

Thermometer Application Manual

(quick and dirty)

This is an excerpt from the full program documentation. It covers only the features of the Thermometer.EXE program, The StripSemicolonLines.exe utility, Commands that are implemented on the ATmega328/Atmega168 and the Temperature reporting protocol. It is intended for use by someone who has received Arduino with a attached temperature sensor and the ThermometerOne software loaded onto the Arduino device. The full document is available from <http://www.keywild.com>.

ArduinoThermometer.exe

ArduinoThermometer is a set of programs written in FreeBasic to run in a console window under Windows or Linux and capture the data from the Arduino Thermometer application. The main program is a customized version of the Arduino Receiver program. The program supports all the standard options for the Receiver program (see *Arduino_Thermometer.ini*) plus it has been modified so that a number of single character keystroke can be used to control the Arduino:

- Keys '1' to '0' set report Times
- Key 'A' toggles AVR mode
- Key 'B' Restore from Backup
- Key 'C' toggles Celsius mode
- Key 'D' toggles Debug mode
- Key 'E' toggles EEPROM mode
- Key 'F' toggles Fahrenheit mode
- Key 'L' lists AVR commands
- Key 'M' turns on Minimal mode (Fahrenheit only)
- Key 'Q' prints AVR storage
- Key 'R' toggles Rounding mode
- Key 'S' prints AVR Status
- Key 'V' toggles Raw Reading mode
- Key '>' increase Degree Offset by 0.25 Fahrenheit
- Key '<' decrease Degree Offset by 0.25 Fahrenheit

Noticeably missing are any commands to change the calibration or write new parameters to the EEPROM. That was intentional because this program is intended to be used to capture the data not to calibrate the device. However there is a way around that. The program allows the user to define five "user" strings via the ".ini" file. These commands are sent by pressing the "U" key followed immediately by a numeric key "1", "2", "3", "4" or "5". There is a one second timeout for the second key. As provide the first three of these are defined as follows:

- Key 'U1' INI defined string: "Z1", write test data set 1 to EEPROM
- Key 'U2' INI defined string: "Z2", write test data set 2 to EEPROM
- Key 'U3' INI defined string: "ZD", Hex/ASCII dump of EEPROM

Because the Thermometer application allows the use of a space as a delimiter a series of commands may be included in a single user defined string. That is the way that the 'M' keystroke works. It sends the command string "RF CF IF FT ST".

```
Command Prompt - Arduino_Thermometer.exe

>Arduino_Thermometer.exe
-----
Report:      True
Debug:       False
Raw:         True
Fahrenheit:  True
Celsius:     True
Avr:         False
Round:       True
Minutes:     1
Voltage:     1.0699
Sensor ID:   SainSmartNano328
Offset:      0.0000
-----

===Arduino_Thermometer.exe Ver1.0.5 (8 October 2013) ===
Using INI file:
Arduino_Thermometer.ini
Using Port Str: COM12:9600,N,8,1,CD,CS,DS,OP,BIN
Appending date/time to data.
Logging data to: Thermometer_0001.LOG
Delimiter: Tab
EOL: carriage return plus line feed
Start: 10-07-2013 05:38:59
Press '?' or '/' to redisplay this message
Press 'Escape' key to exit.

Keys '1' to '0' set report Times
Key 'A' toggles AVR mode
Key 'B' Restore from Backup
Key 'C' toggles Celsius mode
Key 'D' toggles Debug mode
Key 'E' toggles EEPROM mode
Key 'F' toggles Fahrenheit mode
Key 'L' lists AVR commands
Key 'M' turns on Minimal mode (Fahrenheit only)
Key 'Q' prints AVR storage
Key 'R' toggles Rounding mode
Key 'S' prints AVR Status
Key 'U' toggles Raw Reading mode
Key '>' increase Degree Offset by 0.25 Fahrenheit
Key '<' decrease Degree Offset by 0.25 Fahrenheit
Key 'U1' INI defined string: "Z1"
Key 'U2' INI defined string: "Z2"
Key 'U3' INI defined string: "ZD"

2013-10-07 05:40:01 0703 23.00 73.50
2013-10-07 05:41:01 0703 23.00 73.50
2013-10-07 05:42:01 0703 23.00 73.50
2013-10-07 05:43:01 0703 23.00 73.50
```

Strip Semicolon Lines Utility

The main program writes everything it receives from the device to a tab demitted text file specified in the “.ini” file. Every line that Arduino application sends that is not an actual report line is prefixed with a semicolon. **StripSemicolonLines.exe** is a utility used to separate or extract the report lines. It was written with a number of options including the ability to mark a point in the input file where it last processed the lines. Running the program with a “?” as the parameter will print out the options.

Syntax: StripSemicolonLines.exe **file1 file2 file3** [options]

file1 = input filename

file2 = output filename

file3 = output filename with stripped lines (optional)

Options:

/O = Overwrite any existing output file

/A = Append to any existing output file (overrides /O)

/D = Delete input file

/R = Retain blank lines

/S = Split lines at semicolon and delete trailing portion

/X = Deletes all lines with semicolon regardless of location

/M = Mark end of file with “;;--PROCESSED--;;”

/E = Execute program with output file

/V = Verbose prints statistics before exiting

/? = display help and exit

The “**file1**” is the input file that was produced by the main program. “**file2**” is the output file where you want the report lines written. Both of these file names are required and may include a full or relative path specification. “**file3**” is optional. If included the lines that are stripped from the input file will be written to this file. The “**overwrite**” or “**append**” option tells the program what to do if you specify a filename for a file that already exists. If you do not specify an option and the file exists then the program aborts. If the “**append**” option is specified then the “**overwrite**” option is ignored. The “**delete**” option can be used to delete the original file after it is processed. Normally the program skips all blank lines however you can use the “**Retain**” option to keep them (*why you want to I have no idea*). The “**Split**” option divides any lines that have a semicolon somewhere other than the first character. It writes the first part to “**file2**” and the full line to “**file3**”. The “**X**” option has the opposite effect. It deletes any lines with semicolons anywhere in the line. The “**Mark**” option writes the line “;;--PROCESSED--;;” to the end of the input file after it has processed it. When the program is run with the “**Mark**” it reads the entire input file looking for the last occurrence of this line. It then begins processing at the next line. If the line is not found then it begins processing at the beginning of the file. This is useful for extracting data from an active log file. The “**Execute**” will pass the output file to a program such as a spreadsheet or charting program. The program name must include the full path and should be enclosed in quotes (*due to spaces in the path or file name*). The “**Verbose**” option prints the number of input lines, output lines, blank lines and semicolon lines before the program exits.

Although all the options are shown with forward slashes “/” the dash “-” can be used as well. The options may be in in order or case. These are some examples of valid command lines.

```
StripSemicolonLines.exe Thermometer.LOG work.txt
```

```
StripSemicolonLines.exe Thermometer.LOG work.txt dump.txt
```

```
StripSemicolonLines.exe Thermometer.LOG work.txt dump.txt /X /O /D /V
```

```
StripSemicolonLines.exe Thermometer.LOG work.txt dump.txt -o -D -X -V
```

```
StripSemicolonLines.exe Thermometer.LOG work.txt -O /x -V /M /d
```

```
StripSemicolonLines.exe Thermometer.LOG work.txt /E:"c:\program Files\suite\sheet.exe"
```

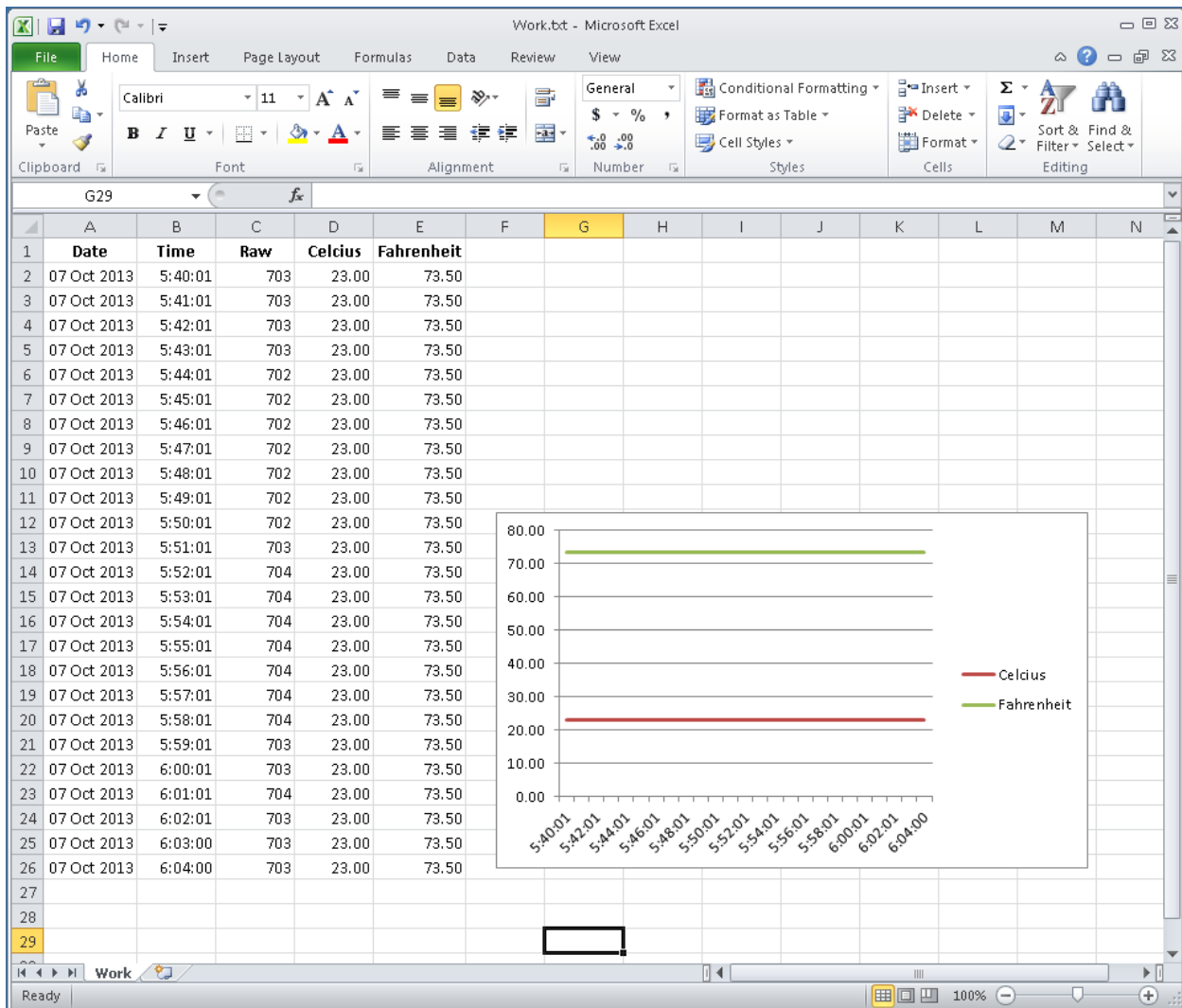
```
C:\Windows\system32\cmd.exe

>cd C:\bin\FreeBASIC\projects\Ardunio_Thermometer

>StripSemicolonLines.exe Thermometer_0001.LOG Work.txt Dump.txt /A /X /U /M /E:"
C:\Program Files (x86)\Microsoft Office\Office14\excel.exe"

    Total Lines: 38
Semicolon Lines: 13
    Output Lines: 25

>pause
Press any key to continue . . .
```



Implemented Commands

Arduino Thermometer One Application		
AtMega328	Command	AtMega168
Output ID string	ID	Output ID string
Output Status	ST	Output Status
Raw=True	RT	Raw=True
Raw=False	RF	Raw=False
Fahrenheit=True	FT	Fahrenheit=True
Fahrenheit=False	FF	Fahrenheit=False
Enter Current Fahrenheit	F=	Enter Current Fahrenheit
Celsius=True	CT	Celsius=True
Celsius=False	CF	Celsius=False
AVR Internal Temperature=True	IT	
AVR Internal Temperature=False	IF	
Enter Current Celsius	C=	
New Degree Offset (Fahrenheit)	DO	New Degree Offset (Fahrenheit)
Same as DO	DF	
New Reference Voltage	RV	New Reference Voltage
Report time = 01 minutes	T1	Report time = 01 minutes
Report time = 02 minutes	T2	Report time = 02 minutes
Report time = 03 minutes	T3	Report time = 03 minutes
Report time = 04 minutes	T4	Report time = 04 minutes
Report time = 05 minutes	T5	Report time = 05 minutes
Report time = 10 minutes	T6	Report time = 10 minutes
Report time = 15 minutes	T7	Report time = 15 minutes
Report time = 20 minutes	T8	Report time = 20 minutes
Report time = 30 minutes	T9	Report time = 30 minutes
Report time = 60 minutes	T0	Report time = 60 minutes
Report time = 02 hours	TA	
Report time = 04 hours	TB	
Report time = 06 hours	TC	
Report time = 08 hours	TD	
Report time = 12 hours	TE	
Report time = 24 hours	TF	
Print mode = False	PF	Print mode = False
Print mode = True	PT	Print mode = True
Debug mode toggle	DB	Debug mode toggle
Rounding mode toggle	00	Rounding mode toggle
New Location	L:	New Location
Write Calibration data to EEPROM	WW	Write Calibration data to EEPROM

Overwrite Backup Calibration data	W+	Overwrite Backup Calibration data
Restore from Backup Calibration data	W-	Restore from Backup Calibration data
Set Flag to send next run to EEPROM	E+	Set Flag to send next run to EEPROM
Clear Flag to send next run to EEPROM	E-	Clear Flag to send next run to EEPROM
Clear EEPROM Storage	EC	Clear EEPROM Storage
Dump data stored in EEPROM	ED	Dump data stored in EEPROM
List implemented commands	LL	List implemented commands
List implemented commands	??	List implemented commands
Shutdown (send twice)	SS	Shutdown (send twice)
Reset (send twice)	!!	Reset (send twice)
Write test data 1	Z1	Write test data 1
Write test data 2	Z2	
Toggle 5 Second reporting	ZZ	
Dump ALLL EEPROM to serial	ZD	
Response 'XX' = not implemented		Response 'XX' = not implemented
Response '??' = not recognized		Response '??' = not recognized

Protocol

Linear Calibrated Temperature Sensor(s) Reporting Protocol

Established: September 2013 by Lewis Balentine

This Protocol is designated to be Public Domain

- A. Default communications will be via RS232 protocol at 9600 Baud
- B. All communications will be done in ASCII 7 bit characters
- C. The device will monitor the serial port for commands as specified below
- D. All commands will be Two Characters of which the first must be an Alpha character
- E. Command terminations/separators may be either a carriage return (ASCII 13) character or a new line (ASCII 10) character or null character (ASCII 0) or a tab character (ASCII 9) or a space character (ASCII 32) or any combination of the these characters
- F. Only one command is accepted at a time but additional data may be sent as required by the command. This data shall be delimited from the command by a command terminations/separator. When that data is an ASCII string then the space character (ASCII 32) is excluded from the list of valid terminations/separators within the length of the string.
- G. The following two character commands will be considered valid
 - 1. **ID** Output sensor/location ID string(s)
For multiple sensors each ID string will be proceeded by a designation digit/character, a colon and a space
 - 2. **ST** Output Status (*as applicable to implementation*)
 - a. Reporting mode, true or false
 - b. Debug mode active, true or false
 - c. Report Raw reading, true or false
 - d. Report Fahrenheit temperature, true or false
 - e. Report Celsius temperature, true or false
 - f. Internal Temperature, true or false
 - g. Minutes between readings
 - h. Reference voltage
 - i. Sensor Parameters
Repeat the ID, Offset, Fahrenheit, Celsius constants as applicable for each sensor.
For multiple sensors each ID string will be preceded by a designation numeral, a colon and a space.
 - l. If any current in memory constants have not been written to storage then include line that to that effect.
 - m. Report storage Mode flag set if it is set
 - 3. **RT** Raw True = include raw reading from temperature sensor
 - 4. **RF** Raw False = do not include raw reading from temperature sensor
 - 5. **RV** New reference voltage
 - 6. **CT** Celsius True = include degrees Celsius
 - 7. **CF** Celsius False = do not include degrees Celsius
 - 8. **C=** Celsius input. Recalculate raw reading offset based on input temperature.
Device with multiple sensors must be placed in "Single Sensor" mode.
 - 9. **FT** Fahrenheit True = include degrees Fahrenheit
 - 10. **FF** Fahrenheit True = do not include degrees Fahrenheit
 - 11. **F=** Fahrenheit input. Recalculate raw reading offset based on input temperature.
Device with multiple sensors must be placed in "Single Sensor" mode.
 - 12. **T#** Sets time between report lines where # is one of the following
 - a. **1** Report reading every 01 minute
 - b. **2** Report reading every 02 minutes

- c. **3** Report reading every 03 minutes
- d. **4** Report reading every 04 minutes
- e. **5** Report reading every 05 minutes
- f. **6** Report reading every 10 minutes
- g. **7** Report reading every 15 minutes
- h. **8** Report reading every 20 minutes
- i. **9** Report reading every 30 minutes
- j. **0** Report reading every 60 minutes
- k. **A** Report reading every 2 hours
- l. **B** Report reading every 4 hours
- m. **C** Report reading every 6 hours
- n. **D** Report reading every 8 hours
- o. **E** Report reading every 12 hours
- p. **F** Report reading every 24 hours

(it is intended that the data is the average temperature for the given period)

- 13. **TT** Followed by data (*units, number*) for other reporting period (*not implemented*)
 - 14. **PF** Stop printing report lines and accept commands only (report printing mode/state)
 - 15. **PT** Resume printing report lines (report printing mode/state)
 - 16. **DB** Toggle debug mode for extended reporting
 - a. Average time for each read cycle
 - b. Number of reads cycles for each report line
 - c. Actual time for each report Line
 - d. Other information according to implementation
 - 17. **DO** New Degree offset for minor adjustment to temperature scale
 - 18. **DF** New Degree offset for minor adjustment to temperature scale (*Fahrenheit*)
 - 19. **DC** New Degree offset for minor adjustment to temperature scale (*Celsius*)
 - 20. **S:** plus command separator plus sensor designator. This command is **ONLY** used for devices with multiple sensors. The designator shall be a single alpha or digit character as determined by the implementation. This command selects the sensor for all following sensor specific commands until another "S:" command is received. This command essentially places the device in "single sensor" mode. To exit this mode enter the "S:" command without a designator.
 - 21. **L:** plus command separator plus new ID/Location string
 - 22. **O:** plus command separator plus new Raw reading offset (*capital O colon*)
 - 23. **C:** plus command separator plus new Celsius scale factor
 - 24. **F:** plus command separator plus new Fahrenheit scale factor
 - 25. **A:** Extended protocol (*zero, colon*).
- These commands are specific to a given specific implementation.
- 26. **WW** Write new constants to device storage
"WW" MUST be upper case!
 This command (**WW:**) shall only update the working copy of the constant data
 - 27. **W+** Overwrite backup constant data with working constant data. "W" MUST be upper case!
 - 28. **W-** Overwrite working constant data with backup constant data. "W" MUST be upper case!
 - 29. **E+** This is a special mode that writes the readings to EEPROM rather than to the serial port
(This will of course require an alternate power source)
 - a. Set device storage flag
 - b. On the next "Reset" or "Startup"
 - 1. Clear device storage flag
 - 2. Read and store RawReading to device storage until space is exhausted
 - 3. Shutdown, Sleep or Resume normal operation as available in implementation

- 30. **E-** Clears Flag for EEPROM mode
- 31. **ED** Dumps data from device storage according to current conversion constants
(*debug mode ignored and all three values are output*)
- 32. **EC** Clears device storage area (if EEPROM writes 0xFF to all locations)
- 33. **AA** Extended protocol .
These commands are specific to a given specific implementation.
- 34. **M?** Undefined, reserved for future use by this protocol specification.
- 35. **N?** Undefined, reserved for future use by this protocol specification.
- 36. **U?** Undefined, reserved for future use by this protocol specification.
- 37. **X?** Undefined, reserved for future use by this protocol specification.
- 38. **Y?** Device specific command(s) (*implementation specific*).
- 39. **Z?** Device specific command(s) (*implementation specific*).
- 40. **LL** Output list of device implemented commands
Each line shall be prefixed with semicolon and space
The first line shall include device Identification and/or serial number
The required output is the 2 character commands
Optionally each line may include a short description
- 41. **??** Same as **LL**
- 42. **SS** Shutdown or Sleep (*implementation specific*).
This command MUST have two consecutive calls.
The device will respond with “; **SHUTDOWN**” or “; **SLEEPING**” as applicable.
- 43. **00** Turn rounding on or off (*implementation dependent*)
- 44. **!!** Reset or reboot device (*that is two exclamation marks*)
This command MUST have two consecutive calls.
The device will respond with “; **RESETTING**” (*implementation limited*)

H. The device/application may implement any set or subset of the commands that include the following commands: **ST, CT, CF, FF, FT, T1, T2, T3, T4, T5, ??**

- I.** Any response line from the device that is NOT a report line shall be prefixed with a semicolon “;” and a space.
- J.** Valid commands that do not otherwise generate responses shall respond with the two character command plus space plus “OK”.
- K.** If the device receives a command it does not recognize then it will respond with “??”.
- L.** If the device receives a command it recognizes but is not implemented then it may respond with either “XX” or “??” but “XX” is preferred.
- M.** Commands with a terminating colon may be used for multiple sensors by replacing the colon with a numeral to identify the sensor number.
- N.** Report lines from the device shall consist of the designated data fields separated by a tab character (ASCII 09) in the following order:
 - 1. Raw reading
 - 2. Calibration corrected Celsius temperature
 - 3. Calibration corrected Fahrenheit temperature
 - 4. Extended debugging data as defined above

That should be enough to confuse the issue. Our device currently only has one sensor but the protocol makes provisions for multiple sensors (*Engineering is the art of “Planning and Forethought”*). Tab characters (ASCII 09) are one of the commonly used delimiters for text files. This makes it easy to import the data file into a spreadsheet program or database for charting and/or analysis. The semicolons prefixed to the devices responses may it easy to strip out those lines from the data file or signal the receiver application that this is NOT a normal reporting line. If the PC application includes device commands in its output stream and/or log then it should prefix these with a semicolon and a space as well. Although the protocol specifies all upper case characters for

command characters it is recommended that the device application accepts either upper or lower case or a combination of both with the exception “W” commands. The reset command “!!” (*that is two exclamation marks*) is intended to be used for “If all else fails then abort and start over”. The reset command may also be used as an entry point to update the device software (*depending on the reset characteristics of the device*).

The commands “A:”, “A?”, “X?” and “Z?” (*the “?” is wild card that is to be interpreted as any character*) are intended to be used to extend the protocol as may be required for a specific sensor(s) while keeping the functionality of the basic protocol in place. This allows for a standard reporting application to use sensor(s) with extended capabilities. However an extended reporting application specific to the implementation may be created to take advantage of the additional features (*for example “wet” and “dry” bulbs or a humidity sensor*). Any command beginning with the letter “M”, “N”, “U” or “X” is defined to be undefined and reserved for future use by this specification.