



Application Note 15

Title: **How to implement a keyboard matrix 4x4**
Product Family: **4 bits Microcontroller**

Part Number: EM6640, EM6540
Keywords: 4 bits micro-controller, keyboard matrix 4x4
Date: January 07, 2002 REV. B/416

The following document describes the implementation of a keyboard matrix 4x4. The process has been implemented as a subroutine.

With a keyboard common at 16 buttons, you need 16 inputs in the micro-controller with small instructions' lines for read directly the value. But with a keyboard matrix 4x4, you need only 4 outputs and 4 inputs with more instructions for read the external value.

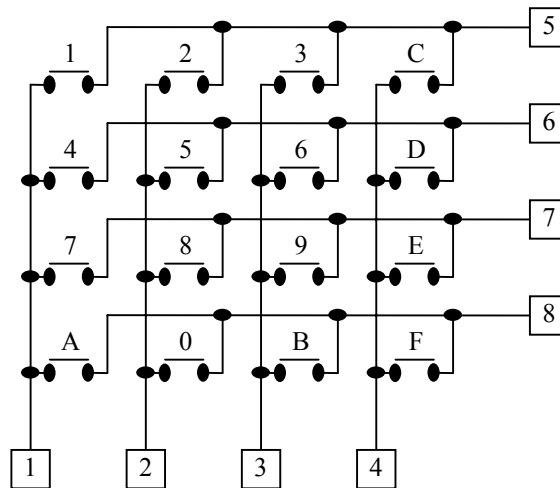


Figure 1: structure of the keyboard.

The connector 1-4 are connected (figure 1) on the output port B and the connector 5-8 are on the input port A for this example.

The interrupt prescaler is used to change by a rotate left the bit selected on the port B. An interrupt port A is generate on a button pressed. The interrupt value is saving in the variable "cord_x" and the output port B is saving in the variable "cord_y". A subroutine is calling for convert the right value returned in a variable.

This example is create on the EM6X40 it use 586 Hz prescaler IRQ, but you can use it with an another micro-controller. You must select an another prescaler's frequency (ex.: 32 Hz). In the principal routine, you can execute another instructions.

In this example, four leds are driven by an output bit to display the result of the buttons pressed. The LED's anode are connected on the port B and the LED's cathode are connect on PC0.



Appendix A: Main program for check the keyboard

```
-----
;
; PROGRAM : MATRIX44.ASM
; Date last mod : 03/12/1998 Ch.Mayer
; Demo of a keyboard matrix 4x4, with 4 external led drive by 1 output.
; Uses the 6640REG.ASM Include file
;
; Port A : Data to the keyboard and the Led.
; Port B : Data from the keyboard.
; PC0 : Data multiplexed to the Led
;
; The time's base for scanning the keyboard is the prescaler 586 Hz.
;
-----
;
; 6540 registers
;
-----
;
; INCLUDE 6640REG.ASM
;
-----
;
; Constants
;
-----
;
; Variables
;
Stack0 EQU 00H ; Stack0
Cord_X EQU 01H ; Save IRQ1
Cord_Y EQU 02H ; Save PortB
Value EQU 03H ; Value from the keyboard
;
-----
;
; Code Offset
;
-----
;
; ORG 0
;
Reset: JMP Main
;
-----
;
; Interrupt Handler
;
-----
;
Handler:
; STA STACK0 ; Save ACCU value
;
Hand00:
; LDR IRQ1 ; Test the interrupt port A
; STA Cord_X
; JPZ Hand10
; LDR RegPBData
; STA Cord_Y
; CALL Table
;
Hand10:
; LDI 04H ; Test the interrupt prescaler 586 Hz
```



```
AND    IRQ2
JPZ    HandEnd
SHL    RegPBData      ; Rotate the bit on the port B
STA    RegPBData
JPNC   Hand15
STI    RegPBData, 01H

Hand15:
CALL   LEDOUT

HandEnd:
STI    RegSys2, 08H    ; Reset the watchdog
LDR    STACK0         ; Reload ACCU
RTI

;-----
;      Main
;-----

Main:
STI    RegSys3, 03H    ; Disable the watchdog
STI    RegSys1, 08H    ; Enable the interrupt general
STI    IRQM1, 0FH     ; Enable the interrupt port A
STI    Presc, 01H     ; Select the debouncer CK[17]
STI    IRQM2, 04H     ; Enable the interrupt prescaler 586 Hz
STI    RegPBCntl, 0FH ; Port B in output mode
STI    RegPCCntl, 0FH ; Port C in output mode
STI    RegPBData, 01H ; Reset the port B
STI    RegPCData, 00H ; Reset the port C

WAIT:  HALT
      NOP
      JMP   WAIT

      INCLUDE    TABLES.ASM ; Include file for convert the matrix in value

;-----
;      END
;-----
```



Appendix B: Module with the table of keyboard

```
-----  
; MODULE      : TABLES.ASM  
; Date last mod : 03/12/1998 Ch.Mayer  
; Module for convert the matrix value in hex value  
; Uses the 6640REG.ASM Include file  
;  
; Cord_X  
;  
; C 1 2 3 C  
; o 4 5 6 D  
; r 7 8 9 E  
; d_Y A 0 B F  
;  
-----  
; Cord_X and Cord_Y => Value  
-----  
TABLE:  
SHRR Cord_X ; Select the coordinate X  
JPC TABLE100  
SHRA  
JPC TABLE200  
SHRA  
JPC TABLE300  
SHRA  
JPC TABLE400  
JMP TABLEND  
  
TABLE100: ; Int portA = 01H  
SHRR Cord_Y  
JPC TABLE110  
SHRA  
JPC TABLE120  
SHRA  
JPC TABLE130  
SHRA  
JPC TABLE140  
JMP TABLEND  
  
TABLE110:  
STI Value, 01H  
JMP TABLEND  
  
TABLE120:  
STI Value, 02H  
JMP TABLEND  
  
TABLE130:  
STI Value, 03H  
JMP TABLEND  
  
TABLE140:  
STI Value, 0CH  
JMP TABLEND  
  
TABLE200: ; Int portA = 02H  
SHRR Cord_Y  
JPC TABLE210  
SHRA
```



```
JPC TABLE220
SHRA
JPC TABLE230
SHRA
JPC TABLE240
JMP TABLEND
TABLE210:
STI Value, 04H
JMP TABLEND
TABLE220:
STI Value, 05H
JMP TABLEND
TABLE230:
STI Value, 06H
JMP TABLEND
TABLE240:
STI Value, 0DH
JMP TABLEND

TABLE300: ; Int portA = 04H
SHRR Cord_Y
JPC TABLE310
SHRA
JPC TABLE320
SHRA
JPC TABLE330
SHRA
JPC TABLE340
JMP TABLEND
TABLE310:
STI Value, 07H
JMP TABLEND
TABLE320:
STI Value, 08H
JMP TABLEND
TABLE330:
STI Value, 09H
JMP TABLEND
TABLE340:
STI Value, 0EH
JMP TABLEND

TABLE400: ; Int portA = 08H
SHRR Cord_Y
JPC TABLE410
SHRA
JPC TABLE420
SHRA
JPC TABLE430
SHRA
JPC TABLE440
JMP TABLEND
TABLE410:
STI Value, 0AH
JMP TABLEND
TABLE420:
STI Value, 00H
JMP TABLEND
```



TABLE430:

STI Value, 0BH
JMP TABLEND

TABLE440:

STI Value, 0FH
JMP TABLEND

TABLEND:

RET

```
-----  
; Value => Cord_X and Cord_Y  
-----
```

LEDOUT:

LDR RegPBData
AND Value
JPNZ LEDOUT100
STI RegPCdata, 01H ; Power off the led
JMP LEDOUTEND

LEDOUT100:

STI RegPCData, 00H ; Power on the led

LEDOUTEND:

RET

References

Application Note #1: *How to rotate left a register through carry*
Application Note #6: *Creating Data Tables in ROM*