Abstract

In this lab, you will write software that will read the keyboard on the HP 20b and display which key is pressed.

1 The Matrix Keyboard

Figure 3 shows the schematic for the HP 20b’s keyboard, which is a standard matrix type consisting of row and column wires that can be shorted together by the keys. It is connected to pins on the SAM7L chip that can be driven by a parallel I/O controller: a peripheral that enables software to control and read the state of each pin. Figure 1 is a more concise table that lists the pair of pins that each key shorts together when pressed.

The lab2.zip file (on the course website) adds keyboard.h and keyboard.c to the files from lab1.zip. Figure 2 lists the keyboard-related functions defined in keyboard.c.

To set up the peripherals to read the keyboard, call keyboard_init. This turns pins PC0–PC6 (the “columns”) into outputs and pins PC11–PC15, and PC26 into inputs with pull-up resistors. The pull-up resistors ensure that if no key along a row is shorted the pin will have the value 1.

To see if a particular key is pressed, set the key’s column to 0, every other column to 1, and read the key’s row. If the key is pressed, the key’s row will return 0, otherwise it will return 1. The easiest way to do this is to call the helper functions in keyboard.h.

main.c contains a program that reads the topmost keys on the keyboard and reports when they are pressed.

2 What To Do

• Write a function keyboard_key that returns a code that indicates either which key is currently being pressed or that no key is being pressed. Add this function to keyboard.c.

• Modify the code in main.c to use this function to report which key is being pressed.
<table>
<thead>
<tr>
<th>“columns”</th>
<th>PC0</th>
<th>PC1</th>
<th>PC2</th>
<th>PC3</th>
<th>PC4</th>
<th>PC5</th>
<th>PC6</th>
</tr>
</thead>
<tbody>
<tr>
<td>“rows”</td>
<td>PC11</td>
<td>PC12</td>
<td>PC13</td>
<td>PC14</td>
<td>PC15</td>
<td>PC16</td>
<td></td>
</tr>
<tr>
<td>N</td>
<td>I/YR</td>
<td>PV</td>
<td>PMT</td>
<td>Bond</td>
<td>FG</td>
<td>Amort</td>
<td></td>
</tr>
<tr>
<td>CshFl</td>
<td>IRR</td>
<td>NPV</td>
<td>Bond</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>INPUT</td>
<td></td>
<td></td>
<td></td>
<td>+/-</td>
<td>←</td>
<td></td>
<td></td>
</tr>
<tr>
<td>▲</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>⇑</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>▼</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>⇑</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>shift</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>¬</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>.</td>
<td>=</td>
<td>+</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 1: The HP 20b’s keyboard layout. When pressed, each key shorts two pins: one for its column, one for its row.

```c
// Initialize the keyboard and set all columns high
// with pullups on the rows
extern void keyboard_init(void);

// Set the given column high
extern void keyboard_column_high(int column);

// Set the given column low
extern void keyboard_column_low(int column);

// Return true if the row is high, false otherwise
extern int keyboard_row_read(int row);
```

Figure 2: Keyboard helper function declarations from keyboard.h
Figure 3: Schematics for the HP 20b keyboard. The top is the keyboard matrix itself; the bottom shows how the matrix is connected to the SAM7L chip. Note that the ON/CE key is separate (not part of the matrix) and that “columns” of keys are actually horizontal.