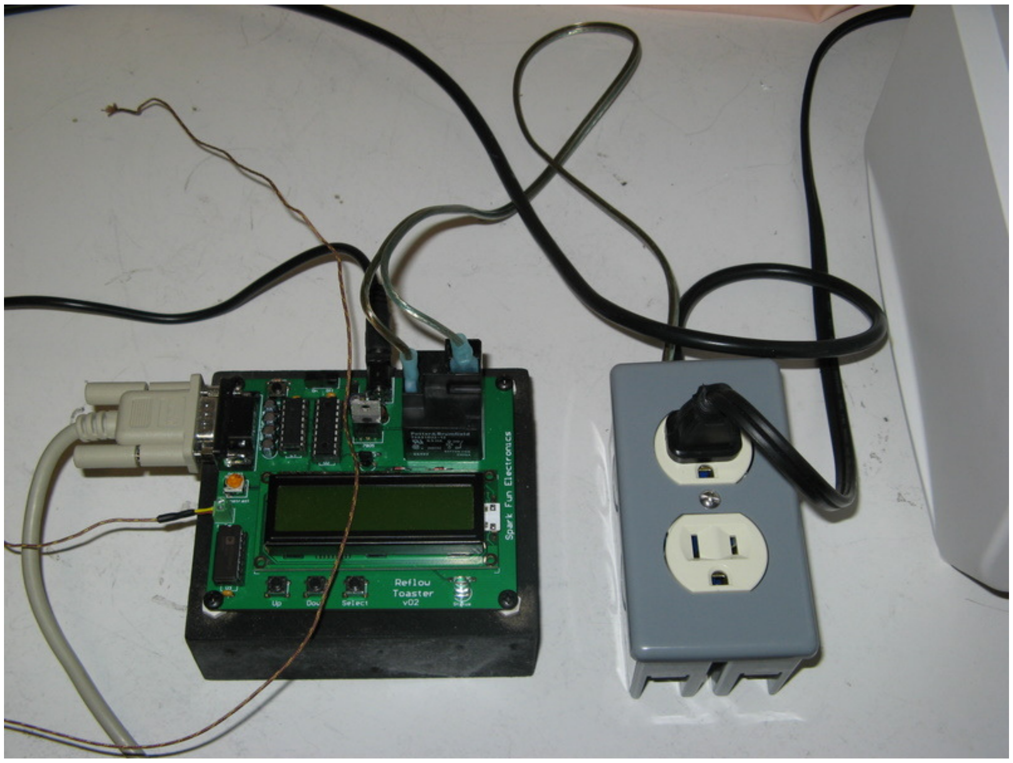
OvenFlow v1.0

User Manual



By Kit Ryan
W4KIT

January 2008

Author's Note

Welcome to OvenFlow 1.0. This software, when installed on the PIC16F88 in the SparkFun Toaster Oven Controller hardware, will turn an ordinary toaster oven into a programmable reflow soldering oven. This program gives the User significant control over the time/temperature profile of the oven and should permit both lead-based and lead-free soldering for surface mount devices. Please note that due to the wide variation in toaster ovens and potential User implementation of the controller, there are no warrantees, implied or otherwise, associated with the use of the this software. Enjoy!

Kit Ryan

1. Safety Warnings

SAFETY NOTICE: This project involves line voltage connections which are potentially fatal if mishandled. Double check all wires before applying line voltages.

SAFETY NOTICE: Using the Controller for reflow soldering can result in operating a toaster oven at extremely high temperatures, which can burn fingers or cause fires.

SAFETY NOTICE: Solder materials, whether lead-based or lead-free, can be toxic if ingested. The toaster oven you use for this project should be permanently dedicated to electronics and no longer used for heating food products.

PLEASE USE CAUTION WHEN OPERATING THIS CONTROLLER. THE AUTHOR CANNOT ASSUME ANY LIABILITY FOR USING THE SPARKFUN ELECTRONICS CONTROLLER WITH THIS SOFTWARE.

2. Getting Started

This manual covers the operation of the OvenFlow software on the SparkFun Electronics Reflow Toaster Oven Controller. It is based on v02 of the Reflow Toaster hardware and may not operate correctly on other versions. The hardware version number is printed on the PCB next to the LED.

The User should start by assembling the Controller hardware according to the SparkFun instructions. Before loading OvenFlow, power-up the Controller with the boot loader from SparkFun still in the PIC 16F88 memory (that's the way it's shipped). If the LED flashes and the LCD readout has their logo on it, then you've probably done a good construction job.

You must also construct a suitable outlet box for plugging in the toaster oven so that the Controller relay can switch it on and off safely. A typical example is shown below. Inside the box, the positive (hot) lead from the power cord has been spliced into the two leads going to the relay. Insulated lugs are used at the relay to minimize accidental shocks.

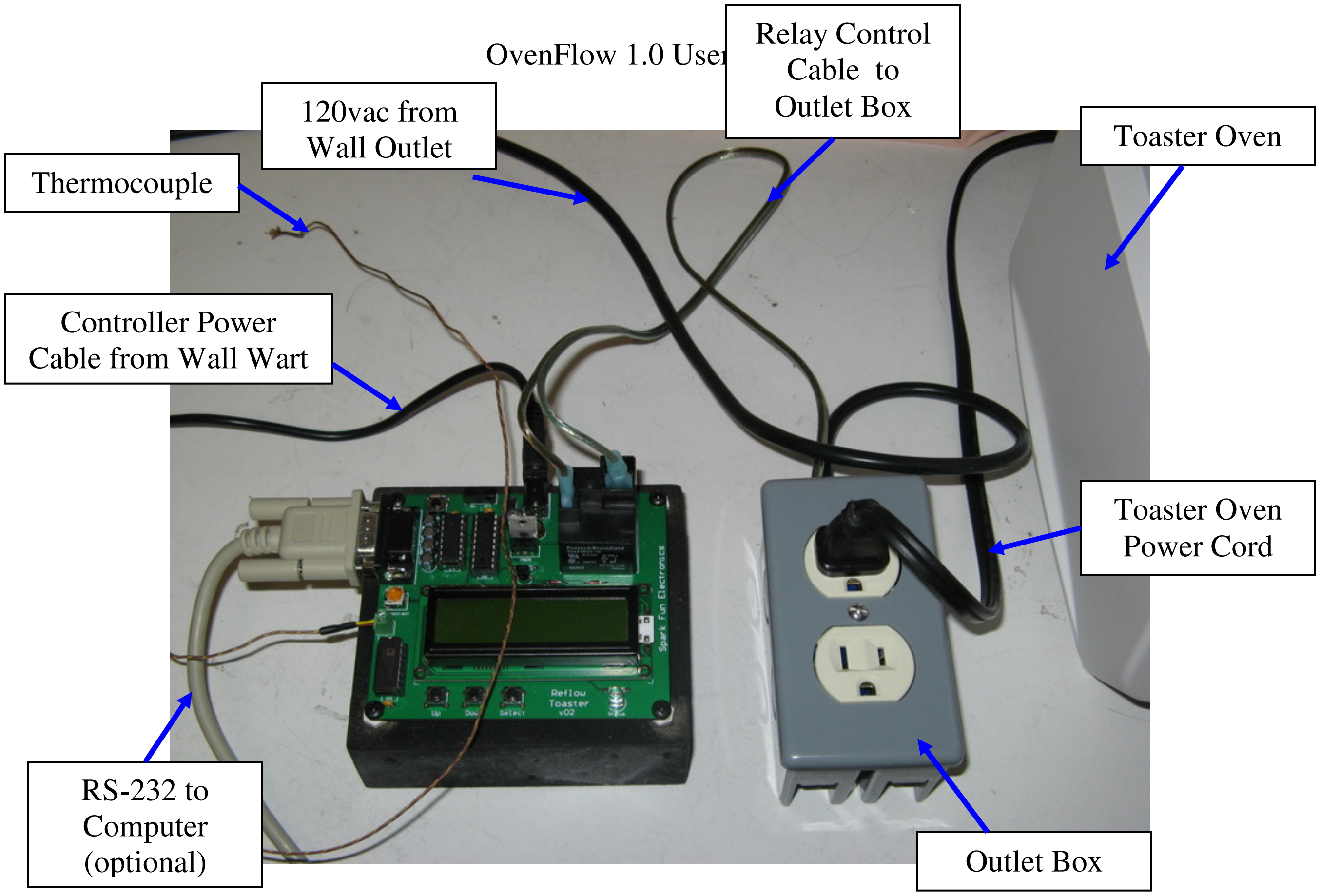

Relay Control Cable to Outlet Box
120vac from Wall Outlet
Toaster Oven
Thermocouple
Controller Power Cable from Wall Wart
Toaster Oven Power Cord
RS-232 to Computer (optional)
Outlet Box

Figure 1 - Assembled Controller System

3. Loading the Software

<u>If you cannot get the 16F88 already programmed with this software, you will need a PIC programmer to load the OvenFlow program</u>. The boot loader that SparkFun provides unfortunately will not work on memory loads greater than 2k and this program takes almost the full 4k of available PIC memory. Programmers are available at modest cost from Microchip. Clones and alternative programmers are also readily available on eBay and from other sources which advertise in Nuts and Volts Magazine.

Be aware that during the programming process (sometimes referred to as “burning”) the boot loader software in the PIC is erased. From that point on, you cannot go back and use the serial port for programming unless you reload the boot loader with the programmer.

Before actually programming the PIC, you will need to download and install the free Microchip IDE software from their web site. It simplifies and standardizes the programming process (but you can use other programming software if you prefer).

Of course, be sure you have downloaded the OvenFlow1.0 “hex” file from the SparkFun web site to your computer.

To load the OvenFlow software follow these steps:

1. Install the PIC 16F88 in the correct programming socket on your programmer

2. Start the Microchip IDE and open the "Programmer" menu.
3. From the "Select Programmer" list, click on the programmer type that matches your hardware.
4. Now, open the "File" menu and select "Import".
5. Find the OvenFlow1.0.hex file on your computer and select it.
6. Go back to the "Programmer" menu and select "Program". This option should have been grayed-out until you selected a programmer but should now be available.

The programming process should start immediately and take only a few seconds. The IDE should report back on the screen that the download was successful. If not, try troubleshooting the programmer IDE connection until it recognizes your hardware. Note that the ICD2 will give an error message if the PIC16F88 is not already loaded into the programming socket when the programmer is selected. No problem, just put the chip in the socket and ask it to connect again.

Put the PIC chip back into the controller and power-up. If the LED flashes and the screen now says "SparkFun & Ryan" as in the picture below, you got a good download. Now for the fun part.

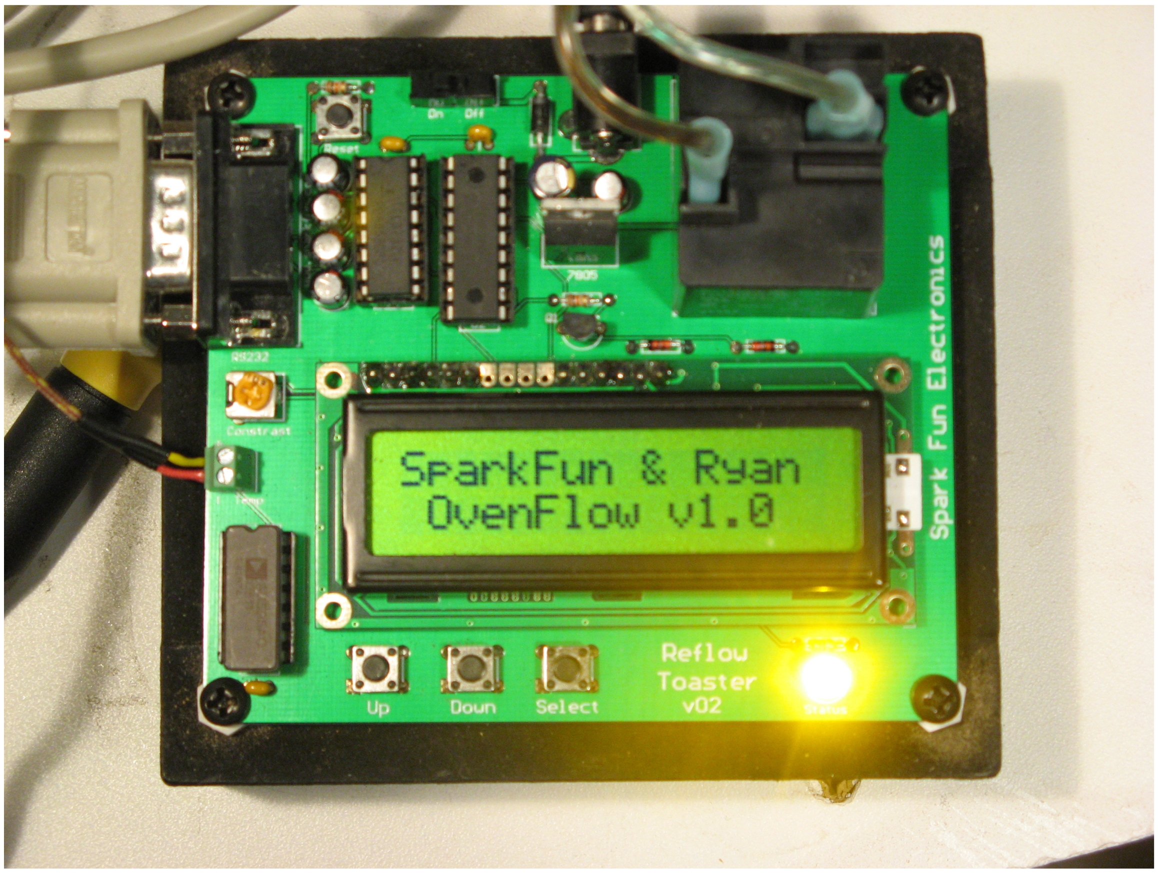


Figure 2 - OvenFlow Splash Screen

4. OvenFlow Software Description

With only a few buttons and 32 characters of display, OvenFlow still manages to squeeze in 3 main operating modes for the controller, plus program editing features, and the ability to change and store selected setup parameters. With a little practice, you will be able to quickly zip to the function menu you want and operate your oven. Each mode will be described in detail in the following sections. But first, here's an overview of the whole program structure.

To start, after the flash screen seen in Figure 2, the program goes into the Main Menu mode with the screen shown in Figure 3. The top line shows the menu option available for selection while the 2nd line screen positions are tied to the 3 buttons on the controller Up, Down and Select. The "x" for the Up and Down buttons indicates that there is no command for them in the current mode. Pushing the Select button will cycle through all the available options.

Since the button operation in all modes is similar, take a moment to try the following. Push and quickly release the Select button. The menu option changes to the "Nx" (Next) position, in this case going from Manual to Semi-Auto. Push again to go to the Program1 and then Program2 modes. One more push gets you to the Setup mode. Finally, another push and you've cycled back to the Manual mode.

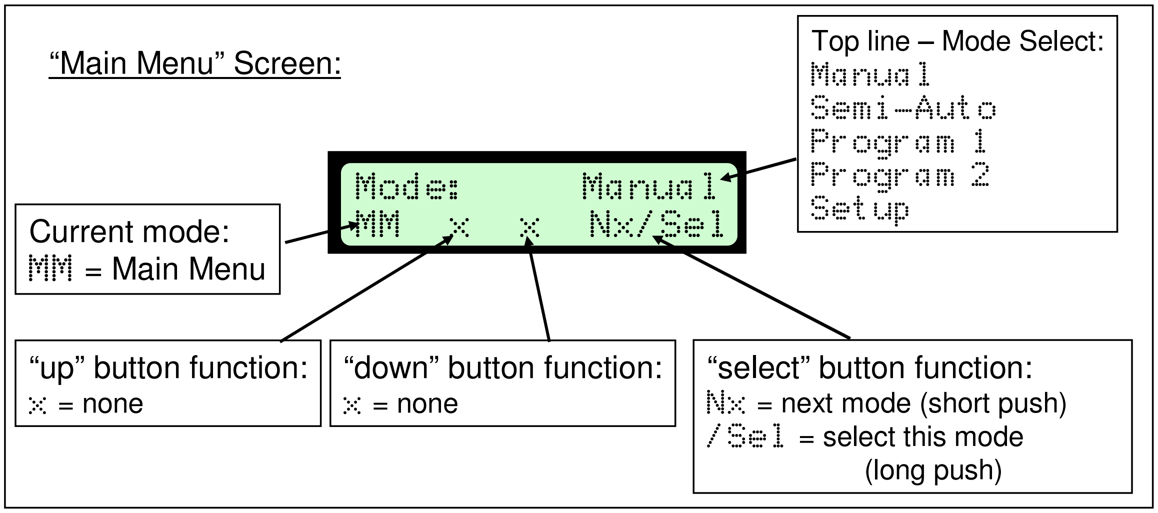


Figure 3 - Main Menu Screen

Now if you notice there is a "/" (slash) next to the Nx text. The command to the right of the slash will execute if a "long" button push is made, that is greater than 2 seconds. If you press and hold the Select button, you will "Sel" (Select) the Manual Mode and its screen appears. To get back to the Main Menu simply do a "long" push on the Select button which executes the "Ex" (Exit) command. The Exit command appears on every screen to get you back to the previous mode.

4.1 Manual Mode

The Manual Mode screen is shown in Figure 4. Each button's commands are shown in the figure. This mode provides the User with independent control of a timer (1 second

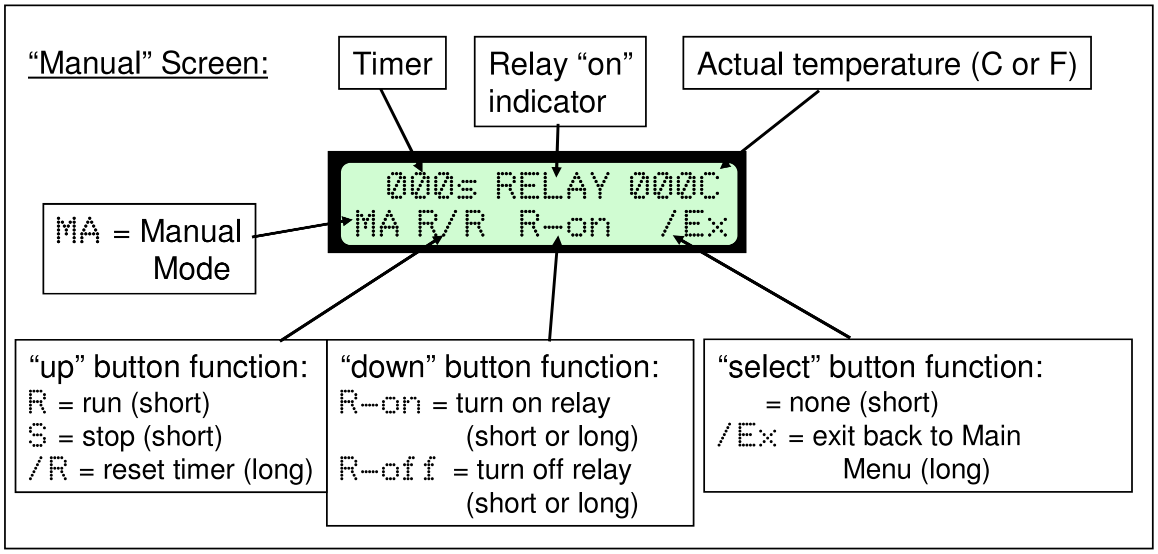


Figure 4 - Manual Mode Screen

increments) with the Up button and the Relay with the Down button. The timer has "R" (Run), "S" (Stop) for short button pushes and a "R" (Reset) for a long push which takes the timer back to 0. The exact timing of 1 second is dependent on the internal PIC oscillator which is not crystal controlled in this circuit design. The 16F88 spec sheet shows +/- 2% variation for this internal PIC timer. OvenFlow provides for making adjustments of the displayed timer (not the internal PIC timer) by altering the countdown timer used by the software. Please see the Setup mode for this function. In order fit multiple readouts in the LDC display, the timer is limited to 3 digits or 999 seconds (over 16 minutes). If you go past this value, it rolls over to 0 seconds and keeps going. It is doubtful that any soldering will take longer than a few minutes so this should provide a reasonable timer function.

The Down button simply toggles the relay on or off. Because the relay is switching potentially dangerous line voltages, the top line of the LCD displays "RELAY" when the relay is closed. You can also select to have the LED turn on as yet another indicator that the relay is closed and the oven is on. This is the default setting but you can turn it off in Setup mode. The "actual" temperature as read by the thermocouple is also shown in the top line so you can act as the feedback element and turn the relay on or off as needed.

To return to the Main Menu, do a "long" press on the Ex (Exit) command of the Select button. This cancels the relay if it was on and resets the timer to 0 for future use.

4.2 Semi Automatic Mode

The SemiAuto Mode screen is shown in Figure 5. Each button's commands are shown in the figure. This mode provides the User with an easy way to set a temperature and have

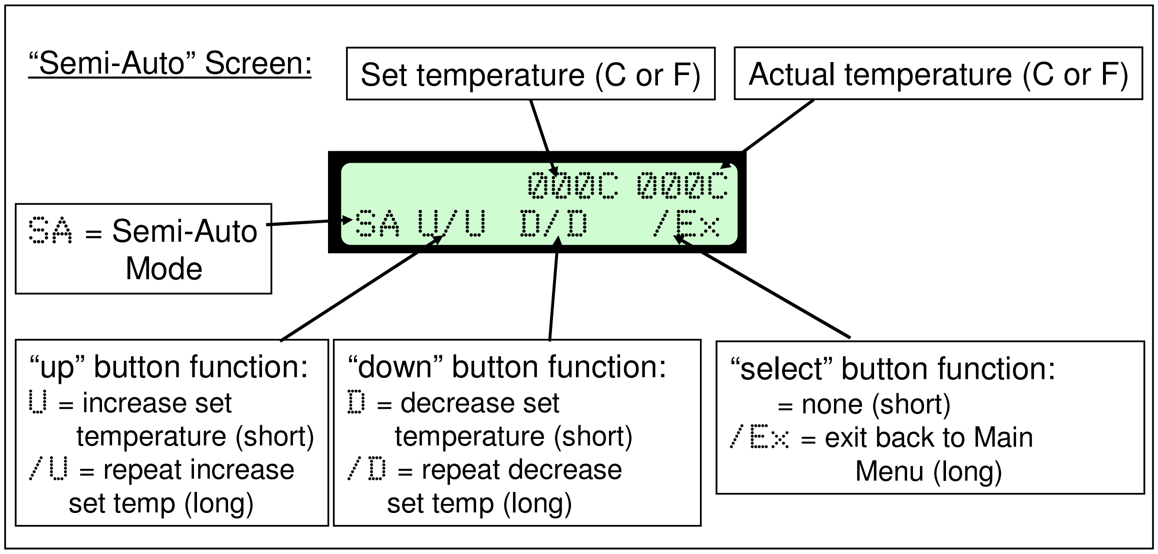

"Semi-Auto" Screen:
Set temperature (C or F)
Actual temperature (C or F)
000C 000C
SA U/U D/D /Ex
SA = Semi-Auto Mode
"up" button function:
U = increase set temperature (short)
/U = repeat increase set temp (long)
"down" button function:
D = decrease set temperature (short)
/D = repeat decrease set temp (long)
"select" button function:
= none (short)
/Ex = exit back to Main Menu (long)

Figure 5 - SemiAuto Mode Screen

the controller keep it there automatically. Due to the limited number of buttons, there is no timer function in this mode.

The Up and Down buttons change the desired "set" temperature. Short button pushes will increment or decrement the desired temperature by 5 degrees (default). The User can select any value between 1 and 10 degrees for the amount to increment/decrement in the Setup mode. For faster changing, just press and hold either button. The temperature will change rapidly after 2 seconds, incrementing or decrementing the standard amount approximately 10 times per second. It is called "Speed Button" and allows the User to quickly get close to the desired value, then use individual button pushes for the fine tuning.

Note that the display also includes the "actual" temperature as read by the thermocouple. As soon as the "set" temperature exceeds the "actual" temperature, the relay will close and start controlling the temperature using the thermocouple for feedback. Once the "actual" temperature reaches the "set" temperature, the software will cycle the relay on and off appropriately to maintain that value within a couple of degrees.

Also note that you can change the "set" temperature at any time and the software will adjust its automatic relay control function to the new setting.

To return to the Main Menu, do a "long" press on the Ex (Exit) command of the Select button. This cancels the relay if it was on.

4.3 Program Mode

This covers Program1 and Program2 modes, which are identical. These modes are the heart of the software and provide the User with a way of pre-setting up to two desired time/temperature profiles and then executing either with just one button push. This should allow meeting the specified profile for lead-based soldering shown in Figure 6 from Kestor's web site or for lead-free soldering which requires slightly higher temperatures.

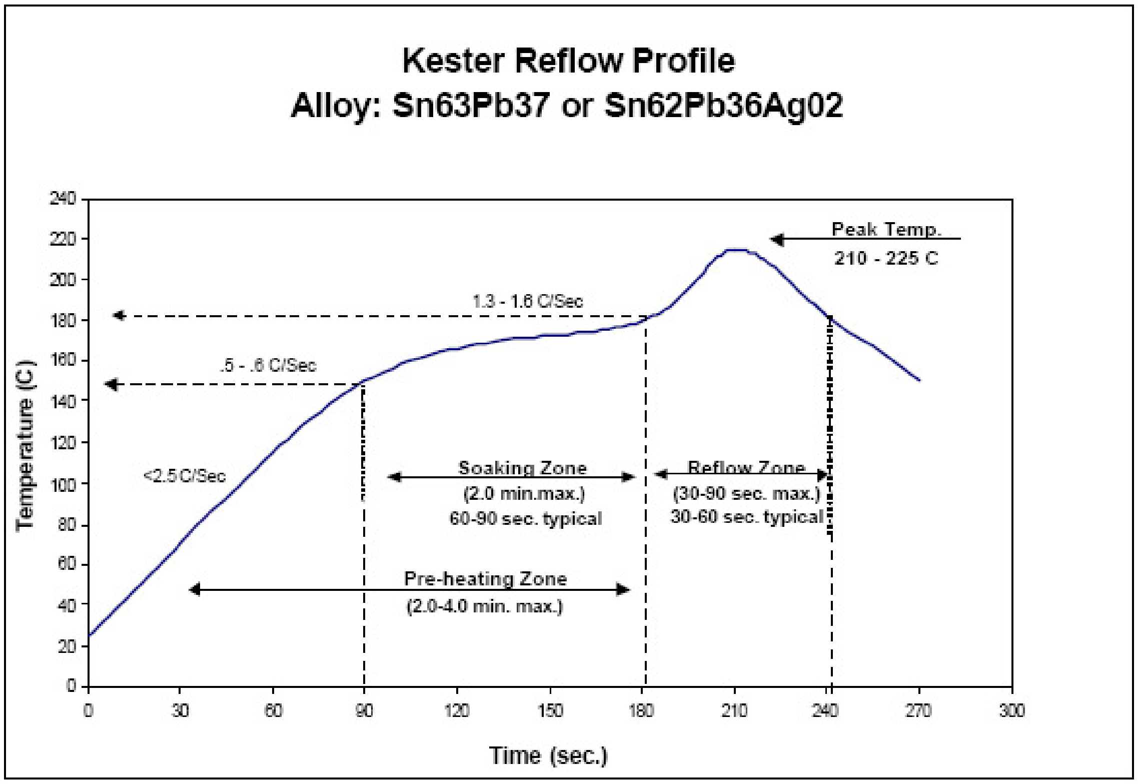


Figure 6 - Kester Lead-Based Surface Mount Soldering Profile

The Program screen is shown in Figure 7. This mode allows for up to a 10 step program to be executed. Steps are number 0-9 to limit the display to 1 character. <u>The program steps must already be saved in the PIC memory.</u> OvenFlow comes with a Program1

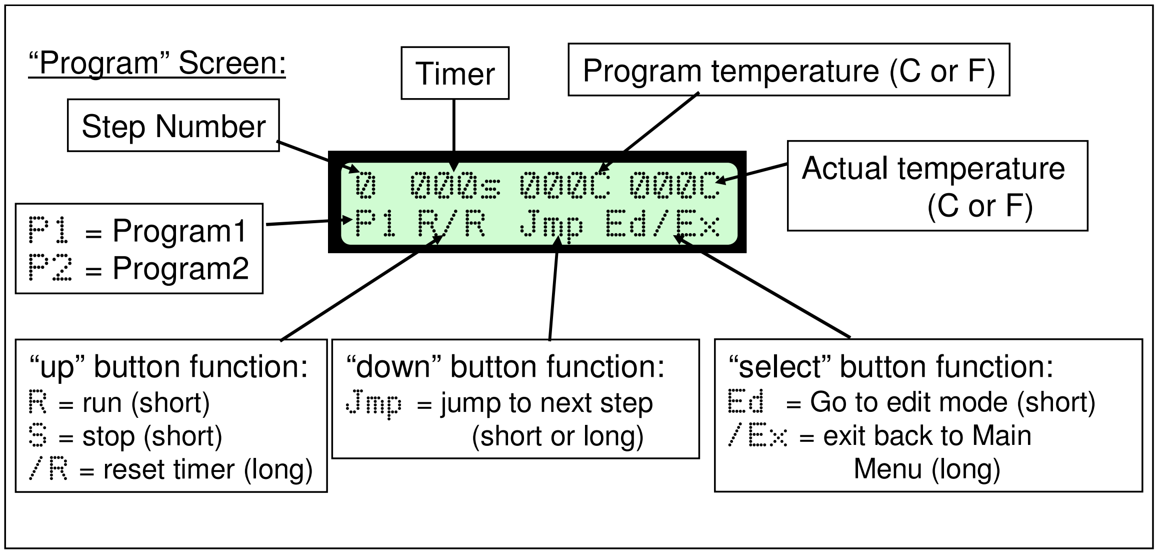


Figure 7 – Program Mode Screen

memory sequence that closely resembles the Kester lead-based soldering profile in a 5 step program as seen in Figure 8. The User can easily modify this profile or create a new one using the Editing Mode described in the next section.

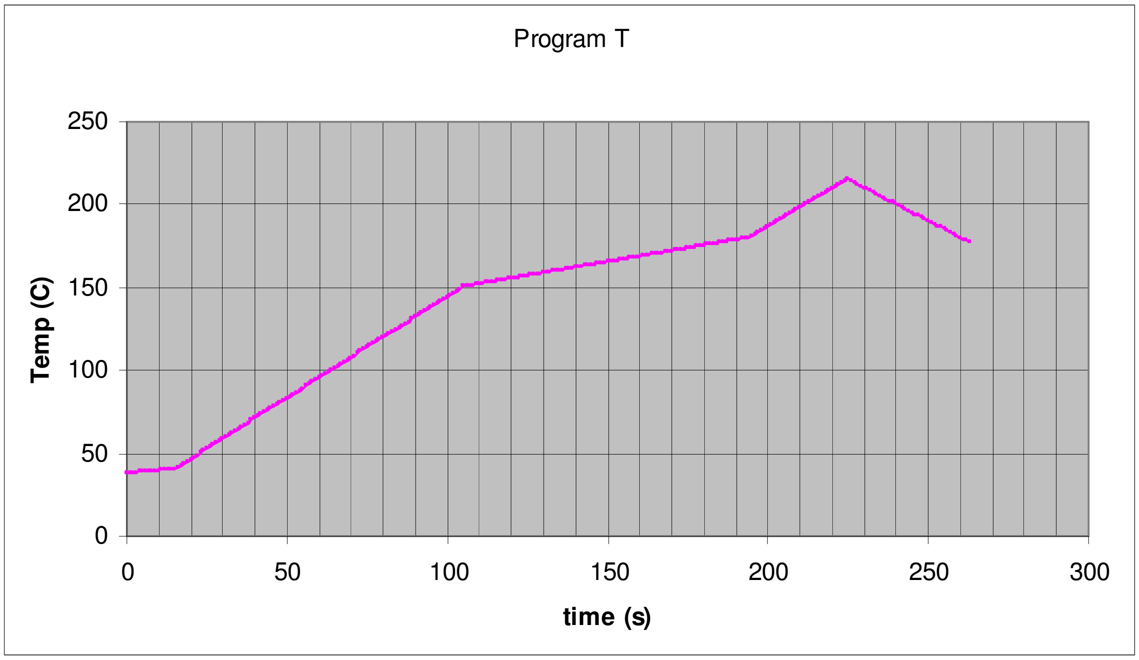


Figure 8 – Built-In Program1

To execute the Program1 (or 2), push the Up button to "R" (Run), "S" (Stop) or (with a long push) "R" (Reset) the program. It starts with step 0 and goes until the next "timer" value in the program goes to zero. This is the indicator for the end of the program. Once the program has started, the controller will turn the relay on and off as appropriate to match the programmed time/temperature profile as closely as possible. It tries to anticipate the needed relay action by looking ahead, that is, looking at where the temperature needs to be from the program and comparing it to where the temperature is predicted to be based on the slope of the temperature curve at the moment. Since toaster ovens have delays in heating and cooling, it is a tricky business to get this right but getting within a few degrees is probably close enough. There is one parameter available to the User to try to make adjustments for his/her particular oven available in the Setup Mode, but it is still hard to get all factors correct with just one variable.

The Down button provides a convenient way to do two things: (a) review the program before executing it and (b) "Jmp" (jump) to a specific step rather than start at the beginning, if desired. Push the Down button to cycle through all the steps in memory for this program. When a "timer = 0" step is detected, it signals the end of the program and the display cycles back to step 0.

The step parameters are shown as "Timer" and "Program Temperature". This can be a little confusing, but the Timer value is the time (in seconds) at the <u>end</u> of a step. The temperature is the corresponding desired temperature at the end of that step. For example, for step 0 (always the first step), a setting of 15s, 40C means that after 15 seconds, the temperature is expected to be 40C. A straight line ramp is assumed between the beginning and ending temperatures for each step. Step 0 always assumes an above room temperature start (38C) to force the relay to turn on right away because of the normal time delay in heating up. So the program constructs a line between 0 seconds at 38C and 15 seconds at 40C and calculates all the points in between as the "set" temperature at a particular time. Looking at the built-in program in Figure 9 will help in understanding this algorithm.

For step 1 (the second step) the "timer" value is again expressed in terms of the time just for that step. If the program shows 90s, 150C for step 1, then it would use the starting point as the end of step 0 (40C) as its beginning, and construct a straight line to step 1's end point 90s later. The total elapsed time is then 105s. This approach provides for a continuous, well behaved programming structure and allows for smooth relay operation. Setting a step timer to 0 indicates the end of the program and the relay shuts off, the display returns to step 0 and the timer resets.

A quick push on the Select button will produce the "Ed" (Edit) screen where the User can change the program parameters. "Ex" (Exit) is the usual command to return to the Main Menu. The relay is automatically turned off when this command is executed.

4.4 Program Editing Mode

This mode is only reached from the Program1 or Program2 mode. A time/temperature profile of up to 10 steps can be entered or modified in this mode and is automatically, instantaneously saved to the EEPROM memory of the PIC microcontroller. The program is saved even after powering off the controller.

The program editing screen is shown in Figure 9. For each step number, the step time

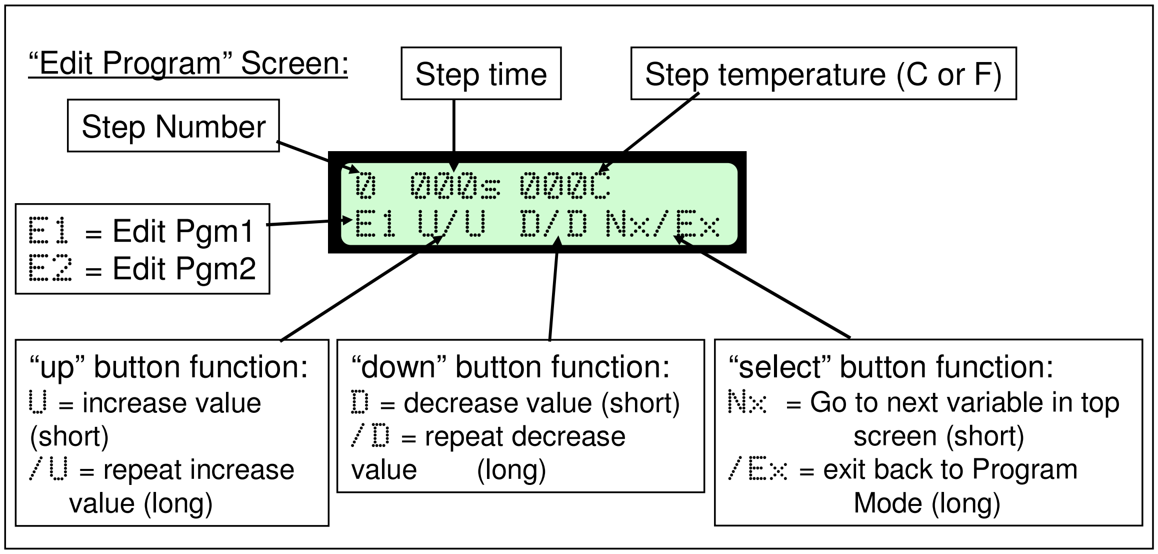

"Edit Program" Screen:
Step time
Step temperature (C or F)
Step Number
0 000s 000C
E1 U/U D/D Nx/Ex
E1 = Edit Pgm1
E2 = Edit Pgm2
"up" button function:
U = increase value (short)
/U = repeat increase value (long)
"down" button function:
D = decrease value (short)
/D = repeat decrease value (long)
"select" button function:
Nx = Go to next variable in top screen (short)
/Ex = exit back to Program Mode (long)

Figure 9 – Program Editing Screen

and step temperature are shown and can be modified. To traverse the program, the "Nx" (Next) command on the Select button will move the cursor among the 3 parameters in the top line of the LCD display, Step, Time and Temperature. The Up and Down buttons then allow incrementing/decrementing of that value according to the following:

Step number = increment or decrement by 1 step at a time.

Step time or temperature = increment or decrement by the standard value (1-10) set by the parameter in the Setup mode. Default is 5 seconds or 5 degrees.

The "long" pushes for Up or Down engage the Speed Button function and allow for rapid changes to large values of time or temperature. Up to 999 seconds can be programmed, limited by the 3-digit display field width. Going over 999 seconds or degrees will cause a "roll-over" back to zero. Similarly, going below zero will roll-over to 999, etc.

As soon as you change a value of time or temperature, it is stored in EEPROM and remembered. The first step in which you leave the time as zero will be the indication of the end of the program.

The built-in program is as follows (in case you erase it and want to re-install it):

Step	Time	Elapsed Time	Temperature
-	-	0s	24C (assumed room temp start)
0	15s	15s	40C
1	90s	105s	150C
2	90s	195s	180C
3	30s	225s	215C
4	60s	285s	150C
5	0s	(end of program)	

Remember that the display will normally show the total elapsed time during the program execution. This is confusing but somewhat driven by the limited memory size of the PIC. Using elapsed time for programming each step time would have taken a lot more computations.

Exiting this mode via the "Ex" (Exit) command will go back to the Program1 (or 2) mode in which you can immediately execute the new program.

4.5 Setup Mode

One of the nice features of the PIC 16F88 is that it can write to its own EEPROM memory. OvenFlow stores 5 parameters in this memory that can be modified by the User to customize its operation. The Setup screen is shown in Figure 10.

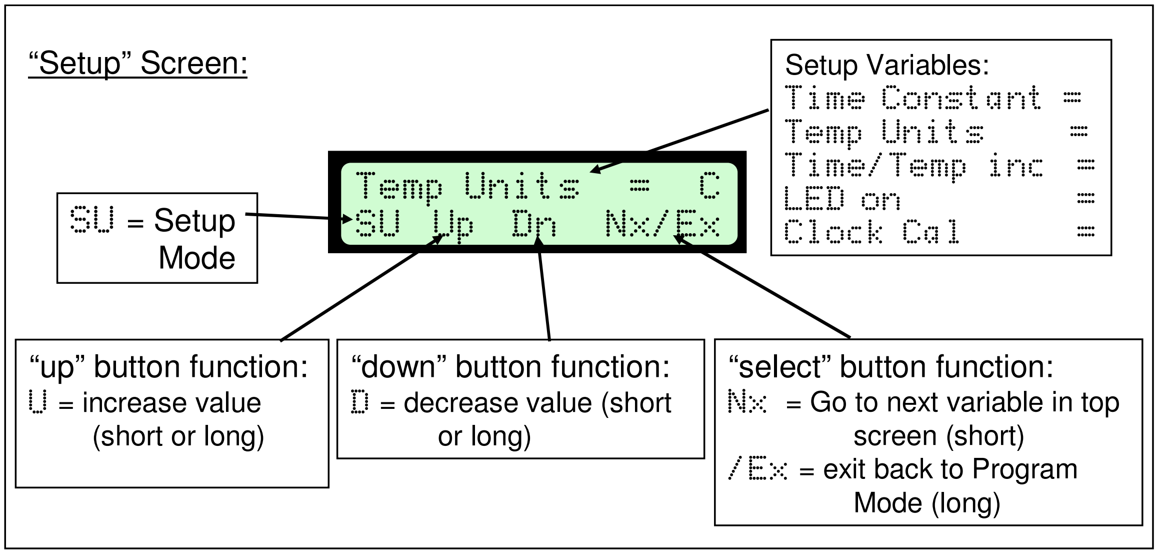


Figure 10 – Setup Screen

Pushing the Select button "Nx" (Next) will cycle through the parameters. Up and Down allow the values to raised or lowered as described under each parameter.

Time Constant: This parameter attempts to account for the time delay between turning on the relay and heating of the oven elements. From a cold start, it can take 20 seconds or more for them to get red hot. During cycling when the relay goes off for a few seconds, then turns back on, the delay can be 5 to 10 seconds. The Time Constant value (default = 12 seconds) tries to average this delay and affects the "look-ahead" time when the oven is following a program in Program Mode. Slower ovens can have longer delays so this is one to experiment with if you are not satisfied with the tracking of the oven against a program. Up and Down will raise or lower the value by 1 second per push.

Temp Units: You can use degrees Celsius or Fahrenheit, whichever you prefer. Celsius is the default units. Remember one thing – the program stored in EEPROM is in the units you used when editing that program, so if you decide to change the units afterward, the display will show the correct "actual" value in the new units but the program will still be in the other units! Recommendation is to pick the units you prefer right in the beginning and don't change them.

Time/Temp Increment: When programming a time or temperature value, the Up and Down buttons will add/reduce the value by this (Time/Temp Increment) amount. Similarly, Speed Button (long pushes) will rapidly increase or decrease the value by this same increment. The User can change the default value of 5 (seconds or degrees) to any value from 1 to 10 in Setup.

LED On: By default, the LED on the controller board is turned on whenever the relay is activated. This is a safety feature to alert the User that the oven is turned on. If you prefer to not have the light turn on, the parameter can be set here to off. The Up and Down buttons toggle this value.

Clock Calibration: This project uses the internal PIC RC clock running at 8 MHz rather than go to the added expense of a crystal clock. It also frees up I/O pins on the device for other functions. The PIC clock is on specified to be within 2% of its nominal value so the 1 second timer in OvenFlow, which depends on the clock frequency, may also be off. If you want to check its accuracy, go to Manual Mode and check the timer against a stop watch or digital watch with a second hand. Go at least 100 seconds (longer is more accurate) and see how far off it is from your watch. The parameter adjustment is from 1 to 10 with 5 as the default. Each increment or decrement away from 5 will have about 0.5% affect on the clock value.

When any of these parameters are changed, the new values are immediately used by OvenFlow and also stored in the EEPROM memory.

Press and hold the Select button "Ex" (Exit) to return to Main Menu.

5. Getting Output to Your Computer

The serial port on the controller is set up to send some data to your computer if you care to capture it. To get the data, you need to install the cable, of course, then run Hyperterminal (or similar program) on your computer. After setting up the right Comm port, leave all the serial control variables as default. You should start seeing the following on the Hyperterminal screen right away. Column headings are not sent by the program – just the data.

```
Elapsed Time   Set Temp   Actual Temp   Relay

000             25        22             1          (Relay: 0 = off, 1 = on)
001             26        22             1
```

Data flow can be started or stopped from your computer by using the On-Hook, Off-Hook telephone icons in the Hyperterminal menu line. This is a convenient way to capture only the data you want. You can also clear any old or unwanted data using the Clear Backscroll or Clear Screen commands.

Data is sent once per second. A good way to collect data is as follows:

1. Hook up the cable and run Hyperterminal.
2. Look to make sure data is coming in.
3. Hit the Off-Hook icon to stop data flow.
4. Clear the Screen of data.
5. Get the controller ready to run with the mode you want but don't press the Run button yet.
6. Now click on the On-Hook icon to restart Hyperterminal.
7. Quickly hit the Run button on the microcontroller.
8. Collect data for as long as you want.

A useful feature of having the data is that you can import it into Excel and easily plot it or save it to a file. To get the data into Excel, at least one way to do it is as follows:

1. Capture the data in Hyperterminal, then stop the data transmission using the Off-Hook icon.
2. Highlight all the data you want to transfer using the mouse.
3. Click on Copy.
4. Open Notepad, the simple editor that comes with Windows.
5. Click on Paste to bring the data into Notepad, then Save it to a text file you name.
6. Open Excel and click on Open from the File menu.
7. Select the Notepad text file you just saved. You will have to select "All Files" in the file dialog box to see the .txt file you want.
8. Excel will start the "Text Import Wizard" to help you get the data into the right columns. On the first screen, select "delimited", then Next. On the second screen, select "Space" as the delimiting character between columns. The image in the screen should clearly show the data elements in separate columns now. Hit Next, then Finish and you will have the data in Excel is a useful Form.
9. Select Insert Chart and select the rows you want to display. You can now plot the performance of your oven. A plot showing both the Program (set) temperature in pink and the actual oven temperature from a typical toaster oven is shown in blue in Figure 11. Some oscillation is visible as the controller tries to deal with the delays in the heating/cooling of the oven coils.

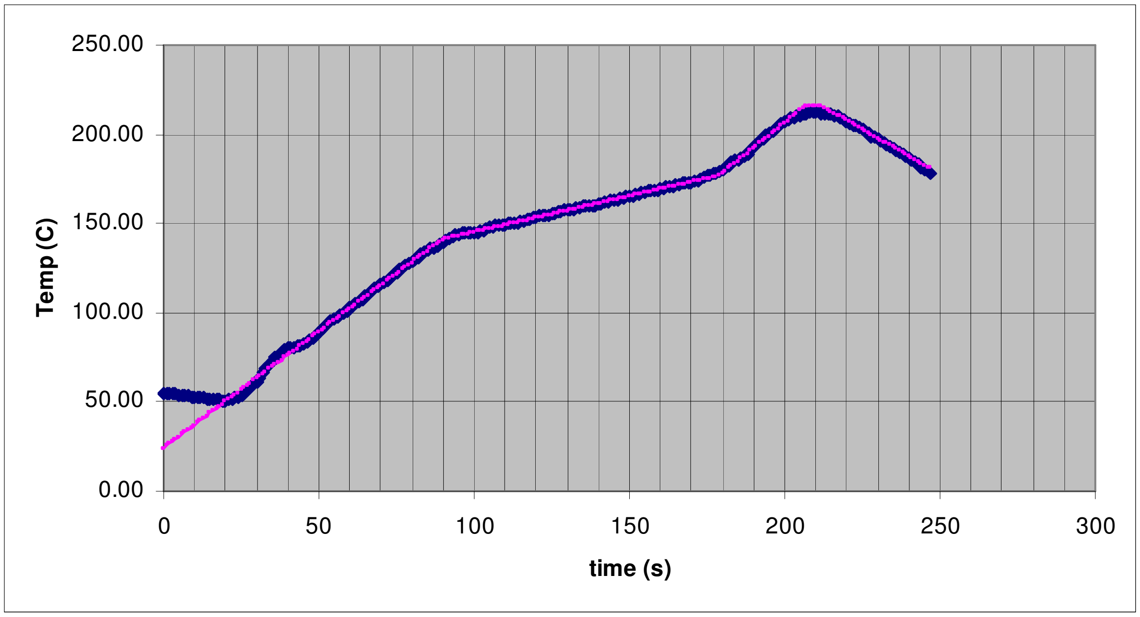


Figure 11 – Serial Output Excel Plot

6. Modifying the Program

If you're reading this section, you probably already know something about programming microcomputers. It is expected that the reader knows the C language although the code itself is heavily commented to assist in understanding what each line is doing.

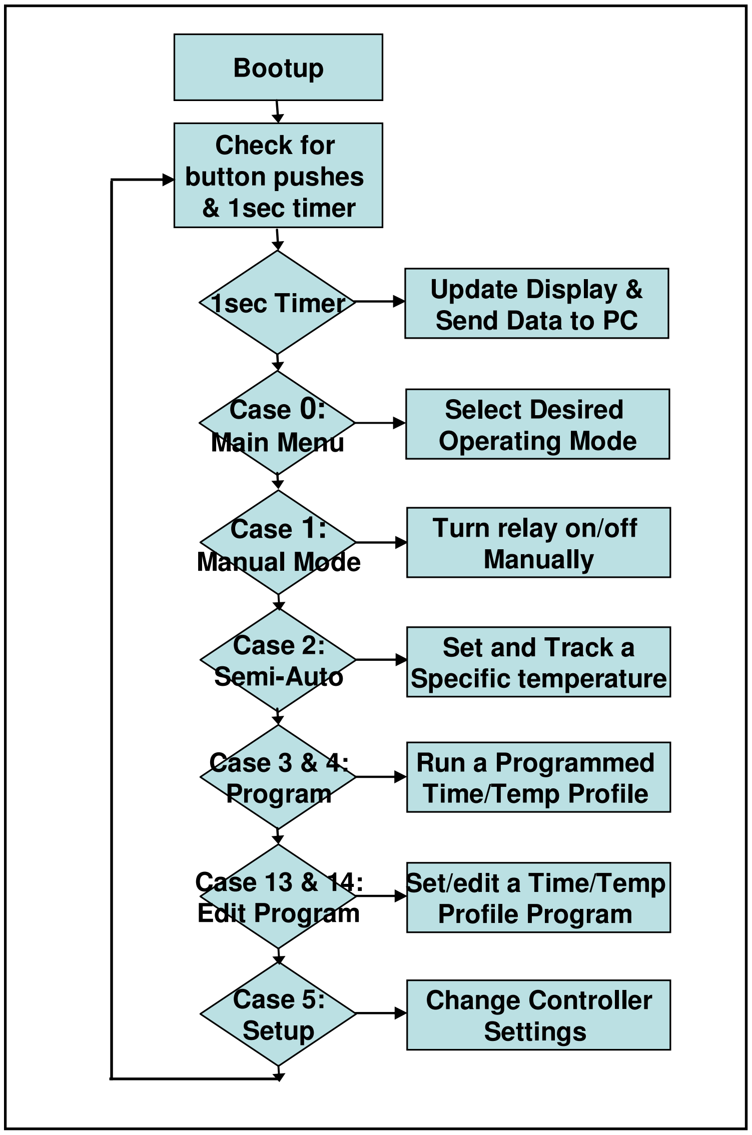
The SourceBoostC Compiler was used to write the program. Some of the major differences between this compiler and the more widely used CC5X compiler which are relevant to this program:
1. PIC register names are not capitalized. Also, individual bit names are not used but referred to by their bit number instead (e.g. t1con.1 is bit #2 in the t1con register (bit numbers start at 0))
2. Binary numbers are written as 0b10010101 for example rather than 0b.1001.0101.
3. "#pragma origin" is not used in BoostSourceC. Instead, a linker command moves the whole program as one block. This command is not needed for this program since it is loaded directly into memory by the hardware programmer.

The reader is encouraged to obtain the Datasheet for the PIC 16F88 before looking at the code because so many essential aspects of the code are tied directly to the MPU's operational design.

The overall program structure is shown in the diagram in Figure 12. Booting up includes setting up the timer interrupt routine, the LCD (in 4-bit communication mode), and user-controlled variables stored in EEPROM memory.

After the splash screen is shown, the program enters the endless loop which starts at the first “while(1)” statement in the Main routine. At the beginning of the loop, each of the 3 push button switches is polled to see if one of them is closed. The logic variables “button_push”, “long_push” (for pushes longer than 2 seconds), and “button_release” are designed to allow one and only one button push to be acted on before any other button can interfere with it. The one that is pushed will be processed all the way through the “while” loop before the program returns to the beginning of the loop where another button can then be activated. All three buttons must be in the released state simultaneously to reset the button_push indicator.

At the end of the button detecting section is the master logic for updating the time and temperature on the display. The PIC hardware counts to 0.1 seconds and then generates an interrupt which causes the variable "m_seconds" to be incremented. After 10 increments, the interrupt routine updates the 1 second counter variable "total_seconds" which is the basis for the master clock for elapsed time. It is also used as the trigger to cause display updates as well as send data to the PC because the time and temperature are


Bootup
Check for button pushes & 1sec timer
1sec Timer
Update Display & Send Data to PC
Case 0: Main Menu
Select Desired Operating Mode
Case 1: Manual Mode
Turn relay on/off Manually
Case 2: Semi-Auto
Set and Track a Specific temperature
Case 3 & 4: Program
Run a Programmed Time/Temp Profile
Case 13 & 14: Edit Program
Set/edit a Time/Temp Profile Program
Case 5: Setup
Change Controller Settings

only updated once each second in order to keep processor overhead and data recording reasonable. In addition, having the temperature updating more often becomes very distracting since the thermocouple produces quite a bit of noise which will cause the display to constantly be changing value.
The rest of the program is structured around each mode of operation, indicated by the

Figure 12 – Serial Output Excel Plot

"state" variable used in the "case" statements. The modes are described above and their corresponding case number is in the diagram. Each case is structured in a similar way; the first section is executed only when that case is first entered. The "prev_state" variable keeps track of this condition.

After that, each case generally has prescribed actions which depend on the particular button which was pushed, as determined by the "choice" variable. The "state" variable is only changed after entering or exiting a new mode.

Due to the very limited available memory in the PIC of 4k instructions, every bit of code that was even slightly redundant has been packaged into a separate subroutine to save space. This sometimes makes the code hard to read but it had to be done. Thus the routine "send_cmd" is used for sending both data and commands to the LCD.

Compiling requires the use just two additional header files, both of which are included in the download. The 16F88.h header file provided with the source code has been somewhat expanded from the original that comes with BoostSourceC to include more register names for easier referencing in the program. It must be used when compiling the OvenFlow source code. The "boostc.h" file is unchanged.

Once compiled, the ".hex" file can be burned into the PIC and you're ready to go. Just note that there are only 198 command words left in memory out of 4096 so you can't add many new features.