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 ST72F264) 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 ST72F264 CONFIGURATION

The application has been validated with a ST72F264. 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 ST72F264 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 fCPU = 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

<table>
<thead>
<tr>
<th>KEY</th>
<th>row value</th>
<th>column value</th>
<th>KEY</th>
<th>row value</th>
<th>column value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0x0E</td>
<td>0x0E</td>
<td>7</td>
<td>0x0B</td>
<td>0x0E</td>
</tr>
<tr>
<td>2</td>
<td>0x0E</td>
<td>0x0D</td>
<td>8</td>
<td>0x0B</td>
<td>0x0D</td>
</tr>
<tr>
<td>3</td>
<td>0x0E</td>
<td>0x0B</td>
<td>9</td>
<td>0x0B</td>
<td>0x0B</td>
</tr>
<tr>
<td>F</td>
<td>0x0E</td>
<td>0x07</td>
<td>D</td>
<td>0x0B</td>
<td>0x07</td>
</tr>
<tr>
<td>4</td>
<td>0x0D</td>
<td>0x0E</td>
<td>A</td>
<td>0x07</td>
<td>0x0E</td>
</tr>
<tr>
<td>5</td>
<td>0x0D</td>
<td>0x0D</td>
<td>0</td>
<td>0x07</td>
<td>0x0D</td>
</tr>
<tr>
<td>6</td>
<td>0x0D</td>
<td>0x0B</td>
<td>B</td>
<td>0x07</td>
<td>0x0B</td>
</tr>
<tr>
<td>E</td>
<td>0x0D</td>
<td>0x07</td>
<td>C</td>
<td>0x07</td>
<td>0x07</td>
</tr>
</tbody>
</table>

![Keypad Diagram]

PC0
PC1
PC2
PC3
PA0
PA1
PA2
PA3
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:

```
.kepad  DC.B  $0E,$0E,$1   ;PC0 PA0
        DC.B  $0E,$0D,$2   ;PC0 PA1
        DC.B  $0E,$0B,$3   ;PC0 PA2
        DC.B  $0E,$07,$F   ;PC0 PA3
        DC.B  $0D,$0E,$4   ;PC1 PA0
        DC.B  $0D,$0D,$5   ;PC1 PA1
        DC.B  $0D,$0B,$6   ;PC1 PA2
        DC.B  $0D,$07,$E   ;PC1 PA3
        DC.B  $0B,$0E,$7   ;PC2 PA0
        DC.B  $0B,$0D,$8   ;PC2 PA1
        DC.B  $0B,$0B,$9   ;PC2 PA2
        DC.B  $0B,$07,$D   ;PC2 PA3
        DC.B  $07,$0E,$A   ;PC3 PA0
        DC.B  $07,$0D,$0   ;PC3 PA1
        DC.B  $07,$0B,$B   ;PC3 PA2
    keypad_top  DC.B  $07,$07,$C   ;PC3 PA3
```
5 FLOWCHARTS

Figure 2. Flowchart: Main program

Initializations

infinite loop

HALT

Figure 3. Flowchart: external interrupt (EI1)

$X = 48 \text{ (3x16 coded values)}$

one column activated (PA0...PA3)

Is there a low level on the intersection of the stored row and column?

yes

no

store into X next column value

 corresponding value of the pressed key stored into A, X and sent on LEDs

EXIT
6 SOFTWARE

All the source files in assembly code are given in the zip file with this application note.

The source files are 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.
ST7 KEYPAD DECODING TECHNIQUES, IMPLEMENTING WAKE-UP ON KEYSTROKE

“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.”