

# kb.c

```

1 // Keyboard Driver
2 // Jason Losh
3
4 //-----
5 // Hardware Target
6 //-----
7
8 // 4x4 Keyboard
9 // Column 0-3 outputs on PA6, PA7, PD2, PD3 are connected to cathode of diodes whose anode
   connects to column of keyboard
10 // Rows 0-3 inputs connected to PE1, PE2, PE3, PF1 which are pulled high
11 // To locate a key (r, c), the column c is driven low so the row r reads as low
12
13 //-----
14 // Device includes, defines, and assembler directives
15 //-----
16
17 #include <stdint.h>
18 #include <stdbool.h>
19 #include "tm4c123gh6pm.h"
20 #include "kb.h"
21
22 //-----
23 // Global variables
24 //-----
25
26 #define COL0 (*(volatile uint32_t *) (0x42000000 + (0x400043FC-0x40000000)*32 + 6*4))
27 #define COL1 (*(volatile uint32_t *) (0x42000000 + (0x400043FC-0x40000000)*32 + 7*4))
28 #define COL2 (*(volatile uint32_t *) (0x42000000 + (0x400073FC-0x40000000)*32 + 2*4))
29 #define COL3 (*(volatile uint32_t *) (0x42000000 + (0x400073FC-0x40000000)*32 + 3*4))
30 #define ROW0 (*(volatile uint32_t *) (0x42000000 + (0x400243FC-0x40000000)*32 + 1*4))
31 #define ROW1 (*(volatile uint32_t *) (0x42000000 + (0x400243FC-0x40000000)*32 + 2*4))
32 #define ROW2 (*(volatile uint32_t *) (0x42000000 + (0x400243FC-0x40000000)*32 + 3*4))
33 #define ROW3 (*(volatile uint32_t *) (0x42000000 + (0x400253FC-0x40000000)*32 + 1*4))
34
35 #define KB_BUFFER_LENGTH 16
36 #define KB_NO_KEY -1
37 char keyboardBuffer[KB_BUFFER_LENGTH];
38 bool debounceRequest = false;
39 uint8_t debounceCount = 0;
40 uint8_t keyboardReadIndex = 0;
41 uint8_t keyboardWriteIndex = 0;
42
43 //-----
44 // Subroutines
45 //-----
46
47 // Non-blocking function called to drive a selected column low for readout
48 void setKeyboardColumn(int8_t col)
49 {
50     COL0 = col != 0;
51     COL1 = col != 1;
52     COL2 = col != 2;
53     COL3 = col != 3;
54 }
55

```

```

56// Non-blocking function called to drive all selected column low for readout
57void setKeyboardAllColumns()
58{
59    //COL0 = COL1 = COL2 = COL3 = 0;
60    COL0 = 0;
61    COL1 = 0;
62    COL2 = 0;
63    COL3 = 0;
64}
65
66// Non-blocking function called to determine is a key is pressed in the selected column
67int8_t getKeyboardRow()
68{
69    int8_t row = KB_NO_KEY;
70    if (!ROW0) row = 0;
71    if (!ROW1) row = 1;
72    if (!ROW2) row = 2;
73    if (!ROW3) row = 3;
74    return row;
75}
76
77// Non-blocking function called by the keyboard ISR to determine if a key is pressed
78int8_t getKeyboardScanCode()
79{
80    uint8_t col = 0;
81    int8_t row;
82    int8_t code = KB_NO_KEY;
83    bool found = false;
84    while (!found && (col < 4))
85    {
86        setKeyboardColumn(col);
87        waitMicrosecond(1);
88        row = getKeyboardRow();
89        found = row != KB_NO_KEY;
90        if (found)
91            code = row << 2 | col;
92        else
93            col++;
94    }
95    return code;
96}
97
98// 5ms keyboard timer interrupt used for key detection and debouncing
99void keyboardIsr()
100{
101    bool full;
102    int8_t code;
103    // Handle key press
104    if (!debounceRequest)
105    {
106        code = getKeyboardScanCode();
107        if (code != KB_NO_KEY)
108        {
109            full = ((keyboardWriteIndex+1) % KB_BUFFER_LENGTH) == keyboardReadIndex;
110            if (!full)
111                {

```

kb.c

```
112         keyboardBuffer[keyboardWriteIndex] = code;
113         keyboardWriteIndex = (keyboardWriteIndex + 1) % KB_BUFFER_LENGTH;
114     }
115     debounceRequest = true;
116 }
117 }
118 // Handle debounce
119 else
120 {
121     setKeyboardAllColumns();
122     waitMicrosecond(1);
123     if (getKeyboardRow() != KB_NO_KEY)
124         debounceCount = 0;
125     else
126     {
127         debounceCount++;
128         if (debounceCount == 10)
129         {
130             debounceCount = 0;
131             debounceRequest = false;
132         }
133     }
134 }
135 TIMER1_ICR_R = TIMER_ICR_TATOCINT;
136 }
137
138 // Non-blocking function called by the user to determine if a key is present in the buffer
139 bool kbhit()
140 {
141     return (keyboardReadIndex != keyboardWriteIndex);
142 }
143
144 // Blocking function called by the user to get a keyboard character
145 char getKey()
146 {
147     const char keyCap[17] = {"123A456B789C*O#D"};
148     while (!kbhit());
149     uint8_t code = keyboardBuffer[keyboardReadIndex];
150     keyboardReadIndex = (keyboardReadIndex + 1) % KB_BUFFER_LENGTH;
151     return (char)keyCap[code];
152 }
153
154
```