Build a Simple Toaster Oven Temperature Profile Controller (Or Anything Else You Want!)

by Rich Testardi (rtestardi@live.com)

Introduction

Using a highly-integrated microcontroller running "StickOS® BASIC", it is possible to quickly build a toaster oven temperature profile controller for performing surface mount (SMT) printed circuit board reflow soldering at home. It is also possible to program a large variety of other general-purpose embedded system projects with minimal software effort, *using only a web-page terminal emulator and high-level BASIC algorithmic statements to manipulate the microcontroller (MCU) pins and peripherals*.

StickOS BASIC is an *entirely MCU-resident interactive programming environment*, which includes an easy-to-use editor, transparent line-by-line compiler, interactive debugger, performance profiler, and flash filesystem, all controlled thru an interactive command-line user interface (see Figure 1). In StickOS, external MCU pins may be mapped to BASIC "pin variables" for manipulation or examination, and internal MCU peripherals may be managed by BASIC control statements and interrupt handlers. A StickOS-capable MCU may be connected to a host computer via a variety of transports *and may then be controlled by web-page terminal emulator*, with no additional software or hardware required on the host computer. Once program development is complete, the MCU may be disconnected from the host computer and configured to autorun its resident BASIC program autonomously.

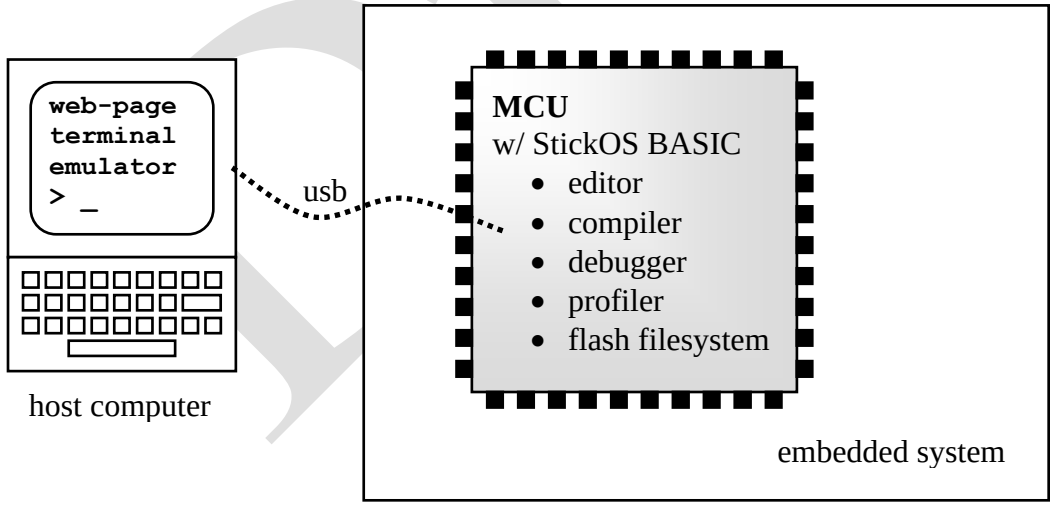


Figure 1

StickOS runs on a wide variety of highly-integrated MCUs with 128k or more of flash memory and 8k or more of RAM, and seamlessly manages the MCU pins and peripherals, including BASIC program support for:

- digital input/output
- analog to digital converters
- pulse width modulators (for analog output or servo motor control)
- output compare timers (i.e., frequency generators)
- UARTs and I2C/SPI master serial interfaces
- periodic timers
- flash memory (i.e., self-programming)

By its very nature, StickOS supports in-circuit emulation -- all you need is a transport connecting the MCU to a host computer, and you have full interactive control over the target embedded system, just as if you were using an in-circuit emulator.

With StickOS, it is no longer necessary to install a software development environment on the host computer, tablet, or phone; likewise, it is no longer necessary to connect any kind of flash programmer or debug hardware to the host computer. More importantly, though, with StickOS it is no longer necessary to study a 500+ page MCU Reference Manual in order to use the MCU external pins and internal peripherals -- StickOS manages them all for you.

The bottom line is that using only a web-page terminal emulator (such as https://rtestardi.github.io/usbte/stickos-basic.html) connected to an MCU running StickOS BASIC, the user can easily edit a BASIC program and interactively debug it using breakpoints, assertions, watchpoints, single-stepping, execution tracing, live variable and pin examination and manipulation, edit-and-continue, etc. The user can then save the BASIC program to the internal flash filesystem, and finally set the BASIC program to autorun autonomously when the MCU powers-up.

Hello World!

So what does the "Hello world!" program look like in StickOS BASIC? Well, if the baseline goal for an embedded system is to configure an I/O pin and get an LED to blink, such as the green LED on pin "e2" of the Flea-Scope™ board, then the "Hello world!" program looks like this (entered text is in **bold**):

```
> 10 dim led as pin e2 for digital output
> 20 while 1 do
> 30   let led = !led
> 40   sleep 500 ms
> 50 endwhile
> run
<Ctrl-C>
STOP at line 40!
> _
```

Line 10 declares a “pin variable” named “led”, then configures the general purpose I/O pin “e2” for digital output, and finally binds the pin variable to the corresponding pin (in traditional BASIC, the “dim” statement is used to “dimension” the shape of a variable prior to use). From then on, any modification of the pin variable is immediately reflected at the I/O pin. Line 20 starts an infinite loop. Line 30 inverts the state of the “e2" digital output pin to blink the LED. Line 40 delays the program for 500 ms. And finally line 50 ends the infinite loop. Type “run” to start the program; press <Ctrl-C> to stop the program.

Of course, if you really *just* wanted to print “Hello world!” to the terminal, you could just do:

```
> 10 print "Hello world!"
> run
Hello world!
> _
```

Hello User!

If you want to use a switch, such as the USER switch on pin “s1” of the Flea-Scope™, to condition the blinking of the LED, so that you can push the switch to stop the blinking, in StickOS BASIC it’s nearly as easy:

```
> 10 dim led as pin e2 for digital output
> 20 dim switch as pin s1 for digital input debounced inverted
> 30 while 1 do
> 40   if !switch then
> 50     let led = !led
> 60   endif
> 70   sleep 500 ms
> 80 endwhile
> run
<Ctrl-C>
STOP at line 70!
> _
```

The bulk of the program is like before, with just a few changes. Line 20 declares a “pin variable” named “switch”, then configures I/O pin “s1” for inverted (i.e., active-low) and debounced (i.e., with a low-pass filter) digital input, and finally binds the pin variable to the corresponding pin. From then on, examination of the pin variables results in the current switch state being read. Lines 40 and 60 simply condition the LED blink at line 50 on the switch not being pressed. Type “run” to start the program; press <Ctrl-C> to stop the program.

Hello Toaster Oven!

Procedure

To build the Toaster Oven Temperature Profile Controller you will need the parts listed in Table 1, and a USB host phone, tablet, or computer (the examples that follow will assume you are running Microsoft Windows, though similar procedures work with Linux, ChromeOS, and Android, as well):

Part Name	Manufacturer and Part Number
Flea-Scope™	(part number coming soon)
K-type thermocouple	Pimoroni Ltd COM1705 https://www.digikey.com/en/products/detail/pimoroni-ltd/COM1705/9975861
3V solid-state relay	Teledyne STH24D25 https://www.mouser.com/ProductDetail/Teledyne-Relays/STH24D25?qs=cFlnt7DBZX%2FYka4cen6X9Q%3D%3D
Op-amp	National Semiconductor LM358
3V Buzzer	TDK PS1240P02BT https://www.mouser.com/ProductDetail/TDK/PS1240P02BT?qs=d7g9p1yFhWaZXSY9MjKMkw%3D%3D
1k ohm resistor	Xicon 299-1K-RC
100k ohm resistor	Xicon 299-100K-RC
36 pin header	FCI 68001-236HLF

Table 1

I soldered two 12 pin headers onto the bottom of the Flea-Scope and mounted it in a solderless breadboard for ease of assembly (see Figure 2), but that is not required.

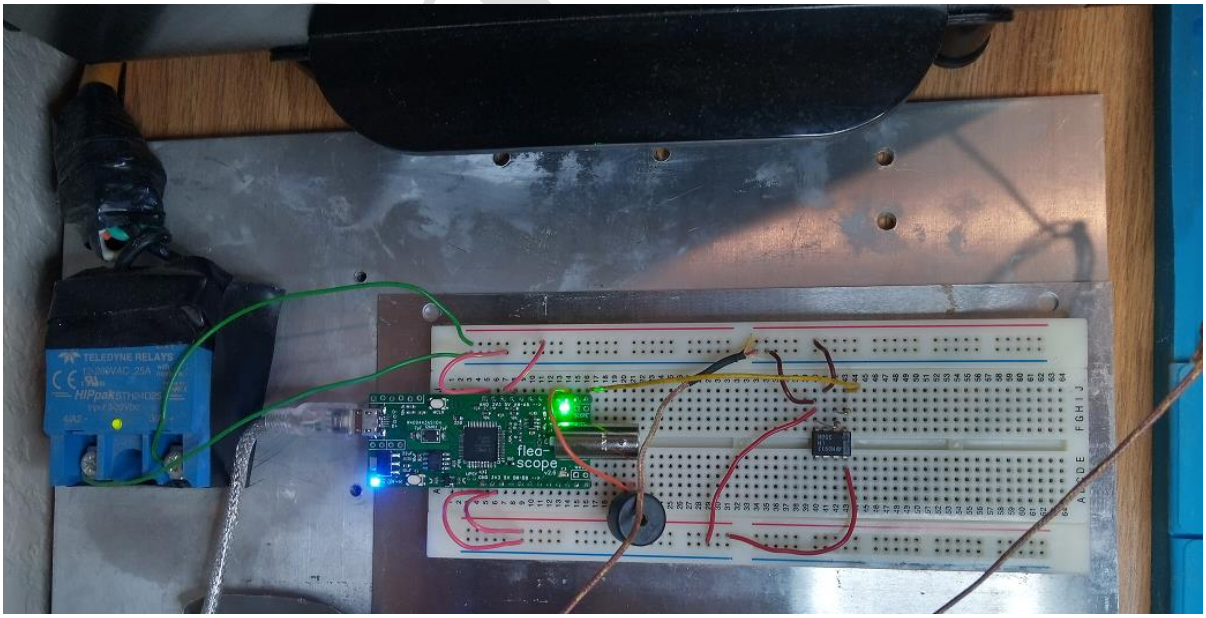


Figure 2

Schematic

The schematic for the Toaster Oven Temperature Profile Controller is shown in Figure 3.

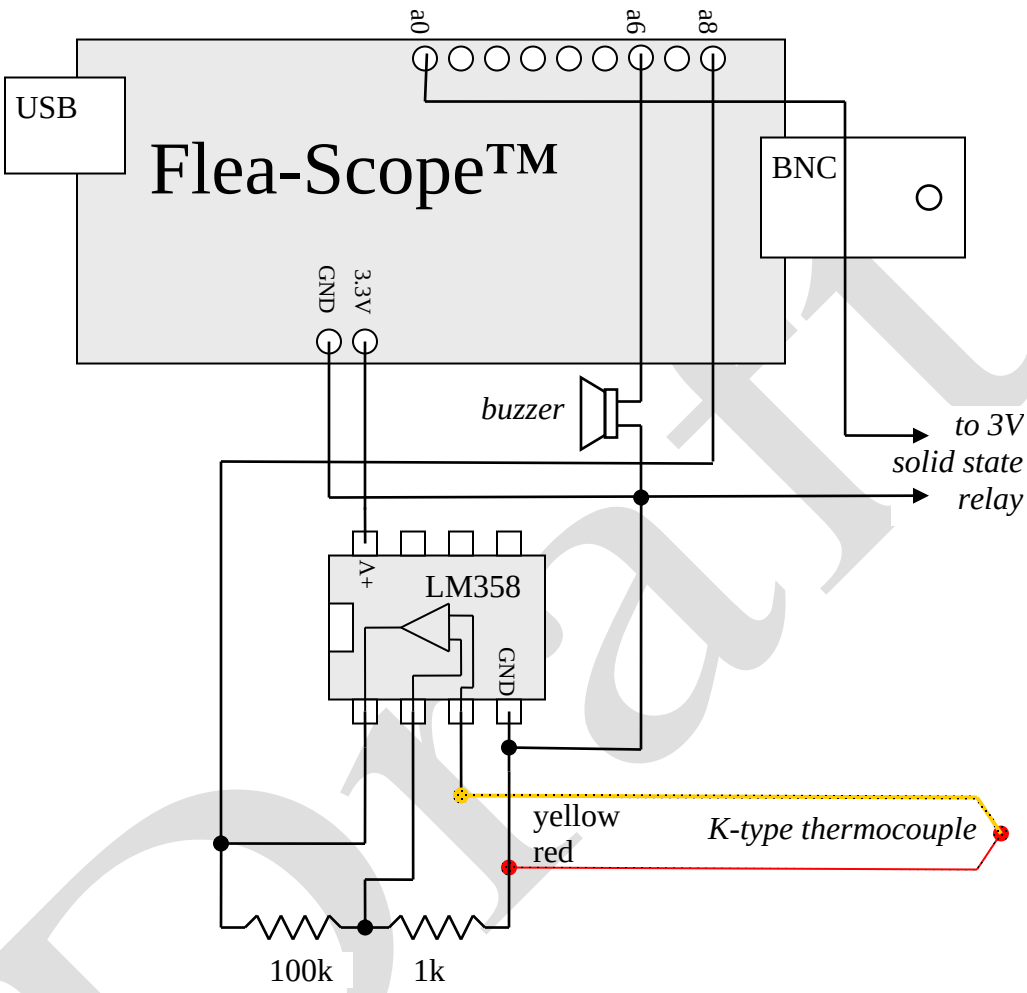


Figure 3

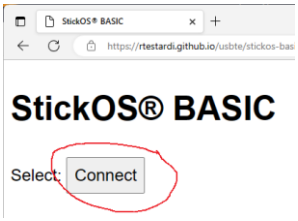
Configuration

Once the Toaster Oven Temperature Profile Controller is built, the next step is to program StickOS BASIC on the Flea-Scope board.

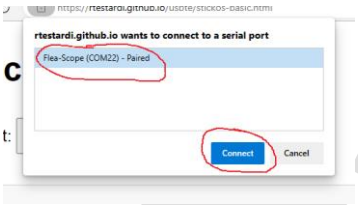
When the board is connected to a USB host computer, tablet, or phone, it will present a virtual COM port to the host computer.

At this point you can use a web-page terminal emulator to connect to the virtual COM port. Simply open: https://rtestardi.github.io/usbte/stickos-basic.html

Then Click the Connect button:



And select your Flea-Scope in the resulting dialog and click Connect again:



You should be connected to the web-page terminal emulator:

You are now ready to enter StickOS commands and/or BASIC program statements.

The BASIC Control Program

Enter the following BASIC control program at the StickOS command prompt to control the Toaster Oven Temperature Profile Controller:

```
 10 dim target, secs
 20 dim thermocouple as pin a8 for analog input
 30 dim relay as pin a0 for digital output
 40 dim buzzer as pin a6 for frequency output
 50 data 512, 90, 746, 105, 894, 20, -1, -1
 60 configure timer 0 for 1 s
 70 on timer 0 do gosub adjust
 80 while target!=-1 do
 90   sleep secs s
100   read target, secs
110 endwhile
120 off timer 0
130 relay = 0, buzzer = 100
140 sleep 1 s
150 buzzer = 0
160 end
170 sub adjust
180   relay = thermocouple<target
190   buzzer = thermocouple
200   sleep 100 ms
210   buzzer = target
220   sleep 100 ms
230   buzzer = 0
240 endsub
```

(You can copy the program to the clipboard and then use the "Paste" button on the web-page user interface to load all the lines to StickOS at once.)

Line 10 declares two simple RAM variables. Line 20 declares a “pin variable” named “thermocouple”, configures Flea-Scope pin “a8” for analog input thru a low-pass filter, and finally binds the pin variable to the corresponding pin; subsequent examination of the pin variables returns in the current ADC values being read, in millivolts (mV). Line 30 declares a digital output pin variable to manipulate the MCU pin attached to the relay; subsequently setting the variable to 1 turns the relay on and setting it to 0 turns the relay off. Line 40 declares a frequency output pin variable to manipulate the MCU pin attached to the buzzer; subsequently setting the variable to a number (in hertz) configures the MCU pin to output that frequency to the buzzer (0 turns the buzzer off).

Line 50 contains the reflow cycle temperature profile ramp target and time information:

- 90 seconds at up to 512 millivolts (125 Celsius)
- 105 seconds at up to 746 millivolts (183 Celsius)
- 20 seconds at up to 894 millivolts (220 Celsius)

Line 60 configures a periodic timer to run every second, and line 70 specifies that when the timer expires, the subroutine "adjust" should be called. The subroutine defined in lines 170 to 240 will first turn on the relay if the thermocouple is below the target voltage (or turn it off otherwise) and then play a brief audio tone of both the current thermocouple value as well as the target value to indicate the cycle progress.

Lines 80 thru 110 simply cycle thru the temperature profile ramp and update the relay and buzzer based on the state of the thermocouple.

Lines 120 to 150 unconfigures the periodic timer and turns off the relay and then plays a short low frequency tone to indicate the end of the cycle.

Then the program ends.

To save the program to the flash filesystem, type:

```
save
```

To run the program, type:

```
run
```

To set the program to autorun on MCU power-up, type:

```
autorun on
```

Diagnostics

What if your program doesn't work? What if you have a bug in your op-amp or relay circuitry? StickOS supports fully interactive control of the MCU. You can start running the program by typing "run" and then press <Ctrl-C> to stop it after a minute or two and you will see:

```
> run
<Ctrl-C>
STOP at line 90!
```

At that point you can print the current value of the thermocouple, in millivolts, with:

```
> print thermocouple
609
```

You can turn on the MCU output connected to the relay with:

```
> let relay = 1
```

You can then watch the thermocouple voltage increase:

```
> print thermocouple
644
```

And turn off the MCU output connected to the relay:

```
> let relay = 0
```

And watch the thermocouple decrease:

```
> print thermocouple
570
```

You can even continue the program from where it left off:

```
> cont
```

You can use breakpoints, assertions, watchpoints, single-stepping, execution tracing, live variable and pin examination and manipulation, and even edit-and-continue as part of the StickOS interactive debugging experience.

More Information

Full StickOS documentation and downloads for various supported MCUs can be found at: https://rtestardi.github.io/StickOS/

Toaster Oven Tips

Fine pitched SMT reflow soldering can be an intimidating task to the newcomer, but I have found that I can reliably reflow “CPUSticks”, including 0.5mm pitch QFP’s and even 0.5mm QFN’s with just a few tricks:

- To mount a QFN (leadless) package, I first flux the component pins and then tin them with my soldering iron; each pin ends up with a little bump of solder on it. I then flux the board pads and tin them, with a similar and opposing bump (it definitely helps to have a solder mask on the board, but I have done it without). Then I flux everything again and align the bumps on the QFN pads with the opposing bumps on the board, and prepare for reflow using your new Toaster Oven Temperature Profile Controller.
- To mount the QFP (leaded) packages, I use a similar procedure but do not tin the delicate component pins.
- After reflow, if touchup is needed (the typical problem will be “opens”, not “shorts”), I use a generous amount of flux at the pin/pad interface and then drag the ever slightest amount of solder across it with my soldering iron. Note that the flux is critical in this step to avoid any bridging/shorts.
- I used a Weller WES51 soldering iron with an ETP (1/32" screwdriver) tip, 62/36/2 silver solder from Radio Shack (in a convenient 1.5 oz spool), a Kester #2331-ZX water soluble flux pen, and a PanaVise PV Jr. Model 201 (to hold things).

About the Author

Rich Testardi lives in Berthoud, Colorado, with his wife and daughter and thinks microcontrollers are the coolest thing; his dream is to make them easy enough to use to introduce into high-school curriculums, since that was where his initial exposure was. Since that time long ago, many forces, notably SMT and ever-increasing MCU integration (and 500+ page reference manuals), have conspired to make these projects less accessible to high-school students; he hopes to reverse that trend with StickOS and Flea-Scope!