APPLICATION NOTE

Driving an Analog Keyboard with the ST7 ADC

by 8-Bit Micro Application

INTRODUCTION

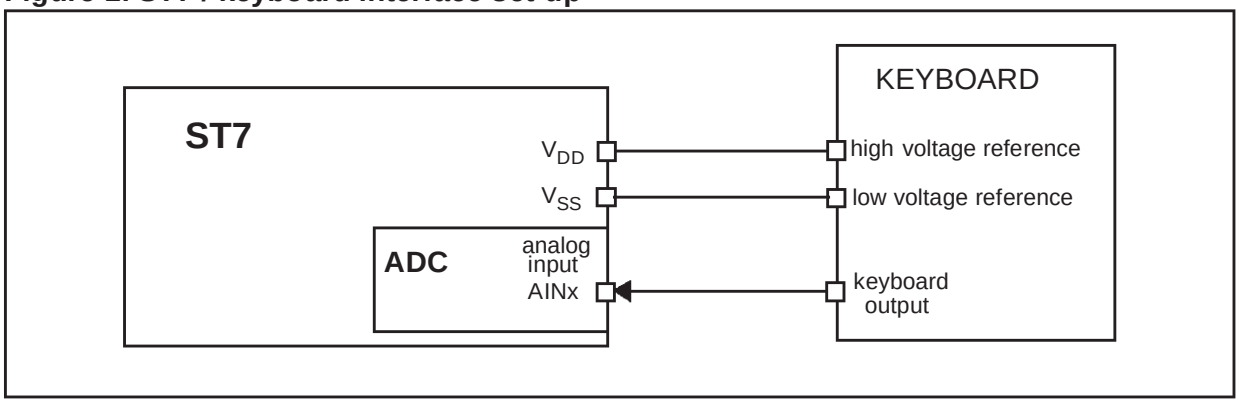
The goal of this application note is to present a standard example of the use of the Analog to Digital Converter (ADC) of the ST7.

In this note, the use of the ADC cell to emulate a 16 keys analog keyboard is presented. The technique for the keyboard is to connect the keys by resistive dividers to one of the converter inputs.

1 ST7 / KEYBOARD INTERFACE

The only point is to connect the analog keyboard with one of the analog inputs of the ST7.

Figure 1. ST7 / keyboard interface set-up



2 ST72311 CONFIGURATION

The application has been validated with a ST72311. Its configuration is described in this part. Refer to your datasheet for more details.

2.1 I/O CONTROL

The flexibility of the I/O port structure of the ST7 allows the multiplexing of up to 8 analog inputs into the ADC. They are the alternate functions of the pins of Port A.

The pins used by the ADC must be declared as inputs to avoid conflicts in alternate function mode.

Please, refer to the Data Book to configure pins properly.

2.2 ANALOG TO DIGITAL CONVERTER

The ST7 ADC is a 8-bits successive approximation converter, with internal sample and hold circuitry. It has up to 8 multiplexed analog input channels.

The ST7 ADC is specified for +/- 2 LSB.

2.2.1 ADC control

This cell is controlled with the ADC Control Status Register (ADCCSR).

7							0
COCO	0	ADON	0	0	CH2	CH1	CH0

The COCO bit is the conversion complete bit:

- When this bit is set, the conversion is done and the result can be read in the ADC Data Register.
- When the bit is reset, the conversion is not complete.

The ADON bit:

- The ADC is switched on when this bit is set.
- The ADC is switched off when it is reset.

CH2-CH0 bits:

- They are used to select which analog input to convert. In the ST7285 there are 8 analog pins.

2.2.2 Characteristics

The conversion time is 64 CPU cycles including a sampling time of 31.5 CPU cycles.

The ADC is linear and the digital result of the conversion is given by the formula:

$$\textbf{Digital Result} = \frac{\textbf{255*Input Voltage}}{\textbf{Reference Voltage}}$$

2.2.3 Process

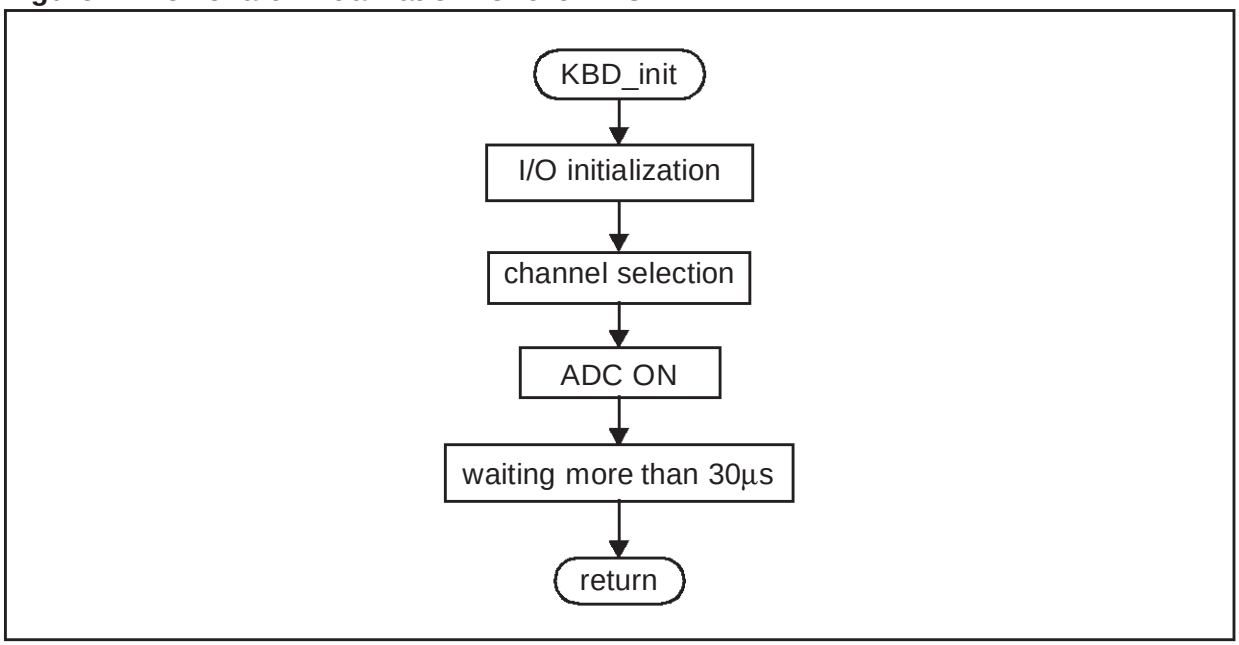
First the analog input pins must be configured as inputs (see Section 2.1).

Then the analog channel to convert must be selected using CH2-CH0 bits of ADCCSR register.

Setting the ADON bit will switch the converter on.

Note that a stabilization time (typically 30µs) is needed before the converter is enabled.

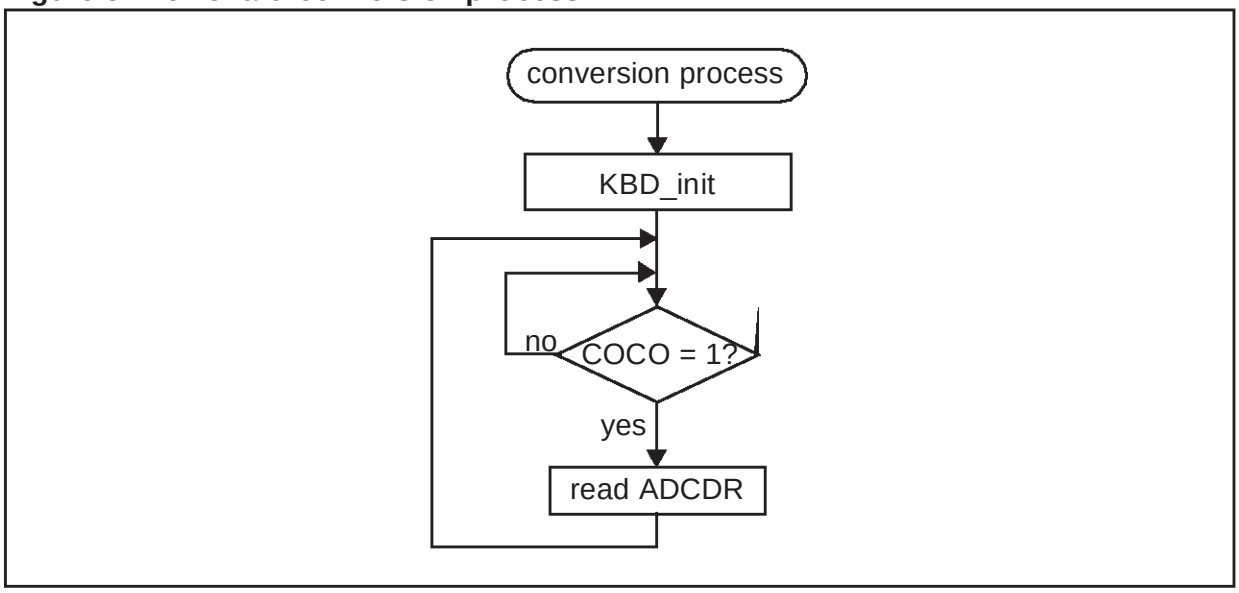
Figure 2. Flowchart: initialization for the ADC



Once a conversion is done, the COCO bit is set by hardware. It will be reset by a read of the ADCDR register or by a write into the ADCCSR register.

Once enabled, conversions will run continuously until the peripheral is disabled.

Figure 3. Flowchart: conversion process

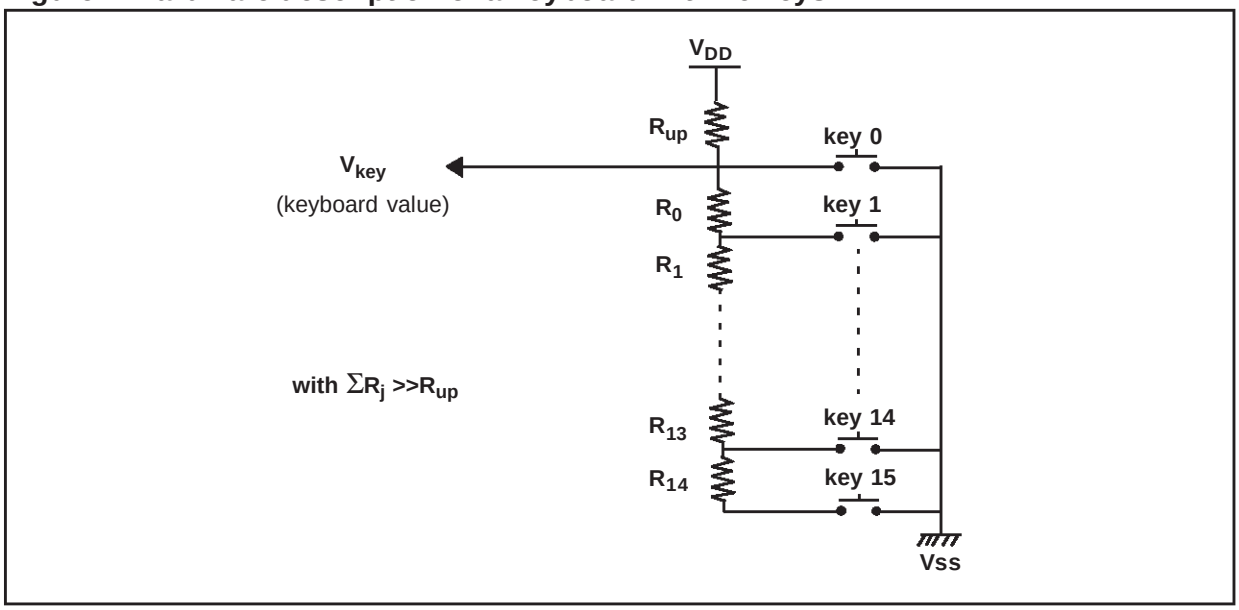


3 ANALOG KEYBOARD

3.1 PRINCIPLE

The purpose is to recognize a key when pressed. In an analog keyboard each key is associated with a voltage. The description of an analog keyboard is given by Figure 4.

Figure 4. hardware description of a keyboard with 16 keys



R_{up} is a pull-up resistor. So, when no key is pressed, V_{key} is equal to V_{DD}.

When 'key i' is pressed, the resistor R_{i-1} is connected to Vss. Then we have a resistive divider and V_{key} is given by the formula:

$$V_{key\ i} = \frac{(V_{DD} - Vss)\sum_{j=0}^{i-1} R_j}{R_{up} + \sum_{j=0}^{i-1} R_j}$$

So the corresponding voltage of each key is given by the values of the resistors. An equal distribution of voltage between V_{DD} and V_{SS} is usually recommended.

To recognize a key, the user will measure V_{key} and will be able to decide which key was pressed.

3.2 PRACTICAL LIMITATIONS

Theoretically, with an 8 bits ADC, 255 keys could be decoded. But potential errors must be taken into account. They can come from the power supply, the key resistivity, the resistor tolerance, the ADC linearity. The first two can normally be neglected.

The resistor tolerance is a main limitation as usually 5% tolerance resistors are used. It is advised to use a 1% tolerance resistor for the pull-up. Changing this resistor highly improves the keyboard as the pull-up has an influence on every key.

The ST7 ADC linearity is specified for +/- 2 LSB. So a margin of 4 LSB must be added to the resistor tolerance to avoid key decision errors.

These parameters will reduce the number of keys that can be decoded.

4 KEY DECISION

The ST7 is a digital microcontroller. It uses its ADC to measure V_{key}. It is then coded in 8 bits.

As the ADC is linear, the best decision is taken when the voltage levels of the keys follow an equal distribution between V_{DD} and V_{SS}.

In our application, a 16 key keyboard is used. So, the best associated V_{key} value of 'key i' is given by:

$$V_{key\ i} = \frac{(V_{DD} - Vss) \times i}{16}$$

The problem is that you cannot choose the perfect values for the resistors. In our application, the following resistor values were used (see Table 1):

Table 1. Example of resistor values for 16 keys

Resistor	Value (Ω)	Resistor	Value (Ω)
Rup	1K	R7	220
R0	68	R8	270
R1	75	R9	390
R2	82	R10	560
R3	100	R11	820
R4	120	R12	1K2
R5	150	R13	2K7
R6	180	R14	75

The digital values of the keys after conversion are given in Table 2.

When a key is pressed and after conversion, a decision must be taken on its value. Upper and lower limits of decision for each key must be defined. These values are the middle of two following typical values, which gives the best noise margin between keys.

The software uses the lower limits to make its decision. They are given in Table 2.

Table 2. Key values

KEY	typical digital value	lower digital limit of KEY	KEY	typical digital value	lower digital limit of KEY
no key	0xFF	0xF8			
‘F‘	0xEF	0xE7	‘7‘	0x70	0x68
‘E‘	0xDF	0xD6	‘6‘	0x60	0x57
‘D‘	0xCE	0xC7	‘5‘	0x4F	0x47
‘C‘	0xC1	0xB8	‘4‘	0x3F	0x37
‘B‘	0xB0	0xA8	‘3‘	0x2F	0x28
‘A‘	0xA0	0x97	‘2‘	0x20	0x18
‘9‘	0x8F	0x87	‘1‘	0x10	0x08
‘8‘	0x7F	0x78	‘0‘	0x00	0x00

5 DRIVING AN ANALOG KEYBOARD WITH THE ST7 ADC

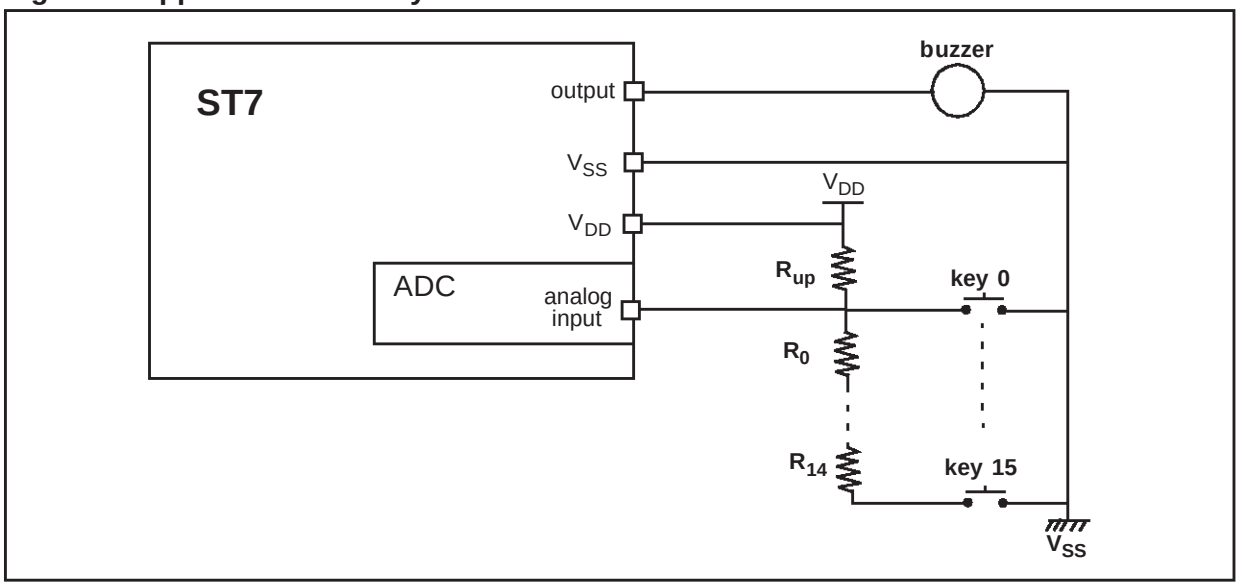
The software at the end of this application is the ADC driver for keyboards. The complete software can be found in the software library. It is only an example. It is the user's duty to adapt it to its application.

5.1 GENERAL DESCRIPTION

A 16 key keyboard is connected to a ST72311 through the analog pin 3 of Port D.

A buzzer is connected to an output pin (PC1).

Figure 5. Application circuitry



The software uses a polling strategy. It makes conversions continuously (no wake-up process).

First the software initializes the I/Os and the ADC.

When a conversion is done, the software decides if a key was pressed or not.

If not, it waits before analysing the result of a new conversion.

If a key is pressed, it compares it with the former key to know if the key is stabilized. Then, a buzzer is switched on to indicate to the user that a key is detected.

5.2 FLOWCHARTS

The main flowcharts of the application are given below Figure 6., Figure 7. & Figure 8. The initialization routine is presented Figure 2.

Figure 6. Flowchart: main program

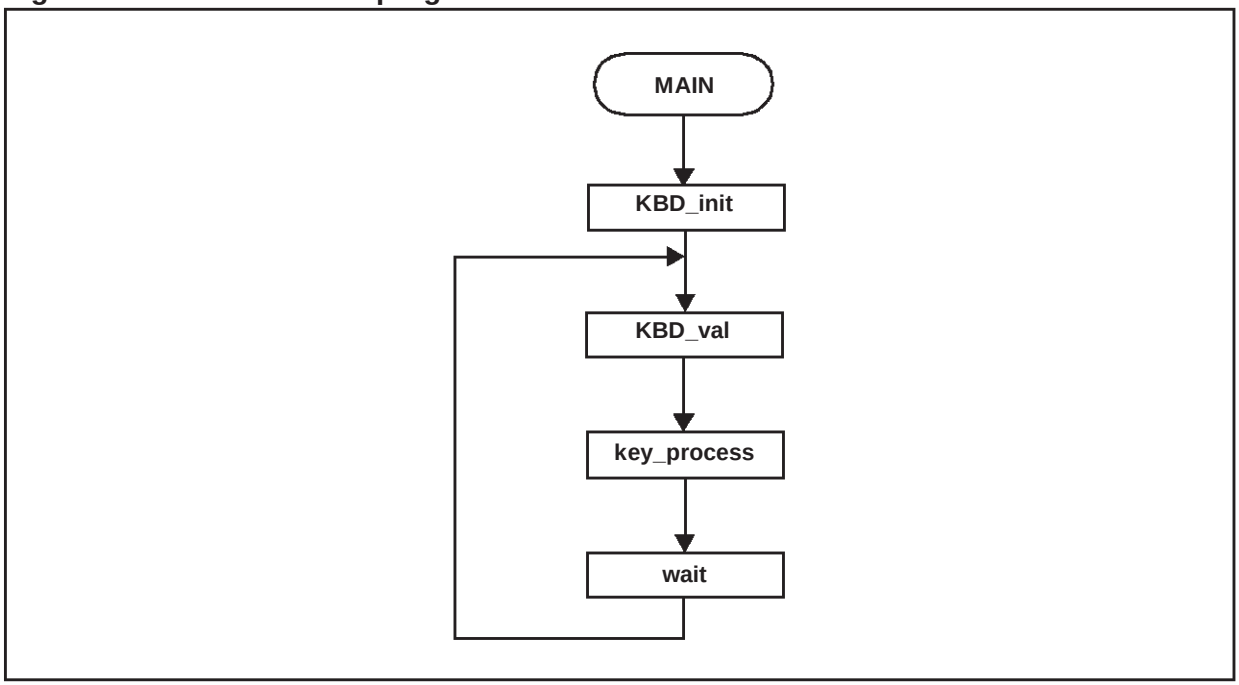


Figure 7. Flowchart: key recognition

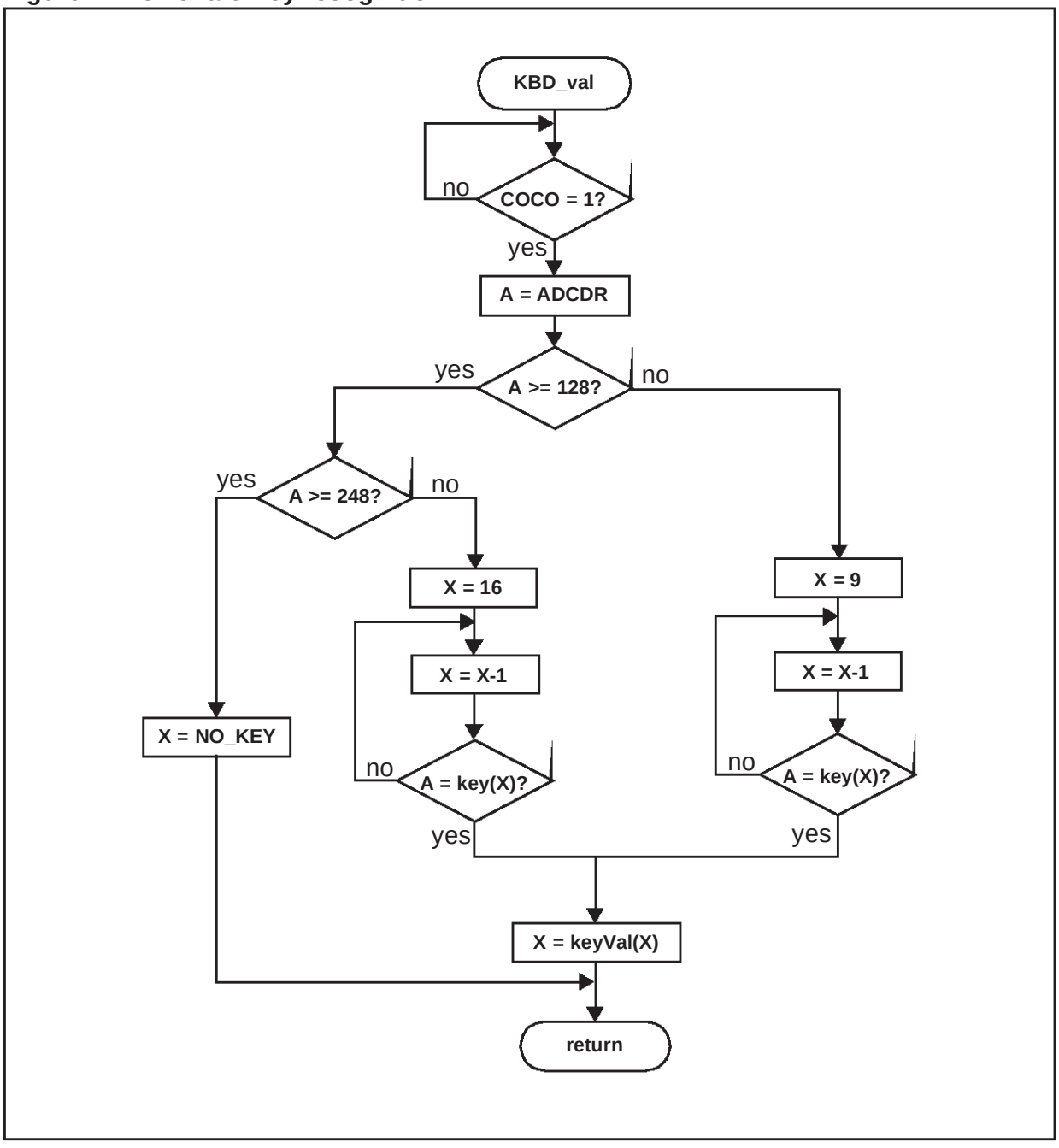
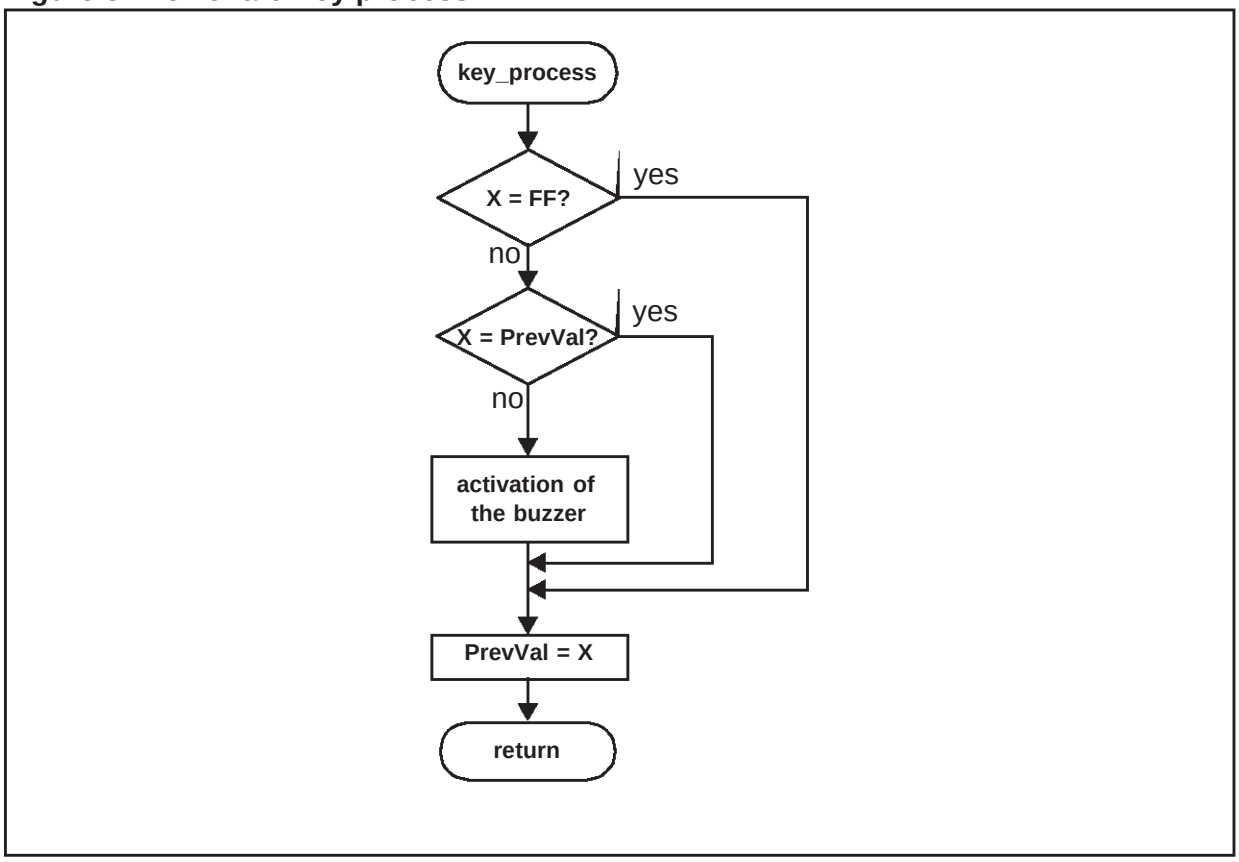


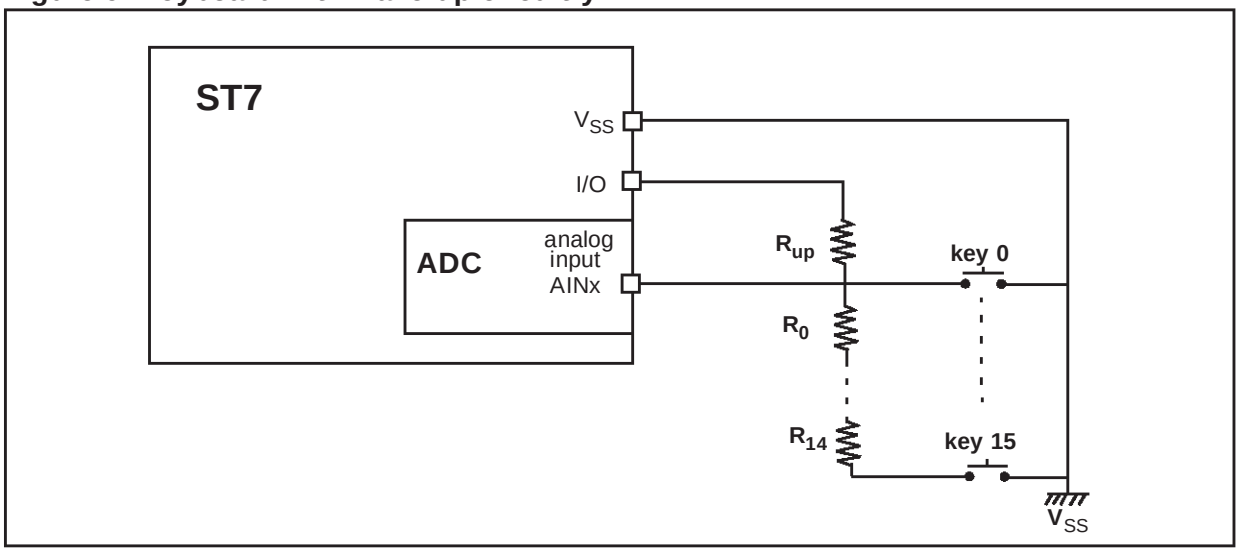
Figure 8. Flowchart: key process



5.3 EXTENSION FOR WAKE UP

With the ST7, it is possible to generate a wake-up operation.This can be achieved by a modification of the circuit. The pull-up resistor must be connected to an additional port pin. During key polling, this pin is in output mode and active high, thus switching V_{DD} to the pull-up resistor.

Figure 9. Keyboard with wake-up circuitry.



During the wait for a key press, the I/O pin used for the pull-up is switched into a high impedance state (e.g. open drain output mode). The pin used as the ADC input while polling is switched to the interrupt input with pull-up mode.

So if any key is pressed an interrupt will be generated if the voltage at this pin is below the Schmitt trigger low level threshold. The serial resistors in the keyboard chain must not be too high in this case, therefore the maximum number of keys is reduced in comparison to the normal mode.

5.4 SOFTWARE

The assembly code given below is guidance only. The complete software with all the files can be found in the software library.

st7/

```
;*************** (c) 1998 STMicroelectronics **********************
;
;PROJECT :       EVALUATION BOARD - ST7
;COMPILER :      ST7 ASSEMBLY CHAIN
;MODULE :        keyb.asm
;CREATION DATE : 02/04/98
;AUTHOR :        8-Bit Micro Application Team
;
;-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
;
;  THE SOFTWARE INCLUDED IN THIS FILE IS FOR GUIDANCE ONLY. STMicroelectronics
;  SHALL NOT BE HELD LIABLE FOR ANY DIRECT, INDIRECT OR CONSEQUENTIAL
;  DAMAGES WITH RESPECT TO ANY CLAIMS ARISING FROM USE OF THIS SOFTWARE.
;
;-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
;
;DESCRIPTION :   ST7 A/D converter driver for the use of an analog keyboard.
;                It is validated on a ST72311 demoboard.
;                Keyboard is connected the Port D analog pin 3.
;                A buzzer, connected to an output pin, is used to
;                notify that a key had been detected.
;                Two routines are dedicated to the A/D converter :
;                      - an initialization routine.
;                      - a routine to recognize the keys of an analog
;                      keyboard & to give the corresponding values.
;
;-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
;
;MODIFICATIONS :
;
; 02/04/98 - V2.0 - New Assembly version.
;
;*****************************************************************************

    TITLE  "sci.ASM"
                            ; This title will appear on each
                            ; page of the listing file.
    MOTOROLA                ; This directive forces the Motorola
                            ; format for the assembly (default).
    #INCLUDE "st72311.inc"  ; Include st72311 registers and
                            ; memory mapping file.
```

```
;*******************************************************************************
;    Macro definitions
;*******************************************************************************

; Miscellaneous values ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

 #define KBD_LINE    3          ; Number of the analog pin of the keyboard.
 #define NO_KEY     $FF

; ADC_CSR Bit definitions ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 #define COCO     7             ; Conversion complete bit.
 #define ADON     5             ; A/D converter ON bit.
 #define CH2      2             ; Channel selection bit2.
 #define CH1      1             ; Channel selection bit1.
 #define CH0      0             ; Channel selection bit0.

;*******************************************************************************
;    RAM SEGMENT
;*******************************************************************************
  BYTES                        ; following addresses are 8 bit length.

  segment byte at 80-FF 'ram0'

; Variable definitions ************************************************

.PrevVal     DC.B       $FF
.tmp1        DC.B       $FF
.tmp2        DC.B       $FF

;*******************************************************************************
;    Public routines (defined here)
;*******************************************************************************
; routines

;*******************************************************************************
;    Extern routines  (defined elsewhere)
;*******************************************************************************
; routines

;*******************************************************************************
;     MACROs SUB-ROUTINES LIBRARY SECTION
;*******************************************************************************

;    (must be placed here and not at the file's end)
```

```
    WORDS

  segment 'rom'

; Constante definitions ****************************************************

; Lower values after conversion for all keys ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
.key         DC.B 0,8,24,40,55,71,87,104,120,135,151,168,184,199,214,231

; Corresponding values for all keys ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
.keyVal      DC.B $0,$1,$2,$3,$4,$5,$6,$7,$8,$9,$A,$B,$C,$D,$E,$F


; Program code *************************************************************

;    ****************************************
;    *                                      *
;    *    SUB-ROUTINES LIBRARY SECTION      *
;    *                                      *
;    ****************************************

;-------------------------------------------------------------------------------
;ROUTINE NAME :  wait_l
;INPUT/OUTPUT :  None.
;DESCRIPTION  :  Waiting routine.
;COMMENTS   :
;-------------------------------------------------------------------------------
.wait_l

        LD   Y,#$2F           ;Give to tmp2 the wanting value. Duration of
        LD   tmp2,Y           ; waiting routine is given by this value.

tst1    LD   Y,tmp1           ; Decrease tmp1.
        DEC  Y
        LD   tmp1,Y
        JRNE tst1             ; Jump to tst1 until tmp1 = 0.
tst2    LD   Y,tmp2           ; Decrease tmp2.
        DEC  Y
        LD   tmp2,Y
        JRNE tst2             ; Jump to tst2 until tmp2 = 0.

        ret
```

```
;-------------------------------------------------------------------------
;ROUTINE NAME :  KBD_val
;INPUT/OUTPUT :  None / value of the keyboard.
;DESCRIPTION  :  Recognize the pressed key of the keyboard,
;                and give the associated value.
;COMMENTS     :  When no key is pressed, the routine returns the value FF.
;-------------------------------------------------------------------------
.KBD_val

wait    btjf ADCCSR,#COCO,wait ; Wait for the end of conversion.
        LD   A,ADCDR           ; Conversion value in A.
        CP   A,#128            ; Compare with 128. Is the value negative?
        JRPL nega              ; Yes, jump to the negative part of the routine.
        JRA  posi              ; No, jump to the positive part of the routine.

   ; Negative value ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
nega    CP   A,#248            ; Special case : A > 247?
        JRPL no_K              ; Yes, no key was pressed, jump to
                               ; no_K. Else, continue.
        LD   X,#16
neg_l   DEC  X                 ; Decrease X, change relative address of key.
        CP   A,(key,X)         ; 'A' greater than value at address X of key?
        JRPL KBD_V             ; Yes, the key is recognized, jump to KBD_V.
        JRA  neg_l             ; No, key not recognized, return into the loop.

   ; Positive value ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
posi    LD   X,#$9
pos_l   DEC  X                 ; Decrease X, change relative address of key.
        CP   A,(key,X)         ; 'A' greater than value at address X of key?
        JRPL KBD_V             ; Yes, the key is recognized, jump to KBD_V.
        JRA  pos_l             ; No, key not recognized, return into the loop.

   ; Key recognized, return value ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
KBD_V   LD   X,(keyVal,X)      ; Put in X the value of the key.
        LD   PFDR,X            ; The converted value appears on LEDS.
        ret

   ; Special case : no key ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
no_K    LD   X,#NO_KEY         ; Put in X the default value : no key pressed.
        ret
```

```
;-------------------------------------------------------------------------
;ROUTINE NAME :  KBD_init
;INPUT/OUTPUT :  None.
;DESCRIPTION  :  Ports and ADC peripheral's initialization routine.
;COMMENTS     :  Do the first conversion to stabilize the converter.
;-------------------------------------------------------------------------
.KBD_init

   ; Initialization of the pin connected to the keyboard ~~~~~~~~~~~~~~~~~~
        bres PDOR,#KBD_LINE
        bres PDDDR,#KBD_LINE   ; Input pin defined as input.
   ; Initialization of the A/D converter ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
        LD   A,#KBD_LINE       ; Select the analog input to convert,
        LD   ADCCSR,A          ; converter disabled.
        bset ADCCSR,#ADON      ; A/D converter switched on.
        LD   A,#$30            ; Waiting more than 30us for
wait11  DEC  A                 ; stabilization.
        JRNE wait11
wait1   btjf ADCCSR,#COCO,wait1 ; Wait the end of the first conversion.

        ret

;    ****************************************
;    *                                      *
;    *     MAIN-ROUTINES SECTION            *
;    *                                      *
;    ****************************************

.main

   ; Initialisations ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
        CALL KBD_init
        BRES PCOR,#1           ; Pin 1 of Port C in output mode.
        BSET PCDDR,#1

   ; Main loop, transmission and reception ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
loop    CALL KBD_val
        CP   X,#$FF            ; Is a key detected?
        JREQ jump              ; No, go to jump, else go on.
        CP   X,PrevVal         ; Is new key the same as previous key?
        JRNE jump              ; No, go to jump, else go on.

   ; Activation of a buzzer when a key is detected ~~~~~~~~~~~~~~~~~~~~~~~~
        LD   Y,#$FF
wait2   LD   A,PCDR
        XOR  A,#$02
        LD   PCDR,A            ; Buzzer on/off.
        LD   A,#$FF
```

```
wait21  DEC  A                ; Waiting to control buzzer frequency.
        JRNE wait217
        DEC  Y                ; Loop to control
        JRNE wait2            ; duration of the buzzer.

jump    LD   PrevVal,X        ; New key value is saved in PrevVal.
        CALL wait_l           ; Waiting before looking at a new value
        JRA  loop


;    ****************************************
;    *                                      *
;    * INTERRUPT SUB-ROUTINES LIBRARY SECTION*
;    *                                      *
;    ****************************************

.dummy_rt    iret      ; Empty subroutine. Go back to main (iret instruction).

    segment 'vectit'
          DC.W  dummy_rt   ;FFE0-FFE1h location
          DC.W  dummy_rt   ;FFE2-FFE3h location
i2c_it:   DC.W  dummy_rt   ;FFE4-FFE5h location
          DC.W  i2c_rt     ;FFE6-FFE7h location
          DC.W  dummy_rt   ;FFE8-FFE9h location
          DC.W  dummy_rt   ;FFEA-FFEBh location
          DC.W  dummy_rt   ;FFEC-FFEDh location
timb_it:  DC.W  dummy_rt   ;FFEE-FFEFh location
          DC.W  dummy_rt   ;FFF0-FFF1h location
tima_it   DC.W  dummy_rt   ;FFF2-FFF3h location
spi_it:   DC.W  dummy_rt   ;FFF4-FFF5h location
          DC.W  dummy_rt   ;FFF6-FFF7h location
io_bc_it: DC.W  dummy_rt   ;FFF8-FFF9h location
io_a_it:  DC.W  dummy_rt   ;FFFA-FFFBh location
softit:   DC.W  dummy_rt   ;FFFC-FFFDh location
reset:    DC.W  main       ;FFFE-FFFFh location

        END

;*** (c) 1998 STMicroelectronics ************************** END OF FILE ****
```

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 CONNEXION WITH THEIR PRODUCTS.