STM32F3 ADC

CUAUHTÉMOC CARBAJAL

1

18/10/2013

References

- http://www.embedds.com/introducing-to-stm32-adc-programming-part1/
- http://controlsoft.nmmu.ac.za/STM32F0-Discovery-Board/Example-programs/Analog
- http://mipsandchips.blogspot.mx/
- STM32F3 Microcontroller Reference Manual

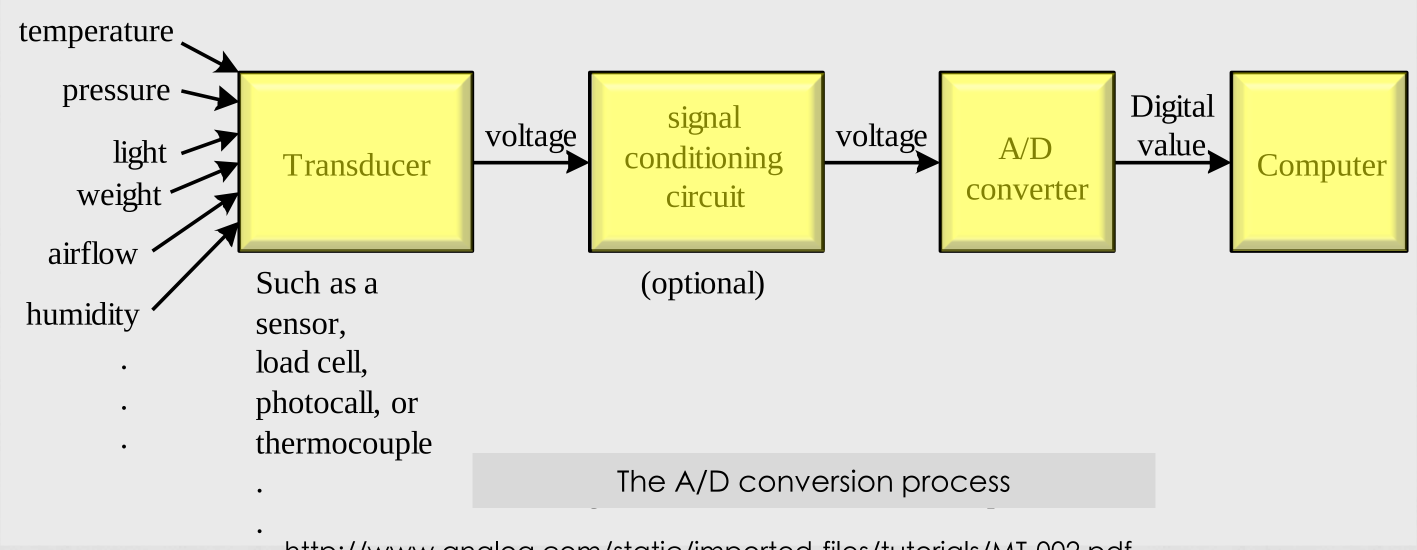
ADC PRINCIPLES

Basics of A/D Conversion (1 of 2)

- Many embedded systems need to deal with nonelectric quantities: weight, humidity, pressure, weight, mass or airflow, temperature, light intensity, and speed.
- These nonelectric quantities are analog in nature.
- Analog quantities must be converted into digital format so that they can be processed by the computer.
- An A/D converter can only deal with electric voltage.

Basics of A/D Conversion (2 of 2)

- Any nonelectric quantity must be converted into an electric quantity using a certain type of transducer.
- A transducer converts a nonelectric quantity into an electric quantity.
- The output of a transducer may not be in a suitable range for A/D conversion.
- A signal conditioning circuit is needed to shift and scale the transducer output to a range suitable for A/D conversion.
- A lowpass/bandpass filter is required to remove unwanted signals outside the bandwidth of interest and prevent aliasing.

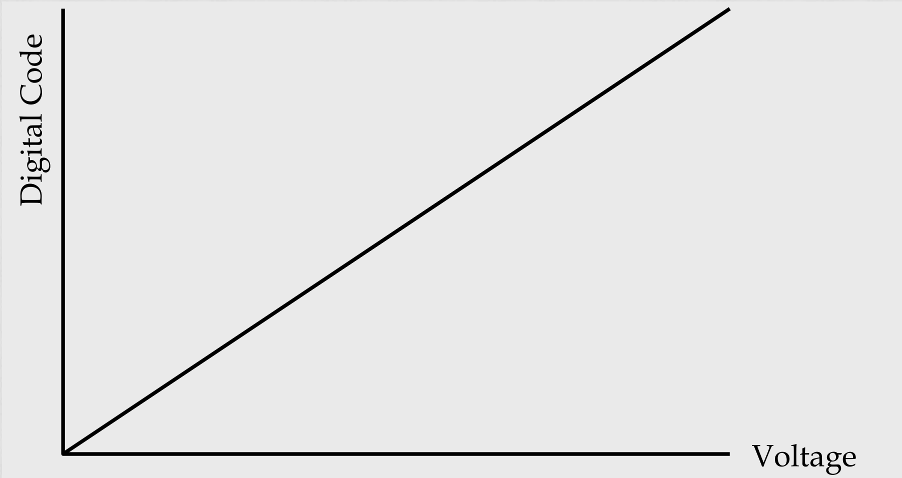


The A/D conversion process

http://www.analog.com/static/imported-files/tutorials/MT-002.pdf

Analog Voltage and Digital Code Characteristic (1 of 2)

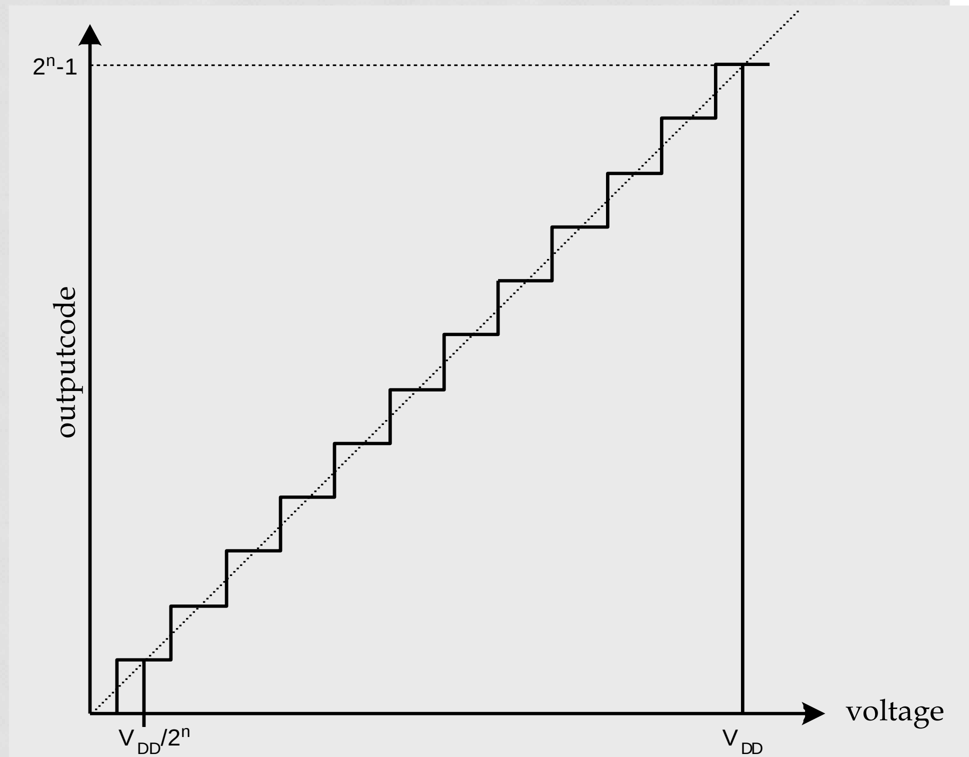
- An ideal A/D converter should have a characteristic as shown.
- An A/D converter with characteristic as shown would need infinite number of bits to represent the A/D conversion result.



An ideal ADC output characteristic

Analog Voltage and Digital Code Characteristic (2 of 2)

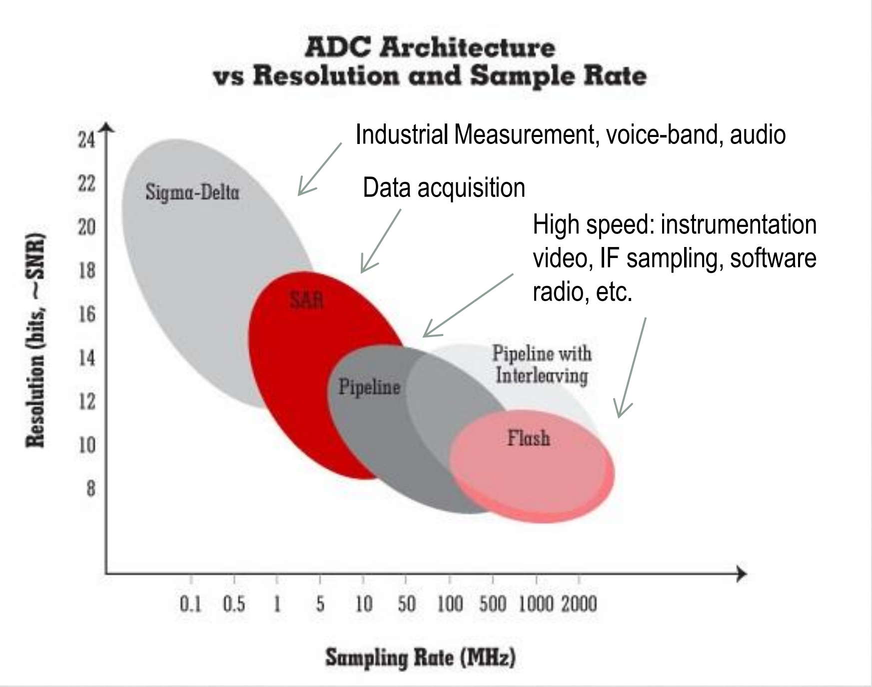
- An n-bit A/D converter has 2^n possible output code values.
- The output characteristic of an n-bit A/D ideal converter is shown.
- The area above and below the dotted line is called quantization error.
- Using n-bit to represent A/D conversion has an average error of $V_{DD}/2^{n+1}$.
- A real A/D converter output may have nonlinearity and nonmonotonicity errors



Output characteristic of an ideal n-bit ADC

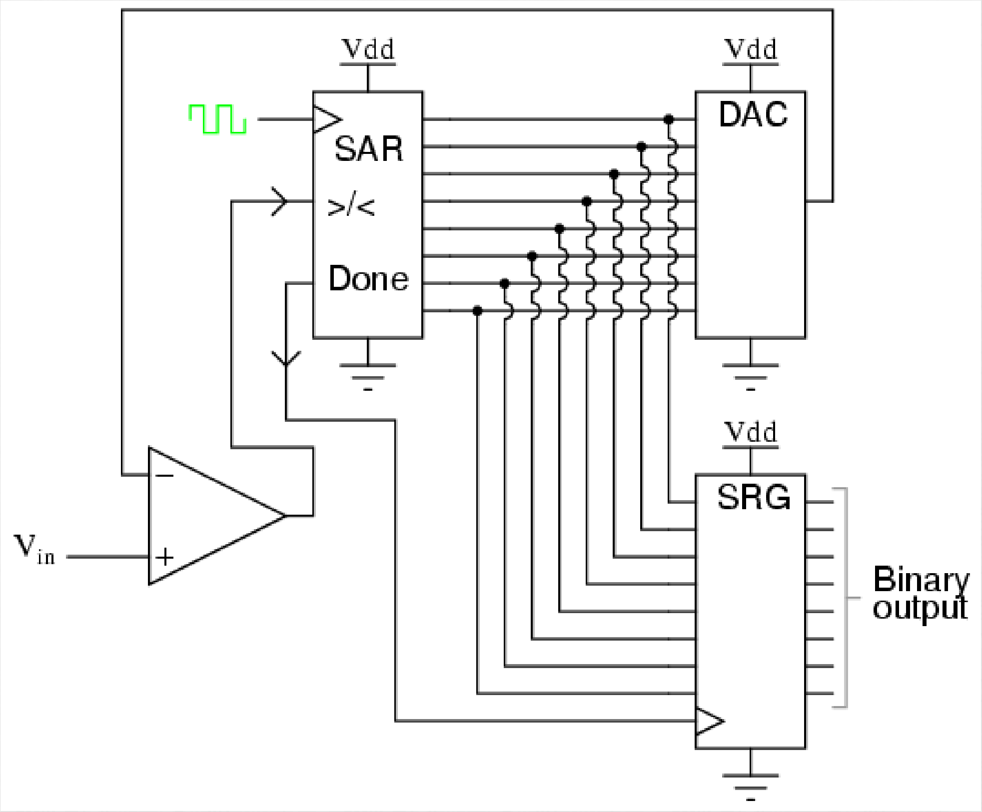
A/D Conversion Algorithms

- Dominant:
  - Delta-Sigma
  - Successive Approximation
  - Pipeline
  - Flash
- Other:
  - Tracking
  - Stair Step Ramp
  - Single and Dual Slope



http://www.analog.com/library/analogdialogue/archives/39-06/architecture.html
http://www.maximintegrated.com/app-notes/index.mvp/id/1041

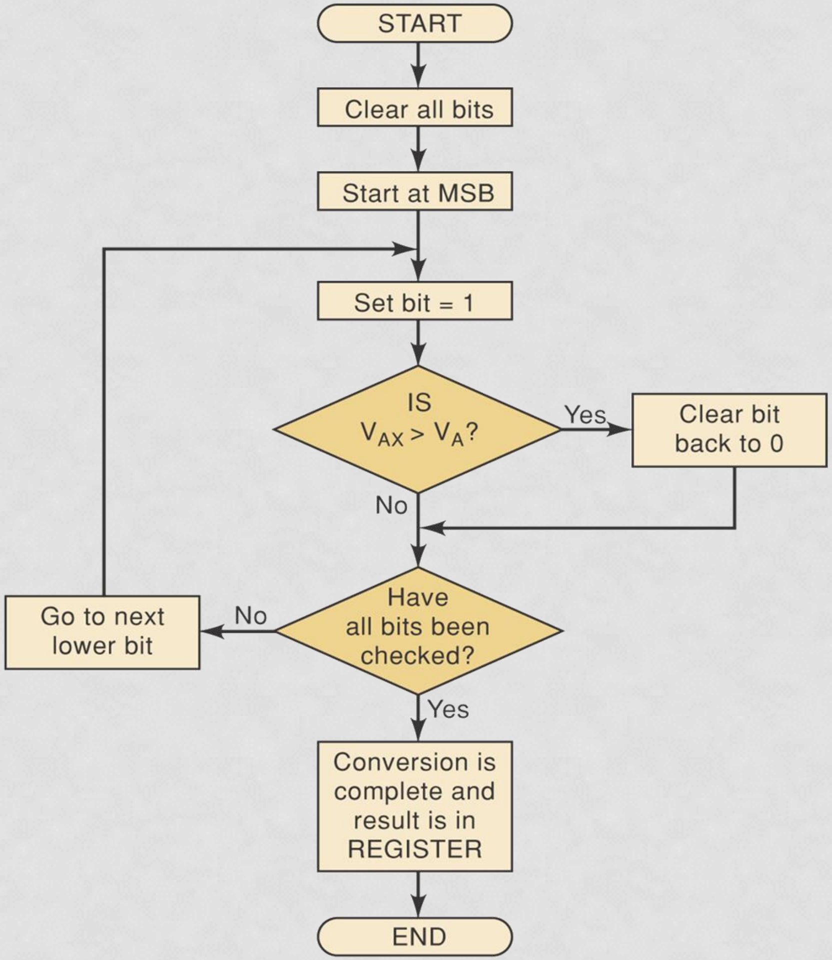
A/D Successive Approximation (1 of 3)



- Most widely used A/D converter
- Faster than other methods except for flash method
- Fixed conversion time

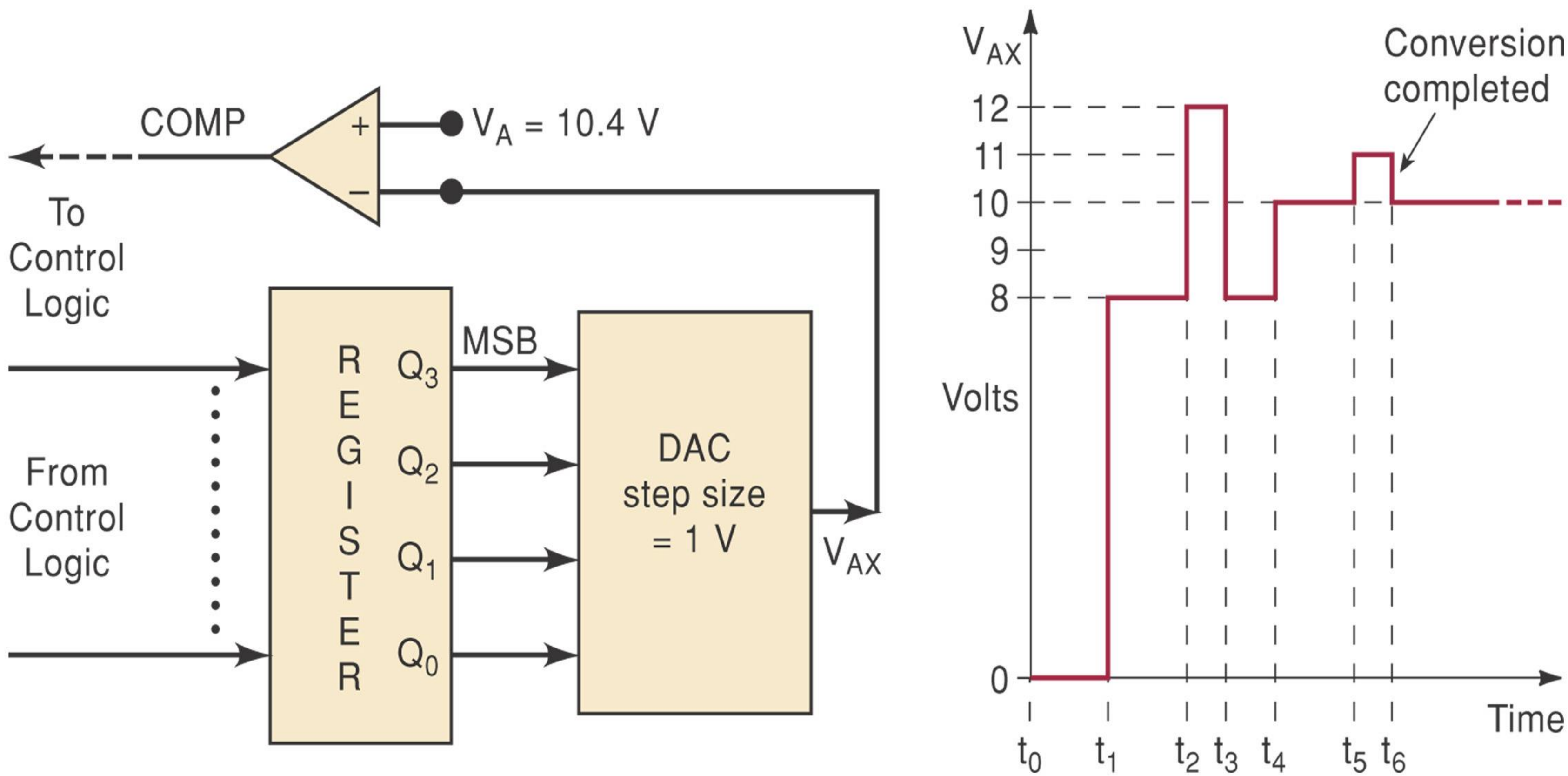
Successive Approximation Method (2 of 3)

- Approximates the analog signal in n steps.
- The first step initializes the SAR register to 0.
- Perform a series of guessing steps that starts from the most significant bit and proceeding toward the least significant bit.
- For every bit in SAR register guess it to be 1.
- Converts the value of the SAR register to analog voltage.
- Compares the D/A output with the analog input and clears the bit to 0 if the D/A output is larger.

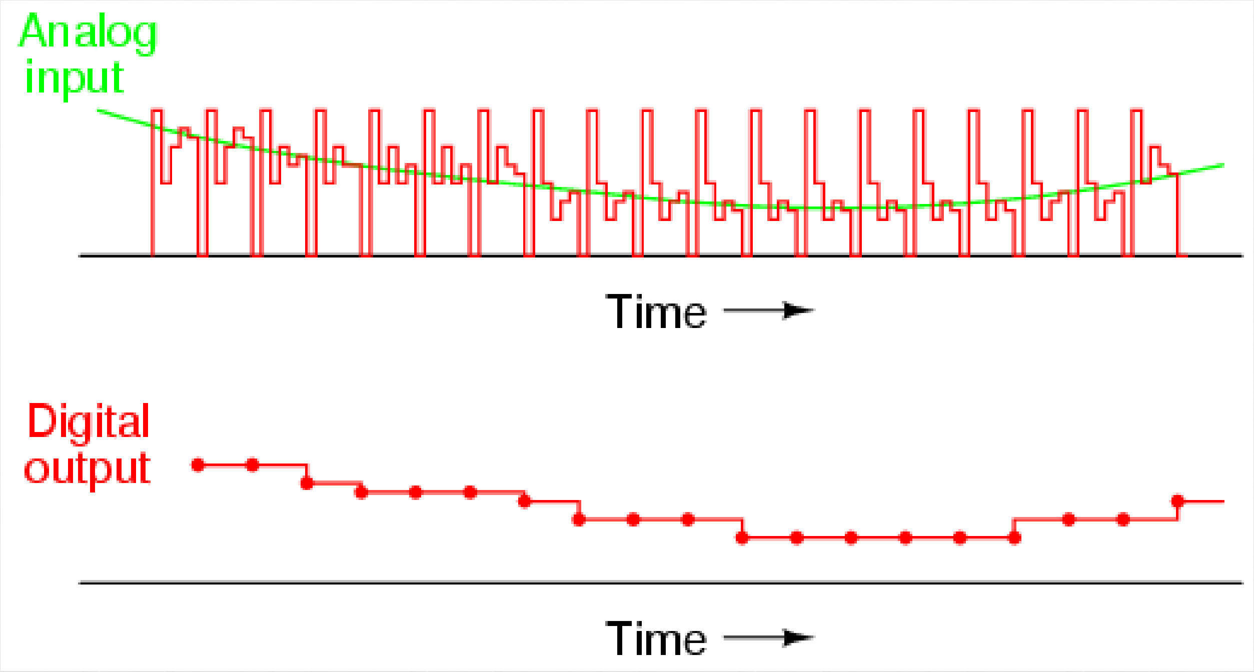


(b)

ILLUSTRATION OF FOUR-BIT SAC OPERATION USING A DAC STEP SIZE OF 1 V AND V_A = 10.4 V.



A/D Successive Approximations



Optimal Voltage Range for A/D Conversion

- Needs a low reference voltage (V_{RL}) and a high reference voltage (V_{RH}) in performing A/D conversion.
- V_{RL} is often set to ground level.
- V_{RH} is often set to VDD.
- Most A/D converter are ratiometric
  - A 0 V (or V_{RL}) analog input is converted to the digital code of 0.
  - A V_{DD} (or V_{RH}) analog input is converted to the digital code of $2^n - 1$.
  - A V_K input will be converted to the digital code $k = V_K \times 2^n \div V_{DD}$.
- The A/D conversion result will be most accurate if the value of analog signal covers the whole voltage range from V_{RL} to V_{RH}.
- The A/D conversion result k can be translated back to an analog voltage V_K by the following equation:

$$V_K = V_{RL} + (\text{range} \times k) \div 2^n$$

Equation 1

Example

Suppose that there is a 10-bit A/D converter with V_{RL} = 1 V and V_{RH} = 4V. Find the corresponding voltage values for the A/D conversion results of 25, 80, 240, 500, 720, 800, and 900.

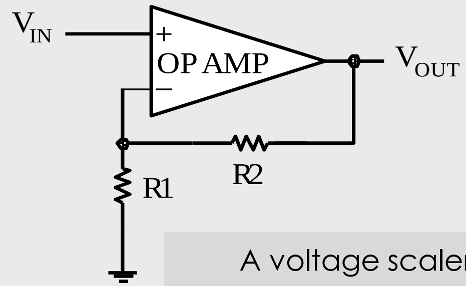
Solution

range = $V_{RH} – V_{RL}$ = 4V – 1V = 3V

$V(25) = 1\text{ V} + (3 \times 25) \div 2^{10} = 1.07324\text{ V}$
$V(80) = 1\text{ V} + (3 \times 80) \div 2^{10} = 1.23438\text{ V}$
$V(240) = 1\text{ V} + (3 \times 240) \div 2^{10} = 1.70313\text{ V}$
$V(500) = 1\text{ V} + (3 \times 500) \div 2^{10} = 2.46484\text{ V}$
$V(720) = 1\text{ V} + (3 \times 720) \div 2^{10} = 3.10938\text{ V}$
$V(800) = 1\text{ V} + (3 \times 800) \div 2^{10} = 3.34375\text{ V}$
$V(900) = 1\text{ V} + (3 \times 900) \div 2^{10} = 3.63672\text{ V}$

Scaling Circuit

- Some transducer has an output voltage in the range of 0 ~ V_Z, where $V_Z < V_{DD}$.
- V_Z can be much smaller than V_{DD}.
- When V_Z is much smaller than V_{DD}, the A/D conversion result cannot be accurate.
- The solution to this problem is to use an scaling circuit to amplify the transducer output to cover the whole range of 0 V VRH to V_{DD}.



A voltage scaler

$$A_V = V_{OUT} \div V_{IN} = (R_1 + R_2) \div R_1$$

Equation 2

$$= 1 + R_2/R_1$$

Example

Choose appropriate values of R1 and R2 in to scale a voltage in the range of 0~200mV to 0~5V.

Solution

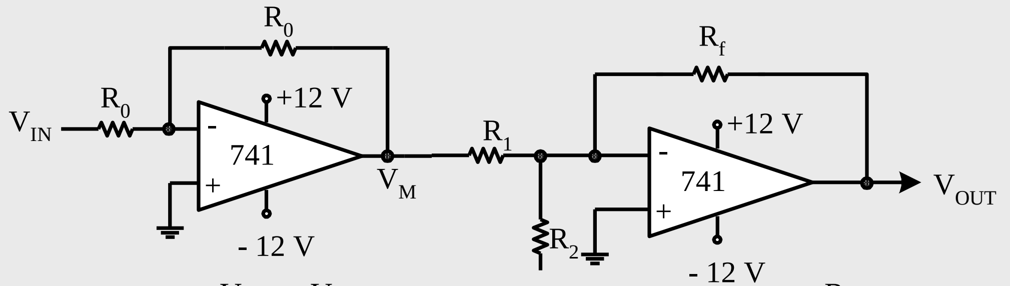
$A_V = 1 + R_2/R_1 = 5V / 200mV = 25$

R2/R1 = 24

Choose R_1 = 4.1 KΩ and R_2 = 100 K Ω to achieve the desired ratio.

Voltage Translation Circuit

- Some transducer has output voltage in the range from V_1 to V_2 ($V_2 > V_1$).
- The accuracy of the A/D conversion will be more accurate if this voltage can be scaled and shifted to $0 \sim V_{DD}$.
- The circuit shown can shift and scale the voltage from V_1 to V_2 to the range of $0 \sim V_{DD}$.



$V_M = -V_{IN}$

$$V_{OUT} = \frac{R_f}{R_1} V_{IN} - \frac{R_f}{R_2} V_1$$

Equation 3

Level shifting and scaling circuit

Example

Choose appropriate resistor values and the adjusting voltage so that the circuit shown in the previous figure can shift the voltage from the range of –1.2 V ~ 3.0 V to the range of 0V ~ 5V.

Solution: Applying Equation 3:

$$0 = -1.2 \times (R_f/R_1) - (R_f/R_2) \times V_1$$
$$5 = 3.0 \times (R_f/R_1) - (R_f/R_2) \times V_1$$

- By choosing R_0 = R_1 = 10 KΩ, R_2 = 100 KΩ, R_f = 12 KΩ, and V_1 = -12V, one can translate and scale the voltage to the desired range.

STM32F3 ADC

Introduction (1)

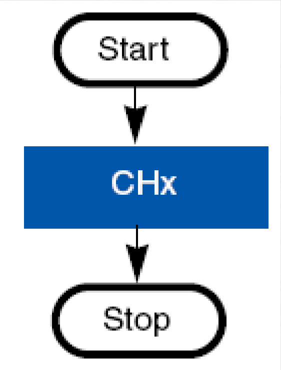
- STM32 microcontrollers have one of the most advanced ADCs on the microcontroller market. You could imagine a multitude of applications based on the STM32 ADC features.
- Some ADC modes are provided to simplify measurements and give efficient results in applications such as motor control.

Introduction (2)

- Each STM32F3 ADC is a 12-bit successive approximation ADC.
- Each ADC has **up** to **18** multiplexed channels allowing the measurements of **up** to **16** external sources and **up** to 4 internal sources.
- A/D conversion can be performed in:
  - single or multiple channels,
  - discontinuous or continuous conversion mode,
  - regular or injected mode,
  - single or dual (simultaneous) mode
- The result of the ADC is stored in a left-aligned or right-aligned 16-bit data register.
- The ADCs are mapped on the AHB bus to allow fast data handling.
- The analog watchdog features allow the application to detect if the input voltage goes outside the user-defined high or low thresholds.

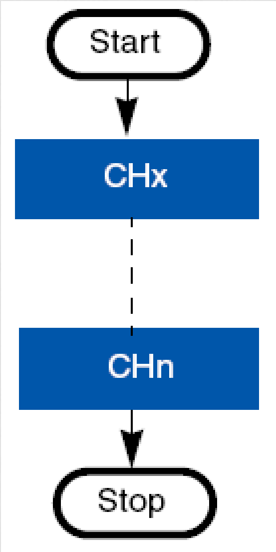
Conversion Modes (1)

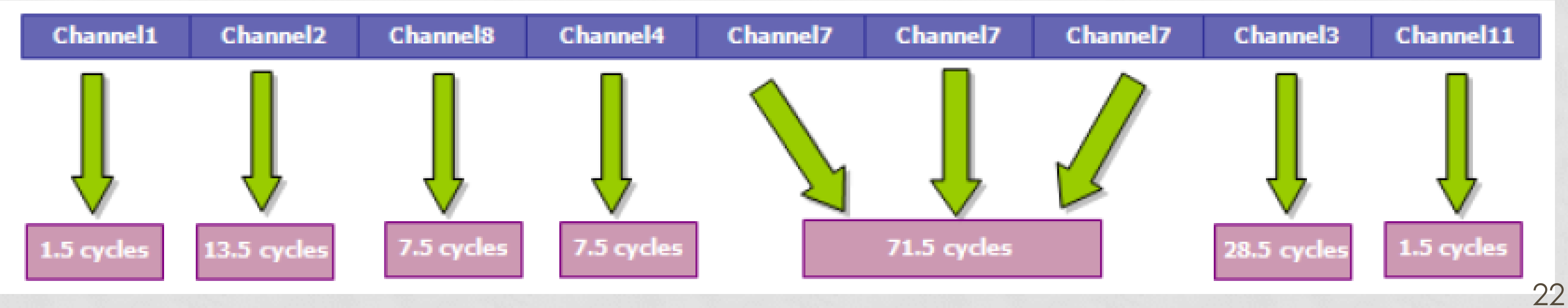
- Single-channel, single conversion mode
  - This is the simplest ADC mode. In this mode, the ADC performs the single conversion (single sample) of a single channel x and stops after completion of the conversion.
  - This mode can be used for the measurement of a voltage level to decide if the system can be started or not. Measure the voltage level of the battery before starting the system: if the battery has a low level, the “low battery” message appears. In this case, do not start the system.



Conversion Modes (2)

- Multichannel (scan), single conversion mode
  - This mode is used to convert some channels successively in independent mode. With the ADC sequencer, you can use this ADC mode to configure any sequence of up to **16** channels successively with different sampling times and in different orders. You can for example carry out the sequence shown in the figure below. In this way, you do not have to stop the ADC during the conversion process in order to reconfigure the next channel with a different sampling time. This mode saves additional CPU load and heavy software development.





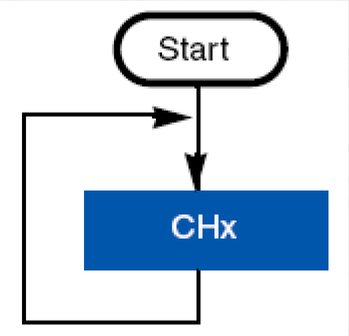
ADC sequencer converting 7 channels with different configured sampling times

Conversion Modes (3)

- Multichannel (scan), single conversion mode (cont)
  - This mode can be used when starting a system depends on some parameters like knowing the coordinates of the arm's tip in a manipulator arm system. In this case, you have to read the position of each articulation in the manipulator arm system at power-on to determine the coordinates of the arm's tip.
  - This mode can also be used to make single measurements of multiple signal levels (voltage, pressure, temperature, etc.) to decide if the system can be started or not in order to protect the people and equipment.
  - It can likewise be used to convert signals coming from strain gauges to determine the directions and values of the different strains and deformations of an object.

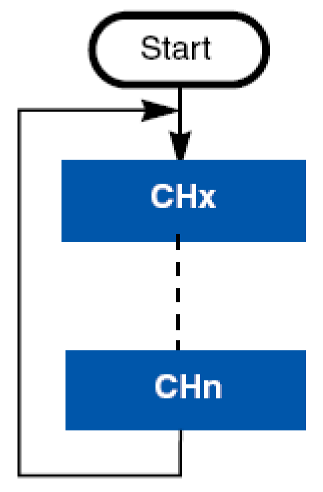
Conversion Modes (4)

- Single-channel continuous conversion mode
  - The single-channel continuous conversion mode converts a single channel continuously and indefinitely in regular channel conversion.
  - The continuous mode feature allows the ADC to work in the background. The ADC converts the channels continuously without any intervention from the CPU. Additionally, the DMA can be used in circular mode, thus reducing the CPU load.
  - This ADC mode can be implemented to monitor a battery voltage, the measurement and regulation of an oven temperature, etc.
  - In the case of the oven temperature regulation, the temperature is read and compared to the temperature set by the user. When the oven temperature reaches the desired temperature, the heating resistor is powered off.



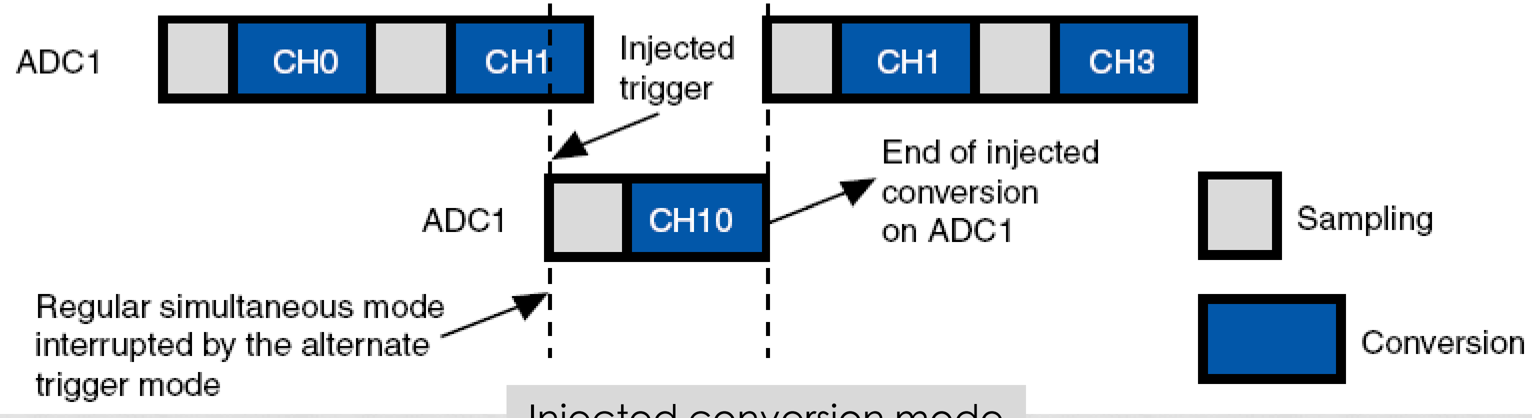
Conversion Modes (5)

- Multichannel (scan) continuous conversion mode
  - The multichannel, or scan, continuous mode can be used to convert some channels successively with the ADC in independent mode. With the sequencer, you can configure any sequence of **up to 16** channels successively with different sampling times and different orders. This mode is similar to the multichannel single conversion mode except that it does not stop converting after the last channel of the sequence but it restarts the conversion sequence from the first channel and continues indefinitely.
  - This mode can be used to monitor multiple voltages and temperatures in a multiple battery charger. The voltage and temperature of each battery are read during the charging process. When the voltage or the temperature reaches the maximum level, the corresponding battery should be disconnected from the charger.



Conversion Modes (6)

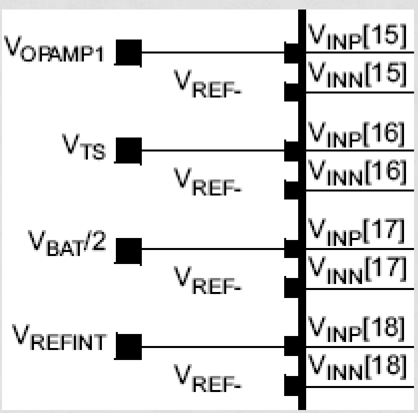
- Injected conversion mode
  - This mode is intended for use when conversion is triggered by an external event or by software.
  - The injected group has priority over the regular channel group. It interrupts the conversion of the current channel in the regular channel group.
  - This mode can be used to synchronize the conversion of channels to an event. It is interesting in motor control applications where transistor switching generates noise that impacts ADC measurements and results in wrong conversions. Using a timer, the injected conversion mode can thus be implemented to delay the ADC measurements to after the transistor switching.

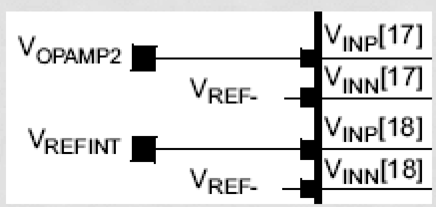


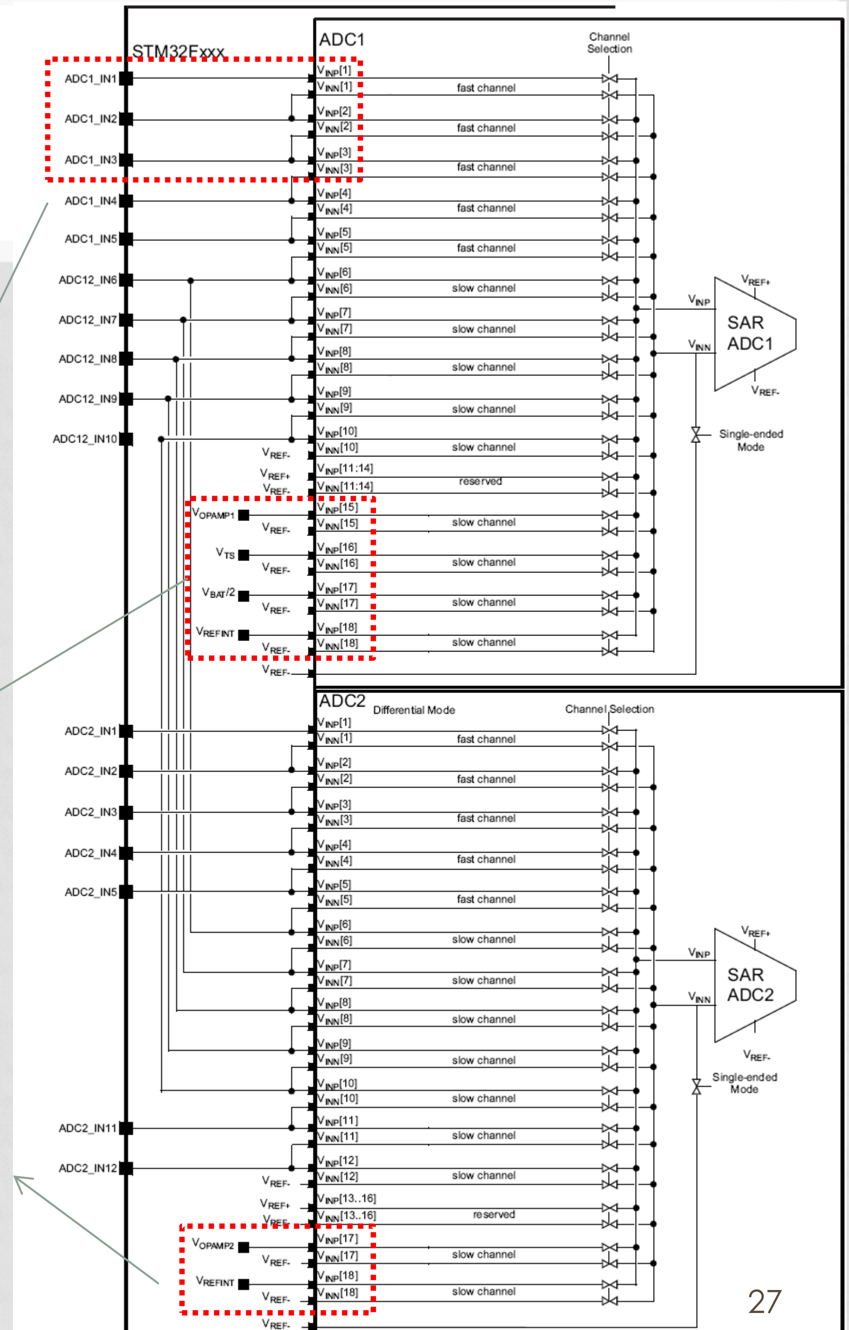
Injected conversion mode

Dual mode (1)

- ADC1 and ADC2 are tightly coupled and can operate in dual mode (ADC1 is master)



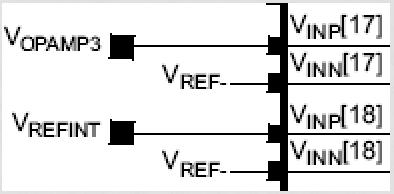


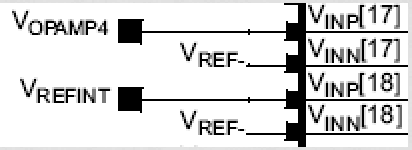


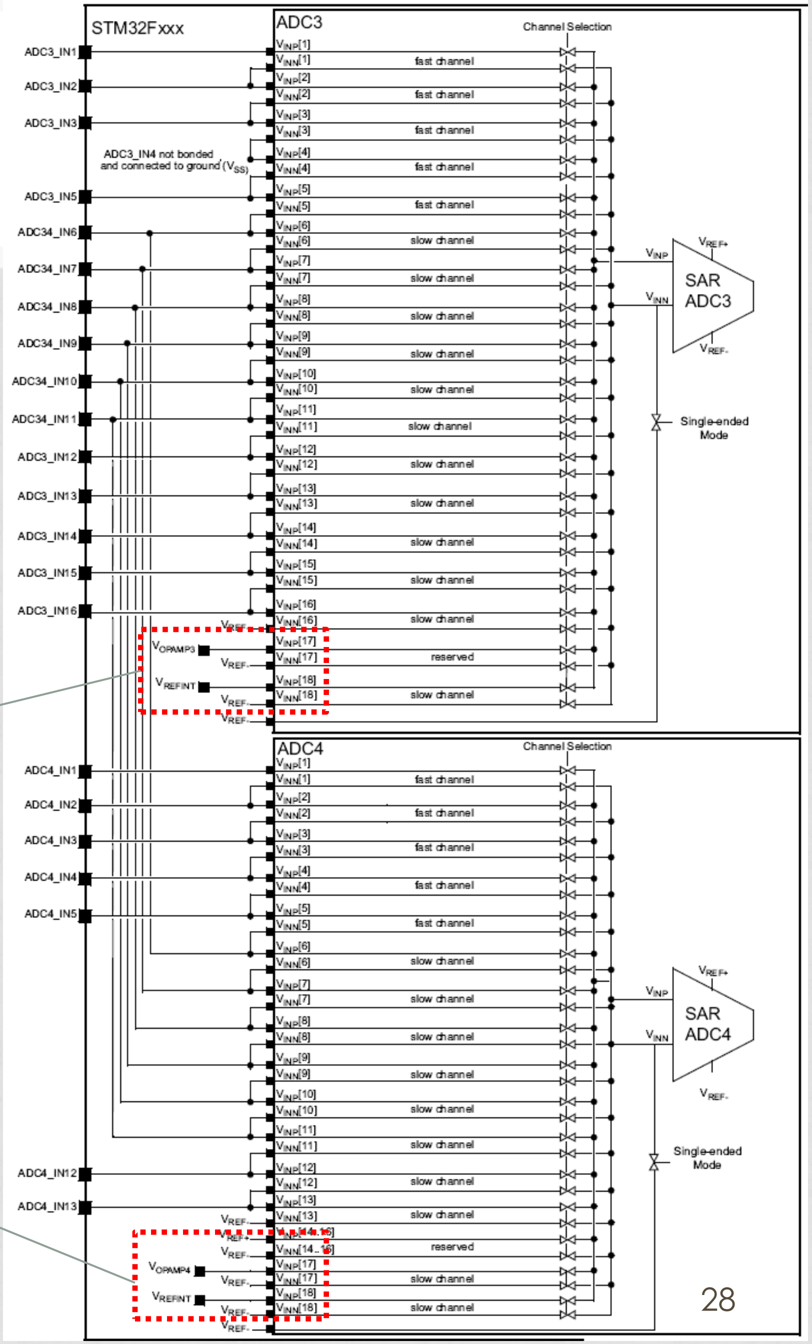
STM32F3 Microcontroller Reference Manual, page 204

Dual mode (2)

- ADC3 and ADC4 are tightly coupled and can operate in dual mode (ADC3 is master)







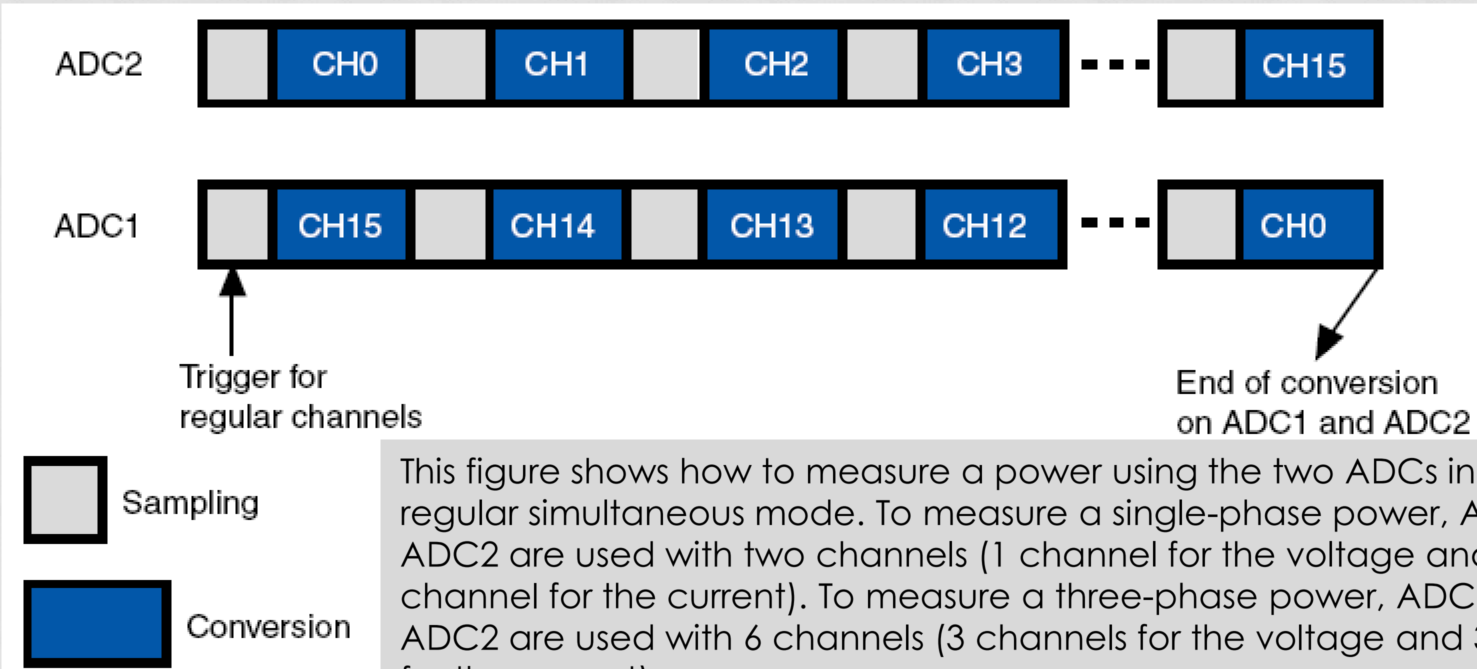
STM32F3 Microcontroller Reference Manual, page 205

Conversion Modes (7)

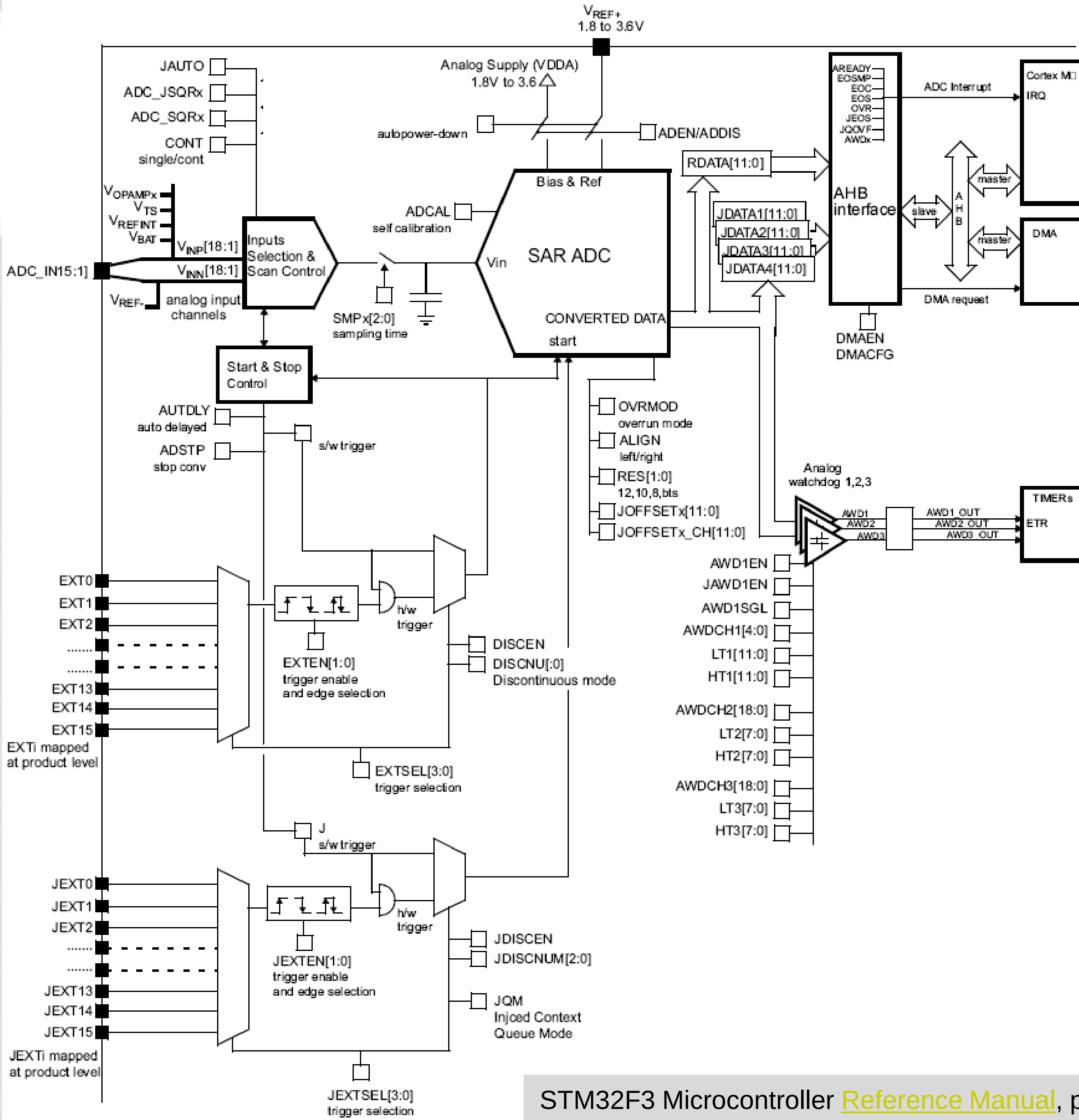
- Dual regular simultaneous mode
  - The dual regular simultaneous ADC mode is used to perform two conversions simultaneously owing to the synchronization of ADC1 and ADC2. Each ADC converts a channel sequence (with scan enabled and the sequencer of each ADC configured) or converts a single channel (scan disabled).
  - The conversion can be started with an external trigger or by software. In this mode, the conversion results of ADC1 and ADC2 are stored in ADC1's data register (32-bit format). The next figure shows how ADC1 and ADC2 convert two sequences simultaneously. ADC1 converts a sequence of 16 channels successively: channel 15 to channel 0 and ADC2 converts a sequence of 16 channels successively: channel 0 to channel 15.

Conversion Modes (8)

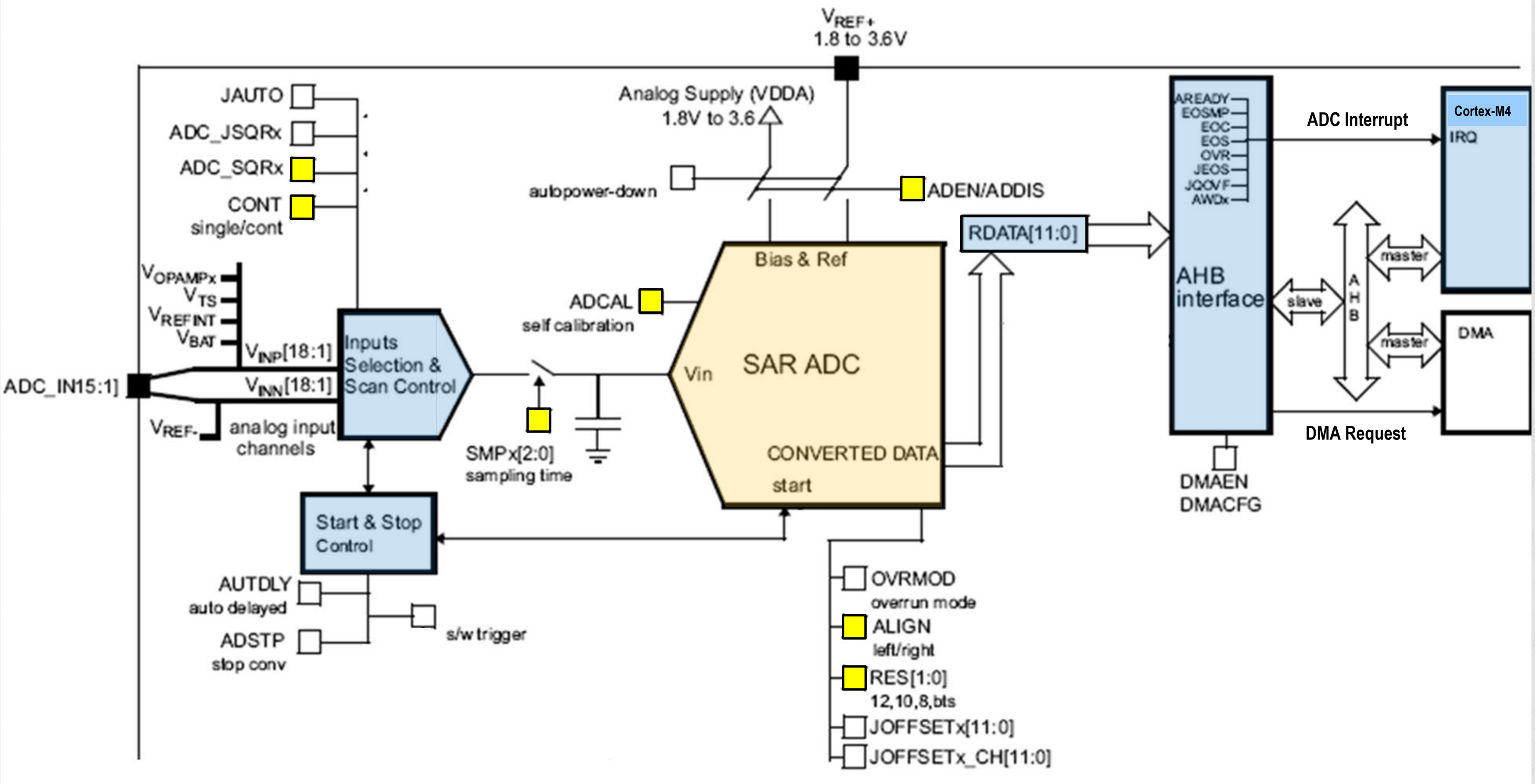
- The dual regular simultaneous mode can be used in applications where two signals should be sampled and converted at the same time. For example, to measure and plot the single phase or three-phase instantaneous electrical power: $p_n(t) = u_n(t) \times i_n(t)$.
- In this case, the voltage and current should be measured simultaneously and then the instantaneous power, which is the product of un(t) and in(t), should be computed.



This figure shows how to measure a power using the two ADCs in dual regular simultaneous mode. To measure a single-phase power, ADC1 and ADC2 are used with two channels (1 channel for the voltage and 1 channel for the current). To measure a three-phase power, ADC1 and ADC2 are used with 6 channels (3 channels for the voltage and 3 channels for the current).



STM32F3 Microcontroller Reference Manual, page 203



VREF+
1.8 to 3.6V
JAUTO
ADC_JSQRx
ADC_SQRx
CONT
single/cont
Analog Supply (VDDA)
1.8V to 3.6
autopower-down
ADEN/ADDIS
RDATA[11:0]
AREADY
EOSMP
EOC
EOS
OVR
JEOS
JQOVF
AWDx
ADC Interrupt
Cortex-M4
IRQ
master
slave
AHB
DMA
AHB interface
VOPAMPx
VTS
VREFINT
VBAT
VINP[18:1]
VINN[18:1]
ADC_IN15:1]
VREF-
analog input channels
Inputs Selection & Scan Control
ADCAL
self calibration
Bias & Ref
Vin
SAR ADC
CONVERTED DATA
start
SMPx[2:0]
sampling time
DMA Request
DMAEN
DMACFG
Start & Stop Control
AUTDLY
auto delayed
ADSTP
stop conv
s/w trigger
OVRMOD
overrun mode
ALIGN
left/right
RES[1:0]
12,10,8,bits
JOFFSETx[11:0]
JOFFSETx_CH[11:0]

ADC Channels

EXTERNAL

INTERNAL

PIN	CHANNEL	PIN	CHANNEL	PIN	CHANNEL	PIN	CHANNEL	PIN	CHANNEL	PIN	CHANNEL
PA0	ADC1_IN1			PA4	ADC2_IN1	PB1	ADC3_IN1			PE14	ADC4_IN1
PA1	ADC1_IN2			PA5	ADC2_IN2	PE9	ADC3_IN2			PE15	ADC4_IN2
PA2	ADC1_IN3			PA6	ADC2_IN3	PE13	ADC3_IN3			PB12	ADC4_IN3
PA3	ADC1_IN4			PA7	ADC2_IN4					PB14	ADC4_IN4
PF4	ADC1_IN5			PC4	ADC2_IN5	PB13	ADC3_IN5			PB15	ADC4_IN5
		PC0	ADC12_IN6					PE8	ADC34_IN6		
		PC1	ADC12_IN7					PD10	ADC34_IN7		
		PC2	ADC12_IN8					PD11	ADC34_IN8		
		PC3	ADC12_IN9					PD12	ADC34_IN9		
		PF2	ADC12_IN10					PD13	ADC34_IN10		
				PC5	ADC2_IN11			PD14	ADC34_IN11		
				PB2	ADC2_IN12	PB0	ADC3_IN12			PD8	ADC4_IN12
						PE7	ADC3_IN13			PD9	ADC4_IN13
						PE10	ADC3_IN14				
OA1	ADC1_IN15					PE11	ADC3_IN15				
TS	ADC1_IN16					PE12	ADC3_IN16				
BT/2	ADC1_IN17			OA2	ADC2_IN17	OA3	ADC3_IN17			OA4	ADC4_IN17
VRI	ADC1_IN18			VRI	ADC2_IN18	VRI	ADC3_IN18			VRI	ADC4_IN18

FAST

SLOW

ADC main registers (x=1..4)

ADCx_ISR	ADC Interrupt and Status Register
ADCx_IER	ADC Interrupt Enable Register
ADCx_CR	ADC Control Register
ADCx_CFGR	ADC ConFiGuration Register
ADCx_SMPR1..2	ADC SaMPle time Register 1..2
ADCx_SQR1..4	ADC Regular SeQuence Register 1..4
ADCx_DR	ADC Regular Data Register
ADCx_CALFACT	ADC CALibration FACTors
ADCx_DIFSEL	ADC DIFferential Mode SELection Register 2

ADCx_CR reset value: 0x2000 0000
ADVREGEN[1:0] ='10': ADC Voltage regulator disabled

Most Important ADC Bits

Register	Bits	Name	Function
ADCx_ISR	3	EOS	End of regular sequence flag
	2	EOC	End of conversion flag
	0	ADRDY	ADC ready
ADCx_CR	31	ADCAL	Start ADC calibration
	30	ADCALDIF	Differential mode for calibration
	29:28	ADVREGEN[1:0]	ADC voltage regulator enable (Reset value: ‘01’ : disabled)
	2	ADSTART	ADC start of regular conversion
	1	ADDIS	ADC disable command
	0	ADEN	ADC enable control
ADCx_CFGR	13	CONT	Single / continuous conversion mode for regular conversions
	5	ALIGN	Right/Left data alignment
	4:3	RES[1:0]	Data resolution (0:12, 1:10, 2:8, 3:6)
ADCx_SMPR1:2	29:0	SMPx[2:0]	Channel x sampling time selection (ADC clock cycles: 0:1.5; 1:2.5, 2:4.5, 3:7.5, 4:19.5; 5:61.5; 6:181.5; 7:601.5)
ADCx_SQR1:4		SQx[4:0]	x conversion in regular sequence (channel to be converted)
		L[3:0]	Regular channel sequence length (0: 1 conversions, 1: 2 conversions,…)
ADCx_DR	15:0	RDATA[15:0	Regular Data converted
ADCx_CALFACT	6:0	CALFACT_S[6:0]	Calibration factor

ADC 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	ADCx_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JQOVF	AWD3	AWD2	AWD1	JEOS	JEOC	OVR	EOS	EOC	EOSMP	ADRDY
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x04	ADCx_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JQOVFIE	AWD3IE	AWD2IE	AWD1IE	JEOSIE	JEOCIE	OVRIE	EOSIE	EOCIE	EOSMPIE	ADRDYIE
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x08	ADCx_CR	ADCAL	ADCALDIF	ADVREGEN[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JADSTP	ADSTP	JADSTART	ADSTART	ADDIS	ADEN
	Reset value	0	0	1	0																							0	0	0	0	0	0
0x0C	ADCx_CFGR	Res.	AWD1CH[4:0]					JAUTO	JAWD1EN	AWD1EN	AWD1SGL	JQM	JDISCEN	DISCNUM [2:0]			DISCEN	Res.	AUTDLY	CONT	OVRMOD	EXTEN[1:0]		EXTSEL [3:0]				ALIGN	RES [1:0]		Res.	DMACFG	DMAEN
	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
0x10	Reserved	Res.																															
0x14	ADCx_SMPR1	Res.	Res.	SMP9 [2:0]			SMP8 [2:0]			SMP7 [2:0]			SMP6 [2:0]			SMP5 [2:0]			SMP4 [2:0]			SMP3 [2:0]			SMP2 [2:0]			SMP1 [2:0]			Res.	Res.	Res.
	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			
0x18	ADCx_SMPR2	Res.	Res.	Res.	Res.	Res.	SMP18 [2:0]			SMP17 [2:0]			SMP16 [2:0]			SMP15 [2:0]			SMP14 [2:0]			SMP13 [2:0]			SMP12 [2:0]			SMP11 [2:0]			SMP10 [2:0]		
	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
0x1C	Reserved	Res.																															
0x20	ADCx_TR1	Res.	Res.	Res.	Res.	HT1[11:0]												Res.	Res.	Res.	Res.	LT1[11:0]											
	Reset value					1	1	1	1	1	1	1	1	1	1	1	1					0	0	0	0	0	0	0	0	0	0	0	0
0x24	ADCx_TR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HT2[[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LT2[7:0]							
	Reset value									1	1	1	1	1	1	1	1									0	0	0	0	0	0	0	0
0x28	ADCx_TR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HT3[[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LT3[7:0]							
	Reset value									1	1	1	1	1	1	1	1									0	0	0	0	0	0	0	0
0x2C	Reserved	Res.																															
0x30	ADCx_SQR1	Res.	Res.	Res.	SQ4[4:0]					Res.	SQ3[4:0]					Res.	SQ2[4:0]					Res.	SQ1[4:0]					Res.	Res.	L[3:0]			
	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
0x34	ADCx_SQR2	Res.	Res.	Res.	SQ9[4:0]					Res.	SQ8[4:0]					Res.	SQ7[4:0]					Res.	SQ6[4:0]					Res.	SQ5[4:0]				
	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
0x38	ADCx_SQR3	Res.	Res.	Res.	SQ14[4:0]					Res.	SQ13[4:0]					Res.	SQ12[4:0]					Res.	SQ11[4:0]					Res.	SQ10[4:0]				
	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
0x3C	ADCx_SQR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SQ16[4:0]					Res.	SQ15[4:0]				
	Reset value																						0	0	0	0	0		0	0	0	0	0
0x40	ADCx_DR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	regular RDATA[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

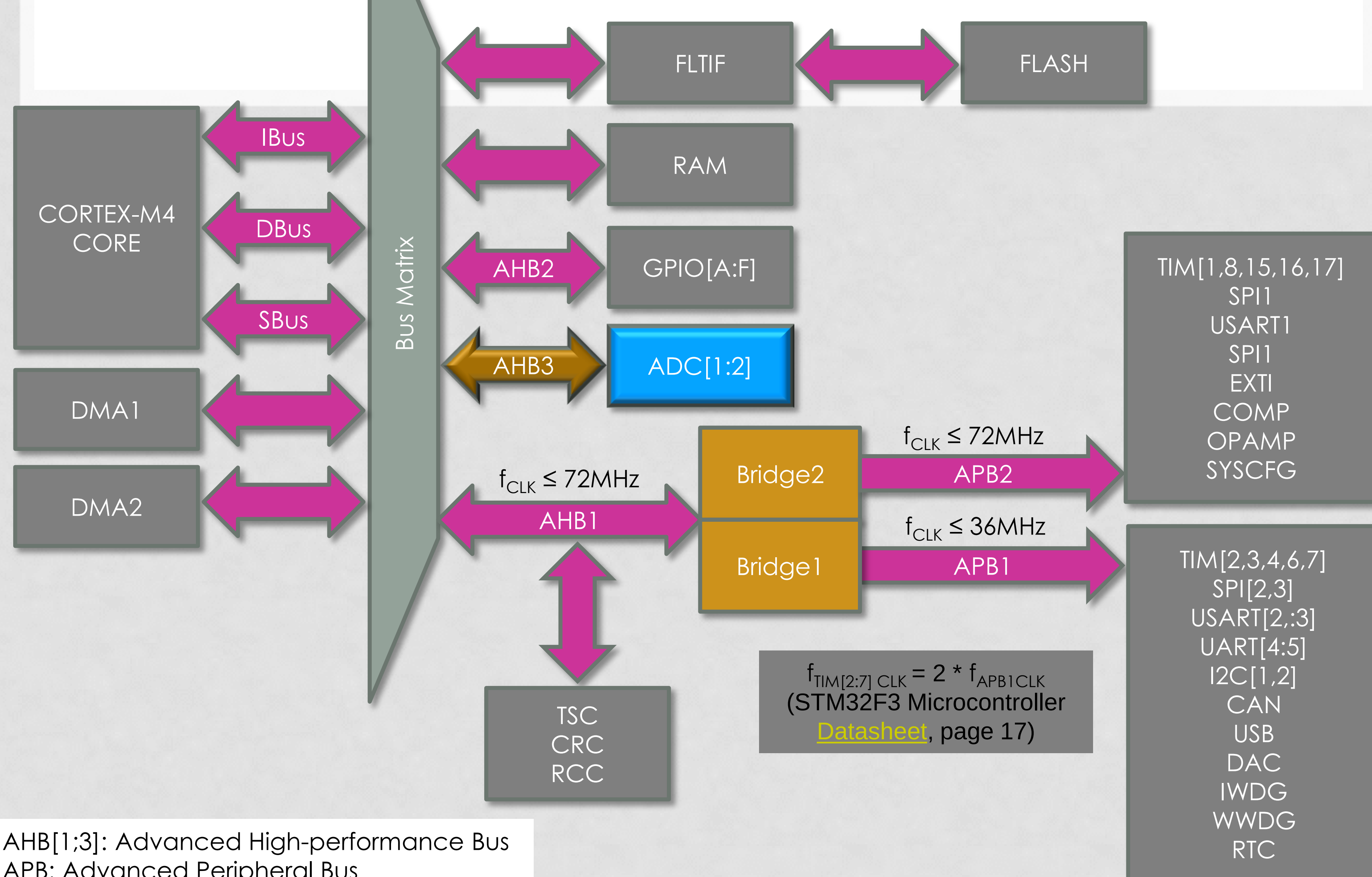
ADC Registers

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
0x44-0x48	Reserved	Res.																															
0x4C	ADCx_JSQR	Res.	JSQ4[4:0]					Res.	JSQ3[4:0]					Res.	JSQ2[4:0]					Res.	JSQ1[4:0]					JEXTEN[1:0]		JEXTSEL [3:0]				JL[1:0]	
	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
0x50-0x5C	Reserved	Res.																															
0x60	ADCx_OFR1	OFFSET1_EN	OFFSET1_ CH[4:0]					Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OFFSET1[11:0]											
	Reset value	0	0	0	0	0	0															0	0	0	0	0	0	0	0	0	0	0	0
0x64	ADCx_OFR2	OFFSET2_EN	OFFSET2_ CH[4:0]					Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OFFSET2[11:0]											
	Reset value	0	0	0	0	0	0															0	0	0	0	0	0	0	0	0	0	0	0
0x68	ADCx_OFR3	OFFSET3_EN	OFFSET3_ CH[4:0]					Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OFFSET3[11:0]											
	Reset value	0	0	0	0	0	0															0	0	0	0	0	0	0	0	0	0	0	0
0x6C	ADCx_OFR4	OFFSET4_EN	OFFSET4_ CH[4:0]					Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OFFSET4[11:0]											
	Reset value	0	0	0	0	0	0															0	0	0	0	0	0	0	0	0	0	0	0
0x70-0x7C	Reserved	Res.																															
0x80	ADCx_JDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JDATA1[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x84	ADCx_JDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JDATA2[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x88	ADCx_JDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JDATA3[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x8C	ADCx_JDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JDATA4[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x8C-0x9C	Reserved	Res.																															
0xA0	ADCx_AWD2CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWD2CH[18:1]																		Res.
	Reset value														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0xA4	ADCx_AWD3CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWD3CH[18:1]																		Res.
	Reset value														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

ADC 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
0xA8-0xAC	Reserved	Res.																															
0xB0	ADCx_DIFSEL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIFSEL[18:1]																		Reserved
	Reset value														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0xB4	ADCx_CALFACT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CALFACT_D[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CALFACT_S[6:0]						
	Reset value										0	0	0	0	0	0	0										0	0	0	0	0	0	0
0x00	ADCx_CSR	Res.	Res.	Res.	Res.	Res.	JQOVF_SLV	AWD3_SLV	AWD2_SLV	AWD1_SLV	JEOS_SLV	JEOC_SLV	OVR_SLV	EOS_SLV	EOC_SLV	EOSMP_SLV	ADRDY_SLV	Res.	Res.	Res.	Res.	Res.	JQOVF_MST	AWD3_MST	AWD2_MST	AWD1_MST	JEOS_MST	JEOC_MST	OVR_MST	EOS_MST	EOC_MST	EOSMP_MST	ADRDY_MST
							slave ADC2 or ADC4																master ADC1 or ADC3										
	Reset value							0	0	0	0	0	0	0	0	0	0							0	0	0	0	0	0	0	0	0	0
0x04	Reserved	Res.																															
0x08	ADCx_CCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VBATEN	TSEN	VREFEN	Res.	Res.	Res.	Res.	CKMODE[1:0]		MDMA[1:0]		DMACFG	Res.	DELAY[3:0]				Res.	Res.	Res.	DUAL[4:0]				
	Reset value								0	0	0					0	0	0	0	0		0	0	0	0				0	0	0	0	0
0x0C	ADCx_CDR	RDATA_SLV[15:0]																RDATA_MST[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

Bus Matrix and Busses



AHB[1;3]: Advanced High-performance Bus
APB: Advanced Peripheral Bus
RCC: Reset and Clock Control

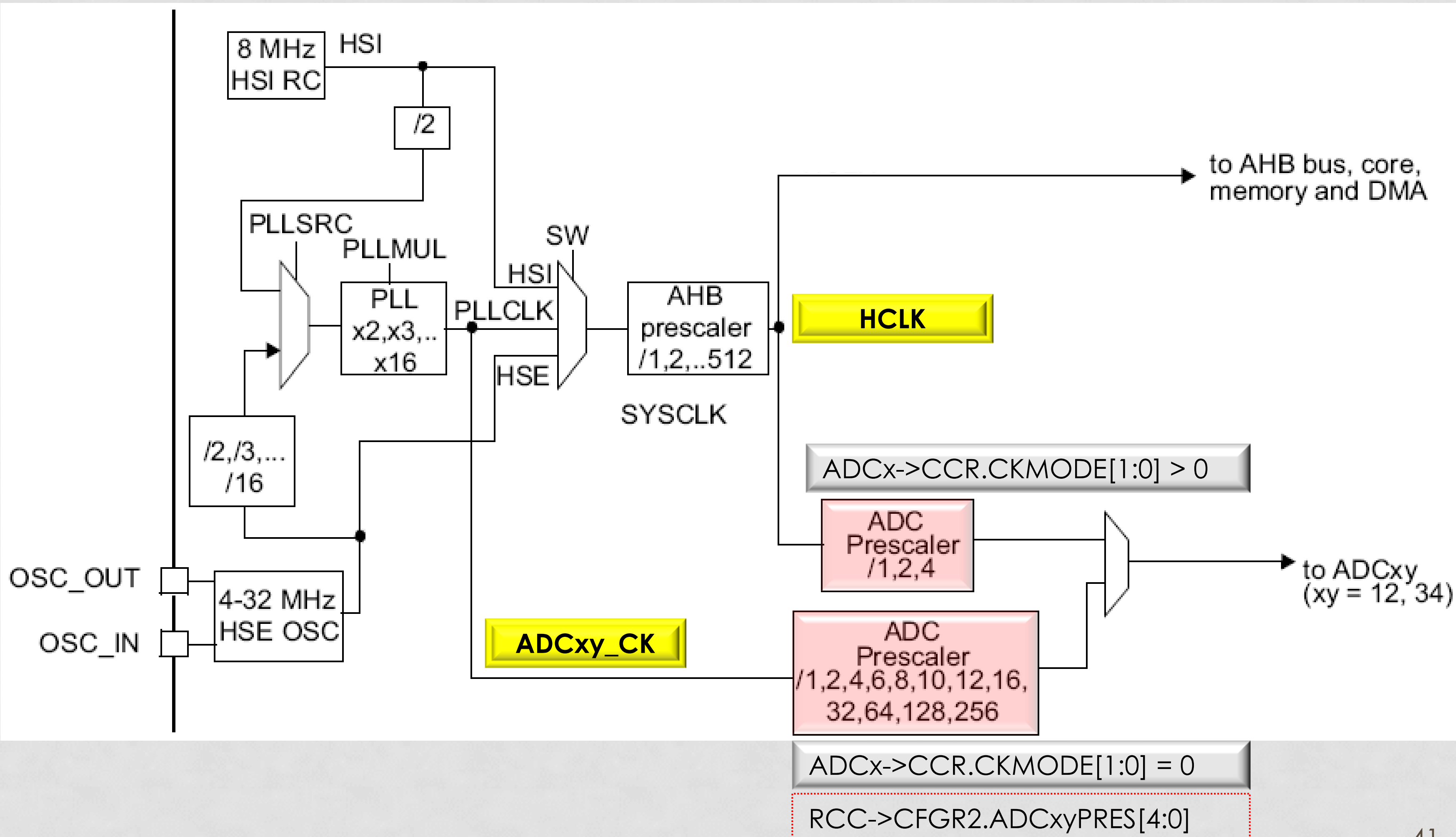
STM32F3 Microcontroller Reference Manual, pages 41-44

ADC clock

- The input clock of the two ADCs (master and slave) can be selected between two different clock sources:
  - The ADC clock can be a specific clock source, named ADCxy_CK (xy=12 or 34) which is independent and asynchronous with the AHB clock. It can be configured in the RCC_CFGR2 to deliver up to 72 MHz (PLL output).
    - To select this scheme, bits CKMODE[1:0] of the ADC_CCR register must be reset.
  - The ADC clock can be derived from the AHB clock of the ADC bus interface, divided by a programmable factor (1, 2 or 4). In this mode, a programmable divider factor can be selected (/1, 2 or 4 according to bits CKMODE[1:0]).
    - To select this scheme, bits CKMODE[1:0] of the ADC_CCR register must be different from "00".

Note: CKMODE[1:0] is valid only if the AHB prescaler is set to 1 (to achieve a clock duty cycle of 50%).

Clock tree (detail)



Clock configuration register 2 (RCC_CFGR2)

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
0x2C	RCC_CFGR2	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res.	Res.	Res.	ADC34PRES[4:0]					ADC12PRES[4:0]					PREDIV[3:0]			
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits			
13:9	ADC34PRES	Set and reset by software to control PLL clock to ADC34 division factor.	0xxxx: ADC34 clock disabled, ADC34 can use AHB clock 10000: PLL clock ÷ 1 10001: PLL clock ÷ 2 10010: PLL clock ÷ 4 …
8:4	ADC12PRES	Set and reset by software to control PLL clock to ADC12 division factor.	0xxxx: ADC12 clock disabled, ADC12 can use AHB clock 10000: PLL clock ÷ 1 10001: PLL clock ÷ 2 10010: PLL clock ÷ 4 …
3:0	PREDIV	These bits are set and cleared by software to select PREDIV1 division factor. They can be written only when the PLL is disabled.	0000: HSE input to PLL not divided 0001: HSE input to PLL ÷ 2 0010: HSE input to PLL ÷ 3 0011: HSE input to PLL ÷ 4 …

ADC voltage regulator (ADVREGEN)

- The sequence below is required to start ADC operations:
  - Enable the ADC internal voltage regulator.
  - The software must wait for the startup time of the ADC voltage regulator (TADCVREG_STUP) before launching a calibration or enabling the ADC. This temporization must be implemented by software. TADCVREG_STUP is equal to 10 μs in the worst case process/temperature/power supply.
- After ADC operations are complete, the ADC is disabled (ADEN=0).
- It is possible to save power by disabling the ADC voltage regulator.

Note: When the internal voltage regulator is disabled, the internal analog calibration is kept.

ADVREG enable sequence

- To enable the ADC voltage regulator, perform the sequence below:
  - Change ADVREGEN[1:0] bits from ‘10’ (disabled state, reset state) into ‘00’.
  - Change ADVREGEN[1:0] bits from ‘00’ into ‘01’ (enabled state).

Single-ended and differential input channels

- Channels can be configured to be either single-ended input or differential input by writing into bits DIFSEL[15:1] in the ADC_DIFSEL register. This configuration must be written while the ADC is disabled (ADEN=0). Note that DIFSEL[18:16] are fixed to single ended channels (internal channels only) and are always read as 0.
  - In single-ended input mode, the analog voltage to be converted for channel "i" is the difference between the external voltage ADC_INi (positive input) and VREF- (negative input).
  - In differential input mode, the analog voltage to be converted for channel "i" is the difference between the external voltage ADC_INi (positive input) and ADC_INi+1 (negative input).

Calibration (ADCAL, ADCALDIF, ADC_CALFACT)

- Each ADC provides an automatic calibration procedure which drives all the calibration sequence including the power-on/off sequence of the ADC.
- During the procedure, the ADC calculates a calibration factor which is 7-bits wide and which is applied internally to the ADC until the next ADC power-off.
- During the calibration procedure, the application must not use the ADC and must wait until calibration is complete.
- Calibration is preliminary to any ADC operation. It removes the offset error which may vary from chip to chip due to process or band-gap variation.
- The calibration factor to be applied for single-ended input conversions is different from the factor to be applied for differential input conversions:
  - Write ADCALDIF=0 before launching a calibration which will be applied for single-ended input conversions.
  - Write ADCALDIF=1 before launching a calibration which will be applied for differential input conversions.

Calibration (ADCAL, ADCALDIF, ADC_CALFACT)

- The calibration is then initiated by software by setting bit ADCAL=1.
- Calibration can only be initiated when the ADC is disabled (when ADEN=0).
- ADCAL bit stays at 1 during all the calibration sequence.
- It is then cleared by hardware as soon the calibration completes.
- At this time, the associated calibration factor is stored internally in the analog ADC and also in the bits CALFACT_S[6:0] or CALFACT_D[6:0] of ADC_CALFACT register (depending on single-ended or differential input calibration)
- The internal analog calibration is kept if the ADC is disabled (ADEN=0). However, if the ADC is disabled for extended periods, then it is recommended that a new calibration cycle is run before re-enabling the ADC.

Software procedure to calibrate the ADC

- Ensure that ADVREGEN[1:0]='01' and ADC voltage regulator startup time has elapsed.
- Ensure that ADEN=0.
- Select the input mode for this calibration by setting ADCALDIF=0 (Single-ended input) or ADCALDIF=1 (Differential input).
- Set ADCAL=1.
- Wait until ADCAL=0.
- The calibration factor can be read from ADC_CALFACT register.

ADC on-off control (ADEN, ADDIS, ADRDY)

- Once ADVREGEN[1:0] = '01', the ADC must be enabled and the ADC needs a stabilization time t_{STAB} before it starts converting accurately (10µs).
- Two control bits enable or disable the ADC:
  - ADEN=1 enables the ADC. The flag ADRDY will be set once the ADC is ready for operation.
  - ADDIS=1 disables the ADC.
  - ADEN and ADDIS are then automatically cleared by hardware as soon as the analog ADC is effectively enabled/disabled.
- Regular conversion can then start by setting ADSTART=1.

ADC on-off control (ADEN, ADDIS, ADRDY)

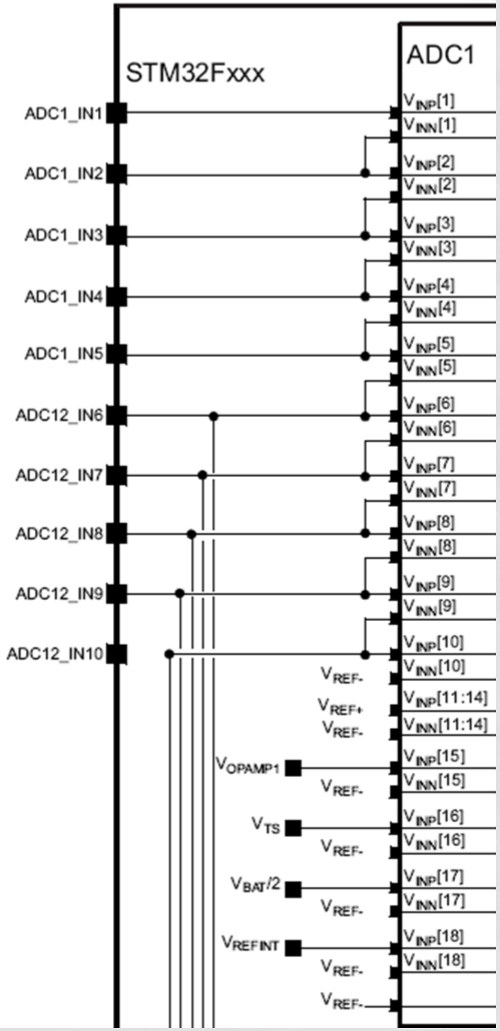
- The internal analog calibration is lost each time the power of the ADC is removed (example, when the product enters in STANDBY or VBAT mode.
- In this case, to avoid spending time recalibrating the ADC, it is possible to re-write the calibration factor into the ADC_CALFACT register without recalibrating, supposing that the software has previously saved the calibration factor delivered during the previous calibration.
- The calibration factor can be written if the ADC is enabled but not converting (ADEN=1 and ADSTART=0). Then, at the next start of conversion, the calibration factor will automatically be injected into the analog ADC.
  - This loading is transparent and does not add any cycle latency to the start of the conversion.

ADC on-off control (ADEN, ADDIS, ADRDY)

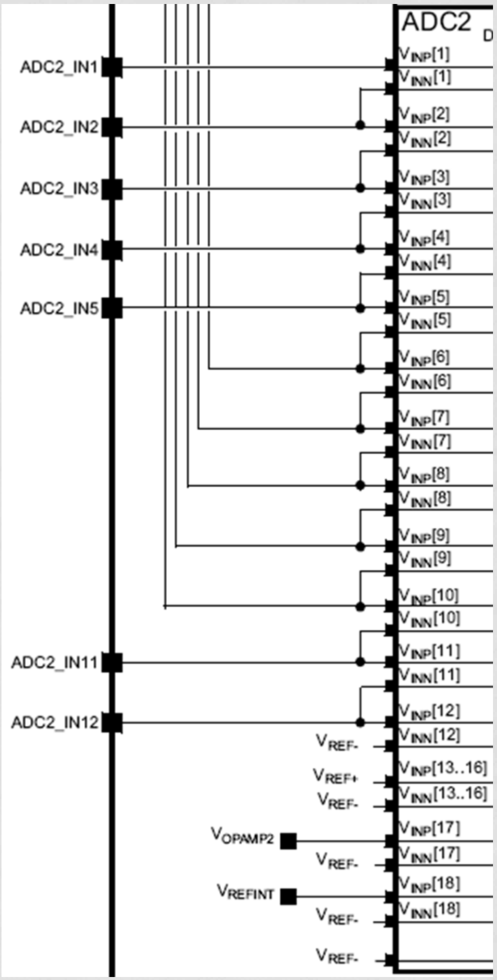
- Software procedure to enable the ADC
  - 1. Set ADEN=1.
  - 2. Wait until ADRDY=1 (ADRDY is set after the ADC startup time). This can be done using the associated interrupt (setting ADRDYIE=1).
- Software procedure to disable the ADC
  - 1. Check that ADSTART=0 to ensure that no conversion is ongoing. If required, stop any regular conversion ongoing by setting ADSTP=1 and then wait until ADSTP=0.
  - 2. Set ADDIS=1.
  - 3. If required by the application, wait until ADEN=0, until the analog ADC is effectively disabled (ADDIS will automatically be reset once ADEN=0).

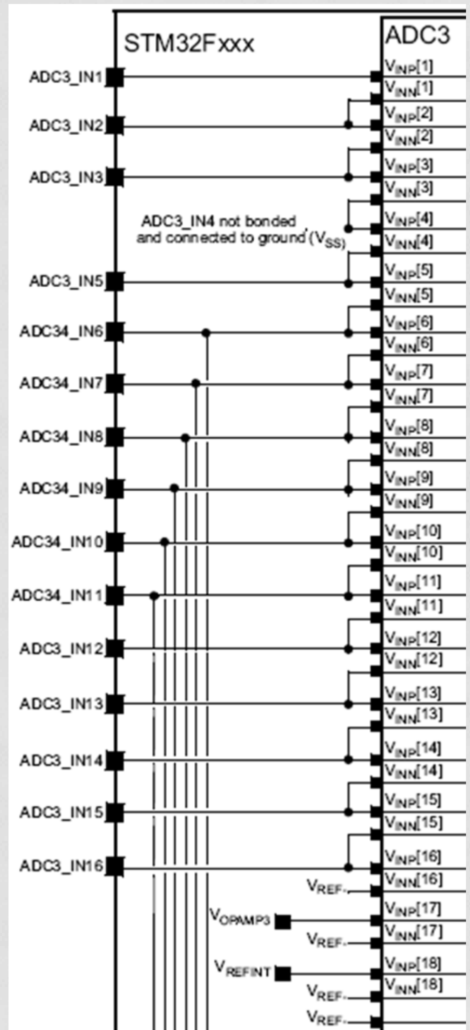
Channel selection (SQRx)

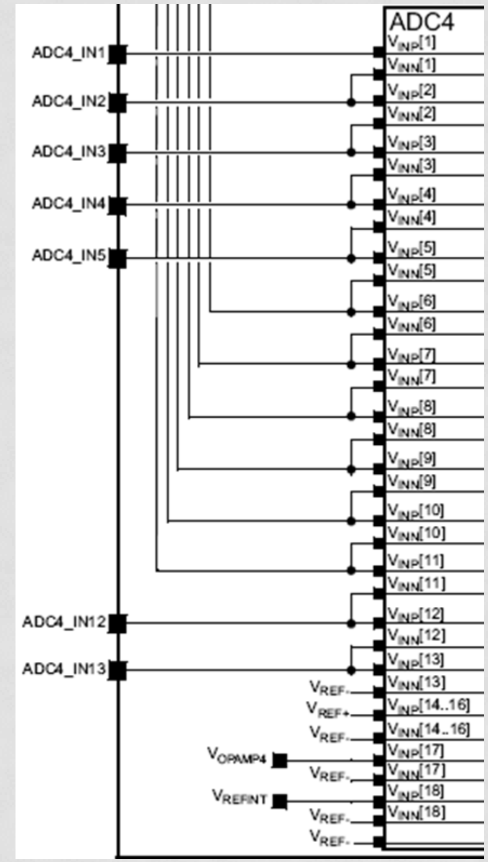
- There are up to 18 multiplexed channels per ADC:
  - 5 fast analog inputs coming from GPIO PADs (ADC_IN1..5)
  - Up to 11 slow analog inputs coming from GPIO PADs (ADC_IN5..16).
    - Depending on the products, not all of them are available on GPIO PADS.
  - ADC1 is connected to 4 internal analog inputs:
    - ADC1_IN15 = VOPAMP1 = Reference Voltage for the Operational Amplifier 1
    - ADC1_IN16 = VTS = Temperature Sensor
    - ADC1_IN17 = VBAT/2 = VBAT channel
    - ADC1_IN18 = VREFINT = Internal Reference Voltage (also connected to ADC2_IN18, ADC3_IN18 and ADC4_IN18).
  - For the other ADCs:
    - ADC_IN17 = VOPAMP2 = Reference Voltage for the Operational Amplifier 2 (ADC2)
    - ADC_IN17 = VOPAMP3 = Reference Voltage for the Operational Amplifier 3 (ADC3)
    - ADC_IN17 = VOPAMP4 = Reference Voltage for the Operational Amplifier 4 (ADC4)



Channel selection (SQRx)







Channel selection (SQRx)

- It is possible to organize the conversions in two groups: regular and injected.
- A group consists of a sequence of conversions that can be done on any channel and in any order.
  - For instance, it is possible to implement the conversion sequence in the following order: ADC_IN3, ADC_IN8, ADC_IN2, ADC_IN2, ADC_IN0, ADC_IN2, ADC_IN2, ADC_IN15.
- A regular group is composed of up to 16 conversions. The regular channels and their order in the conversion sequence must be selected in the ADC_SQRx registers.
- The total number of conversions in the regular group must be written in the L[3:0] bits in the ADC_SQR1 register.

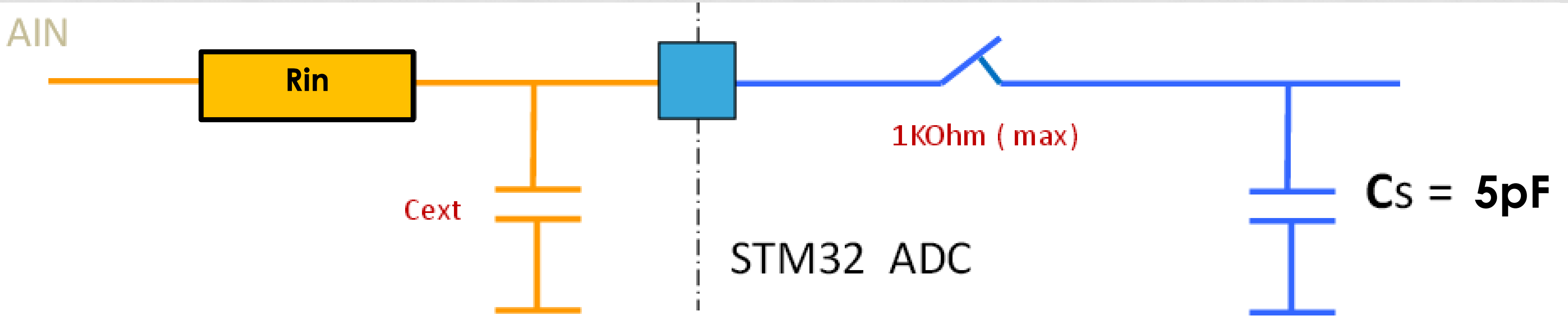
Channel-wise programmable sampling time (SMPR1, SMPR2)

- Before starting a conversion, the ADC must establish a direct connection between the voltage source under measurement and the embedded sampling capacitor of the ADC. This sampling time must be enough for the input voltage source to charge the embedded capacitor to the input voltage level.
- Each channel can be sampled with a different sampling time which is programmable using the SMP[2:0] bits in the ADC_SMPR1 and ADC_SMPR2 registers. It is therefore possible to select among the following sampling time values:

SMP	ADC clock cycles
000	1.5
001	2.5
010	4.5
011	7.5
100	19.5
101	61.5
110	181.5
111	601.5

- The total conversion time is calculated as follows (resolution = 12 bits):
  - Tconv = Sampling time + 12.5 ADC clock cycles
- Example:
  - With F_{ADC_CLK} = 72 MHz and a sampling time of 1.5 ADC clock cycles:
    - Tconv = (1.5 + 12.5) ADC clock cycles = 14 ADC clock cycles = 0.194 μs (for fast channels)

- **Cext** represents the capacitance of the PCB (dependent on soldering and PCB layout quality) plus the pad capacitance (roughly 7 pF). A high **Cext** value will downgrade conversion accuracy. To remedy this, fADC should be reduced.



- The time constant required for an RC circuit to settle to within 1/4 LSB with 12 bits resolution is :

  ∂ = time constant * ln (2^{12+2})= 9.7 time constant ~ 10 time constant

Constraints on the sampling time for fast and slow channels

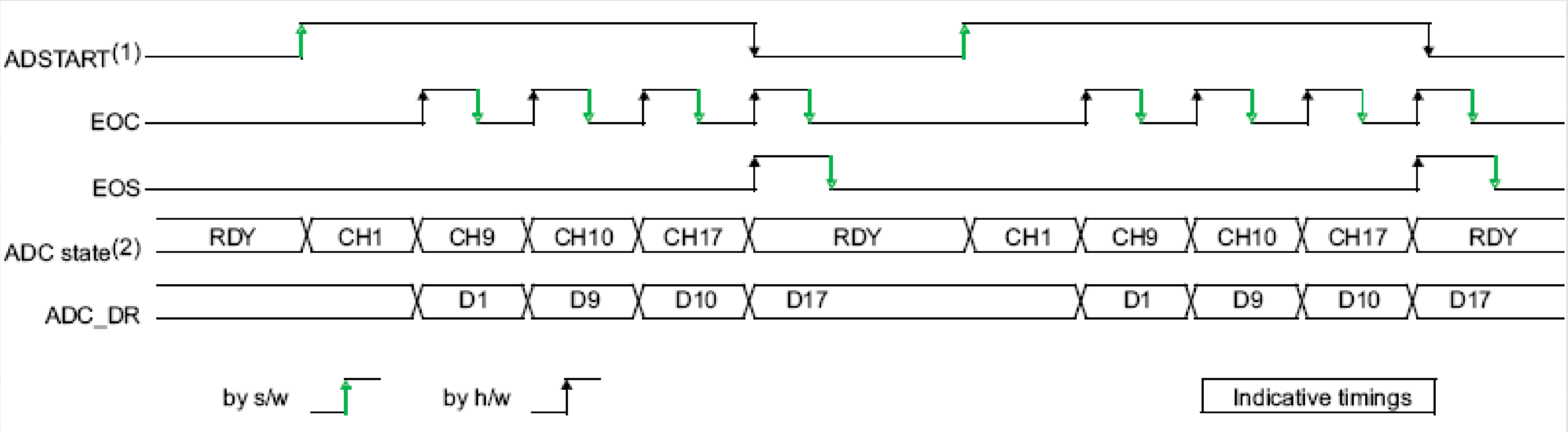
- For each channel, bits SMP[2:0] must be programmed to respect a minimum sampling time which depends on:
  - the type of channel (fast or slow)
  - the resolution
  - the output impedance of the external signal source to be converted (Rin)

Resolution	Rin (K Ohm)	Minimum sampling time (ns)	
		Fast channels	Slow channels
12-bit	0	12	17
	0.05	16	21
	0.1	20	25
	0.2	27	33
	0.5	52	58
	1	94	99
	5	430	435
	10	849	854
	20	1690	1690
	50	4190	4200
	100	8350	8350

Single conversion mode (CONT=0)

- In Single conversion mode, the ADC performs once all the conversions of the channels. This mode is started with the CONT bit at 0 by either:
  - Setting the ADSTART bit in the ADC_CR register
  - External hardware trigger event
- Inside the regular sequence, after each conversion is complete:
  - The converted data are stored into the 16-bit ADC_DR register
  - The EOC (end of regular conversion) flag is set
  - An interrupt is generated if the EOCIE bit is set
- After the regular sequence is complete:
  - The EOS (end of regular sequence) flag is set
  - An interrupt is generated if the EOSIE bit is set
- Then the ADC stops until a new external regular trigger occurs or until bit ADSTART is set again.

Single conversions of a sequence, software trigger (Timing Diagram)

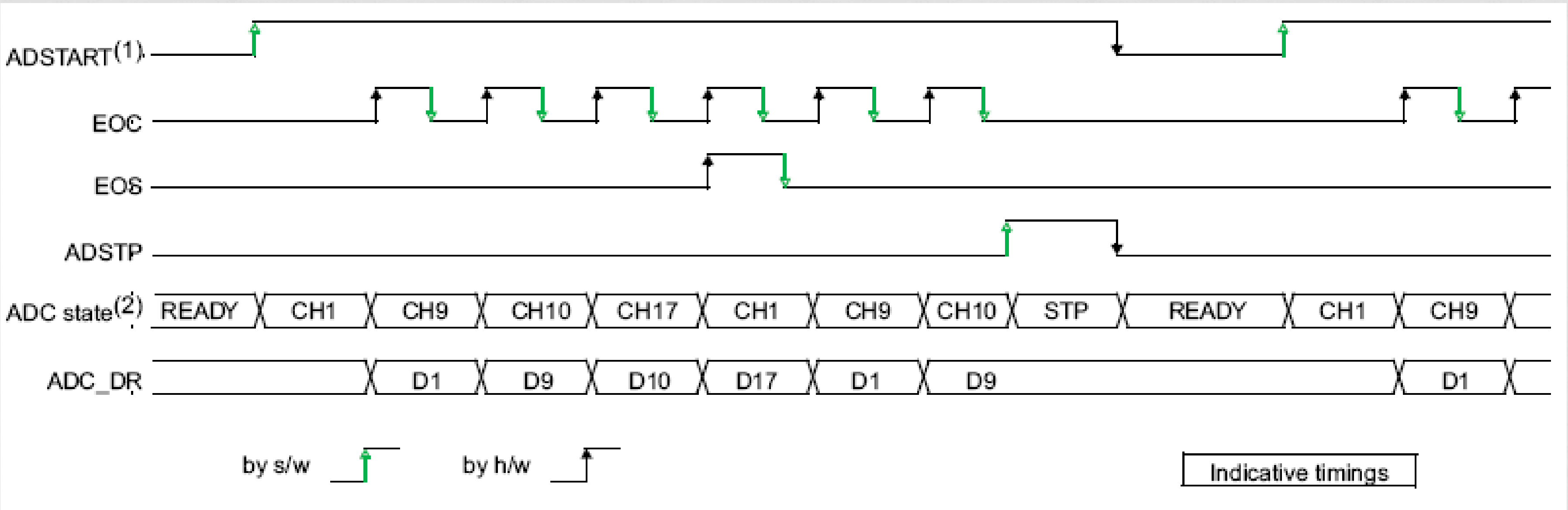


Continuous conversion mode (CONT=1)

- This mode applies to regular channels only.
- In continuous conversion mode, when a software or hardware regular trigger event occurs, the ADC performs once all the regular conversions of the channels and then automatically re-starts and continuously converts each conversions of the sequence. This mode is started with the CONT bit at 1 either by external trigger or by setting the ADSTART bit in the ADC_CR register.
- Inside the regular sequence, after each conversion is complete:
  - The converted data are stored into the 16-bit ADC_DR register
  - The EOC (end of conversion) flag is set
  - An interrupt is generated if the EOCIE bit is set
- After the sequence of conversions is complete:
  - The EOS (end of sequence) flag is set
  - An interrupt is generated if the EOSIE bit is set
- Then, a new sequence restarts immediately and the ADC continuously repeats the conversion sequence.

Note: To convert a single channel, program a sequence with a length of 1.

Continuous conversion of a sequence, software trigger (Timing Diagram)



ADC Example (1)

```
#include "stm32f30x.h"

void Delay (uint32_t nTime);

uint16_t  ADC1ConvertedValue = 0;
uint16_t  ADC1ConvertedVoltage = 0;
uint16_t  calibration_value = 0;
Volatile uint32_t TimingDelay = 0;

int main(void)
{
  // At this stage the microcontroller clock tree is already configured
  RCC->CFGR2 |= RCC_CFGR2_ADCPRE12_DIV2; // Configure the ADC clock
  RCC->AHBENR |= RCC_AHBENR_ADC12EN; // Enable ADC1 clock
  // Setup SysTick Timer for 1 µsec interrupts
  if (SysTick_Config(SystemCoreClock / 1000000))
  {
    // Capture error
    while (1)
    {}
  }
```

ADC Example (2)

```
// ADC Channel configuration  PC1 in analog mode
RCC->AHBENR |= RCC_AHBENR_GPIOCEN; // GPIOC Periph clock enable
GPIOC->MODER |= 3 << (1*2); // Configure ADC Channel7 as analog input

/* Calibration procedure */
ADC1->CR &= ~ADC_CR_ADVREGEN;
ADC1->CR |= ADC_CR_ADVREGEN_0; // 01: ADC Voltage regulator enabled
Delay(10); // Insert delay equal to 10 µs
ADC1->CR &= ~ADC_CR_ADCALDIF; // calibration in Single-ended inputs Mode.
ADC1->CR |= ADC_CR_ADCAL; // Start ADC calibration
// Read at 1 means that a calibration in progress.
while (ADC1->CR & ADC_CR_ADCAL); // wait until calibration done
calibration_value = ADC1->CALFACT; // Get Calibration Value ADC1
```

ADC Example (3)

```
// ADC configuration
ADC1->CFGR |= ADC_CFGR_CONT; // ADC_ContinuousConvMode_Enable
ADC1->CFGR &= ~ADC_CFGR_RES; // 12-bit data resolution
ADC1->CFGR &= ~ADC_CFGR_ALIGN; // Right data alignment

/* ADC1 regular channel7 configuration */
ADC1->SQR1 |= ADC_SQR1_SQ1_2 | ADC_SQR1_SQ1_1 | ADC_SQR1_SQ1_0; // SQ1 = 0x07, start converting ch7
ADC1->SQR1 &= ~ADC_SQR1_L; // ADC regular channel sequence length = 0 => 1 conversion/sequence
ADC1->SMPR1 |= ADC_SMPR1_SMP7_1 | ADC_SMPR1_SMP7_0; // = 0x03 => sampling time 7.5 ADC clock cycles
ADC1->CR |= ADC_CR_ADEN; // Enable ADC1
while(!ADC1->ISR & ADC_ISR_ADRD); // wait for ADRDY

ADC1->CR |= ADC_CR_ADSTART; // Start ADC1 Software Conversion

while (1)
{
  while(!(ADC1->ISR & ADC_ISR_EOC)); // Test EOC flag
  ADC1ConvertedValue = ADC1->DR; // Get ADC1 converted data
  ADC1ConvertedVoltage = (ADC1ConvertedValue *3300)/4096; // Compute the voltage
}
}
```

ADC Example (4)

```
void SysTick_Handler(void)
{
  TimingDelay--;
}

void Delay (uint32_t nTime)
{
  TimingDelay = nTime;
  while (TimingDelay !=0);
}
```