## Keypad/LCD reference

## #include

The .h folder on the GDrive contains 3 files: bit.h, keypad.h, lcd.h that work as interfaces to the Keypad and LCD. You need to change the pin definitions at the top of these files based on how you wire up your board. Also you need to make sure you set up the DDR properly to use these pins.

These files will all work independently of the task scheduler. We assume these function calls are non-finite near-zero time tasks, so it is appropriate to use them with the task scheduler.

## Keypad

A keypad is comprised of several buttons. If each button had its own pin, the keypad below would require 16 pins.

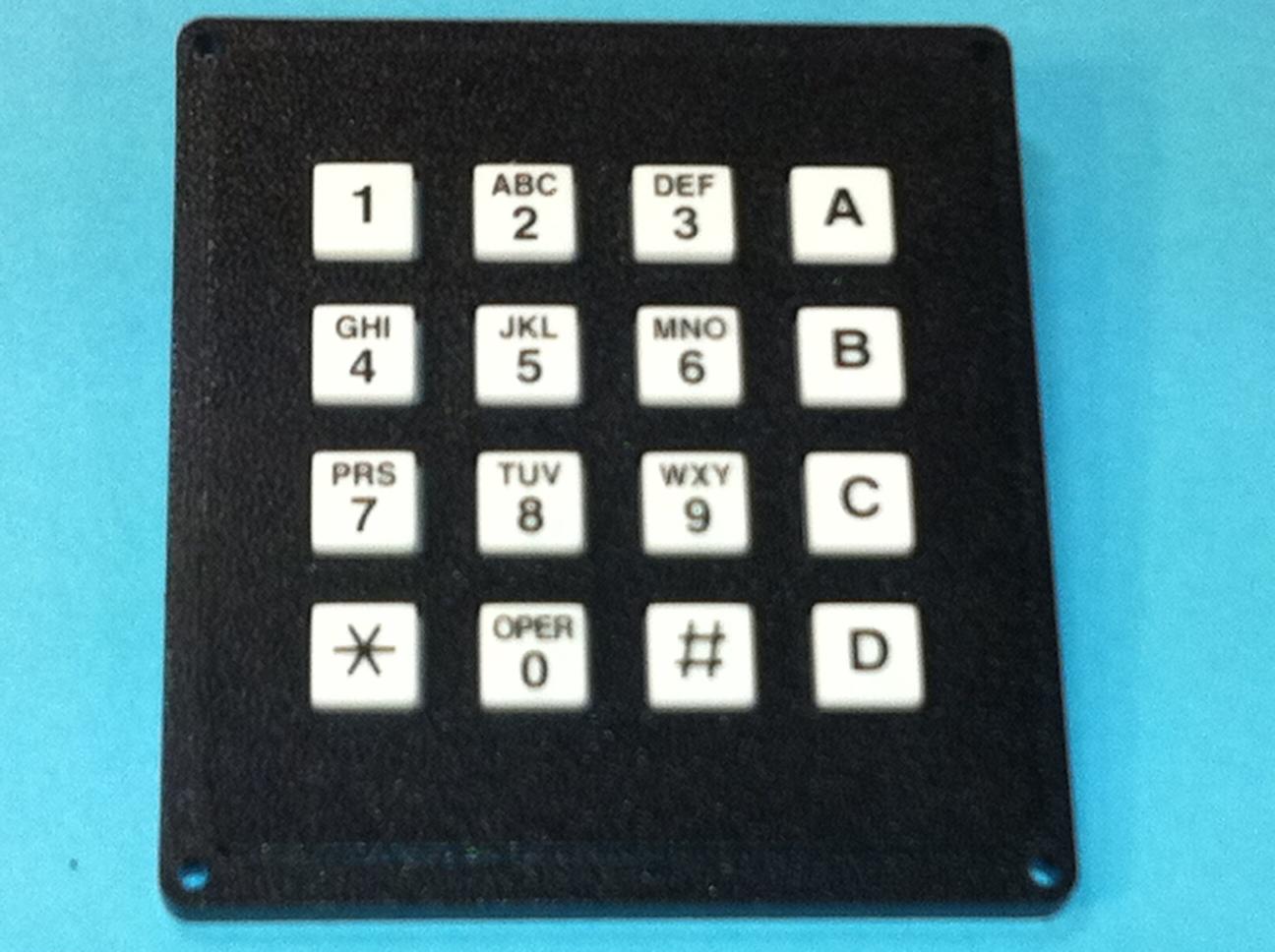


Figure 1: Keypad GH5004-ND

To reduce pin count, keypads commonly have a row/column arrangement as shown below.

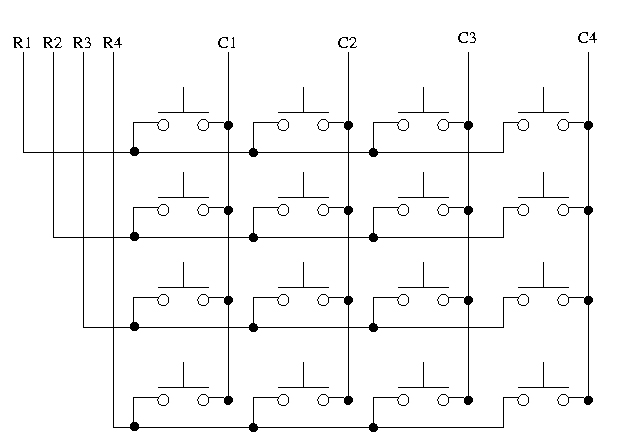


Figure 2: High-Level Connection diagram for Keypad

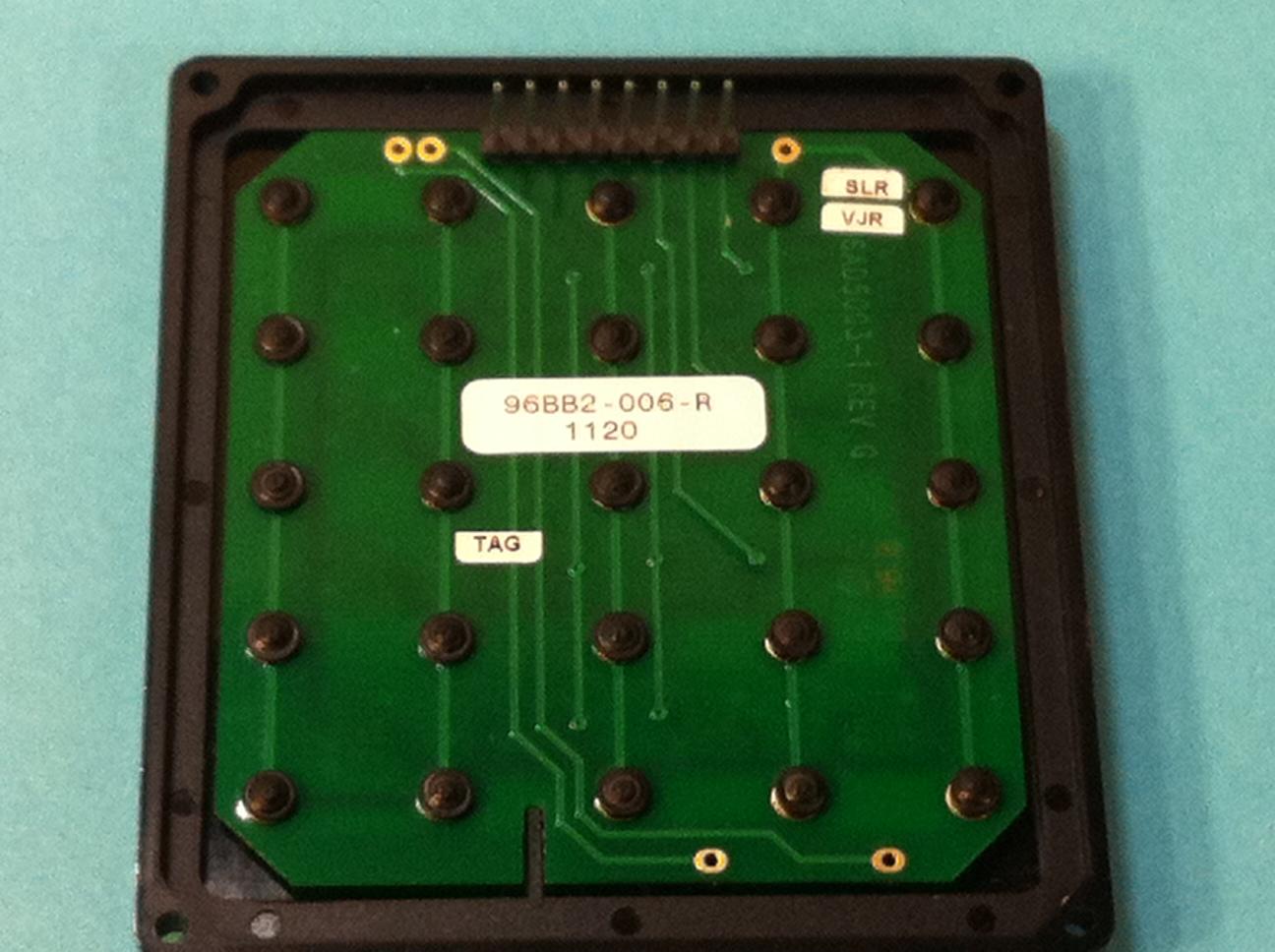


Figure 3: Keypad GH5004-ND Pins - C4C3C2C1 R4R3R2R1 (left to right in figure)

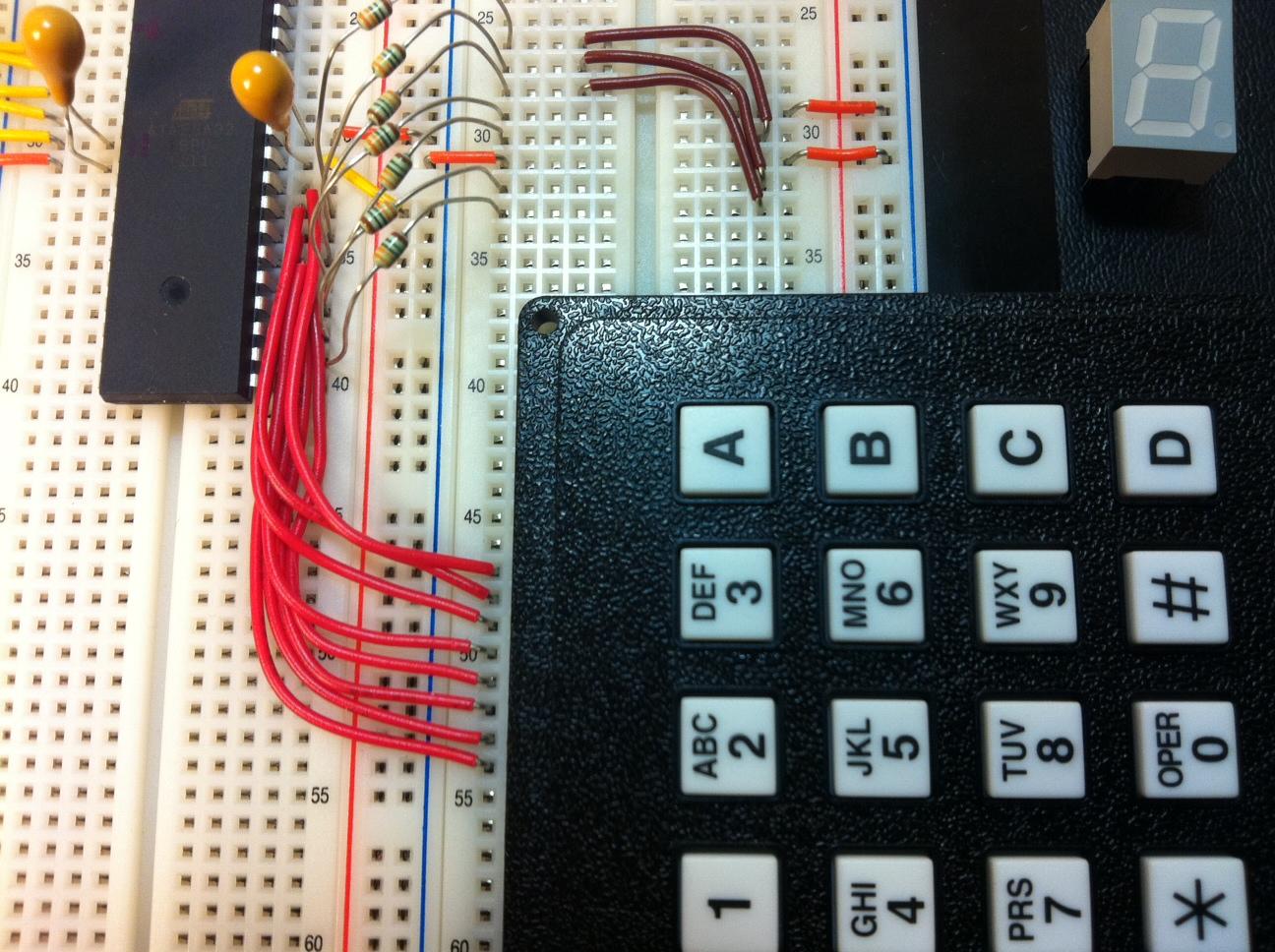
Each row has a pin (R1-R4), and each column has a pin (C1-C4), for a total of 8 pins. Pressing a button uniquely connects one column pin with one row pin. For example, pressing the upper-left button connects pin C1 with pin R1. Pressing the bottom-right button connects C4 and R4.

To accomplish accepting input from 16 buttons with only 8 pins a technique known as time multiplexing is employed. The idea is simple, we shall use common row wires and common column wires the achieve our lower pin count. This however causes a problem, by sharing the rows and columns we have cross talk. To overcome this, we will selectively enable one column at a time, check the 4 pins connected to that row, and then continue by enabling the next column, repeating the process for all columns. This is time multiplexing -- simultaneous transmission of several messages along a single channel of communication by having those signals transmit at specific times (in this case a specific sequence).

We can get away with this because the microcontroller can operate much faster than humans can react/perceive. In the time it takes a person to press one of the buttons, the microcontroller can make many passes of the keypad to check for input. Thus the process is transparent to the user.

Below is an example setup for the keypad:

In this case we connect the keypad to port C as shown (R1 connected to PC0, …, C4 to PC7).



Keypad Connections

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Keypad  Pin # | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| Term | R1 | R2 | R3 | R4 | C1 | C2 | C3 | C4 |
| AVR Port | C0  Output | C1  Output | C2  Output | C3  Output | C4  Input | C5  Input | C6  Input | C7  Input |

***Note****: Don’t forget to uncheck the JTAG fuse if you are using port C.*

**LCD Display:**

LCD Display Pin Connections

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **LCD**  **PIN #** | **1** | **2** | **3** | **4** | **5** | **6** | **7-14** | **15-16** |
| Connection | GND | 5 Volts | Resistor (10KΩ)  thru. to  GND | AVR PORT  Reset (RS) | GND | AVR PORT  Enable (E) | AVR PORT  D0-D7 | Un- used |

### 

Make sure to change the values of DATA\_BUS, CONTROL\_BUS, RS, and E in lcd.h to reflect the connections of the LCD screen. Also set the DDR's appropriately, in this case all will be output.

In the example below, for simplicity we used all the pins of port B as the data lines for the LCD, and then we used the first two pins or port D PD0 and PD1.

Notice that we used multiple 10KΩ resistors in parallel for pin #3, this adjusts the contrast of your display. You may need to vary this depending on how steady your power supply is.

***Be careful not to use too low of a resistance as you may burn out your LCD.***

