MICROS 32 BITS
STM - GPIO

ROBINSON JIMENEZ MORENO

LUISA FERNANDA GARCÍA VARGAS

UNIVERSIDAD MILITAR NUEVA GRANADA

INGENIERÍA MECATRÓNICA UMNG

ADC- STM32F4

Módulo ADC:

Permite la conversión de una señal analógica a digital, mediante el siguiente procedimiento:

1.- **Muestreo** de la señal analógica.
2.- **Cuantización** de la propia señal
3.- **Codificación** del resultado de la cuantización, en código binario.

a. Tasa de muestreo
b. Resolución

http://www.asifunciona.com/electronica/af_conv_ad/conv_ad_5.htm

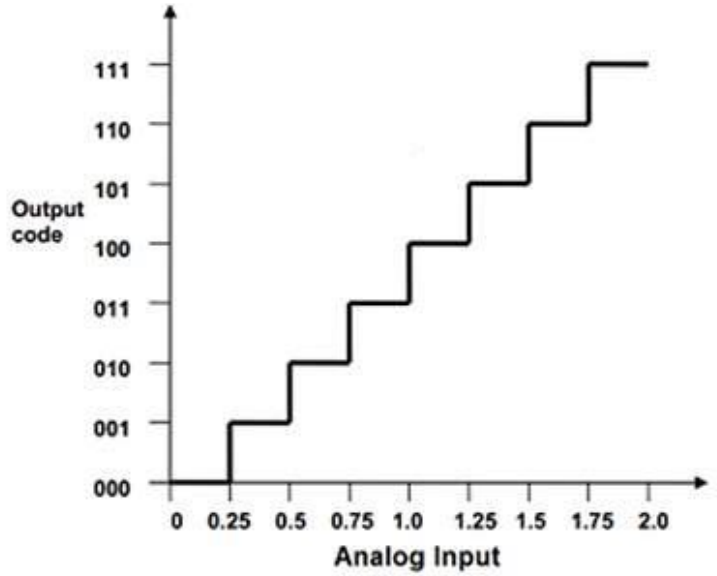
ADC- STM32F4

Cuanto mayor sea la resolución del ADC, más discretos serán los pasos que pueden ser adoptados este.

“ Un ADC tiene una salida binaria que es representativa de la tensión de alimentación. Si la tensión de alimentación es de 10 V y hay un ADC de 8 bits, entonces existe una posibilidad de 256 pasos.

Para determinar la resolución se utiliza la ecuación 2^n. El "2" es una constante y N es el número de bits que hay. Al tomar 2^8 terminamos con 256 pasos. Utilizando 256 pasos con una alimentación de 10 V significa que cada paso será de 39.0625 mV. Por lo tanto, habrá un código binario diferente para cada uno de estos pasos.

Si se tuvieran que explorar todas las posibles opciones de entrada en un ADC comenzando desde la más bajo a la más alta, se vería el perfil de una escalera. Esta escalera es la función de transferencia del ADC. La figura ilustra la transferencia mediante un ADC de 3 bits con una referencia de 2 V. “



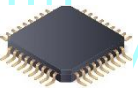
https://www.digikey.com/es/articles/techzone/2017/sep/adc-dac-tutorial

ADC- STM32F4

Módulo ADC:

La STM32F cuenta con tres canales de conversión y guarda el valor análogo en un código binario de 12 bits. Además, posee hasta 19 entradas multiplexadas que le permiten medir señales de 16 fuentes externas, dos fuentes internas, y una del canal VBAT (voltaje de batería).

La conversión ADC de los canales se puede realizar en diferentes modos: modo único, continuo, escaneo o discontinuo. El resultado de la conversión ADC se almacena en un registro de 16 bits, en donde el código de 12 bits puede ser justificado a la derecha o a la izquierda del registro.



ADC- STM32F4

MODO INDEPENDIENTE DE ADC

El ADC es configurado utilizándolo de forma independiente de otro canal, en este modo, podemos establecer otros subtipos de configuraciones posibles:

1. Mono canal (Single Channel) El ADC realiza una conversión de un canal, graba el valor resultante en el registro de salida y se detiene.
2. Individual continúo (Single-channel continuous conversion mode) Este modo es similar al primero, pero el ADC no se detiene y sigue trabajando con el canal seleccionado. El resultado se sobrescribe de forma continua con el valor más actual.

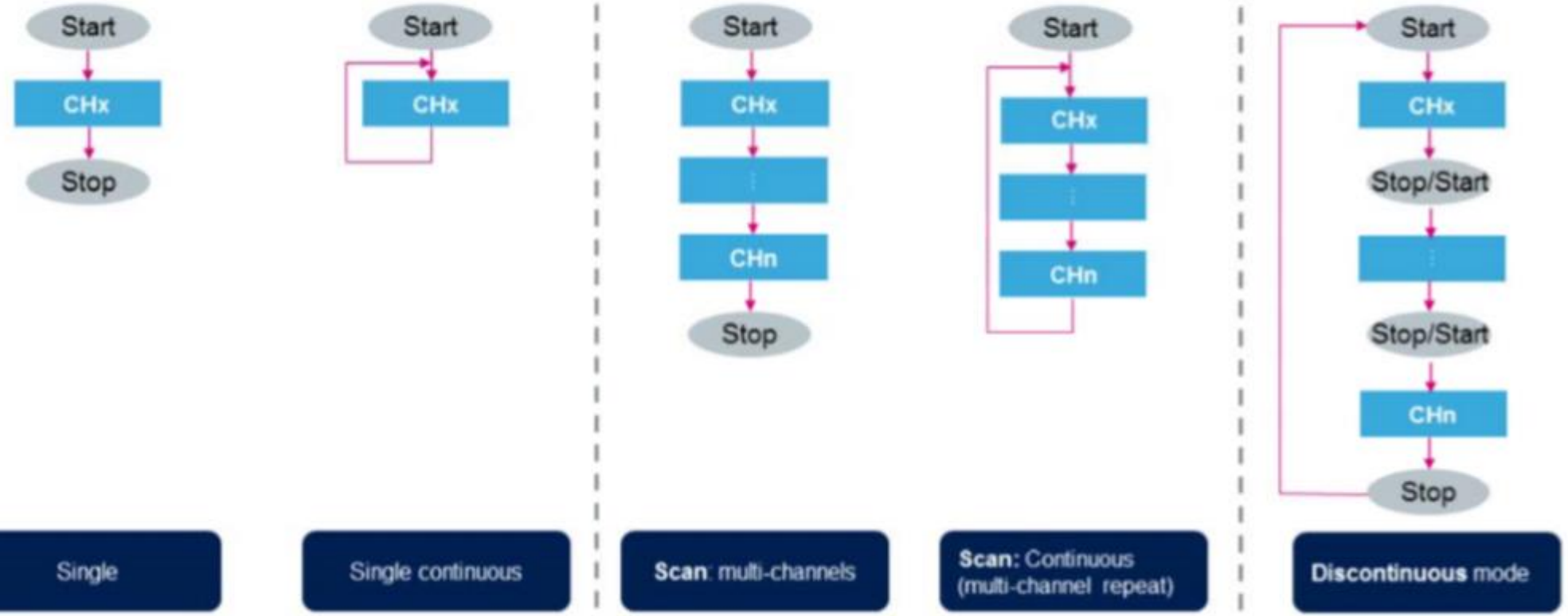
ADC- STM32F4

3. Scan (Multichannel (scan) continuous conversion mode) Este modo puede configurarse para realizar transformaciones sucesivas en múltiples canales en una secuencia dada. El tiempo de conversión se ajusta por separado para cada canal. Después de procesar el número especificado de canales, el ADC se detiene.

4. Escaneo continuo (Multichannel continuous) Igual que en el modo “Scan”, el ADC realizará conversiones en varios canales programados y una vez procesados todos, el ciclo vuelve a comenzar, sobrescribiéndose el valor final resultante cuando se reinicia el proceso.

5. Discontinuo (Intermitente) En este modo, no se escanearán todos los canales a la vez, sino solo unos pocos predefinidos. Cuando se produce el evento que desencadena la transformación, será escaneado y se pasará al siguiente grupo de canales y así sucesivamente.

ADC conversion modes



ADC- STM32F4

Módulo ADC:

Las características principales del módulo ADC son:

- Resolución configurable a 12 bits, 10 bits, 8 bits o 6 bits.
- Generación de una interrupción al final de la conversión.
- Modos de conversión único y continuo.
- Opción de activación externa con polaridad configurable
- Modo discontinuo.
- Modo Doble / Triple (en dispositivos con 2 ADCs o más).
- Retardo configurable entre conversiones en modo entrelazado Dual / Triple
- Suministro de energía al módulo ADC: 2,4 V a 3,6 V a toda velocidad y hacia abajo a 1,8 V a velocidad más lenta
- Rango de entrada ADC: VREF- ≤ ≤ VIN VREF +

ADC- STM32F4

Módulo ADC: Pines principales del módulo ADC

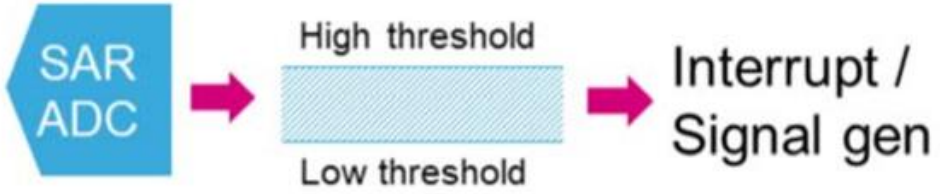
Rango de entrada ADC:
VREF- ≤ ≤ VIN VREF +

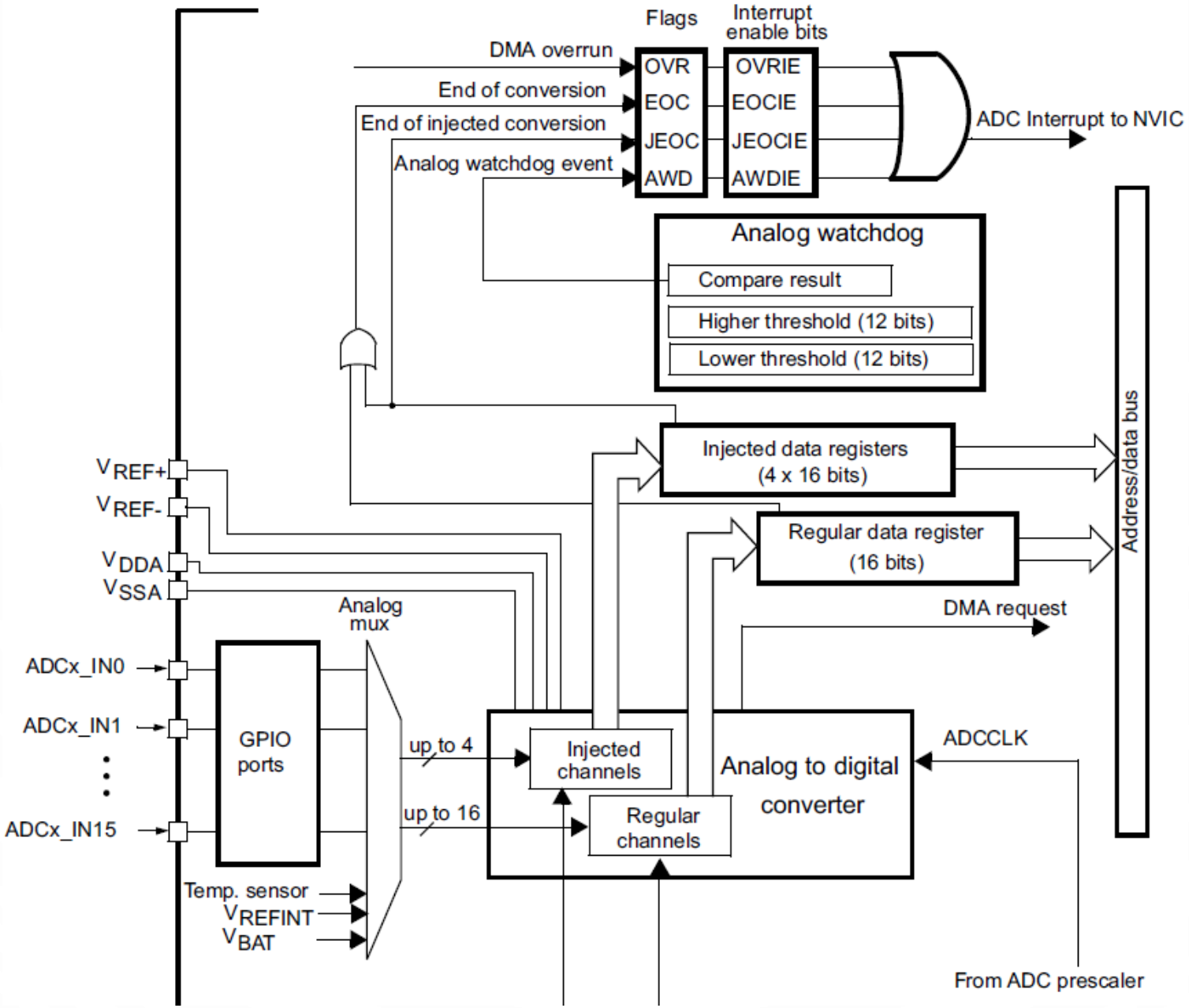
NOMBRE	TIPO DE SEÑAL	DESCRIPCION
V_{REF+}	Entrada, Referencia Análoga Positiva	El mayor voltaje (positivo) para ADC 1.8 v ≤ V_{REF+} ≤ V_{DDA}
V_{DDA}	Entrada, Suministro Análogo	Suministro de voltaje análogo igual a VDD: • 2.4 v ≤ V_{DDA} ≤ V_{DD} (3.6 v) para máxima velocidad • 1.8 v ≤ V_{DDA} ≤ V_{DD} (3.6 v) para velocidad reducida
V_{REF-}	Entrada, Referencia Análoga Negativa	El menor voltaje (negativo) para ADC 1.8 V_{REF-} = V_{SSA}
V_{SSA}	Entrada, Tierra Análoga	Tierra para suministro de voltaje análogo igual a V_{SS}
ADCX_IN[15:0]	Señales de Entrada Análogas	16 Canales Análogos de entrada

Analog watchdog

Reduced software overhead

- Each of the 3 ADCs has a window comparator
  - One 12-bit analog watchdog can monitor one selected channel or all enabled channels
- Each watchdog continuously monitors an over- and/or under-threshold condition, then generates either an interrupt or external signal or stops a timer.




Flags
Interrupt enable bits
DMA overrun
End of conversion
End of injected conversion
Analog watchdog event
OVR
EOC
JEOC
AWD
OVRIE
EOCIE
JEOCIE
AWDIE
ADC Interrupt to NVIC
Analog watchdog
Compare result
Higher threshold (12 bits)
Lower threshold (12 bits)
Injected data registers (4 x 16 bits)
Regular data register (16 bits)
Address/data bus
DMA request
VREF+
VREF-
VDDA
VSSA
Analog mux
ADCx_IN0
ADCx_IN1
ADCx_IN15
GPIO ports
up to 4
up to 16
Injected channels
Regular channels
Analog to digital converter
ADCCLK
Temp. sensor
VREFINT
VBAT
From ADC prescaler

Single conversion mode

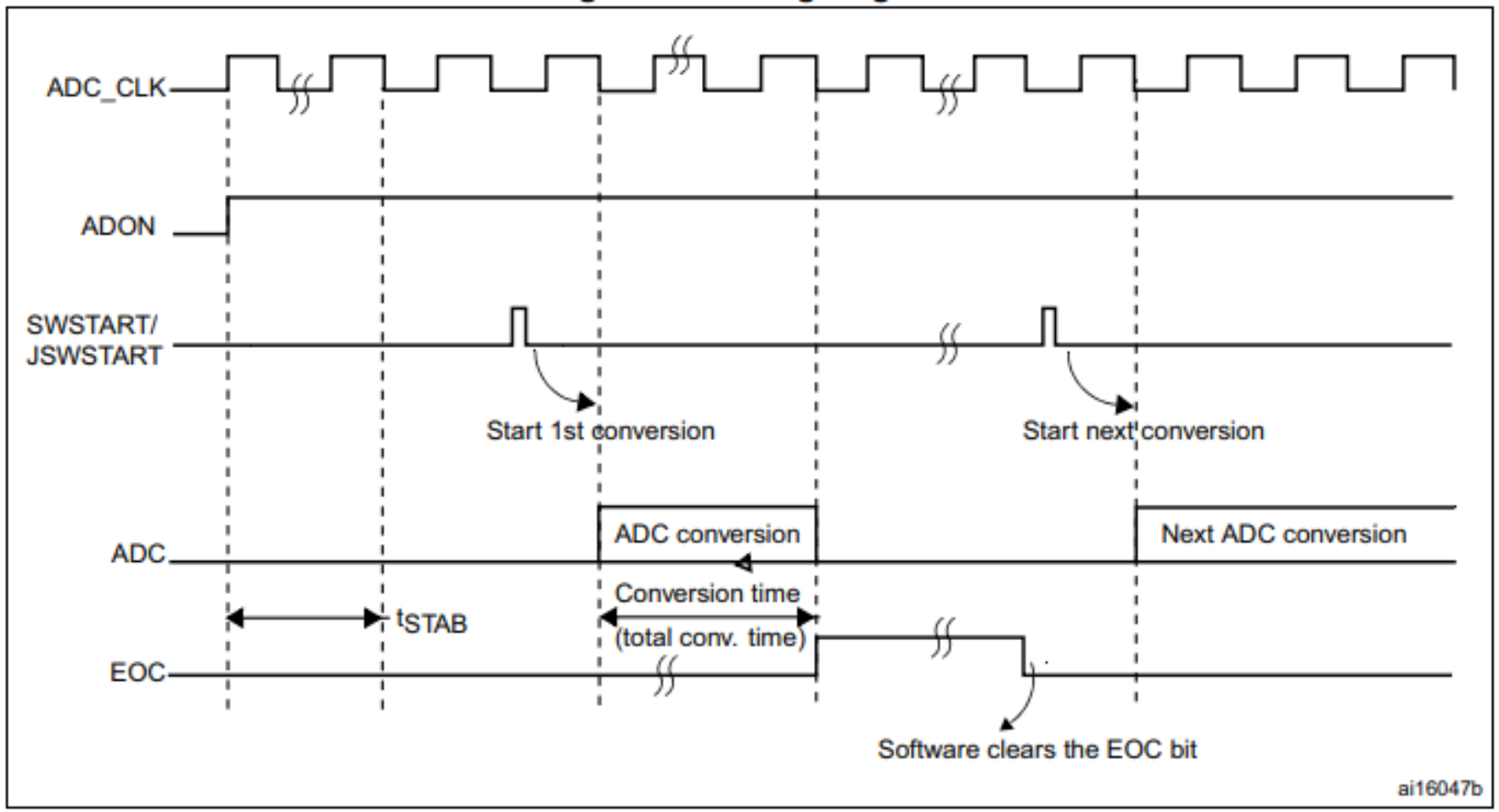
In Single conversion mode the ADC does one conversion. This mode is started with the CONT bit at 0 by either:

- setting the SWSTART bit in the ADC_CR2 register (for a regular channel only)
- setting the JSWSTART bit (for an injected channel)
- external trigger (for a regular or injected channel)

Once the conversion of the selected channel is complete:

- If a regular channel was converted:
  - 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
- If an injected channel was converted:
  - The converted data are stored into the 16-bit ADC_JDR1 register
  - The JEOC (end of conversion injected) flag is set
  - An interrupt is generated if the JEOCIE bit is set

Then the ADC stops.


ADC_CLK
ADON
SWSTART/
JSWSTART
Start 1st conversion
Start next conversion
ADC
ADC conversion
Next ADC conversion
Conversion time
(total conv. time)
tSTAB
EOC
Software clears the EOC bit
ai16047b

MULTI ADC MODE

15.9 Multi ADC mode

In devices with two ADCs or more, the Dual (with two ADCs) and Triple (with three ADCs) ADC modes can be used (see *Figure 80*).

In multi ADC mode, the start of conversion is triggered alternately or simultaneously by the ADC1 master to the ADC2 and ADC3 slaves, depending on the mode selected by the MULTI[4:0] bits in the ADC_CCR register.

Note: In multi ADC mode, when configuring conversion trigger by an external event, the application must set trigger by the master only and disable trigger by slaves to prevent spurious triggers that would start unwanted slave conversions.

The four possible modes below are implemented:

- Injected simultaneous mode
- Regular simultaneous mode
- Interleaved mode
- Alternate trigger mode

It is also possible to use the previous modes combined in the following ways:

- Injected simultaneous mode + Regular simultaneous mode
- Regular simultaneous mode + Alternate trigger mode

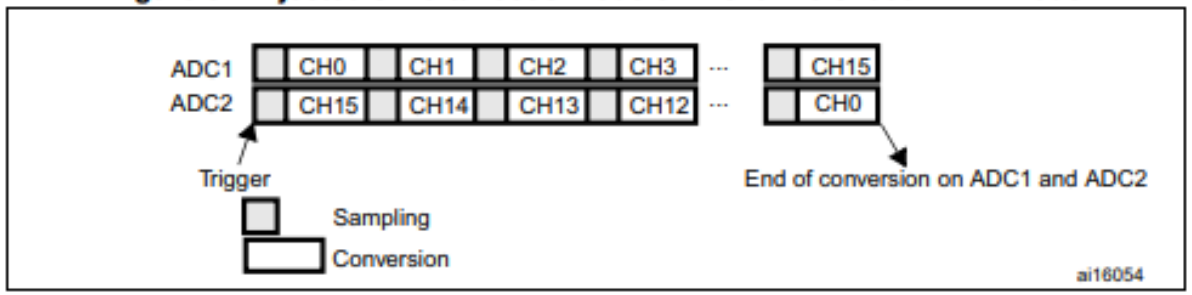
Note: In multi ADC mode, the converted data can be read on the multi-mode data register (ADC_CDR). The status bits can be read in the multi-mode status register (ADC_CSR).

Dual ADC mode

At the end of conversion event on ADC1 or ADC2:

- The converted data are stored into the ADC_JDRx registers of each ADC interface.
- A JEOC interrupt is generated (if enabled on one of the two ADC interfaces) when the ADC1/ADC2's injected channels have all been converted.

Figure 81. Injected simultaneous mode on 4 channels: dual ADC mode

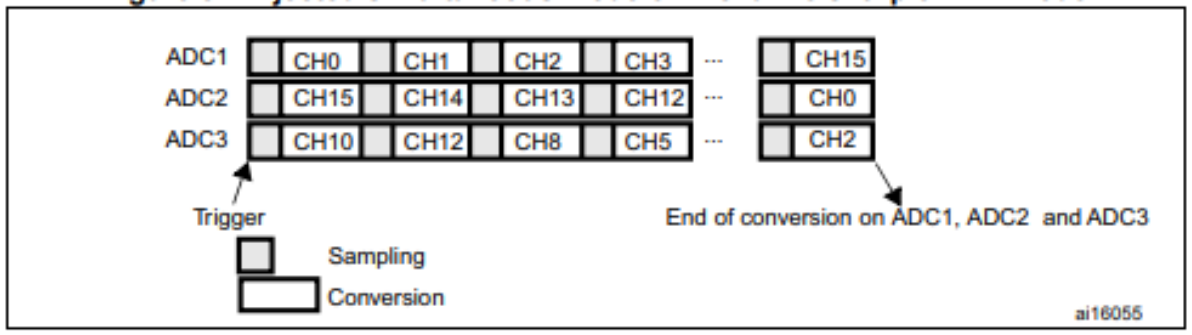


Triple ADC mode

At the end of conversion event on ADC1, ADC2 or ADC3:

- The converted data are stored into the ADC_JDRx registers of each ADC interface.
- A JEOC interrupt is generated (if enabled on one of the three ADC interfaces) when the ADC1/ADC2/ADC3's injected channels have all been converted.

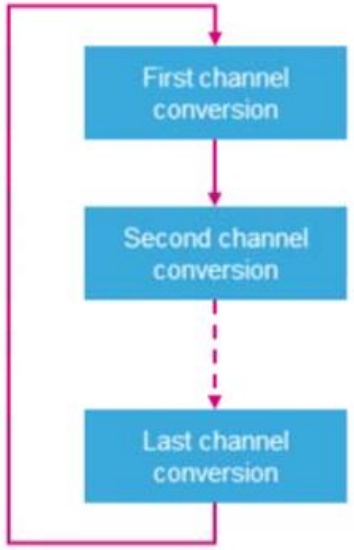
Figure 82. Injected simultaneous mode on 4 channels: triple ADC mode

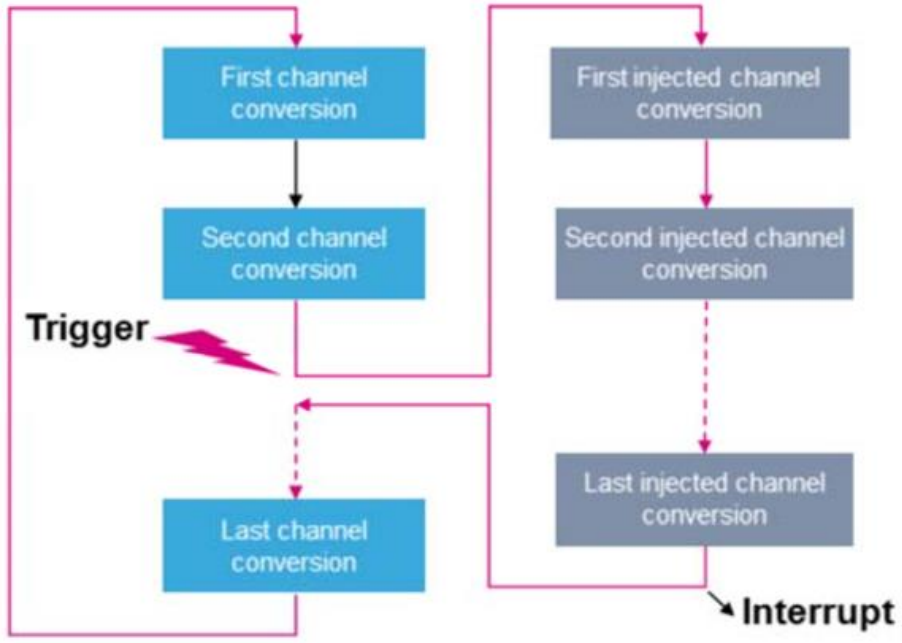


Conversion modes

Regular and injected conversions

- Regular Scan mode
- Scan mode with injected high priority trigger





life.augmented

Channel selection

There are 16 multiplexed channels. 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.
- An **injected group** is composed of up to 4 conversions. The injected channels and their order in the conversion sequence must be selected in the ADC_JSQR register. The total number of conversions in the injected group must be written in the L[1:0] bits in the ADC_JSQR register.

Ver AN3116 Application note

15.3.6 Continuous conversion mode

In continuous conversion mode, the ADC starts a new conversion as soon as it finishes one. This mode is started with the CONT bit at 1 either by external trigger or by setting the SWSTRT bit in the ADC_CR2 register (for regular channels only).

After each conversion:

- If a regular group of channels was converted:
  - The last 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

Note: Injected channels cannot be converted continuously. The only exception is when an injected channel is configured to be converted automatically after regular channels in continuous mode (using JAUTO bit), refer to Auto-injection section).

5.3.14 RCC APB2 peripheral clock enable register (RCC_APB2ENR)

Address offset: 0x44

Reset value: 0x0000 0000

Access: no wait state, word, half-word and byte access.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	LTDC EN	Res.	Res.	SAI2EN	SAI1EN	SPI6EN	SPI5EN	Res.	TIM11 EN	TIM10 EN	TIM9 EN
					rw			rw	rw	rw	rw		rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SYSCFG EN	SPI4 EN	SPI1 EN	SDMMC1 EN	ADC3 EN	ADC2 EN	ADC1 EN	Res.	Res.	USART6 EN	USART1 EN	Res.	Res.	TIM8 EN	TIM1 EN
	rw	rw	rw	rw	rw	rw	rw			rw	rw			rw	rw

ADC clock

The ADC features two clock schemes:

- Clock for the analog circuitry: ADCCLK, common to all ADCs

  This clock is generated from the APB2 clock divided by a programmable prescaler that allows the ADC to work at f_{PCLK2}/2, /4, /6 or /8. Refer to the datasheets for the maximum value of ADCCLK.

- Clock for the digital interface (used for registers read/write access)

  This clock is equal to the APB2 clock. The digital interface clock can be enabled/disabled individually for each ADC through the RCC APB2 peripheral clock enable register (RCC_APB2ENR).

1: SDMMC1 module clock enabled

Bit 10 **ADC3EN:** ADC3 clock enable
This bit is set and cleared by software.
0: ADC3 clock disabled
1: ADC3 clock enabled

Bit 9 **ADC2EN:** ADC2 clock enable
This bit is set and cleared by software.
0: ADC2 clock disabled
1: ADC2 clock enabled

Bit 8 **ADC1EN:** ADC1 clock enable
This bit is set and cleared by software.
0: ADC1 clock disabled
1: ADC1 clock enabled

ADC on-off control

The ADC is powered on by setting the ADON bit in the ADC_CR2 register. When the ADON bit is set for the first time, it wakes up the ADC from the Power-down mode.

The conversion starts when either the SWSTART or the JSWSTART bit is set.

The user can stop conversion and put the ADC in power down mode by clearing the ADON bit. In this mode the ADC consumes almost no power (only a few µA).

6.4.1 GPIO port mode register (GPIOx_MODER) (x =A..K)

Address offset:0x00

Reset values:

- 0xA800 0000 for port A
- 0x0000 0280 for port B
- 0x0000 0000 for other ports

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[1:0]	
rw	rw	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
MODER7[1:0]		MODER6[1:0]		MODER5[1:0]		MODER4[1:0]		MODER3[1:0]		MODER2[1:0]		MODER1[1:0]		MODER0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 2y+1:2y **MODERy[1:0]:** Port x configuration bits (y = 0..15)
These bits are written by software to configure the I/O mode.
00: Input mode (reset state)
01: General purpose output mode
10: Alternate function mode
11: Analog mode

15.13.3 ADC control register 2 (ADC_CR2)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	SWSTART	EXTEN		EXTSEL[3:0]				Res.	JSWSTART	JEXTEN		JEXTSEL[3:0]			
	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
Res.	Res.	Res.	Res.	ALIGN	EOCS	DDS	DMA	Res.	Res.	Res.	Res.	Res.	Res.	CONT	ADON
				rw	rw	rw	rw							rw	rw

Bit 10 **EOCS:** End of conversion selection

This bit is set and cleared by software.

0: The EOC bit is set at the end of each sequence of regular conversions. Overrun detection is enabled only if DMA=1.

1: The EOC bit is set at the end of each regular conversion. Overrun detection is enabled.

Bit 0 **ADON:** A/D Converter ON / OFF

This bit is set and cleared by software.

Note: 0: Disable ADC conversion and go to power down mode

1: Enable ADC

Bit 30 **SWSTART**: Start conversion of regular channels
This bit is set by software to start conversion and cleared by hardware as soon as the conversion starts.
0: Reset state
1: Starts conversion of regular channels
Note: This bit can be set only when ADON = 1 otherwise no conversion is launched.

Bits 29:28 **EXTEN**: External trigger enable for regular channels
These bits are set and cleared by software to select the external trigger polarity and enable the trigger of a regular group.
00: Trigger detection disabled
01: Trigger detection on the rising edge
10: Trigger detection on the falling edge
11: Trigger detection on both the rising and falling edges

Bits 27:24 **EXTSEL**[3:0]: External event select for regular group
These bits select the external event used to trigger the start of conversion of a regular group:
0000: Timer 1 CH1
0001: Timer 1 CH2
0010: Timer 1 CH3
0011: Timer 2 CH2
0100: Timer 5 TRGO
0101: Timer 4 CH4
0110: Timer 3 CH4
0111: Timer 8 TRGO
1000: Timer 8 TRGO(2)
1001: Timer 1 TRGO
1010: Timer 1 TRGO(2)
1011: Timer 2 TRGO
1100: Timer 4 TRGO
1101: Timer 6 TRGO
1110: Reserved
1111: EXTI line11

Bit 23 Reserved, must be kept at reset value.

Bit 22 **JSWSTART**: Start conversion of injected channels
This bit is set by software and cleared by hardware as soon as the conversion starts.
0: Reset state
1: Starts conversion of injected channels
This bit can be set only when ADON = 1 otherwise no conversion is launched.

Bits 21:20 **JEXTEN**: External trigger enable for injected channels
These bits are set and cleared by software to select the external trigger polarity and enable the trigger of an injected group.
00: Trigger detection disabled
01: Trigger detection on the rising edge
10: Trigger detection on the falling edge
11: Trigger detection on both the rising and falling edges

Bits 19:16 JEXTSEL[3:0]: External event select for injected group
These bits select the external event used to trigger the start of conversion of an injected group.
0000: Timer 1 TRGO
0001: Timer 1 CH4
0010: Timer 2 TRGO
0011: Timer 2 CH1
0100: Timer 3 CH4
0101: Timer4 TRGO
0110: Reserved
0111: Timer 8 CH4
1000: Timer 1 TRGO(2)
1001: Timer 8 TRGO
1010: Timer 8 TRGO(2)
1011: Timer 3 CH3
1100: Timer 5 TRGO
1101: Timer 3 CH1
1110: Timer 6 TRGO
1111: Reserved

Bit 1 **CONT**: Continuous conversion
This bit is set and cleared by software. If it is set, conversion takes place continuously until it is cleared.
0: Single conversion mode
1: Continuous conversion mode

15.13.1 ADC status register (ADC_SR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OVR	STRT	JSTRT	JEOC	EOC	AWD
										rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0

Bits 31:6 Reserved, must be kept at reset value.

Bit 5 **OVR:** Overrun

This bit is set by hardware when data are lost (either in single mode or in dual/triple mode). It is cleared by software. Overrun detection is enabled only when DMA = 1 or EOCS = 1.

0: No overrun occurred

1: Overrun has occurred

Bit 4 **STRT:** Regular channel start flag

This bit is set by hardware when regular channel conversion starts. It is cleared by software.

0: No regular channel conversion started

1: Regular channel conversion has started

Bit 3 **JSTRT:** Injected channel start flag

This bit is set by hardware when injected group conversion starts. It is cleared by software.

0: No injected group conversion started

1: Injected group conversion has started

Bit 2 **JEOC:** Injected channel end of conversion

This bit is set by hardware at the end of the conversion of all injected channels in the group. It is cleared by software.

0: Conversion is not complete

1: Conversion complete

Bit 1 **EOC:** Regular channel end of conversion

This bit is set by hardware at the end of the conversion of a regular group of channels. It is cleared by software or by reading the ADC_DR register.

0: Conversion not complete (EOCS=0), or sequence of conversions not complete (EOCS=1)

1: Conversion complete (EOCS=0), or sequence of conversions complete (EOCS=1)

Indican cuando se terminó la conversión del ADC

15.13.2 ADC control register 1 (ADC_CR1)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	OVRIE	RES		AWDEN	JAWDEN	Res.	Res.	Res.	Res.	Res.	Res.
					rw	rw	rw	rw	rw						

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DISCNUM[2:0]			JDISCEN	DISCEN	JAUTO	AWDSGL	SCAN	JEOCIE	AWDIE	EOCIE	AWDCH[4:0]				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 5 **EOCIE:** Interrupt enable for EOC

This bit is set and cleared by software to enable/disable the end of conversion interrupt.

0: EOC interrupt disabled

1: EOC interrupt enabled. An interrupt is generated when the EOC bit is set.

15.13.2 ADC control register 1 (ADC_CR1)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	OVRIE	RES		AWDEN	JAWDEN	Res.	Res.	Res.	Res.	Res.	Res.
					rw	rw	rw	rw	rw						

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DISCNUM[2:0]			JDISCEN	DISCEN	JAUTO	AWDSGL	SCAN	JEOCIE	AWDIE	EOCIE	AWDCH[4:0]				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 25:24 **RES[1:0]:** Resolution

These bits are written by software to select the resolution of the conversion.

00: 12-bit (minimum 15 ADCCLK cycles)

01: 10-bit (minimum 13 ADCCLK cycles)

10: 8-bit (minimum 11 ADCCLK cycles)

11: 6-bit (minimum 9 ADCCLK cycles)

Hexa	Decimal	Voltaje
0	0	0V
FFF	4095	3,3V

15.13.5 ADC sample time register 2 (ADC_SMPR2)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	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
SMP5_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

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:0 **SMPx[2:0]:** Channel x sampling time selection

These bits are written by software to select the sampling time individually for each channel. During sample cycles, the channel selection bits must remain unchanged.

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

15.13.4 ADC sample time register 1 (ADC_SMPR1)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	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

Bits 31: 27 Reserved, must be kept at reset value.

Bits 26:0 **SMPx[2:0]:** Channel x sampling time selection

These bits are written by software to select the sampling time individually for each channel. During sampling cycles, the channel selection bits must remain unchanged.

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

Data alignment

The ALIGN bit in the ADC_CR2 register selects the alignment of the data stored after conversion. Data can be right- or left-aligned as shown in *Figure 77* and *Figure 78*.

The converted data value from the injected group of channels is decreased by the user-defined offset written in the ADC_JOFRx registers so the result can be a negative value. The SEXT bit represents the extended sign value.

For channels in a regular group, no offset is subtracted so only twelve bits are significant.

Figure 77. Right alignment of 12-bit data

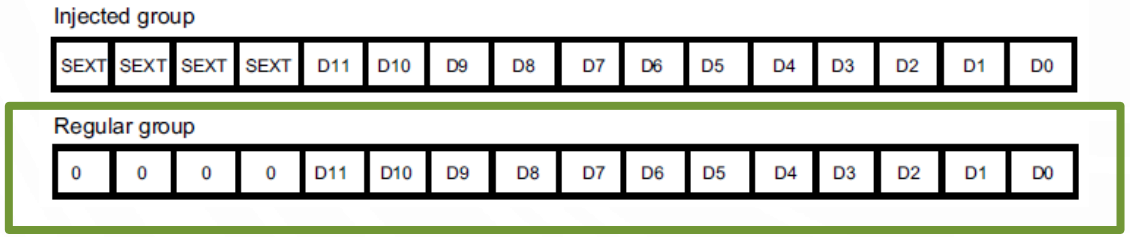
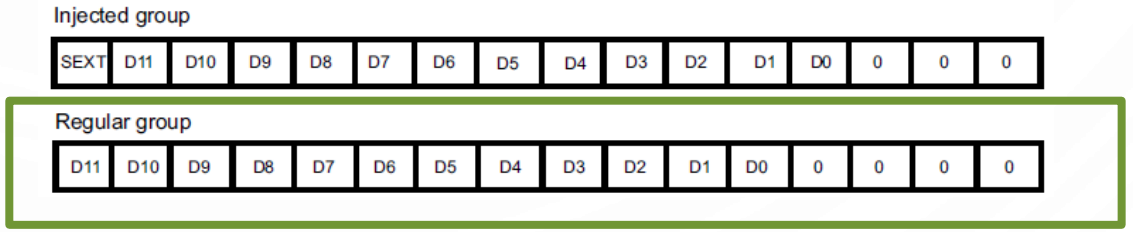


Figure 78. Left alignment of 12-bit data



15.13.14 ADC regular data register (ADC_DR)

Address offset: 0x4C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **DATA[15:0]:** Regular data

These bits are read-only. They contain the conversion result from the regular channels. The data are left- or right-aligned as shown in *Figure 77* and *Figure 78*.

15.13.9 ADC regular sequence register 1 (ADC_SQR1)

Address offset: 0x2C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:20 **L[3:0]:** Regular channel sequence length

These bits are written by software to define the total number of conversions in the regular channel conversion sequence.
0000: 1 conversion
0001: 2 conversions
...
1111: 16 conversions

Bits 19:15 **SQ16[4:0]:** 16th conversion in regular sequence

These bits are written by software with the channel number (0..18) assigned as the 16th in the conversion sequence.

Bits 14:10 **SQ15[4:0]:** 15th conversion in regular sequence

Bits 9:5 **SQ14[4:0]:** 14th conversion in regular sequence

Bits 4:0 **SQ13[4:0]:** 13th conversion in regular sequence

15.13.10 ADC regular sequence register 2 (ADC_SQR2)

Address offset: 0x30

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	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

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:26 **SQ12[4:0]:** 12th conversion in regular sequence

These bits are written by software with the channel number (0..18) assigned as the 12th in the sequence to be converted.

Bits 24:20 **SQ11[4:0]:** 11th conversion in regular sequence

Bits 19:15 **SQ10[4:0]:** 10th conversion in regular sequence

Bits 14:10 **SQ9[4:0]:** 9th conversion in regular sequence

Bits 9:5 **SQ8[4:0]:** 8th conversion in regular sequence

Bits 4:0 **SQ7[4:0]:** 7th conversion in regular sequence

15.13.11 ADC regular sequence register 3 (ADC_SQR3)

Address offset: 0x34

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	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

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:25 **SQ6[4:0]:** 6th conversion in regular sequence

These bits are written by software with the channel number (0..18) assigned as the 6th in the sequence to be converted.

Bits 24:20 **SQ5[4:0]:** 5th conversion in regular sequence

Bits 19:15 **SQ4[4:0]:** 4th conversion in regular sequence

Bits 14:10 **SQ3[4:0]:** 3rd conversion in regular sequence

Bits 9:5 **SQ2[4:0]:** 2nd conversion in regular sequence

Bits 4:0 **SQ1[4:0]:** 1st conversion in regular sequence

SELECCIÓN DEL CANAL – Tabla 10 Funciones Alternantes

A0	ADC	PA3	ADC1_IN3
A1	ADC	PC0	ADC1_IN10
A2	ADC	PC3	ADC1_IN13
A3	ADC	PC1	ADC1_IN11
A4	ADC	PC4 or PB9(1)	ADC1_IN14 (PC4) or I2C1_SDA (PB9)
A5	ADC	PC5 or PB8(1)	ADC1_IN15 (PC5) or I2C1_SCL (PB8)

```c
#include <stdio.h>
#include "stm32f7xx.h"

int main(void){

  RCC -> AHB1ENR = 0X6; //PUERTO B Y C
  RCC -> APB2ENR = 0X100; //HABILITAR EL ADC 1
//  RCC -> APB2ENR = 0X200; //HABILITAR EL ADC 2

  GPIOB -> MODER = 0X10004001; //COLOCAR EN SALIDA PARA ENCENDER LOS LEDS
  GPIOB -> OTYPER = 0X0; //PUSH PULL
  GPIOB -> OSPEEDR = 0X10004001; //VELOCIDAD MEDIA
  GPIOB -> PUPDR = 0X10004001; //PULL UP

  GPIOC -> MODER = 0X3; //ANALOGO PARA EL PIN 0

  ADC1 -> CR1 = 0X0; //RESOLUCION DE 12 BITS (POR DEFECTO)
  ADC1 -> CR2 = 0X1; //ENCENDER EL ADC
  ADC1 -> SMPR1 = 0X7FFFFF; //TIEMPO DE MUESTREO EN CICLOS
  ADC1 -> SMPR2 = 0X37777777; //TIEMPO DE MUESTREO EN CICLOS
  ADC1 -> SQR3 = 10; //CANAL 10 DEL ADC
  while(1){
    ADC1 -> CR2 |= 0X40000000; //INICIAR LA CONVERSION EN CANALES REGULARES
    if(ADC1 -> DR == 2048){
      GPIOB -> ODR = 0X4000;
    }
    if(ADC1 -> DR == 4095){
      GPIOB -> ODR = 0X1;
    }
    if(ADC1 -> DR == 0){
      GPIOB -> ODR = 0X80;
    }
  }
}
```

```c
#include <stdio.h>
#include "stm32f7xx.h"
int temp=0;
int main(void){

  RCC -> AHB1ENR |= 0X6; //PUERTO B Y C
  RCC -> APB2ENR |= 0X400; //HABILITAR EL ADC 3

  GPIOB -> MODER |= 0X10004001; //COLOCAR EN SALIDA PARA ENCENDER LOS LEDS
  GPIOB -> OTYPER = 0X0; //PUSH PULL
  GPIOB -> OSPEEDR = 0X10004001; //VELOCIDAD MEDIA
  GPIOB -> PUPDR |= 0X10004001; //PULL UP

  GPIOC -> MODER |= 15; //ANALOGO PARA EL PIN 0 y 1

  ADC3 -> CR1 |= 0X0; //RESOLUCION DE 12 BITS (POR DEFECTO)
  ADC3 -> CR2 |= 0X201; //ENCENDER EL ADC

  while(1){
    ADC3 -> SQR3 =10; //CANAL 10 DEL ADC
    ADC3 -> CR2 |= 0X40000000; //INICIAR LA CONVERSION EN CANALES REGULARES
    while((ADC3->SR &=0X2)==1);
    temp=(ADC3 -> DR)*8058/100000;
    if(temp> 50){
      GPIOB -> ODR |= 0X1;
    }
    else {GPIOB -> ODR &= 0;}
      //
  }
}
```

```c
#include <stdio.h>
#include "stm32f7xx.h"
int temp=0;
int main(void){

  RCC -> AHB1ENR |= 0X6; //PUERTO B Y C
  RCC -> APB2ENR |= 0X400; //HABILITAR EL ADC 3

  GPIOB -> MODER |= 0X10004001; //COLOCAR EN SALIDA PARA ENCENDER LOS LEDS
  GPIOB -> OTYPER = 0X0; //PUSH PULL
  GPIOB -> OSPEEDR = 0X10004001; //VELOCIDAD MEDIA
  GPIOB -> PUPDR |= 0X10004001; //PULL UP

  GPIOC -> MODER |= 15; //ANALOGO PARA EL PIN 0 y 1

  ADC3 -> CR1 |= 0X0; //RESOLUCION DE 12 BITS (POR DEFECTO)
  ADC3 -> CR2 |= 0X201; //ENCENDER EL ADC

  while(1){
    ADC3 -> SQR3 =10; //CANAL 10 DEL ADC
    ADC3 -> CR2 |= 0X40000000; //INICIAR LA CONVERSION EN CANALES REGULARES
    while((ADC3->SR &=0X2)==1);
    temp=(ADC3 -> DR)*8058/100000;
    if(temp> 50){
      GPIOB -> ODR |= 0X01;
    }
    else {GPIOB -> ODR &= 0xFFFFFFF0;}
      //
    ADC3 -> SQR3 =11; //CANAL 11 DEL ADC
    ADC3 -> CR2 |= 0X40000000; //INICIAR LA CONVERSION EN CANALES REGULARES
    while((ADC3->SR &=0X2)==1);
    temp=(ADC3 -> DR)*8058/100000;
    if(temp > 40){
      GPIOB -> ODR |= 0X80;
    }
    else {GPIOB -> ODR &= 0XFFFFFF01;}
  }
}
```

TAREA

Emplear un sensor acelerómetro para indicar la dirección de inclinación de un panel solar con capacidad de seguimiento solar, emplear una LCD para visualizar dicho movimiento, mediante barras de desplazamiento, desde el centro de la misma con pixeles hacia arriba, abajo, izquierda o derecha según la información del acelerómetro. Simular las coordenadas X y Y con dos potenciómetros.