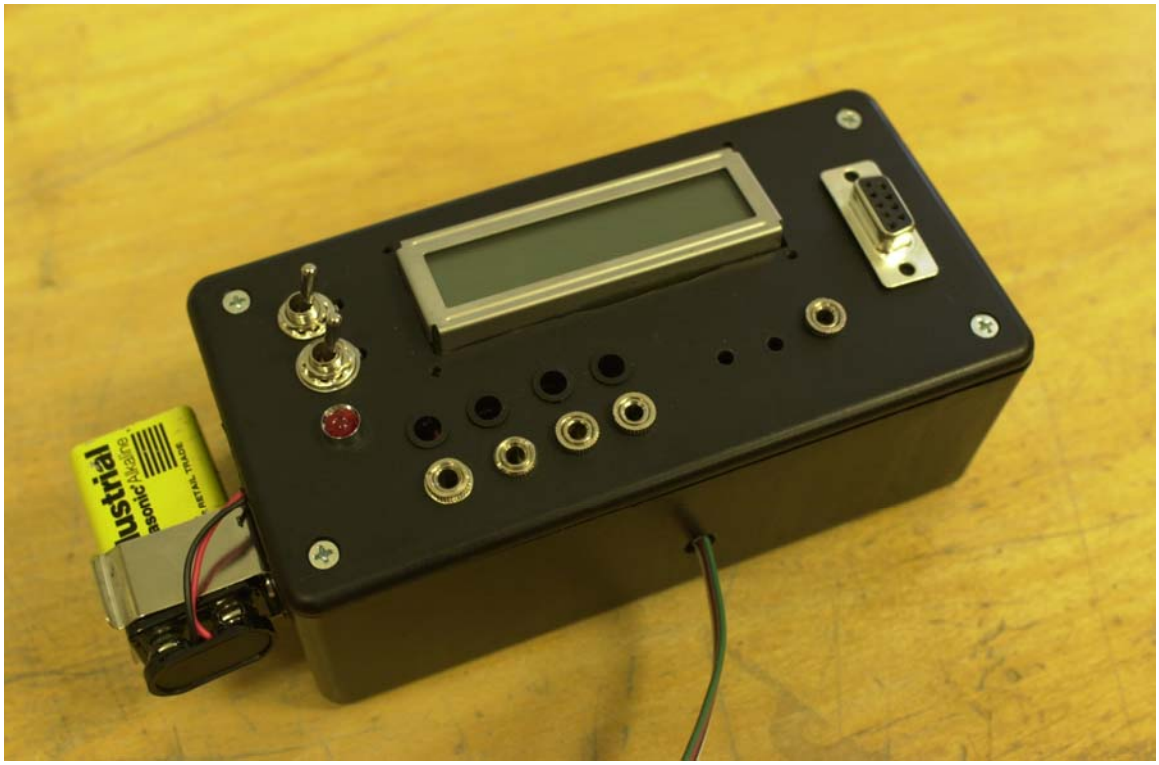


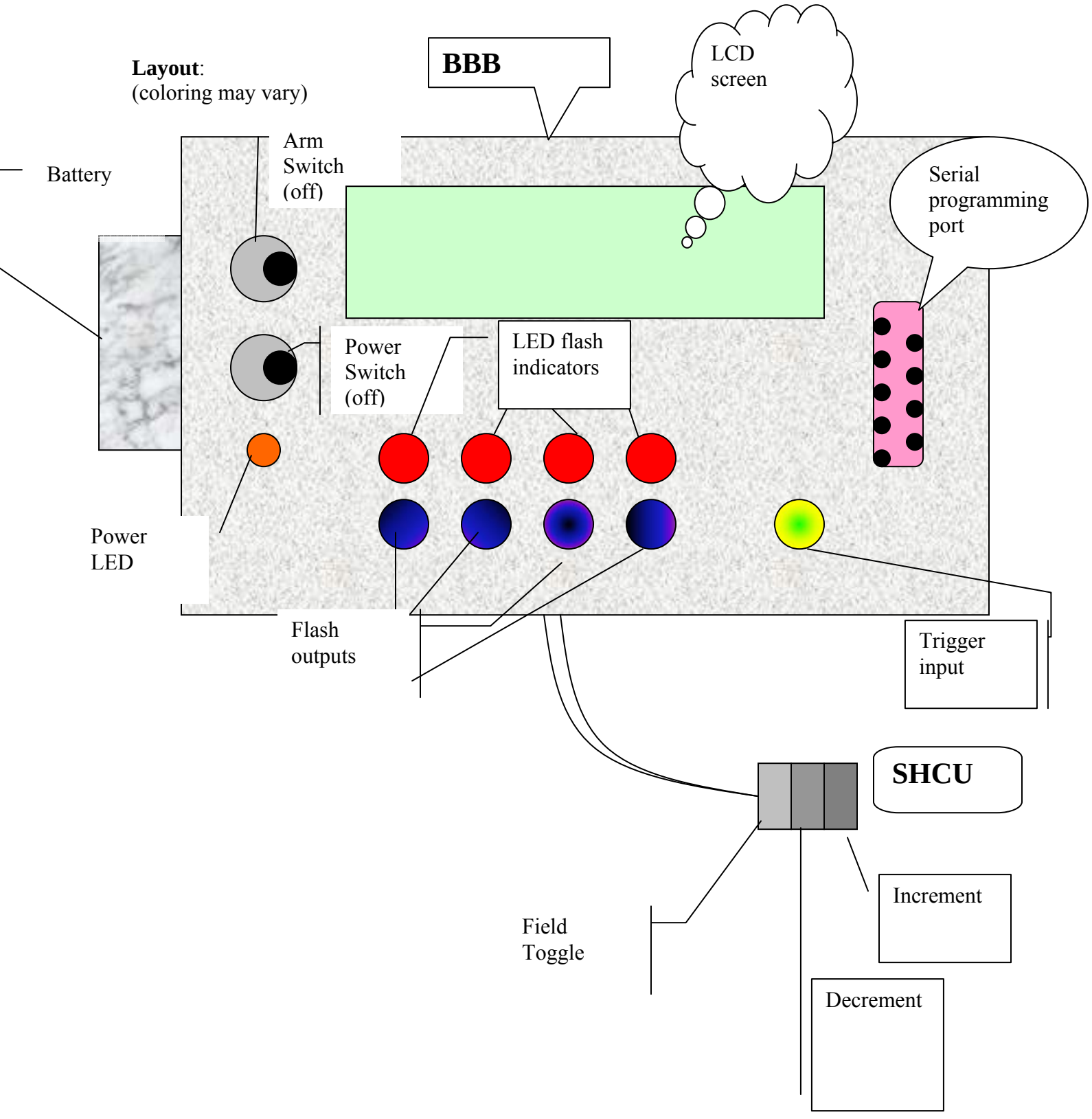
North Carolina School of Science and Math Portable High Speed Photography Control Device (NCSSMPHSPCD)

Masterfully designed and built by Alex Hornstein ’03.

Description

The NCSSMPHSPCD is a microcontroller-based user portable user interface that controls the timing and triggering of up to 4 flashes. It was designed with potential upgrades in mind, so it contains a serial port interface which allows reprogramming of the EEPROM microcontroller. The NCSSMPHSPCD uses three control switches in the Small Handheld Control Unit (SHCU) which allow the user the set the initial delay between the trigger and the flash, and also the delays between flashes. The default program had accuracy of 100 ms, but there is pre-written software allowing accuracy of 10 microseconds. There is a toggle switch on the Big Black Box (BBB) which “arms” the device, setting it in watch mode for a signal from the trigger. The device is powered by a nine volt battery.

Layout:
(coloring may vary)



Note: A standard serial cable doesn't work. Instead, use Alex's special 4-pin cable with white pin farthest left.

Specifications:

Power: 9V battery
Dimensions: About a foot by 4 inches, give or take

16X2 serial LCD display
Basic Micro Atom24 microcontroller
All plugs must be 1/8" mono

Instructions for basic use of the NCSSMPHSPCD

- Make sure the battery is plugged in. Otherwise, chances are it won't work.
- Plug the flash units(up to 4) into the flash output jacks.

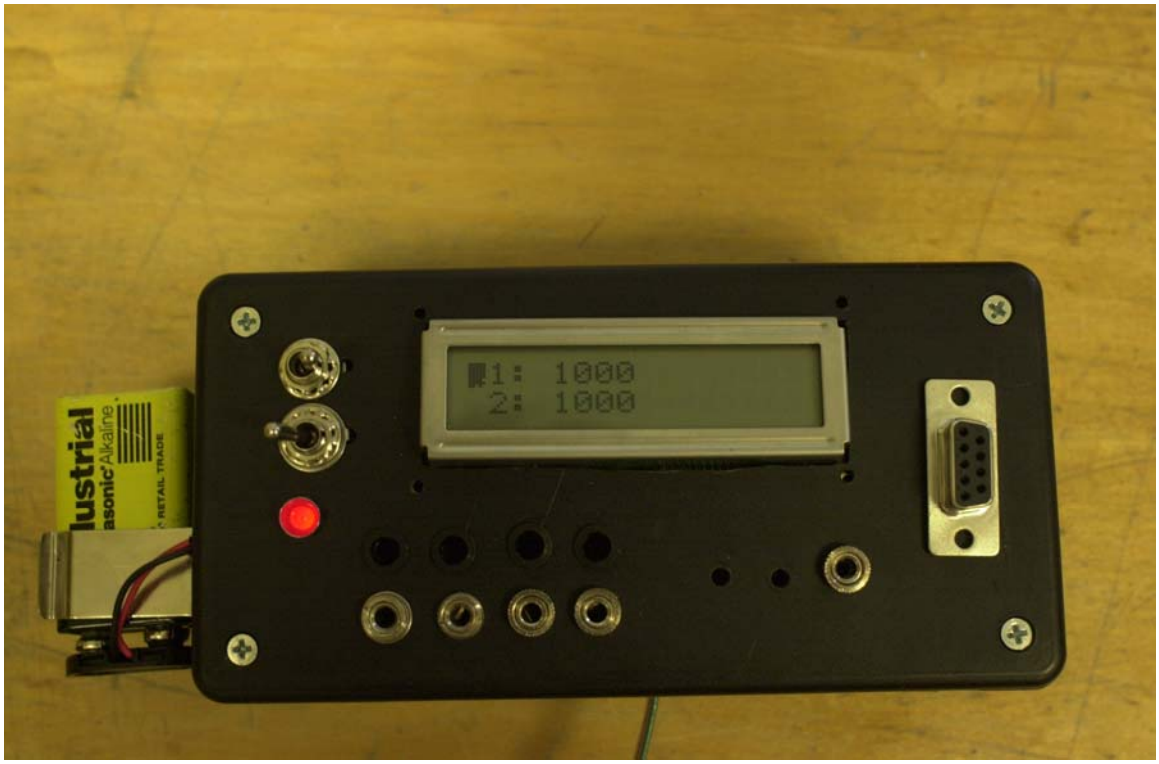
IMPORTANT The device simply connects the two wires in the flash. It does not generate a voltage. Make sure you are using the right kind of flashes!

- Plug the trigger you are using into the trigger input jack.

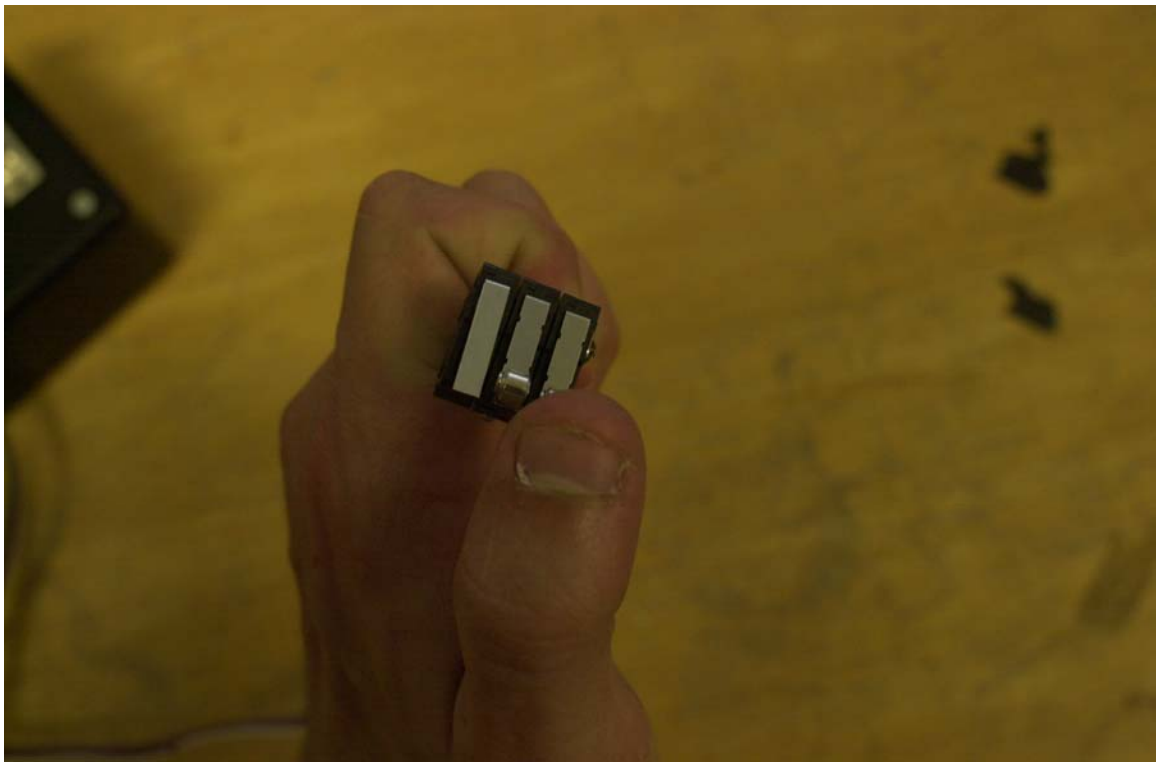
ALSO IMPORTANT The trigger must be the type that simply connects its terminals, NOT the kind that generates a voltage. This is extremely important to get right, because the trigger input is not buffered: it connects directly to the microcontroller, and if you plug a voltage-generating trigger into it, you could burn out the microcontroller, and then you would feel really bad about yourself.

- Turn the power switch (the lower toggle switch) to ON which is the left position. The LCD screen should flicker but then remain blank. This is normal.
- Press any button on the SHCU. You should see something like this on the screen:

  ▮1: 1000
  2: 1000



- The ▉ character indicates which delay you are modifying. Delay 1 is the delay between the trigger and the first flash, and delay 2 is the time between flashes.
- Pressing the leftmost button briefly toggles the ▉ character, allowing you to modify the other delay. Pressing it again briefly toggles it back. Holding it down makes it toggle continuously, which effectively wastes your time.



- The rightmost button is the increment button. With the basic software, it will increase the delay being modified by 100 milliseconds. The leftmost button is the decrement button. It will decrease the delay being modified by 100 milliseconds.
- Using the SHCU, set your delays to appropriate values. Then arm the device by mowing the top toggle switch on the BBB to the left position. The screen should display 3,2,1, GO! At this point, the device will initiate the flash sequence upon being triggered.
- Once the flashes are triggered, the device should be turned off. To reset it, turn it off and on again. In the current software, the delays are not saved—they must be set every time.

That's the basics. The rest of this manual includes details such as the basic code, instructions on how to reprogram the device, the circuit diagram, and details on the electrical nuances of the device.

About programming:

1. Start the program. Have cable connected to computer and box on.
2. Change time increments 1 and 2 to desired values.
3. Click program button.
4. When finished, turn box off, then on. This saves the new values.

Have new version of code to replace this. See orange disk.

CODE

The code is written in Mbasic, which has a compiler and manual that can be downloaded for free from www.basicmicro.com

```
'*************************************************************************
*****
'*high speed photo program v1.01 designed for                          *
'*Basic Micro 28 pin module(www.basicmicro.com)                          *
'*And a serial LCD(www.imagesco.com)                                    *
'*Programmed 1-15-03 by Alex Hornstein, boy genius and egomaniac extraordinaire   *
'*                                                *
'*Everyone loves a slinky                                    *
'*************************************************************************
*****
'variable declarations pins 10-15 are unused and can be outputs
high 6
high 7
high 8
high 9
myPause var byte
myPause=10     'the gimp LCD wants a pause after every statement
'the work variables are used by the atom's button statement.  Just initiate them to 0
work var byte
work=0
work2 var byte
work2=0
work3 var byte
work3=0
work4 var byte
work4=0
char1 var byte
char2 var byte
char1=0
char2=32
num var word   'delay 1
num=1000
num2 var word  'delay 2
num2=1000
```

```
delay var nib  'which delay we are working on
delay=0

'*************************************************************************
*******
'                          some serial LCD codes:
'data can be sent at 2400 or 9600 baud, normal with (I think) no parity
'instructions are preceeded by 254
'[254,1] clears the display
'[254,192] locates the cursor at the lower line
'text is printed in quotes  ex:  ["hello, I idolize alex hornstein.  I wish I were him"]
'use the serout command as follows: serout <pin>,<mode of communication>,<data to be sent>
'The LCD is currently on pin 5, and mode of communication is n2400
'*************************************************************************
*******
input p1
input p2
input p3
input p4
goto display
'this is the main loop.  The buttons statements watch the buttons, which are all in state 1 (See
atom documentation)
loop:
        button 1,1,20,2,work,1,add
        button 2,1,20,2,work2,1,subtract
        button 3,1,20,2,work3,1,delay0
        button 4,1,20,2,work4,1,run
goto loop

display:
'displays the current delays and the box thingy
        serout 5,n2400,[254,1]
                'top left, 1st row
                pause myPause
                serout 5,n2400,[254,128]
                pause myPause
                serout 5,n2400,[char1, "1: ", DEC num]
                pause myPause
                '1st position, 2nd row
                serout 5,n2400,[254,192]
                pause myPause
                serout 5,n2400,[char2, "2: ", DEC num2]
                pause myPause
goto loop

add:
        if delay=0 then
                num=num+100
        elseif delay=1
                num2=num2+100
```

```
        endif
goto display

subtract:
        if delay=0 then
                num=num-100
        elseif delay=1
                num2=num2-100
        endif
goto display


delay0:
'figure out which delay we are editing, and move the little triangle around accordingly
        if delay=1 then
                delay=0
                char1=0
                char2=32
        elseif delay=0
                delay=1
                char1=32
                char2=0
        endif
goto display


'this just gives a countdown and then enables the interrupt
run:
OnInterrupt ExtInt,flash
setExtInt EXT_L2H  'sets the external interrupt on pin 0 when the pin is pulled from low to high
enable ExtInt      'enables the interrupt
serout 5,n2400,[254,1]
                pause myPause
serout 5,n2400,["3"]
pause 1000
serout 5,n2400,[254,1]
                pause myPause
serout 5,n2400,["2"]
pause 1000
serout 5,n2400,[254,1]
                pause myPause
serout 5,n2400,["1"]
pause 1000
serout 5,n2400,[254,1]
                pause myPause
serout 5,n2400,["GO!"]
'loop until the interrupt is triggered
wait:
goto wait
```

```
disable   'disables the interrupt from here on down-we don't want it interrupting the flash sequence
flash:
'go ahead and bring the flash outputs low in sequence, from pin 6-15
pauseus num
low 6
pauseus num2
low 7
pauseus num2
low 8
pauseus num2
low 9
```

Additional notes

The atom has 4 analog inputs which require direct soldering to the chip. They are 4 solder pads located on the bottom of the atom. In the event analog inputs are required, the atom provides fairly simple A-D conversion functions (see the manual)

Pins P10-P15 (an additional 6 pins on the atom) are free, which can allow it to drive 6 more flashes if additional optoisolators are wired.

By using 1 more pin, it is possible to replace the 3-button system with a keypad. This replacement would require restructuring of the code. If you need help with it, look me up. I'll be at MIT until 2007.

If you ever reprogram the atom, it needs to be turned on and connected to the computer before you compile the code. Otherwise you get an error.

The LCD requires a significant pause when writing to it. Otherwise, you get a bitshift in your serial line and everything goes screwy. Just be sure to pause between writing instructions. Also, pause for 1 second when you boot up, so that it can initialize.

That's about it. Hope you enjoy using the NCSSMPHSPCD!