



## IMPLEMENTATION OF SIGMA-DELTA ADC WITH ST7FLITE05/09

---

### INTRODUCTION

The purpose of this document is to describe how to implement a 10-bit Sigma-Delta A/D converter using a simple external circuit and a Sigma-Delta conversion program.

The ST7FLITE05(09) has an on-chip ADC with 8-bit resolution and an input range of  $0-V_{CC}$ .

The external Sigma-Delta ADC described in this application is designed for relatively slow alternating signals (0,2 - 5Hz) in the range 0 - 10mV(p-p). The focus of the project is to provide good relative accuracy, repeatable parameters and simplicity, resulting therefore a low cost of the device (in its simplest form, apart from the microcontroller, only three RC elements are necessary).

## 1 SIGMA-DELTA CONVERTER THEORY

The basic Sigma-Delta converter (Figure 1) is built of 2 basic circuits: a modulator and a digital filter. In the modulator the input signal is summarised with the signal of negative feedback from the D/A converter. The signal's difference, after passing through the integrating circuit, reaches the input of the comparator, where it is compared to the reference voltage (the comparator works as a 1-bit quantizator). The input signal from the comparator controls the 1-bit converter and reaches the input of the digital filter, which decreases flowability and transforms the 1-bit stream into 10-bit words.

**Figure 1. Sigma-Delta Conversion**

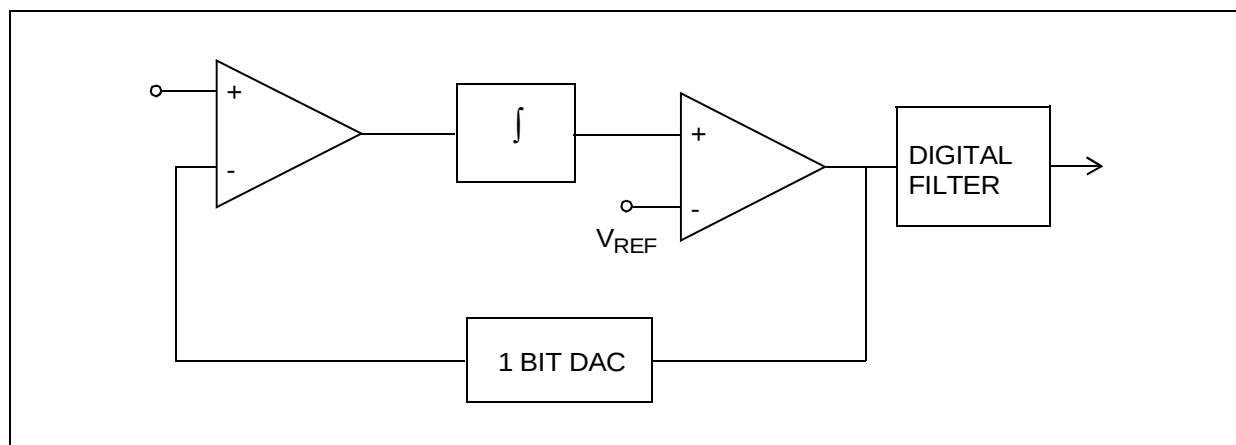
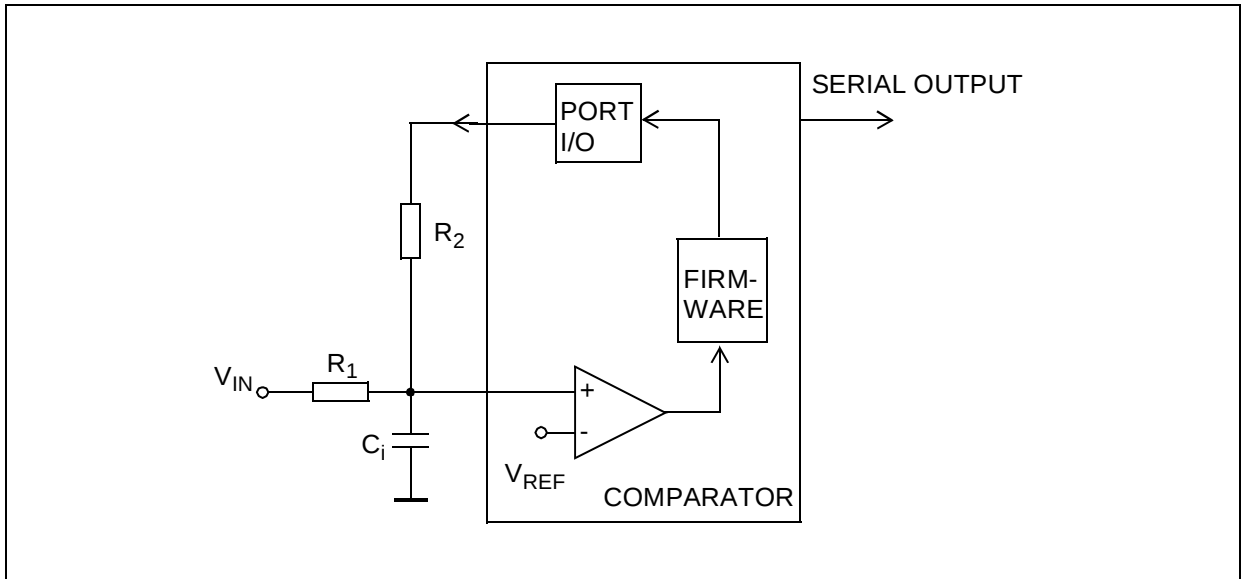


Figure 2 shows the general scheme of the microcontroller implementation. Within the system the function of an integrator is fulfilled by the  $C_i$  capacitor, and the range of converted voltages depends on the resistance ratio  $R_1$  and  $R_2$ . The principle of operation is as follows: from the moment of setting the high status on the output, the voltage on the  $C_i$  capacitor and simultaneously on the input of the comparator begins to rise. As soon as the reference voltage is reached the output status is altered and the voltage decreases. After falling below the reference voltage the output status is altered to the high one and the cycle repeats. Provision of the state of balance and the correct conversion of signal into its digital form is realised by a program-operated feedback loop and digital filter.

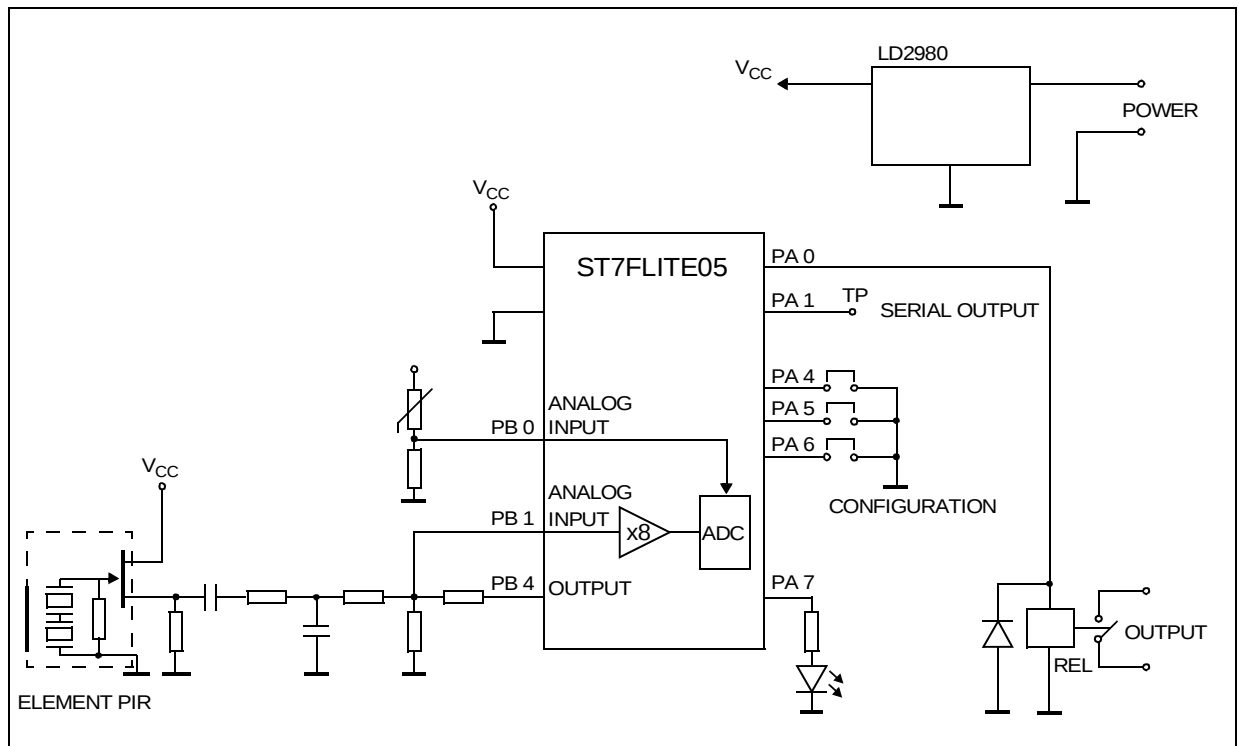
**Figure 2. Sigma-Delta Implementation with Microcontroller**

## 2 REALISATION WITH THE USE OF THE MICROCONTROLLER ST7FLITE05(09)

The practical realisation of the Sigma-Delta converter has been used with our passive infrared detector. A sensor of passive infrared characterises with a low output voltage (below 10mV) as well as with a usable range of frequencies from 0,2Hz to 7Hz. Therefore the Sigma-Delta converter is a perfect solution. It provides an accurate enough conversion of a signal into its digital form with use of a minimal number of external elements. That means simplicity of construction, relatively low production costs and good using properties.

Figure 3 presents the block diagram of the detector.

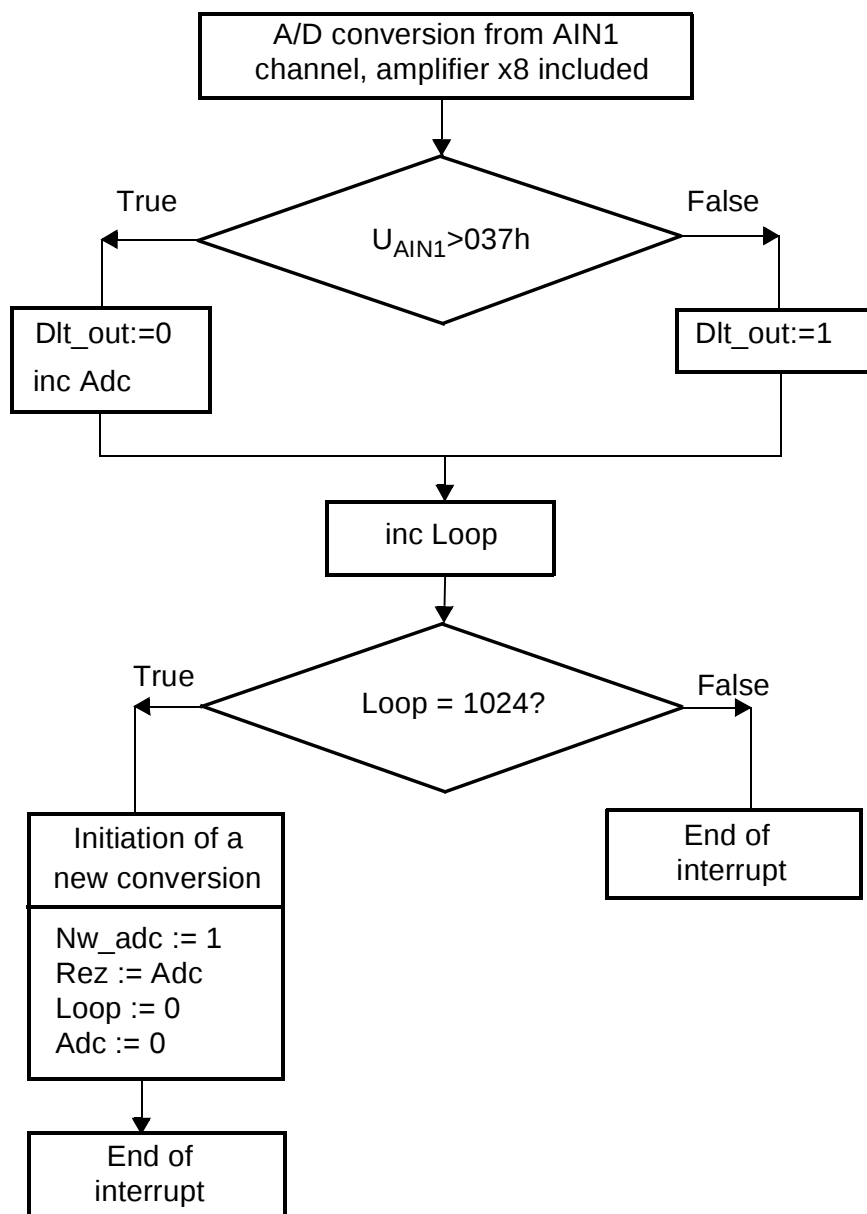
**Figure 3. ST7FLITE05 interfacing with PIR Sensor**



A signal from the sensor passes through the capacitor to the input of the converter. The A/D converter has been used as a comparator in the microcontroller as well as the source of the reference voltage. By enabling the internal x8 amplifier, the amplitude of an output signal has been decreased 20 times to provide correct operation of the amplifier. This allows a decrease in the resistors' value span in the same ratio and to increase the operational accuracy of the A/D converter working as a comparator. Additionally the temperature compensation of sensitivity has been introduced.

Figure 4 shows the work algorithm of the Sigma-Delta converter.

Figure 4. Flow Chart (Sigma-Delta Conversion)



### 3 THE DESCRIPTION OF THE SIGMA-DELTA CONVERSION PROGRAM

The Sigma-Delta conversion procedure is executed cyclically in an interrupt from the AT timer with a period of 50 $\mu$ s. The result of the Sigma-Delta conversion (a 10-bit word) is achieved every 1024 interrupts, that is to say every 51.2ms. The main loop of the program awaits the Sigma-Delta conversion result and sends it through the RS serial interface to the computer.

The start of the Sigma-Delta conversion is preceded with clearing of the loop conversion counter Loop\_h, Loop\_l and the result registry Adc\_h, Adc\_l.

At the first stage, the service routine of the AT timer interrupt clears the marker of request and executes the A/D conversion from the AIN1 channel (the x8 amplifier is on). Following this, the result of conversion is compared with the constant value represented by the reference voltage 037h).

If, in the result of comparison, the voltage from the AIN1 channel is lower than the reference voltage, then the B-Dlt port line will be set and the loop conversions counter will be increased. Loop\_h, Loop\_l.

In a reverse situation, when the voltage from the AIN1 channel is higher than the reference voltage, the B-Dlt port line will be set to zero, and the Adc\_h, Adc\_l result registry as well as the Loop\_h, Loop\_l. loop conversion counter will be increased.

The next task of the interrupt is to check whether the Loop\_h, Loop\_l loop conversion counter has reached the value 1024. When the Loop\_h, Loop\_l loop conversion counter is lower than 1024, the service routine of the interrupt finishes. If this condition is fulfilled the value of the result registry is rewritten into the result Rez\_h, Rez\_l buffer and a request of a new result to the main loop is set and the start of a new conversion is initiated (clearing of the Loop\_h, Loop\_l counter and of the Adc\_h, Adc\_l. result registry)

The period of an AT timer interrupt is not critical. In the below example, because of the frequency of the signal from the passive infrared detector and the speed of RS communication (19200bps), it has been defined at 50 $\mu$ s.

```
;*****  
;  
;Listing of S-D conversion program.  
;*****  
;  
st7/  
  
        TITLE    "pir.asm"  
        #INCLUDE "st72flt0.inc"      ; definitions of registers st72lite  
;*****  
;  
;The program that realises the sigma/delta conversion  
;sends the results of conversion to PC through the RS port  
;*****  
;  
;Definitions of variables
```

```

;*****
Rez_h          equ $80          ; conversion result buffer H
Rez_l          equ $81          ; conversion result buffer L
Loop_h         equ $82          ; loop conversions counter H
Loop_l         equ $83          ; loop conversions counter L
Adc_h          equ $84          ; conversion result H
Adc_l          equ $85          ; conversion result L

FLAGS0         equ $90          ; binary variables
Nw_adc         equ 7            ; new data from the conversion
m50us          equ 6            ; marker 50us
;*****
;Definitions of constants
;*****
;port A                ; destination of bits in port A
RS                  equ 6
;*****
;port B                ; destination of bits in port B
Adc_in           equ 1
Dlt_out          equ 4
;*****
;Processor initiation segment 'rom'
;*****
INIT:            sim

                rsp
                ld a,$FFDE
                ld RCCR,a        ; oscillator frequency calibrator RC
                ld a, #%01000000 ; settings of ports
                ld PADDR,a
                ld a, #%01000000
                ld PAOR,a
                ld PADR,a
                ld a, #%00010000
                ld PBDDR,a
                ld a, #%00010000
                ld PBOR,a
                ld PBDR,a

                ld x,$80          ; All RAM clearing (80-FF)
loop_clrRAM:    clr (x)
                inc x
                jrne loop_clrRAM

                ld a,$0E          ; initiation of an AT timer interrupt with
                                ; the period of 50us

```

```
ld ATRH, a
ld a, #$70
ld ATRL, a
ld a, #%00010010
ld ATCSR, a
rim

;*****
;Main loop of the program
;*****
IDLE:      btjf FLAGS0, #Nw_adc, Idle ; waiting for the conversion result
           bres FLAGS0, #Nw_adc      ; new result

                                           ; sending the conversion result
                                           ; through RS
           ld a, #$FF                ; frame format:
           call Send_rs              ; 0FFh, 0FFh, Rez_H, Rez_L
           ld a, #$FF
           call Send_rs
           ld a, Rez_h
           call Send_rs
           ld a, Rez_l
           call Send_rs
           jra IDLE

;*****
;Interrupts
;*****
; delta/sigma conversion
; sampling of 1 bit with period of 50us
; conversion result- 10 bit word, received every (1024*50us)=51,2ms
;*****
timoverfl_rt: bres ATCSR, #2          ; cancelling of the interrupt re-
                                           ; quest
           ld a, #00000100           ; adding the amplifier into the A/D
                                           ; processing
           ld ADCCAMP, a
           ld a, #%01100001          ; voltage measurement at AIN1
           ld ADCCSR, a

loop_adc:   btjf ADCCSR, #7, loop_adc
           ld a, ADCDR
           cp a, #$37                ; comparison U_ain1 z 037h
           jrc set_Dlt
           bres PBDR, #Dlt_out        ; U_ain1>037h
           inc Adc_l                  ; increase of the conversion result
```

```

        jrne inc_loop
        inc Adc_h
        jra inc_loop
set_Dlt: bset PBDR, #Dlt_out      ; U_ain1<037h
inc_loop: bset FLAGS0, #m50us    ; setting of a marker 50us
        inc Loop_l              ; increase of the samples counter
        jrne tst_loop
        inc Loop_h

tst_loop: ld a, Loop_l           ; is the samples counter equal to
                                           ; 1024?
        jreq tst_loopH
        ired
tst_loopH: ld a, Loop_h
        cp a, #$04
        jreq ok_knw            ; yes
        ired
ok_knw:  clr Loop_l             ; conversion completed
        clr Loop_h             ; initiating of the loop conversion
                                           ; counter
        ld a, Adc_h            ; rewriting of the result into
                                           ; Rez_h, Rez_l
        ld Rez_h, a
        ld a, Adc_l
        ld Rez_l, a
        clr Adc_l              ; initiating of the conversion re-
                                           ; sult variable
        clr Adc_h
        bset FLAGS0, #Nw_adc    ; marker for IDLE (data ready)
        ired
; *****
spi_rt    ired
ltime_rtc_rt  ired
ltime_outcmp_rt  ired
time_outcmp_rt  ired
lvd_rt    ired
ext3_rt    ired
ext2_rt    ired
ext1_rt    ired
ext0_rt    ired
sw_rt      ired
dummy      ired
; *****
; Procedures and subprograms
; *****

```

```
Send_rs:      bres FLAGS0, #m50us      ; procedure of RS data byte sending
              call wait                ; 19200bps, 1b start, 1b stop, 8b data
              bres PADR, #RS           ; setting of a start bit
              call wait
              ld x, #$08

loop_snd:     rrc a
              call Send_bt             ; setting of 8 data bits
              dec x
              jrne loop_snd
              bset PADR, #RS           ; setting of a stop bit
              jra wait

Send_bt:      jrc sd_H                 ; setting of a data bit at the port
              bres PADR, #RS
              jra wait

sd_H:         bset PADR, #RS

wait:         btjf FLAGS0, #m50us, wait ; latency for the duration of the
                                           ; data bit
              bres FLAGS0, #m50us
              ret

;*****
;Interrupt vectors segment 'vectit'
;*****
              DC.W  dummy              ; FFE0-FFF1h location
spi_it        DC.W  spi_rt             ; FFE2-FFE3h location
lttimrtc_it    DC.W  lttimrtc_rt       ; FFE4-FFE5h location
ltimoutcmp_it  DC.W  ltimoutcmp_rt     ; FFE6-FFE7h location
timoverfl_it   DC.W  timoverfl_rt     ; FFE8-FFE9h location
timoutcmp_it   DC.W  timoutcmp_rt     ; FFEA-FFEBh location
lvd_it         DC.W  lvd_rt            ; FFEC-FFEDh location
              DC.W  dummy              ; FFEE-FFEFh location
              DC.W  dummy              ; FFF0-FFF1h location
ext3_it        DC.W  ext3_rt           ; FFF2-FFF3h location
ext2_it        DC.W  ext2_rt           ; FFF4-FFF5h location
ext1_it        DC.W  ext1_rt           ; FFF6-FFF7h location
ext0_it        DC.W  ext0_rt           ; FFF8-FFF9h location
              DC.W  dummy              ; FFFA-FFFBh location
softit         DC.W  sw_rt             ; FFFC-FFFDh location
reset          DC.W  INIT              ; FFFE-FFFFh location
              end
```

"THE PRESENT NOTE WHICH IS FOR GUIDANCE ONLY AIMS AT PROVIDING CUSTOMERS WITH INFORMATION REGARDING THEIR PRODUCTS IN ORDER FOR THEM TO SAVE TIME. AS A RESULT, STMICROELECTRONICS SHALL NOT BE HELD LIABLE FOR ANY DIRECT, INDIRECT OR CONSEQUENTIAL DAMAGES WITH RESPECT TO ANY CLAIMS ARISING FROM THE CONTENT OF SUCH A NOTE AND/OR THE USE MADE BY CUSTOMERS OF THE INFORMATION CONTAINED HEREIN IN CONNECTION WITH THEIR PRODUCTS."

Information furnished is believed to be accurate and reliable. However, STMicroelectronics assumes no responsibility for the consequences of use of such information nor for any infringement of patents or other rights of third parties which may result from its use. No license is granted by implication or otherwise under any patent or patent rights of STMicroelectronics. Specifications mentioned in this publication are subject to change without notice. This publication supersedes and replaces all information previously supplied. STMicroelectronics products are not authorized for use as critical components in life support devices or systems without express written approval of STMicroelectronics.

The ST logo is a registered trademark of STMicroelectronics.

All other names are the property of their respective owners

© 2004 STMicroelectronics - All rights reserved

STMicroelectronics GROUP OF COMPANIES

Australia – Belgium - Brazil - Canada - China – Czech Republic - Finland - France - Germany - Hong Kong - India - Israel - Italy - Japan - Malaysia - Malta - Morocco - Singapore - Spain - Sweden - Switzerland - United Kingdom - United States

[www.st.com](http://www.st.com)