

Microcontroller Engineering

TMIK13

Lecture 9

ADC

ANDREAS AXELSSON (ANDREAS.AXELSSON@JU.SE)

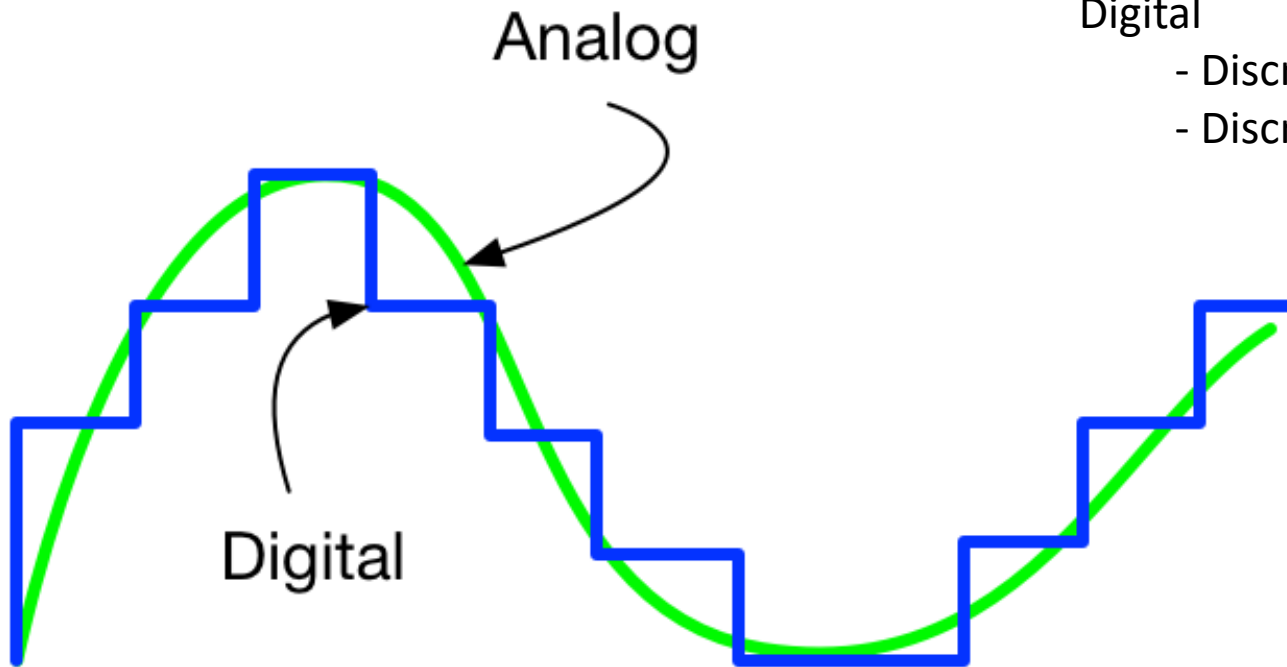
Analog/Digital Signals

Analog

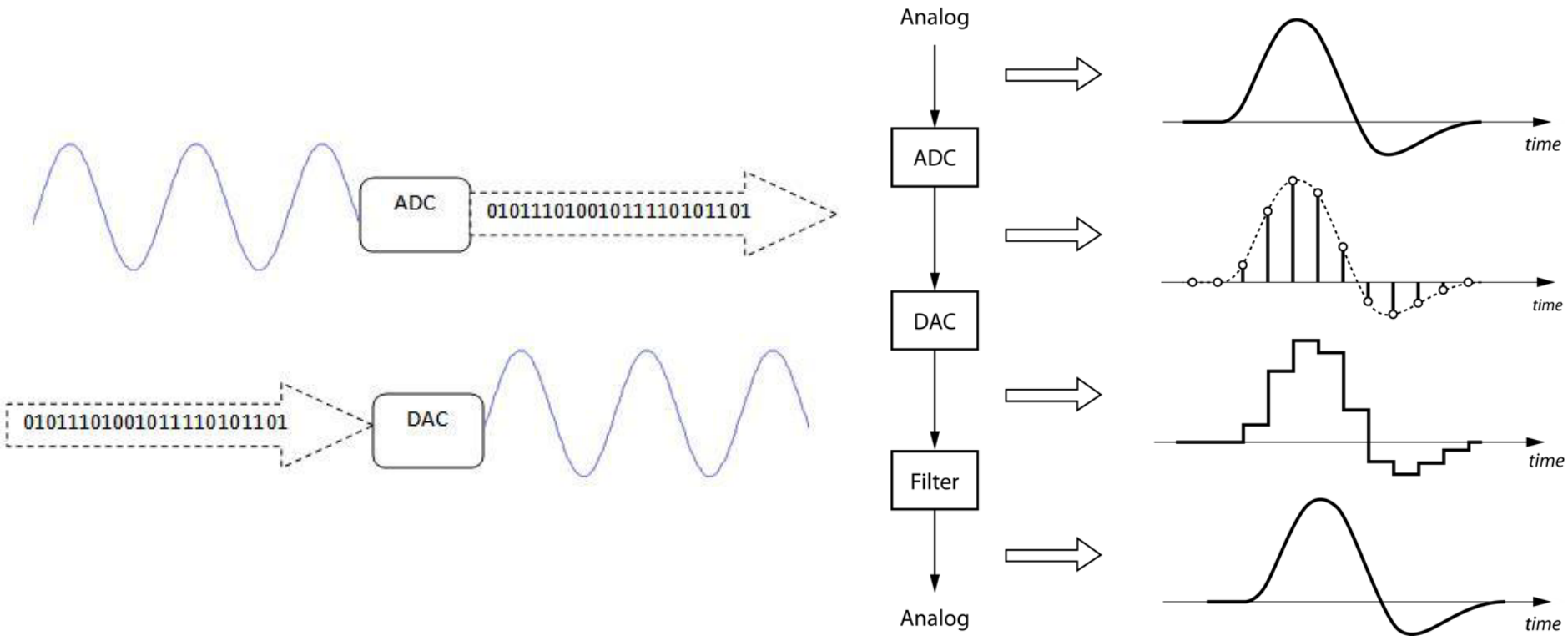
- Continuous in Time
- Continuous in Amplitude

Digital

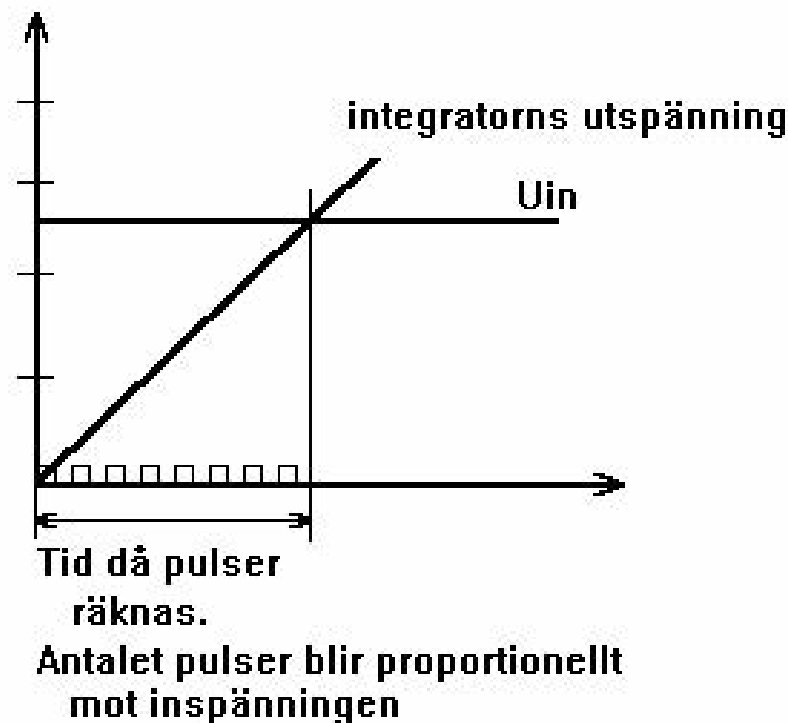
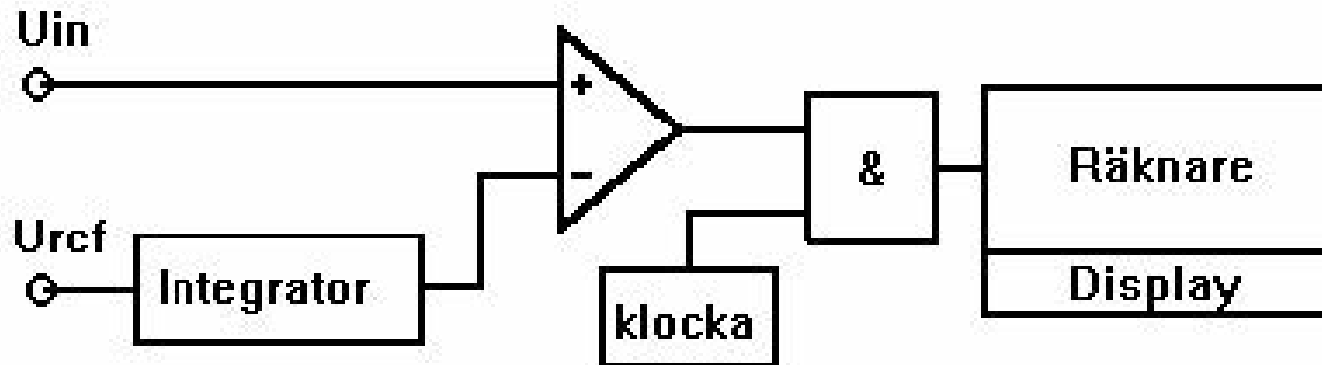
- Discrete in Time
- Discrete in Amplitude



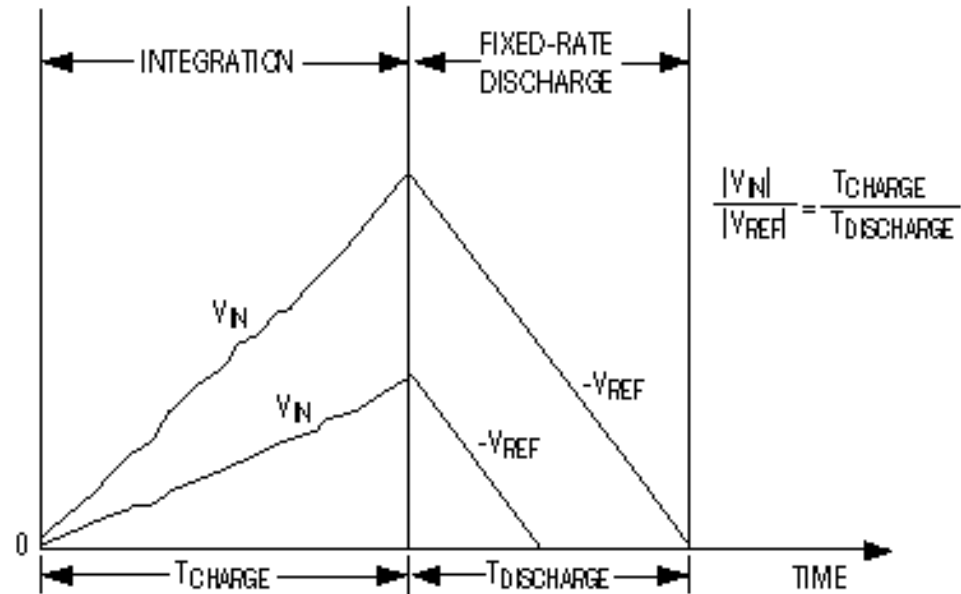
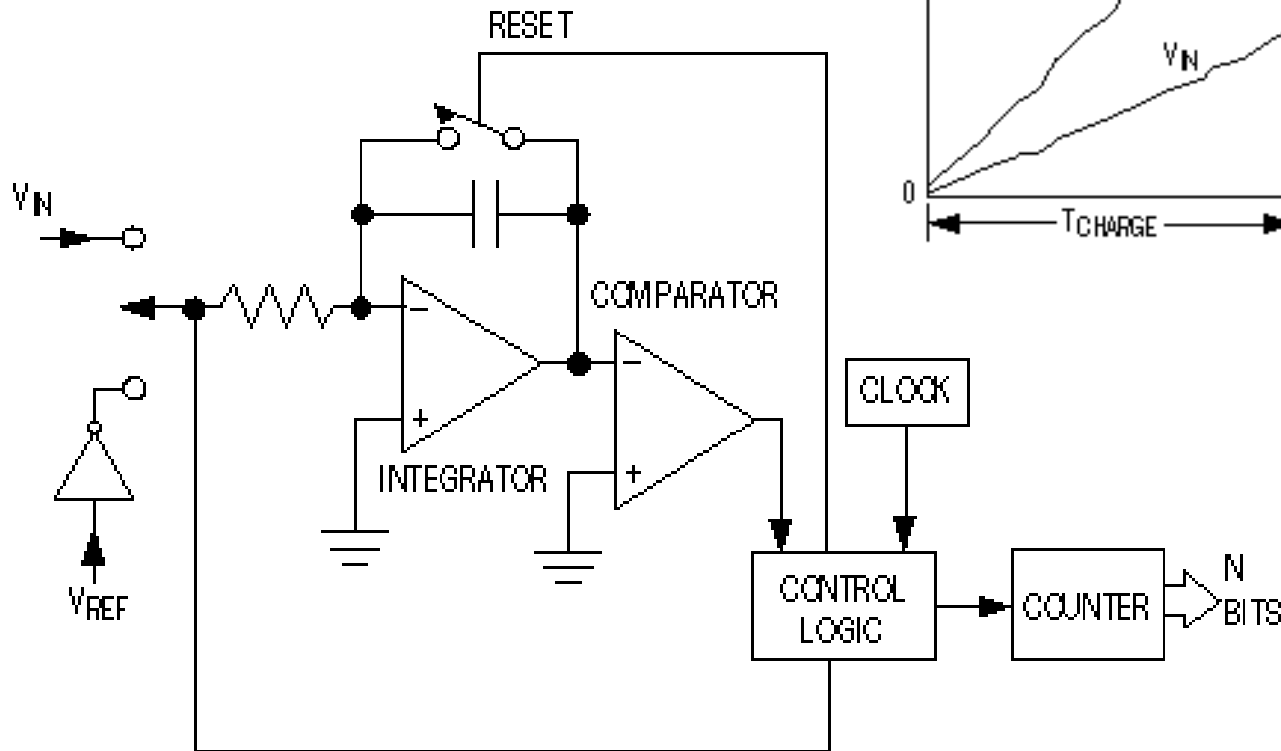
Analog/Digital Signals



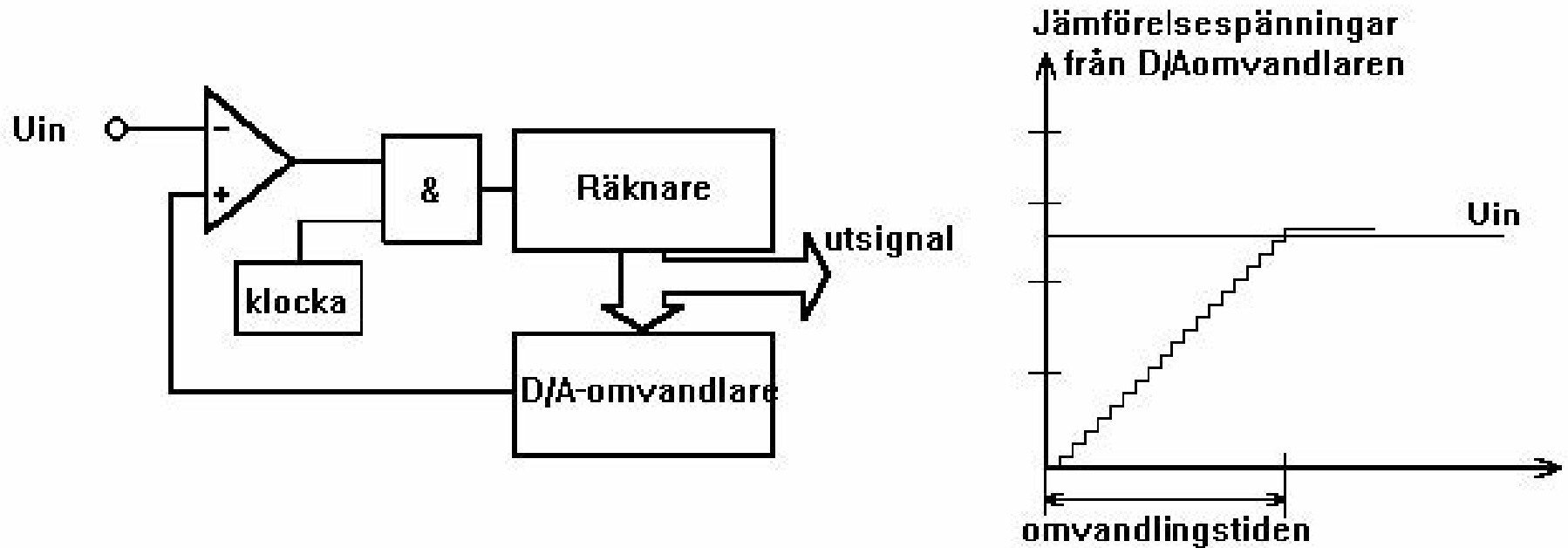
ADC – Integrating Converter



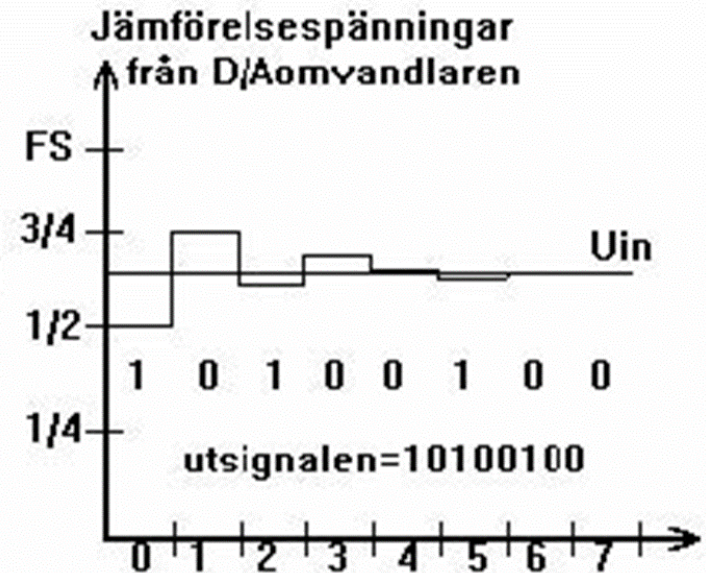
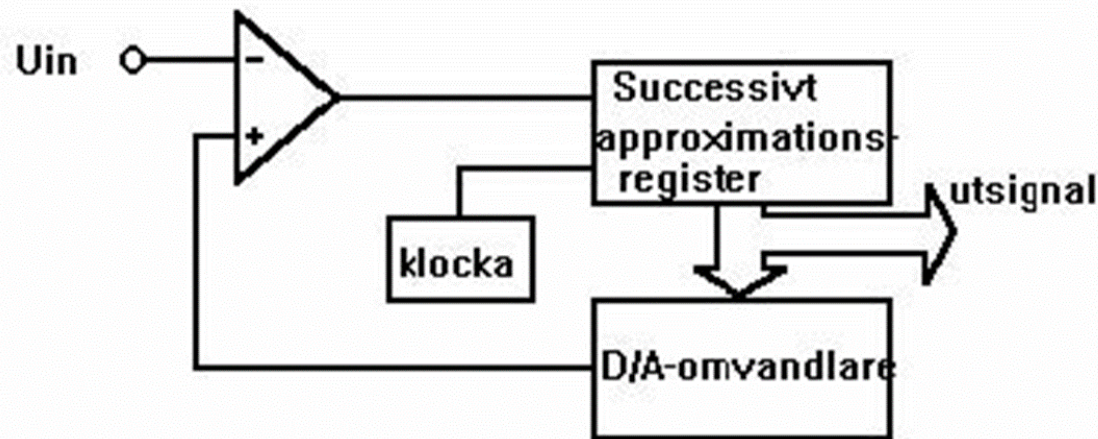
ADC – Integrating Converter



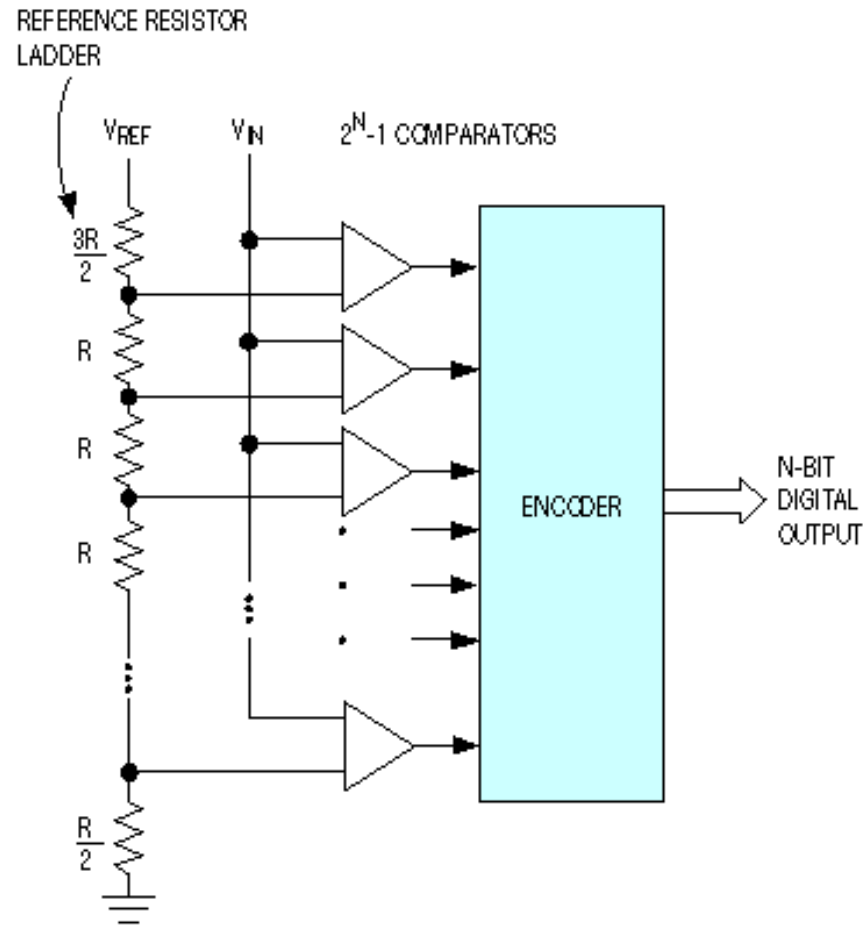
ADC – Digital Ramp Converter



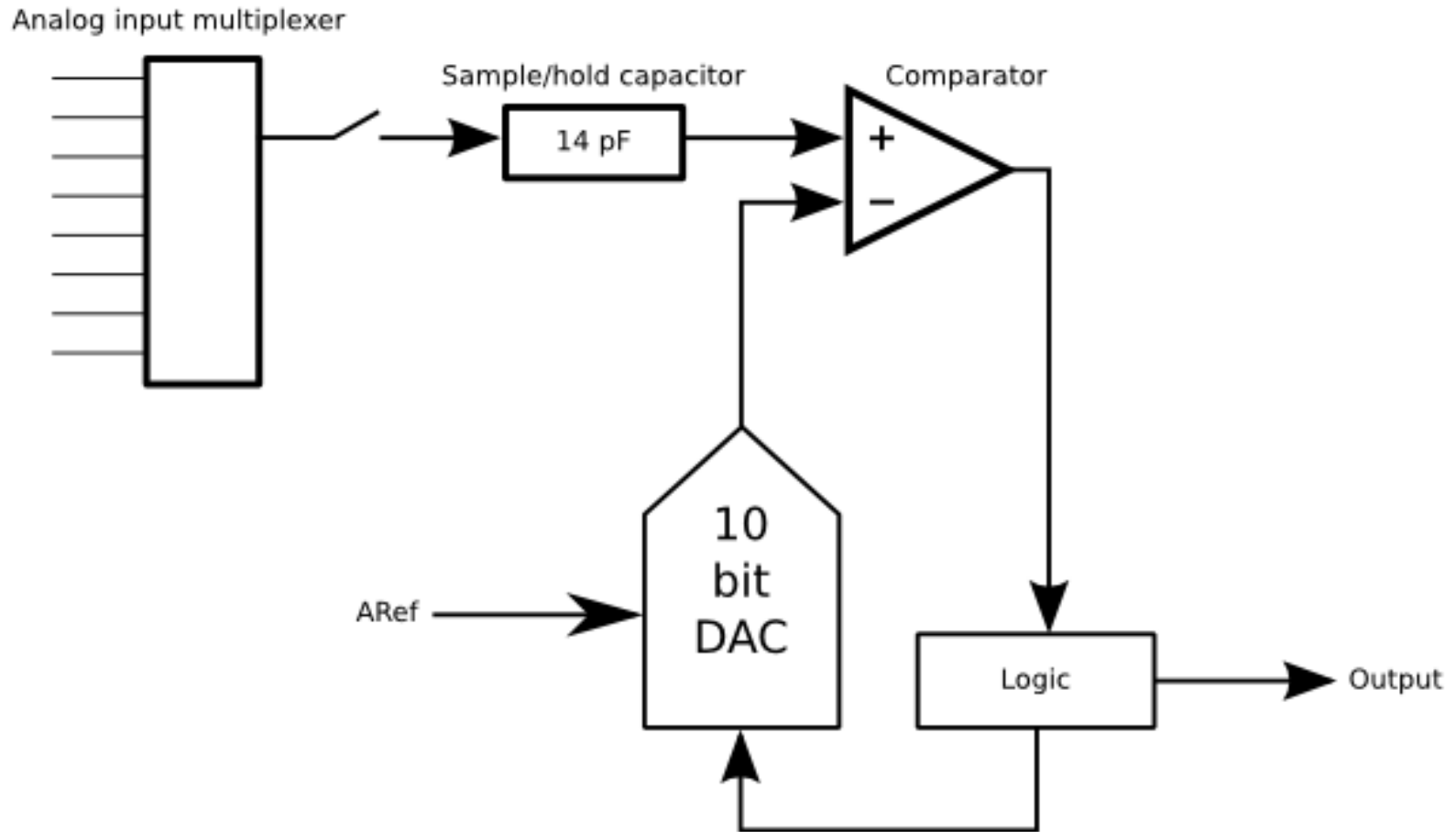
ADC – Successive Approximation



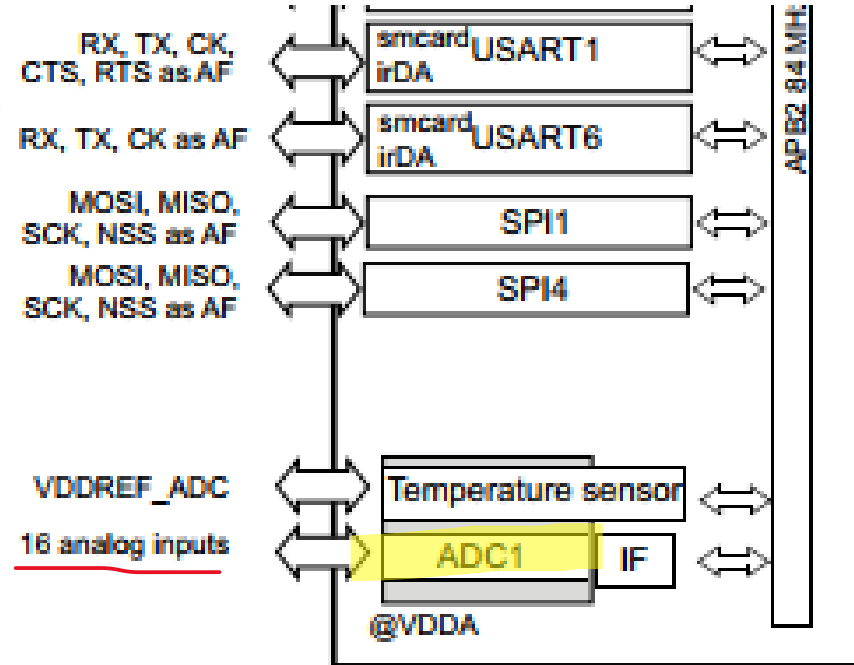
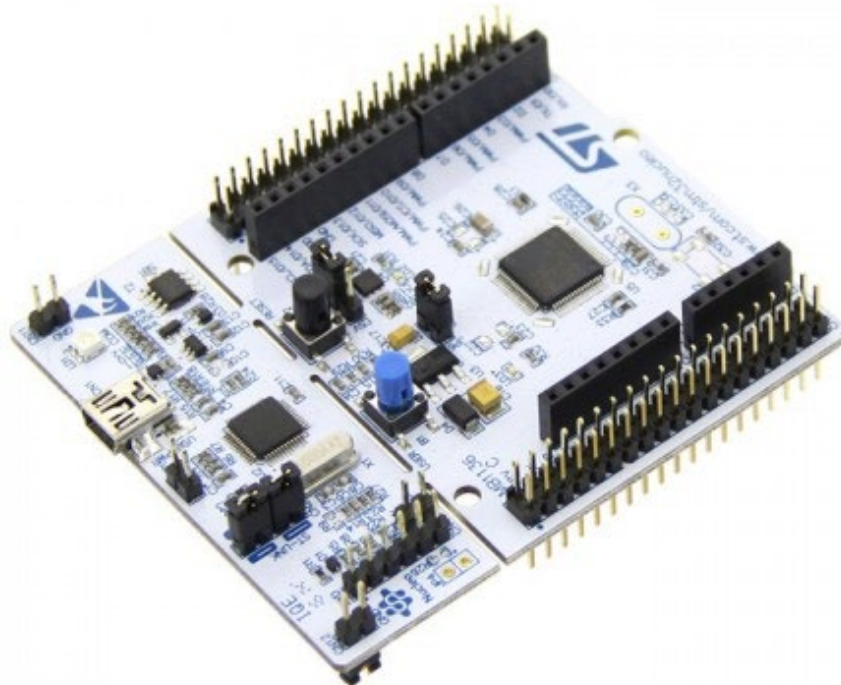
ADC – Parallel Converter (Flash)



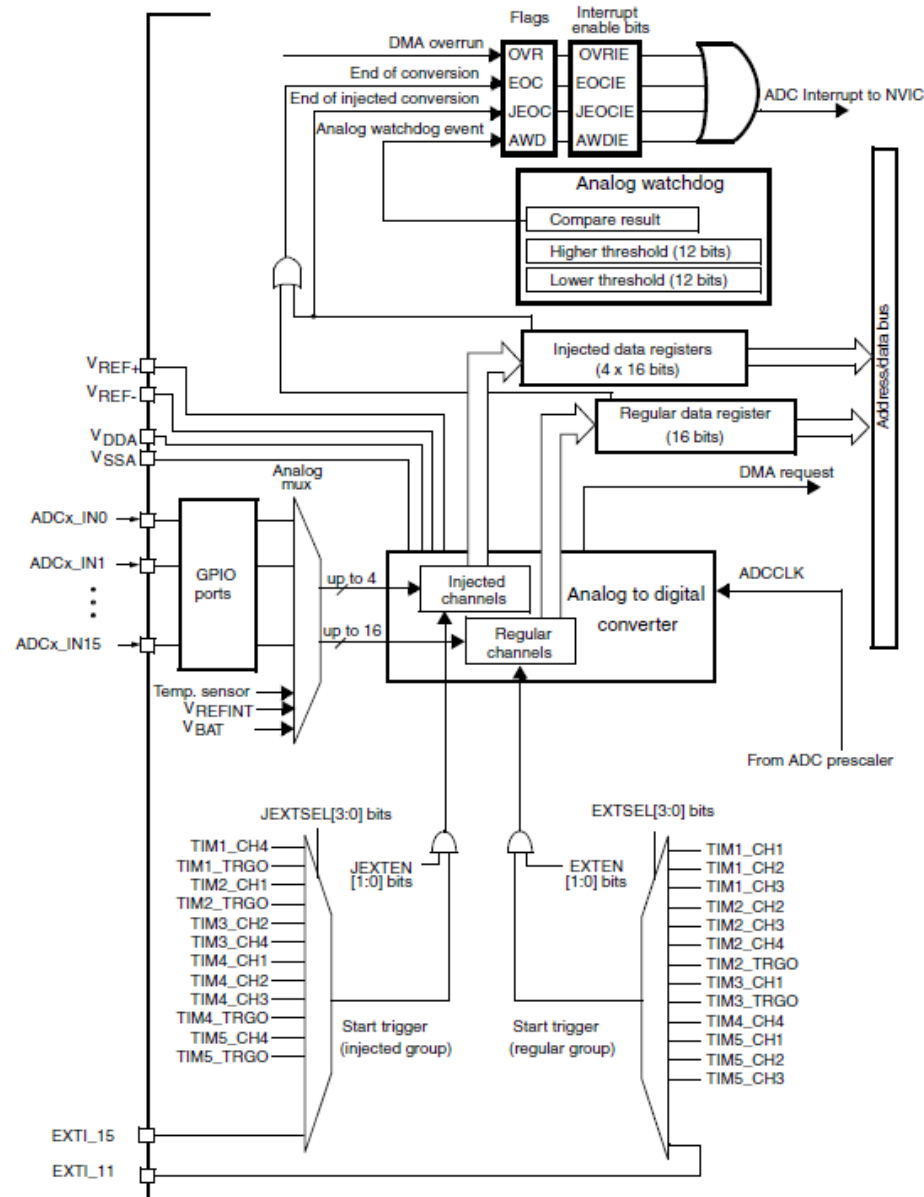
ADC – Analog Multiplexer



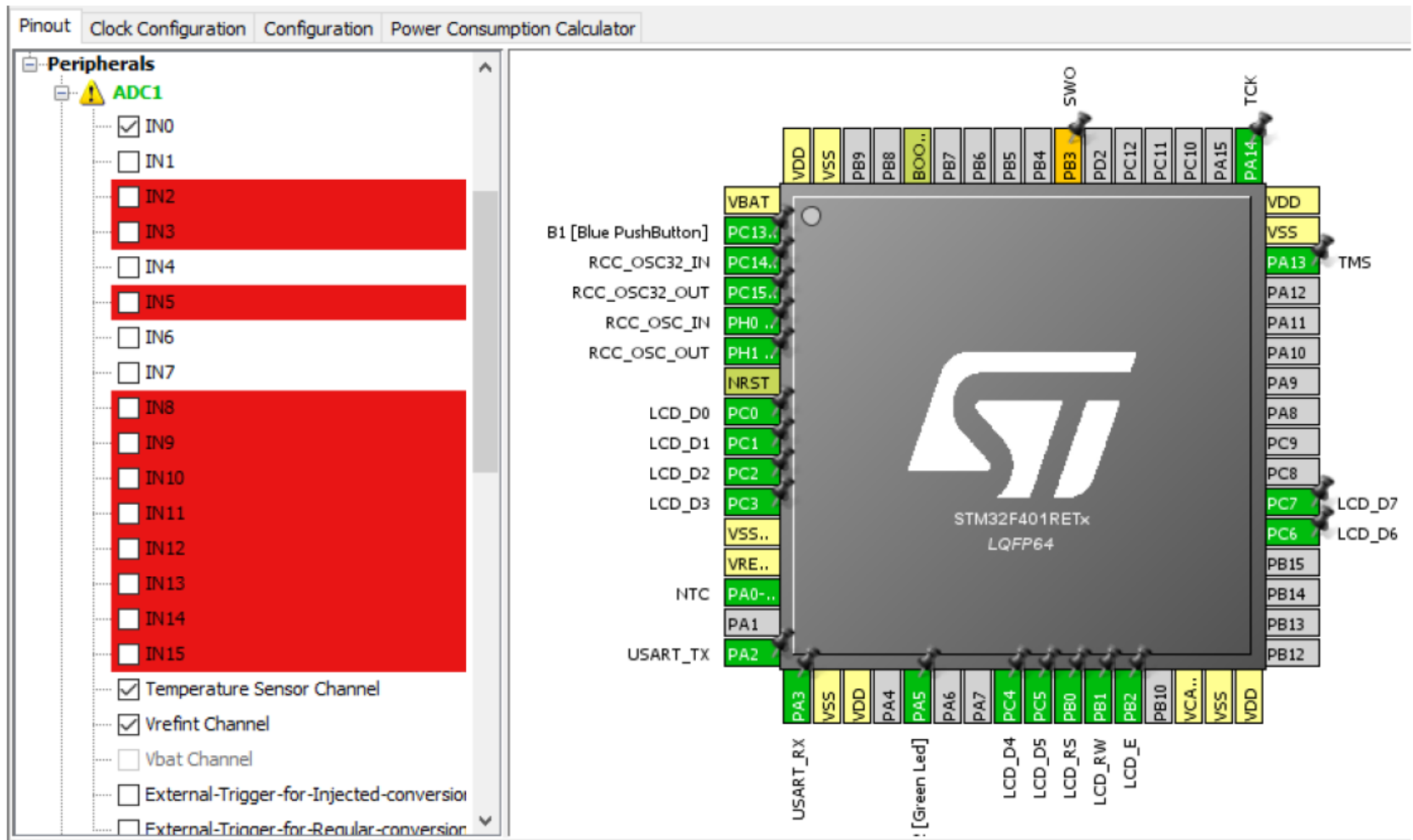
Nucleo-64 STM32F401RE



ADC – STM32F401RE



STM32CubeMX



STM32CubeMX

ADC1 Configuration

✓ Parameter Settings

✓ User Constants

✓ NVIC Settings

✓ DMA Settings

✓ GPIO Settings

Configure the below parameters :

Search :

ADC_Settings

Clock Prescaler	PCLK2 divided by 4
Resolution	12 bits (15 ADC Clock cycles)
Data Alignment	Right alignment
Scan Conversion Mode	Enabled
Continuous Conversion Mode	Enabled
Discontinuous Conversion Mode	Disabled
DMA Continuous Requests	Disabled
End Of Conversion Selection	EOC flag at the end of single channel conversion

ADC_Regular_ConversionMode

Number Of Conversion	3
External Trigger Conversion Source	Regular Conversion launched by software
External Trigger Conversion Edge	None

Rank

Channel	Channel Vrefint
Sampling Time	480 Cycles

Rank

Channel	Channel Temperature Sensor
Sampling Time	480 Cycles

Rank

Channel	Channel 0
Sampling Time	480 Cycles

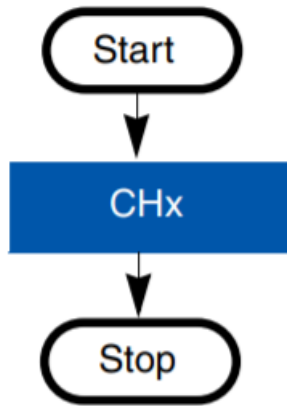
ADC_Injected_ConversionMode

Number Of Conversions	0
-----------------------	---

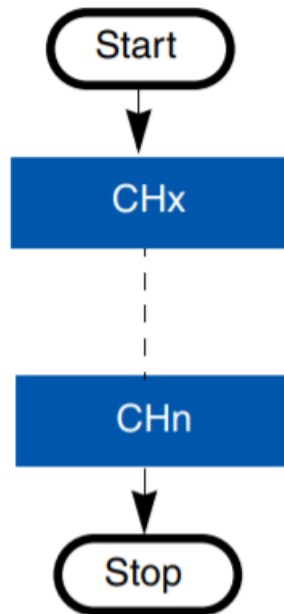
WatchDog

Enable Analog WatchDog Mode	<input type="checkbox"/>
-----------------------------	--------------------------

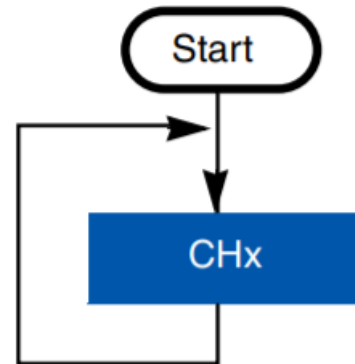
ADC – Conversion Modes



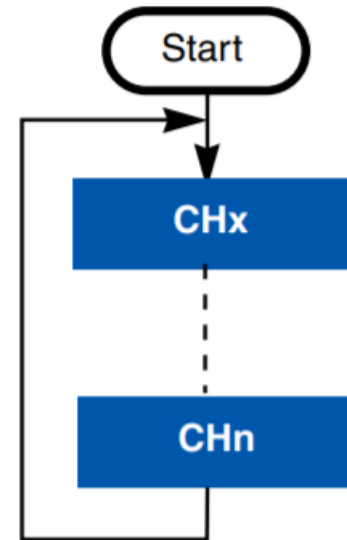
Single Channel
Single Conversion



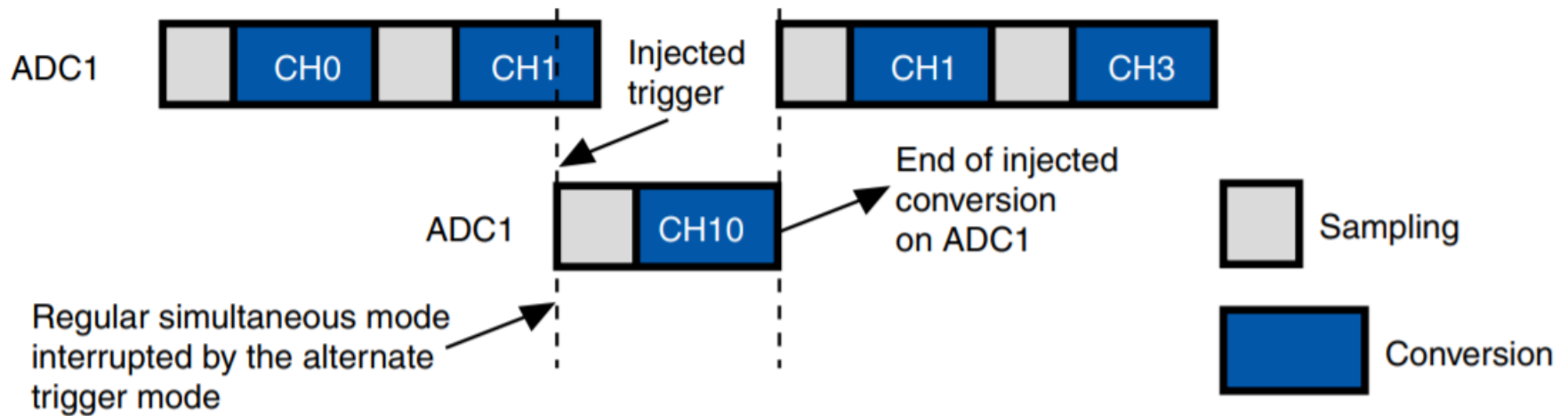
Multiple Channels
Single Conversion



Single Channel
Continuous
Conversion



Multiple Channels
Continuous
Conversion



Injected conversion has higher priority and interrupts the regular conversion if needed

ADC – VRef and Temperature

$$V_{DDA} = 3.3 \text{ V} * \frac{V_{REFINT_CAL}}{V_{REFINT_DATA}}$$

$$\text{Temperature in degC} = \frac{110 \text{ degC} - 30 \text{ degC}}{TS_CAL2 - TS_CAL1} * (TS_DATA - TS_CAL1) + 30 \text{ degC}$$

ADC – VRef and Temperature Calib

Table 73. Temperature sensor calibration values

Symbol	Parameter	Memory address
TS_CAL1	TS ADC raw data acquired at temperature of 30 °C, $V_{DDA} = 3.3 \text{ V}$	0x1FFF 7A2C - 0x1FFF 7A2D
TS_CAL2	TS ADC raw data acquired at temperature of 110 °C, $V_{DDA} = 3.3 \text{ V}$	0x1FFF 7A2E - 0x1FFF 7A2F

Table 76. Internal reference voltage calibration values

Symbol	Parameter	Memory address
V_{REFIN_CAL}	Raw data acquired at temperature of 30 °C $V_{DDA} = 3.3 \text{ V}$	0x1FFF 7A2A - 0x1FFF 7A2B

```
#define VREFIN_CAL *((uint16_t*)((uint32_t) 0x1FFF7A2A))
#define TS_CAL30 *((uint16_t*)((uint32_t) 0x1FFF7A2C))
#define TS_CAL110 *((uint16_t*)((uint32_t) 0x1FFF7A2E))
```

ADC – HAL Driver Usage

Polling mode IO operation

- Start the ADC peripheral using HAL_ADC_Start()
- Wait for end of conversion using HAL_ADC_PollForConversion(), at this stage user can specify the value of timeout according to his end application
- To read the ADC converted values, use the HAL_ADC_GetValue() function.
- Stop the ADC peripheral using HAL_ADC_Stop()

Interrupt mode IO operation

- Start the ADC peripheral using HAL_ADC_Start_IT()
- Use HAL_ADC_IRQHandler() called under ADC_IRQHandler() Interrupt subroutine
- At ADC end of conversion HAL_ADC_ConvCpltCallback() function is executed and user can add his own code by customization of function pointer HAL_ADC_ConvCpltCallback
- In case of ADC Error, HAL_ADC_ErrorCallback() function is executed and user can add his own code by customization of function pointer HAL_ADC_ErrorCallback
- Stop the ADC peripheral using HAL_ADC_Stop_IT()

ADC – HAL Driver Usage

Polling mode IO operation

- Start the ADC peripheral using `HAL_ADC_Start()`
- Wait for end of conversion using `HAL_ADC_PollForConversion()`, at this stage user can specify the value of timeout according to his end application
- To read the ADC converted values, use the `HAL_ADC_GetValue()` function.
- Stop the ADC peripheral using `HAL_ADC_Stop()`

Interrupt mode IO operation

- Start the ADC peripheral using `HAL_ADC_Start_IT()`
- Use `HAL_ADC_IRQHandler()` called under `ADC_IRQHandler()` Interrupt subroutine
- At ADC end of conversion `HAL_ADC_ConvCpltCallback()` function is executed and user can add his own code by customization of function pointer `HAL_ADC_ConvCpltCallback`
- In case of ADC Error, `HAL_ADC_ErrorCallback()` function is executed and user can add his own code by customization of function pointer `HAL_ADC_ErrorCallback`
- Stop the ADC peripheral using `HAL_ADC_Stop_IT()`

ADC – HAL Driver Usage

Polling mode IO operation

- Start the ADC peripheral using `HAL_ADC_Start()`
- Wait for end of conversion using `HAL_ADC_PollForConversion()`, at this stage user can specify the value of timeout according to his end application
- To read the ADC converted values, use the `HAL_ADC_GetValue()` function.
- Stop the ADC peripheral using `HAL_ADC_Stop()`

Interrupt mode IO operation

- Start the ADC peripheral using `HAL_ADC_Start_IT()`
- Use `HAL_ADC_IRQHandler()` called under `ADC_IRQHandler()` Interrupt subroutine
- At ADC end of conversion `HAL_ADC_ConvCpltCallback()` function is executed and user can add his own code by customization of function pointer `HAL_ADC_ConvCpltCallback`
- In case of ADC Error, `HAL_ADC_ErrorCallback()` function is executed and user can add his own code by customization of function pointer `HAL_ADC_ErrorCallback`
- Stop the ADC peripheral using `HAL_ADC_Stop_IT()`

ADC – HAL Driver Usage

Polling mode IO operation

- Start the ADC peripheral using HAL_ADC_Start()
- Wait for end of conversion using HAL_ADC_PollForConversion(), at this stage user can specify the value of timeout according to his end application
- To read the ADC converted values, use the HAL_ADC_GetValue() function.
- Stop the ADC peripheral using HAL_ADC_Stop()

Interrupt mode IO operation

- Start the ADC peripheral using HAL_ADC_Start_IT()
- Use HAL_ADC_IRQHandler() called under ADC_IRQHandler() Interrupt subroutine
- At ADC end of conversion HAL_ADC_ConvCpltCallback() function is executed and user can add his own code by customization of function pointer HAL_ADC_ConvCpltCallback
- In case of ADC Error, HAL_ADC_ErrorCallback() function is executed and user can add his own code by customization of function pointer HAL_ADC_ErrorCallback
- Stop the ADC peripheral using HAL_ADC_Stop_IT()

```
/* Check End of conversion flag */  
__HAL_ADC_GET_FLAG(&hadc1, ADC_FLAG_EOC);
```

ADC – STM32CubeIDE Demo

Lets go!!!

Microcontroller Engineering

Questions?

Contact information

Andreas Axelsson

Email: andreas.axelsson@ju.se

Mobile: 0709-467760