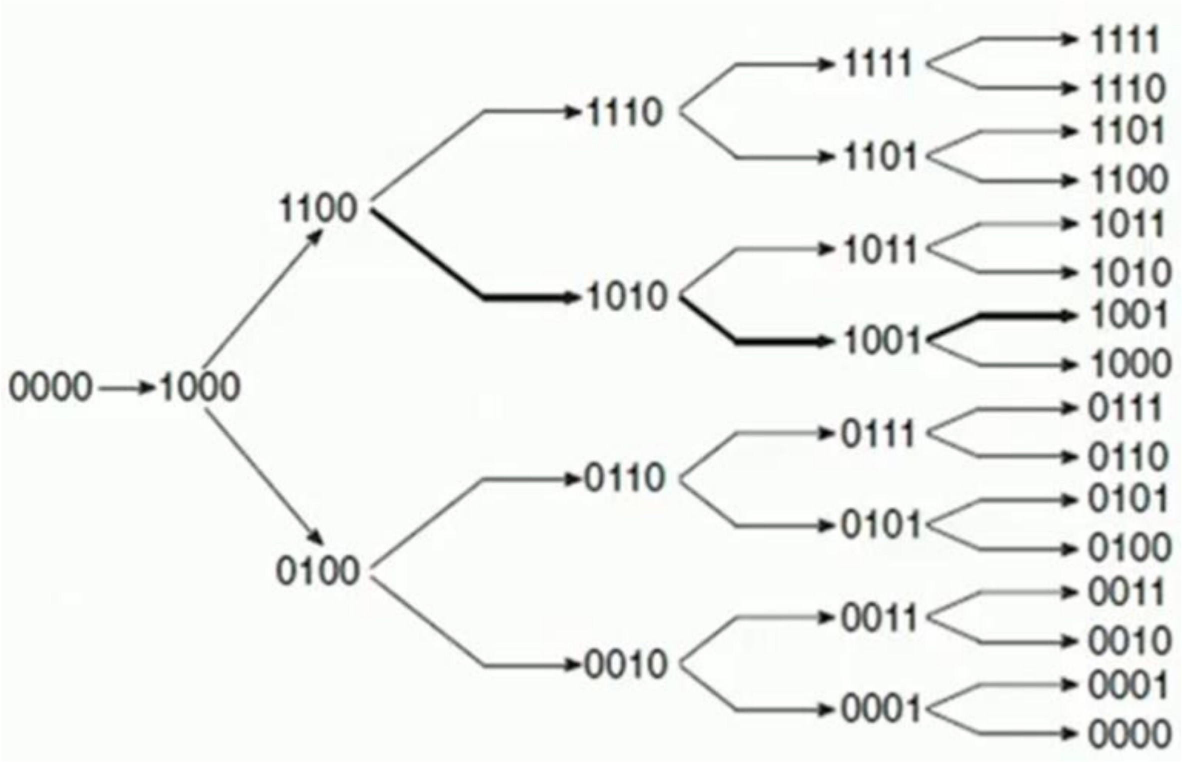
ADC Architecture

- Comparison of the speed/resolution characteristics of various ADCs



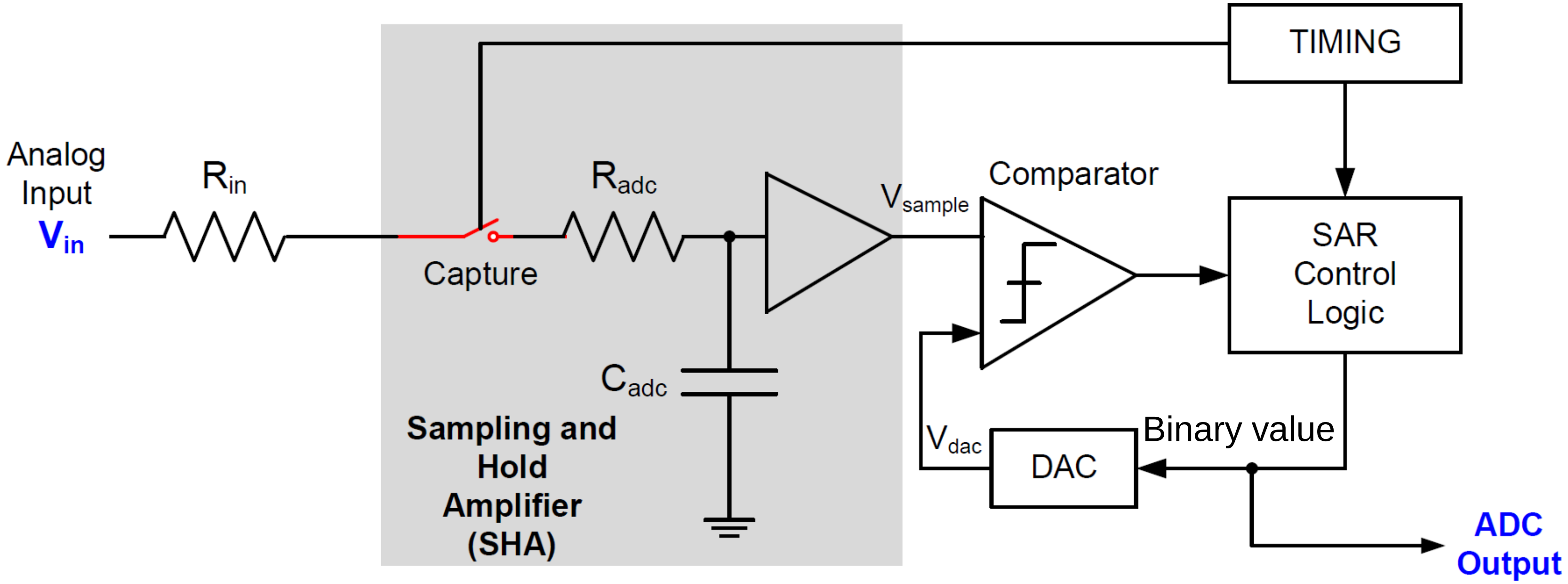
Successive-approximation (SAR)

- 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).



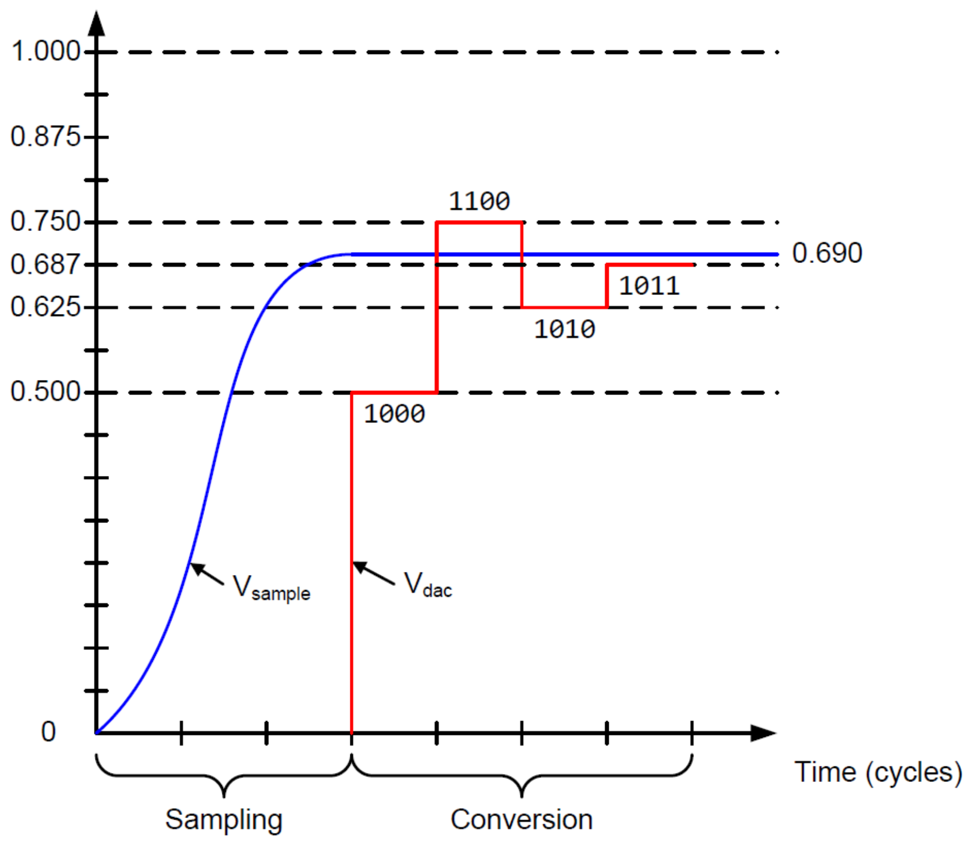
Successive-approximation (SAR)

- 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)

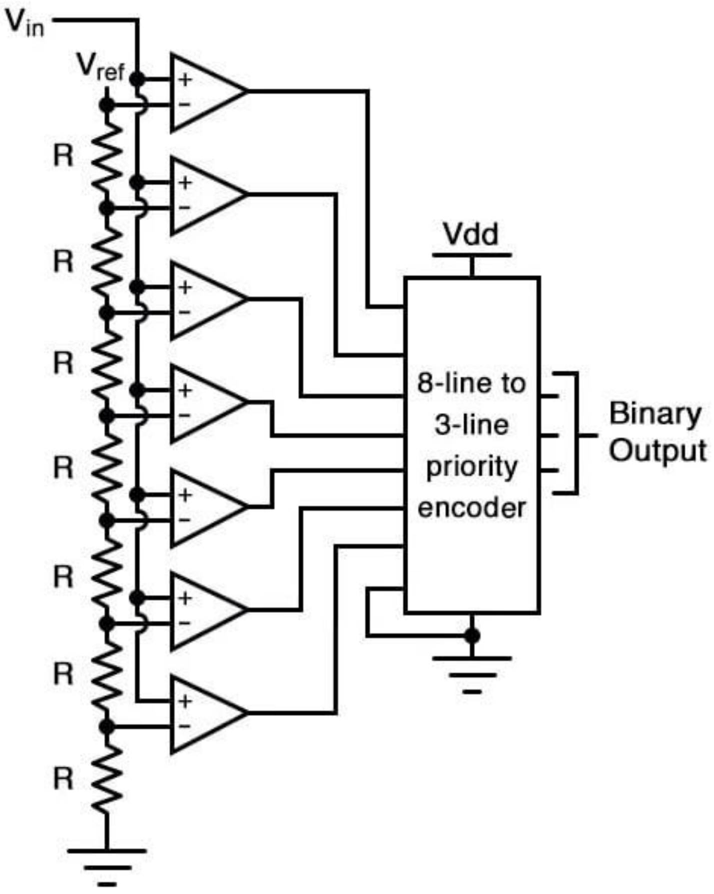
- Conversion time: number of iterations, usually N cycles for N-bit ADC. In STM32 12-bit ADC, T conversion = 12.5 cycles



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

Flash ADC

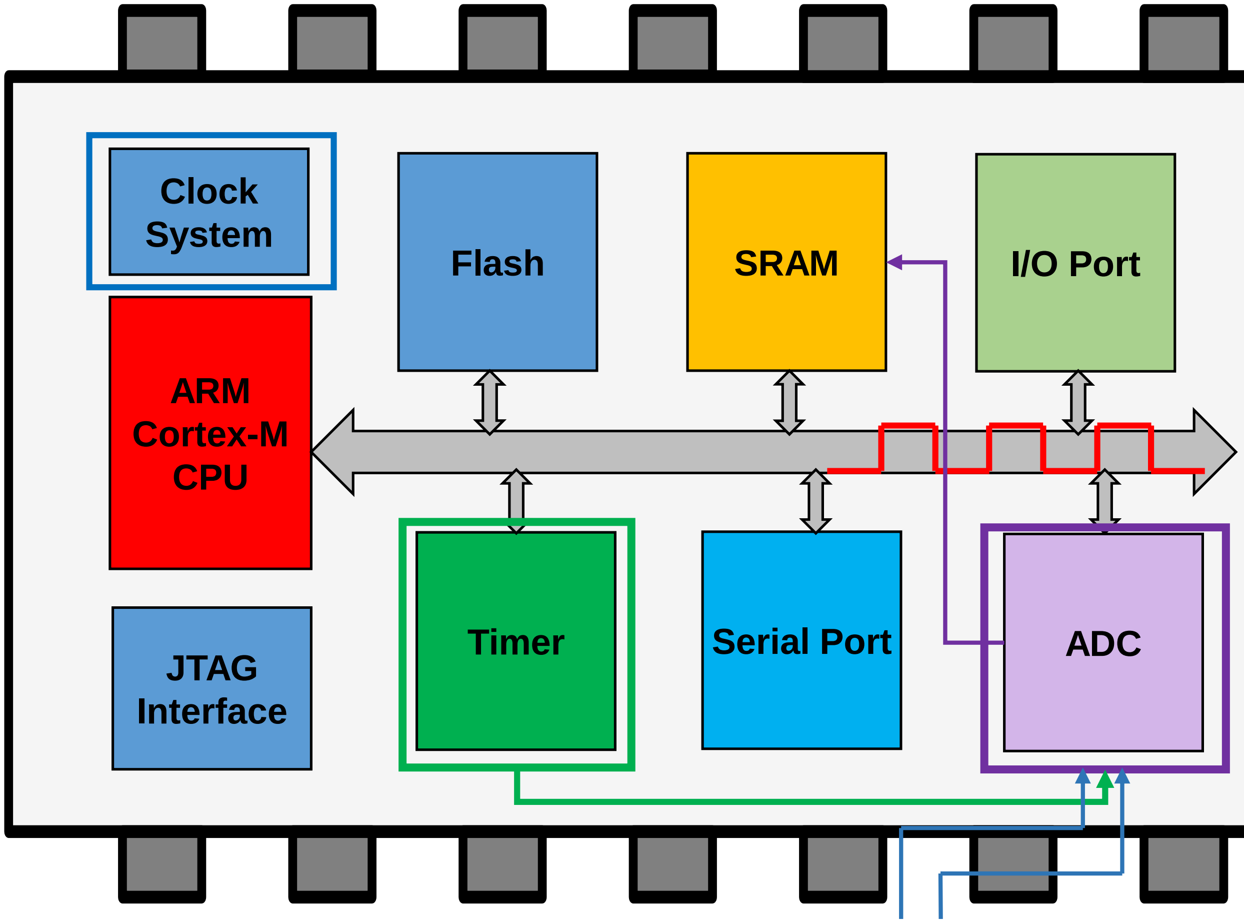
- Aka. Parallel ADC
- A series of comparators, each one comparing the input signal to a unique reference voltage. The comparator outputs connect to the inputs of a priority encoder circuit, which then produces a binary output.



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

ADC working flow

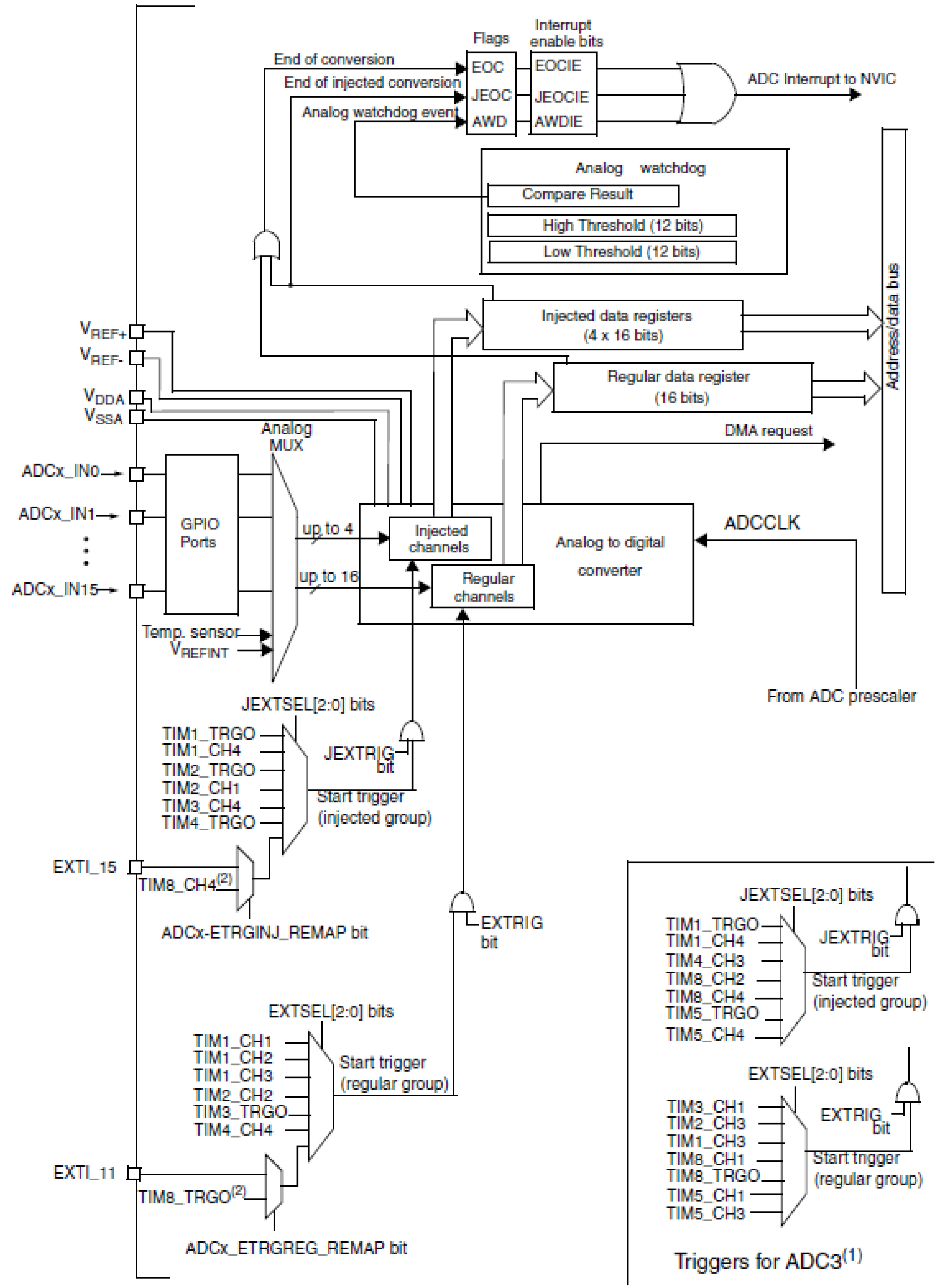
- Provide continuous sampling of multiple analog inputs and store sampled data



Outline

- ADC introduction
- **STM32 ADC**

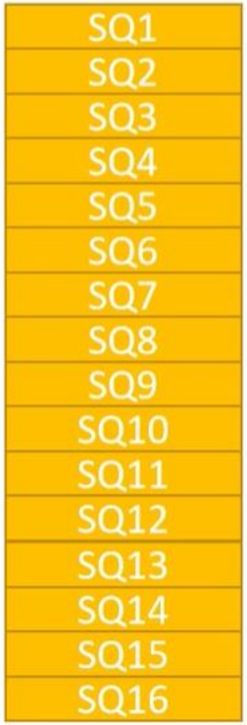
STM32 ADC



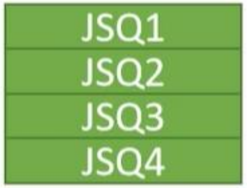
Flags
Interrupt enable bits
End of conversion
End of injected conversion
Analog watchdog event
EOC
JEOC
AWD
EOCIE
JEOCIE
AWDIE
ADC Interrupt to NVIC
Analog watchdog
Compare Result
High Threshold (12 bits)
Low Threshold (12 bits)
Injected data registers (4 x 16 bits)
Regular data register (16 bits)
Address/data bus
DMA request
V REF+
V REF-
V DDA
V SSA
Analog MUX
ADCx_IN0
ADCx_IN1
ADCx_IN15
GPIO Ports
up to 4
up to 16
Injected channels
Regular channels
Analog to digital converter
ADCCLK
From ADC prescaler
Temp. sensor
V REFINT
JEXTSEL[2:0] bits
TIM1_TRGO
TIM1_CH4
TIM2_TRGO
TIM2_CH1
TIM3_CH4
TIM4_TRGO
JEXTRIG bit
Start trigger (injected group)
EXTI_15
TIM8_CH4(2)
ADCx-ETRGINJ_REMAP bit
EXTRIG bit
EXTSEL[2:0] bits
TIM1_CH1
TIM1_CH2
TIM1_CH3
TIM2_CH2
TIM3_TRGO
TIM4_CH4
Start trigger (regular group)
EXTI_11
TIM8_TRGO(2)
ADCx_ETRGREG_REMAP bit
JEXTSEL[2:0] bits
TIM1_TRGO
TIM1_CH4
TIM4_CH3
TIM8_CH2
TIM8_CH4
TIM5_TRGO
TIM5_CH4
JEXTRIG bit
Start trigger (injected group)
EXTSEL[2:0] bits
TIM3_CH1
TIM2_CH3
TIM1_CH3
TIM8_CH1
TIM8_TRGO
TIM5_CH1
TIM5_CH3
EXTRIG bit
Start trigger (regular group)
Triggers for ADC3(1)

ADC Conversion Modes

- regular group: composed of up to 16 channels
- injected group: composed of up to 4 channels
- Injected group has higher priority

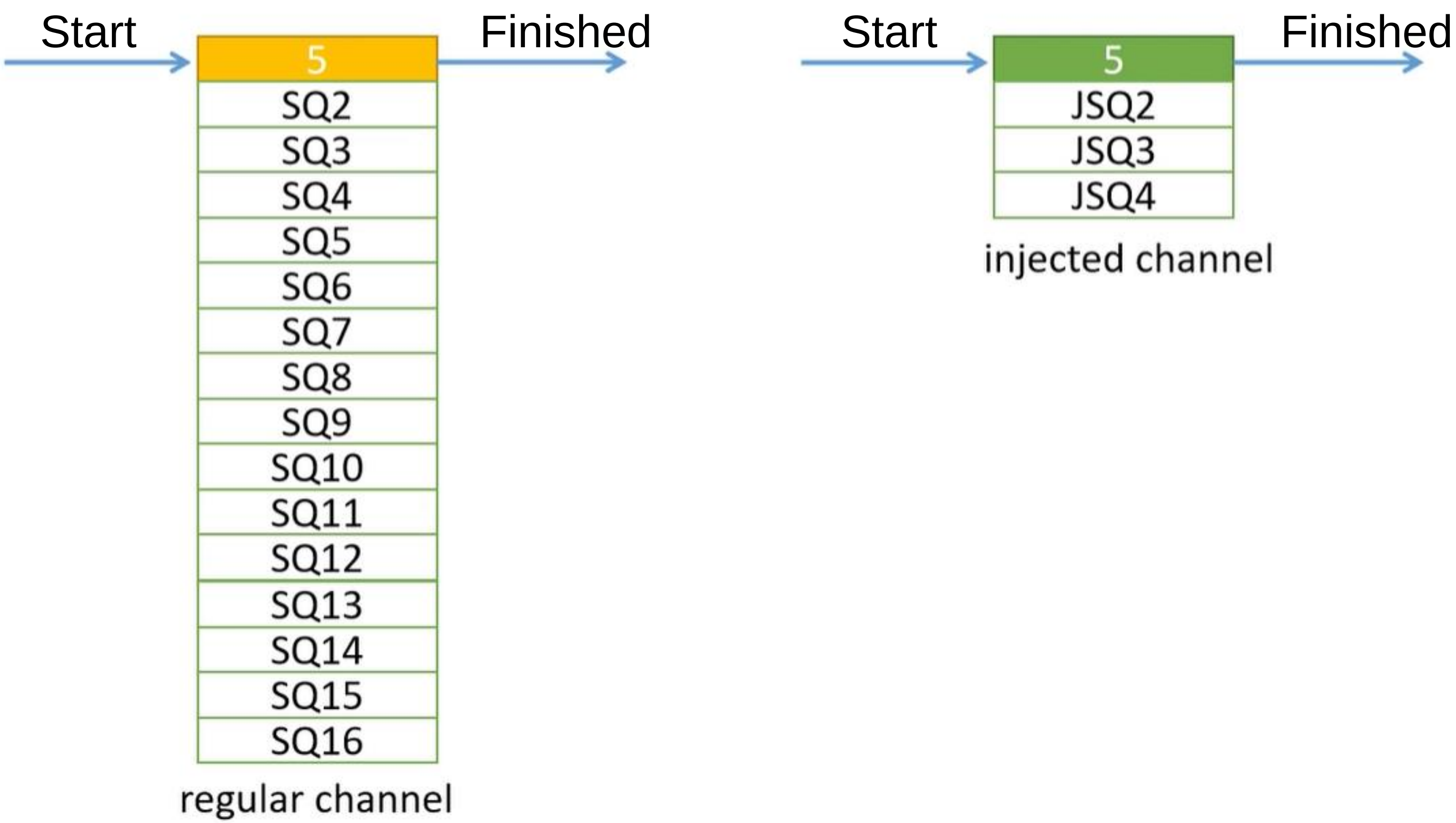


regular channel

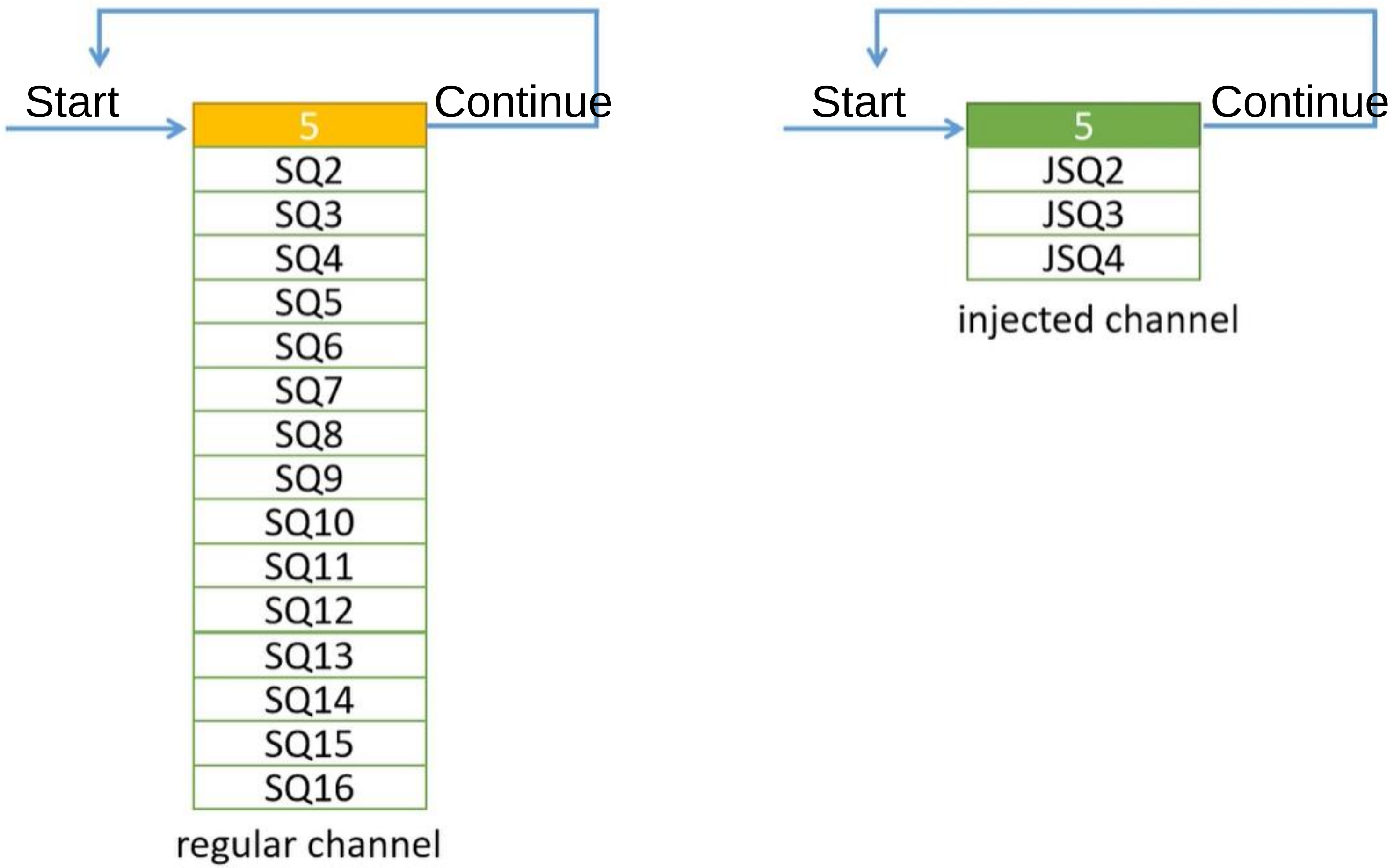


injected channel

Single conversion mode

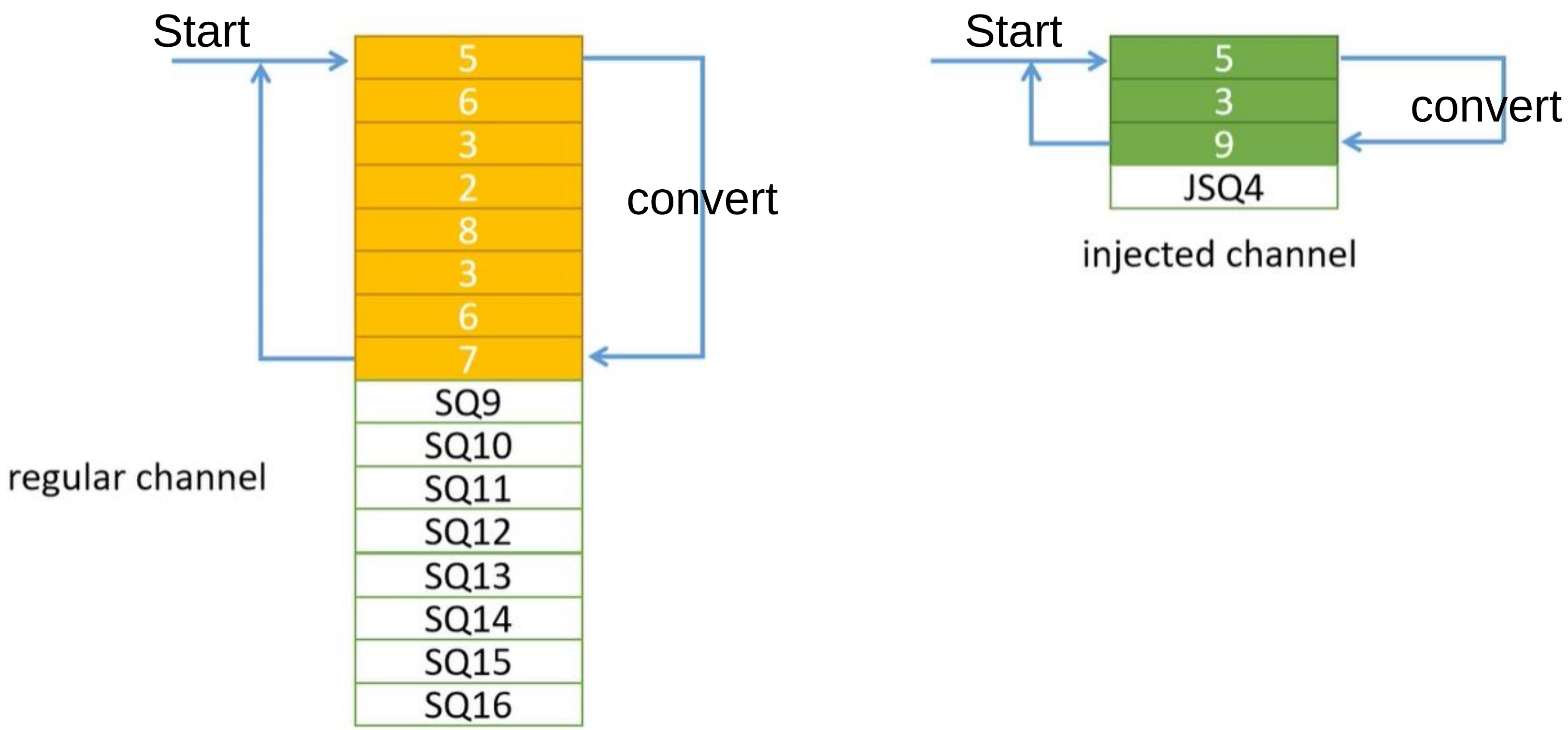


Continuous conversion mode

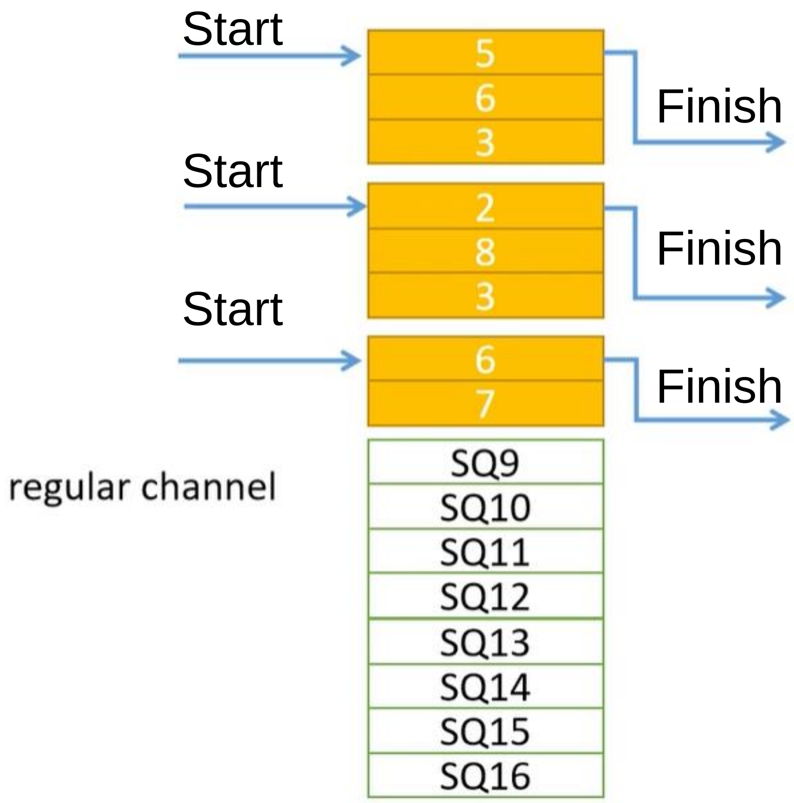


Scan mode

- E.g. continued scan mode



Discontinuous mode



Sample time

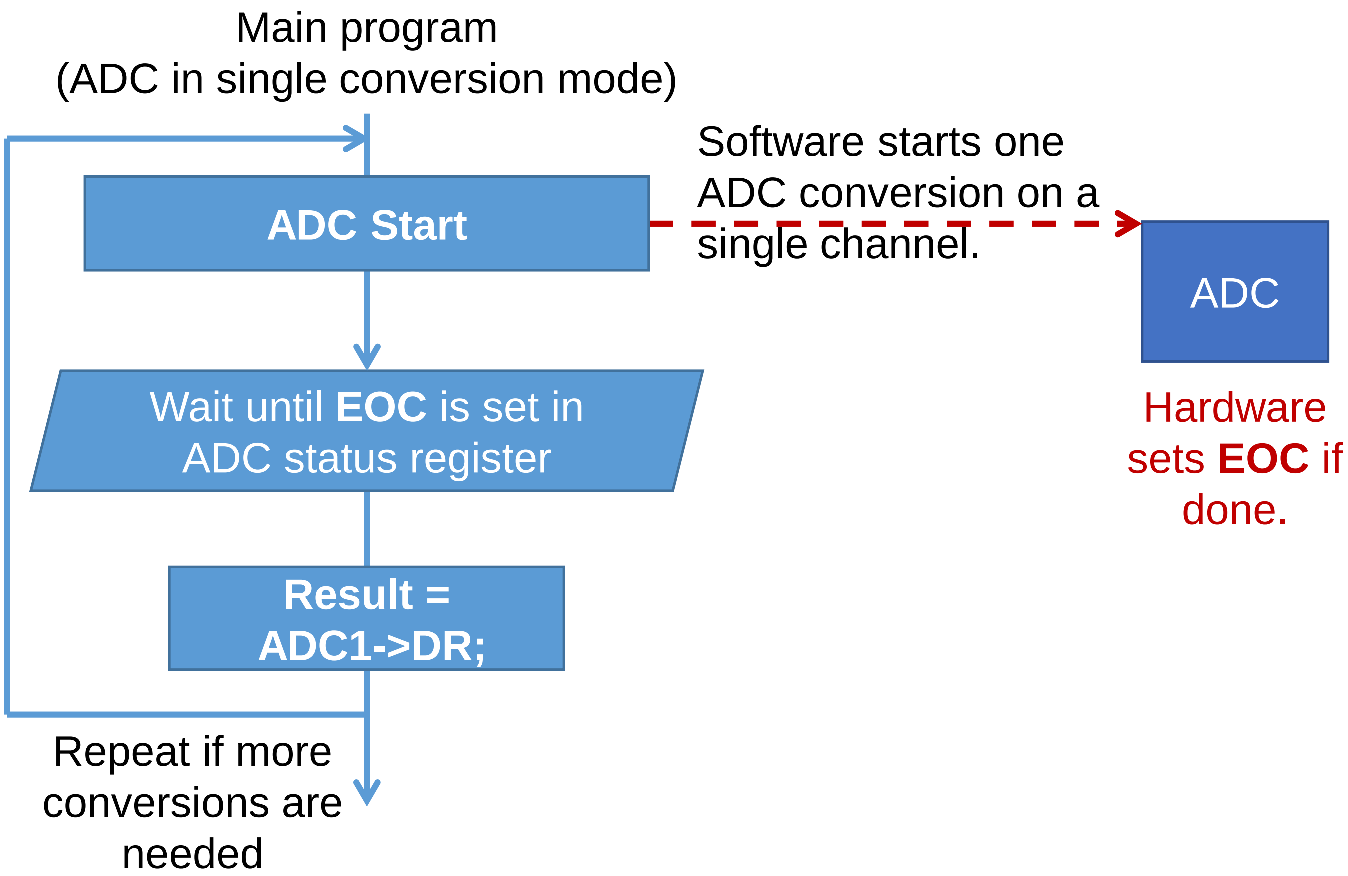
- The total conversion time is calculated as follows:

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

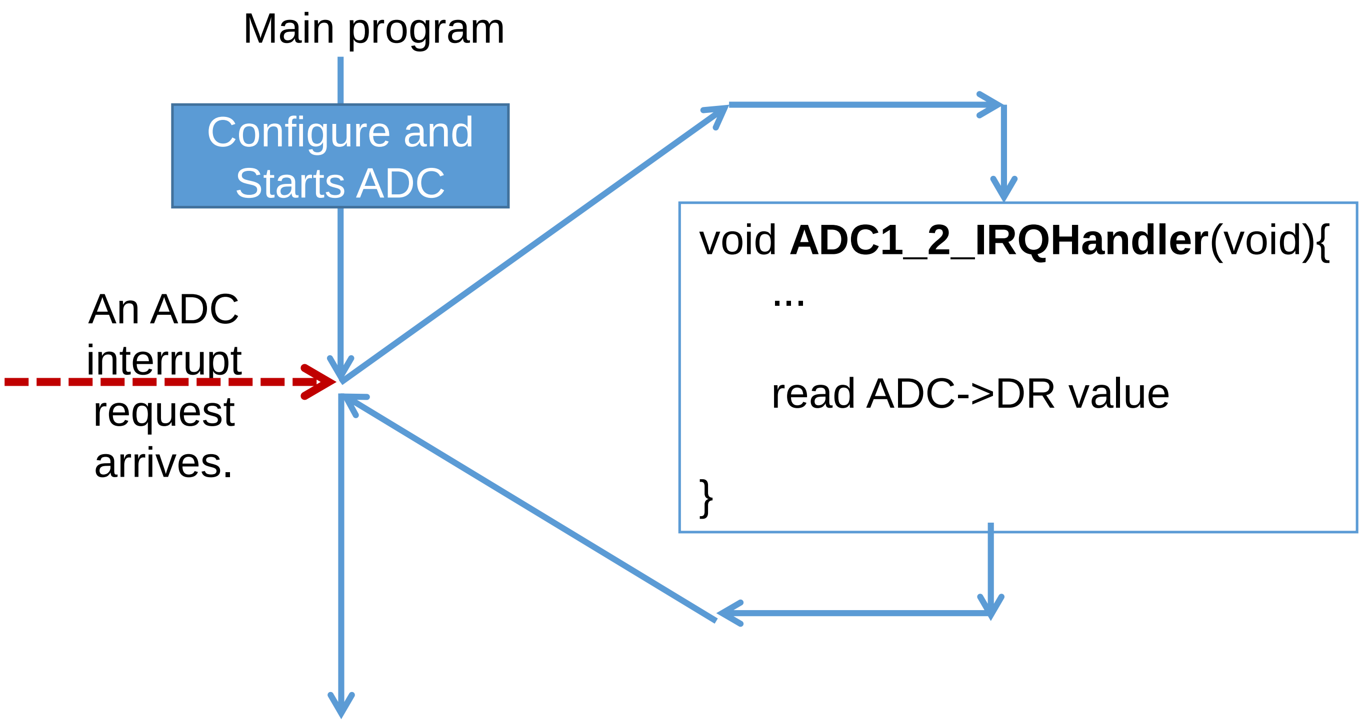
  - Sampling time is defined in ADCx_SMPR register (1.5 ~ 239.5)
  - Conversion time is 12.5 ADC clock cycles
- Example:
  - With an ADCCLK =14 MHz and a sampling time of 1.5 cycles:
  - T_{conv} = 1.5 + 12.5 = 14 cycles = 1 μs

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

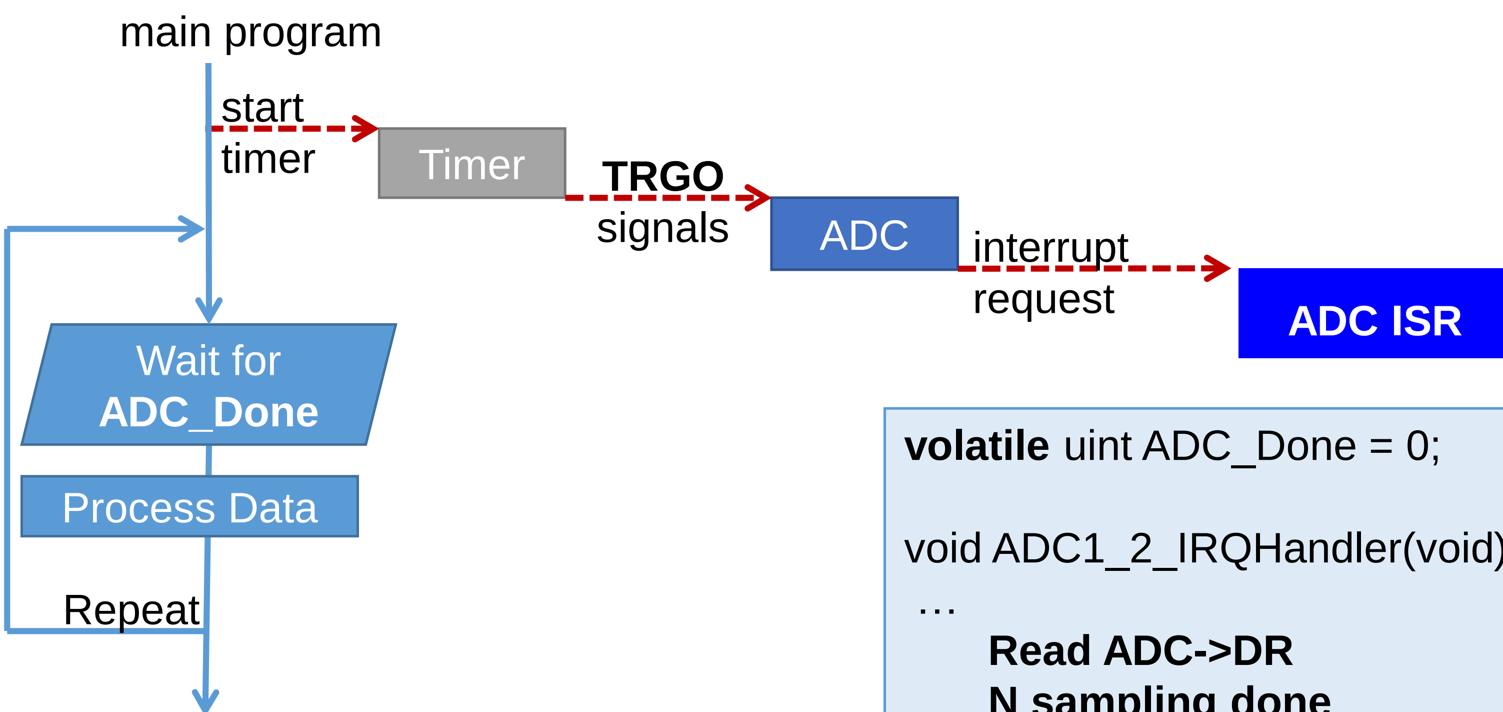
ADC with Polling



ADC with Interrupts



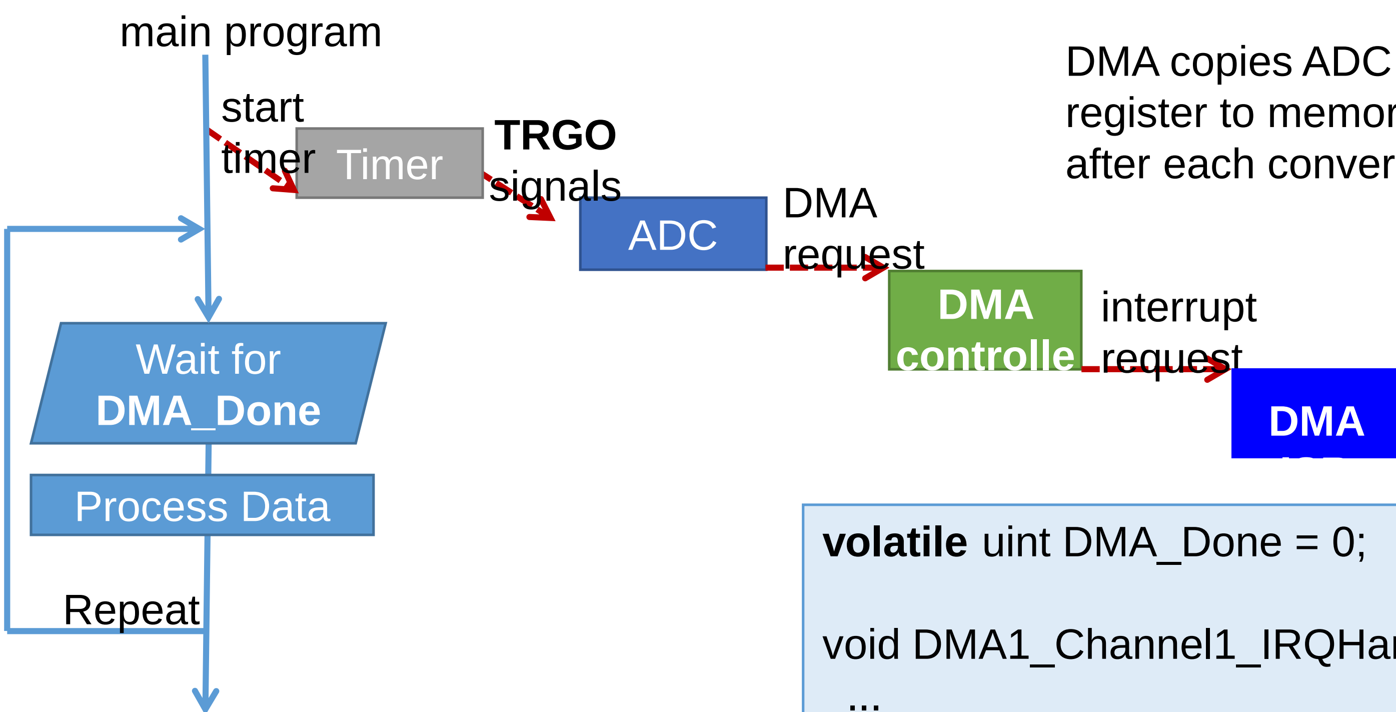
ADC Triggered by a Timer




```
volatile uint ADC_Done = 0;

void ADC1_2_IRQHandler(void){
 …
      Read ADC->DR
      N sampling done
      ADC_Done = 1;
 }
 …    Make N sampling
}
```

ADC Triggered by a Timer with DMA



DMA copies ADC DR register to memory after each conversion.

```
volatile uint DMA_Done = 0;

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

Make N sampling