APPLICATION NOTE

ST7 KEYPAD DECODING TECHNIQUES, IMPLEMENTING WAKE-UP ON KEYSTROKE

by Microcontroller Division Application Team

INTRODUCTION

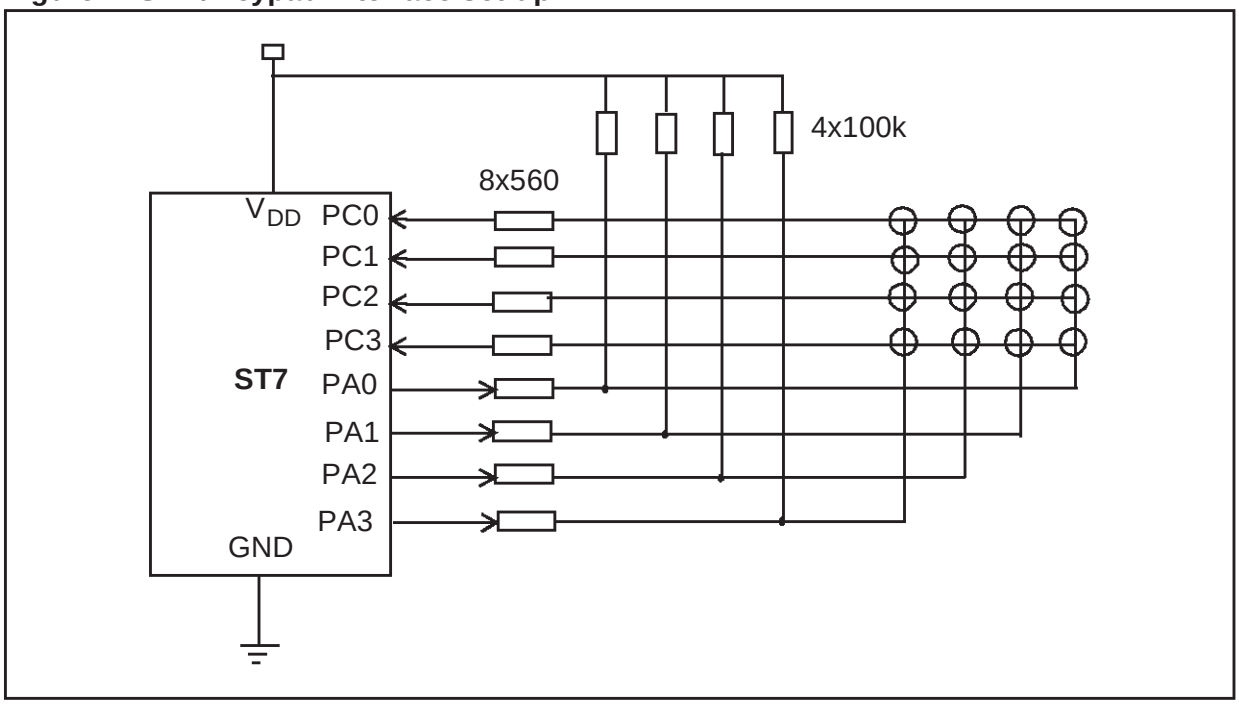
The goal of this application note is to present an example of the use of the HALT mode.

In this application, the MCU (here a ST72251) is waked up by an external interrupt caused by someone pressed a key on the 4x4 matrixed keypad.

1 ST7 / KEYBOARD INTERFACE

Rows are connected to inputs with pull-up and interrupts (Port C). Columns are connected to Port A configured as output. The result of the interrupt (the value of the pressed key) is sent on LEDS (Port B) and stored into the X register. In our configuration, we have to add 4 pull-up resistors on Port A (from PA0 to PA3) to be able to apply a high level on the corresponding pads.

Figure 1. ST7 / keypad interface set-up



2 ST72251 CONFIGURATION

The application has been validated with a ST72251. Its configuration is described in this part. Refer to your datasheet for more details.

2.1 I/O CONTROL

Rows are connected to pins configured as inputs (Port C as input with pull up and interrupts). Columns are connected to pins configured as outputs (Port A).

External interrupts are caused by a low level applied to a pin of Port C (caused by a key pressed), they wake up the MCU which was in HALT mode.

Port B is configured as outputs to send the value of the pressed key on LEDS.

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

2.2 MISCELLANEOUS REGISTER

Bits 7 and 6 have to be set to configure events correctly: the external interrupt (EI1) has here to be caused by a falling edge only.

Please, refer to the datasheet for more details.

2.3 HALT MODE

The HALT instruction places the ST72251 in its lowest power consumption mode. The core and all peripherals are frozen. In this mode, the internal oscillator is turned off, causing all internal processing to be halted. The data remain unchanged. During the HALT mode, external interrupts are still enabled. The MCU stays in this state until an external interrupt or a reset occurs. Then the internal oscillator is restarted and the core waits for 4096 CPU clock cycles (512 µs for a f_{CPU} = 8MHz) before running the external interrupt subroutine. Then the MCU comes back to the main program (in our application to the HALT state).

Please, refer to the datasheet for more details.

3 EXTERNAL INTERRUPTS

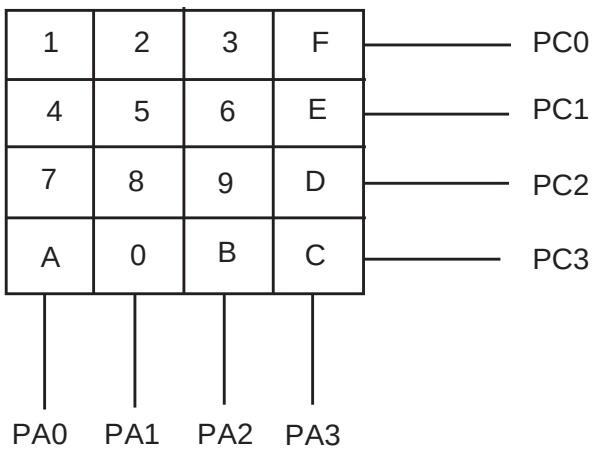
The MCU is in HALT mode. When a key is pressed, a low level is applied to the pin corresponding to the row the key belongs (pins configured as inputs with pull-up). It's a falling edge applied to a pin of Port C which creates an external interrupt (EI1) and wakes up the MCU. The MCU executes then the external interrupt subroutine (decoding the pressed key) and comes back to its previous state (HALT state in the main program).

4 KEYPAD

The keypad used is a 4x4 matrixed keypad. Rows are connected to pins configured as inputs with pull-up. So the initial state of these pins are a high level (1). When a key is pressed, a low level is applied to the corresponding pin. For this reason, the keypad is coded as follows:

Table 1. Key values

KEY	row value	column value	KEY	row value	column value
1	0x0E	0x0E	7	0x0B	0x0E
2	0x0E	0x0D	8	0x0B	0x0D
3	0x0E	0x0B	9	0x0B	0x0B
F	0x0E	0x07	D	0x0B	0x07
4	0x0D	0x0E	A	0x07	0x0E
5	0x0D	0x0D	0	0x07	0x0D
6	0x0D	0x0B	B	0x07	0x0B
E	0x0D	0x07	C	0x07	0x07



You have to press the chosen key at least 0.5 to 1 second depending on which key you choose (table read from keypad_top to keypad). The faster the key is read into the table, the faster it will be decoded and the faster the result will be sent on LEDS.

The keypad code is in the file constant.asm as follows:

```
.keypad     DC.B    $0E,$0E,$1      ;PC0PA0
            DC.B    $0E,$0D,$2      ;PC0PA1
            DC.B    $0E,$0B,$3      ;PC0PA2
            DC.B    $0E,$07,$F      ;PC0PA3
            DC.B    $0D,$0E,$4      ;PC1PA0
            DC.B    $0D,$0D,$5      ;PC1PA1
            DC.B    $0D,$0B,$6      ;PC1PA2
            DC.B    $0D,$07,$E      ;PC1PA3
            DC.B    $0B,$0E,$7      ;PC2PA0
            DC.B    $0B,$0D,$8      ;PC2PA1
            DC.B    $0B,$0B,$9      ;PC2PA2
            DC.B    $0B,$07,$D      ;PC2PA3
            DC.B    $07,$0E,$A      ;PC3PA0
            DC.B    $07,$0D,$0      ;PC3PA1
            DC.B    $07,$0B,$B      ;PC3PA2
keypad_top  DC.B    $07,$07,$C      ;PC3PA3
```

5 FLOWCHARTS

Figure 2. Flowchart: Main program

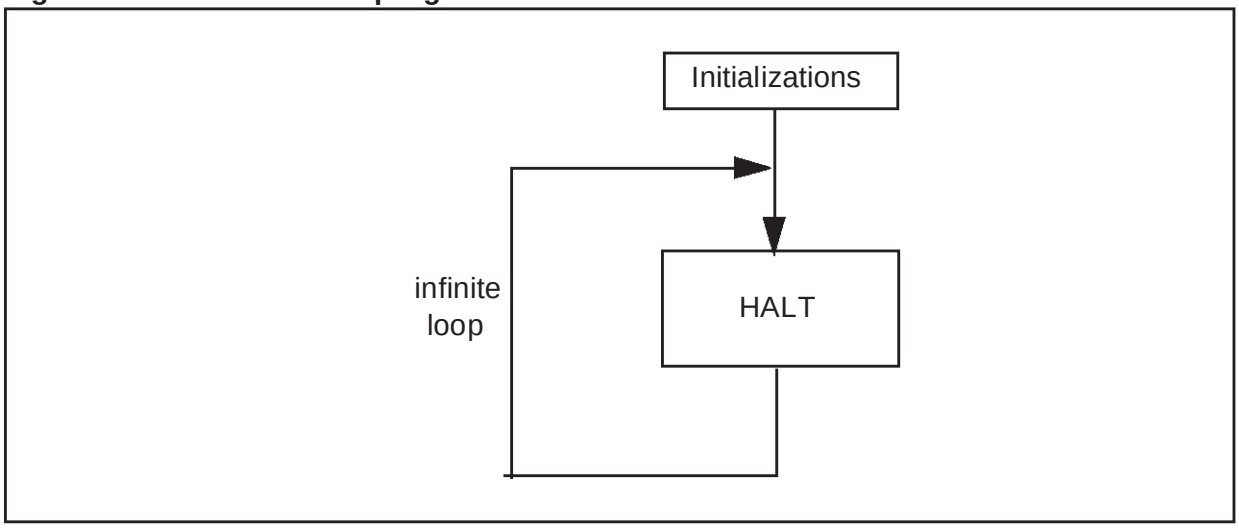
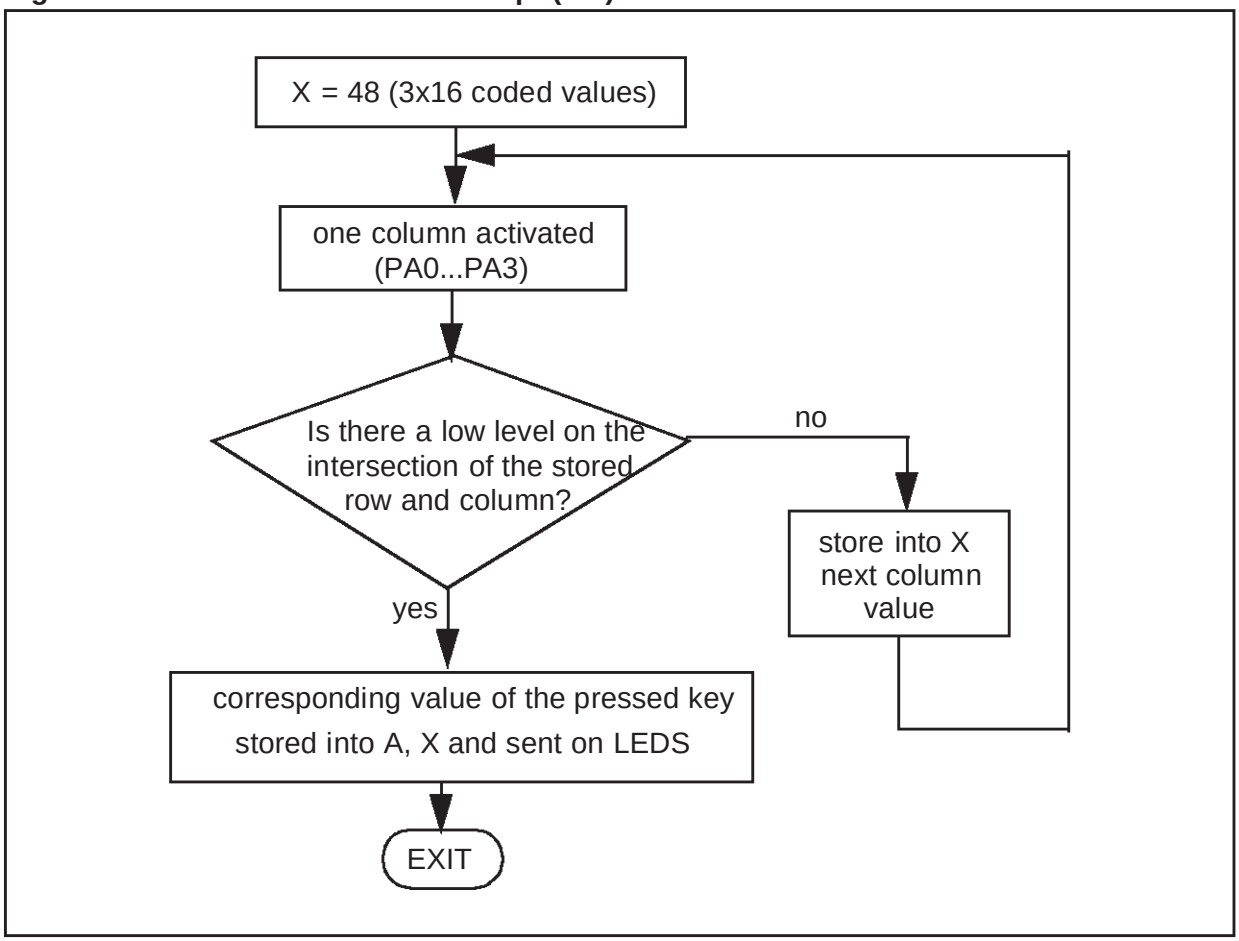


Figure 3. Flowchart: external interrupt (EI1)



6 SOFTWARE

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

```
st7/                ; the first line is reserved
                    ; for specifying the instruction set
                    ; of the target processor
;*********************************************************************
; TITLE:            WAKE.ASM
; AUTHOR:           PPG Microcontroller Applications Team
; DESCRIPTION:       Main program (use of the Halt mode for decoding
;                   a keypad).
;
;*********************************************************************
    TITLE  "WAKE.ASM"
                     ; this title will appear on each
                     ; page of the listing file
    MOTOROLA        ; this directive forces the Motorola
                     ; format for the assembly (default)
    #INCLUDE "st72251.inc"; include st72251 registers and memory mapping file
    #INCLUDE "constant.inc"; include general constants file

;*********************************************************************
;    Variables, constants defined and referenced locally
;    You can define your own values for a local reference here
;*********************************************************************
;*********************************************************************
;    Public routines (defined here)
;*********************************************************************
    WORDS
  segment 'rom'
.Init
    LD   A,#$80     ; interrupts are caused by falling edge (on Port C).
    LD   MISCR,A

    LD   A,#$0F
    LD   PBDDR,A
    LD   PBOR,A     ; PB0 to PB3 configures as outputs(LEDS).

    LD   PADDR,A
    CLR PAOR        ; PA0 to PA3 configured as outputs.
    CLR   PADR
    LD   A,#$0F
```

```
    LD   PCOR,A
    CLR   PCDDR     ; PC0 to PC3 configured as input with pull-up and interrupt.
    CLR   PCDR

    RET

.delay_45
    LD   A,#200
loop2  LD   X,#$FF
loop1  DEC   X
    JRNE  loop1
    DEC   A
    JRNE  loop2

    RET

;*************************************************************
;    Program code
;*************************************************************
.main
      CALL   Init

loop
      HALT          ; MCU put in lowest power mode.
      JRA   loop      ; Infinite loop, wait an interrupt occurs.

; *****************************************************************
; This set of instructions uses simple assembly mnemoniques.
; We can notice that the loop label is defined only locally (no dot
; in front of it) so it can not be seen by others modules linked
; with this file.
; *****************************************************************

;    ****************************************
;    *                      *
;    * INTERRUPT SUB-ROUTINES LIBRARY SECTION *
;    *                      *
;    ****************************************
.dummy  iret
.sw_rt  iret   ; Empty subroutine. Go back to main (iret instruction)
.ext0_rt iret
.ext1_rt
      LD   X,#48              ; Size of the table stored in X.
                              ; 3 x 16 coded values.
begin
```

```
       LD    A,PADR
       OR    A,#$0F              ; Outputs forced to 1 (initial value).
       AND    A,({keypad+1},X)
       LD    PADR,A              ; one column of the tested row is activated.

       PUSH   X
       CALL   delay_45           ; wait 45ms (debouncing procedure).
       POP    X

       LD    A,PCDR
       AND    A,#$0F             ; value of Port C stored into A.
       CP    A,(keypad,X)        ; search of the low level column after column.
       JREQ   OK                 ; if key found -> OK.
       DEC    X
       DEC    X
       DEC    X                  ; 3 times to have next value of the column.
       JRPL   begin              ; do it again to read the whole table(matrix).

       JRA    exit
OK
       LD    A,({keypad+2},X)    ; Store the key value in A.
       LD    X,A                 ; Copy it into X.
       LD    PBDR,X              ; Output the result on LEDS.


exit
       CLR    PADR
       CLR    PCDR

       iret
.spi_rt iret
.tima_rt iret
.timb_rt iret
.i2c_rt iret


    segment 'vectit'
; *****************************************************************
; This last segment should always be there in your own programs.
; It defines the interrupt vector adresses and the interrupt routines' labels
; considering the microcontroller you are using.
; Refer to the MCU's datasheet to see the number of interrupt vector
; used and their addresses.
; Remind that this example is made for a ST72251 based application.
; *****************************************************************
```

```
; *************************************************************
; Each interrupt vector uses two addresses in rom, that's what the directive
; DC.W means. It says "reserve a word location (.W) in rom (DC) and code
; the routine's label in those two addresses.
; Yet, when an interrupt occurs, for example from the timerB, timerb's routine
; address (timb_rt) will be loaded in the PC and the program will jump to this
; label if allowed. It will execute this routine and then will go back to the main
; program (see interrupt chapter in the datasheet for a more precise description
; of how to handle interrupts in ST72 micros).
; *************************************************************
          DC.W  dummy    ;FFE0-FFE1h location
          DC.W  dummy    ;FFE2-FFE3h location
.i2c_it     DC.W  i2c_rt  ;FFE4-FFE5h location
          DC.W  dummy    ;FFE6-FFE7h location
          DC.W  dummy    ;FFE8-FFE9h location
          DC.W  dummy    ;FFEA-FFEBh location
          DC.W  dummy    ;FFEC-FFEDh location
.timb_it    DC.W  timb_rt;FFEE-FFEFh location
          DC.W  dummy    ;FFF0-FFF1h location
.tima_it    DC.W  tima_rt;FFF2-FFF3h location
.spi_it     DC.W  spi_rt  ;FFF4-FFF5h location
          DC.W  dummy    ;FFF6-FFF7h location
.ext1_it    DC.W  ext1_rt;FFF8-FFF9h location
.ext0_it    DC.W  ext0_rt;FFFA-FFFBh location
.softit   DC.W  sw_rt    ;FFFC-FFFDh location
.reset    DC.W  main     ;FFFE-FFFFh location
       ; This last line refers to the first line.
       ; It used by the compiler/linker to determine code zone
    END ; Be aware of the fact that the END directive should not
       ; stand on the left of the page like the labels's names.
```

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.