# Embedded System and Microcomputer Principle

LAB12 Analog-to-digital Converter (ADC)

2024 Fall

wangq9@mail.sustech.edu.cn

南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

## CONTENTS

南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

01

## ADC Description

## 1. ADC Description
### -- What is ADC

- Analog-to-digital converter
- A class of equipment used to convert continuous signals in analog form into discrete signals in digital form.
- In the computer control system, it is necessary to provide the relevant parameters (such as speed, pressure, temperature, etc.) of the controlled object at any time through various detection devices and take the continuously changing voltage or current as the analog quantity.
- The input of the computer must be **digital**, so an analog-to-digital converter is needed to achieve the purpose of control.

## 1. ADC Description
## -- What is ADC(continued)

- A typical analog / digital converter converts an analog signal into a digital signal representing a certain proportional voltage value.
- The opposite device is called DAC, that is, digital / analog converter.

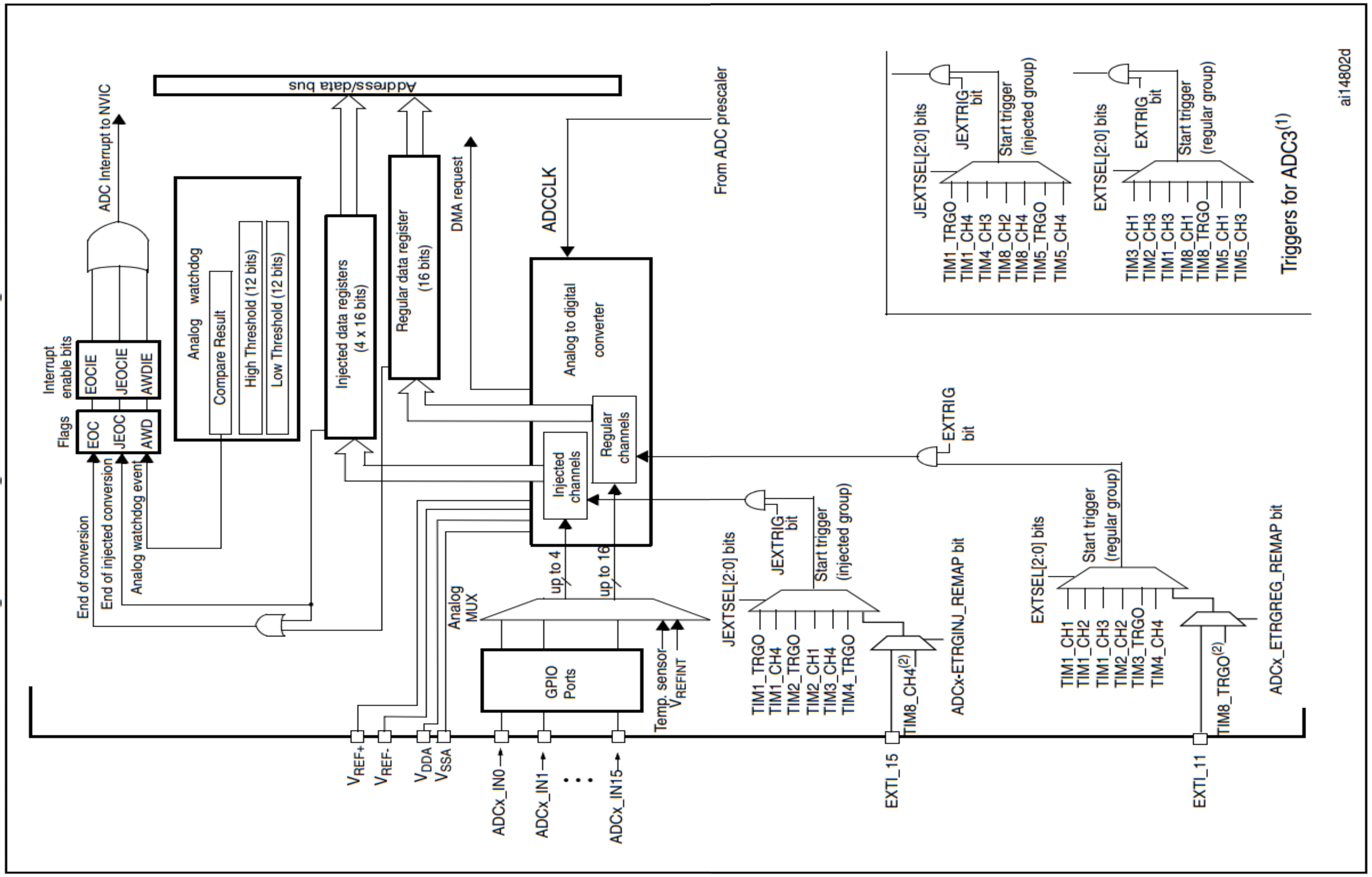
## 1. ADC Description
### -- Main features

- 12-bit resolution
- Interrupt generation at End of Conversion, End of Injected conversion and Analog watchdog event
- Single and continuous conversion modes
- Scan mode for automatic conversion of channel 0 to ch ‘n’
- Self-calibration
- Data alignment with in-built data coherency
- Channel by channel programmable sampling time
- External trigger option for both regular and injected conversion

## 1. ADC Description
### -- Main features(continued)

- Discontinuous mode
- Dual mode (on devices with 2 ADCs or more)
- ADC conversion time(STM32F103xx performance line devices)：
  - 1 µs at 56 MHz, ADCCLK=14MHz
  - 1.17 µs at 72 MHz
- ADC supply requirement： 2.4V to 3.6V
- ADC input range： $V_{REF-} \leq V_{IN} \leq V_{REF+}$
- DMA request generation during regular channel conversion
- The ADC clock provided by the clock controller is synchronized with the PCLK2 (APB2 clock)

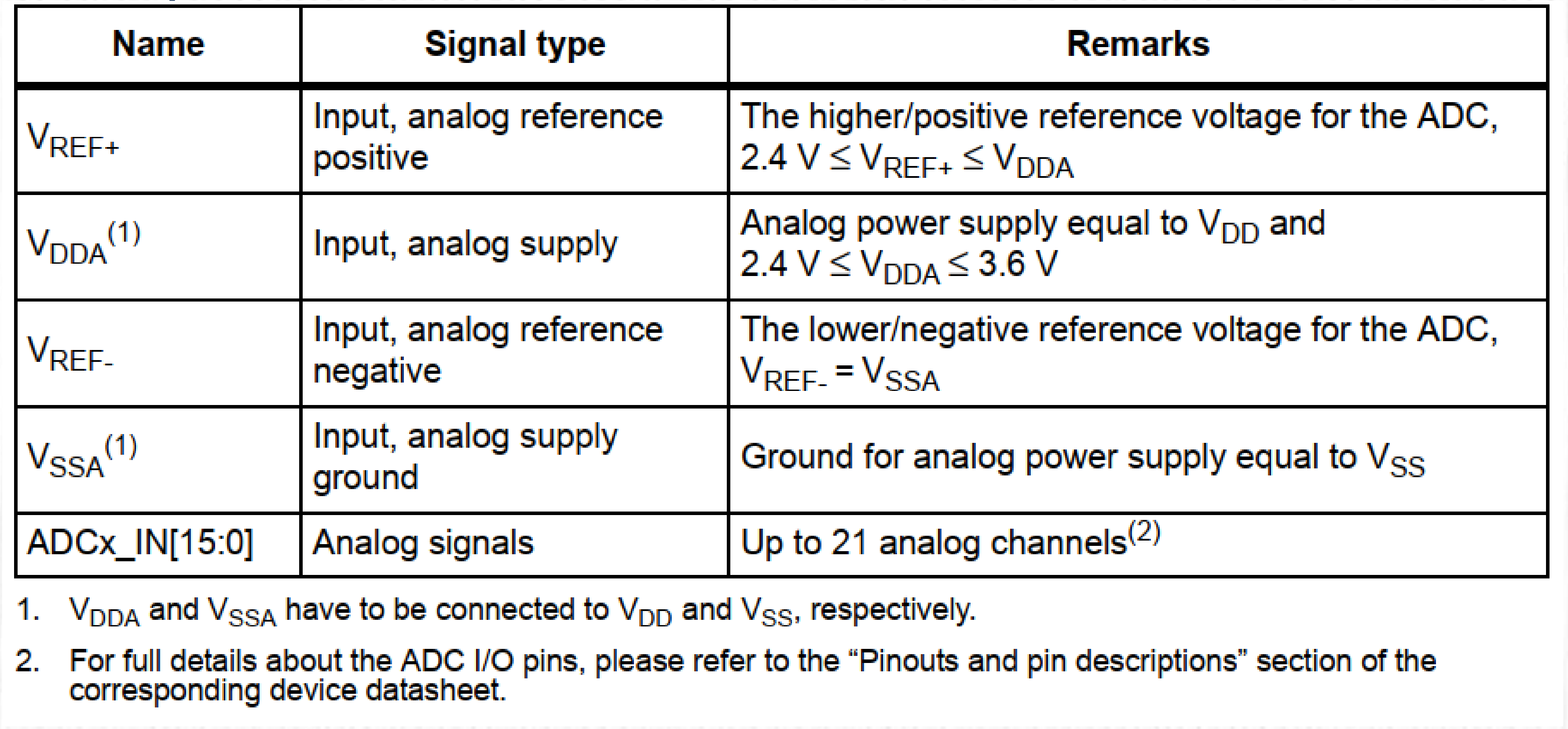
# 1. ADC Description
## -- ADC functional description



1. ADC3 has regular and injected conversion triggers different from those of ADC1 and ADC2.
2. TIM8_CH4 and TIM8_TRGO with their corresponding remap bits exist only in High-density and XL-density

## 1. ADC Description
## -- ADC functional description(continued)

### ADC pins



| Name | Signal type | Remarks |
|---|---|---|
| $V_{REF+}$ | Input, analog reference positive | The higher/positive reference voltage for the ADC, $2.4\ V \le V_{REF+} \le V_{DDA}$ |
| $V_{DDA}$(1) | Input, analog supply | Analog power supply equal to $V_{DD}$ and $2.4\ V \le V_{DDA} \le 3.6\ V$ |
| $V_{REF-}$ | Input, analog reference negative | The lower/negative reference voltage for the ADC, $V_{REF-} = V_{SSA}$ |
| $V_{SSA}$(1) | Input, analog supply ground | Ground for analog power supply equal to $V_{SS}$ |
| ADCx_IN[15:0] | Analog signals | Up to 21 analog channels(2) |

1. $V_{DDA}$ and $V_{SSA}$ have to be connected to $V_{DD}$ and $V_{SS}$, respectively.
2. For full details about the ADC I/O pins, please refer to the "Pinouts and pin descriptions" section of the corresponding device datasheet.

## 1. ADC Description

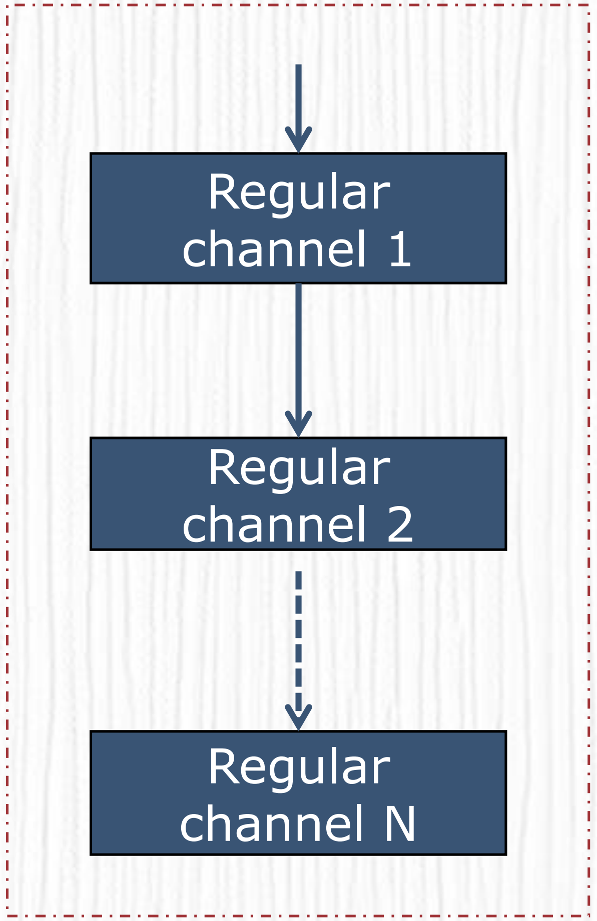
### -- Channel selection

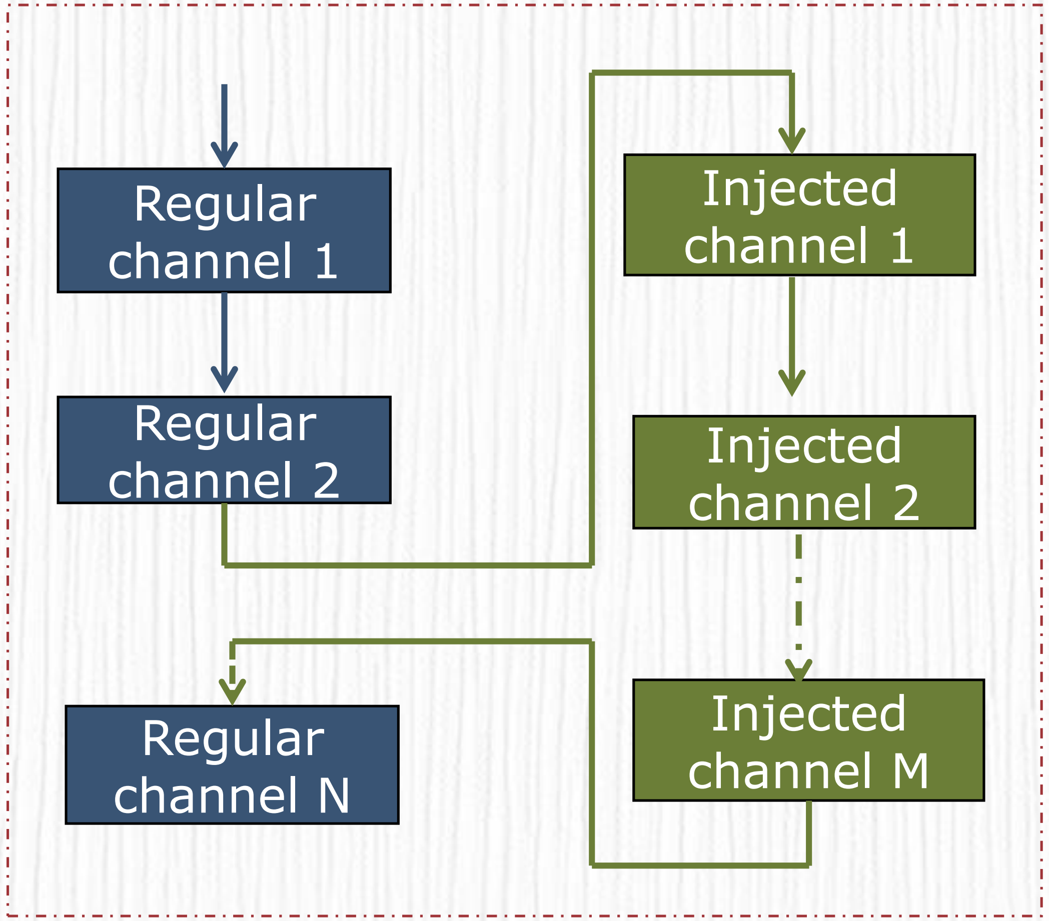
- There are 16 multiplexed channels.
- It is possible to organize the conversions in two groups: regular and injected.
- A group consists of a sequence of conversions which can be done on any channel and in any order. For instance, it is possible to do the conversion in the following order: Ch3, Ch8, Ch2, Ch2, Ch0, Ch2, Ch2, Ch15.
- The Temperature sensor is connected to channel ADCx_IN16 and the internal reference voltage $V_{REFINT}$ is connected to ADCx_IN17. These two internal channels can be selected and converted as injected or regular channels.

## 1. ADC Description
### -- Channel selection(continued)

- Regular group
  - The 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
- Injected group
  - The 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

## 1. ADC Description
## -- Channel selection(continued)





## 1. ADC Description
## -- Channel relationship of STM32F103

Relationship between ADC channel and GPIO

| Channel | ADC1 | ADC2 | ADC3 | | Channel | ADC1 | ADC2 | ADC3 |
|---|---|---|---|---|---|---|---|---|
| 0 | PA0 | PA0 | PA0 | | 9 | PB1 | PB1 | |
| 1 | PA1 | PA1 | PA1 | | 10 | PC0 | PC0 | PC0 |
| 2 | PA2 | PA2 | PA2 | | 11 | PC1 | PC1 | PC1 |
| 3 | PA3 | PA3 | PA3 | | 12 | PC2 | PC2 | PC2 |
| 4 | PA4 | PA4 | | | 13 | PC3 | PC3 | PC3 |
| 5 | PA5 | PA5 | | | 14 | PC4 | PC4 | |
| 6 | PA6 | PA6 | | | 15 | PC5 | PC5 | |
| 7 | PA7 | PA7 | | | 16 | Temperature Sensor Channel | | |
| 8 | PB0 | PB0 | | | 17 | $V_{REFINT}$ Channel | | |

## 1. ADC Description

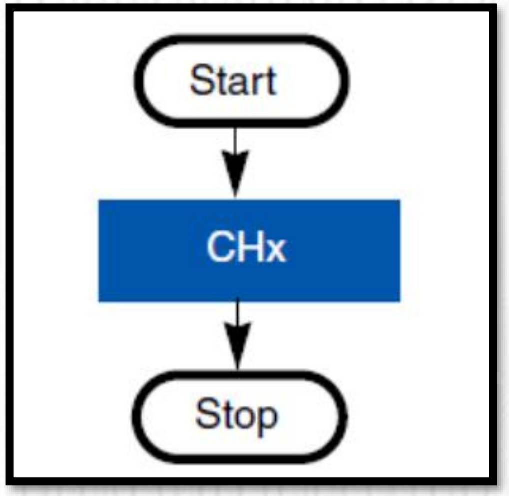
### -- Single conversion mode

- In Single conversion mode the ADC does one conversion
- If a regular channel conversion is complete:
  - The converted data is stored in the 16-bit ADC_DR register
  - The EOC (End Of Conversion) flag is set
  - and an interrupt is generated if the EOCIE is set
- If an injected channel conversion is complete:
  - The converted data is stored in the ADC_DRJ1 register
  - The JEOC (End Of Conversion Injected) flag is set
  - and an interrupt is generated if the JEOCIE bit is set
- The ADC is then stopped

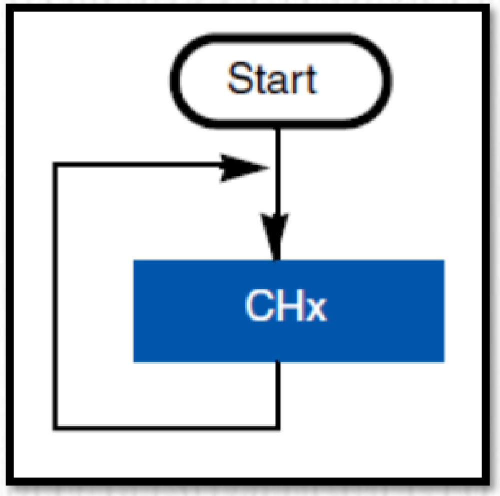
## 1. ADC Description
### -- Continuous conversion mode

- In continuous conversion mode ADC starts another conversion as soon as it finishes one
- If a regular channel conversion is complete:
  - The converted data is stored in the 16-bit ADC_DR register
  - The EOC (End Of Conversion) flag is set
  - An interrupt is generated if the EOCIE is set
- If an injected channel conversion is complete：
  - The converted data is stored in the 16-bit ADC_DRJ1 register
  - The JEOC (End Of Conversion Injected) flag is set
  - An interrupt is generated if the JEOCIE bit is set

## 1. ADC Description
## -- Single conversion VS Continuous conversion



Single conversion mode



Continuous conversion mode

## 1. ADC Description

### -- Scan mode

- This mode is used to scan a group of analog channels
- Scan mode can be selected by setting the SCAN bit in the ADC_CR1 register. Once this bit is set, ADC scans all the channels selected in the ADC_SQRx registers (for regular channels) or in the ADC_JSQR (for injected channels).
- A single conversion is performed for each channel of the group. After each end of conversion the next channel of the group is converted automatically. If the CONT bit is set, conversion does not stop at the last selected group channel but continues again from the first selected group channel.

## 1. ADC Description
### -- Scan mode(continued)

- When using scan mode, DMA bit must be set and the direct memory access controller is used to transfer the converted data of regular group channels to SRAM after each update of the ADC_DR register.
- The injected channel converted data is always stored in the ADC_JDRx registers

## 1. ADC Description

### -- Discontinuous mode(regular group)

- This mode is enabled by setting the DISCEN bit in the ADC_CR1 register.
- It can be used to convert a short sequence of n conversions (n <=8) which is a part of the sequence of conversions selected in the ADC_SQRx registers.
- The value of n is specified by writing to the DISCNUM[2:0] bits in the ADC_CR1 register.
- When an external trigger occurs, it starts the next n conversions selected in the ADC_SQRx registers until all the conversions in the sequence are done.
- The total sequence length is defined by the L[3:0] bits in the ADC_SQR1 register.

## 1. ADC Description
## -- Discontinuous mode(regular group)

- Example:
  - n = 3, channels to be converted = 0, 1, 2, 3, 6, 7, 9, 10
  - 1st trigger: sequence converted 0, 1, 2. An EOC event is generated at each conversion
  - 2nd trigger: sequence converted 3, 6, 7. An EOC event is generated at each conversion
  - 3rd trigger: sequence converted 9, 10. An EOC event is generated at each conversion
  - 4th trigger: sequence converted 0, 1, 2. An EOC event is generated at each conversion

## 1. ADC Description

### -- Discontinuous mode(injected group)

- This mode is enabled by setting the JDISCEN bit in the ADC_CR1 register.
- It can be used to convert the sequence selected in the ADC_JSQR register, channel by channel, after an external trigger event.
- When an external trigger occurs, it starts the next channel conversions selected in the ADC_JSQR registers until all the conversions in the sequence are done.
- The total sequence length is defined by the JL[1:0] bits in the ADC_JSQR register.

## 1. ADC Description

## -- Discontinuous mode(injected group)

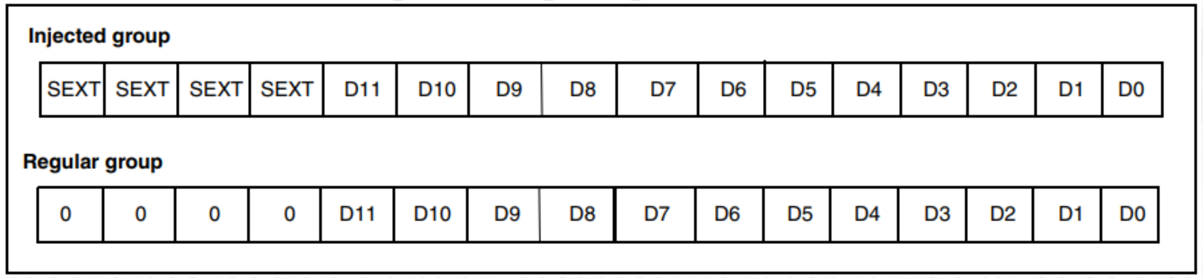
- Example:
  - n = 1, channels to be converted = 1, 2, 3
  - 1st trigger: channel 1 converted
  - 2nd trigger: channel 2 converted
  - 3rd trigger: channel 3 converted and EOC and JEOC events generated
  - 4th trigger: channel 1

## 1. ADC Description

### -- Data alignment

- Data can be left or right aligned
- The injected group channels converted data value is decreased by the user-defined offset written in the ADC_JOFRx registers so the result can be a negative value.
- The SEXT bit is the extended sign value.
- For regular group channels no offset is subtracted so only twelve bits are significant.

Right alignment of data



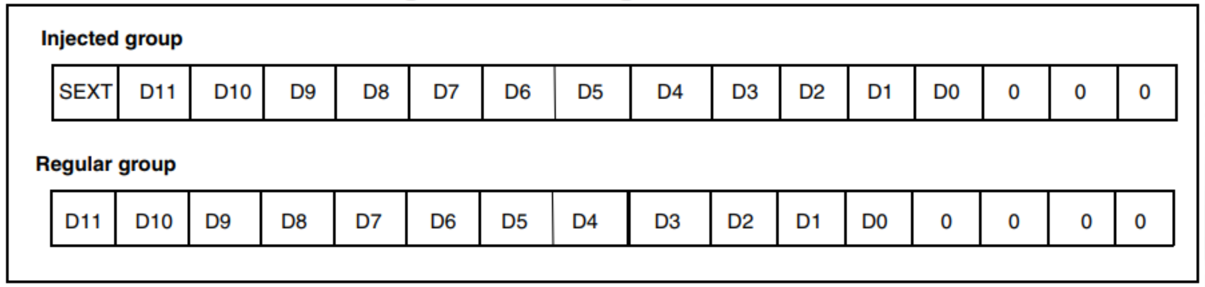
**Injected group**

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SEXT | SEXT | SEXT | SEXT | D11 | D10 | D9 | D8 | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |

**Regular group**

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | D11 | D10 | D9 | D8 | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |

Left alignment of data



**Injected group**

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SEXT | D11 | D10 | D9 | D8 | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | 0 | 0 | 0 |

**Regular group**

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| D11 | D10 | D9 | D8 | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | 0 | 0 | 0 | 0 |

## 1. ADC Description
### -- ADC sample time

- Channel-by-channel programmable sample time
- ADC samples the input voltage for a number of ADC_CLK cycles which can be modified using the SMP[2:0] bits in the ADC_SMPR1 and ADC_SMPR2 registers.
- Each channel can be sampled with a different sample time.
- The total conversion time is calculated as follows:
  $T_{CONV}$ = Sampling time + 12.5 cycles
- Example：
  With an ADCCLK = 14MHz and a sampling time of 1.5 cycles
  $T_{CONV}$ = 1.5 + 12.5 = 14 cycles = 1μs

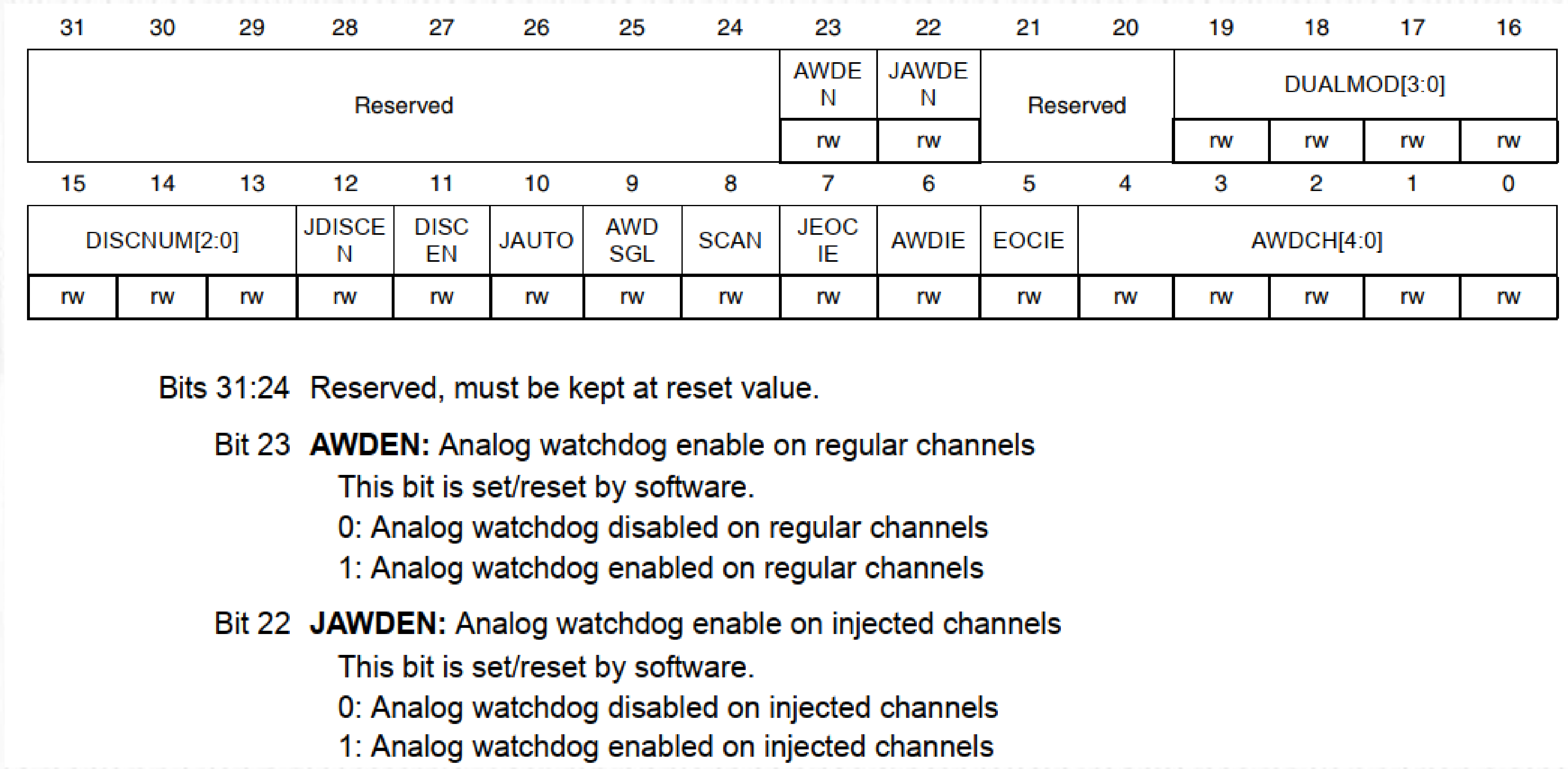
南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

02

## ADC Registers

# 2. ADC Registers
## -- ADC_CR1

- ADC control register1



| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | AWDEN | JAWDEN | Reserved | | DUALMOD[3:0] | | | |
| | | | | | | | | rw | rw | | | rw | rw | rw | rw |

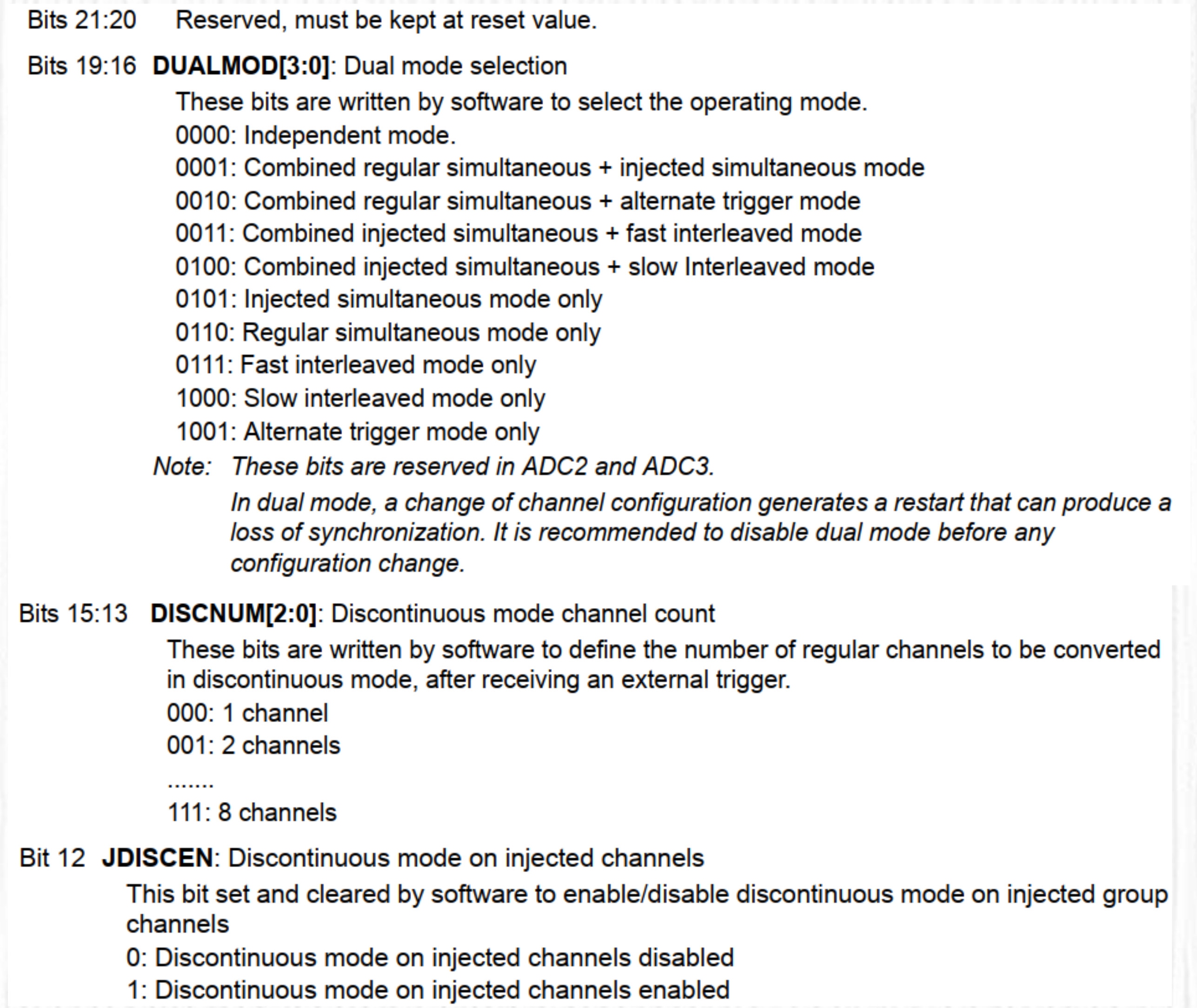
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DISCNUM[2:0] | | | JDISCEN | DISCEN | JAUTO | AWDSGL | SCAN | JEOCIE | AWDIE | EOCIE | AWDCH[4:0] | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw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

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

Bit 23 **AWDEN:** Analog watchdog enable on regular channels
This bit is set/reset by software.
0: Analog watchdog disabled on regular channels
1: Analog watchdog enabled on regular channels

Bit 22 **JAWDEN:** Analog watchdog enable on injected channels
This bit is set/reset by software.
0: Analog watchdog disabled on injected channels
1: Analog watchdog enabled on injected channels

## 2. ADC Registers
## -- ADC_CR1(continued)



Bits 21:20 Reserved, must be kept at reset value.

Bits 19:16 **DUALMOD[3:0]**: Dual mode selection

These bits are written by software to select the operating mode.

0000: Independent mode.
0001: Combined regular simultaneous + injected simultaneous mode
0010: Combined regular simultaneous + alternate trigger mode
0011: Combined injected simultaneous + fast interleaved mode
0100: Combined injected simultaneous + slow Interleaved mode
0101: Injected simultaneous mode only
0110: Regular simultaneous mode only
0111: Fast interleaved mode only
1000: Slow interleaved mode only
1001: Alternate trigger mode only

*Note: These bits are reserved in ADC2 and ADC3.*

*In dual mode, a change of channel configuration generates a restart that can produce a loss of synchronization. It is recommended to disable dual mode before any configuration change.*

Bits 15:13 **DISCNUM[2:0]**: Discontinuous mode channel count

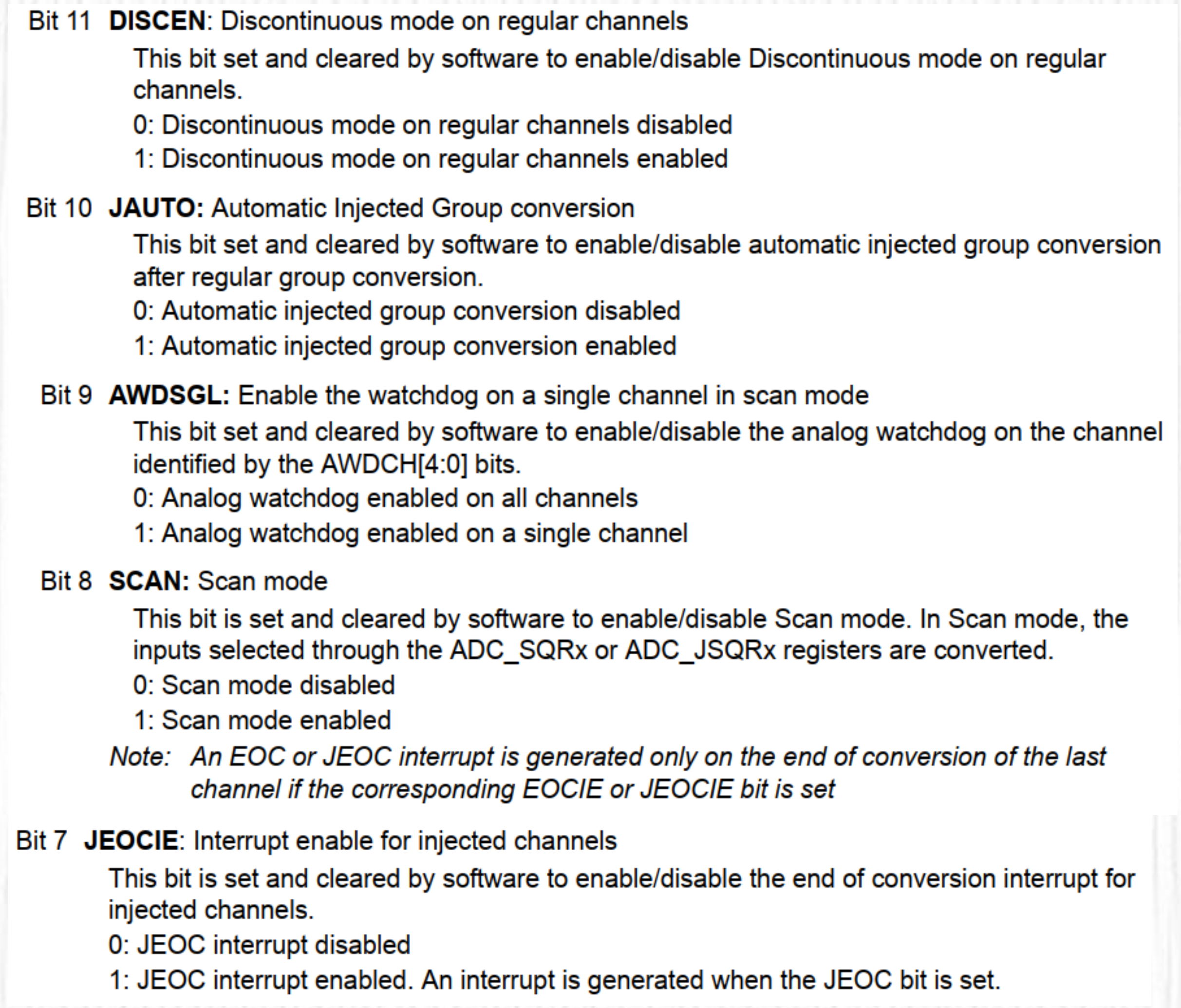
These bits are written by software to define the number of regular channels to be converted in discontinuous mode, after receiving an external trigger.

000: 1 channel
001: 2 channels
.......
111: 8 channels

Bit 12 **JDISCEN**: Discontinuous mode on injected channels

This bit set and cleared by software to enable/disable discontinuous mode on injected group channels

0: Discontinuous mode on injected channels disabled
1: Discontinuous mode on injected channels enabled



Bit 11 **DISCEN**: Discontinuous mode on regular channels

This bit set and cleared by software to enable/disable Discontinuous mode on regular channels.

0: Discontinuous mode on regular channels disabled
1: Discontinuous mode on regular channels enabled

Bit 10 **JAUTO:** Automatic Injected Group conversion

This bit set and cleared by software to enable/disable automatic injected group conversion after regular group conversion.

0: Automatic injected group conversion disabled
1: Automatic injected group conversion enabled

Bit 9 **AWDSGL:** Enable the watchdog on a single channel in scan mode

This bit set and cleared by software to enable/disable the analog watchdog on the channel identified by the AWDCH[4:0] bits.

0: Analog watchdog enabled on all channels
1: Analog watchdog enabled on a single channel

Bit 8 **SCAN:** Scan mode

This bit is set and cleared by software to enable/disable Scan mode. In Scan mode, the inputs selected through the ADC_SQRx or ADC_JSQRx registers are converted.

0: Scan mode disabled
1: Scan mode enabled

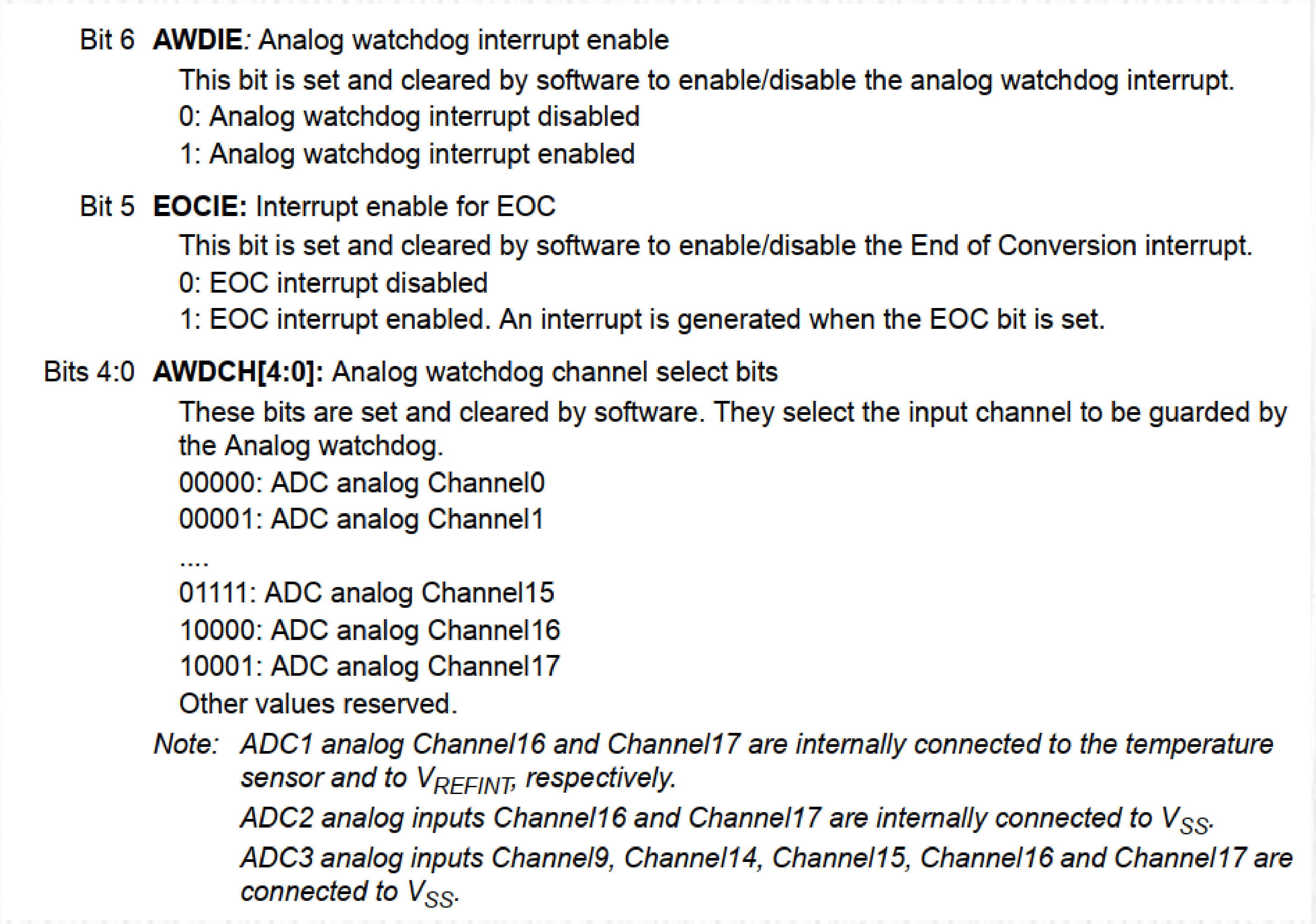
*Note: An EOC or JEOC interrupt is generated only on the end of conversion of the last channel if the corresponding EOCIE or JEOCIE bit is set*

Bit 7 **JEOCIE**: Interrupt enable for injected channels

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

0: JEOC interrupt disabled
1: JEOC interrupt enabled. An interrupt is generated when the JEOC bit is set.

## 2. ADC Registers
## -- ADC_CR1(continued)



Bit 6 **AWDIE**: Analog watchdog interrupt enable

This bit is set and cleared by software to enable/disable the analog watchdog interrupt.

0: Analog watchdog interrupt disabled

1: Analog watchdog interrupt enabled

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.

Bits 4:0 **AWDCH[4:0]:** Analog watchdog channel select bits

These bits are set and cleared by software. They select the input channel to be guarded by the Analog watchdog.

00000: ADC analog Channel0

00001: ADC analog Channel1

....

01111: ADC analog Channel15

10000: ADC analog Channel16

10001: ADC analog Channel17

Other values reserved.

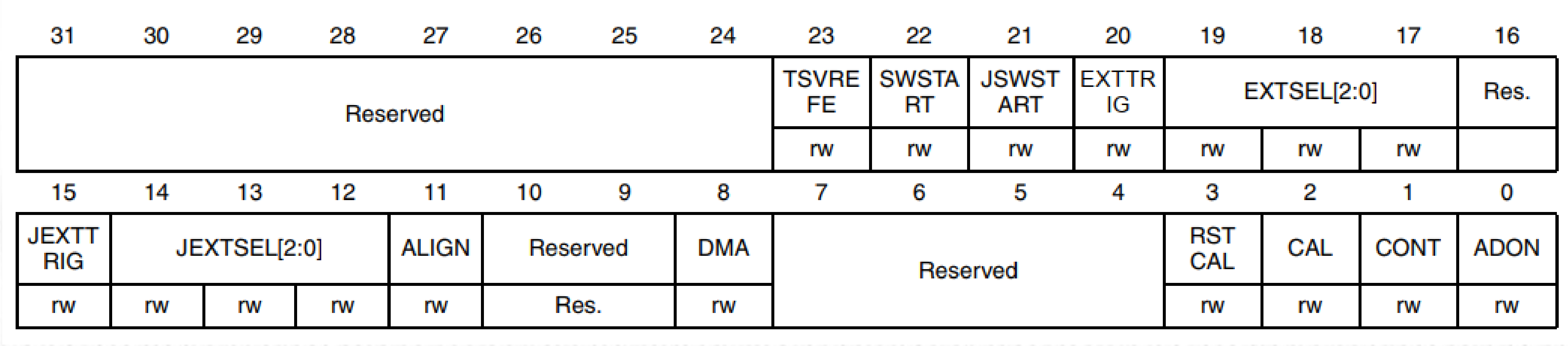
*Note: ADC1 analog Channel16 and Channel17 are internally connected to the temperature sensor and to $V_{REFINT}$, respectively.*

*ADC2 analog inputs Channel16 and Channel17 are internally connected to $V_{SS}$.*

*ADC3 analog inputs Channel9, Channel14, Channel15, Channel16 and Channel17 are connected to $V_{SS}$.*

## 2. ADC Registers
### -- ADC_CR2

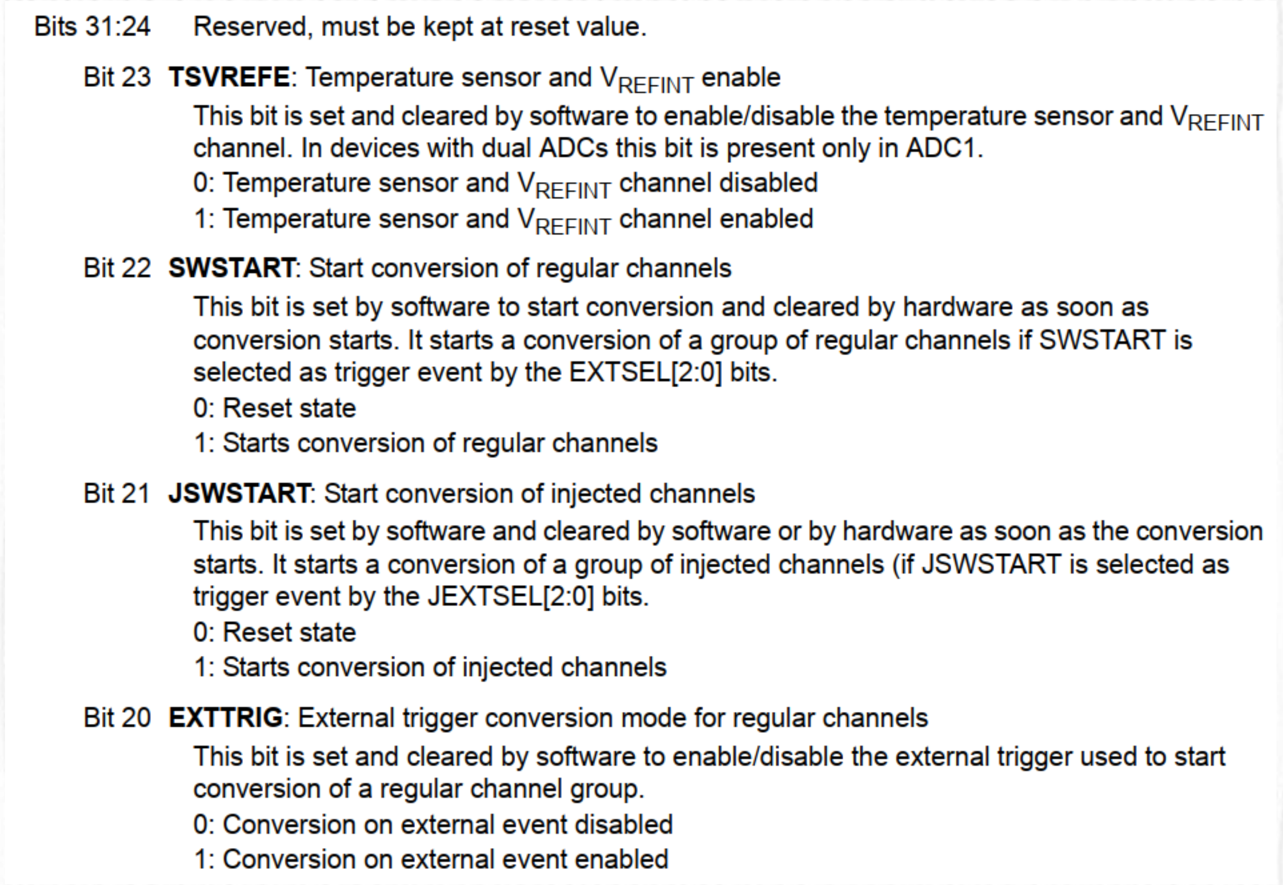
- ADC control register2



| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | TSVREFE | SWSTART | JSWSTART | EXTTRIG | EXTSEL[2:0] | | | Res. |
| | | | | | | | | rw | rw | rw | rw | rw | rw | rw | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| JEXTTRIG | JEXTSEL[2:0] | | | ALIGN | Reserved | | DMA | Reserved | | | | RSTCAL | CAL | CONT | ADON |
| rw | rw | rw | rw | rw | Res. | | rw | | | | | rw | rw | rw | rw |

# 2. ADC Registers -- ADC_CR2(continued)



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

Bit 23 **TSVREFE**: Temperature sensor and $V_{REFINT}$ enable

This bit is set and cleared by software to enable/disable the temperature sensor and $V_{REFINT}$ channel. In devices with dual ADCs this bit is present only in ADC1.

0: Temperature sensor and $V_{REFINT}$ channel disabled

1: Temperature sensor and $V_{REFINT}$ channel enabled

Bit 22 **SWSTART**: Start conversion of regular channels

This bit is set by software to start conversion and cleared by hardware as soon as conversion starts. It starts a conversion of a group of regular channels if SWSTART is selected as trigger event by the EXTSEL[2:0] bits.

0: Reset state

1: Starts conversion of regular channels

Bit 21 **JSWSTART**: Start conversion of injected channels

This bit is set by software and cleared by software or by hardware as soon as the conversion starts. It starts a conversion of a group of injected channels (if JSWSTART is selected as trigger event by the JEXTSEL[2:0] bits.

0: Reset state

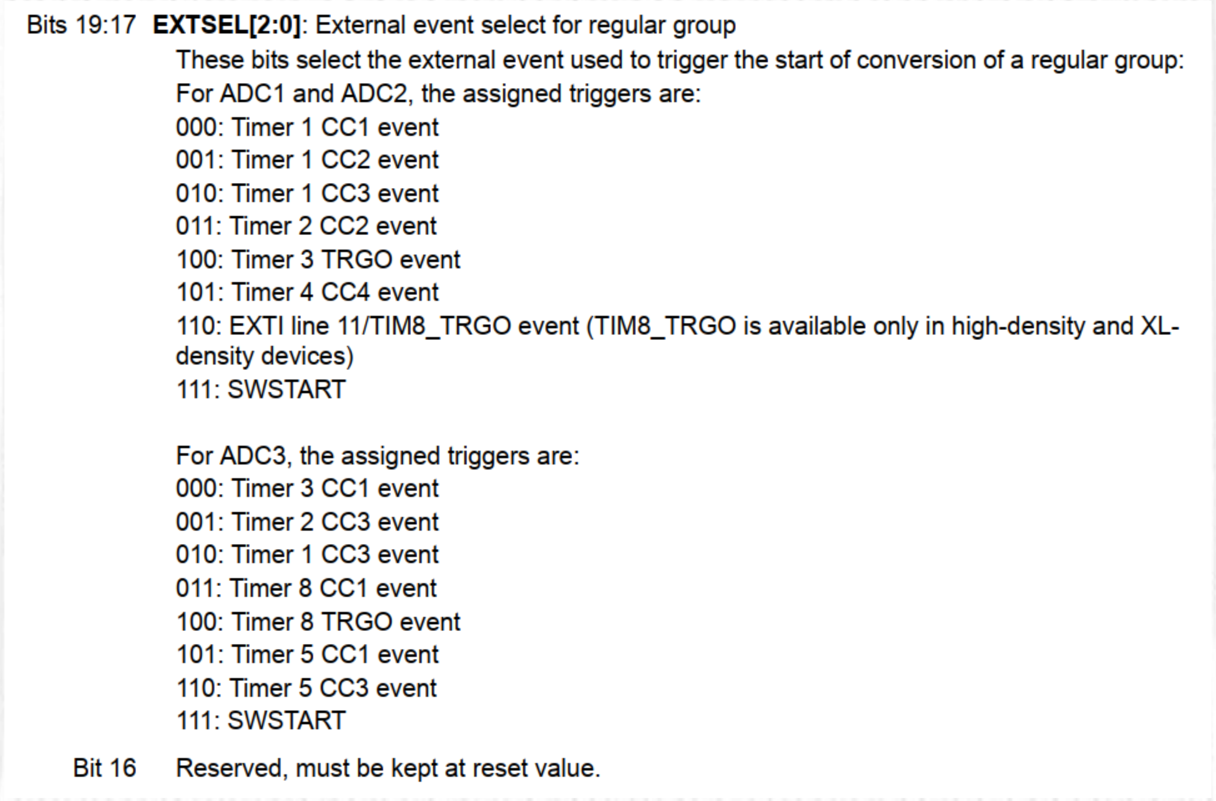
1: Starts conversion of injected channels

Bit 20 **EXTTRIG**: External trigger conversion mode for regular channels

This bit is set and cleared by software to enable/disable the external trigger used to start conversion of a regular channel group.

0: Conversion on external event disabled

1: Conversion on external event enabled



Bits 19:17 **EXTSEL[2:0]**: External event select for regular group

These bits select the external event used to trigger the start of conversion of a regular group:

For ADC1 and ADC2, the assigned triggers are:

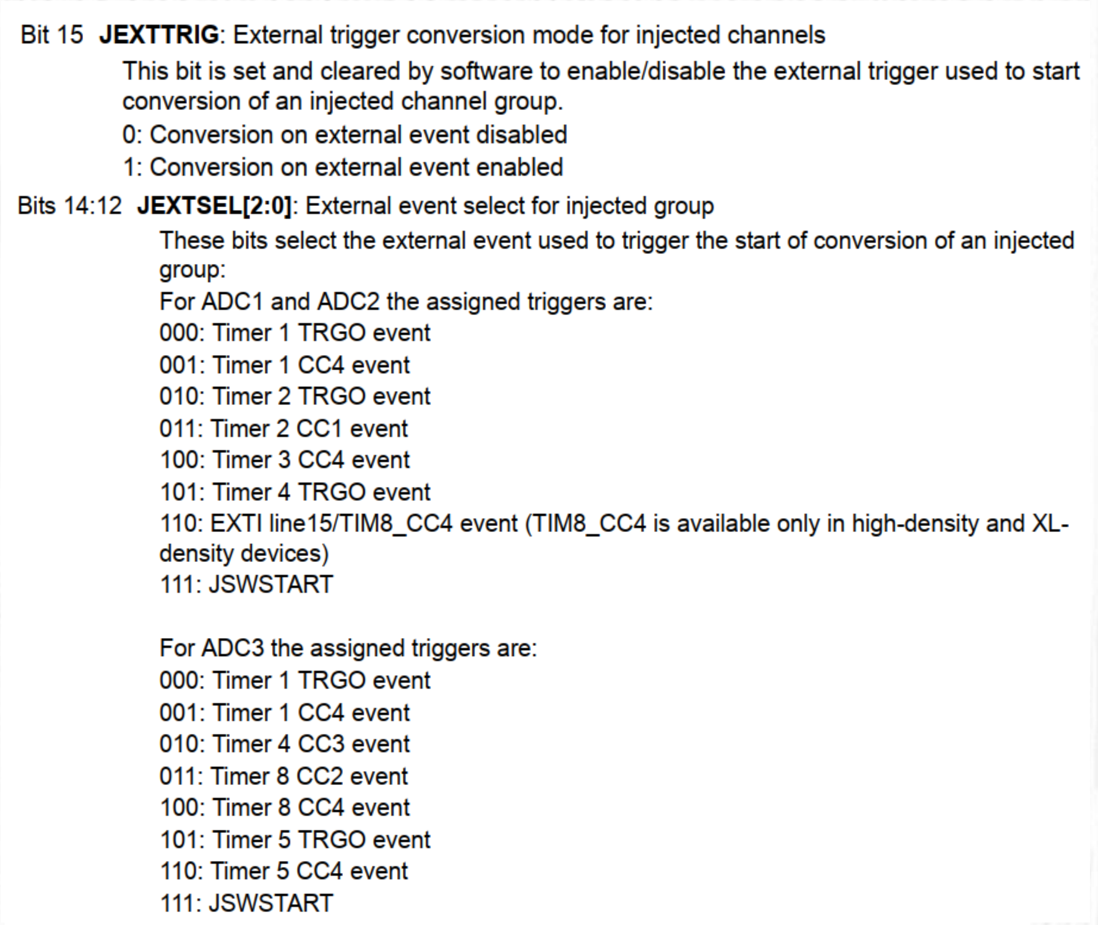
000: Timer 1 CC1 event
001: Timer 1 CC2 event
010: Timer 1 CC3 event
011: Timer 2 CC2 event
100: Timer 3 TRGO event
101: Timer 4 CC4 event
110: EXTI line 11/TIM8_TRGO event (TIM8_TRGO is available only in high-density and XL-density devices)
111: SWSTART

For ADC3, the assigned triggers are:

000: Timer 3 CC1 event
001: Timer 2 CC3 event
010: Timer 1 CC3 event
011: Timer 8 CC1 event
100: Timer 8 TRGO event
101: Timer 5 CC1 event
110: Timer 5 CC3 event
111: SWSTART

Bit 16 Reserved, must be kept at reset value.

# 2. ADC Registers
## -- ADC_CR2(continued)



Bit 15 **JEXTTRIG**: External trigger conversion mode for injected channels

This bit is set and cleared by software to enable/disable the external trigger used to start conversion of an injected channel group.

0: Conversion on external event disabled

1: Conversion on external event enabled

Bits 14:12 **JEXTSEL[2:0]**: External event select for injected group

These bits select the external event used to trigger the start of conversion of an injected group:

For ADC1 and ADC2 the assigned triggers are:

000: Timer 1 TRGO event

001: Timer 1 CC4 event

010: Timer 2 TRGO event

011: Timer 2 CC1 event

100: Timer 3 CC4 event

101: Timer 4 TRGO event

110: EXTI line15/TIM8_CC4 event (TIM8_CC4 is available only in high-density and XL-density devices)

111: JSWSTART

For ADC3 the assigned triggers are:

000: Timer 1 TRGO event

001: Timer 1 CC4 event

010: Timer 4 CC3 event

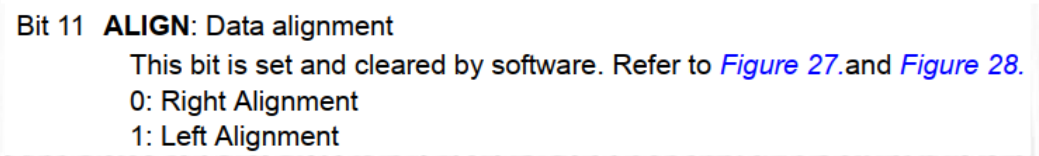
011: Timer 8 CC2 event

100: Timer 8 CC4 event

101: Timer 5 TRGO event

110: Timer 5 CC4 event

111: JSWSTART

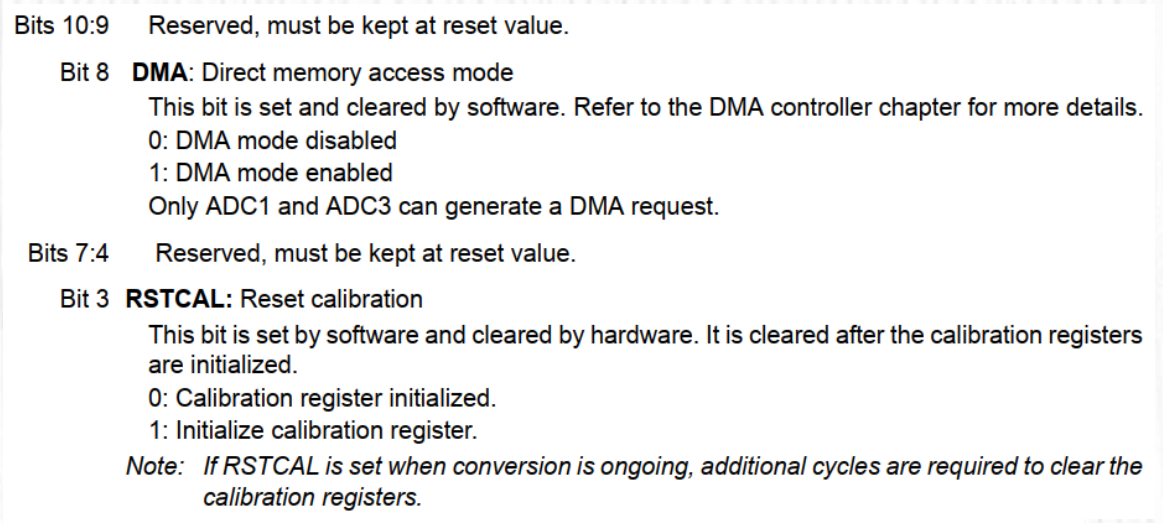


Bit 11 **ALIGN**: Data alignment

This bit is set and cleared by software. Refer to *Figure 27*.and *Figure 28*.

0: Right Alignment

1: Left Alignment



Bits 10:9 Reserved, must be kept at reset value.

Bit 8 **DMA**: Direct memory access mode

This bit is set and cleared by software. Refer to the DMA controller chapter for more details.

0: DMA mode disabled

1: DMA mode enabled

Only ADC1 and ADC3 can generate a DMA request.

Bits 7:4 Reserved, must be kept at reset value.

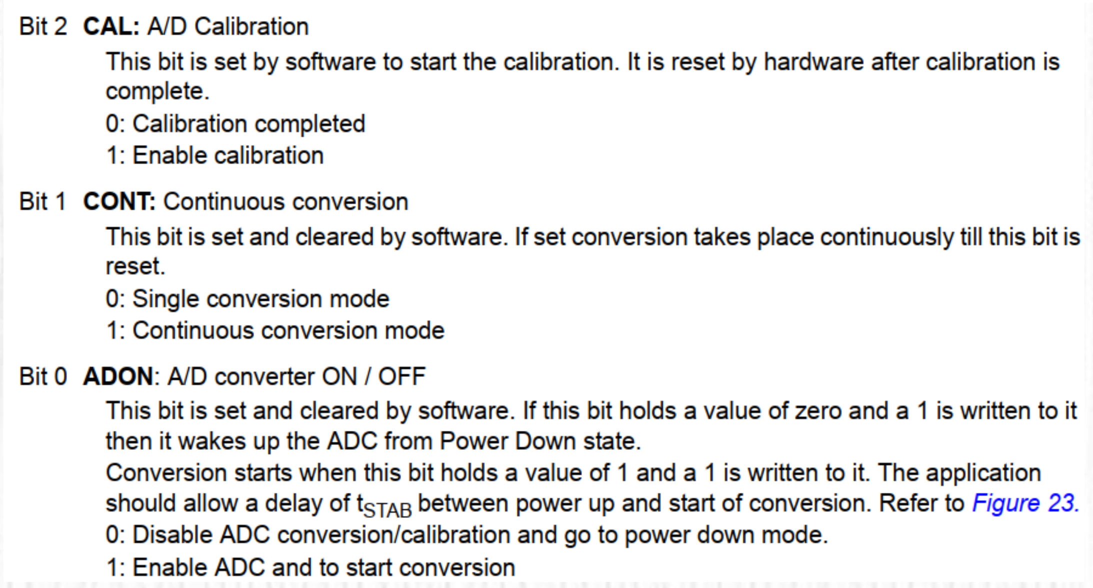
Bit 3 **RSTCAL:** Reset calibration

This bit is set by software and cleared by hardware. It is cleared after the calibration registers are initialized.

0: Calibration register initialized.

1: Initialize calibration register.

*Note: If RSTCAL is set when conversion is ongoing, additional cycles are required to clear the calibration registers.*



Bit 2 **CAL:** A/D Calibration

This bit is set by software to start the calibration. It is reset by hardware after calibration is complete.

0: Calibration completed

1: Enable calibration

Bit 1 **CONT:** Continuous conversion

This bit is set and cleared by software. If set conversion takes place continuously till this bit is reset.

0: Single conversion mode

1: Continuous conversion mode

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

This bit is set and cleared by software. If this bit holds a value of zero and a 1 is written to it then it wakes up the ADC from Power Down state.

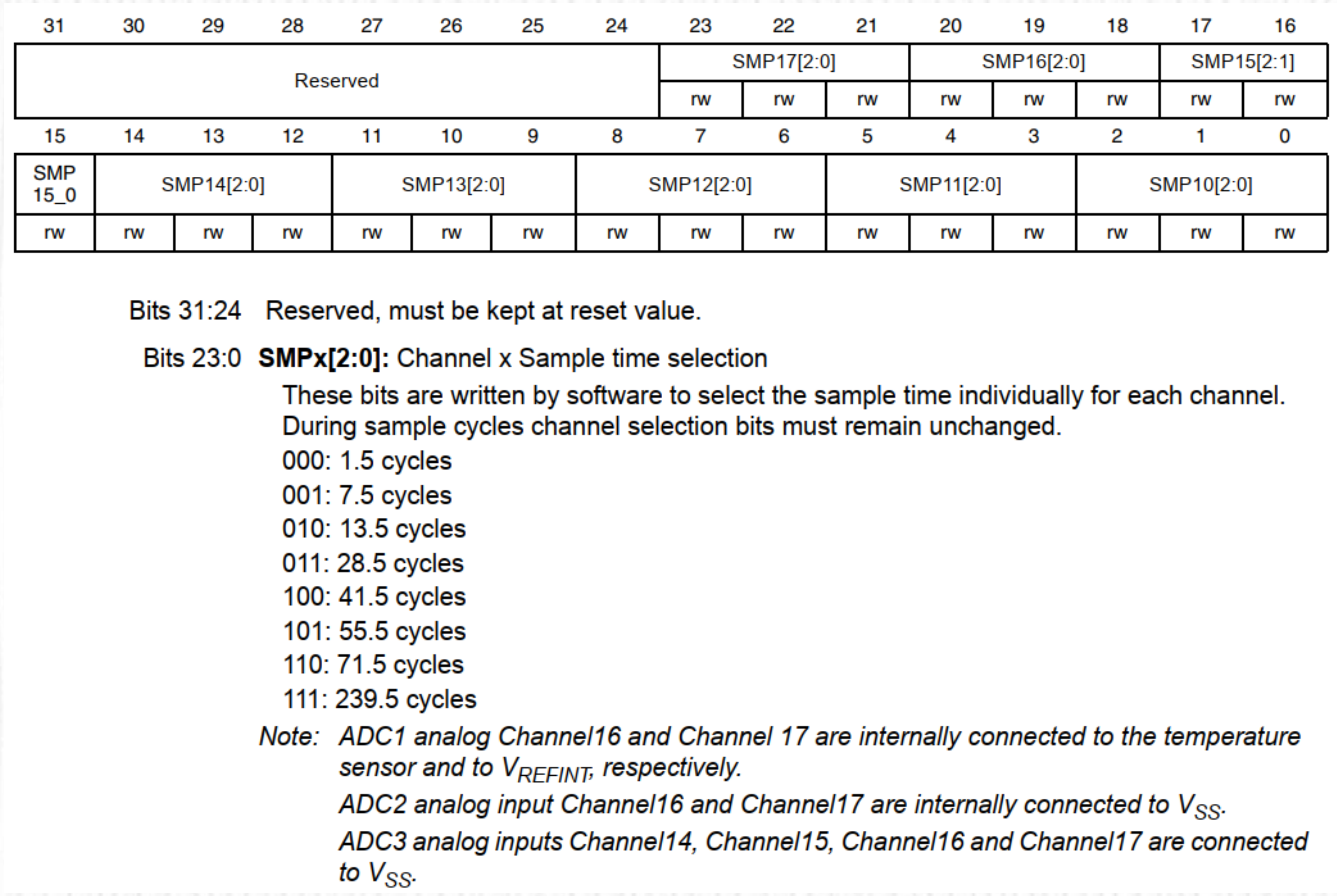
Conversion starts when this bit holds a value of 1 and a 1 is written to it. The application should allow a delay of $t_{STAB}$ between power up and start of conversion. Refer to *Figure 23*.

0: Disable ADC conversion/calibration and go to power down mode.

1: Enable ADC and to start conversion

## 2. ADC Registers
## -- ADC_SMPR1

- ADC sample time register 1



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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SMP 15_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 31:24 Reserved, must be kept at reset value.

Bits 23:0 **SMPx[2:0]:** Channel x Sample time selection

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

000: 1.5 cycles
001: 7.5 cycles
010: 13.5 cycles
011: 28.5 cycles
100: 41.5 cycles
101: 55.5 cycles
110: 71.5 cycles
111: 239.5 cycles

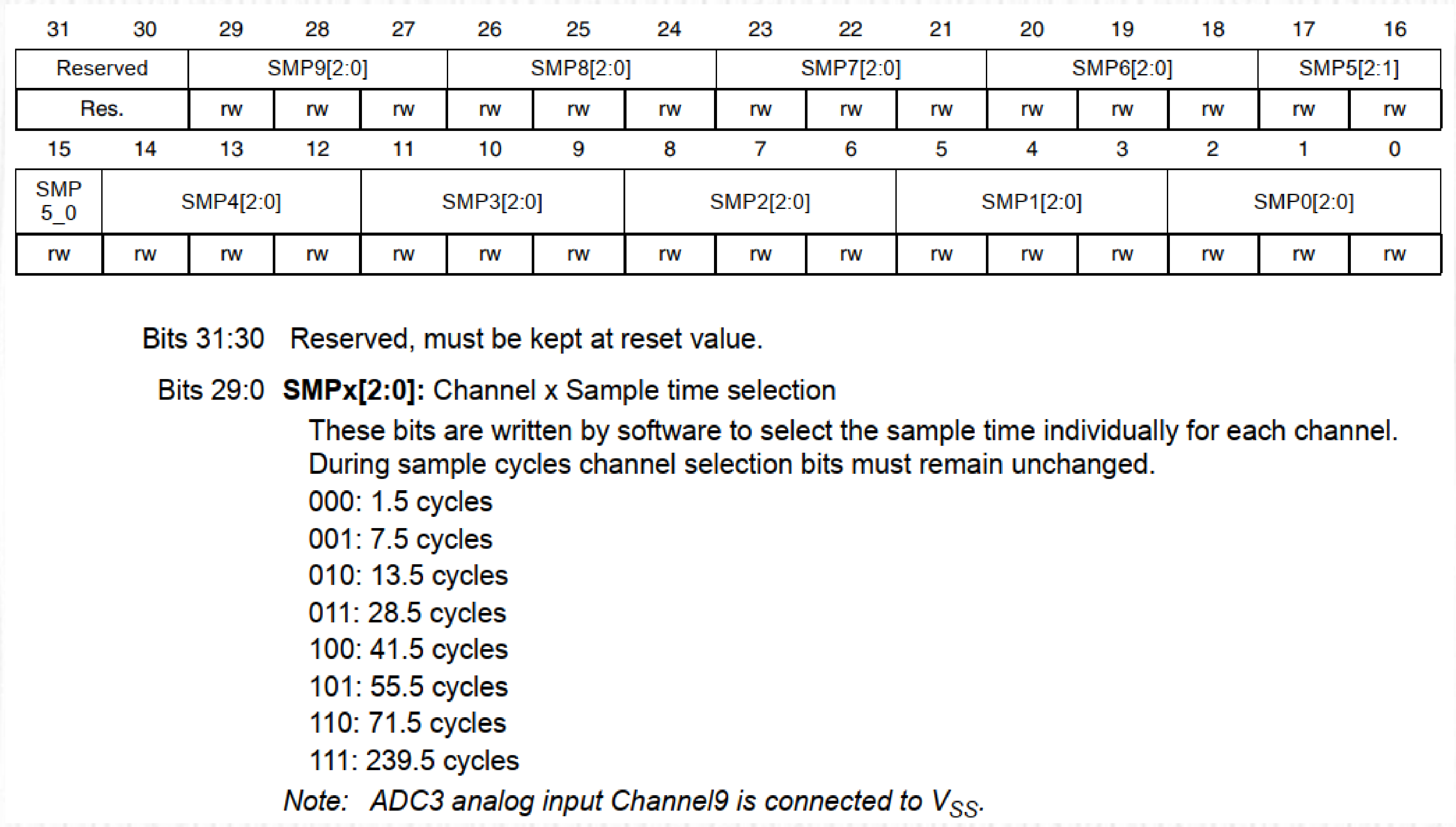
*Note: ADC1 analog Channel16 and Channel 17 are internally connected to the temperature sensor and to* $V_{REFINT}$*, respectively.*

*ADC2 analog input Channel16 and Channel17 are internally connected to* $V_{SS}$*.*

*ADC3 analog inputs Channel14, Channel15, Channel16 and Channel17 are connected to* $V_{SS}$*.*

# 2. ADC Registers
## -- ADC_SMPR2

- ADC sample time register 2



| 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] | |
| Res. | | 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 |

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

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

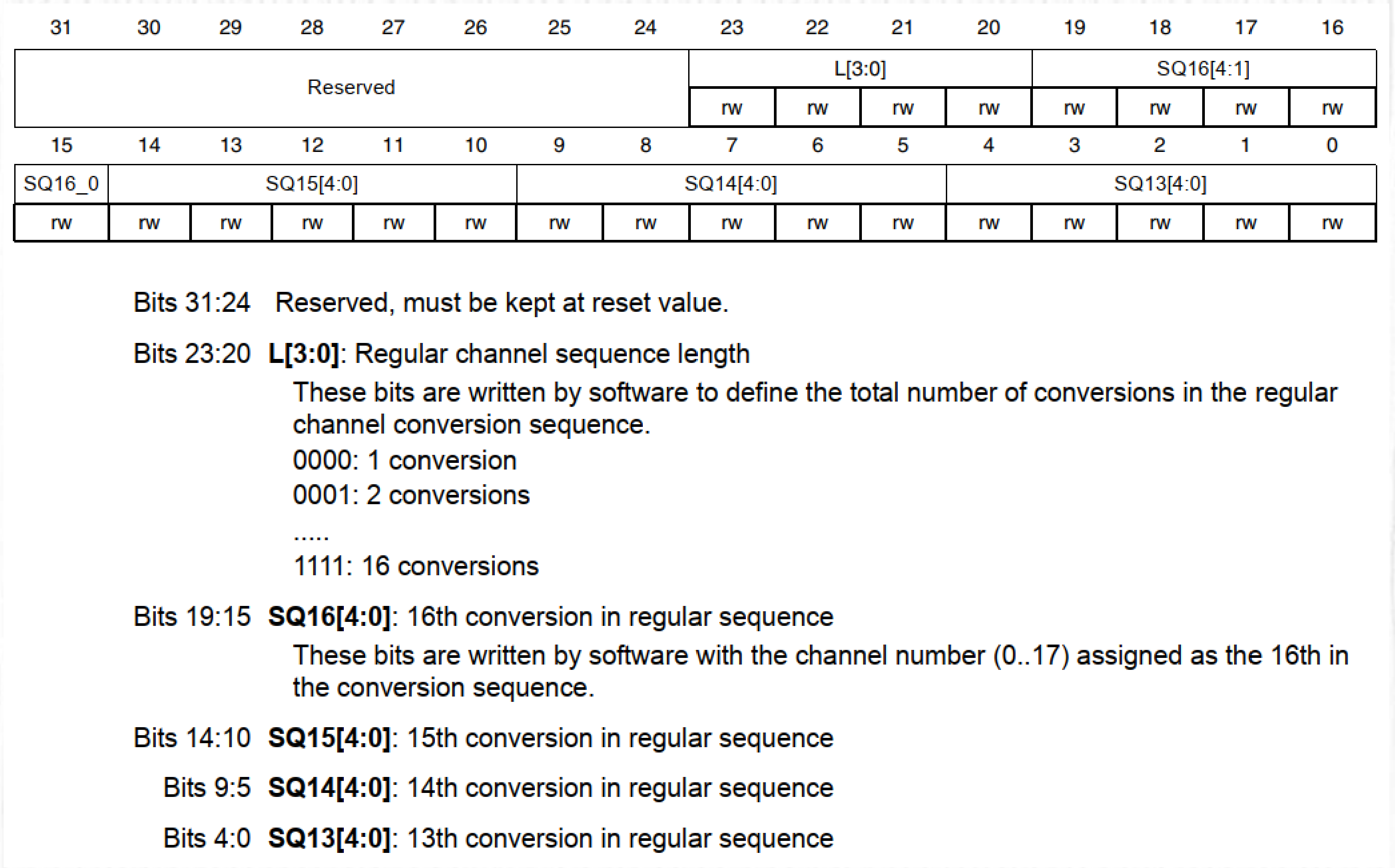
These bits are written by software to select the sample time individually for each channel. During sample cycles channel selection bits must remain unchanged.

000: 1.5 cycles
001: 7.5 cycles
010: 13.5 cycles
011: 28.5 cycles
100: 41.5 cycles
101: 55.5 cycles
110: 71.5 cycles
111: 239.5 cycles

*Note: ADC3 analog input Channel9 is connected to* $V_{SS}$.

# 2. ADC Registers
## -- ADC_SQR1

- ADC regular 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 | 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..17) assigned as the 16th in the conversion sequence.

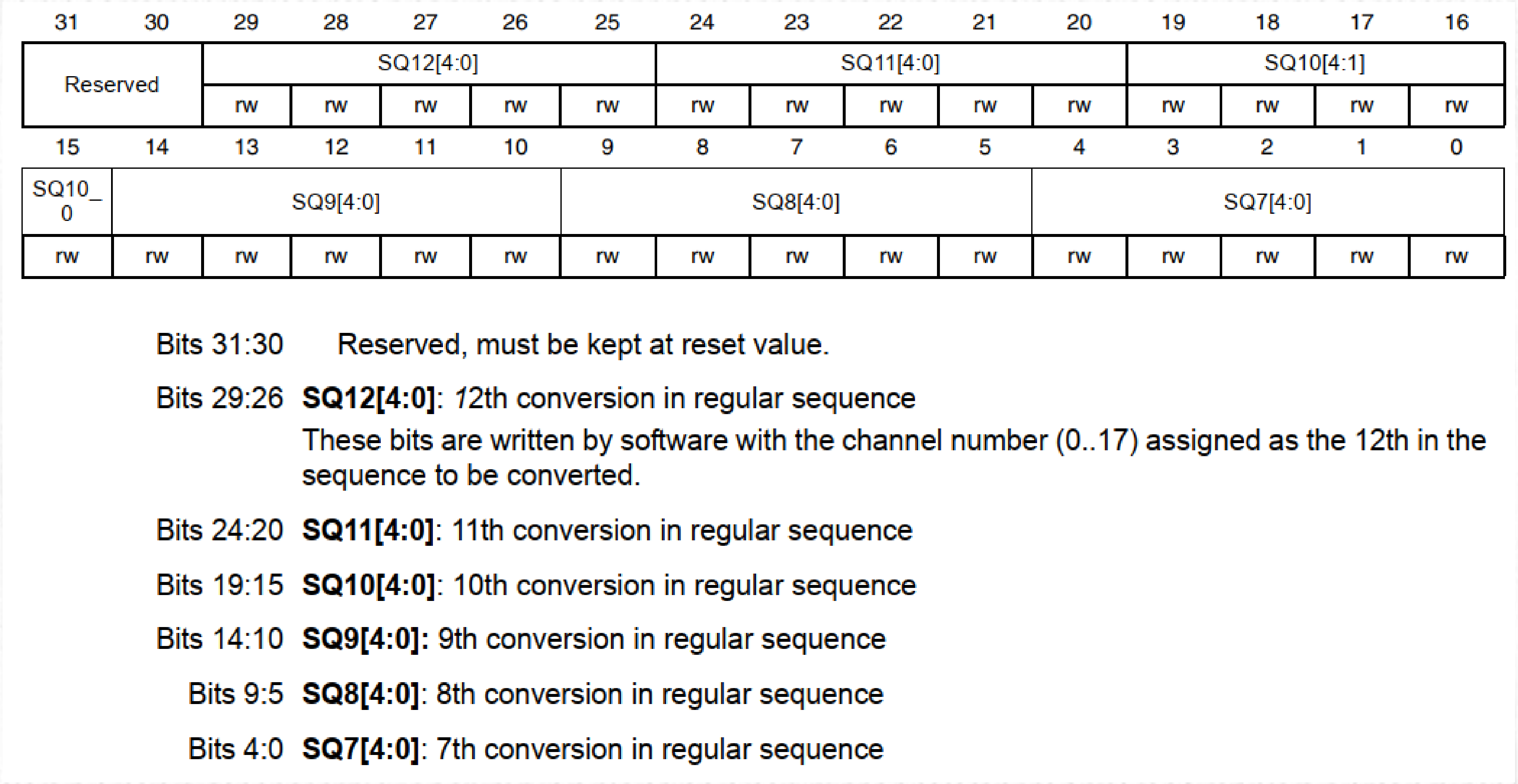
Bits 14:10 **SQ15[4:0]**: 15th conversion in regular sequence

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

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

## 2. ADC Registers
## -- ADC_SQR2

- ADC regular 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 |

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

Bits 29:26 **SQ12[4:0]**: *12*th conversion in regular sequence
These bits are written by software with the channel number (0..17) 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

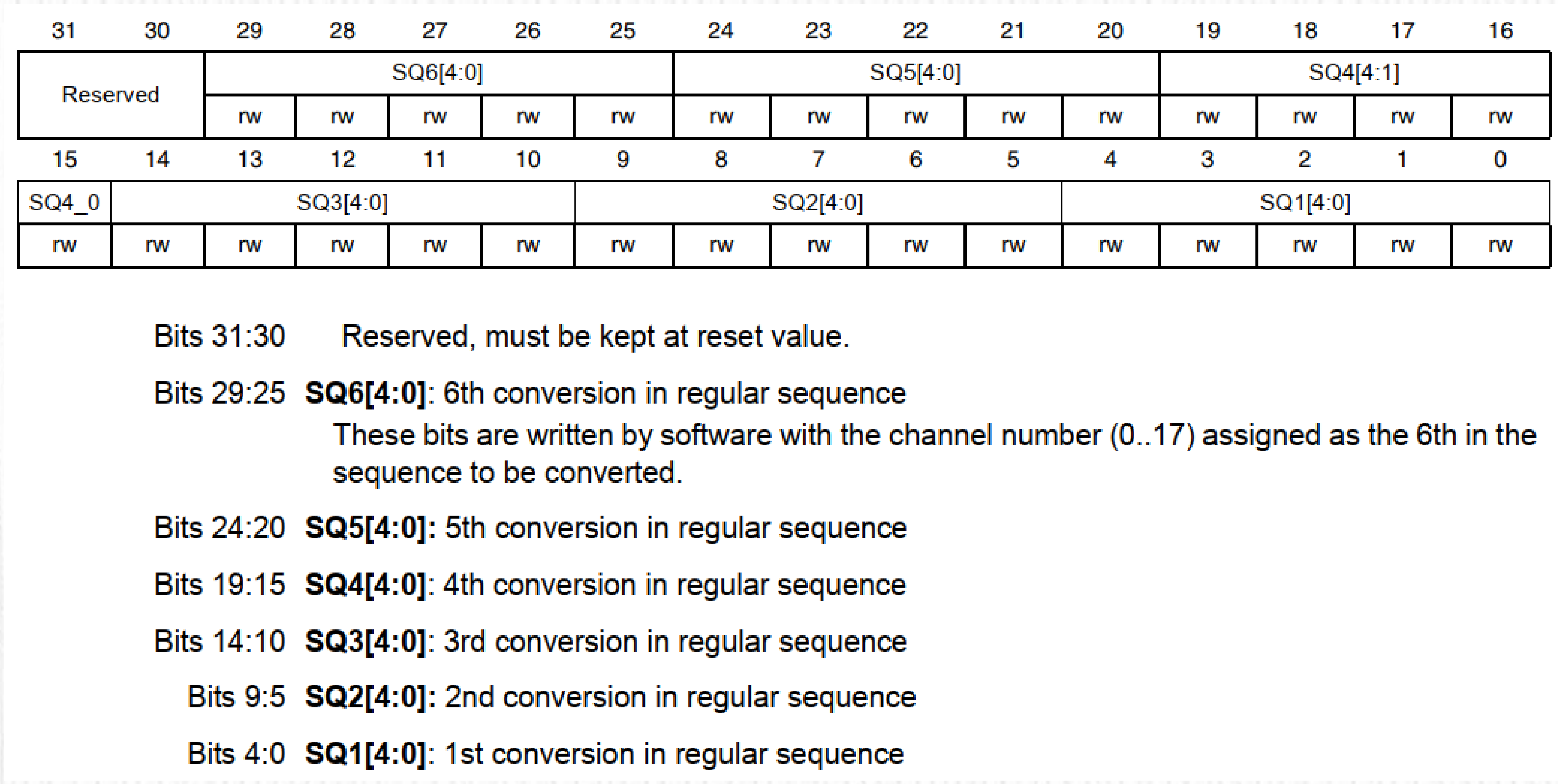
Bits 14:10 **SQ9[4:0]:** 9th conversion in regular sequence

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

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

## 2. ADC Registers
## -- ADC_SQR3

- ADC regular 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 |

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..17) 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

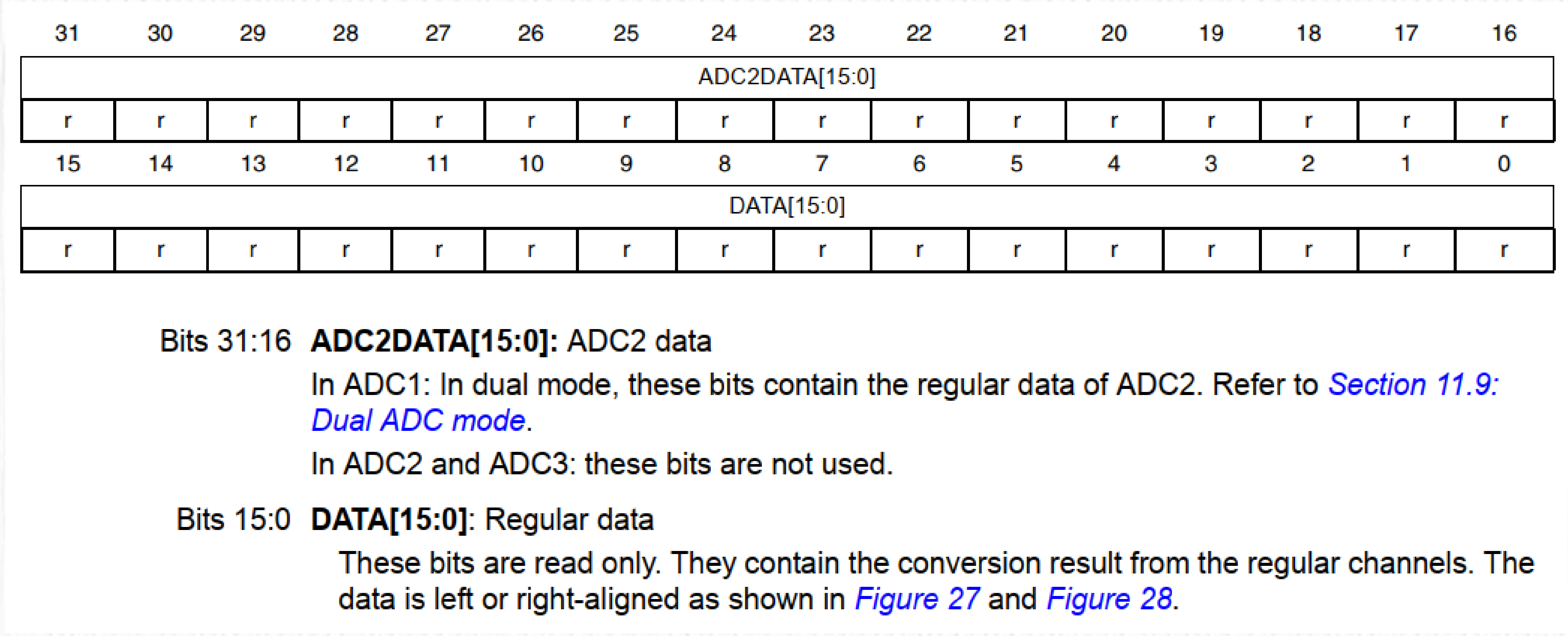
Bits 14:10 **SQ3[4:0]**: 3rd conversion in regular sequence

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

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

## 2. ADC Registers
### -- ADC_DR

- ADC regular data register



| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ADC2DATA[15:0] | | | | | | | | | | | | | | | |
| r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r |
| **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 **ADC2DATA[15:0]:** ADC2 data

In ADC1: In dual mode, these bits contain the regular data of ADC2. Refer to *Section 11.9: Dual ADC mode*.

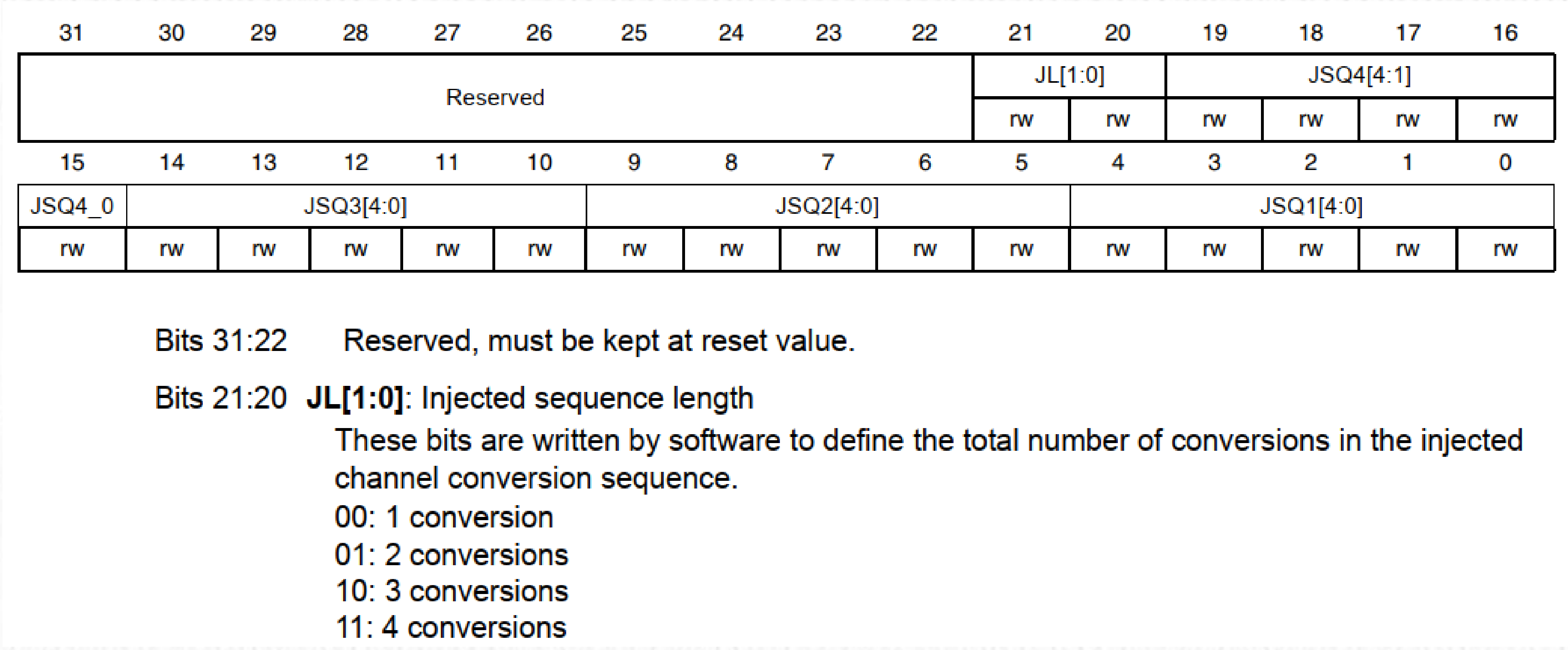
In ADC2 and ADC3: these bits are not used.

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

These bits are read only. They contain the conversion result from the regular channels. The data is left or right-aligned as shown in *Figure 27* and *Figure 28*.

# 2. ADC Registers
## -- ADC_JSQR

- ADC injected sequence register



| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | JL[1:0] | | JSQ4[4:1] | | | |
| | | | | | | | | | | rw | rw | rw | rw | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| JSQ4_0 | JSQ3[4:0] | | | | | JSQ2[4:0] | | | | | JSQ1[4:0] | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bits 31:22 Reserved, must be kept at reset value.

Bits 21:20 **JL[1:0]**: Injected sequence length

These bits are written by software to define the total number of conversions in the injected channel conversion sequence.

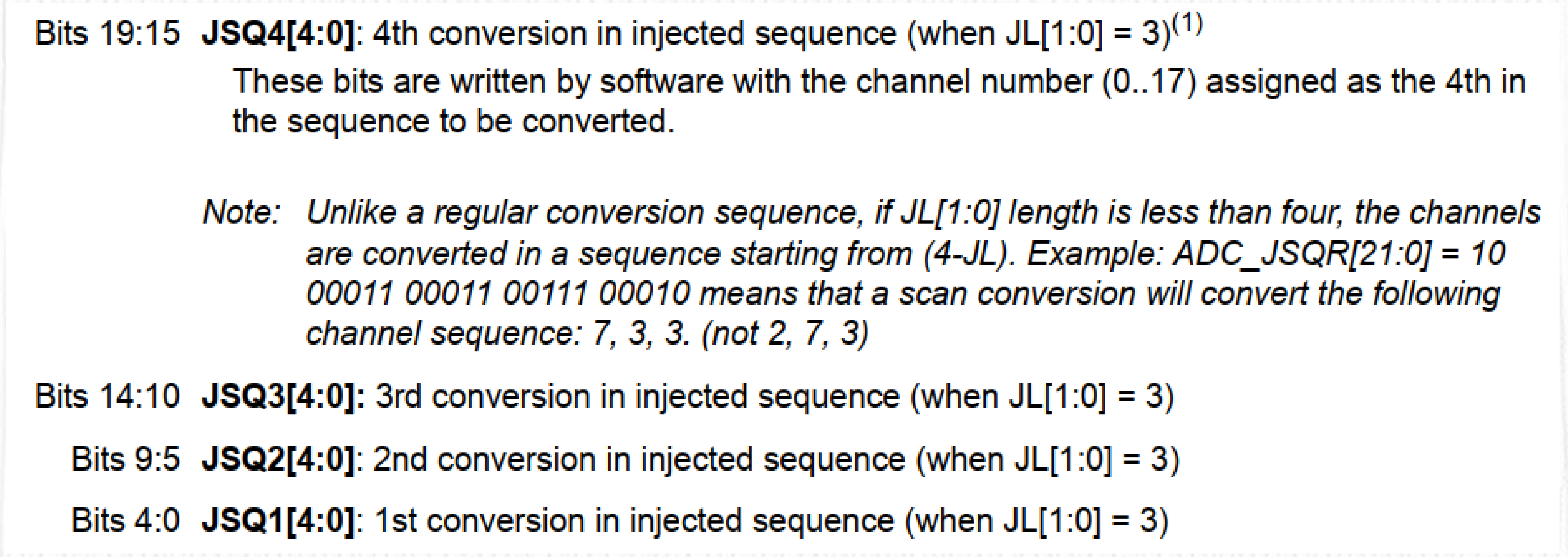
00: 1 conversion

01: 2 conversions

10: 3 conversions

11: 4 conversions

## 2. ADC Registers -- ADC_JSQR(continued)



Bits 19:15 **JSQ4[4:0]**: 4th conversion in injected sequence (when JL[1:0] = 3)(1)

These bits are written by software with the channel number (0..17) assigned as the 4th in the sequence to be converted.

*Note: Unlike a regular conversion sequence, if JL[1:0] length is less than four, the channels are converted in a sequence starting from (4-JL). Example: ADC_JSQR[21:0] = 10 00011 00011 00111 00010 means that a scan conversion will convert the following channel sequence: 7, 3, 3. (not 2, 7, 3)*

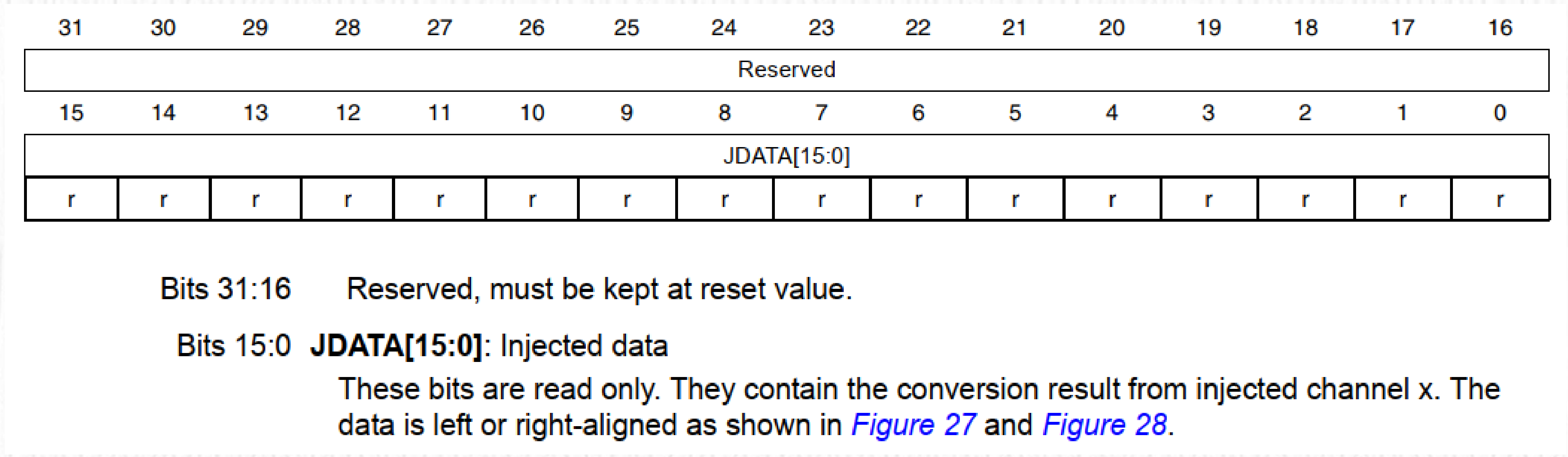
Bits 14:10 **JSQ3[4:0]:** 3rd conversion in injected sequence (when JL[1:0] = 3)

Bits 9:5 **JSQ2[4:0]**: 2nd conversion in injected sequence (when JL[1:0] = 3)

Bits 4:0 **JSQ1[4:0]**: 1st conversion in injected sequence (when JL[1:0] = 3)

## 2. ADC Registers
### -- ADC_JDRx

- ADC injected data register x



| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| JDATA[15:0] | | | | | | | | | | | | | | | |
| r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r |

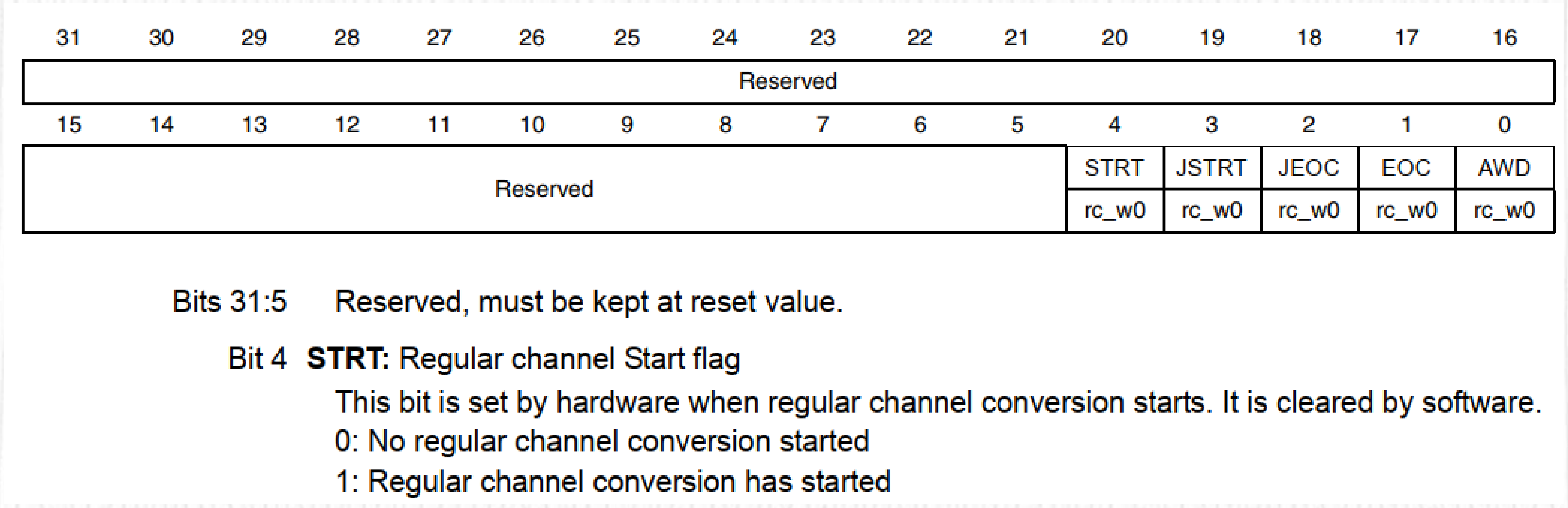
Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **JDATA[15:0]**: Injected data

These bits are read only. They contain the conversion result from injected channel x. The data is left or right-aligned as shown in *Figure 27* and *Figure 28*.

## 2. ADC Registers
## -- ADC_SR

- ADC status register



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

Bits 31:5 Reserved, must be kept at reset value.

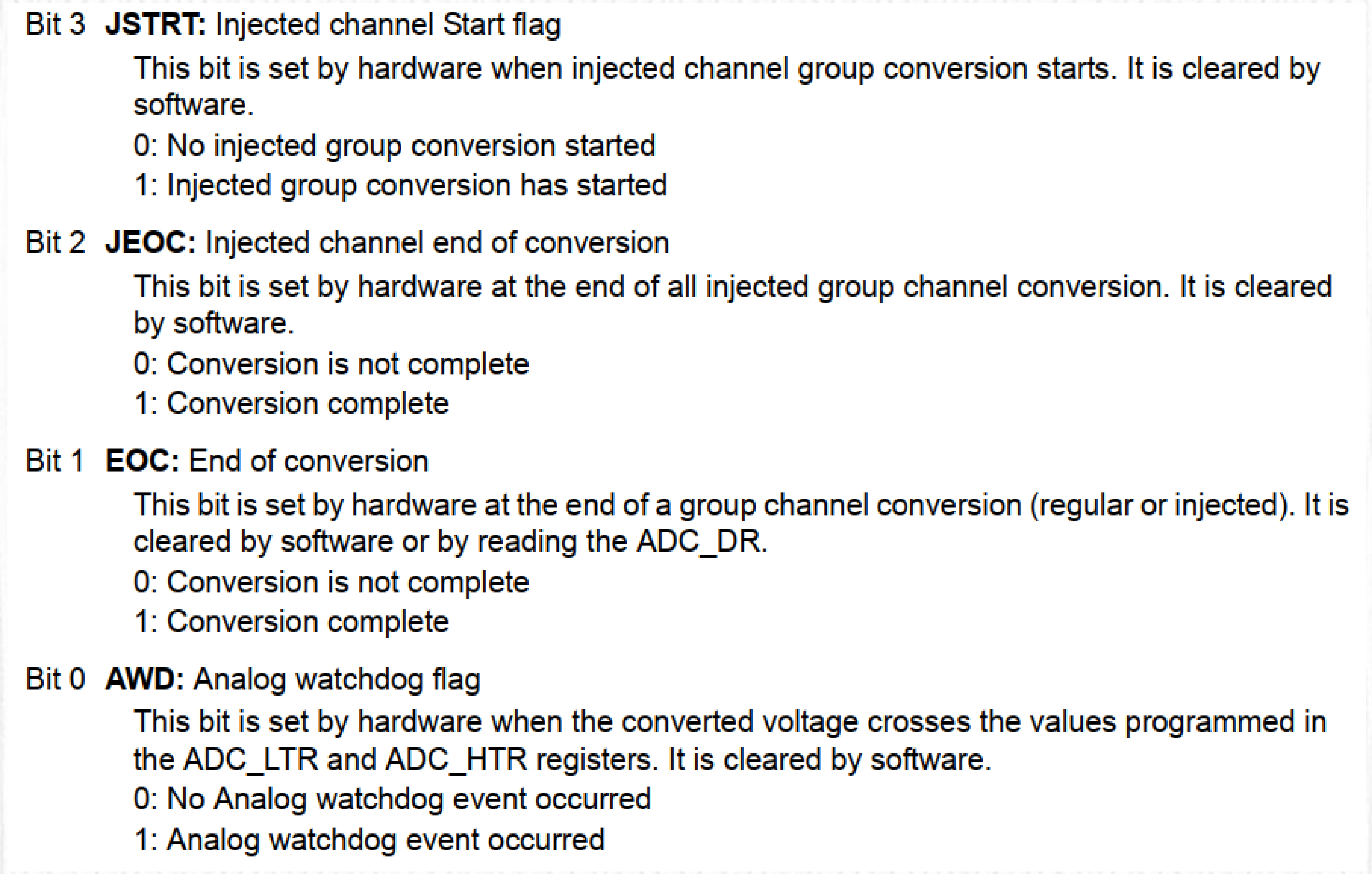
Bit 4 **STRT:** Regular channel Start flag

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

## 2. ADC Registers
## -- ADC_SR(continued)



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

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

0: Conversion is not complete

1: Conversion complete

Bit 1 **EOC:** End of conversion

This bit is set by hardware at the end of a group channel conversion (regular or injected). It is cleared by software or by reading the ADC_DR.

0: Conversion is not complete

1: Conversion complete

Bit 0 **AWD:** Analog watchdog flag

This bit is set by hardware when the converted voltage crosses the values programmed in the ADC_LTR and ADC_HTR registers. It is cleared by software.

0: No Analog watchdog event occurred

1: Analog watchdog event occurred

南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

03

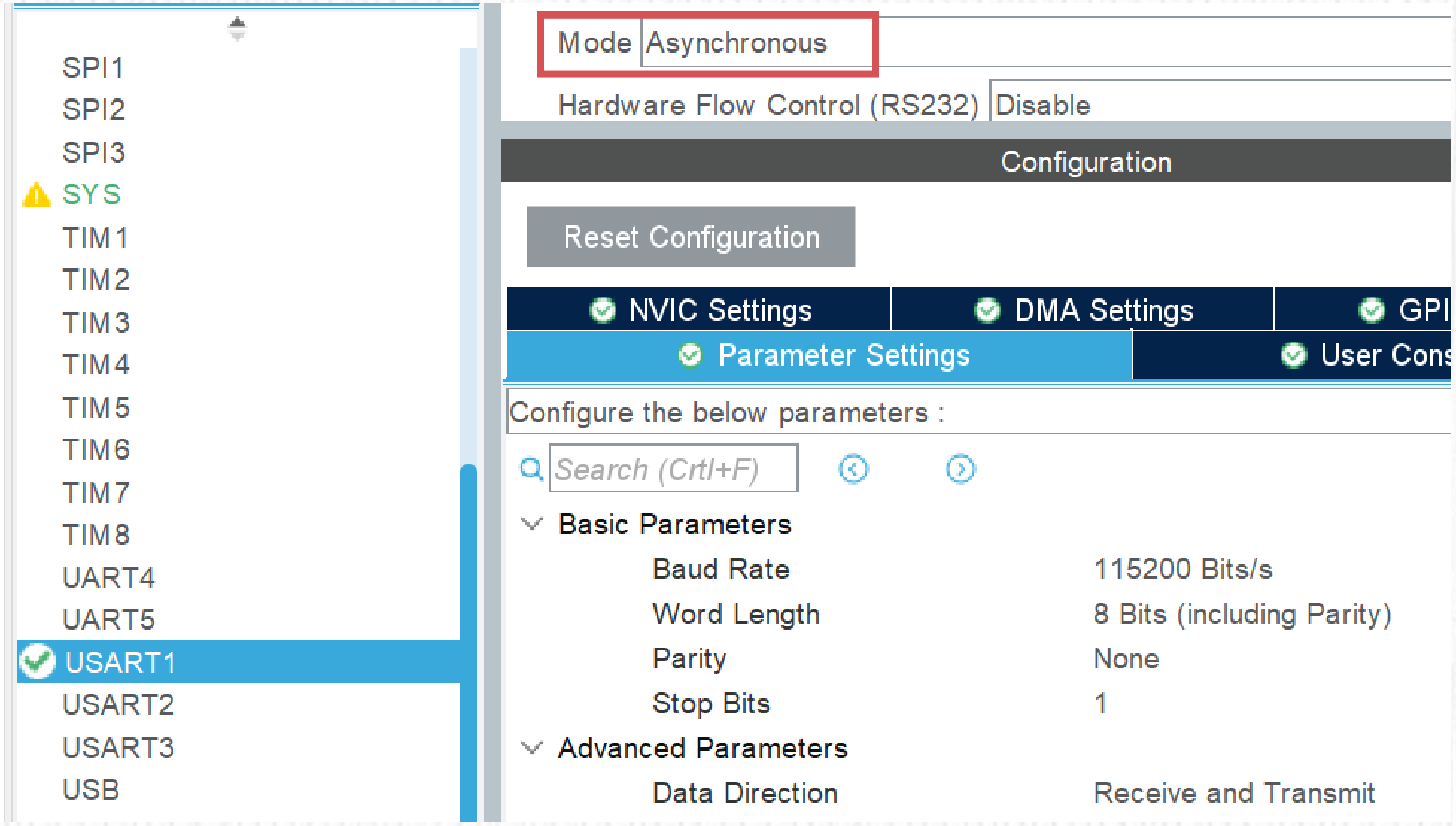
## How to Program

## 3. How to Program

- Our Goal
  - Use ADC1 to get the voltage of GPIO pin， and show it with USART1 or on LCD screen.

## 3. How to Program

- Configure USART1
  - Set the USART1 as asynchronous mode



Mode Asynchronous

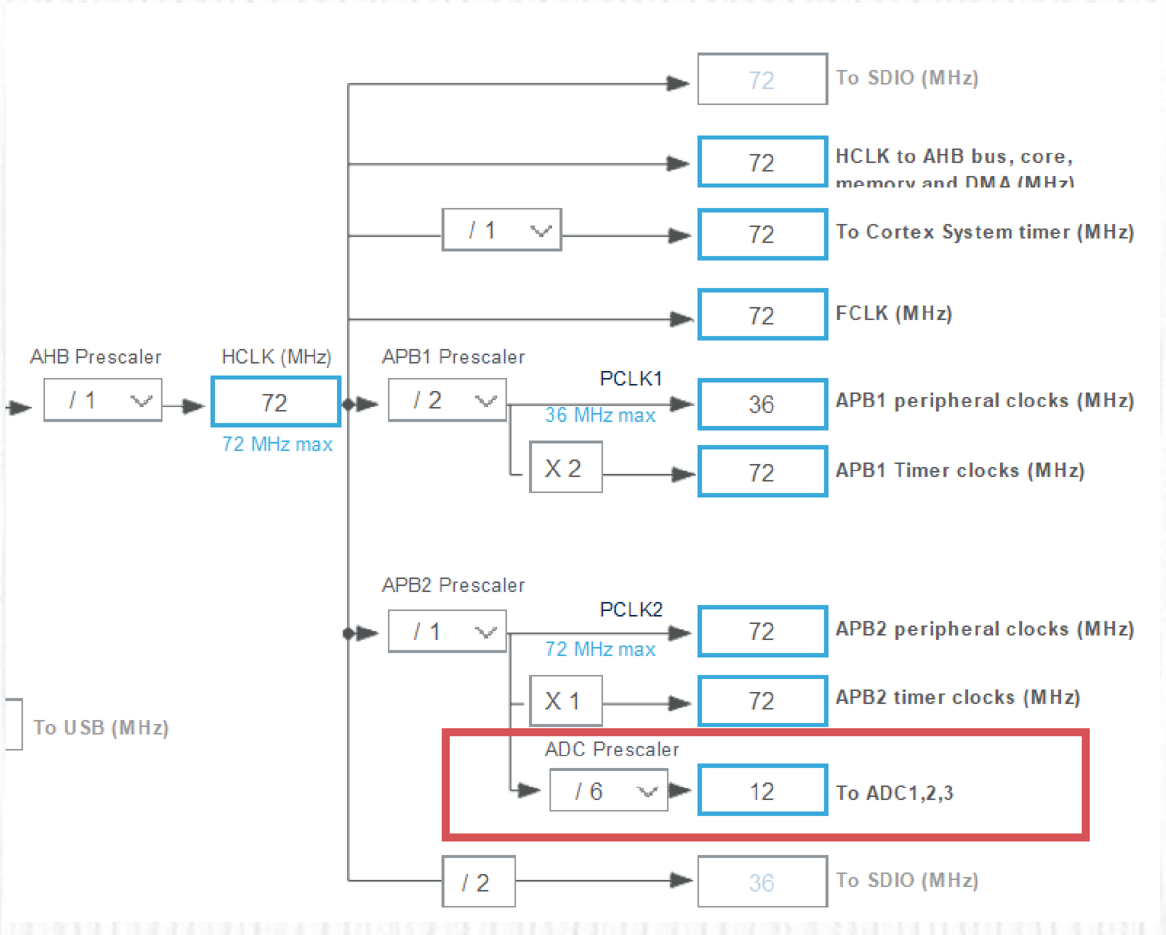
Hardware Flow Control (RS232) Disable

Configure the below parameters :

| | |
|---|---|
| **Basic Parameters** | |
| Baud Rate | 115200 Bits/s |
| Word Length | 8 Bits (including Parity) |
| Parity | None |
| Stop Bits | 1 |
| **Advanced Parameters** | |
| Data Direction | Receive and Transmit |

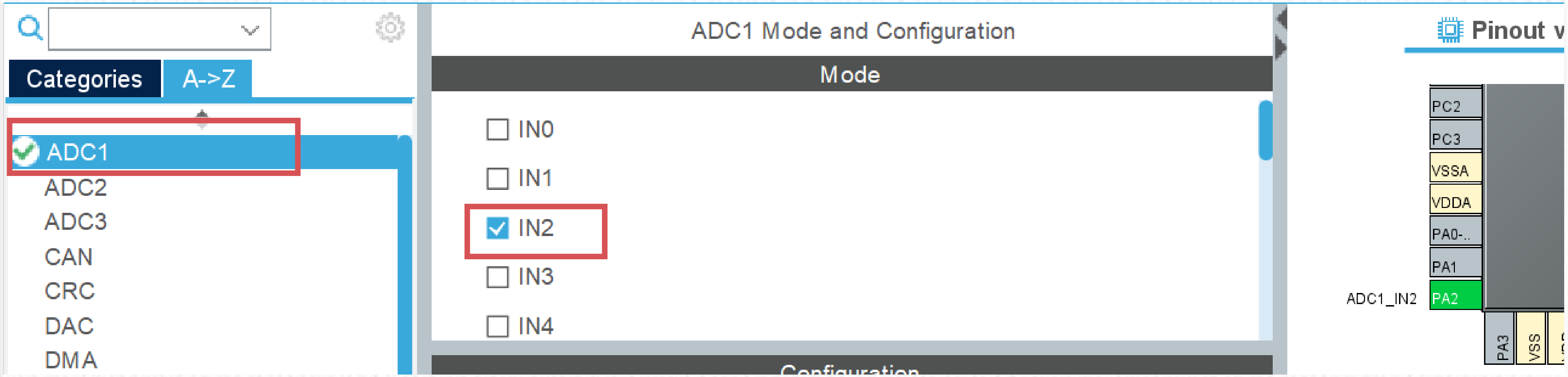
## 3. How to Program

- Configure Clock
  - ADC Clock frequency should be less than or equal to 14MHz



## 3. How to Program

- Configure ADC1
  - Go to the Analog categories, click ADC1 and select IN2, which means we enable the channel 2 of ADC1 and we are able to measure the voltage of PA2



ADC1 Mode and Configuration — Mode:

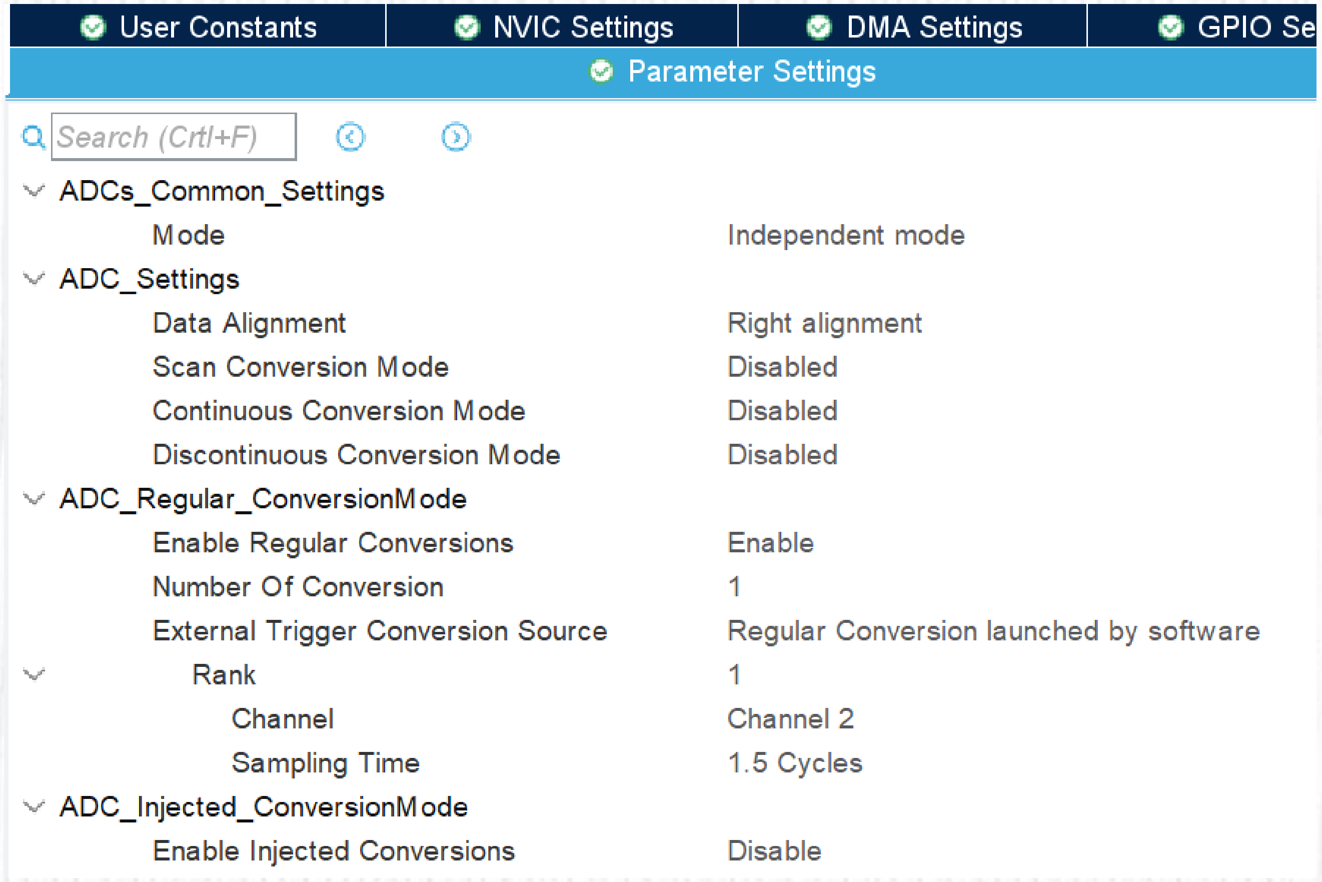
- [ ] IN0
- [ ] IN1
- [x] IN2
- [ ] IN3
- [ ] IN4

## 3. How to Program

- Some API of HAL we used
- ADC conversion by polling(轮询):
- Activate the ADC peripheral and start conversions using function **HAL_ADC_Start()**
- Wait for ADC conversion completion using function **HAL_ADC_PollForConversion()** (or for injected group: HAL_ADCEx_InjectedPollForConversion() )
- Retrieve conversion results using function **HAL_ADC_GetValue()** (or for injected group: HAL_ADCEx_InjectedGetValue() )
- Stop conversion and disable the ADC peripheral using function **HAL_ADC_Stop()**

## 3. How to Program

- Configure ADC1
  - Single channel, single conversion mode
  - In single conversion mode, ADC only does one conversion, so we need to restart conversion if we want to update the measurements



| | |
|---|---|
| ADCs_Common_Settings | |
| Mode | Independent mode |
| ADC_Settings | |
| Data Alignment | Right alignment |
| Scan Conversion Mode | Disabled |
| Continuous Conversion Mode | Disabled |
| Discontinuous Conversion Mode | Disabled |
| ADC_Regular_ConversionMode | |
| Enable Regular Conversions | Enable |
| Number Of Conversion | 1 |
| External Trigger Conversion Source | Regular Conversion launched by software |
| Rank | 1 |
| Channel | Channel 2 |
| Sampling Time | 1.5 Cycles |
| ADC_Injected_ConversionMode | |
| Enable Injected Conversions | Disable |

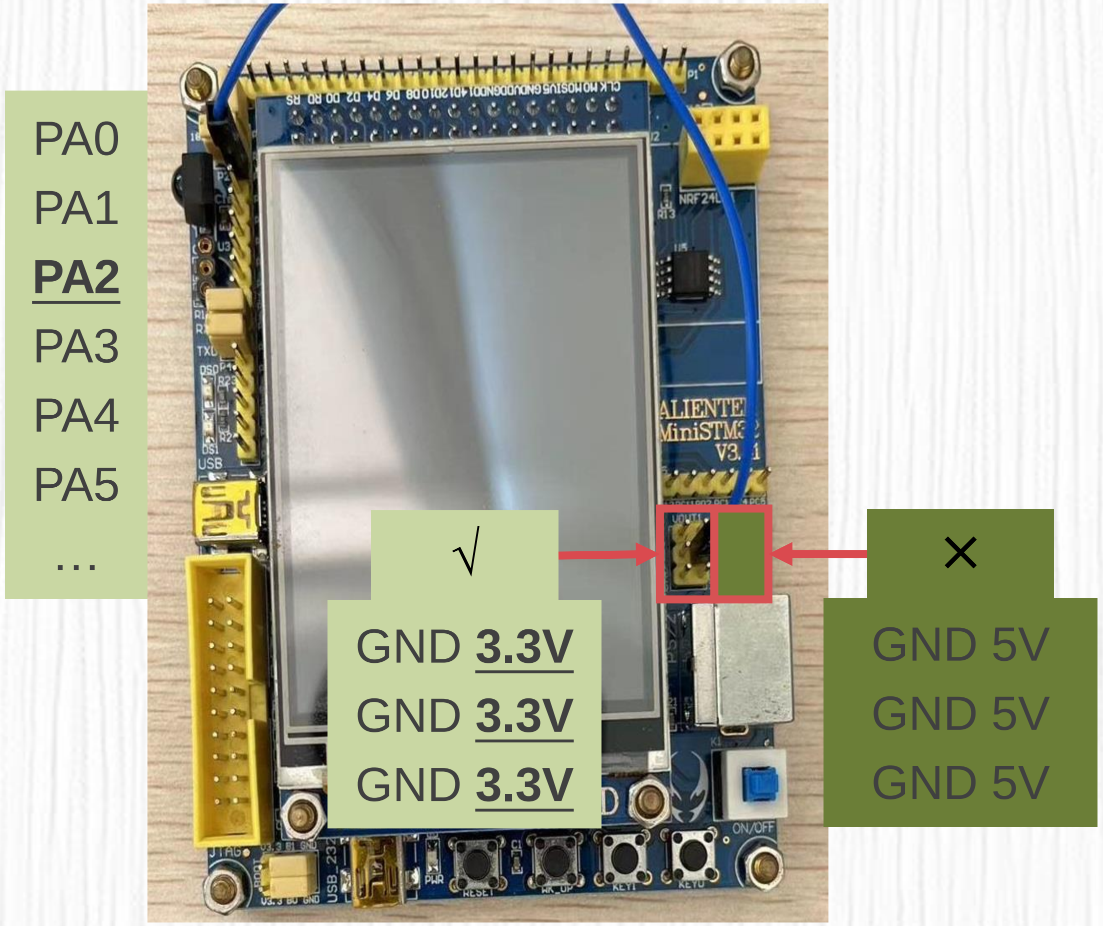
## 3. How to Program

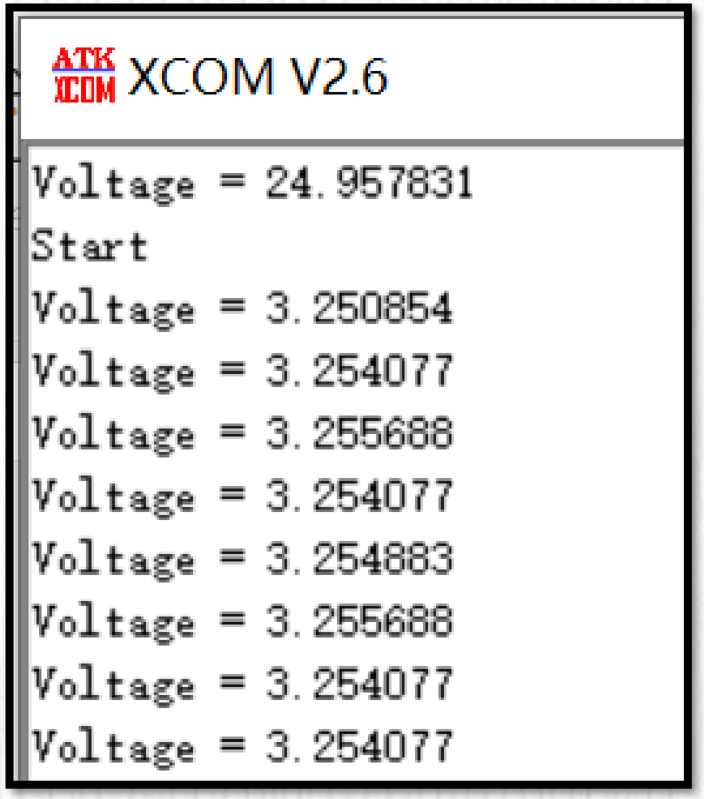
- Single channel, single conversion mode

```
// in main.c file
int main(void){
        //……
        uint16_t raw;
        char msg[20];
        double vol;
        HAL_UART_Transmit(&huart1, "Start\r\n", 7, HAL_MAX_DELAY);
        while (1){
                HAL_ADC_Start(&hadc1);
                // Wait for regular group conversion to be completed
                HAL_ADC_PollForConversion(&hadc1, HAL_MAX_DELAY);
                raw = HAL_ADC_GetValue(&hadc1);  // Get ADC value
                vol = (double)raw * (3.3/4096);  // the voltage should be raw * (3.3/4096)(12 bits)
                sprintf(msg, “Voltage = %f\r\n", vol);  // Convert to string and print
                HAL_UART_Transmit(&huart1, (uint8_t*)msg, strlen(msg), HAL_MAX_DELAY);
        }
}
```

## 3. How to Program

- Harware connection and result (V3)



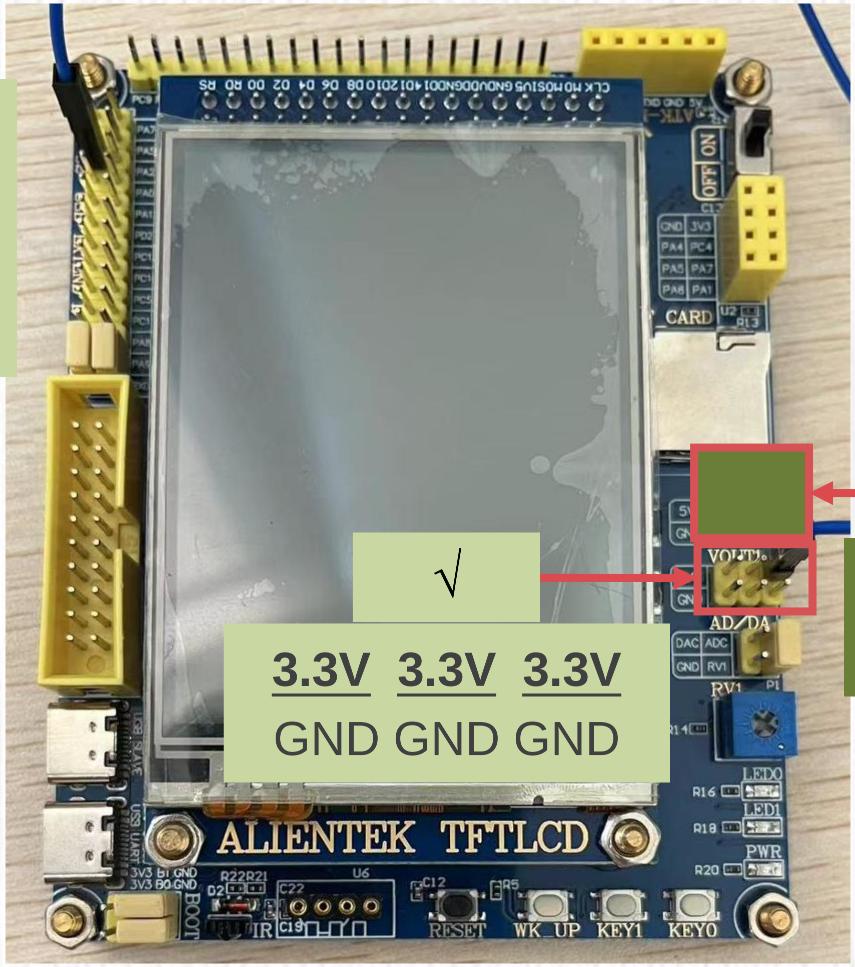


```
Voltage = 24.957831
Start
Voltage = 3.250854
Voltage = 3.254077
Voltage = 3.255688
Voltage = 3.254077
Voltage = 3.254883
Voltage = 3.255688
Voltage = 3.254077
Voltage = 3.254077
```

## 3. How to Program

- Harware connection and result (V4)

PA7 PA6
PA5 PA3
**PA2** PA1
…

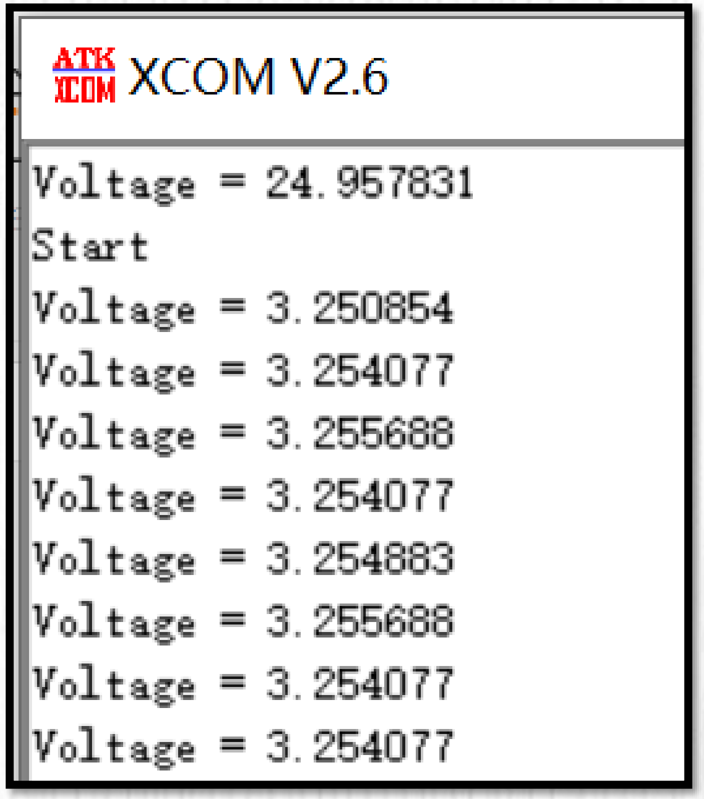


√

**3.3V** **3.3V** **3.3V**
GND GND GND

×

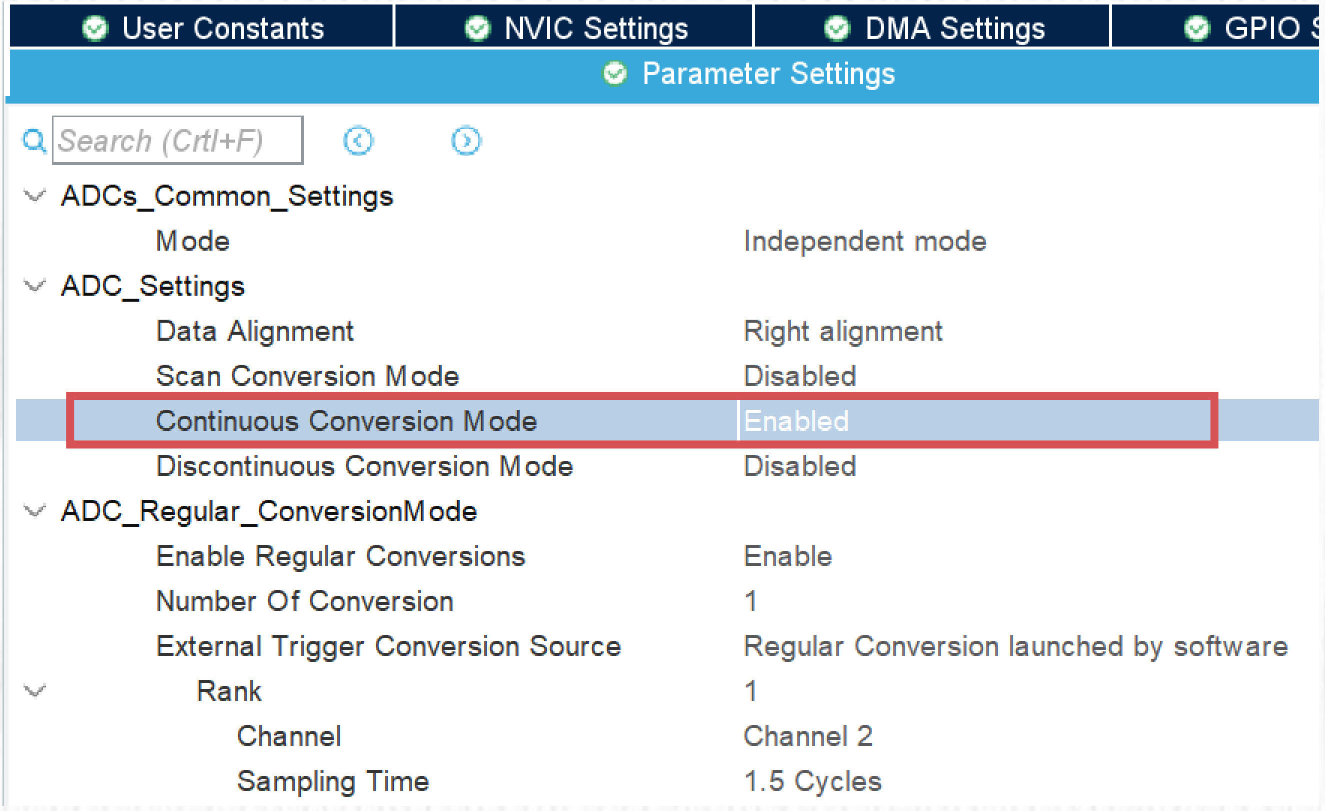
5V 5V 5V
GND GND GND



```
Voltage = 24.957831
Start
Voltage = 3.250854
Voltage = 3.254077
Voltage = 3.255688
Voltage = 3.254077
Voltage = 3.254883
Voltage = 3.255688
Voltage = 3.254077
Voltage = 3.254077
```

## 3. How to Program

- Configure ADC1
  - Single channel, continuous conversion mode
  - In continuous conversion mode, ADC starts another conversion as soon as it finishes one, so we don't need to restart the conversion.



| | |
|---|---|
| ADCs_Common_Settings | |
| Mode | Independent mode |
| ADC_Settings | |
| Data Alignment | Right alignment |
| Scan Conversion Mode | Disabled |
| Continuous Conversion Mode | Enabled |
| Discontinuous Conversion Mode | Disabled |
| ADC_Regular_ConversionMode | |
| Enable Regular Conversions | Enable |
| Number Of Conversion | 1 |
| External Trigger Conversion Source | Regular Conversion launched by software |
| Rank | 1 |
| Channel | Channel 2 |
| Sampling Time | 1.5 Cycles |

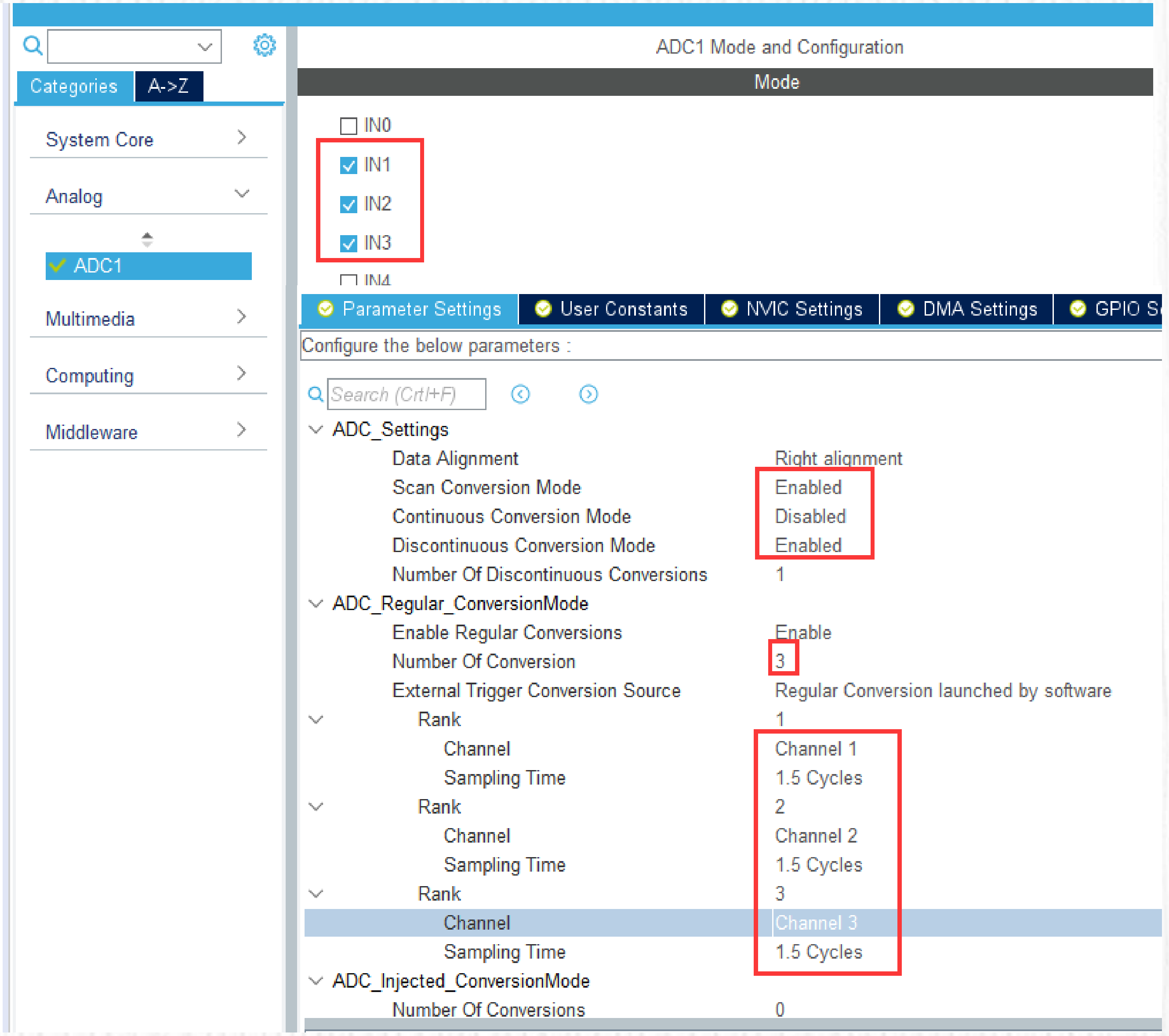
## 3. How to Program

- Single channel, continuous conversion mode

```
// in main.c file
int main(void){
	//……
	uint16_t raw;
	char msg[20];
	double vol;
	HAL_UART_Transmit(&huart1, "Start\r\n", 7, HAL_MAX_DELAY);
	HAL_ADC_Start(&hadc1);
	while (1){
		// Wait for regular group conversion to be completed
		HAL_ADC_PollForConversion(&hadc1, HAL_MAX_DELAY);
		raw = HAL_ADC_GetValue(&hadc1);  // Get ADC value
		vol = (double)raw * (3.3/4096);  // the voltage should be raw * (3.3/4096)(12 bits)
		sprintf(msg, “Voltage = %f\r\n", vol);  // Convert to string and print
		HAL_UART_Transmit(&huart1, (uint8_t*)msg, strlen(msg), HAL_MAX_DELAY);
	}
}
```

# 3. How to Program

- Configure ADC1
  - Multiple channels, single conversion mode
  - If we want to get the measurements of multiple channels and don't want to use DMA (Direct Memory Access) and interrupt, only single conversion mode can be used.



☐ IN0
☑ IN1
☑ IN2
☑ IN3
☐ IN4

Configure the below parameters :

| | |
|---|---|
| ADC_Settings | |
| Data Alignment | Right alignment |
| Scan Conversion Mode | Enabled |
| Continuous Conversion Mode | Disabled |
| Discontinuous Conversion Mode | Enabled |
| Number Of Discontinuous Conversions | 1 |
| ADC_Regular_ConversionMode | |
| Enable Regular Conversions | Enable |
| Number Of Conversion | 3 |
| External Trigger Conversion Source | Regular Conversion launched by software |
| Rank | 1 |
| Channel | Channel 1 |
| Sampling Time | 1.5 Cycles |
| Rank | 2 |
| Channel | Channel 2 |
| Sampling Time | 1.5 Cycles |
| Rank | 3 |
| Channel | Channel 3 |
| Sampling Time | 1.5 Cycles |
| ADC_Injected_ConversionMode | |
| Number Of Conversions | 0 |

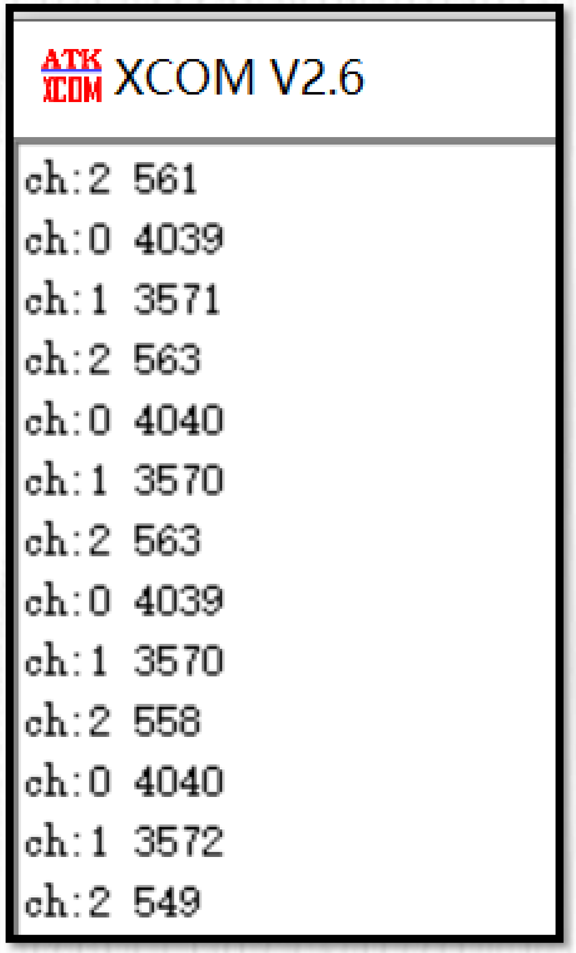
## 3. How to Program

- Multiple channels, single conversion mode

```
// in main.c file
while (1)
{
        /* USER CODE END WHILE */
        /* USER CODE BEGIN 3 */
        for (int i = 0; i < 3; i++)
        {
                HAL_ADC_Start(&hadc1);
                HAL_ADC_PollForConversion(&hadc1, HAL_MAX_DELAY);
                adcBuf[i]=HAL_ADC_GetValue(&hadc1);
                sprintf(msg, "ch:%d %d\r\n", i, adcBuf[i]);
                HAL_UART_Transmit(&huart1, (uint8_t*)msg, strlen(msg), HAL_MAX_DELAY);
        }
        HAL_Delay(500);
}
```

## 3. How to Program

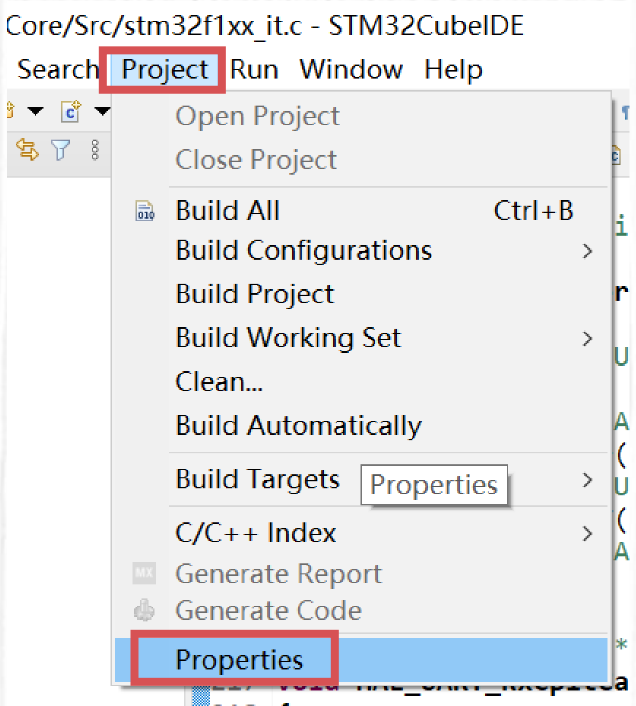
- Result

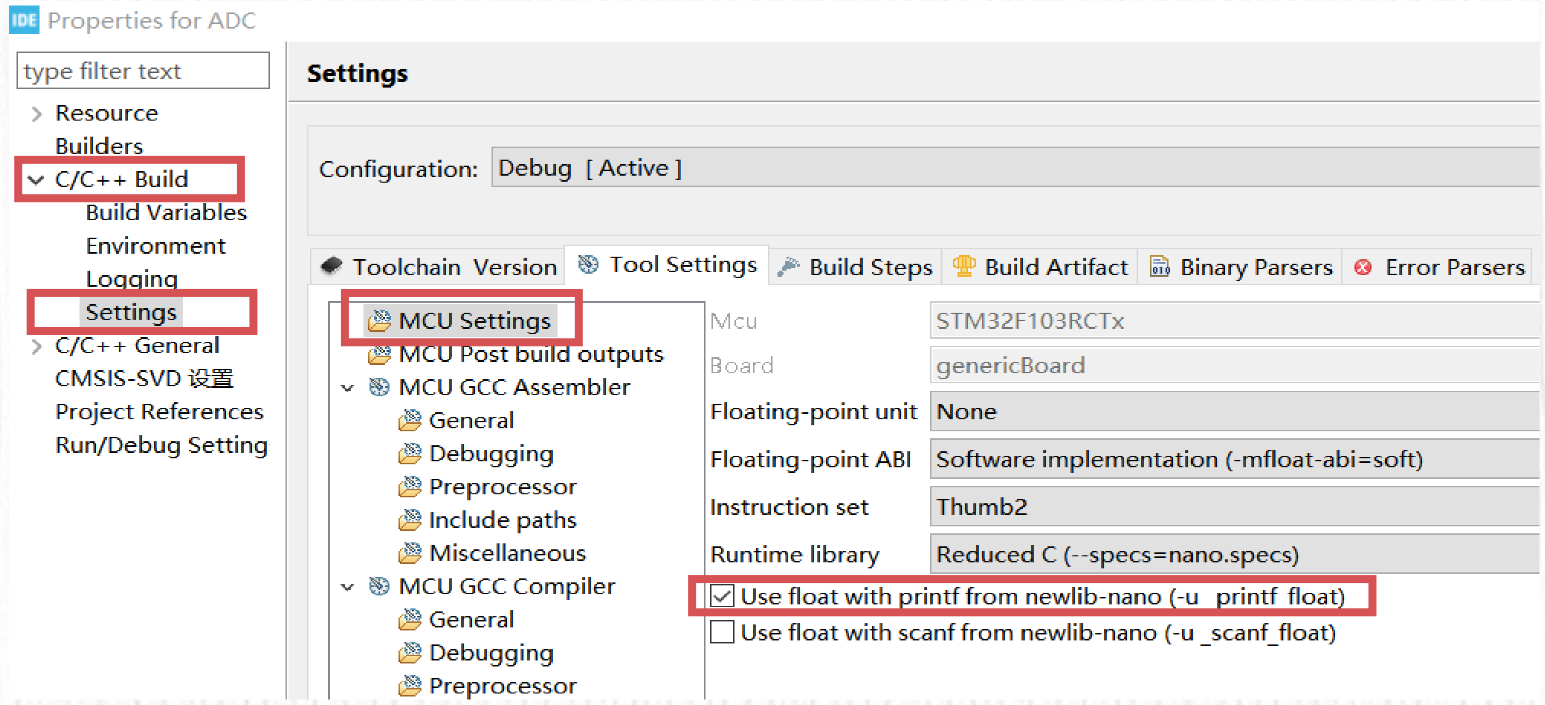

XCOM V2.6

```
ch:2 561
ch:0 4039
ch:1 3571
ch:2 563
ch:0 4040
ch:1 3570
ch:2 563
ch:0 4039
ch:1 3570
ch:2 558
ch:0 4040
ch:1 3572
ch:2 549
```

## 3. How to Program

- To print double or float value, you should change the settings as the following figure
- [Project] -> [Properties] -> [C/C++ Build] -> [Settings] -> [MCU Settings] -> [Use float with printf from newlib-nano(-u printf float)] -> [Apply and Close]





Configuration: Debug [ Active ]

| | |
|---|---|
| Mcu | STM32F103RCTx |
| Board | genericBoard |
| Floating-point unit | None |
| Floating-point ABI | Software implementation (-mfloat-abi=soft) |
| Instruction set | Thumb2 |
| Runtime library | Reduced C (--specs=nano.specs) |

[✓] Use float with printf from newlib-nano (-u _printf_float)

[ ] Use float with scanf from newlib-nano (-u _scanf_float)

南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

04

## Practice

## 4. Practice

- Configure ADC1 in different modes and run the demos on MiniSTM32 board.

- Use LCD screen to show the voltage value.
- Use CH5, CH6 and CH7 of ADC1 in multiple channels, single conversion mode to measure the voltage values continuously.