AN3023
Application note

Using the analog-to-digital converter of the STM8A microcontroller

Introduction

The purpose of this application note is to explain how to use the analog-to-digital converter implemented in the STM8A microcontroller family. The document provides useful information on how to configure the ADC registers and microcontroller resources and use the analog-to-digital converter in different modes.

The STM8A firmware library, containing source code of all the examples described in this application note, can be downloaded from the STMicroelectronics website: *www.st.com.*

Contents

1 ADC hardware description

1.1 General properties

STM8A family microcontrollers include an analog-to-digital converter which has up to 16 multiplexed inputs. The ADC resolution is 10 bits. The number of external analog inputs depends on the package size of the particular STM8A family device.

The ADC is a successive approximation analog-to-digital converter. Conversion can be performed in continuous mode or single mode. The digital result is then stored in registers. Because the ADC resolution is 10 bits and the ADC data register length is 16 bits, the analog-to-digital converter result is stored in two 8-bit registers, and the data is either right or left aligned (this is selectable).

The event used to start conversion can be generated by software or by the STM8A microcontroller's internal timer 1 and timer 2[a] (any type of timer event can be used). The start of A/D conversion can also be triggered by an external pin.

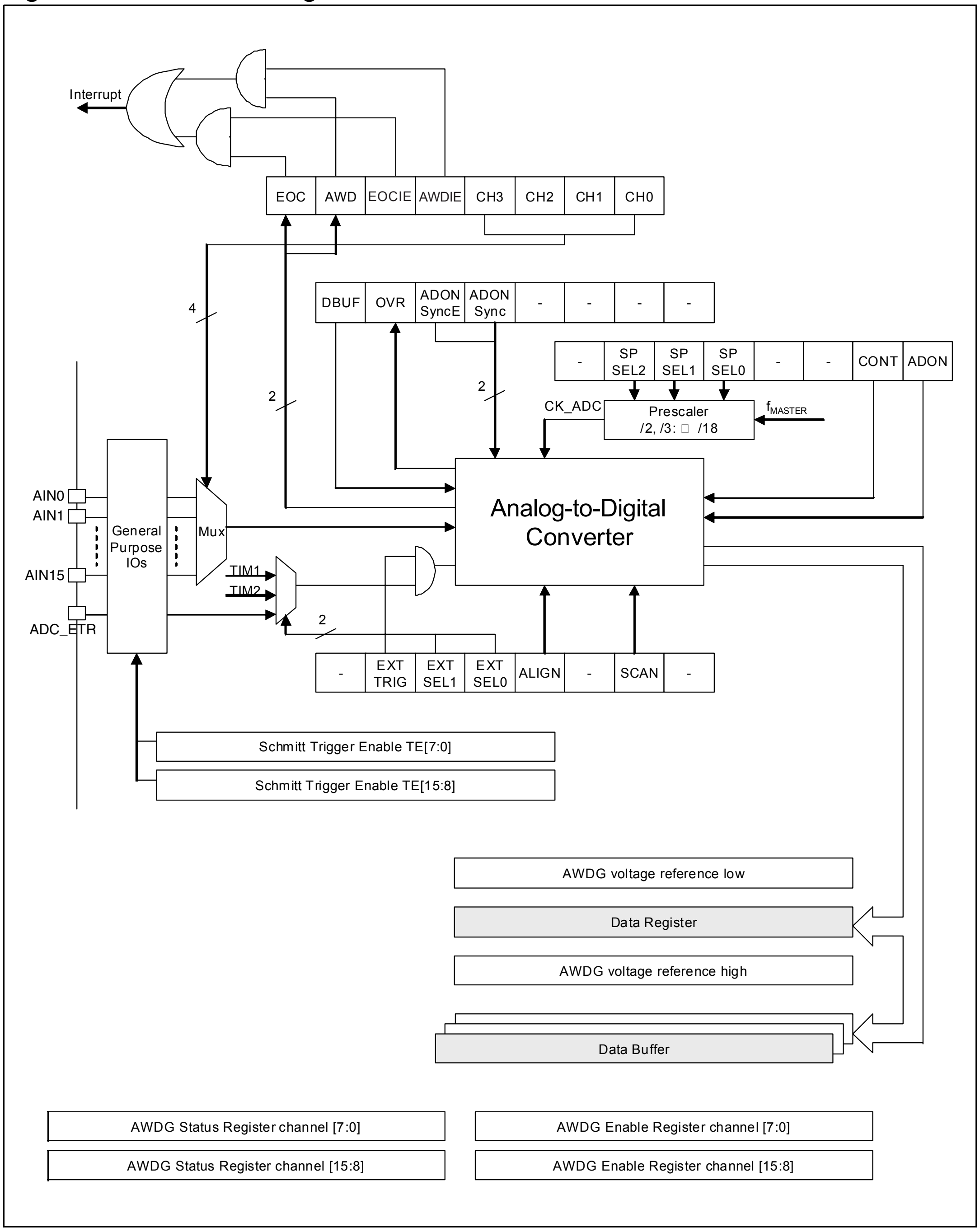
The reference voltage for the analog-to-digital converter depends on the package. It comes either from an external source—in this case the reference must be connected to two external pins—or is connected internally to analog power supply pins. The value of this reference voltage is limited to within the range from 2.75 V up to the voltage applied on the V_{DDA} input. The measured voltage must be between V_{REF+} and V_{REF-}. Resolution can be increased by so-called analog zooming—using a smaller reference voltage around the range of values to be measured.

If needed, an interrupt can be generated at the end of conversion and if an analog watchdog event occurred.

The analog-to-digital converter is driven by a clock derived from the MCU master clock through a programmable divider. This allows you to select the ADC clock speed according to your application requirements.

a. Trigger feature available on selected devices

Figure 1. ADC block diagram


Interrupt
EOC
AWD
EOCIE
AWDIE
CH3
CH2
CH1
CH0
4
DBUF
OVR
ADON SyncE
ADON Sync
-
2
2
-
SP SEL2
SP SEL1
SP SEL0
CONT
ADON
CK_ADC
Prescaler /2, /3: /18
f_{MASTER}
AIN0
AIN1
AIN15
ADC_ETR
General Purpose IOs
Mux
TIM1
TIM2
Analog-to-Digital Converter
2
EXT TRIG
EXT SEL1
EXT SEL0
ALIGN
SCAN
Schmitt Trigger Enable TE[7:0]
Schmitt Trigger Enable TE[15:8]
AWDG voltage reference low
Data Register
AWDG voltage reference high
Data Buffer
AWDG Status Register channel [7:0]
AWDG Status Register channel [15:8]
AWDG Enable Register channel [7:0]
AWDG Enable Register channel [15:8]

1. Refer to STM8A reference manual for ADC registers bit description

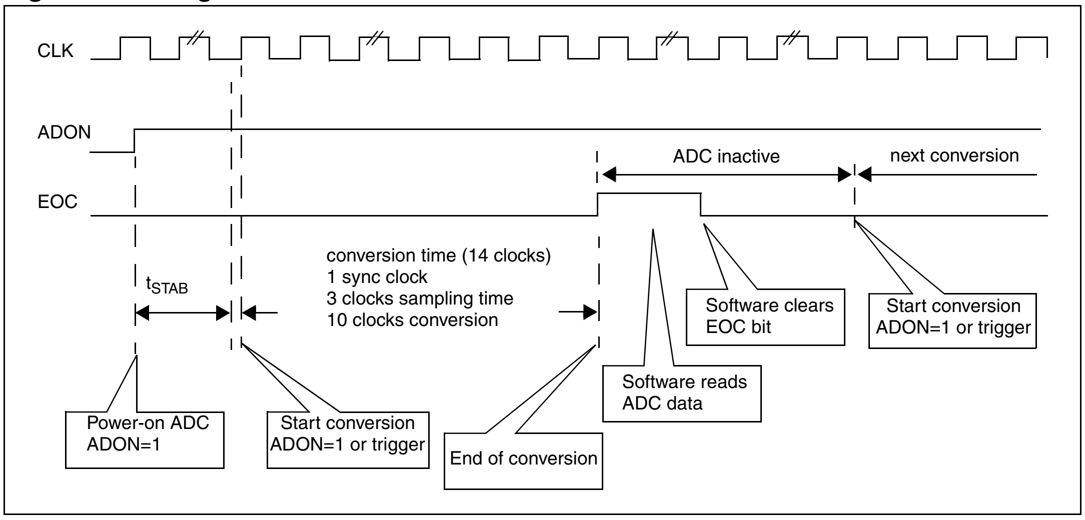
1.2 ADC operation

The analog-to-digital converter supports two basic conversion modes: single mode and continuous mode.

Figure 2 and *Figure 3* describe single and continuous mode analog-to-digital converter operation.

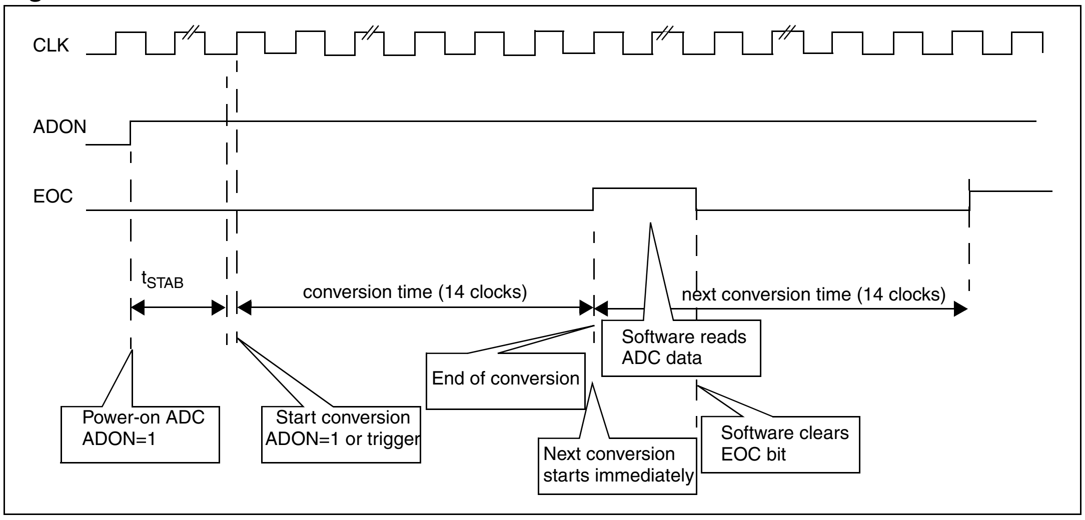
In single conversion mode, the analog-to-digital converter performs only one conversion on the selected channel.

Figure 2. Single conversion mode



In continuous conversion mode, the analog-to-digital converter starts another conversion as soon as the previous one is concluded.

Figure 3. Continuous conversion mode

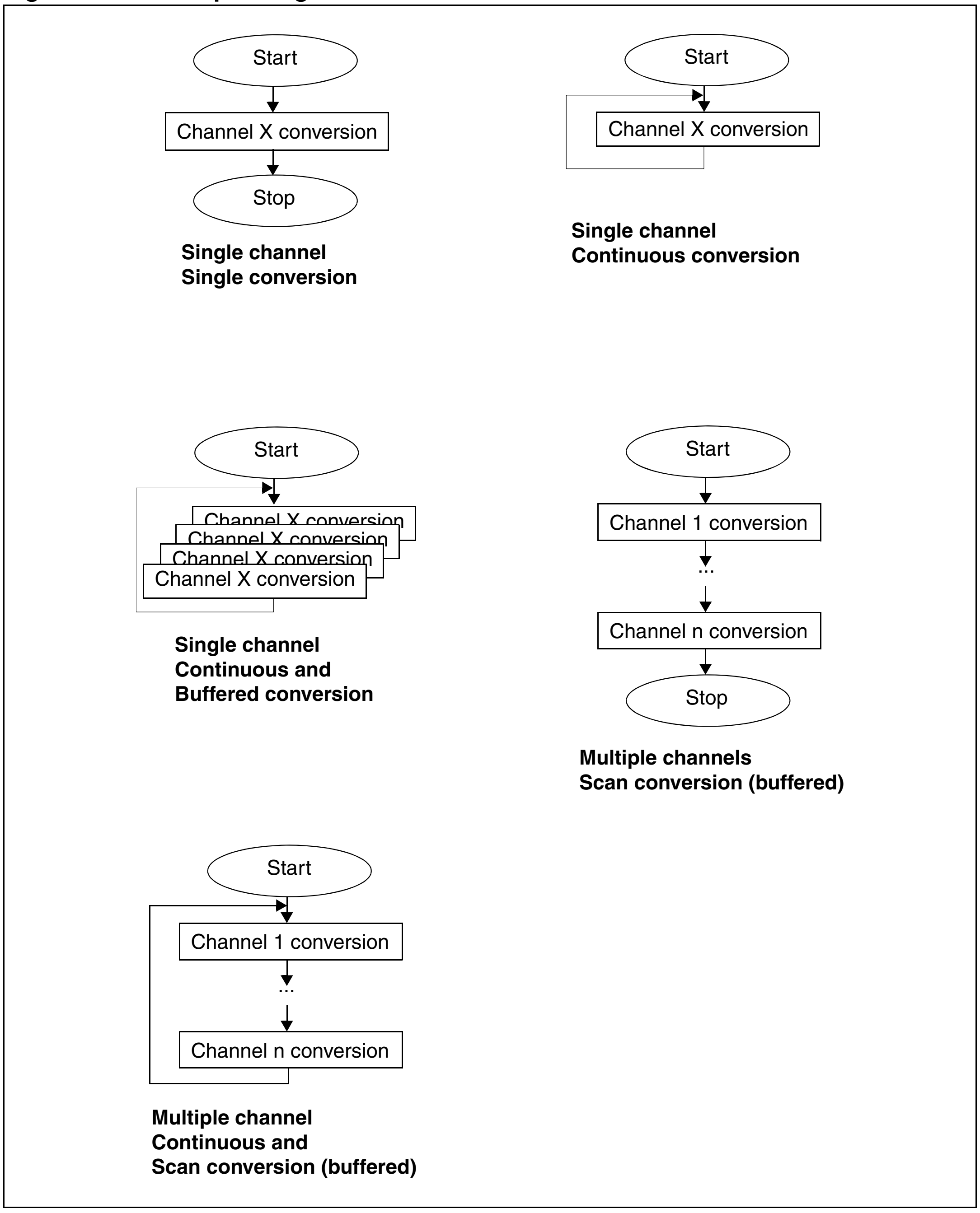

CLK
ADON
EOC
tSTAB
conversion time (14 clocks)
next conversion time (14 clocks)
Software reads ADC data
End of conversion
Power-on ADC ADON=1
Start conversion ADON=1 or trigger
Next conversion starts immediately
Software clears EOC bit

As shown in these figures, after ADC power on, the analog-to-digital converter needs a stabilization time t_{STAB} (equivalent to one conversion time t_{CONV}) before it starts converting accurately. For subsequent conversions there is no stabilization delay and ADON needs to be set only once.

The ADC conversion time takes 14 clock cycles. In addition to the basic conversion modes, some packages have extended features for scan mode and buffered continuous mode.

Figure 4 summarizes all the possible analog-to-digital converter operating modes.

Figure 4. ADC operating modes



2 Summary of the features

The STM8A family contains two types of analog-to-digital converter which are named ADC and ADC2. Depending on the device, it may contain both (ADC and ADC2) or only one (ADC).

There are two variants of the first analog-to-digital converter ADC:

1. With standard features
2. With extended features for scan mode, buffered continuous mode and analog watchdog

The second analog-to-digital converter, ADC2, contains a features subset of the first ADC module.

Refer to the STM8A datasheet for more information on the available features.

Table 1. ADC and ADC2 features on different devices(1)

Module	Features			
	Parameter	**32 KB die based(2)**	**128 KB die based(2)**	**256 KB die based(2)**
ADC	Resolution	10-bit		
	Programmable prescaler	f_{MASTER} divided by 2 to 18		
	Operating modes	Single or continuous mode		
		Scan and buffered continuous mode	—	Scan and buffered continuous mode
	Clock speed	f_{ADC} = 1 to 4 MHz		
	ADC input range	$V_{SSA} = V_{IN} = V_{DDA}$		
	Conversion speed	min. 3.5 µs @ 4 MHz f_{ADC}		
	Number of input channels	up to 10	up to 16	up to 16
	Triggers	Timer 1 and external ADC_ETR pin	External ADC_ETR pin	Timer 1, Timer 2 and external ADC_ETR pin
	Data format	left or right alignment		
	ADC interrupt	EOC and AWD	EOC	EOC and AWD
	Analog watchdog	Yes	—	Yes
	Schmitt trigger enable/disable	Yes	Yes	Yes
	Synchronization mechanism between ADC and ADC2	—	—	Yes
	Analog zooming	—	Through dedicated VREF pins	Through dedicated VREF pins

Table 1. ADC and ADC2 features on different devices[1] (continued)

Module	Features			
	Parameter	32 KB die based[2]	128 KB die based[2]	256 KB die based[2]
ADC2	Resolution	—	—	10-bit
	Programmable prescaler	—	—	f_{MASTER} divided by 2 to 18
	Operating modes	—	—	Single or continuous mode
	Clock speed	—	—	f_{ADC} = 1 to 4 MHz
	ADC input range	—	—	$V_{SSA} = V_{IN} = V_{DDA}$
	Conversion speed	—	—	min. 3.5 µs @ 4 MHz f_{ADC}
	Number of input channels	—	—	12
	Triggers	—	—	Timer1 and Timer2
	Data format	—	—	left or right alignment
	ADC interrupt	—	—	EOC and AWD
	Analog watchdog	—	—	Yes
	Schmitt trigger enable/disable	—	—	Yes
	Synchronization mechanism between ADC and ADC2	—	—	Yes
	Analog zooming	—	—	—

1. For feature details, refer to the STM8A reference manual and the device datasheet.
2. Die information is stored in Chip ID registers. Refer to the reference manual for further details.

3 Configuring ADC registers for conversion

3.1 ADC setup

Before starting an ADC conversion, ADC setup is done using the following steps:

- ADC clock selection (define conversion speed)
- Channel selection
- Power on the ADC (set ADON bit)
- Wait for 14 ADC clock cycles (14 * t_{ADC})
- Conversion mode selection (single or continuous)
- Trigger source selection and only after trigger enable
- Apply trigger source or set ADON bit

3.1.1 ADC clock

The clock supplied to the ADC peripheral is the prescaled f_{MASTER} clock. The prescaler is applied on bit SPSEL[2:0] in the ADC_CR1 register and it can be selected from $f_{MASTER}/2$ to $f_{MASTER}/18$.

The prescaler selection must ensure a maximum 4 MHz ADC frequency.

ADC conversion time depends on the ADC clock frequency, the sampling time duration and the ADC resolution. The minimum ADC conversion time is 3.5 µs @ 4 MHz ADC clock frequency.

Note: It is recommended to change bits SPSEL[2:0] when the ADC is in power-down mode. This is because, internally, there may be a glitch in the clock during this change. If this change is made when ADC is not in power-down, the user is required to ignore the first converted result.

3.1.2 ADC on/off control (power on)

The ADC can be powered on by setting the ADON bit in the ADC_CR1 register. When the ADON bit is set for the first time, it wakes up the ADC from power-down mode.

After power ON, wait for stabilization time and then, to start a conversion, either set the ADON bit again in the ADC_CR1 register or use an external trigger source (timers or ADC_ETR pin).

At the end of conversion, the ADC remains powered on and the ADON bit has to be set only once to start the next conversion. If the ADC is not used for a long time, it is recommended to switch it off to decrease power consumption. This is done by clearing the ADON bit.

When the ADC is powered on, the output stage of the selected channel is disabled; therefore it is recommended to select the channel first before powering on the ADC.

Note: If any bit of the ADC_CR1 register other than ADON is changed at the same time, the conversion is not triggered. This is to prevent triggering an erroneous conversion.

Moreover, be aware that any bit manipulation instruction on the ADC_CR1 register can start a new conversion due to the fact that the ADON bit is read as ‘1’ and then written back as ‘1’ by this operation.

3.1.3 Channel selection

There are up to 16 input channels in ADC and up to 12 input channels in ADC2 (the maximum number of channels per ADC peripheral depends on the MCU package size). External input channels can be selected by bits CH[3:0] in the ADC_CSR register. If the channel is changed during an ongoing conversion, the current conversion is reset and a new start pulse is sent to ADC.

3.1.4 Conversion on external trigger

Conversion can also be started by an external trigger event after ADC power ON.

The external trigger source can be a timer[(b)] TRGO event or a rising edge on the external ADC_ETR[(b)] pin. Bit EXTTRIG is set in the ADC_CR2 register to enable the conversion by external trigger and the bits EXTSEL[1:0] select the trigger source as either the timer or the external trigger pin.

The different software sequences (shown below) to select the trigger source and enable the external trigger depend on the ADC state (powered-ON or powered-OFF).

Case 1: ADC is in powered-ON state

```
ADC_CR2 = value1;   Select the source (EXTSEL[1:0] bits)
ADC_CR2 |= 0x40;   Enable external trigger (EXTTRIG bit)
```

Case 2: ADC is in powered-OFF state

When the ADC is in powered-OFF state, the user is free to choose between the above software sequence, that is, for ADC powered-ON state or the sequence below:

```
ADC_CR2 = value;   Select the source and enable the external trigger
```

3.1.5 Conversion modes

Single conversion mode

In single conversion mode, the ADC performs one conversion on the channel selected by bits CH[3:0] in the ADC_CSR register. At the end of conversion, the EOC bit is set and the result can be read from the registers ADC_DRH and ADC_DRL. The interrupt is generated if the EOC interrupt is enabled in the ADC_CSR register.

Example 1

- f_{MASTER} = 16 MHz
- f_{ADC} = 4 MHz
- Channel = AIN1
- Single conversion mode, left alignment mode
- Starting ADC by software

b. Refer to the datasheet for the details on the trigger source available.

For *Example 1*, the ADC registers should be configured as follows:

ADC_CR1:SPSEL[2:0] = 010 $\rightarrow$ f_{ADC} = 4 MHz (that is, $f_{MASTER}/4$)

ADC_CSR:CH[3:0] = 0001 $\rightarrow$ AIN1 channel selected

ADC_CR1:ADON = 1 $\rightarrow$ ADC power on

Wait (14/f_{ADC}) $\rightarrow$ stabilization time before starting the conversion

ADC_CR1:CONT = 0 $\rightarrow$ single conversion mode

Note: *Do not perform the operation "ADC_CR1:CONT = 0 $\rightarrow$ single conversion mode" if the CONT bit is already 0 (see note in Section 3.1.2).*

ADC_CR2:ALIGN = 0 $\rightarrow$ left alignment

ADC_CR1:ADON = 1 $\rightarrow$ starting the conversion by software

Wait (ADC_CSR:EOC = 1) $\rightarrow$ end of conversion

See the result of the conversion in ADC data registers ADC_DRH and ADC_DRL.

Continuous conversion mode

In continuous conversion mode, the ADC starts another conversion as soon as it finishes the last one. A new conversion starts automatically. For this conversion mode, set the CONT bit to '1' in the ADC_CR1 register.

Example 2

- f_{MASTER} = 4 MHz
- f_{ADC} = 1 MHz
- Channel = AIN2
- Continuous conversion model, right alignment
- Starting ADC by external trigger event on ADC_ETR pin

For *Example 2*, the ADC registers should be configured as follows:

ADC_CR1:SPSEL[2:0] = 010 $\rightarrow$ f_{ADC} = 1 MHz (that is, $f_{MASTER}/4$)

ADC_CSR:CH[3:0] = 0010 $\rightarrow$ AIN2 channel is selected

ADC_CR1:ADON = 1 $\rightarrow$ ADC power on

Wait (14/f_{ADC}) $\rightarrow$ stabilization time before starting the conversion

ADC_CR1:CONT = 1 $\rightarrow$ continuos conversion mode

ADC_CR2:ALIGN = 1 $\rightarrow$ left alignment

ADC_CR2:EXTSEL[1:0] = 01 $\rightarrow$ External trigger on ADC_ETR

ADC_CR2:EXTTRIG = 1 $\rightarrow$ conversion on external event is enabled

Rising edge on ADC_ETR pin

Wait (ADC_CSR:EOC = 1) $\rightarrow$ end of conversion

If you want to save the result of the current conversion, save the values of data register ADC_DRH and ADC_DRL before starting the next conversion.

Practical use

Continuous conversion mode is useful for collecting a set of analog values from an external analog signal (for example, to read an AC voltage waveform).

Single scan mode

Single scan mode is started by triggering a conversion while the SCAN bit is set in the ADC_CR2 register and the CONT bit is 0 in the ADC_CR1 register. The scan is performed from channel 0 up to the channel n programmed in bits CH[3:0] in register ADC_CSR.

Do not clear the SCAN bit while the conversion sequence is in progress. Single scan mode can be stopped immediately by clearing the ADON bit in the ADC_CR1 register.

Example 3

- f_{MASTER} = 4 MHz
- f_{ADC} = 1 MHz
- Single scan mode from AIN0 to AIN5 channel
- Right alignment
- Starting ADC by software

For *Example 3*, the ADC registers should be configured as follows:

ADC_CR1:SPSEL[2:0] = 010 $\rightarrow$ f_{ADC} = 1 MHz (that is, $f_{MASTER}/4$)

ADC_CSR:CH[3:0] = 0101 $\rightarrow$ select the scan from AIN0 to AIN5

ADC_CR1:ADON = 1 $\rightarrow$ ADC power on

Wait (14/f_{ADC}) $\rightarrow$ stabilization time before starting the conversion

ADC_CR2:SCAN = 1 $\rightarrow$ scan mode enable

ADC_CR2:ALIGN = 1 $\rightarrow$ right alignment

ADC_CR1:ADON = 1 $\rightarrow$ starting the conversion by software

Wait (ADC_CSR:EOC = 1) $\rightarrow$ end of conversion

Read the result of the conversion in ADC_DBxRH and ADC_DBxRL data buffer registers, where x = 0..5.

Practical use

Single scan mode is useful for collecting different analog values from external sensors. This conversion mode permits a faster conversion operation than the simple single conversion mode (which requires conversion to be performed channel per channel).

Continuous scan mode

Continuous scan mode is started by triggering a conversion while the SCAN bit is set in the ADC_CR2 register and CONT bit is set in the ADC_CR1 register. The scan is performed from channel 0 up to the channel n programmed in bits CH[3:0], and, when the EOC bit is set, a new scan is started automatically. The data buffer registers must be read before the completion of new scan sequence, otherwise the data buffers are overwritten.

Do not clear the SCAN bit while scan conversion is in progress. Continuous scan mode can be stopped immediately by clearing the ADON bit in the ADC_CR1 register. Alternatively, if the CONT bit is cleared while conversion is ongoing, conversion stops the next time the last channel has been converted.

Example 4

- f_{MASTER} = 4 MHz
- f_{ADC} = 2 MHz
- Continuous scan from AIN0 to AIN3 channel
- Right alignment
- Starting ADC by TIM1

For *Example 4*, the ADC registers should be configured as follows:

ADC_CR1:SPSEL[2:0] = 000 $\rightarrow$ f_{ADC} = 2 MHz (that is, $f_{MASTER}/2$)

ADC_CSR:CH[3:0] = 0011 $\rightarrow$ channel from AIN0 to AIN3

ADC_CR1:ADON = 1 $\rightarrow$ ADC power on

Wait (14/f_{ADC}) $\rightarrow$ stabilization time before starting the conversion

ADC_CR1:CONT = 1 $\rightarrow$ continuous conversion mode

ADC_CR2:SCAN = 1 $\rightarrow$ scan mode enable

ADC_CR2:ALIGN = 1 $\rightarrow$ right alignment

ADC_CR2:EXTSEL[1:0] = 00 $\rightarrow$ Internal TIM1 TRGO event

ADC_CR2:EXTTRIG = 1 $\rightarrow$ conversion on external event is enabled

Timer initialization for trigger output: The timer has to be configured in PWM mode, with a period of at least 7 µs, in order to guarantee a good ADC conversion.

f_{MASTER} = 4 MHz, Prescaler = 1, TIM1 counter clock = 2 MHz

PWM signal @50 kHz (period = 20 µs)

TIM1_SMCR:MSM = 1 $\rightarrow$ TIM1 configured as master mode

TIM1_CR2:MMS[6:4] = 100 $\rightarrow$ OC1REF signal is used ad trigger output (TRGO)

TIM1_CR1:CEN = 1 $\rightarrow$ Timer counter enable

ADC conversion will start at each rising edge event of OC1REF signal

Wait (ADC_CSR:EOC = 1) $\rightarrow$ end of conversion

Read the result of the current conversion from ADC_DBxRH and ADC_DBxRL registers (x = 0..3) before of the end of next conversion. The OVR flag is set if one of the buffer registers is overwritten.

Note: 1 *In scan mode, do not use the bit manipulation instruction (BRES) to clear the EOC flag because it performs a read-modify-write on the whole ADC_CSR register, reading the current channel number from the CH[3:0] register and writing it back. This changes the last channel number for the scan sequence. The correct way to clear the EOC flag in continuous scan mode is to load a byte in the CSR register from a RAM variable, clearing the EOC flag and reloading the last channel number for the scan sequence.*

2 *When using scan mode, it is not possible to use channels AIN0 to AINn in output mode because the output stage of each channel is disabled when it is selected by the ADC multiplexer.*

Buffered continuous mode

The buffered mode is enabled in the continuous conversion mode if bit DBUF = 1 in the ADC_CR3 register. The data buffer registers are filled with the results for up to 16 consecutive conversions performed on a single channel.

Example 5

- f_{MASTER} = 2 MHz
- f_{ADC} = 0.5 MHz
- Buffered continuous mode for channel 3
- Left alignment
- Starting ADC by software

For *Example 5*, the ADC registers should be configured as follows:

ADC_CR1:SPSEL[2:0] = 010 $\rightarrow$ f_{ADC} = 0.5 MHz (that is, $f_{MASTER}/4$)

ADC_CSR:CH[3:0] = 0011 $\rightarrow$ AIN3 channel

ADC_CR1:ADON = 1 $\rightarrow$ ADC power on

Wait (14/f_{ADC}) $\rightarrow$ stabilization time before starting the conversion

ADC_CR1:CONT = 1 $\rightarrow$ continuos conversion mode

ADC_CR3:DBUF = 1 $\rightarrow$ data buffer enable

ADC_CR2:ALIGN = 0 $\rightarrow$ left alignment

ADC_CR1:ADON = 1 $\rightarrow$ starting the conversion of the channel 3 by software

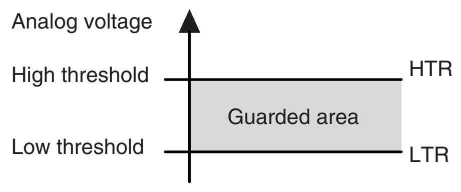
Wait (ADC_CSR:EOC = 1) $\rightarrow$ end of conversion

The results of the 16 consecutive conversions should be read from on ADC_DBxRH and ADC_DBxRL registers (x = 0..15) before the end of the next conversions. The OVR flag is set if one of the buffer registers is overwritten.

Analog watchdog

In continuous mode, the analog watchdog is always enabled. The AWD analog watchdog flag is set if the analog voltage converted by ADC is out of the guard band limited by low and high threshold values. The low and high threshold values are programmed in the ADC_HTR and ADC_LTR registers.

Figure 5. Analog watchdog



The ADC_HTR and ADC_LTR values can be calculated by the formula

$$ADC\_xTR = \frac{V_{AIN}}{V_{REF+} - V_{REF-}} \cdot (2^N - 1),$$

where V_{AIN} is the higher or lower voltage value of the guarded area and N is the ADC resolution.

Example 6

- f_{MASTER} = 4 MHz
- f_{ADC} = 2 MHz
- Channel = AIN0 and AIN1
- Continuous scan mode, left alignment
- Analog watchdog:
  Monitoring AIN0 within 1.0 V and 4.0 V
  V_{REF+} = 5.0V, V_{REF-} = 0.0 V
- Starting ADC by software

For *Example 6*, the ADC registers should be configured as follows:

ADC_CR1:SPSEL[2:0] = 000 $\rightarrow$ f_{ADC} = 2 MHz (that is, $f_{MASTER}/2$)

ADC_CSR:CH[3:0] = 1100 $\rightarrow$ AIN0 to AIN12 channels

ADC_CR1:ADON = 1 $\rightarrow$ ADC power on

Wait (14/f_{ADC}) $\rightarrow$ stabilization time before starting the conversion

ADC_CR1:CONT = 1 $\rightarrow$ continuous conversion mode

ADC_CR2:SCAN = 1 $\rightarrow$ scan mode enable

ADC_HTRL = 0x03;

ADC_HTRH = 0xCC; $\rightarrow$ ADC high threshold is 819

ADC_LTRL = 0x01;

ADC_LTRH = 0x33; $\rightarrow$ ADC low threshold is 205

ADC_AWCRL = 0x01; watchdog enabled on AIN0

ADC_CR1:ADON = 1 $\rightarrow$ starting the conversion by software

Wait (ADC_CSR:AWD = 1) $\rightarrow$ Analog watchdog event has occurred

Read the result of the conversion in ADC_DBxRH and ADC_DBxRL data buffer registers, where x = 0..12. If any of the conversion happens out of high and low limit, the AWD flag in ADC_CSR register and the corresponding bits in the AWSRL register will be set to '1'.

Depending on whether or not the EOC interrupt, the AWD interrupt, or both, are enabled, there could be different combination of managing the analog watchdog. Please refer to the STM8A reference manual for details.

Note: *In the buffered continuous mode, the ADC1_AWCRx register indicates which data buffered has to be monitored.*

Synchronization between two ADCs

This feature can be applied only if the device contains both ADC modules (ADC and ADC2). The synchronization mechanism can only be used for single conversion and can be triggered either by software or by hardware.

To start the synchronous conversion, the ADSE bit must be set to ‘1’ in the CR3 registers of both ADC and ADC2.

When the ADS bit of the master ADC module is set, or another event triggers a conversion on the master ADC module, a conversion on the slave module (ADC2) is triggered synchronously.

Example 7

- f_{MASTER} = 4 MHz
- $f_{ADC} = f_{ADC2}$ = 2 MHz
- Channels: AIN2 on ADC and AIN1 on ADC2
- Left alignment
- Starting ADC and ADC2 by software

For *Example 7*, the ADC registers should be configured as follows:

//ADC2 setting - SLAVE

ADC2_CSR:CH[3:0] = 0001 → AIN1 channel

ADC2_CR1 ADON = 1; → ADC2 power on

Wait (14/f_{ADC}) → stabilization time before starting the conversion

//ADC setting - MASTER

ADC_CSR:CH[3:0] = 0010 AIN2 channel

ADC_CR1 ADON = 1; → ADC power on

Wait (14/f_{ADC}) → stabilization time before starting the conversion

ADC_CR3:ADSE = 1; //enable sync on ADC

ADC2_CR3:ADSE = 1; //enable sync on ADC2

ADC_CR3:ADS = 1; //ADS start trigger, parallel conversion is started

Note: *Before starting synchronization conversion, the two ADC modules have to be programmed with the same clock prescaler values. It is recommended to program the prescaler register before switching on the ADCs.*

3.2 Storing converted values

In the scan mode and buffered continuous mode, the converted ADC values are stored in the data buffer registers ADC_DBxRH and ADC_DBxRL, where x = 1..15.

In the normal mode, the converted ADC values are stored in the ADC data registers ADC_DRH and ADC_DRL. ADC_DRH and ADC_DRL also contain the last converted value in scan and buffered continuous mode.

Different data alignment modes (left or right) can be selected depending on the application. If right alignment mode is chosen, the LSB register (ADC_DRL) must be read before the MSB register (ADC_DRH), otherwise the values of register ADC_DRH are not updated. If left alignment mode is chosen, the MSB register (ADC_DRH) must be read before the LSB register (ADC_DRL), otherwise the values of register ADC_DRL are not updated. This prevents reading data register values from two different conversions. Therefore the reading order must be respected.

Practical use

Left alignment is useful for fast read access to the conversion result in 8-bit resolution.

Right alignment is useful for fast read access when measuring low level signals. In cases where the result does not affect the high order bits, you can read only the LSB byte.

3.3 ADC interrupt management

ADC interrupt sources are selectable from one or more of the following events:

- End of conversion (EOC) in nonbuffered mode—generated after each AD conversion
- End of conversion (EOC) in buffered mode—generated after data buffer registers are full
- Analog watchdog alert—when converted values reach the high or low threshold values

You can enable the ADC interrupt to give the main program fast asynchronous event notification at the end of conversion(s). This means the main program does not need to poll the ADC conversion status flags but only has to process the ADC data results (for example, after storing a set of conversions in a buffer). The reception of the ADC converted data values is performed by an interrupt routine in the background. In addition, the analog watchdog interrupt can be used for monitoring a given analog range on the measured channels.

Register settings:

ADC_CSR: AWDIE, EOCIE bits

4 Practical application

4.1 Areas of use

- Temperature measurement
  - Process calibration
  - Device thermal protection
  - Fan control
- Power supply measurement
  - Auto save configuration
  - Battery charging/protection
- Measurement of physical values in other types of applications
  - Automation—from sensors with analog outputs
  - Household—for example, automatic lights dimmers, weather stations, thermometers, security sensors
  - Industry—for example, lights, thermostats, humidity control
  - Electrical quantity measurements—for example, voltage, current, capacity, resistance

4.2 Hardware connection examples

Figure 6. Simple unipolar DC signal measurement

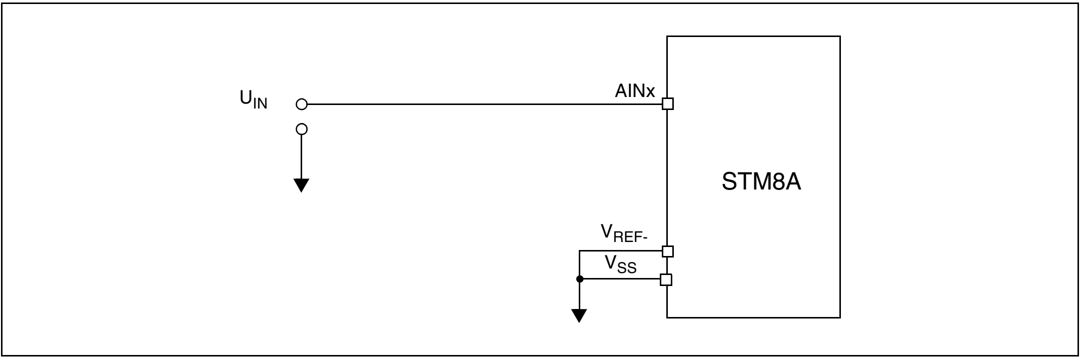


Figure 7. External preamplifier usage (with high impedance input)

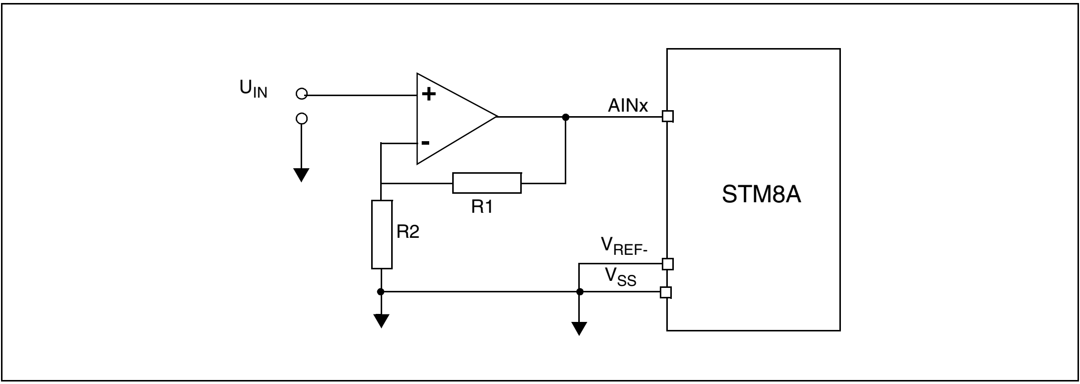


Figure 8. Differential DC signal measurement with preamplifier

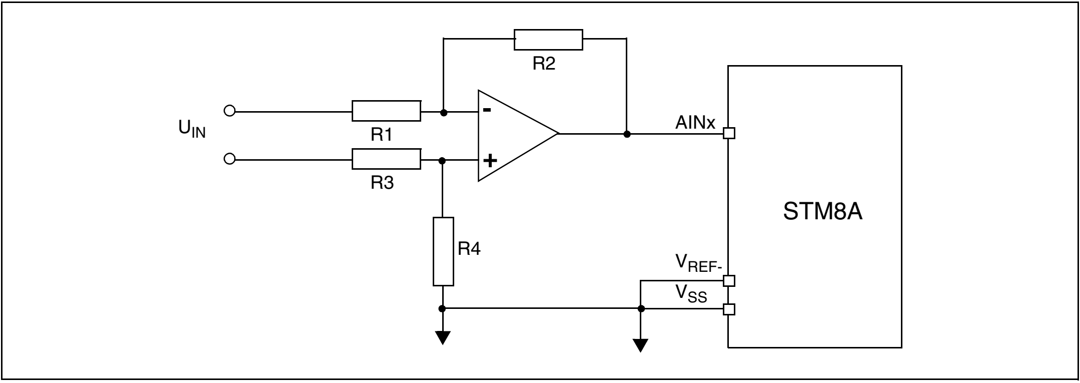


Figure 9. AC signal (FFT) measurement

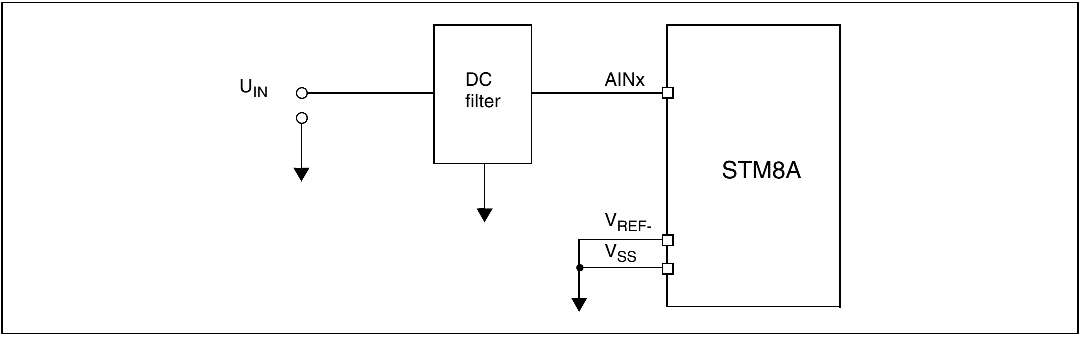


Figure 10. Analog zooming

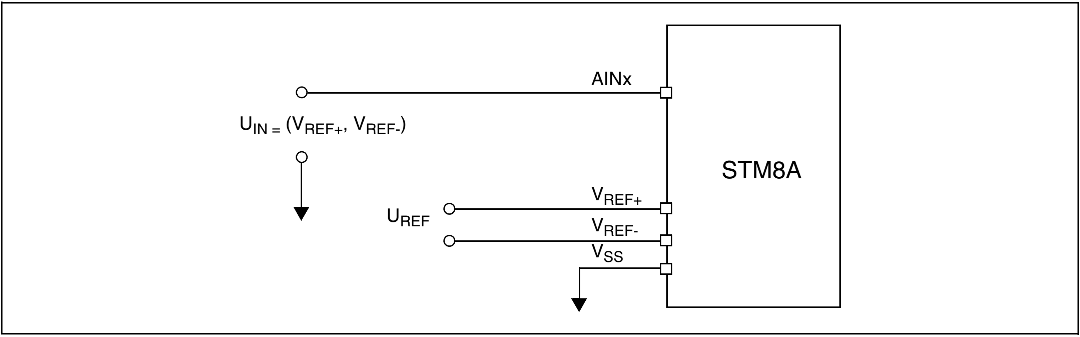
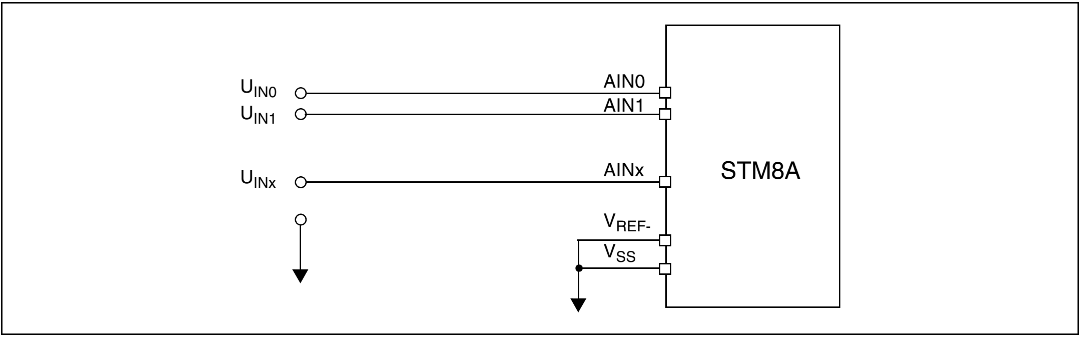


Figure 11. Multiple channel measurement



4.3 Methods for precision improvement

This section presents some hardware and software methods to minimize errors that can occur in analog-to-digital conversions.

4.3.1 Hardware methods

- Analog zooming (use appropriate V_{REF} voltage and V_{REF} offset)
  - selects reference voltage between input signal ranges
  - gives full ADC range – minimum voltage per bit
- White noise added to measured signal
  - wobbling of input signal over several bits makes it possible to use averaging (if input signal is very stable)
  - white noise gives independence from sampling frequency
- Hardware design considerations
  - grounding
  - reference voltage filtering
  - supply filtering
  - preamplifier usage
  - frequency independence

4.3.2 Software methods

- Averaging samples
  - averaging decreases speed but can improve accuracy
- Digital filtering (50/60 Hz suppression from DC value)
  - proper sampling frequency to be set (trigger from Timer1 is useful in this case)
  - software post processing to be performed on sampled data (comb filter)
- Fast Fourier Transform (FFT) for AC measurements
  - shows harmonic parts in measured signal
  - slower due to more computational power requirements
- Calibration of ADC—offset, gain, bit weight calibration
  - decreases internal ADC errors
  - internal ADC structure must be known

5 Design recommendations

The main design recommendations for using the analog-to-digital converter are listed below.

- Grounding of analog/digital power
  - implement star topology
- V_{DDA}, V_{SSA} filtering
  - implement RC, LC filtering
  - avoid noise from digital power supply
- V_{REF} selection—offset, value, precision
  - reference voltage source precision and stability to correspond to the required precision in the application and the ADC’s capability
  - reference voltage source value and precision to correspond to the expected measurement range
- Source impedance vs. input impedance knowledge
  - use input buffers for measured signal
  - impedance relation with required conversion speed
- External preamplifier usage
  - for low (and also high) level signals
  - amplifier speed and precision properties
  - amplifier dependency on frequency
- Appropriate ADC mode, speed and trigger to be selected

6 Revision history

Table 2. Document revision history

Date	Revision	Changes
16-Oct-2009	1	Initial release.