```
/*********************************************************************
* Key.c - A keypad module for a 4x4 matrix keypad.
* Revised to avoid shorts when two keys are pressed. This version
* allows for multiple column keys to be pressed and mapped to a
* different code by changing ColTable[] in KeyScan(). Multiple rows
* can not be resolved. The topmost button will be used.
* The KeyCodeTable[] is currently set to generate ASCII codes.
* 02/20/2001 Todd Morton
*
* Rev.1
* Changed the ASCII codes for the function keys, A-D, to DC1-DC4.
* This is more appropriate for applications that use Multitap alpha
* entry using the numeric keys.
* 02/08/07 Todd Morton

*Modified by Mark Moerdyk
*********************************************************************
* 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
#define DB_PORT PTP
#define DB_DDR DDRP
#define PP5 32

/*********************************************************************
* Public Resources
*********************************************************************/
INT8U GetKey(void);           /* Returns current value of KeyBuffer*/
void KeyInit(void);             /* Keypad Initialization          */
void KeyTask(void);             /* Main keypad read task          */

/*********************************************************************
* Private Resources
*********************************************************************/
static INT8U KeyScan(void);         /* Makes a single keypad scan  */
static INT8U KeyBuffer;             /* Holds the ASCII code for key*/
static const INT8U KeyCodeTable[16] =
   {'1','2','3',DC1,'4','5','6',DC2,'7','8','9',DC3,'*','0','#',DC4};
/*********************************************************************
* GetKey() - A function to provide access to KeyBuffer. This function
*            provides public access to KeyBuffer. It clears KeyBuffer
*            after for read-once handshaking. Note: this handshaking
*            method only works when 0x00 is not a valid keycode. If it
*            is a valid keycode then a semaphore should be added.
*    - Public
*********************************************************************/
INT8U GetKey(void){
    INT8U key;
    key = KeyBuffer;
    KeyBuffer = 0;
    return key;
}

/*********************************************************************
* 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    */
    KeyBuffer = 0x00;           /* Init KeyBuffer      */
}

/*********************************************************************
* 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.
* (Public)
*********************************************************************/
void KeyTask(void) {

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

    DB_PORT |= PP5;

    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;
            KeyBuffer = KeyCodeTable[cur_key - 1]; /*update 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 */

    DB_PORT &= ~PP5;
}

/*********************************************************************
* 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 if 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;
}
/*********************************************************************/
```