```
/*********************************************************************
* KeyUcos.c - A MicroC/OS keypad module for a 4x4 matrix keypad.
*             Based on Rev1 of Key.c
* 02/15/2007 Todd Morton
* 03/03/09 TDM Bug fixes
* Modified by Mark Moerdyk 2/6/13 - 2/12/13
*********************************************************************
* Project master header file
*********************************************************************/
#include "includes.h"
/*********************************************************************
* Module Defines
*********************************************************************/
typedef enum{KEY_OFF,KEY_EDGE,KEY_VERF} KEYSTATES;
#define  KEY_DIR_PORT DDRB
#define  KEY_DATA_PORT PORTB
#define  COLS (INT8U)0x0F
#define DC1 (INT8U)0x11
#define DC2 (INT8U)0x12
#define DC3 (INT8U)0x13
#define DC4 (INT8U)0x14



typedef struct{
    INT8U buffer;
    OS_EVENT *flag;
}SYNCHBUF;
/*********************************************************************
* Public Resources
*********************************************************************/
INT8U KeyPend(INT16U tout, INT8U *err);  /* Returns current value of KeyBuffer*/
void KeyInit(void);             /* Keypad Initialization    */
/*********************************************************************
* Allocate MicroC/OS task stack space.
*********************************************************************/
static OS_STK  KeyTaskStk[KEY_STK_SIZE];
/*********************************************************************
* Private Resources
*********************************************************************/
static void KeyTask(void *pdata);      /* Main keypad read task    */
static INT8U KeyScan(void);         /* Makes a single keypad scan  */
static const INT8U KeyCodeTable[16] =
   {'1','2','3',DC1,'4','5','6',DC2,'7','8','9',DC3,'*','0','#',DC4};
static SYNCHBUF Key;          /* Key buffer and flag.             */
/*********************************************************************
* KeyPend() - A function to provide access to the key buffer via a
*             semaphore.
*    - Public
*********************************************************************/
INT8U KeyPend(INT16U tout, INT8U *err){
    OSSemPend(Key.flag,tout,err);
    return Key.buffer;
}

/*********************************************************************
* KeyInit() - Initialization routine for the keypad module
*             The columns are normally set as inputs and, since they
*             are pulled high, they are one. Then to pull a row low
*             during scanning, the direction for that pin is changed
*             to an output.
*********************************************************************/
void KeyInit(void){
    KEY_DATA_PORT = 0x00;            /* Preset all rows to zero    */
    Key.flag = OSSemCreate(0);
    (void)OSTaskCreate(KeyTask, (void *)0, (void *)&KeyTaskStk[KEY_STK_SIZE],
                       KEYTASK_PRIO);
}

/*********************************************************************
* KeyTask() - Read the keypad and updates KeyBuffer.
*             A task decomposed into states for detecting and
*             verifying keypresses. This task should be called
*             periodically with a period greater than the worst case
*             switch bounce time and less than the shortest switch
*             activation time minus the bounce time. The switch must
*             be released to have multiple acknowledged presses.
*********************************************************************/
static void KeyTask(void *p_arg) {

    INT8U cur_key;
    INT8U last_key = 0;
    KEYSTATES KeyState = KEY_OFF;

    (void)p_arg;
    FOREVER(){
        DBUG_PORT &= ~PP6;
        OSTimeDly(8);
        DBUG_PORT |= PP6;
        cur_key = KeyScan();
        if(KeyState == KEY_OFF){    /* Key released state */
            if(cur_key != 0){
                KeyState = KEY_EDGE;
            }else{ /* wait for key press */
            }
        }else if(KeyState == KEY_EDGE){     /* Keypress detected state*/
            if(cur_key == last_key){        /* Keypress verified */
                KeyState = KEY_VERF;
                Key.buffer = KeyCodeTable[cur_key - 1]; /*update buffer */
                (void)OSSemPost(Key.flag);   /* Signal new data in buffer */
            }else if( cur_key == 0){        /* Unvalidated, start over */
                KeyState = KEY_OFF;
            }else{                          /*Unvalidated, diff key edge*/
            }
        }else if(KeyState == KEY_VERF){     /* Keypress verified state */
            if((cur_key == 0) || (cur_key != last_key)){
                KeyState = KEY_OFF;
            }else{ /* wait for release or key change */
            }
        }else{ /* In case of error */
            KeyState = KEY_OFF;             /* Should never get here */
        }
        last_key = cur_key;                 /* Save key for next time */
    }
}

/*********************************************************************
* KeyScan() - Scans the keypad and returns a keycode.
*           - Designed for 4x4 keypad with columns pulled high.
*           - Current keycodes follow:
*              1->0x01,2->0x02,3->0x03,A->0x04
*              4->0x05,5->0x06,6->0x07,B->0x08
*              7->0x09,8->0x0A,9->0x0B,C->0x0C
*              *->0x0D,0->0x0E,#->0x0F,D->0x10
*           - Returns zero is no key is pressed.
*           - ColTable[] can be changed to distinguish multiple keys
```

```
*            pressed in the same row.
* (Private)
*********************************************************************/
static INT8U KeyScan(void) {

    INT8U kcode;
    INT8U roff, rbit;
    const INT8U ColTable[16] = {0,1,2,2,3,3,3,3,4,4,4,4,4,4,4,4};

    rbit = 0x10;
    roff = 0x00;
    while(rbit != 0x00){ /* Until all rows are scanned */
        KEY_DATA_PORT = 0x00;
        KEY_DIR_PORT = rbit;    /* Pull row low */
        kcode = ((~KEY_DATA_PORT) & COLS);  /*Read columns */
        KEY_DIR_PORT = 0x00;
        if(kcode != 0){        /* generate key code if key pressed */
            kcode = roff + ColTable[kcode];
            break;
        }
        rbit = rbit<<1;       /* setup for next row */
        roff += 4;
    }
    return kcode;
}
/*********************************************************************/
```