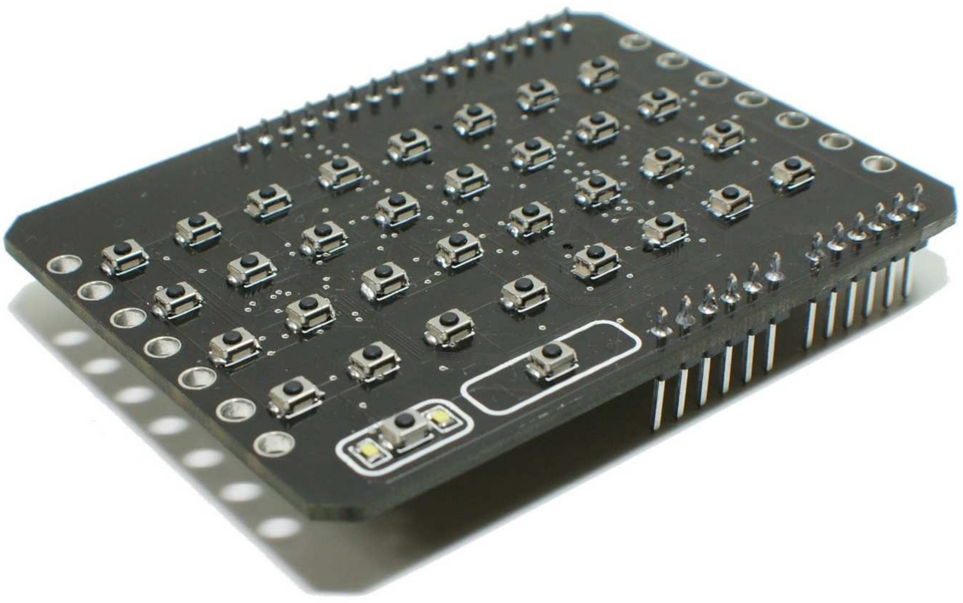
liquidware
INVENT, CREATE & BUILD

ButtonShield



Typical Applications:

- Keyboard Simulation
- Physical Computing
- Display Input Control

Product Features:

- 30 General Purpose Buttons
- 2 Steady State Buttons
- 1 Lock Switch
- 1 Steady State Space Switch
- 2 Operating Modes
- Back Light Control

Contents

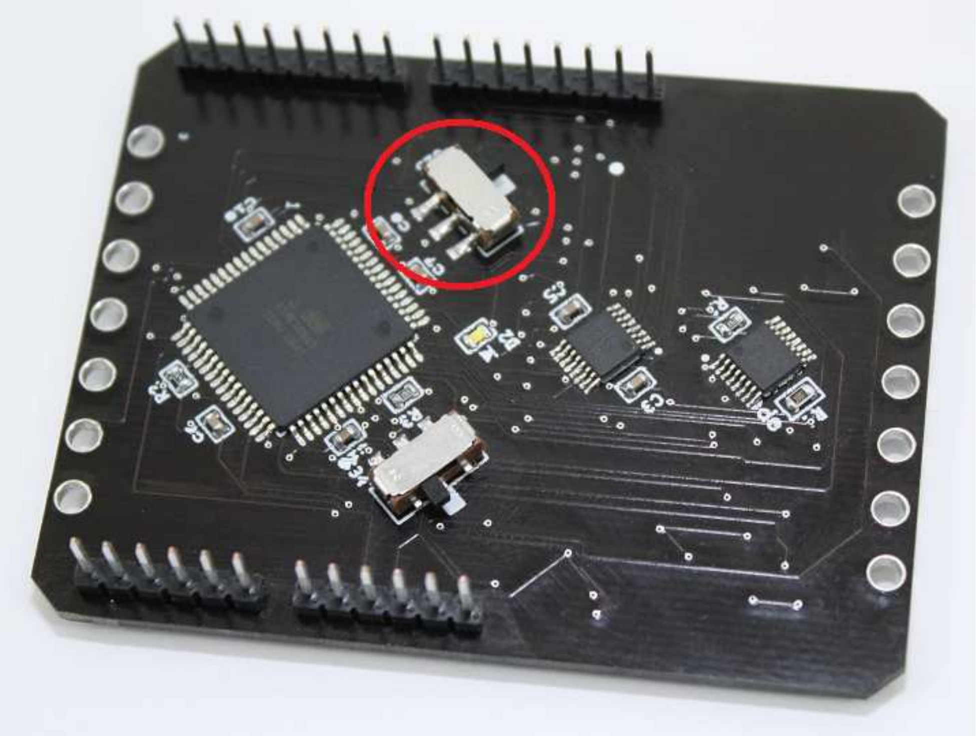
1. Functional Description

The ButtonShield provides the Arduino access to 30 buttons, 3 steady state general purpose buttons, and a user-controllable back light LED.

An A/B Mode selection switch changes the Arduino occupied pins. This mode selection allows two ButtonShields to be mounted on a single Arduino via an ExtenderShield, preventing signal collision on the same pin.

2. Pin Description

The ButtonShield has two operating modes: "Mode A" and "Mode B" (see table in Section 2, "Pin Description"). These Modes are controlled by a switch located on the bottom of the shield, identified by the red circle in the figure below.



The ButtonShield relays button information along 6 different pins. Each button in a given mode is represented by a 6-digit binary value, with each digit transmitted along a single pin. A 6-digit binary value allows for a total of 64 different values to be passed to the Arduino in this fashion.

3. Button Mapping

There are a total of 64 different numbers the ButtonShield can send to the Arduino. The button in the bottom left corner is the Lock button. When this button is pressed, it acts similarly to a “Caps Lock” key on a key board.

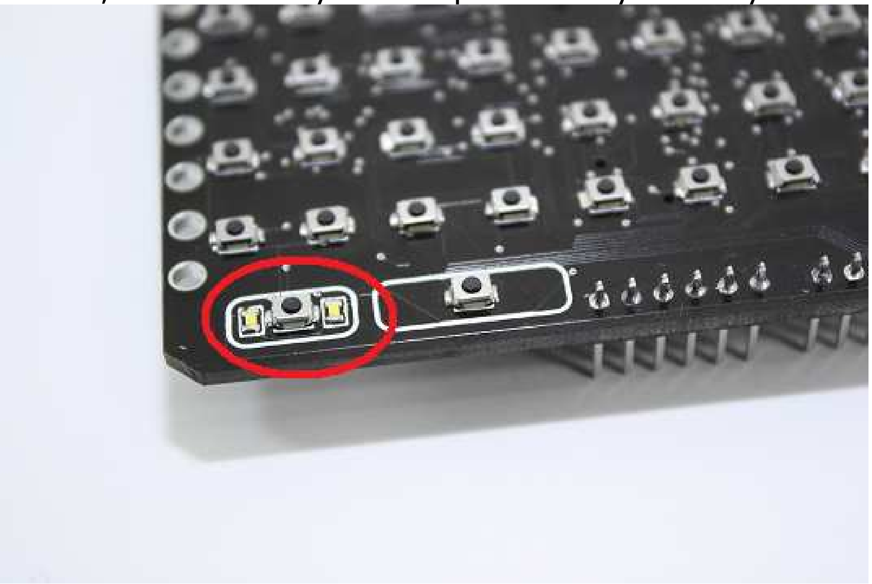
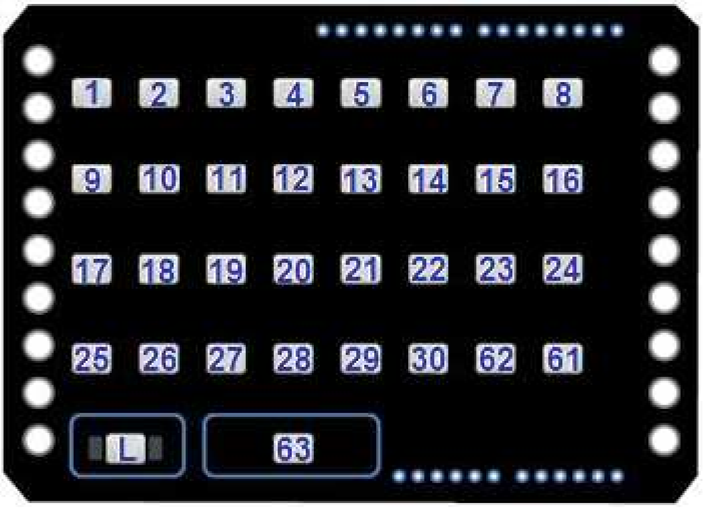
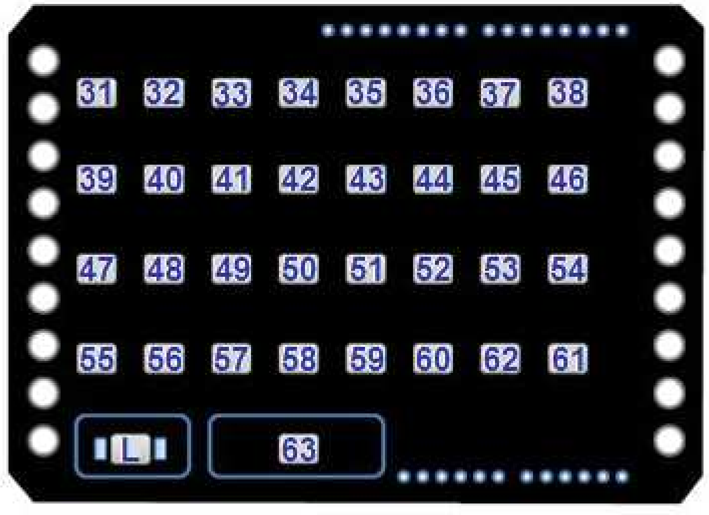


Figure 1
The Lock Button (circled in red)

When the Lock button (“L”) isn’t pressed, the ButtonShield is in normal mode. The two LEDs next to the Lock button will be off when the ButtonShield isn’t in Lock mode. The diagram on the left illustrates the number sent to the Arduino when the button is pressed.



If the Lock ("L") button is pressed, and the Lock LEDs are illuminated, the ButtonShield is in Lock mode. In Lock mode, most buttons are shifted up by 30. The buttons that are excluded are 61, 62, and 63; these buttons will send the same number to the Arduino regardless of whether Lock mode is on. The diagram on the left illustrates the numbers sent to the Arduino when Lock mode is on, and the respective button is pressed.



31	32	33	34	35	36	37	38
39	40	41	42	43	44	45	46
47	48	49	50	51	52	53	54
55	56	57	58	59	60	62	61
L	63						

Figure 2
Lock Mode On

4. Back Light Control

An LED is attached to the bottom of the Shield. The LED will turn on, when the switch on the bottom is flipped.

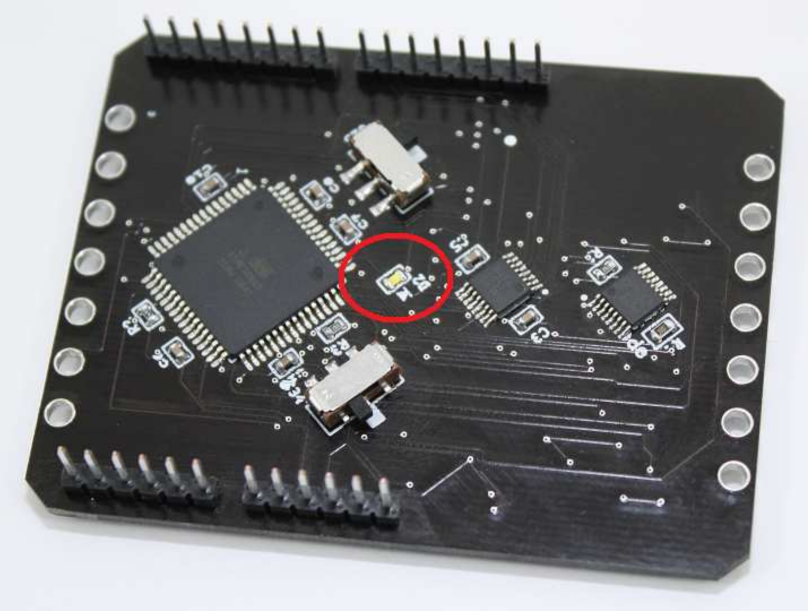


Figure 3
Back Light LED

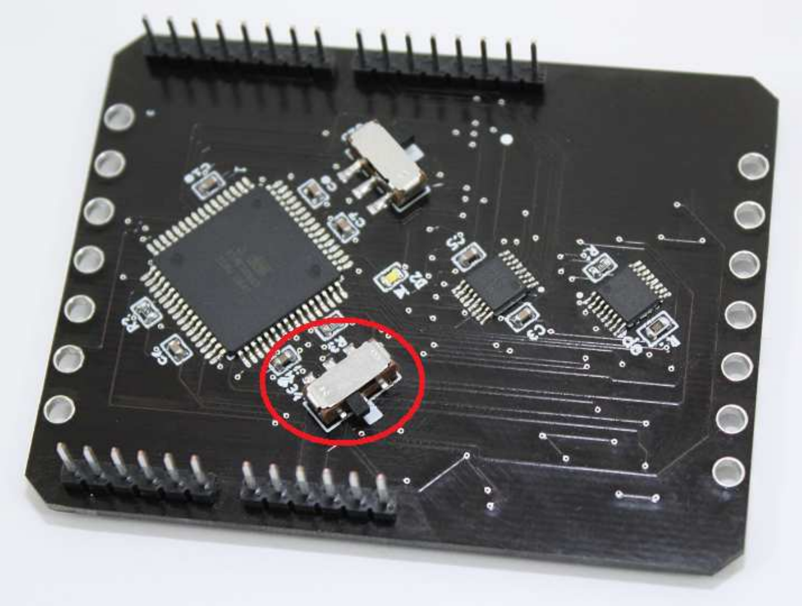


Figure 4
Back Light Control Switch

5. Arduino Test Code

The ButtonShield Library is available in the Aardvark IDE:
Mac OS X: http://www.liquidware.com/apps/show/49
Windows: http://www.liquidware.com/apps/show/43

```
//Uploaded this to an Arduino and run the serial monitor.

#include < ButtonShield.h >

ButtonShield buttonsA = ButtonShield(0); /* Created a new ButtonShield
on Mode A */
ButtonShield buttonsB = ButtonShield(1); /* Created a new ButtonShield
on Mode A */

void setup()
{
Serial.begin(9600);
}

void loop()
{
Serial.print("ModeA: ");
Serial.print(buttonsA.readButtons());
Serial.print(" ModeB: ");
Serial.print(buttonsB.readButtons());
}
```

The Arduino Library code is opensource and shared on github:
www.github.com
liquidware / **antipasto_arduino** / hardware / libraries / ButtonShield

6. Firmware Theory

The ButtonShield is powered by an ATmega645 processor.

If you describe the Buttons as a big array, then you can easily scan through the entire list. Here is snippet of the ButtonShield 'core' code.

```
/*=====================================================================
=========
 * CONSTANTS

*======================================================================
======*/

/* The user pin lookup table.
   This table provides a mapping to the ButtonShield hardware pins. */
```

```
PIN_DESC_T const pinTable[] = {

    /* 0 */  { &PORTA, 0, &PINA, &DDRA }, //not a button

    /* 1 */  { &PORTA, 5, &PINA, &DDRA }, //the top left button
    /* 2 */  { &PORTA, 6, &PINA, &DDRA },
    /* 3 */  { &PORTE, 3, &PINE, &DDRE },
    /* 4 */  { &PORTE, 4, &PINE, &DDRE },
    /* 5 */  { &PORTD, 3, &PIND, &DDRD },
    /* 6 */  { &PORTB, 5, &PINB, &DDRB },
    /* 7 */  { &PORTD, 7, &PIND, &DDRD },
    /* 8 */  { &PORTD, 0, &PIND, &DDRD },

    /* 9 */  { &PORTA, 4, &PINA, &DDRA },
    /* 10 */ { &PORTC, 7, &PINC, &DDRC },
    /* 11 */ { &PORTE, 2, &PINE, &DDRE },
    /* 12 */ { &PORTE, 7, &PINE, &DDRE },
    /* 13 */ { &PORTD, 6, &PIND, &DDRD },
...
};
```

Now with your table, you can easily index through a huge list of Buttons and set them all to inputs in a few lines of code, for example like this:

```
for(pin=1;pin<36;pin++) // loop that sets pins 1 to 36 as inputs
    {
        pinMode(pin, INPUT);  //sets the pin as an input pin on the
buttonshield
        digitalWrite(pin, HIGH); //sets the input to be high by default
(enable internal pull up resistor)
    }
```

To poll the Button inputs again, scan through the entire PinTable[] in a for() loop, ditching out when you've found a pressed button.

That's your *foundPin*

Using the *foundPin* id, the ButtonShield directly writes to 6 Arduino input pins, or 'bits' if you'd like to think about it that way. Like this:

```
/* Write the button value onto the Arduino pins */
    for(i=0;i<6;i++)
               {
               digitalWrite((i+kArduinoPinsOffset), foundPin&1);
               foundPin =  foundPin>>1;
        }
```