```
// S/W Environment : AVR Studio + WINAVR Compiler
// Target : M128
// Crystal: 16Mhz
//
// Author: chowk.

// Key Scan Output Port : ROW_PORT(PC4 - PC7)
// Key Input Port : COL_PORT(PE4 - PE7)
// Debouncing : delay_10uSec(unsigned int time_10us) 함수를 이용 한다.

#include <avr/io.h>
#include <avr/interrupt.h>
#include <stdio.h>

#include "get_keypad.h"
#include "delay_u_mSec.h"

#define ROW_PORT PORTC                  // Output Port
#define ROW_PORT_DDR DDRC
#define COL_PORT PORTE                  // Input Port
#define COL_PORT_DDR DDRE
#define COL_PORT_PIN PINE               // Input Port

#define ROW_MASK 0xf0
#define COL_MASK 0x70
#define NUM_ROWS 4
#define NUM_COLS 3
#define COL_TEST 0x40   // NUM_COLS = 3 : 0x40, NUM_COLS =4 : 0x80

#define TRUE 0xff
#define FALSE 0x00

void keypad_init(void);   // I/O Port, timer3 init
unsigned char get_key( void );

volatile unsigned char keyCode;
volatile static unsigned int keyPressed=FALSE,pressedKeyNum;

// Keyboard matrix key mapping(4 by 3).
```

```
struct keyMatrix {
        unsigned char rowScanCode;
        unsigned char keyCode[3];};

typedef const struct keyMatrix keyMatrixType;
keyMatrixType *Pt;

keyMatrixType keyPad[5]={
{   0x10, "123" }, // row 0
{   0x20, "456" }, // row 1
{   0x40, "789" }, // row 2
{   0x80, "*0#" }, // row 3
{   0x00, "     " }};

void key_port_init(void)
{
    // Row Key Scan Output Port : PORTC(PC4 - PC7)
        ROW_PORT |= ROW_MASK;          // Key Scan Output Port Disable <- 1
        ROW_PORT_DDR  |= ROW_MASK;  // Key Scan Output Port : PC4-PC7

    // Col Key Input Port : PORTE(PE4 - PE6)
        COL_PORT |= COL_MASK;              // External SW PullUp
        COL_PORT_DDR  &= (~COL_MASK);  // External Interrupt Sensing Port : PE4 - PE6
}

void keypad_init(void)   // I/O Port, timer3 init
{
        key_port_init();
        keyPressed = FALSE;
}

void key_scan(){
        keyMatrixType *pt;
        unsigned char colData;
        signed int j;

        pressedKeyNum = 0;    // 현재 Pressed Key의 수(정상 상태에서는 1,
                              // 그러나 경우에 때라 2개의 Key 가 동시에 Pressed될 수 있다.)
        keyCode=0;            // Default values
```

```
        pt=&keyPad[0];

        ROW_PORT |= ROW_MASK;   // Scan Disable, Input port Rg를 1로 초기화
        while(pt->rowScanCode){
                // Scan Code가 출력되고 현재 SW 상태가 Input port에 Set 된다.
                ROW_PORT &= ~(pt->rowScanCode);
                asm volatile(" NOP");
                colData = COL_PORT_PIN & COL_MASK;      // Read columns
                ROW_PORT |= ROW_MASK;   // Scan Disable, Input port Rg를 1로 초기화
                for(j = (NUM_COLS - 1); j >= 0; j--){
                        if((colData & COL_TEST) == 0){
                                keyCode = pt->keyCode[j];
                                pressedKeyNum++;
                        }
                        colData <<= 1;    // Shift into position
                }

                pt++;
        }

        // Clear External Interrupt Flag
        // Key scaning 과정에서 External Interrupt Pin의 상태가 변동(현재 Key가 Pressed된 상태
에서
        // Key scan을 위하여 해당 Row의 상태 1 -> 0 로 변동 시키면 External Interrupt Flag 가
Set 되기
        // 때문에 GetKey() 함수에서 다음 Key Stroke 을 읽기 위하여 External Interrupt을 Enable
하면
        // 현재 Key 상태가 반복하여 입력되는 문제가 발생한다.
        // 이러한 문제를 예방 하기 위하여 External Interrupt Flag를 Clear 한다.
        EIFR = COL_MASK;
}

#define DEBOUNCE_TIME 16    // 16mSec Delay
unsigned char get_key( void ){
        unsigned char colData;

        keyCode = 0;
        // 유효한 keyCode를 얻을 때 까지 while loop를 반복 한다.
        while((keyCode == 0)){
```

```
                ROW_PORT &= ~ROW_MASK;      // Row Scan Enable
                asm volatile(" NOP");

                colData = COL_PORT_PIN & COL_MASK;

                if(colData == COL_MASK){          // Key 가 모두 Release 된 경우
                        if(keyPressed == TRUE){   // 만약 이전에 Key 가 Pressed 상태 였으면
                                keyPressed = FALSE;
                                delay_1mSec(DEBOUNCE_TIME);
                        }
                        // 만약 이전에 Key 가 Release 인 경우 while loop를 계속 한다.
                }
                else {              // Pressed 된 Key 가 있는 경우
                  // 새 Key가 이제 막 Pressed 됨
                  if(keyPressed != TRUE){
                      delay_1mSec(DEBOUNCE_TIME);
                      key_scan();
                      if( keyCode != FALSE){
                          keyPressed = TRUE;
                          // 유효한 keyCode 가 입력 되었기 때문에 keyCode 값을 return 한다.
                       }
                   }
                          // Key Pressed 상태가 계속되고 있는 경우 while loop를 계속 한다.
                }
        }
        return(keyCode);
}
```