AN1798
Application note

How to use the STR71x A/D converter and apply a conversion speed-up technique

Introduction

This application note describes how to use the Sigma-Delta Analog to Digital Converter (ADC) in the STR71x microcontroller.

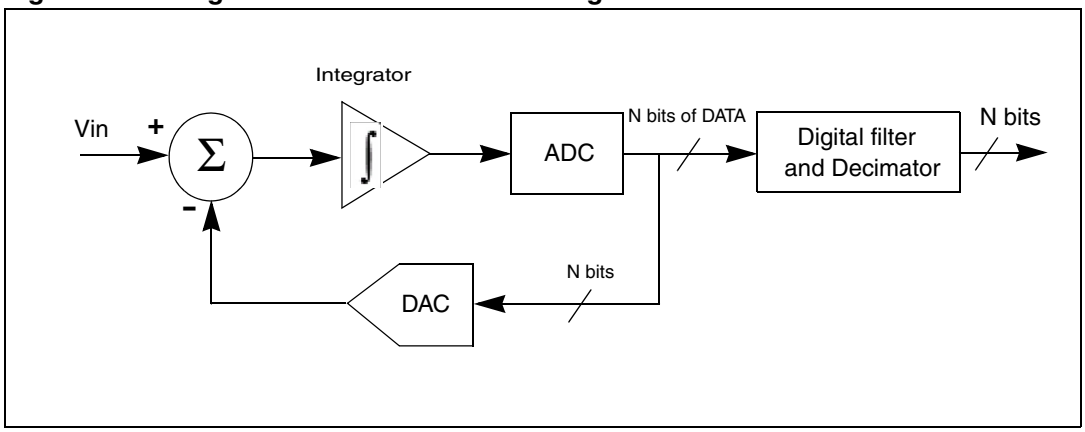
It describes the Sigma-Delta ADC calibration and linearization technique and shows how to apply a conversion speed-up technique to reduce the conversion time.

Contents

1 Principle of operation

Sigma-Delta converters, known also as oversampling converters, sample the input signal many times for each output sample with a frequency rate much greater than the Nyquist frequency (twice the input bandwidth). A Sigma-Delta converter consists of two blocks, the first is the Sigma-Delta modulator which produces the bit stream to the second part which consists of a digital filter and decimator.

Figure 1. A Sigma-Delta converter block diagram

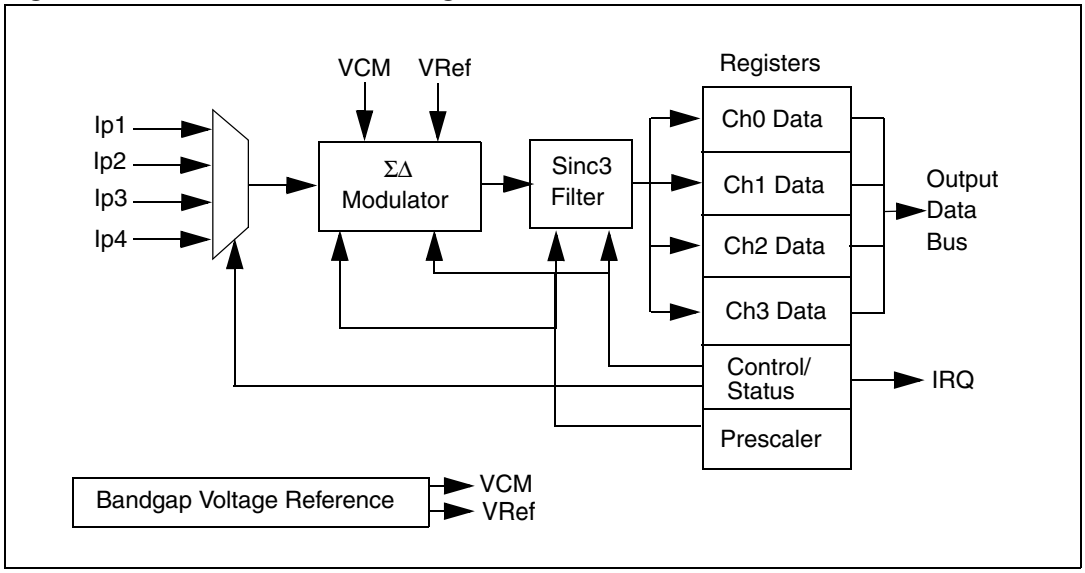


The STR71x ADC is a Sigma-Delta converter which composed of a second-order Sigma-Delta modulator followed by a sinc3 digital filter and decimator.

2 STR71x ADC features

The STR71x ADC consists of four channels (Ip1 to Ip4 or AIN0 to AIN3) to convert signals with an input range of 0 to 2.5 V into 12-bit format. The following figure shows the ADC module of STR71x.

Figure 2. ADC STR71x block diagram



2.1 Clock timing

Each channel supports up to 1 kHz for sampling frequency (f_s), consequently the signal frequency can not exceed 500 Hz (to respect the Nyquist frequency condition).

$$fs = f_{Mod} / 512 * 4$$

f_{Mod} (oversampling frequency) is the clock frequency which clocks the Sigma-Delta modulator. This frequency can not exceed 2.1 MHz and therefore the maximum sampling frequency is equal to 1 kHz (fs = 2.1 MHz / 512 * 4 = 1 kHz).

f_{Mod} is generated by PCLK2 and divided down by the prescaler factors configured in the ADC_CPR register. The prescaler output frequency must be not greater than 2.1 MHz.

$$f_{Mod} = f_{PCLK2} / \text{Prescaling factor}$$

Example: if f_{PCLK2}= 16 MHz and the desired sampling frequency f_s= 1 kHz.

$f_{Mod} = f_s * 512 * 4 = 1$ kHz * 512 * 4 ~ 2 MHz

The Prescaling factor = f_{PCLK2} / f_{Mod} = 16 MHz / 2 MHz = 8

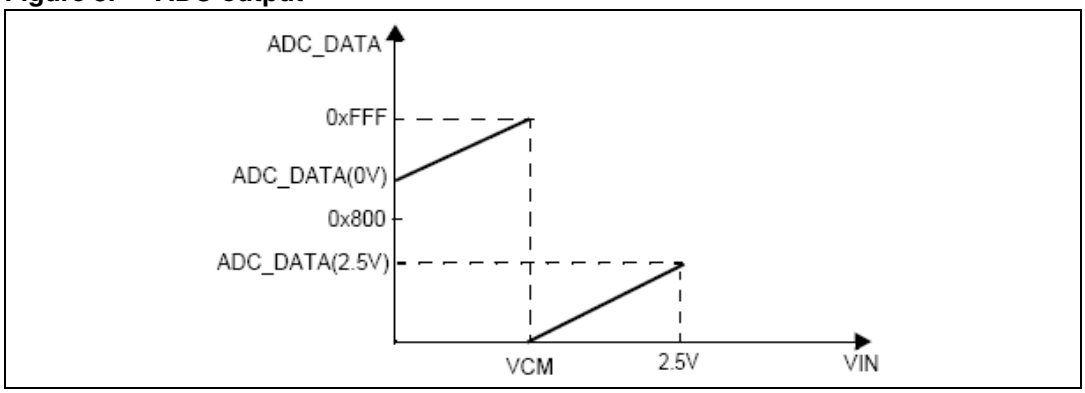
This value being twice the prescaler register value: so, the value to be configured in the ADC_CPR register is equal to 0x4. (see STR71x Reference Manual, ADC section: ADC prescaler setting table)

2.2 ADC output, Gain and offset features

The converted values stored in ADC_DATA[n] 16-bit register are signed two's complement values and only the 12 most significant bits are used.

The following figure gives the ADC output versus the input voltage.

Figure 3. ADC output



ADC_DATA(0V) and ADC_DATA(2.5V) are the conversion of 0V and 2.5V respectively. The gain of the ADC is given by:

$$G = [0xFFF - ADC\_DATA(0V) + ADC\_DATA(2.5V)] / 2.5$$

The quantum is calculated by the following equation:

$$q = 2.5V / [0xFFF - ADC\_DATA(0V) + ADC\_DATA(2.5V)]$$

The offset is the digital unsigned value of 0V (ADC_DATA(0V)) and the application should subtract this offset when reading all the conversion results.

2.3 Round-Robin and Single channel modes

These modes are provided to simplify the use of the ADC.

- Single channel mode selects one channel that is the only channel input to the Sigma-Delta modulator. To use single channel mode, bit 6 in the ADC_CSR register must be set to 1. A valid sample in this mode is produced every 2048 modulator clock cycles.
- Round-Robin mode (which is the normal mode) allows you to simplify your software code and avoid using an endless loop (while (1)) to get the converted values of all channels continuously. This process is repeated for each of the channels continually in a round-robin fashion. A valid sample in this mode is produced every 512 modulator clock cycles for each channel.

2.4 Conversion data availability and interrupt generation

The End of Conversion is indicated by four flags or by interrupt generation. The four flags are the Data Available flags (DA[n]) in the Control Status register. They allow the application software to determine which channel data register has a new sample ready to be read. Each DA[n] flag corresponds to ADC channel n. They are set by hardware as soon as a new sample on the corresponding channel is available and they are automatically cleared when the corresponding data register is read.

The STR71x ADC is able to generate an interrupt at the end of conversion. This interrupt depends on the conversion mode. In Single Channel mode an interrupt is generated if, in the CSR register, the interrupt bit and Data Available flag for the selected channel are set. In Round Robin mode, an interrupt is generated if all interrupt bits and all Data Available flags in the CSR are set.

3 Calibration and linearization technique

3.1 Principle

ADC calibration is done by software, it consists of a new calculation of the gain, the quantum and the offset each time the application is executed.

Initially, the you have to obtain the ADC conversion digital result for 0 V and 2.5 V by applying two stable voltage external sources (0 V and 2.5 V) to the ADC input in order to calculate the ADC gain and the quantum.

Next, you need to calculate the average of these two values by reading each value n times and dividing by n to reach the average values.

After that, calculate the unsigned values of ADC_DATA(2.5V) and ADC_DATA(0V) (these values are two's complement).

Then, calculate the ADC range or the sweep of conversion in count / volt:

sweep = Unsigned[ADC_DATA(2.5V)] - Unsigned[ADC_DATA(0V)].

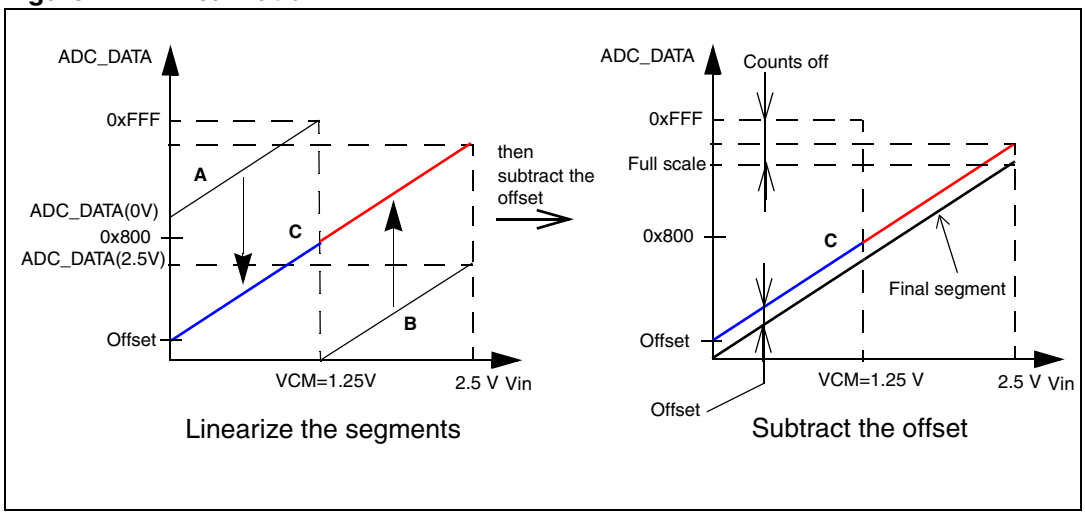
Finally, calculate the gain and the quantum:

G = sweep / 2.5 V

q = 2.5 V / sweep

The principle of linearization is to shift the half-lines 'A' and 'B' respectively down and up to obtain one segment: 'C'. Then, subtract the offset as shown in the following figure.

Figure 4. Linearization



These segments are linearized by software by adding 0x800 if the reading value is positive (the most significant bit is set to 0). If this value is negative (the most significant bit is set to 1) subtract its two's complement from 0x800. Then, subtract the offset from this value. The offset value is equal to Unsigned[ADC_DATA(0V)].

3.2 Hardware implementation

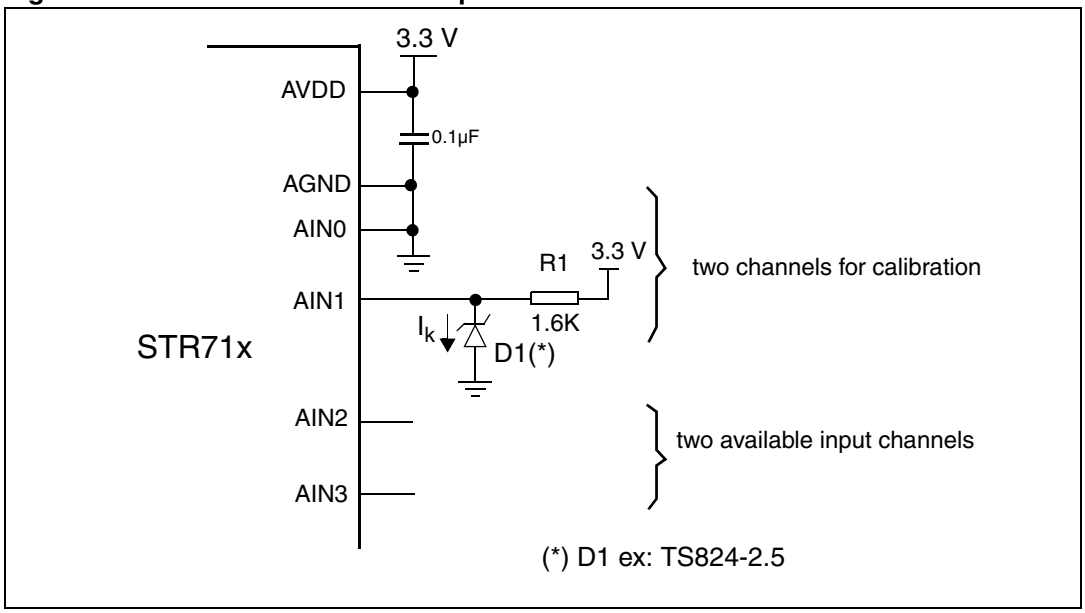
In order to calibrate the ADC, apply two stable voltage sources and read these values to obtain the dynamic of the ADC.

This can be done by applying the 2.5 V source to one of four ADC channels (for example AIN1) and connecting the ground to another channel (for example AIN0). Two channels are reserved for hardware calibration.

It is possible to obtain the 2.5 V source by a zener diode (ex: TS824-2.5) and a dedicated resistor.

The following figure shows the calibration hardware implementation:

Figure 5. Calibration hardware implementation



The resistor value is chosen to have a current I_k < 1mA (Reverse break down current).

This condition (see TS824-2.5 datasheet) allows to have a V_{Ref} more stable than if I_k >1 mA.

$$I_k = (V_{in} - V_{Ref}) / R1$$

By choosing I_k = 500µA < 1mA

Afterwards: $R1 = (V_{in} - V_{Ref}) / I_k$

$= (3.3 - 2.5) / 500 * 10^{-6}$

$= 1.6K$

3.3 Firmware description

The calibration, the linearization and offset compensation are done by software. As described in *Section 3.1*, the ADC conversation range, the gain and the quantum are computed. *Figure 6* shows an example flowchart of the software:

Figure 6. Calibration firmware implementation

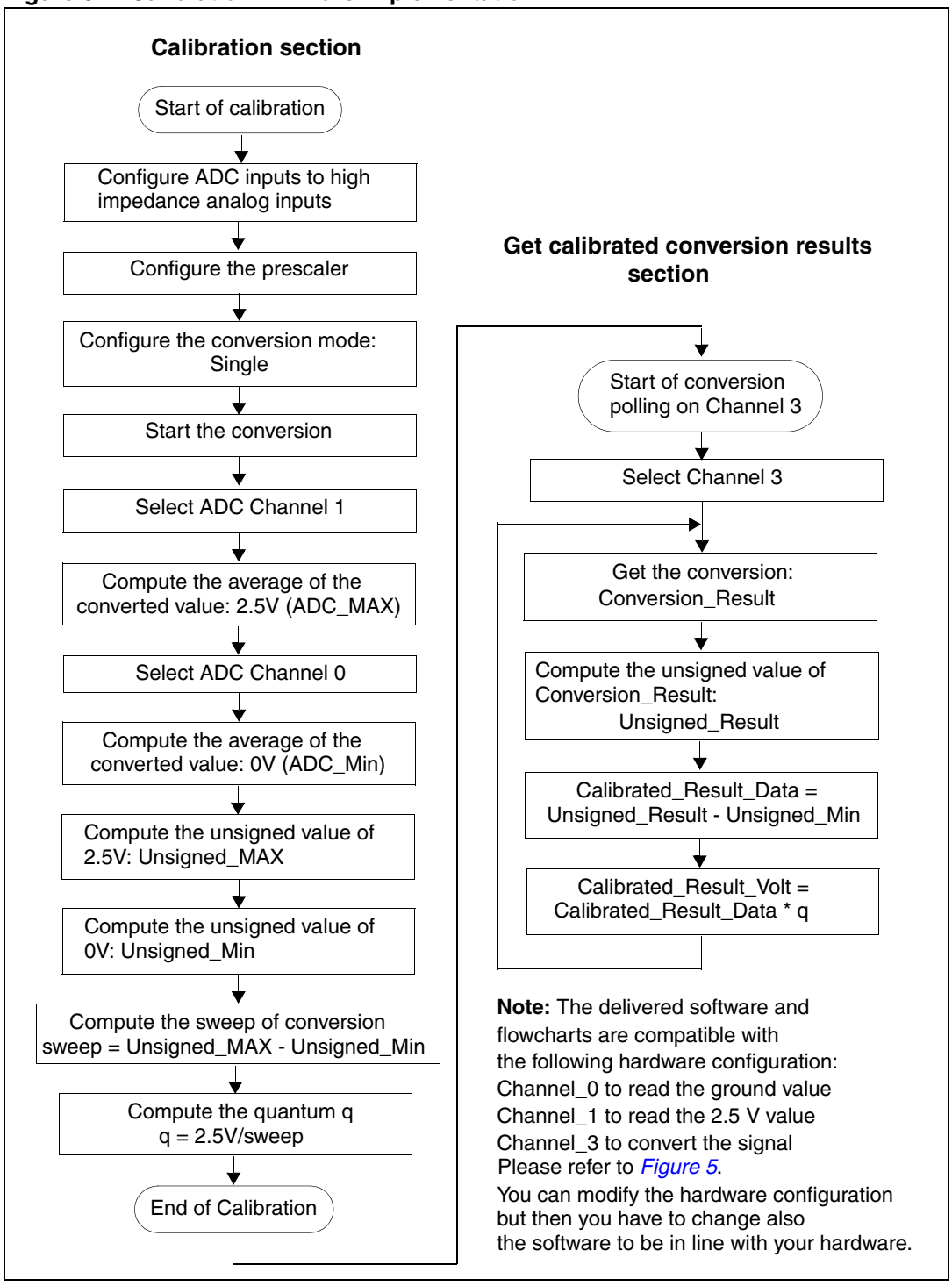


Figure 7. Compute the unsigned value of a converted value

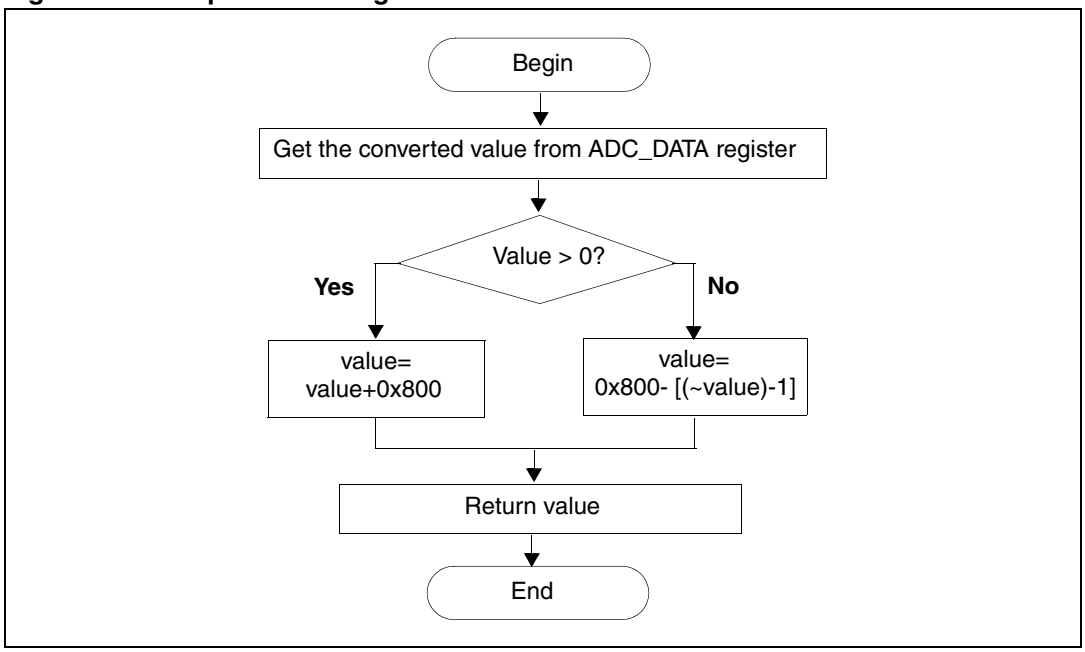
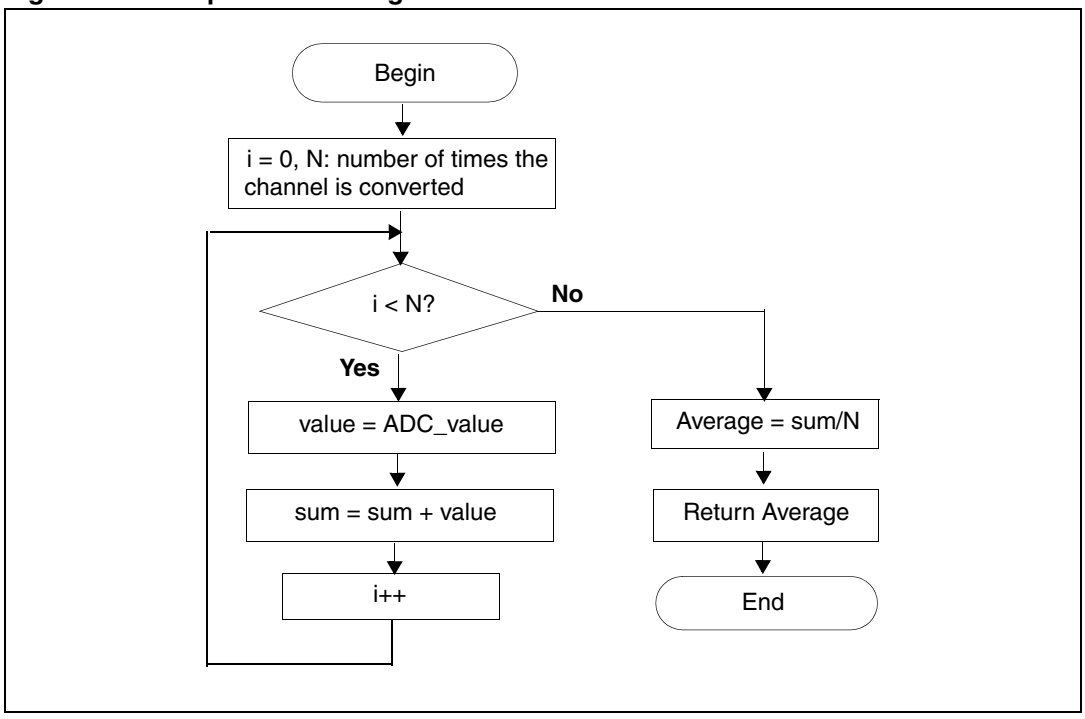


Figure 8. Compute the average of the converted value



Example of results with this software:

The software determine the averages of the two voltage levels (0 and 2.5 V) which are applied to Channel 0 and Channel 1 respectively and these results are shown in the debugger:

ADC_Min: 0x963 (0 V) = 2403 decimal (average value)

ADC_Max: 0x66E (2.5 V) = 1646 decimal (average value)

Then, the software determines their unsigned values:

Unsigned_Min (0 V) = 0x800 - ((~0x963 & 0xFFF) + 1)

= 0x165

= 357 decimal

Unsigned_Max (2.5 V) = 0x800 + 0x66E

= 0xE6E

= 3694 decimal

Then the software determines the sweep of conversion:

Sweep = Unsigned_Max - Unsigned_Min

= 0xE6E - 0x165

= 0xD09

= 3337 decimal

Finally it determines the gain G and the quantum q:

G = 3337 / 2.5 V = 1334.8 Volt $^{-1}$

q = 2.5V/3337 = 749 µV

For example a voltage level = 1.60 V is applied to Channel 0, we get:

Conversion_Result = 0x124

Unsigned_Result = 0x800 + 0x124 = 0x924

The calibrated conversion result after subtracting the offset is:

Calibrated_Result_Data = Unsigned_Result - Unsigned_Min

= 0x9C0 - 0x165

= 0x85B

= 2139 decimal

It is possible also to determine the applied voltage value in Channel 0 by multiplying this result by the quantum:

Calibrated_Result_Volt = 2139 * 749 µV = 1602111µV ~ 1.60 V

Ideally 1.60V converted value is: 1334.8 * 1.60 = 2136

The accuracy of conversion is: 2136 / 2139 = 99.8%

4 STR71x ADC conversion speed-up

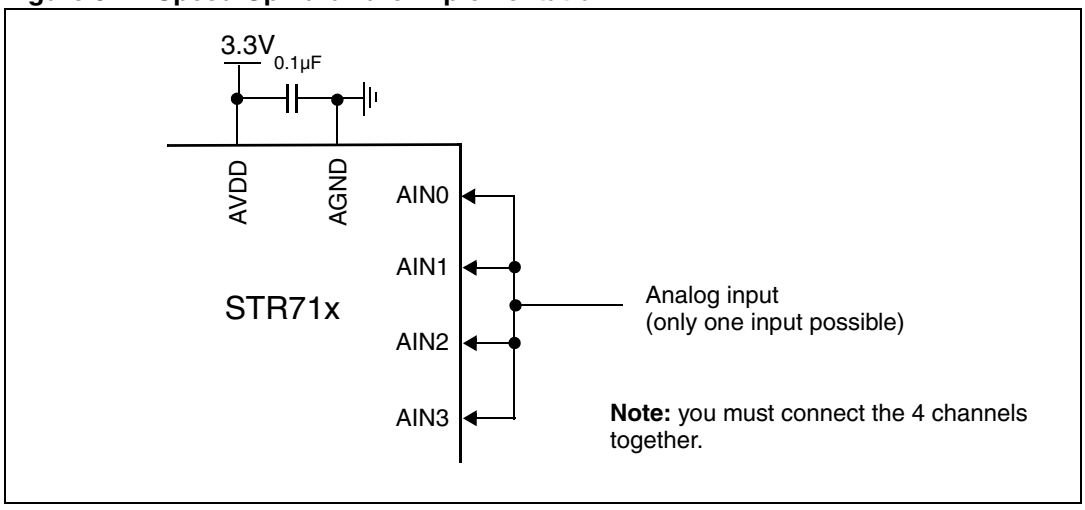
This section gives an example of how to reduce the conversion time of the A/D Converter of the STR71x microcontroller, when only one input channel needs to be converted. This is achieved using Round Robin mode instead of Single Channel mode to convert an analog input signal which must be connected to all four analog input pins of the converter. By configuring the conversion time to 1ms per channel, the conversion Speed-Up technique allow conversion time to be reduced down to 250 µs.

4.1 Hardware implementation

In Single Channel mode, a valid sample for the selected channel is produced only every 2048 clock cycles, with the same output frequency as Round Robin mode which samples the four channels (512 clock cycles for each channel). Consequently, the conversion time for one analog voltage can be reduced by using Round Robin mode instead of Single Channel mode.

The four ADC inputs have to be connected to the same analog voltage input in order to use Round Robin conversion mode. As shown in *Figure 9*, only one input can be used when this technique is applied.

Figure 9. Speed-Up hardware implementation



4.2 Firmware description

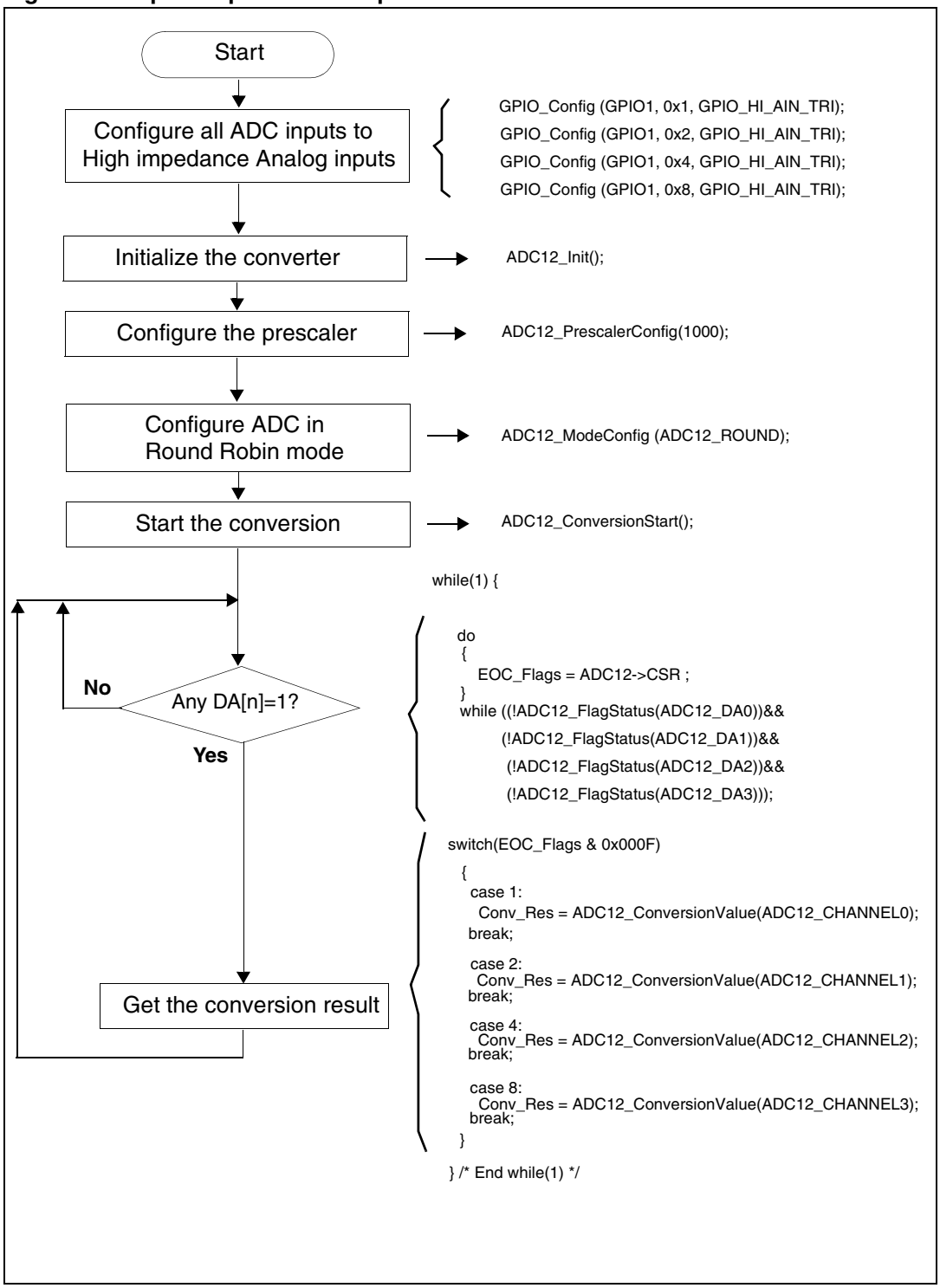
To use the procedure described above, the ADC has to be driven in polling mode because in Round Robin mode, an interrupt is generated only when the four input channels have been converted and all the Data Available flags in the Control Status Register (CSR) are set. In polling mode, it is possible to get the conversion result of the channel corresponding to the DAn flag that has been set without waiting for the end of conversion on all the channels.

First, all the STR71x analog pins must be configured as high impedance analog inputs. Next, the STR71x ADC has to be selected in Round Robin conversion mode. Then, the prescaler has to be configured. Next, the ADC has to be enabled by setting the ADC_En bit in the PCU_BOOTCR register. To speed-up the ADC conversion you have to check all the

Data Available flags DA[n] and each time a flag is set you have to read the corresponding data register ADC_DATA[n] to get the last converted value. Thus, after every 512 cycles of the oversampling clock, a new conversion value becomes available in the ADC_DATA[n] register, the next new conversion value will be available after 512 cycles in the ADC_DATA[n+1] register. The converted value to be read after the ADC_DATA4 one will be ADC_DATA0. The following flowchart shows how to implement this procedure.

Note: To obtain the desired results with this software, you must follow the hardware configuration shown in Figure 9: connect the four channels together.

Figure 10. Speed-Up software implementation



5 Revision history

Table 1. Document revision history

Date	Revision	Changes
21-Jan-2004	1	Initial release
11-Feb-2007	2	Revised and updated
21-Nov-2007	3	Added *Section 1: Principle of operation* *Section 2: STR71x ADC features* *Section 3: Calibration and linearization technique* Updated *Section 4: STR71x ADC conversion speed-up*.