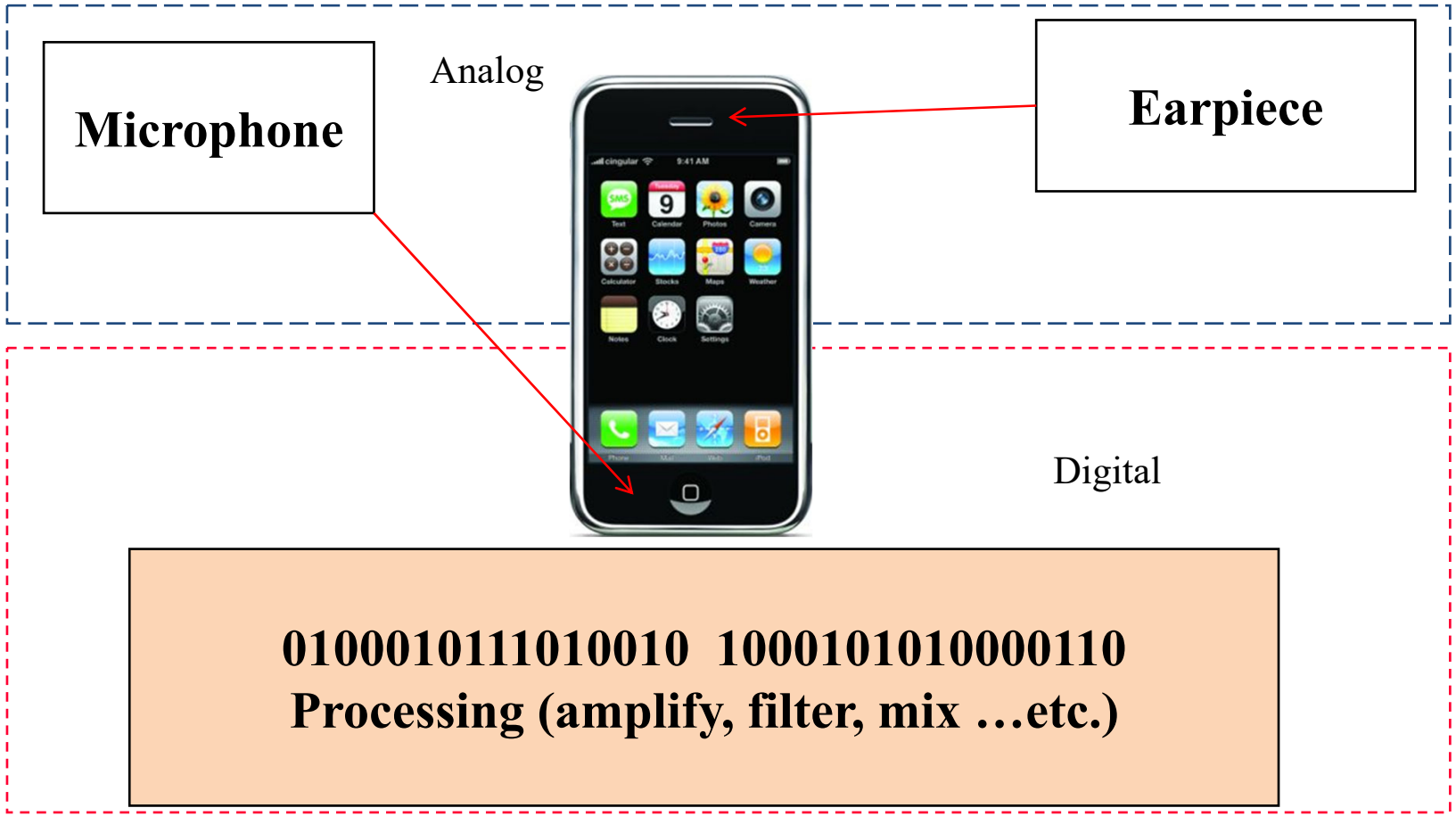
ADC / DAC

Analog-to-digital converter
Digital-to-analog converter

Computer Engineering 2

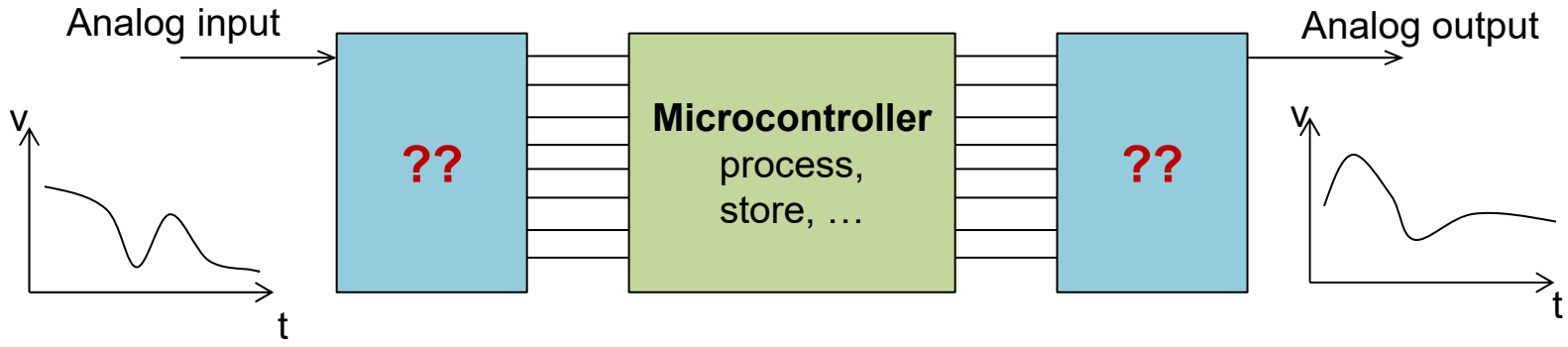
Motivation

- **Analog ←→ digital conversion needed in many applications**



Motivation

- **Modern information processing done in digital domain**
  - Easier to process, store, make copies, … etc.
- **But, the real world is analog (not digital)**
  - Signals are continuous and not discrete
    - Pressure sensor that continuously delivers a voltage
    - The music you hear
- **Need for devices to convert analog to digital (and vice-versa)**



Agenda

School of Engineering
InES Institute of Embedded Systems

- **ADC and DAC**
  - What it is
  - How it works
  - Characteristics
  - Types of error
- **ADCs on STM32F429**
  - Features and functionality
  - Programming the ADC
- **DACs on STM32F429 (optional)**
  - Features and registers
  - DAC configuration example
- **Conclusions**

Learning Objectives

At the end of this lesson, you will be able

- To explain what an ADC/DAC is and what it is used for
- To describe how a (simple) Flash ADC/DAC works
- To name some application examples of ADC/DAC
- To name and explain some important characteristics/error sources
  - Sampling rate, voltage reference, offset and gain error …
- To name basic features of ADC in the STM32F429
- To set up and use simple features of ADC in STM32F429
- To use the device documentation
  - To understand features and functionality
  - Interpret simple parameters (e.g. energy consumption, sampling rate, gain/offset error)
  - To derive simple configuration and control of ADC using the CPU
  - To find out, understand and use other features of the ADC

Learning Objectives

School of Engineering
InES Institute of Embedded Systems

- **Modern microcontrollers have lots of features**
  - Impossible to discuss all of them in a lesson
  - Important that you learn to interpret the information in the documentation
  - In this lesson you will do that, under the guidance of your lecturer
  - Lecture focuses on application of ADCs in microcontrollers and not on ADC design

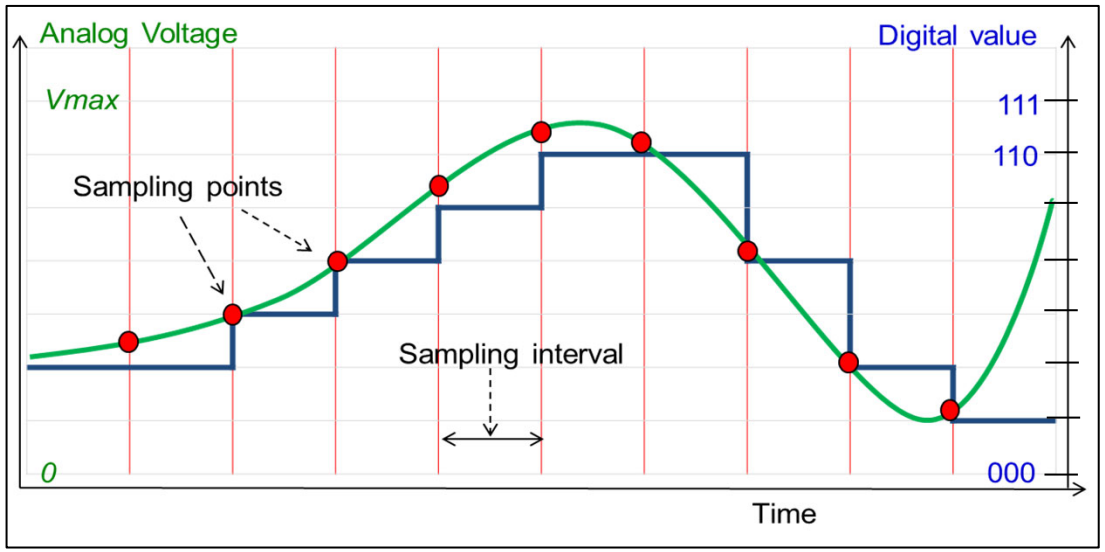
- **Advanced modes STM32F4**
  - Advanced features of ADC/DAC are not described in this class
    - e.g. injected mode, dual/triple modes, … etc.
  - Interested? See application notes and datasheets (references)

ADC: What it is

- **ADC – Analog to Digital Converter**
  - Converts input signal (voltage) to a digital value (N-bit)
  - Conversion results in one of 2^N possible numerical levels
  - Raw input signal can be dynamic[1)] or static[2)]
    - Dynamic signal (green) sampled at specific time intervals
    - Samples transformed into series of discrete values (blue)

Example

3-bit ADC
- 8 possible levels (000 – 111)
- Each conversion corresponds to one out of 8 levels



1) changing over time

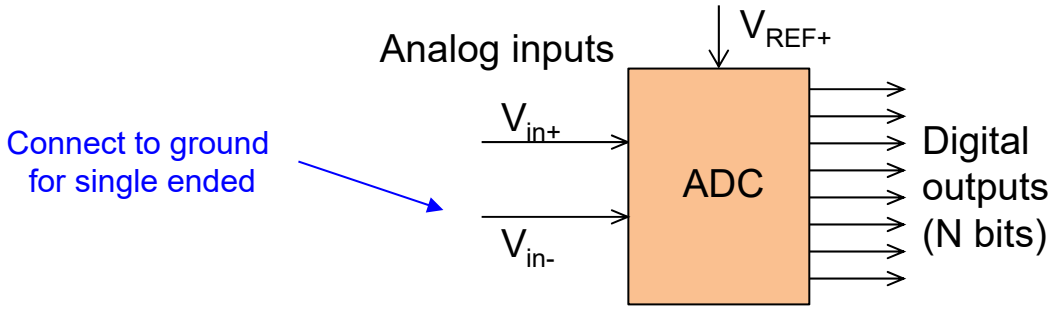
2) time-invariant

ADC: What it is

- **Input signals**
  - Differential inputs
  - V_{in+} signal to convert (non-inverting input)
  - V_{in-} signal to convert (inverting input)

$$V_{in} = V_{in+} - V_{in-}$$

- **Single ended mode**
  - Only V_{in+} used
  - V_{in-} is grounded



- **Reference voltage V_{REF+}**
  - Internal or external stable voltage
  - Needed to weight input voltage

$$V_{in} = (\textit{digital value}) * V_{REF+} \; / \; (2^N)$$

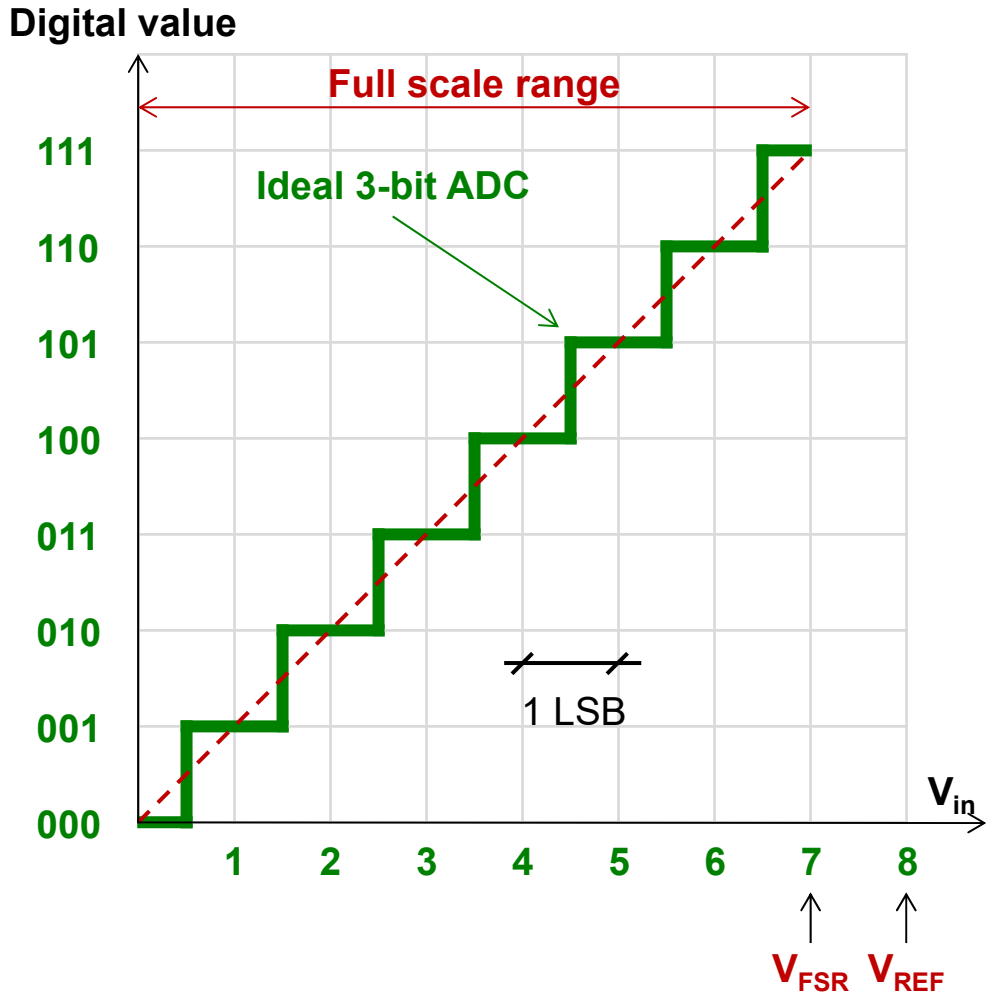
We will concentrate on single ended

ADC: What it is

- **Resolution**
  - Number of bits N
  - Size of digital word
- **LSB[1]**
  - 1 LSB ≙ $V_{REF} / (2^N)$
- **Full Scale Range (FSR)**
  - Range between analog levels of minimum and maximum digital codes
  - **V_{FSR}** is one LSB less than **V_{REF}**

Example

V_{REF} = 8 V, N = 3 bits	→ 1 LSB = 8 V / 8 = 1 V
	→ FSR from 0 V to 7 V



1) Least Significant Bit

ADC: How it works

- **Example Flash ADC**
  - Network of 2^N resistors to divide V_{REF} into 2^N levels
  - 2^N - 1 analog comparators
    - Compare input signal to divided reference voltages
  - Encoder transforms digital comparator results into N-bit word

3-bit ADC

Example: V_{REF} = 8V, V_{in} = 2.3V

V_{REF} / 16 = 0.5 V

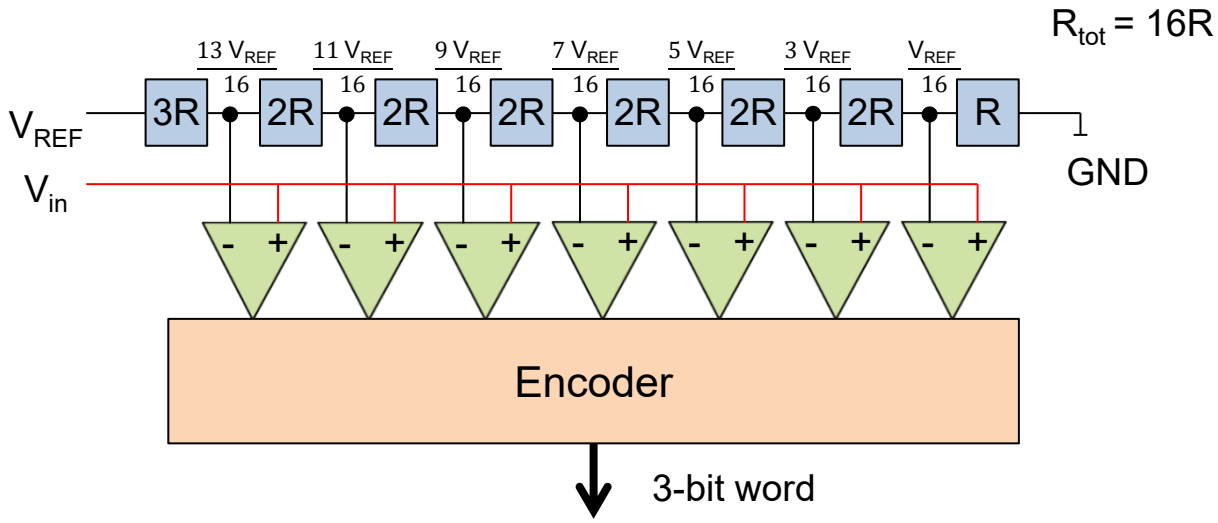
3 V_{REF} / 16 = 1.5 V

5 V_{REF} / 16 = 2.5 V

….

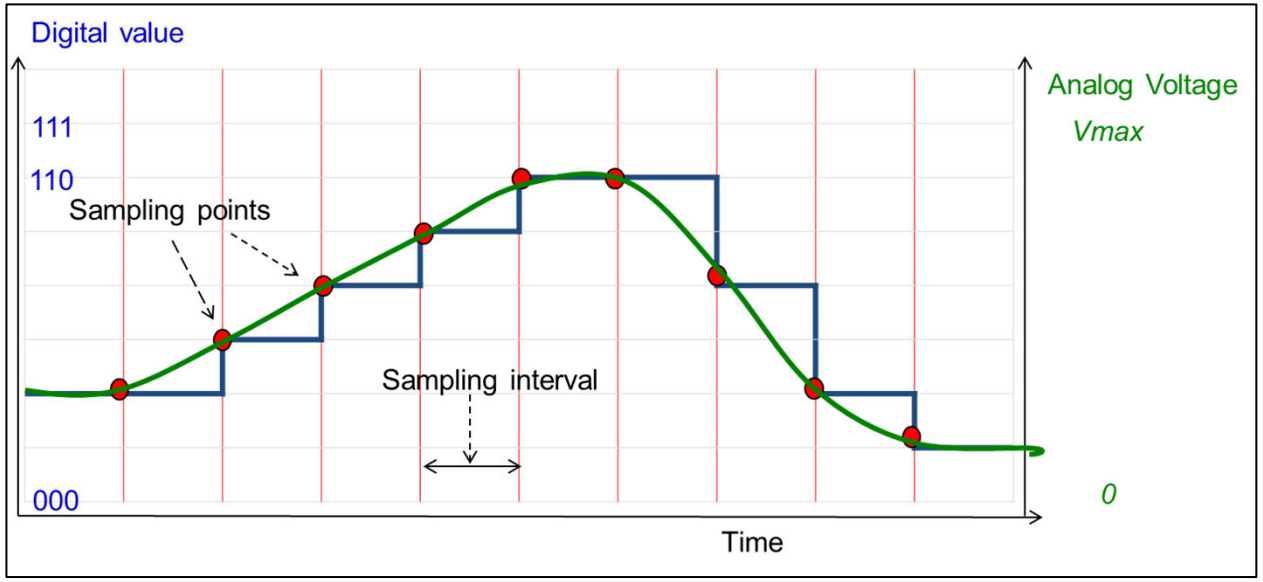
Comparator stream = 0000011

3-bit output word = 010



DAC: What it is

- **DAC – Digital to Analog Converter**
  - Converts N-bit digital input to analog voltage level
  - E.g. music from your MP3 player is read and converted back to sound
    - A series of different values in the digital domain leads to a series of steps in the analog domain. The result is a dynamic output signal
    - “Play-back” time depends on time between conversions (sampling interval)

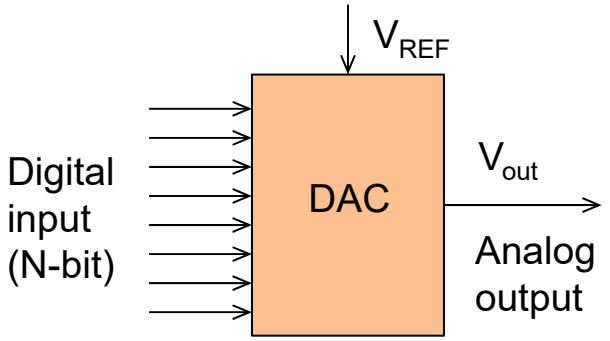


DAC: What it is

- **Reference voltage V_{REF}**
  - Accurate reference voltage (from internal or external source)
  - Needed to relate digital value to a voltage

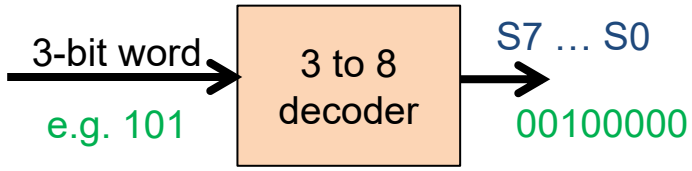
- **Output signal V_{out}**
  - Analog output
    - Unipolar (only positive)
    - Bipolar (positive or negative)
  - Conversion yields approximation of digital signal

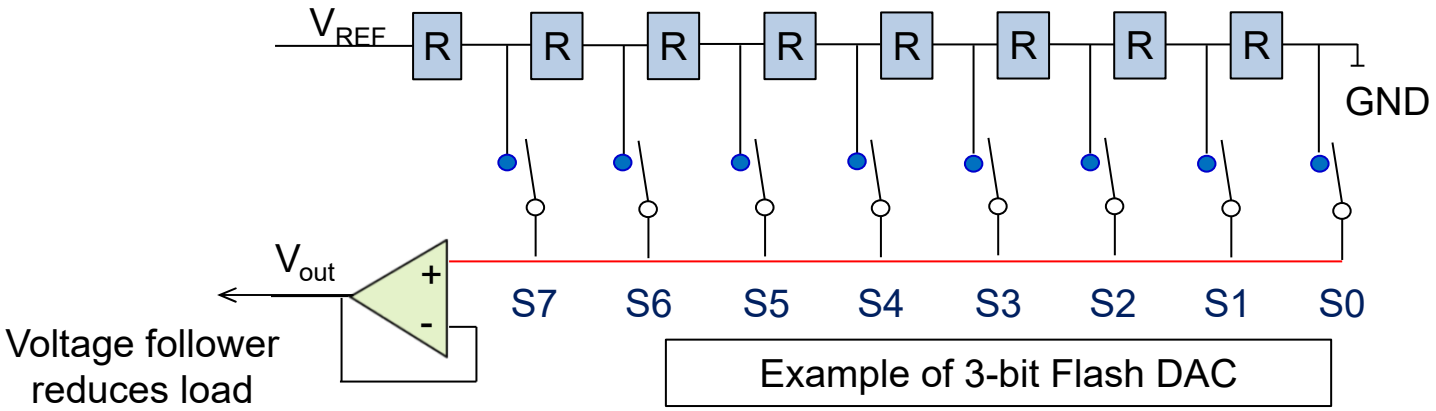
$$V_{out} = (\textit{digital value}) * V_{REF} \; / \; (2^N)$$



DAC: How it works

- **Example Flash DAC**
  - Network of resistors (of same value) creates 2^N voltage levels
  - N-bit digital input decoded into 2^N values (S0 … Sx)
    - Select single voltage level as DAC output



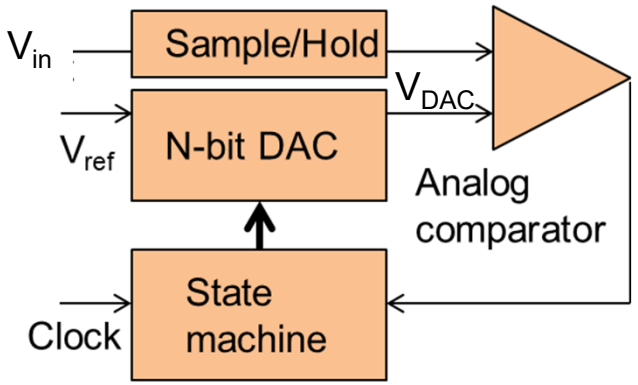


ADC re-visited: How it works

School of Engineering
InES Institute of Embedded Systems

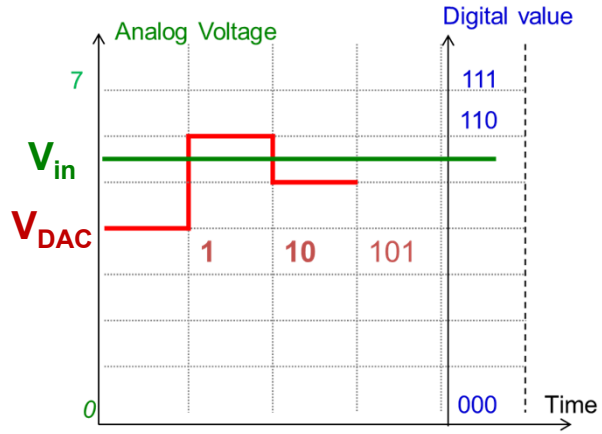
- **Successive Approximation Register (SAR) ADC**
  - Approach V_{in} with successive division by 2
    - Binary search
  - Start with half the digital value
    - MSB = 1, all other bits at 0
  - DAC generates analog value V_{DAC} that is compared to V_{in}
    - If $V_{DAC} < V_{in}$ → keep MSB at 1, otherwise set MSB to 0
  - Continue with other N bits in same way (N steps)

SAR ADC as an alternative to the previously introduced Flash ADC



1. Higher than 100 → 1
2. Lower than 110 → 10
3. Higher than 101 → 101

3-bit result = 101



ADC Characteristics

- **Flash ADC**
  - Fast conversion
  - Requires many elements
    - e.g. 255 comparators for 8-bit resolution
    - Power hungry
    - Consumes large chip area
- **SAR ADC**
  - Used on most microcontrollers
  - Good trade-off between speed, power and cost
    - Up to 5 Msps
    - Resolution from 8 to 16 bits

ADC Characteristics

School of Engineering
InES Institute of Embedded Systems

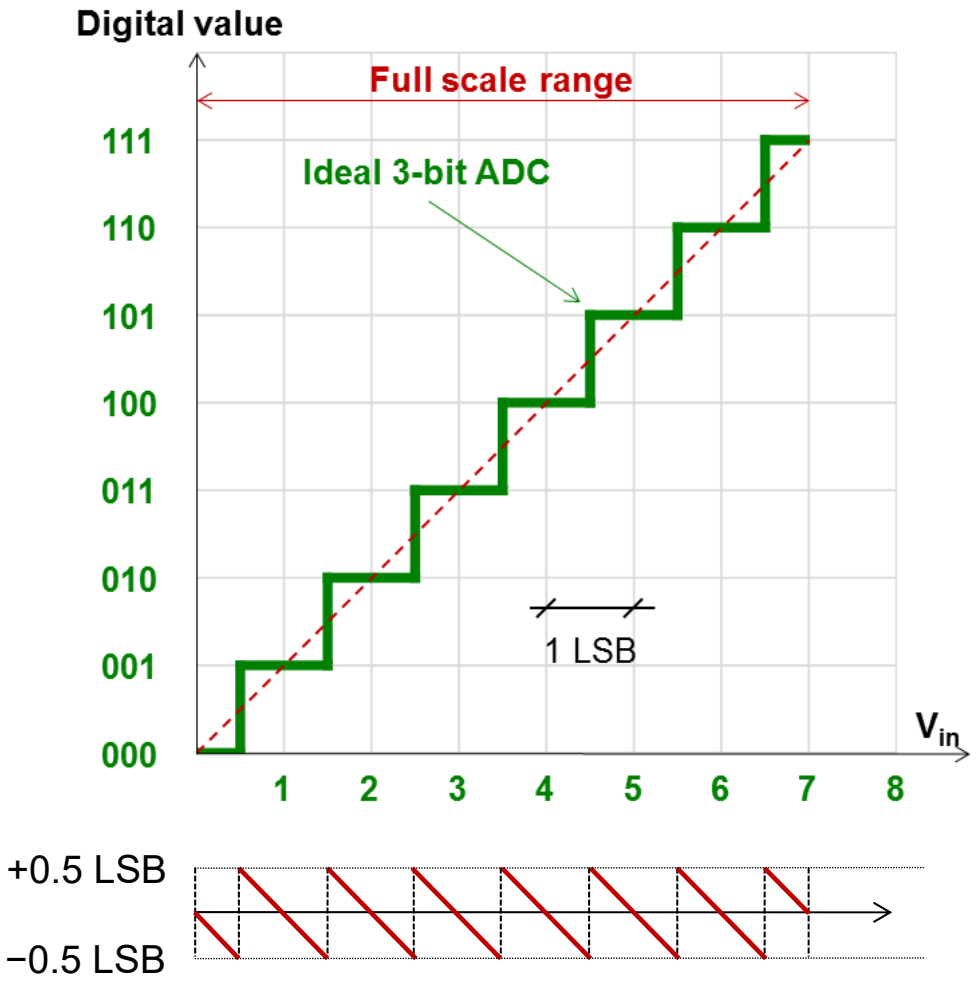
- **Sampling rate**
  - Input signal sampled at discrete points in time → discontinuities
  - Should be at least twice the highest frequency component of input signal
    - Nyquist–Shannon sampling theorem
- **Conversion time**
  - Time between start of sampling and digital output available
  - Programming a higher resolution may increase conversion time
- **Monotonicity**
  - Increase of Vin results in increase or no change of digital output and vice-versa

ADC: Types of Error

- **Quantization error**
  - Analog input is continuous
    - Infinite number of states
  - Digital output is discrete
    - Finite number of states
  - Introduces an error between −0.5 LSB and +0.5 LSB

Quantization error can be reduced by reducing LSB, e.g. either by increasing number of bits (resolution) or by reducing V_{REF}

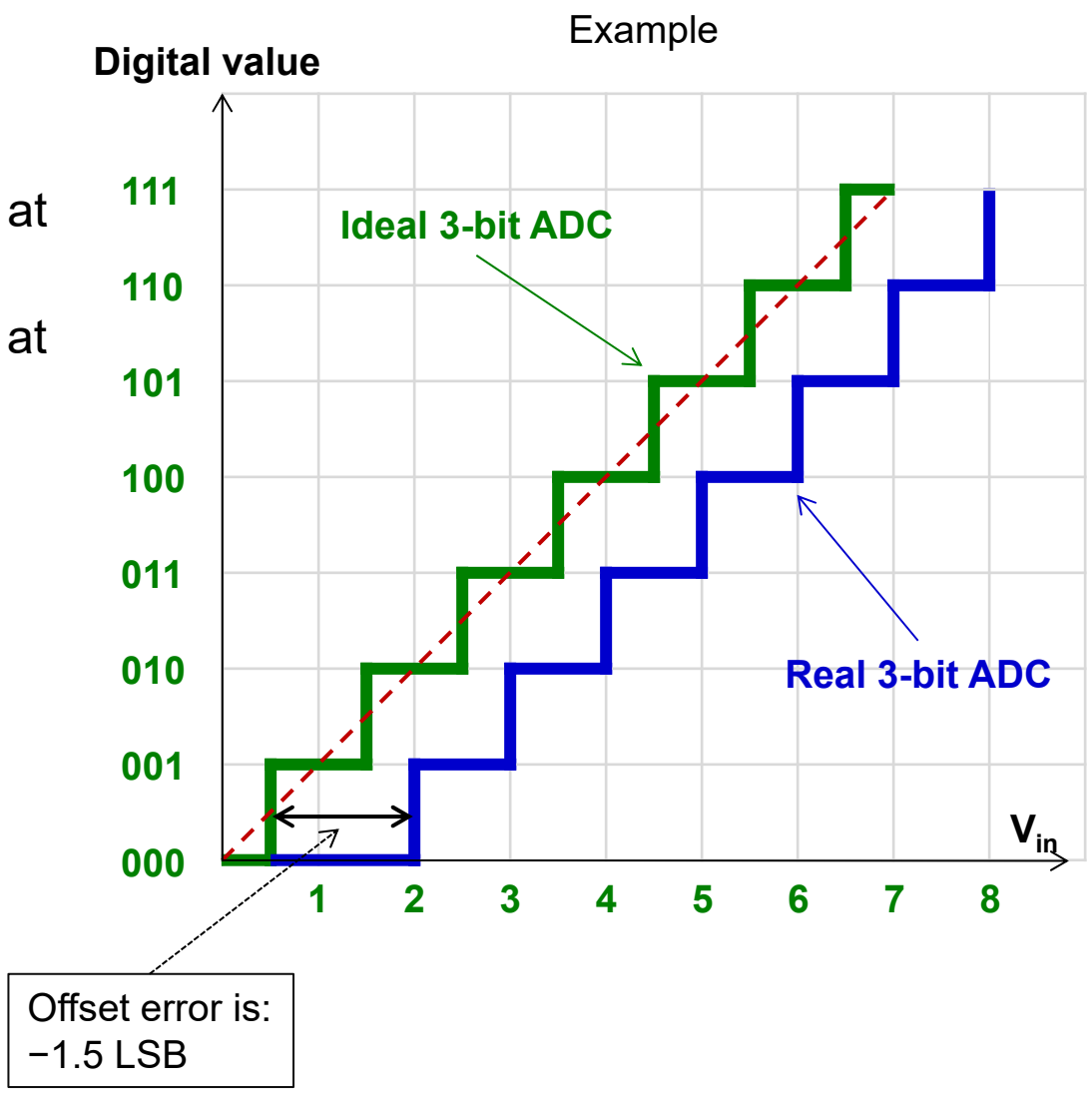
Reducing V_{REF} also reduces full scale range



ADC: Types of Error

- **Offset error**
  - Also called zero-scale error
  - Deviation of real N-bit ADC from ideal N-bit ADC at input point zero
  - For an ideal N-bit ADC, the first transition occurs at 0.5 LSB above zero
  - Can be corrected using the microcontroller

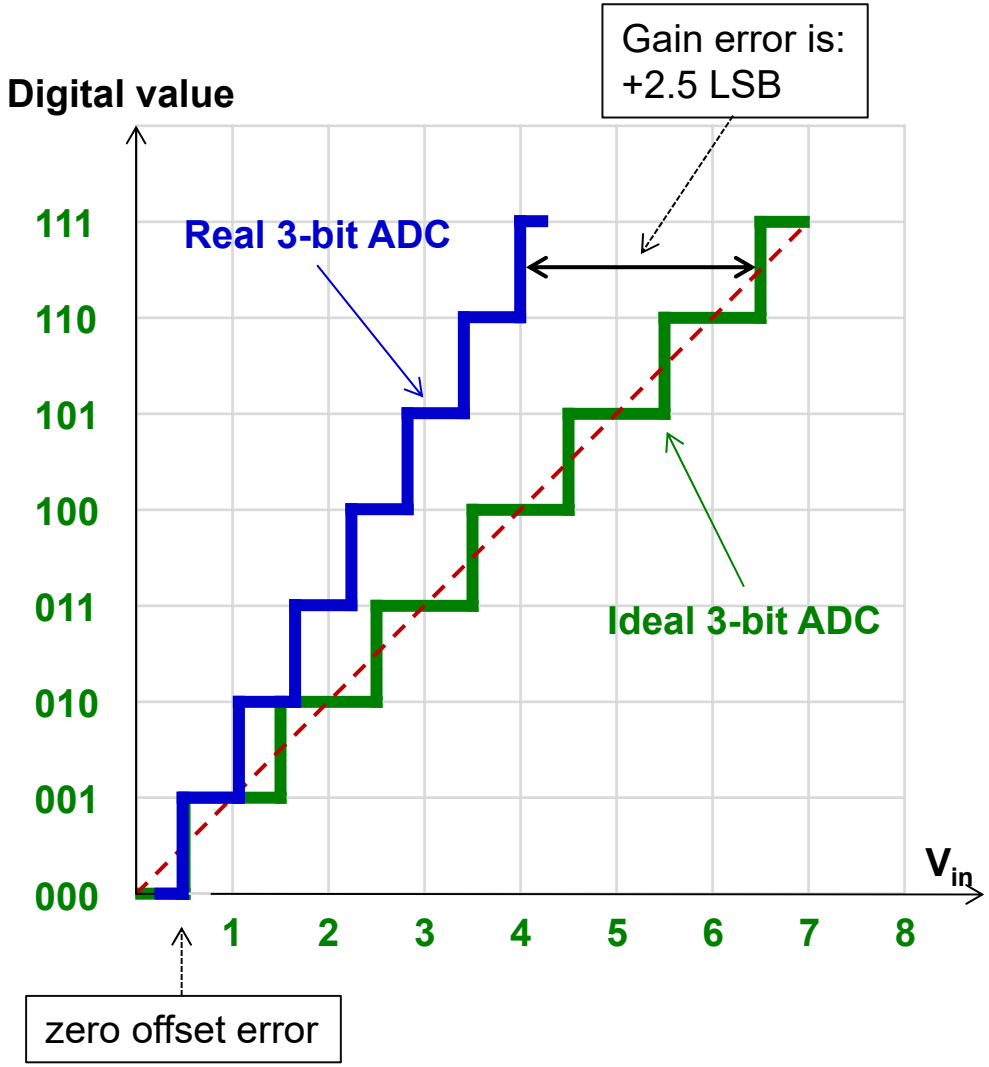
Measuring the offset error:
Zero-scale voltage is applied to analog input and is increased until first transition occurs



ADC: Types of Error

- **Gain error**
  - Indicates how well the slope of an actual transfer function matches the slope of the ideal transfer function
  - Expressed in LSB or as a percent of full-scale range (%FSR)
  - Calibration with hardware or software possible

full-scale error = offset error + gain error



Features and Functionality

ADCs on STM32F429

ADC Exercise: Reading the Datasheet

School of Engineering
InES Institute of Embedded Systems

- **Q1: How many ADCs are there in the STM32F429?**
- **Q2: How many sources for each ADC?**
  - What is meant by internal/external source?
  - How many external sources?
  - How many internal sources? What are they?
  - What is temperature monitoring?
- **Q3: How many bits can the result of a conversion have?**
- **Q4: How is the result of the conversion stored?**
  - Can it be overwritten? What are the consequences?
  - What is meant by left aligned/right aligned?
- **Q5: How can the result of conversion be transferred to memory?**
- **Q6: To which ADC can pins PA5/PA6 be connected?**
- **Q7: What is meant by single/continuous conversion vs single channel/scan mode?**
- **Q8: How does the CPU know that the conversion is done?**

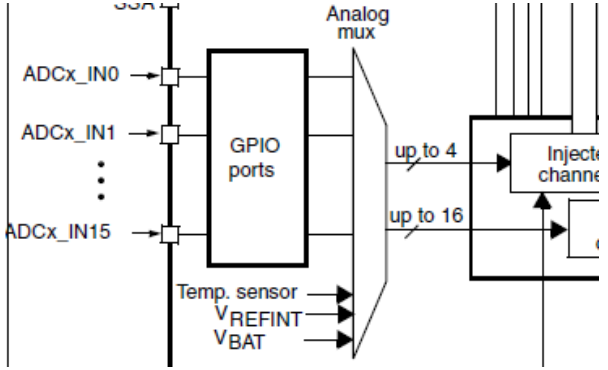
solution

ADC Exercise: Reading the Datasheet

School of Engineering
InES Institute of Embedded Systems

ADC features

- 3 ADCs **(Q1)**
  - SAR (Successive Approximation Register)
  -
- 19 channels **(Q2)**
  - 16 external sources (from outside STM32)
  - Internal sources: V_{REF}, V_{BAT}
  - Temperature sensor internally connected to same input channel as V_{BAT} (ADC1_IN18). Measurement of chip temperature. Not very accurate



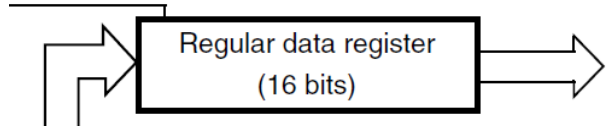
- Each ADC configurable as 12/10/8/6 - bit **(Q3)**
  - N-bit SAR ADC generally requires N clocks to complete a conversion.

Note: It is unclear where the number of 24 channels in the data sheet comes from.

solution

ADC Exercise: Reading the Datasheet

School of Engineering
InES Institute of Embedded Systems



- **Result of conversion (Q4)**
  - Can be 6, 8, 10, 12 bits
  - Stored as 16-bit in regular data register ADC_DR
  - In some cases, result overwritten by next conversion (overrun)
    - Therefore, read result ON TIME
  - Right aligned: Result in 12 most right bits of 16-bit word
  - Left aligned: Result in 12 most left bits of 16-bit word
  - Same for 10/8–bit results (exception for 6-bit mode. See references)

- **Transfer of conversion result (Q5)**
  - After conversion, data can be read by the CPU
    - Alternatively use DMA (Direct Memory Access)

solution

ADC Exercise: Reading the Datasheet

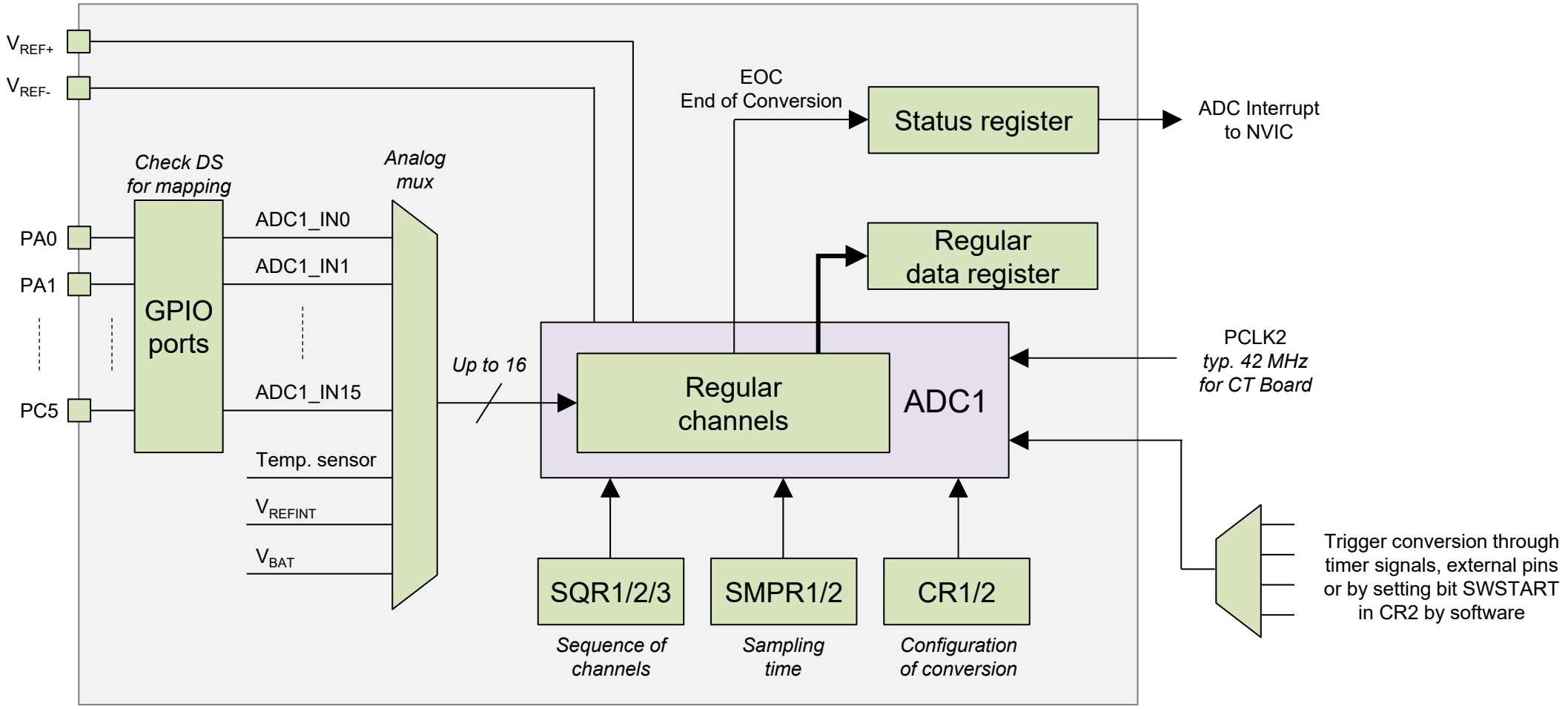
School of Engineering
InES Institute of Embedded Systems

- **Port pins that can be connected to ADC/DAC (Q6)**
  - From table: PA5 can be used with ADC1(in5) or ADC2(in5) or DAC2(out2)
  - PA6 can be an input6 for ADC1 or ADC2
  - Other possibilities can be worked out from the device datasheet

Table 10. STM32F427xx and STM32F429xx pin and ball definitions (continued)

Pin number							Pin name (function after reset)(1)	Pin type	I / O structure	Notes	Alternate functions	Additional functions
LQFP100	LQFP144	UFBGA176	LQFP176	WLCSP143	LQFP208	TFBGA216						
30	41	P4	51	M9	54	P4	PA5	I/O	TC	(4)	TIM2_CH1/TIM2_ETR, TIM8_CH1N, SPI1_SCK, OTG_HS_ULPI_CK, EVENTOUT	ADC12_IN5/ DAC_OUT2
31	42	P3	52	N10	55	P3	PA6	I/O	FT	(4)	TIM1_BKIN, TIM3_CH1, TIM8_BKIN, SPI1_MISO, TIM13_CH1, DCMI_PIXCLK, LCD_G2, EVENTOUT	ADC12_IN6

Simplified ADC Diagram



ADC Conversion Modes (Q7)

School of Engineering
InES Institute of Embedded Systems

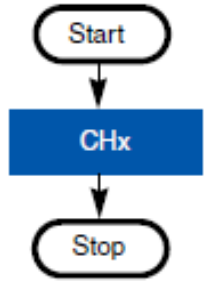
- **Single channel / multi-channel vs Single / continuous conversion**

	Single channel	Multi-channel (scan mode)
Single conversion	Convert 1 channel, then stop. This is the simplest mode.	Convert all channels in group, one after the other, then stop. The group of channels is in a sequence that can be programmed.
Continuous conversion	Continuously convert 1 channel until stop order is given. Minimal CPU intervention.	Continuously convert a group of several channels until stop order is given. The group of channels is in a sequence that can be programmed.

ADC Single Conversion Modes

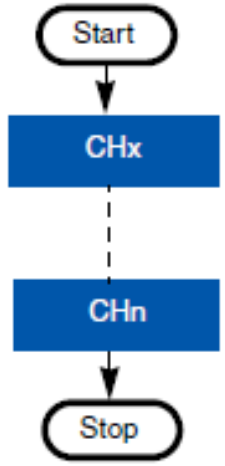
School of Engineering
InES Institute of Embedded Systems

- **Single channel single conversion**
  - Possible use
    - Sensor tied to ADC input
    - On a push button you read and display it



Channel programmed in SQR1/2/3

- **Multi-channel single conversion**
  - ADC sequencer used to set up order of conversions.
  - Possible use
    - Group of sensors need to be monitored
    - For each sensor there is an ADC input
    - Sensors are grouped and scanned one after the other every time a reading needs to be done

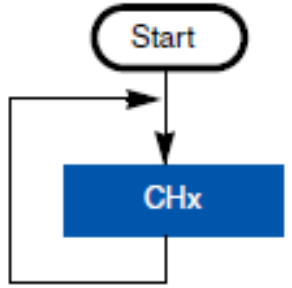


Sequence of channels programmed in SQR1/2/3

ADC Continuous Conversion Modes

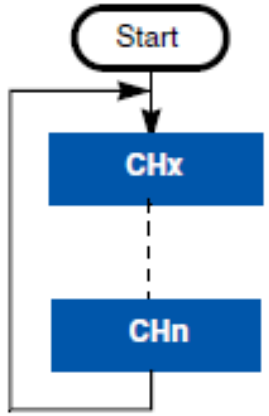
- **Usefulness**
  - Minimal CPU intervention for setups → ADC does not require CPU intervention to restart.
  - Beware overwriting previous results

- **Single channel continuous conversion**
  - Possible use: Continuously monitor a single sensor
    - Result can be compared in background to specific value and action taken only when that value reaches limit



Channel programmed in SQR1/2/3

- **Multi channel continuous conversion**
  - Possible use: Continuously monitor several sensors



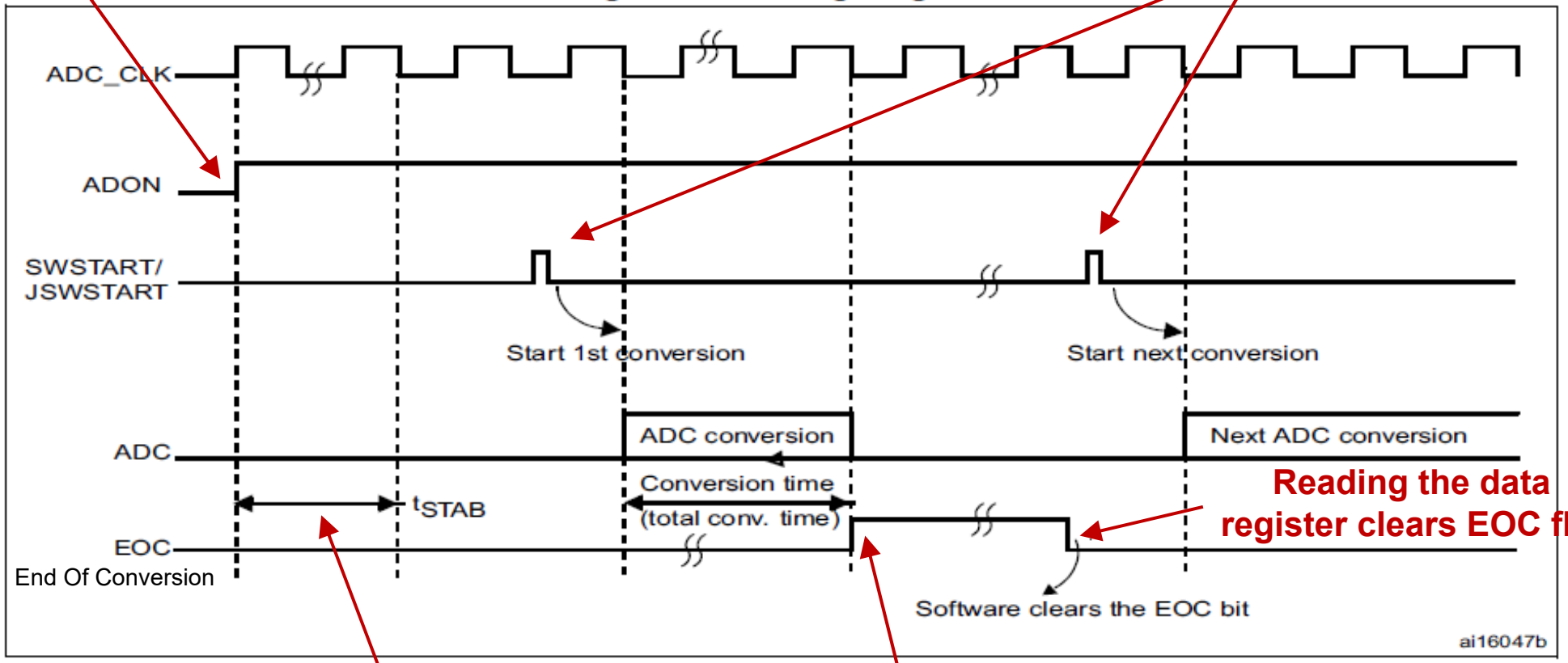
Sequence of channels programmed in SQR1/2/3

ADC Timing

ADC activation by software

Start of conversion by software or trigger signal

Figure 45. Timing diagram



Reading the data register clears EOC flag

ADC stabilization

EOC flag signals end of conversion

ADC Timing

- **Total conversion time**
  - Depends on time for sampling and conversion

$$T_{total} = T_{sample} + T_{conv}$$

  - T_{sample} individually programmable for each channel
    - Registers ADC_SMPR1 and ADC_SMPR2
    - Between 3 and 480 cycles
  - T_{conv} depends on resolution
    - 12 bits 12 ADCCLK cycles
    - 10 bits 10 ADCCLK cycles
    - 8 bits 8 ADCCLK cycles
    - 6 bits 6 ADCCLK cycles

Example

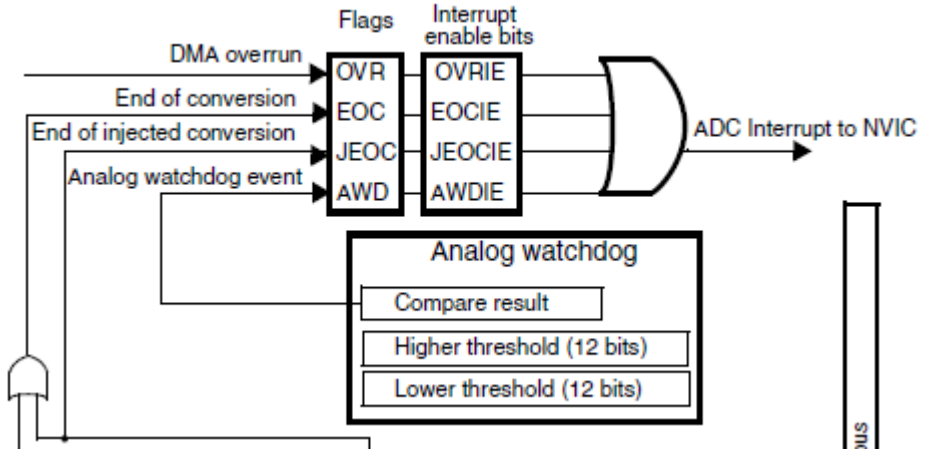
given: APB2 clock = 48 MHz
Prescaler 2 → ADCCLK = 24 MHz
3 cycles sampling time
12 bit resolution

$T_{total} = (3 + 12) * 1/ 24\ MHz = 0.625\ us$

sampling rate < $1 / T_{total}$ = 1.6 Msps

Analog Watchdog

- **Allows guarded monitoring of one or more channels with very little CPU overhead**
- **Converted value is compared to programmable guards (min and max)**
- **Flag (interrupt) if monitored signal outside limits**
- **Example**
  - potentiometer can be guarded so that an activity is started only if knob is in certain area



STM32F429 ADC: Characteristics

School of Engineering
InES Institute of Embedded Systems

- **Datasheet gives information about important parameters**
  - Current consumption: Useful for the system power budget
    - Goes up by 1.6 mA for each ADC that is active
  - Max. sampling rate
    - Useful for defining the max. speed of the input signal
    - Sampling at up to 2.4 Msps possible
  - Offset error (max = +-2.5 LSB)
  - Gain error (typ = +-1.5 LSB)

Table 78. ADC static accuracy at f_{ADC} = 30 MHz(1)

Symbol	Parameter	Test conditions	Typ	Max(2)	Unit
ET	Total unadjusted error	f_{ADC} = 30 MHz, R_{AIN} < 10 kΩ, V_{DDA} = 2.4 to 3.6 V, V_{REF} = 1.8 to 3.6 V $V_{DDA} - V_{REF}$ < 1.2 V	±2	±5	LSB
EO	Offset error		±1.5	±2.5	
EG	Gain error		±1.5	±3	
ED	Differential linearity error		±1	±2	
EL	Integral linearity error		±1.5	±3	

Operating power supply range	ADC operation
V_{DD} =1.8 to 2.1 V(3)	Conversion time up to 1.2 Msps
V_{DD} = 2.1 to 2.4 V	Conversion time up to 1.2 Msps
V_{DD} = 2.4 to 2.7 V	Conversion time up to 2.4 Msps
V_{DD} = 2.7 to 3.6 V(5)	Conversion time up to 2.4 Msps

STM32F429

Programming the ADC

Programing the ADC

School of Engineering
InES Institute of Embedded Systems

- **ADC registers**
  - Some registers are common for all ADCs (common registers)
  - Each ADC also has specific registers
    - Structure of specific registers similar for all 3 ADCs

ADC region		Offset		Address of register
0x4001 2000 - 0x4001 23FF	ADC1	0x000 - 0x04C	Specific registers	0x4001 2000 + 0x000 + register offset
			Reserved	
	ADC2	0x100 - 0x14C	Specific registers	0x4001 2000 + 0x100 + register offset
			Reserved	
	ADC3	0x200 - 0x24C	Specific registers	0x4001 2000 + 0x200 + register offset
			Reserved	
	Common	0x300 - 0x308	Common registers	0x4001 2000 + 0x300 + register offset

Programing the ADC

- **Common registers**

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	ADC_CSR	Reserved										OVR	STRT	JSTRT	JEOC	EOC	AWD	Reserved		OVR	STRT	JSTRT	JEOC	EOC	AWD	Reserved		OVR	STRT	JSTRT	JEOC	EOC	AWD
	Reset value											0	0	0	0	0	0			0	0	0	0	0	0			0	0	0	0	0	0
												ADC3								ADC2								ADC1					
0x04	ADC_CCR	Reserved								TSVREFE	VBATE	Reserved				ADCPRE[1:0]		DMA[1:0]		DDS	Reserved	DELAY [3:0]				Reserved		MULTI [4:0]					
	Reset value									0	0					0	0	0	0	0		0	0	0	0			0	0	0	0	0	
0x04	ADC_CCR	Reserved								TSVREFE	VBATE	Reserved				ADCPRE[1:0]		DMA[1:0]		DDS	Reserved	DELAY [3:0]				Reserved							
	Reset value									0	0					0	0	0	0	0		0	0	0	0								
0x08	ADC_CDR	Regular DATA2[15:0]																Regular DATA1[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_CCR ADC Common Control Register

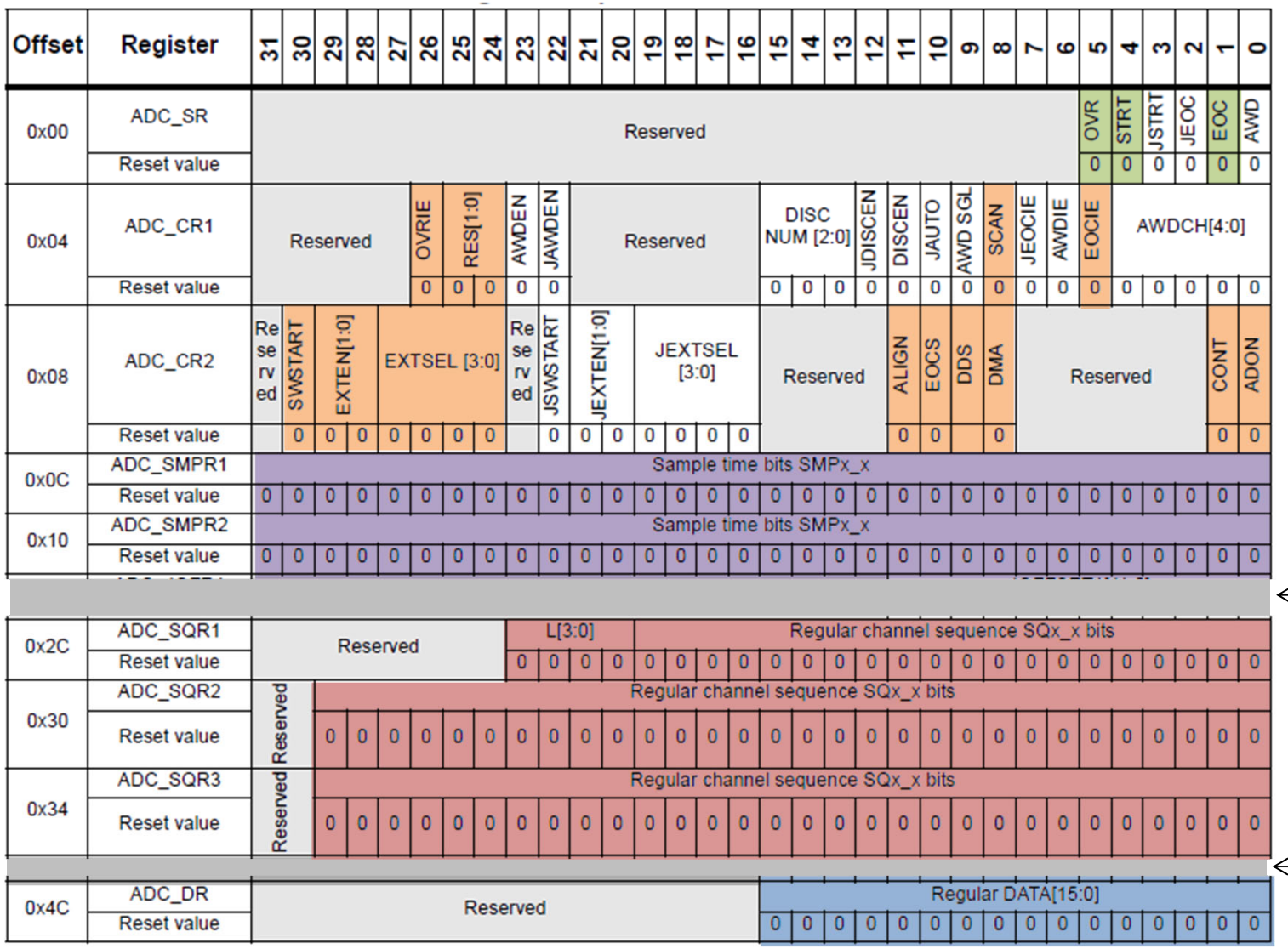
TSVREFE Enable/disable temp sensor and V_{REFINT}

VBATE Enable/disable V_{BAT}

ADCPRE[1:0] Prescaler for ADCCLK: 00/01/10/11 → APB2 clock divided by 2/4/6/8
APB2 clock = 42 MHz on CT_Board

All other positions kept at 0.
Features not relevant for this class

Specific Registers for each ADC



Specific Registers for each ADC

School of Engineering
InES Institute of Embedded Systems

- **Register Bits**

In our cases keep all other bits to '0'

ADC_SR	**ADC Status Register**
OVR	Overrun. 1→ Result data overwritten
STRT	Conversion started (regular channel)
EOC	End Of Conversion. Cleared by reading result of conversion

ADC_CR1	**ADC Control Register 1**
OVRIE	OVR Interrupt Enable
EOCIE	EOC Interrupt Enable
RES[1:0]	Conversion resolution: 00/01/10/11 → 12/10/8/6–bit
SCAN	Enable scan mode

ADC_CR2	**ADC Control Register 2**
SWSTART	Start conversion of regular channel by software. Cleared by HW
EXTEN[1:0]	Ext. trigger for regular channels 00/01/10/11 → disabled/pos edge/neg edge/pos & neg edge
EXTSEL	External event select to trigger conversion of a regular group
ALIGN	Data alignment : '0' → right aligned, '1' → left aligned
EOCS	EOC selection. **'0'**/ **'1'**→ EOC shows end of each sequence of conversions/end of each conversion
DMA	Enable DMA
CONT	Continuous mode: 0→ Single conversion mode 1→Continuous conversion mode
ADON	ADC On

Specific Registers for each ADC

- **Sample Time**

ADC_SMPR1 ADC Sample Time Register1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved					SMP18[2:0]			SMP17[2:0]			SMP16[2:0]			SMP15[2:1]	
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SMP15_0	SMP14[2:0]			SMP13[2:0]			SMP12[2:0]			SMP11[2:0]			SMP10[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

ADC_SMPR2 ADC Sample Time Register2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		SMP9[2:0]			SMP8[2:0]			SMP7[2:0]			SMP6[2:0]			SMP5[2:1]	
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SMP 5_0	SMP4[2:0]			SMP3[2:0]			SMP2[2:0]			SMP1[2:0]			SMP0[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

SMPx[2:0] Sampling time for channel x

'000'	3 cycles	'001'	15 cycles	'010'	28 cycles
'011'	56 cycles	'100'	84 cycles	'101'	112 cycles
'110'	144 cycles	'111'	480 cycles		

Specific Registers for each ADC

- **Sequence of Channels**

ADC_SQR1 ADC Sequence Register 1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								L[3:0]				SQ16[4:1]			
								rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SQ16_0	SQ15[4:0]					SQ14[4:0]					SQ13[4:0]				
rw	rw	rw	rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

ADC_SQR2 ADC Sequence Register 2

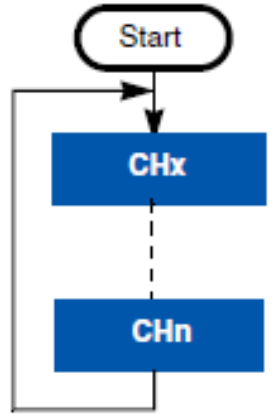
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		SQ12[4:0]					SQ11[4:0]					SQ10[4:1]			
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SQ10_0	SQ9[4:0]					SQ8[4:0]					SQ7[4:0]				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

ADC_SQR3 ADC Sequence Register 3

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		SQ6[4:0]					SQ5[4:0]					SQ4[4:1]			
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SQ4_0	SQ3[4:0]					SQ2[4:0]					SQ1[4:0]				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

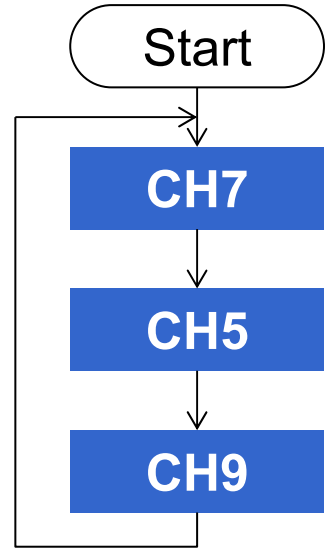


Specific Registers for each ADC

- **Sequence of Channels**

ADC_SQR1/2/3	**ADC Sequence Register**
L[3:0]	Sequence length: L+1 = Number of conversions in regular sequence '0000' → 1 conversion upto '1111' → 16 conversions
SQx[4:0]	Channel number of xth conversion in sequence, 1 <= x <= 16 e.g. SQ3[4:0] = 0x7 means the 3rd conversion takes place on channel 7

Example



SCAN = 0x1
L[3:0] = 0x2
SQ1[4:0] = 0x7
SQ2[4:0] = 0x5
SQ3[4:0] = 0x9

Programing the ADC → Example for PF8

School of Engineering
InES Institute of Embedded Systems

```
hal_rcc_set_peripheral(PER_GPIOF, ENABLE); // enable GPIO
hal_rcc_set_peripheral(PER_ADC3, ENABLE);  // enable ADC3

GPIOF->MODER |= (0x3 << 16);     // analog pin conf on PF8
ADCCOM->CCR = (0x3 << 16);       // ADC prescaler 8

ADC3->CR1 = 0x0;                 // 12 bit resolution, no scan
ADC3->CR2 = 0x1;                 // single conv., enable ADC, right align

ADC3->SMPR1 = 0x0;
ADC3->SMPR2 = (0x2 << (3*6));    // sample time = 28 cycles (= binary code 010) for channel 6
                                 // The sample times of all other channels are set to 3 cycles (= binary
                                 // code 000), but since these channels are unused, this does not matter

ADC3->SQR1 = 0x0;                // L = '0000' -> sequence length: 1
ADC3->SQR2 = 0x0;
ADC3->SQR3 = 0x6;                // ch6 is first in sequence

while (1) {
    ADC3->CR2 |= (0x1 << 30);        // start conversion
    while(!(ADC3->SR & 0x2)) {       // wait while conversion not finished
    }
    CT_SEG7->BIN.HWORD = ADC3->DR;  // show on 7-segment display
}
```

PF8	I/O	FT	(4)	SPI5_MISO, SAI1_SCK_B, TIM13_CH1, FMC_NIOWR, EVENTOUT	ADC3_IN6

What is the max. achievable sampling rate with these settings and APB2 clock = 42 MHz?

Functional Summary of ADC

3) Signals used to inform the CPU of the state of the conversion process. Status register can be checked or interrupts generated.

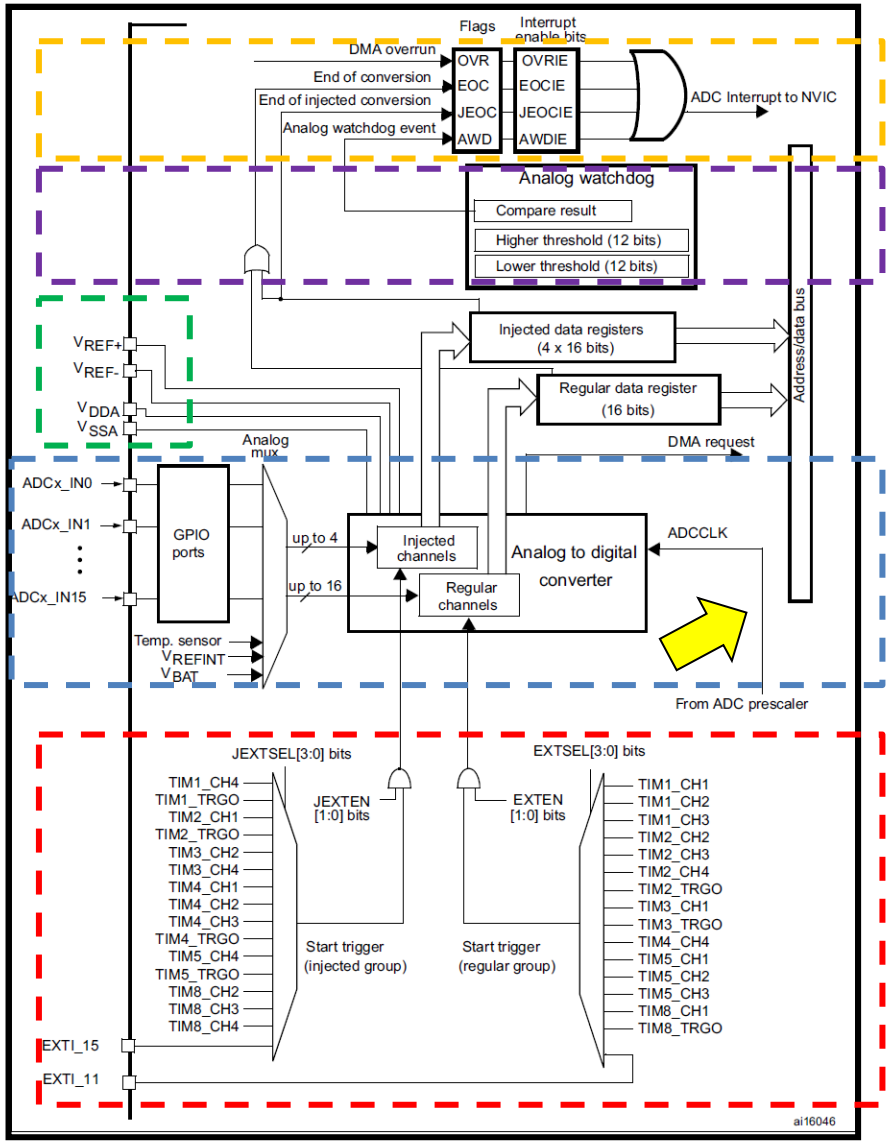
4) Analog watchdog compares conversion results with programmed min/max limits.

5) Analog pins to set the references

1) 3 ADCs (capable of 12/10/8/6 bit resolution)
- 16 possible external sources (pins) each
- 3 possible internal sources

2) Conversion triggered by CPU, by internal sources or by external sources

6) ADCCLK is used as clock for conversions.
Generated by dedicated prescaler that allows APB2 clock to be scaled down by 2/4/6/8



Features and Functionality

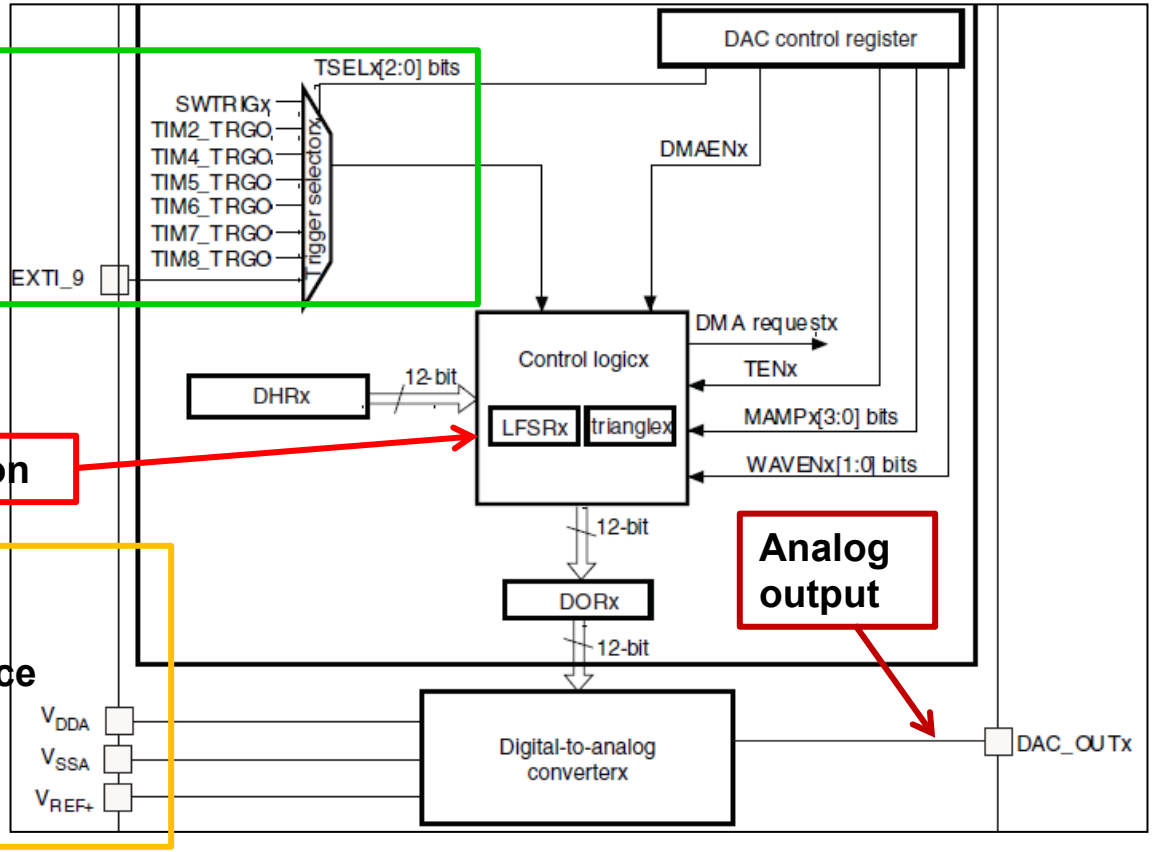
DACs on STM32F429 (Optional)

Features of STM32F429 DAC (optional)

2 voltage output DACs. Can be combined with DMA
- **Independent output each**
- **Both support 12-bit and 8-Bit modes. Left or right data alignment in 12-bit mode**

Conversion can be triggered by:
- **SW**
- **internal timers**
- **external event**

Source is SW selectable



Noise/Triangular-wave generation

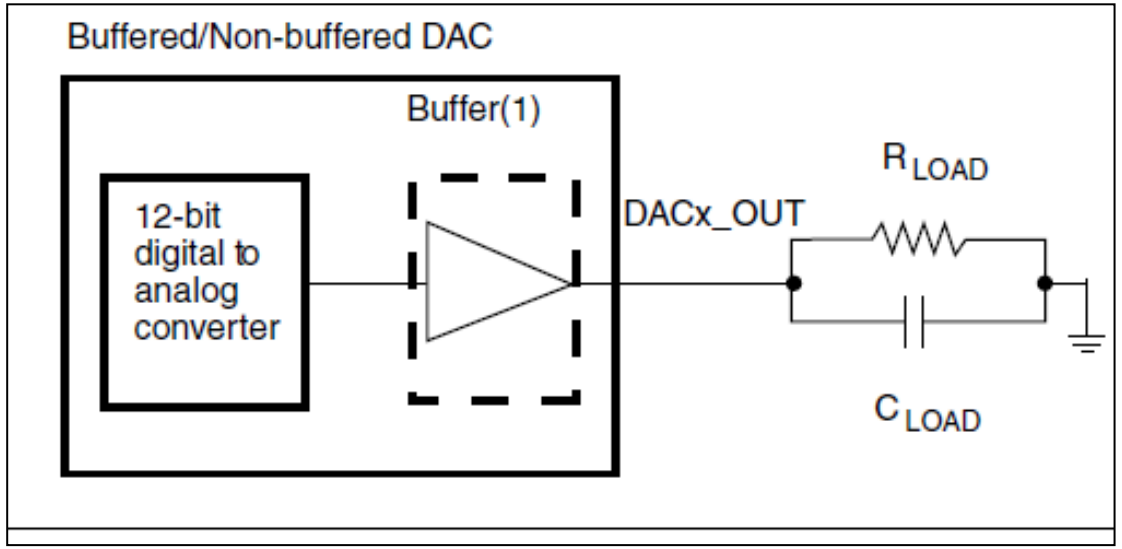
$1.8V <= V_{DDA} <= 3.6V$
$1.8V <= V_{REF+} <= V_{DDA}$
- V_{REF+}: External voltage reference
- V_{DDA}: Analog power supply
- V_{SSA}: Analog GND

Features of STM32F429 DAC

School of Engineering
InES Institute of Embedded Systems

Overview

- The 2 DACs can be grouped for synchronous update
  - Independent or simultaneous modes
- Integrated output buffers can be used to reduce the output impedance, and to drive external loads directly
  - No need for external operational amplifier



Features of STM32F429 DAC

School of Engineering
InES Institute of Embedded Systems

- **Overview of some DAC electrical parameters**
  - The use of the output buffer reduces external components and helps improve performance if the load is important

Table 87. DAC characteristics

Symbol	Parameter	Min	Typ	Max	Unit	Comments
R_{LOAD} (2)	Resistive load with buffer ON	5	-	-	kΩ	
R_O (2)	Impedance output with buffer OFF	-	-	15	kΩ	When the buffer is OFF, the Minimum resistive load between DAC_OUT and V_{SS} to have a 1% accuracy is 1.5 MΩ
C_{LOAD} (2)	Capacitive load	-	-	50	pF	Maximum capacitive load at DAC_OUT pin (when the buffer is ON).
DAC_OUT min (2)	Lower DAC_OUT voltage with buffer ON	0.2	-	-	V	It gives the maximum output excursion of the DAC. It corresponds to 12-bit input code (0x0E0) to (0xF1C) at V_{REF+} = 3.6 V and (0x1C7) to (0xE38) at V_{REF+} = 1.8 V
DAC_OUT max (2)	Higher DAC_OUT voltage with buffer ON	-	-	V_{DDA} – 0.2	V	

DAC conversion

School of Engineering
InES Institute of Embedded Systems

- **DAC output voltage**
  - Digital values are converted to output voltages on a linear conversion between 0 and $\mathbf{V_{REF+}}$
  - Analog output voltages on each DAC channel pin are determined by the following equation: $DAC_{output} = \mathbf{V_{REF+}} * DOR / 4095$
    - DOR is the digital value that should be converted
    - Use of an external $\mathbf{V_{REF+}}$ can help to improve results.
      - By choosing a smaller $\mathbf{V_{REF+}}$,the minimal “analog step” is reduced
      - Using a dedicated pin allows the use of less noisy references

- **Automatic waveforms generation**
  - DAC can be set to produce noise/triangular signals
    - Useful for testing

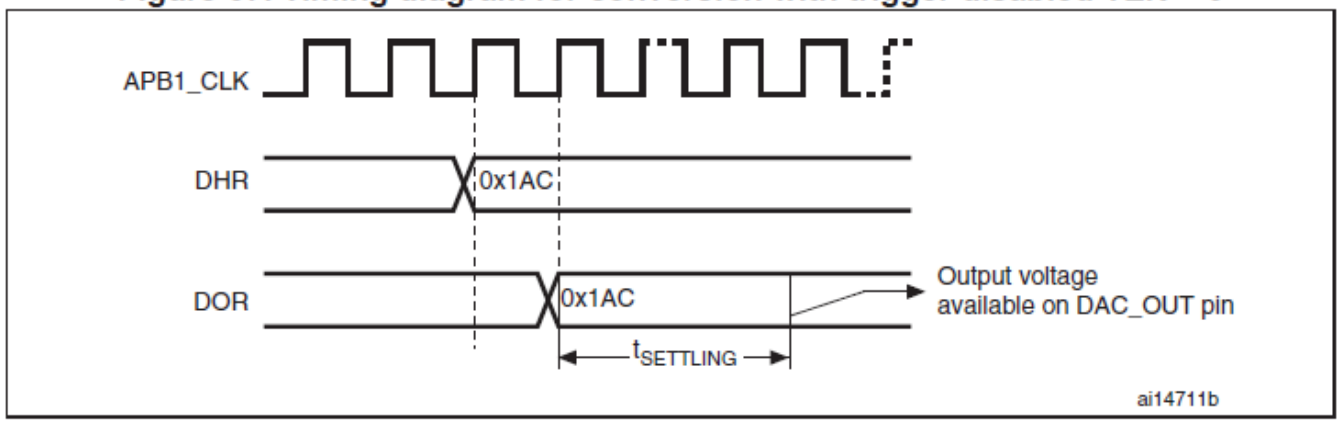
DAC conversion

School of Engineering
InES Institute of Embedded Systems

Conversion timing

- DAC_DORx cannot be written directly
  - Data transfer to the DAC performed by loading DAC_DHRx register
  - Data in DAC_DHRx transferred to DAC_DORx after one APB1 clock cycle
    - If HW trigger selected, transfer performed 3 APB1 clock cycles after trigger
- When DAC_DORx loaded with DAC_DHRx contents, the analog output voltage available after a time $t_{SETTLING}$
  - $t_{SETTLING}$ depends on power supply voltage and analog output load.

Figure 67. Timing diagram for conversion with trigger disabled TEN = 0



DAC conversion

School of Engineering
InES Institute of Embedded Systems

Conversion timing

- $t_{SETTLING}$ is given in the table. Value higher when many bits change
- With changes of 1LSB, update rates of 1MS/s are possible

Table 87. DAC characteristics (continued)

Symbol	Parameter	Min	Typ	Max	Unit	Comments
$t_{SETTLING}$(4)	Settling time (full scale: for a 10-bit input code transition between the lowest and the highest input codes when DAC_OUT reaches final value ±4LSB	-	3	6	µs	$C_{LOAD} \leq$ 50 pF, $R_{LOAD} \geq$ 5 kΩ
THD(4)	Total Harmonic Distortion Buffer ON	-	-	-	dB	$C_{LOAD} \leq$ 50 pF, $R_{LOAD} \geq$ 5 kΩ
Update rate(2)	Max frequency for a correct DAC_OUT change when small variation in the input code (from code i to i+1LSB)	-	-	1	MS/s	$C_{LOAD} \leq$ 50 pF, $R_{LOAD} \geq$ 5 kΩ

Some DAC characteristics

School of Engineering
InES Institute of Embedded Systems

Current consumption, offset error are examples of important characteristics

Table 87. DAC characteristics (continued)

Symbol	Parameter	Min	Typ	Max	Unit	Comments
I_{DDA}(4)	DAC DC VDDA current consumption in quiescent mode(3)	-	280	380	µA	With no load, middle code (0x800) on the inputs
		-	475	625	µA	With no load, worst code (0xF1C) at V_{REF+} = 3.6 V in terms of DC consumption on the inputs
DNL(4)	Differential non linearity Difference between two consecutive code-1LSB)	-	-	±0.5	LSB	Given for the DAC in 10-bit configuration.
		-	-	±2	LSB	Given for the DAC in 12-bit configuration.
INL(4)	Integral non linearity (difference between measured value at Code i and the value at Code i on a line drawn between Code 0 and last Code 1023)	-	-	±1	LSB	Given for the DAC in 10-bit configuration.
		-	-	±4	LSB	Given for the DAC in 12-bit configuration.
Offset(4)	Offset error (difference between measured value at Code (0x800) and the ideal value = V_{REF+}/2)	-	-	±10	mV	Given for the DAC in 12-bit configuration
		-	-	±3	LSB	Given for the DAC in 10-bit at V_{REF+} = 3.6 V
		-	-	±12	LSB	Given for the DAC in 12-bit at V_{REF+} = 3.6 V

DAC Registers

School of Engineering
InES Institute of Embedded Systems

Table 76. DAC register map

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	DAC_CR	Reserved		DMAUDRIE2	DMAEN2	MAMP2[3:0]				WAVE 2[2:0]		TSEL2[2:0]			TEN2	BOFF2	EN2	Reserved		DMAUDRIE1	DMAEN1	MAMP1[3:0]				WAVE 1[2:0]		TSEL1[2 :0]			TEN1	BOFF1	EN1
0x04	DAC_ SWTRIGR	Reserved																														SWTRIG2	SWTRIG1
0x08	DAC_ DHR12R1	Reserved																				DACC1DHR[11:0]											
0x0C	DAC_ DHR12L1	Reserved																DACC1DHR[11:0]													Reserved		
0x10	DAC_ DHR8R1	Reserved																								DACC1DHR[7:0]							
0x14	DAC_ DHR12R2	Reserved																				DACC2DHR[11:0]											
0x18	DAC_ DHR12L2	Reserved																DACC2DHR[11:0]													Reserved		
0x1C	DAC_ DHR8R2	Reserved																								DACC2DHR[7:0]							

Control reg (DAC1/2)

Swtrigger (DAC1/2)

Data regs DAC1
12/8-Bit
L/R aligned

Data regs DAC2
12-bit / 8-Bit
Left or Right

Register with offset <0x20 - 0x34> are not relevant for this class

- Once the DAC channelx is enabled, the corresponding GPIO pin (PA4 or PA5) is automatically connected to the analog converter output (DAC_OUTx)
- In order to avoid parasitic consumption, the PA4 or PA5 pin should first be configured to analog (AIN)

DAC Registers

Table 76. DAC register map

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	DAC_CR	Reserved		DMAUDRIE2	DMAEN2	MAMP2[3:0]				WAVE 2[2:0]		TSEL2[2:0]			TEN2	BOFF2	EN2	Reserved		DMAUDRIE1	DMAEN1	MAMP1[3:0]				WAVE 1[2:0]		TSEL1[2 :0]			TEN1	BOFF1	EN1
0x04	DAC_ SWTRIGR	SWTRIG software trigger DAC1/2 1 → SW trigger enabled													Reserved			Cleared by HW once DHR loaded into DOR. Also disabled by SW														SWTRIG2	SWTRIG1
0x08	DAC_	Reserved																		DACC1DHR[11:0]													

Command Register

EN (Enable) 1→ DAC enabled
BOFF (Buffer Off) 1 → output buffer disabled, 0 → output buffer enabled
TEN (Trigger enable) DAC Channel trigger 0→ disabled, 1→ enabled,
TSEL (Select ext trigger, TEN must be 1) 111 → select Swtrigger
WAVE sel wave generator keep at 00 → Wave generation disabled
MAMP select mask/amplitude in waveform generation keep at 000 if no wavegeneration

DMAEN, DMAUDRIE Keep at 0. not relevant for this class

Using the DAC

School of Engineering
InES Institute of Embedded Systems

- **Steps needed to get DAC running**
  - Address of register: DAC Base Address + Register Offset
    - DAC range is <0x4000 7400 - 0x4000 77FF>
  - Choosing a DAC
    - Choose the DAC you want to use
    - Configure port pin(s) accordingly. Do you need the output buffer?
  - Setting some parameters (conversion mode, clocking, …)
    - Decide how to clock the DAC. Set trigger source
      - Set up sources to trigger conversion start
      - Or CPU to control the conversion
  - Starting the DAC, CPU mode
    - Switch on the DAC, write the data in the data register.
    - Wait (the time interval you want)
    - Write the next sample

Using the DAC (example in assembler)

```
    ; Configuration
init_dac      ; Clock configuration
            LDR  R6, =REG_RCC_AHB1ENR
            LDR  R7, =0x1         ; Enable GPIOA clock
            BL   set_sfr

            LDR  R6, =REG_RCC_APB1ENR
            LDR  R7, =0x20000000      ; Enable DAC clock
            BL   set_sfr

            ; Analog pin configuration (PA.4)
            LDR  R6, =REG_GPIOA_MODER
            LDR  R7, =0x300
            BL   set_sfr

            ; DAC configuration
            LDR  R6, =REG_DAC_CR
            LDR  R7, =0x1             ; Enable ch. 1
            BL   set_sfr

            LDR  R6, =REG_DAC_DHR8R1
            LDR  R7, =0x0             ; Set initial value (=0)
            BL   set_sfr
```

```
        ; Starting conversion

LDR  R6, =REG_DAC_DHR8R1
            LDR  R0, [R6]
            LDR  R1, =0xff
            BICS R0, R0, R1           ; Clear bits
            STR  R0, [R6]
            BL   set_sfr              ; Set value in R7
```

Conclusions

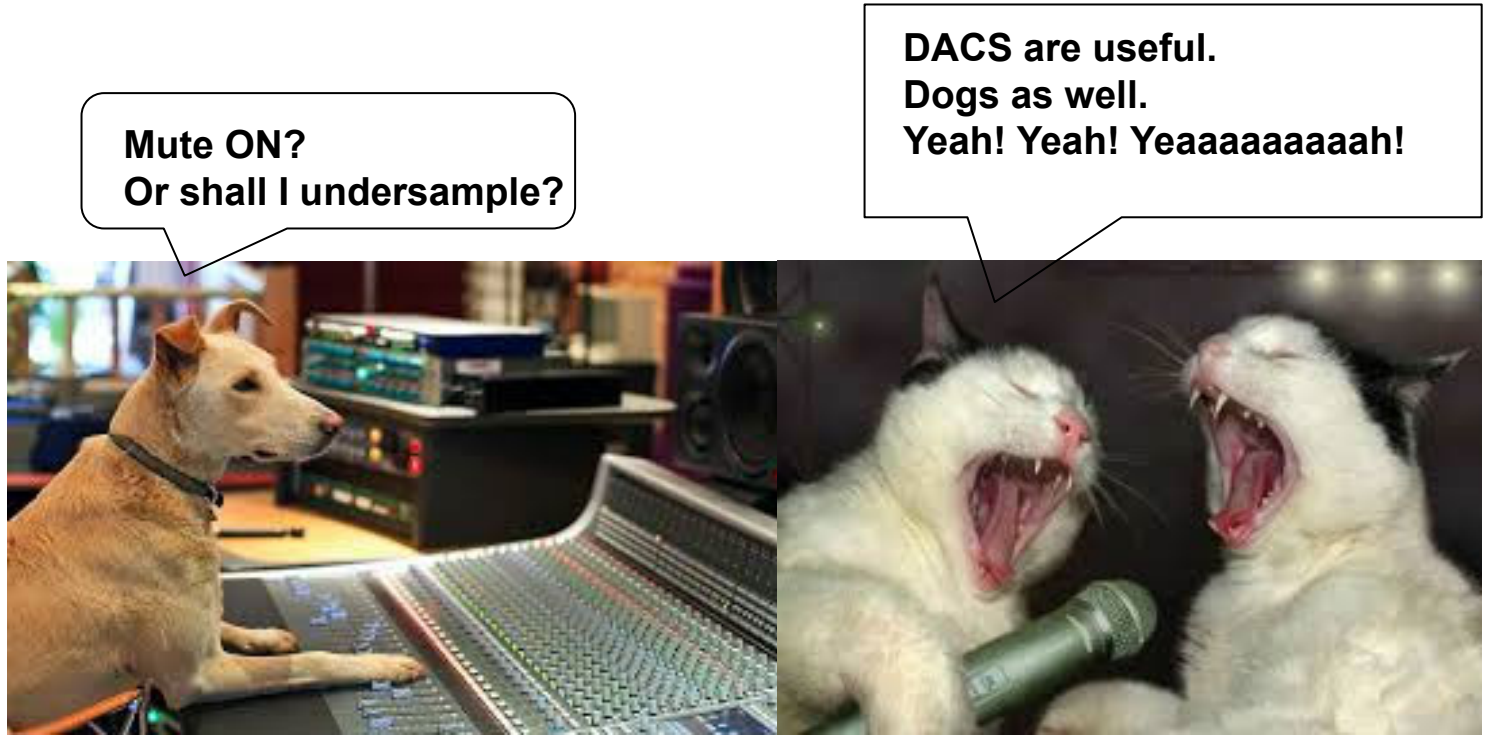
School of Engineering
InES Institute of Embedded Systems

- **ADC/DAC used to interface the analog and digital world**
  - Applications such as instrumentation, audio, control
- **Rate at which data is converted and number of bits used are important parameters**
- **Devices introduce errors that affect results of conversion**
- **STM32F429 provides ADC and DAC with several features**
  - Those features and the way to use them are found in the datasheet and reference manual
- **ADC and DAC have analog and digital parts**
  - They are known as mixed-signal devices

Applications

- **Full of ADCs and DACs (DACs and CADs) (dogs and cats)**
- **When you have forgotten everything about ADCs and DACs**

  Just remember the music



Sources: http://www.recordproduction.com/multimedia.htm http://rayhiltz.com/wp-content/uploads/2011/07/69-Drunk-Karaoke-Cats-568x384.jpg

References

- **References**

  Documents used in this lesson

  - [1] STM32F42xxx Datasheet
  - [2] STM32F42xxx Reference manual (RM0090)
  - [3] AN3116 “STM32™’s ADC modes and their applications”
  - [4] http://www.maximintegrated.com/en/app-notes/index.mvp/id/641
  - [5] Atmel AVR127: Understanding ADC Parameters