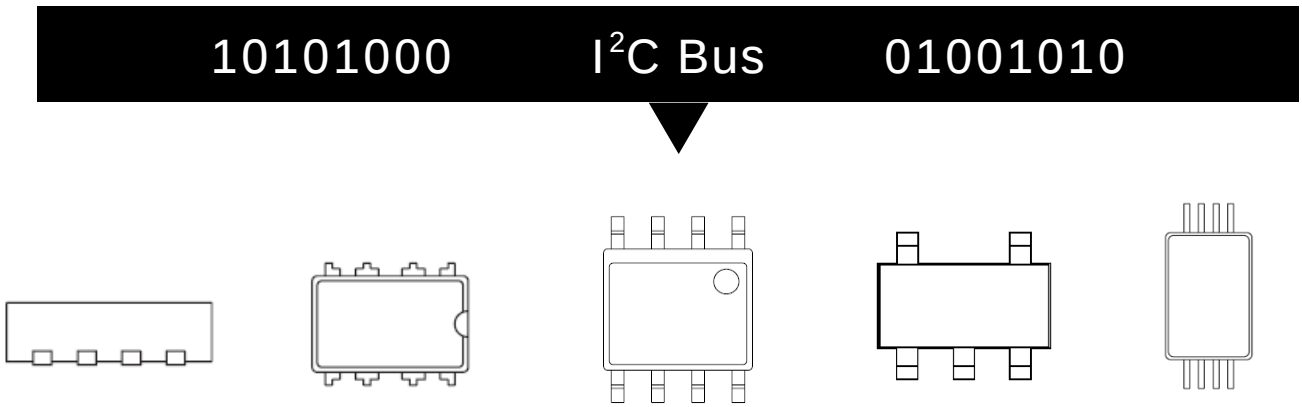
User's Guide

iBURNER™

I²C Bus Serial EEPROM Programmer Software

for MCC ASCII Interface
I²C Bus Host Adapters

Version 3.0



MCC
Micro Computer Control
Small Area Network Specialists

www.mcc-us.com

Introduction

The MCC iBurner™ I²C Bus Serial EEPROM Programmer Software provides a quick and easy way to program, read, and verify a wide variety of I²C Bus EEPROMS.

iBurner is compatible with Windows XP, Vista, 7, or higher, running .NET Version 2 or above. iBurner is also compatible with MCC ASCII interface based I²C Bus host adapters including the iPort/AI (#MIIC-202), iPort/AFM (#MIIC-203), iPort/USB (#MIIC-204), and i2cStick (#MIIC-207).

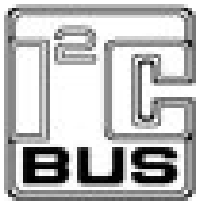
This user's guide describes the operation of the iBurner software.

Are you new to I²C? Want to know more? We suggest you review "What is I²C?" at www.mcc-us.com/I2CBusTechnicalOverview.pdf.

MCC products use NXP(Philips) components and are licensed to use the I²C Bus.

"Purchase of NXP (Philips) I²C components conveys a license under the NXP (Philips') I²C patent to use the components of the I²C system, provided the system conforms to the I²C specifications defined by NXP (Philips)."

I²C is a trademark of NXP (Philips) Corporation.
iBurner is a trademark of Micro Computer Control Corporation



iBurner Revision Report

Release Ver. 3.0

1. Add support for i2cStick I2C Bus adapter.
2. Add iBurner Script Engine exit code (0=Success, -1=Fail).
3. Make slave address spin box up arrow increment slave address.
4. Revise GUI and Script Creator user interface.
5. Add Beep operation to Script Creator.
6. Add Script Engine Automation Examples.

Release Ver. 2.2

1. Enhance Script Engine validation, error handling, and logging.
2. Abort script processing immediately on command error detection. Quit command accepted but no longer required.
3. Correct Script Creator “Load” command offset generation.
4. Add script “*beep” command (beep computer speaker).
5. Add script “*settings” command (log current device settings).
6. Optimize Script Creator script generation.
7. Set initial Load/Save Buffer and Script folder.

Release Ver. 2.1

1. Correct EEPROM access algorithm.

Release Ver. 2.0

1. Easy-to-use scripting engine with Script Builder tool.
2. Intelligent iPort configuration determines optimum settings.
3. New checksum (CRC16) capability.
4. More I2C EEPROM device definitions.
5. More robust programming and logging facilities.

Release V1.1

1. Add support for Motorola S-Record file format.
2. Expand device library export function to support the exportation of one or more selected device definitions.

Initial Release V1.0

DISCLAIMER: Micro Computer Control Corporation makes no representations or warranties with respect to the contents hereof and specifically disclaims any implied warranties of merchantability or fitness for any particular purpose. Further, Micro Computer Control Corporation reserves the right to revise the product described in this publication and to make changes from time to time in the content hereof without the obligation to notify any person of such revisions or changes.

WARNING - Life Support Applications: MCC products are not designed for use in life support appliances, devices, or systems where the malfunction of the product can reasonably be expected to result in a personal injury.

WARNING - Radio Frequency Emissions: This equipment can radiate levels of radio frequency energy that may cause interference to communications equipment. Operation of this equipment may cause interference with radio, television, or other communications equipment. The user is responsible for correcting such interference at the expense of the user.

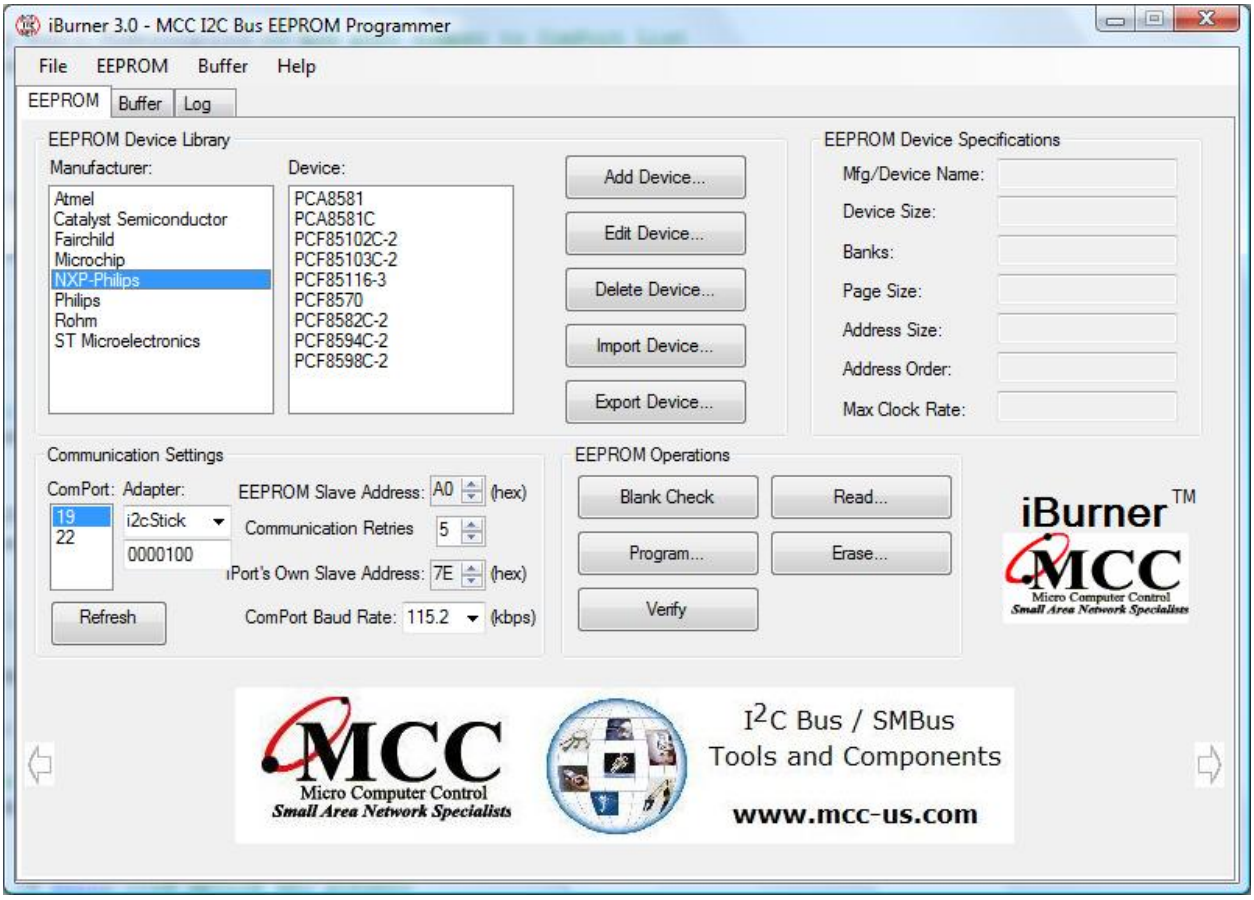
WARNING - Electrostatic Discharge (ESD) Precautions: Any damage caused by Electrostatic Discharge (ESD) through inadequate earth grounding is NOT covered under the warranty of this product. See the “Electrostatic (ESD) Precautions” section of this guide for more information.

Printed in the United States of America

iBurner™ 3.0 User's Guide

Introduction to the iBurner software

iBurner is an EEPROM programmer for I²C Bus based products, developed to interface with Micro Computer Control Corporation's (MCC) iPort line of I²C Bus host adapters. iBurner can blank check, program, read, erase, and verify data stored on most I²C Bus EEPROMs, and includes a robust scripting engine for EEPROM programming automation. iBurner includes a library of EEPROM device definitions for popular devices from major manufacturers, and can be easily updated with new devices. iBurner provides a simple yet powerful tool for working with I²C Bus EEPROMs.



New features for iBurner 3.0

- Add support for i2cStick I²C Bus adapter.
- Add iBurner Script Engine exit code (0=Success, -1=Fail).
- Make slave address spin box up arrow increment slave address.
- Revise GUI and Script Creator user interface.
- Add Beep operation to Script Creator.
- Add Script Engine Automation Examples.

System requirements

In order to use the iBurner software you must satisfy the following system requirements.

1. Microsoft Windows operating system
    - Windows XP
    - Windows Vista
    - Windows 7

2. Microsoft .Net Runtime Environment
    - Version 2 or newer
3. MCC I^2C Bus host adapter
    - iPort/AI (#MIIC-202)
    - iPort/AFM (#MIIC-203)
    - iPort/USB (#MIIC-204)
    - i2cStick (#MIIC-207)

For more information about MCC I^2C Bus host adapters, visit www.mcc-us.com.

Installing and Uninstalling iBurner

iBurner 3.0 is provided as a Microsoft Windows Installer (*.msi) file that can be downloaded from the MCC website: www.mcc-us.com. iBurner is available to all MCC customers who have purchased a compatible MCC I^2C Bus host adapter.

You do not need to uninstall old versions of iBurner before installing a later version. You can uninstall older versions before or after installing later versions. All iBurner installations use separate folders, and any folders or files added to installation folders remain after an uninstall.

To install iBurner, download the iBurner setup file (iBurner3_0Setup.msi) and double-click to install. iBurner requires the Microsoft .Net 2.0 or above framework and Windows Installer. Both are included with many Windows versions, or can be downloaded via Windows Updater. If your PC already has Microsoft .NET Framework installed, it is visible in the Windows Control Panel, Add/Remover Programs or Programs and Features list.

iBurner can be started from the Start menu, under the “iBurner” folder. Also provided are shortcuts for the iBurner Script Creator tool, user’s manual (this file), EEPROM Programming Hardware Setup diagram, and the End User License Agreement (EULA). Script Engine Automation Examples are available on the iBurner Help menu.

Uninstall iBurner using the Windows Control Panel, Add/Remover Programs or Programs and Features list.

I²C EEPROMs

I²C Bus EEPROMs are serial EEPROMs that can be programmed or read over an I²C bus. The EEPROMs are I²C slave devices that implement a communication protocol through which they can be accessed. In addition to the total size of the EEPROM, there are several important parameters that you should be familiar with.

The **page size** of an EEPROM is the number of bytes that can be programmed or read at a single time. This is usually a multiple of 16. After a page of data is transmitted to the device the EEPROM will commit the data to memory; thus the larger the page size the faster the chip can be programmed.

The **addressing bytes** of the EEPROM are the bytes that must be transmitted to select a particular location to read or write. Small EEPROMs require a single byte while larger devices require two. If two addressing bytes are required, the **byte ordering** (most-significant-byte to least-significant-byte or vice versa) must be known. The most common I²C Bus EEPROMs from major manufacturers use MSB-to-LSB ordering.

Some EEPROMs allow for larger memories without resorting to a second address byte; this is accomplished through **banks**. A bank is a slave address at which the EEPROM will access a particular section of its memory. For example, a 512 byte EEPROM would typically require two addressing bytes, as a single addressing byte can only index 256 bytes of memory. Such an EEPROM might instead use two banks, or two I²C slave addresses at which it will respond; the first would access the lower 256 bytes, and the second would access the upper 256 bytes.

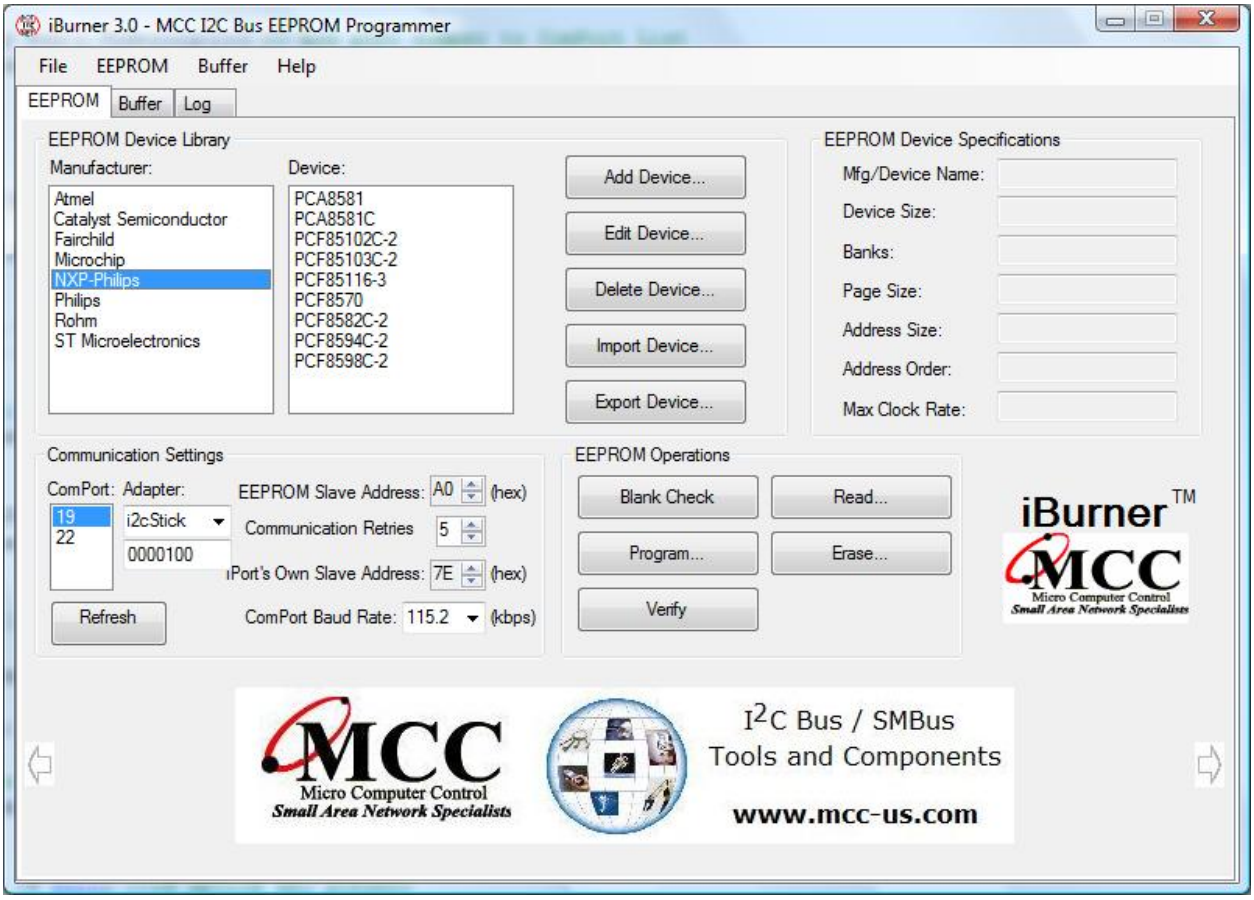
The I²C address of the first bank (the only bank for many EEPROMs) is referred to as the **slave address** of the device. On some EEPROMs, external pins are available to change this slave address. All EEPROM slave addresses are between A0 and AE, inclusive (and like all I²C Bus addresses, include only the even numbers).

iBurner can program most I²C Bus based EEPROMs for which the slave address, total size, number of addressing bytes, byte ordering, and the number of banks is known. The slave address can be easily determined from the configuration of the EEPROM's external pins (if available; otherwise the address is fixed at A0) and the total size is readily available from the device documentation. The number of addressing bytes and banks can be determined from documentation, or from the following table, once the total size is known.

Total size	*Banks*	*Addressing bytes*	*Possible slave addresses (depending on pin configuration)*
16 bytes, 32 bytes, 64 bytes, 1 kbit, 2 kbit	1	1	A0, A2, A4, A6, A8, AA, AC, AE
4 kbit	2	1	A0, A4, A8, AC
8 kbit	4	1	A0, A8
16 kbit	8	1	A0
32 kbit, 64 kbit, 128 kbit, 256 kbit, 512 kbit	1	2	A0, A2, A4, A6, A8, AA, AC, AE
1 Mbit	2	2	A0, A4, A8, AC
2 Mbit	4	2	A0, A8
4 Mbit	8	2	A0

iBurner has a built-in database (to which new devices can be added or removed, and which can be exported for importation to other iBurner installations) with definitions for the most popular devices from major manufacturers.

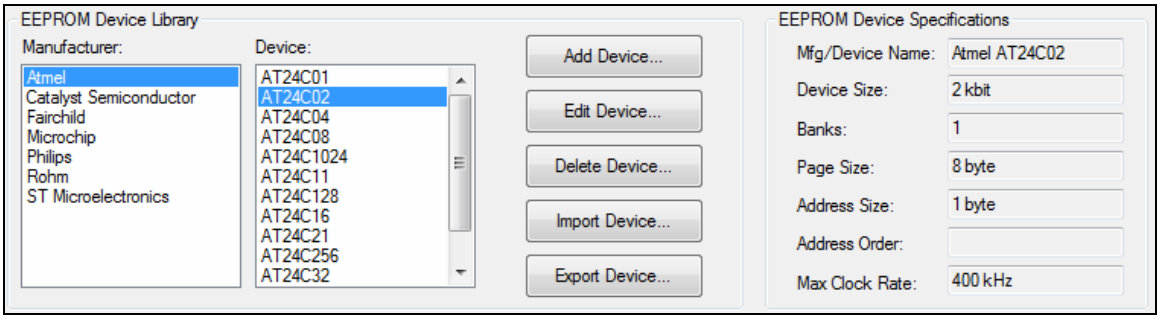
Getting started



When iBurner is first started, you will see a window like that above. There are three tabs; the *EEPROM* tab is used to select an EEPROM and provide iPort configuration settings, as well as accessing some common actions. The *Buffer* tab provides a display of the current memory buffer upon which work is being done. Reading the EEPROM will read data into this buffer, and programming the EEPROM will write data from this buffer. The *Log* tab displays information on current and past operations.

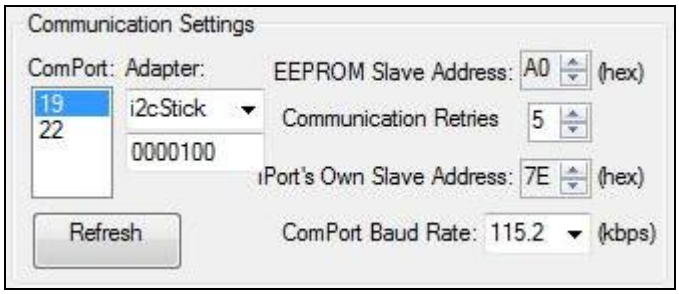
The first thing that should be done is configuration of the device and iPort. Once a device is selected the buffer will become available, and once the iPort is configured you will be able to perform operations.

Selecting a device



Use the *Manufacturer* list to select the manufacturer of your EEPROM. When you choose a manufacturer, the *Device* list will be populated with the available devices produced by that manufacturer. Selecting a device will populate the *EEPROM* fields with the stored device description.

Configuring the iPort



Use the *Communication settings* fields to configure the iPort. If necessary, click the *Refresh* button to have iBurner redetect available serial ports (including virtual serial ports provided by iPort/USB and i2cStick devices). Select the serial port that corresponds to your device.

The *EEPROM Slave Address* can be set to any valid I^2C address, but most manufacturer-produced EEPROMs will have slave addresses between A0 and AE, depending on available addresses and pin configuration (see the section "I^2C EEPROMs").

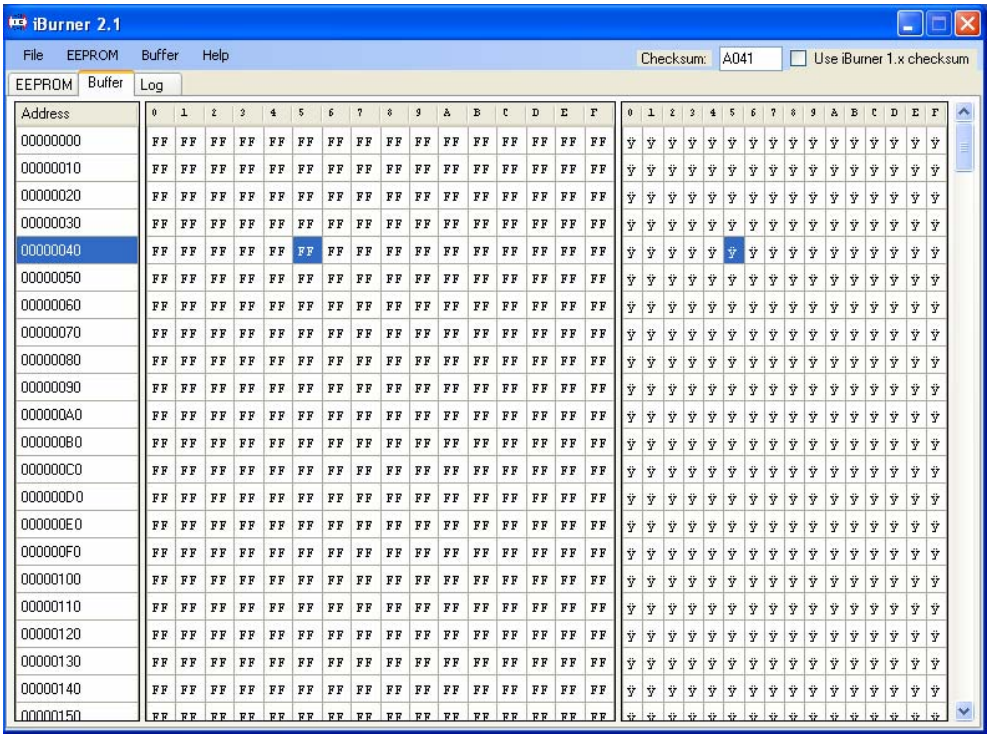
The *Communication Retries* count is used to repeat individual failed reads and writes in the context of an EEPROM operation. The default value of 5 will suffice for most users.

The *iPort's Own Slave Address* can be left at the default 7E for most users; if you are programming an EEPROM on an active bus (a bus on which other I^2C masters and slaves are operating at the same time as iBurner and the target EEPROM) you will want to verify that the iPort slave address is not shared with another device on the bus. You can select any valid I^2C address for the iPort slave address.

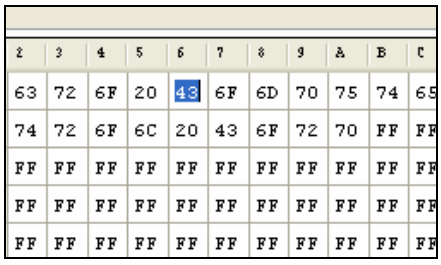
The *ComPort Baud Rate* serial link speed should be left at the maximum value, 115.2 kbps, unless the user desires slower operation. iBurner will automatically adjust this value if the connected iPort cannot support the selected speed.

Once an EEPROM is selected and the desired iPort settings are chosen, buffer and EEPROM operations may be executed.

Using the buffer

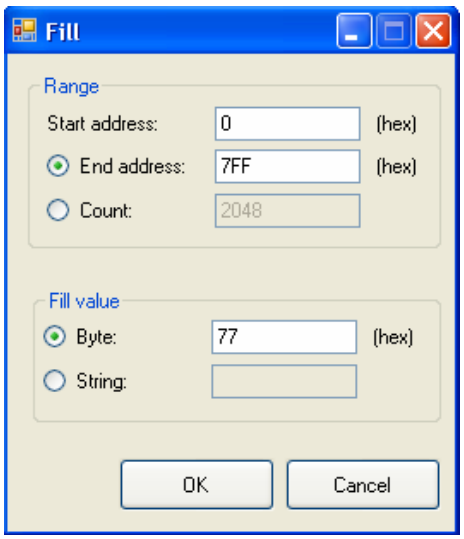


Buffer operations are executed from the *Buffer* tab and the *Buffer* menu. The *Buffer* tab has two main panels; the left panel displays the contents of the buffer in hexadecimal while the right displays the same information as printable ASCII characters (if possible). The contents of the buffer may be edited manually by double-clicking on any cell and entering a new value (either hex in the left panel or ASCII in the right panel).

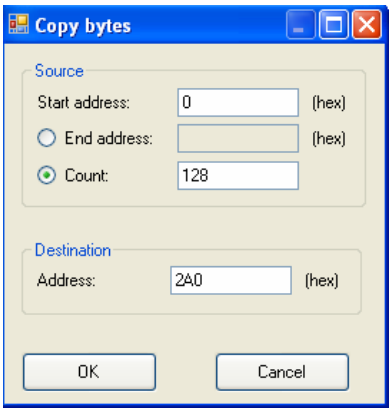


Buffer operations

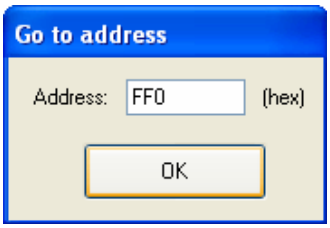
The *Buffer* menu provides three operations: *Fill*, *Copy*, and *Go to address*.



The *Fill* operation fills some portion of the buffer with data; either a repetition of a single byte value or a single instance of an ASCII string. The fill range can be described by a start address and either an end address or a count of bytes to be filled. The start address defaults to the address of the currently selected byte in the buffer and the end address defaults to the last address in the buffer (similarly, the count defaults to the total size of the buffer).

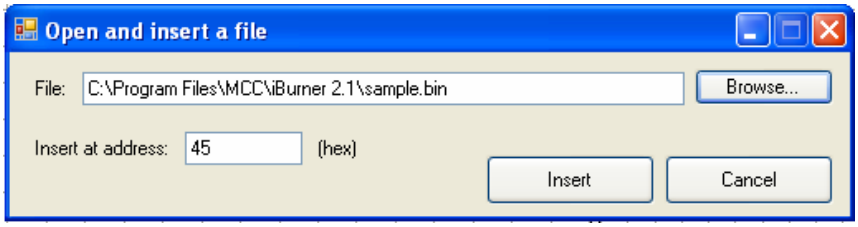


The *Copy* operation copies some portion of the buffer to a different location. The source and destination locations can overlap (eg. bytes 0 through 4 could be copied to bytes 1 through 5, if desired). The start address defaults to the address of the currently selected byte in the buffer; the end address (or count) and the destination address have no default value.

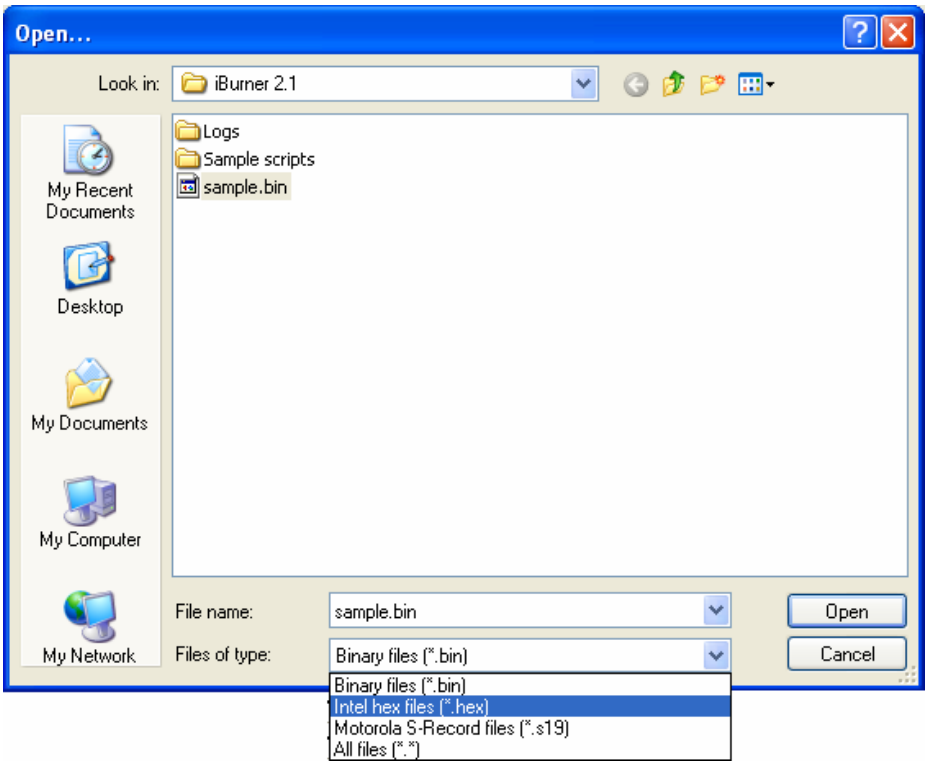


The *Go to address* operation simply moves the buffer view such that the user can see and edit the memory at the desired address.

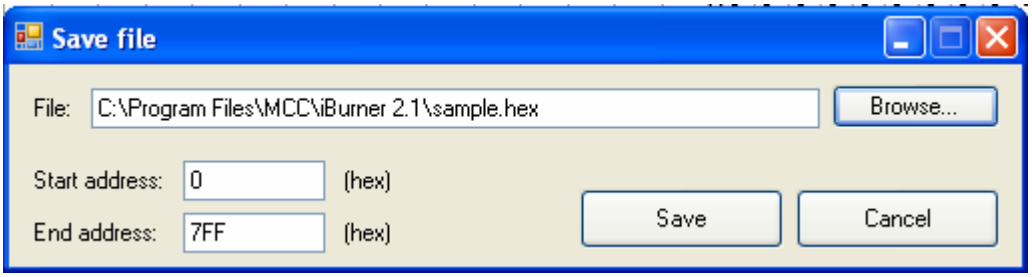
Loading files into the buffer



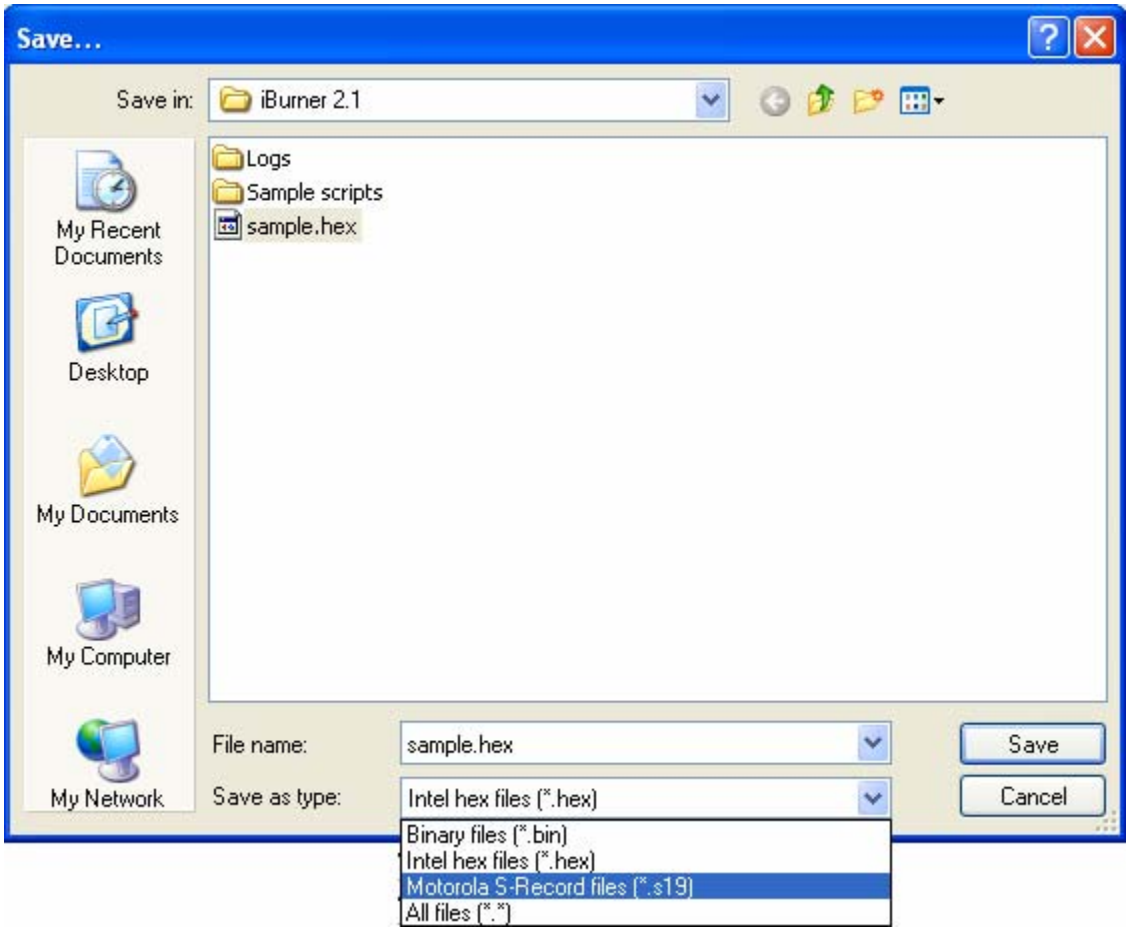
iBurner can load three types of files into the buffer: binary (or flat) files, Intel Hex files, and Motorola S-Record (S19) files. A binary file is any file that the user wishes to load byte-for-byte into the buffer. All files can be loaded with offsets: eg. a 128 byte binary file could be loaded into the buffer starting at address 0x70. The offset defaults to the address of the currently selected byte in the buffer.



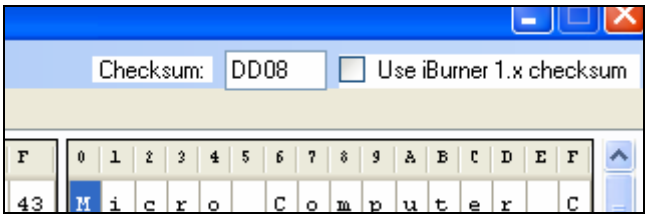
Saving the buffer to a file



The buffer, or some portion thereof, can be saved to a file. As with loading files, binary, Intel Hex, and Motorola S-Record files are supported. The selected range to save defaults to the entire buffer.

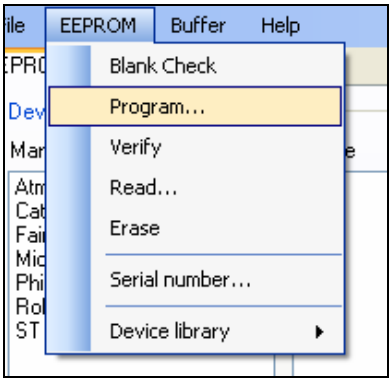


Checksums



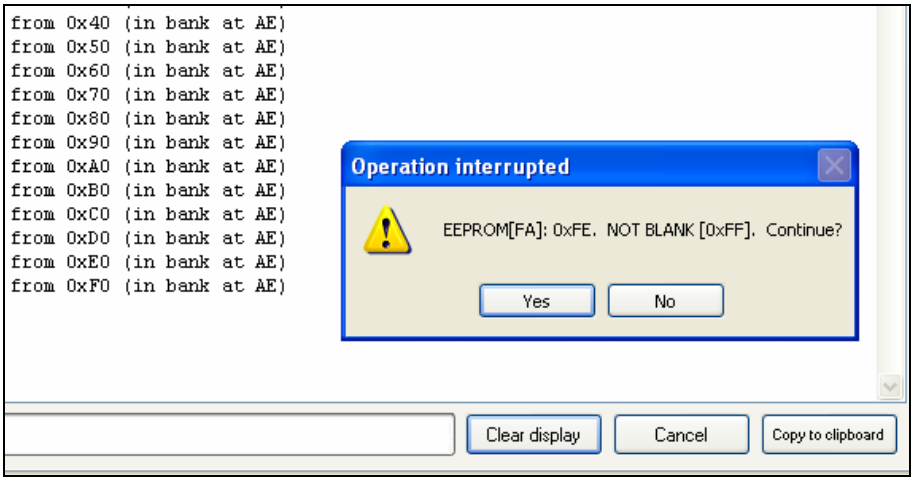
The *Buffer* tab displays the current checksum for the buffer in a text box at the top of the window. This checksum is a standard CRC16 computation (CRC polynomial A001) and can be used to quickly verify the contents of the buffer. To preserve compatibility with older versions of the iBurner software, the "Use iBurner 1.x checksum" checkbox may be checked; this is not recommended for use beyond backwards compatibility – the default CRC16 provides a more common and robust checksum.

Manipulating EEPROMs



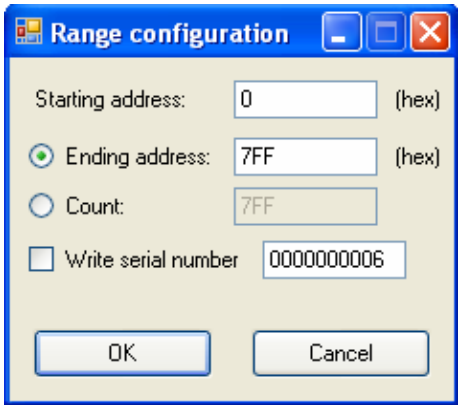
iBurner can perform five separate device operations, as detailed below.

Blank check



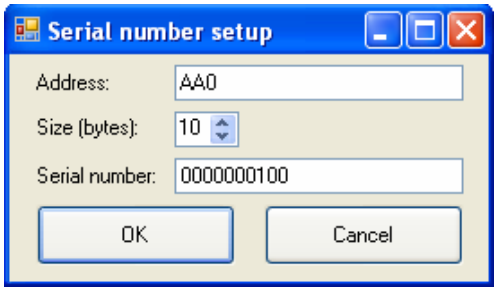
Clicking the *Blank check* button or *Blank check* from the *EEPROM* menu initiates a scan of the connected EEPROM to determine if the device contains any data (non-FF bytes). If such a byte is detected, the user is prompted to stop checking or continue. The information is logged as it is processed, including user decisions.

Program



Clicking the *Program…* button or *Program…* from the *EEPROM* menu initiates a full or partial program of the connected EEPROM with data from the buffer. The configuration window can be used to select the range to be programmed; the range defaults to the entire memory. Before a program operation begins the user is warned that the contents of the EEPROM will be changed and prompted to continue.

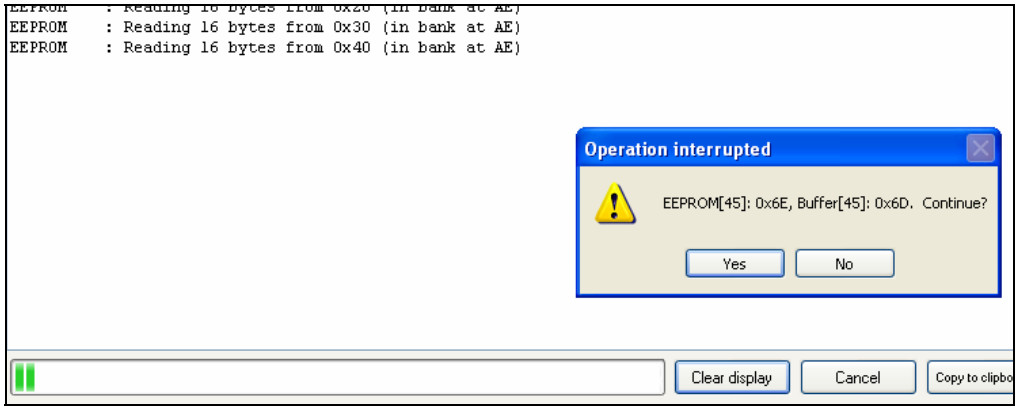
iBurner can insert a serial number string into the buffer prior to programming, and automatically increment this serial number upon successful completion of the program operation. To configure the serial number, click *Serial number…* from the *EEPROM* menu. A window such as that below will be displayed.



The address field is the hexadecimal address in the buffer where the serial number string should be inserted before each program operation. Enter the size of the serial number string, and the current value. You may now click the "Write serial number" checkbox in the program operation configuration dialog; the displayed serial number will be inserted into the buffer. Following the program operation the serial number will be incremented.

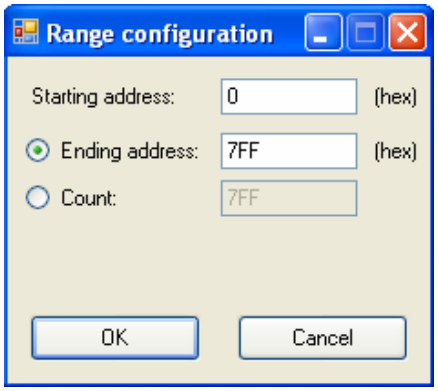
The serial number configuration is maintained between iBurner sessions.

Verify



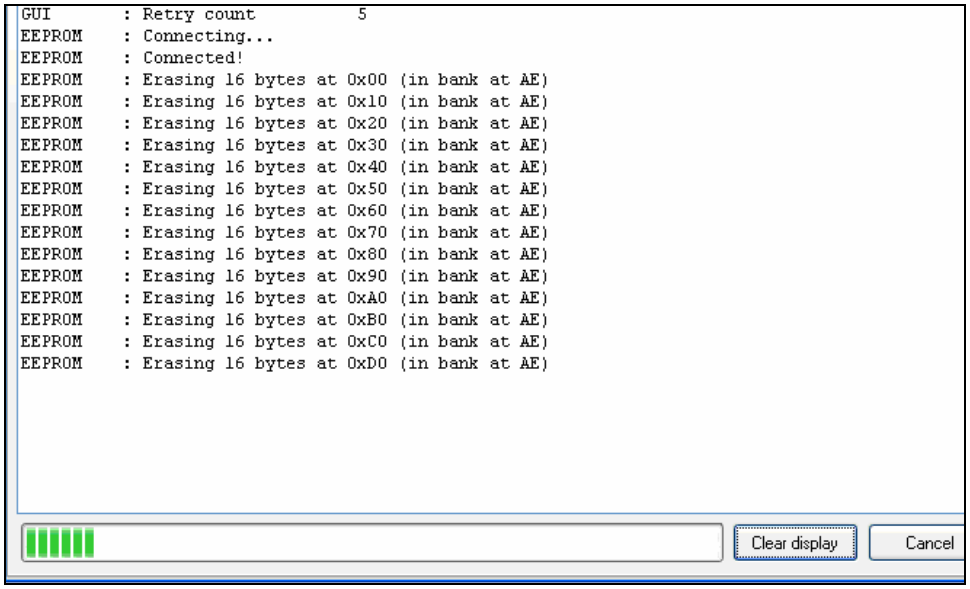
Clicking the *Verify* button or *Verify* from the *EEPROM* menu initiates a comparison between the contents of the EEPROM and the buffer. If any bytes are found to be different, the user is prompted to stop checking or continue. The information is logged as it is processed, including user decisions.

Read



Clicking the *Read…* button or *Read…* from the *EEPROM* menu initiates a full or partial loading of data from the EEPROM into the buffer. The configuration window can be used to select the range to be loaded; the range defaults to the entire memory. Before a read operation begins the user is warned that the contents of the buffer will be changed and prompted to continue.

Erase



Clicking the *Erase* button or *Erase* from the *EEPROM* menu initiates a complete erasure of the EEPROM device by programming it with FF bytes. Before the erase operation begins the user is warned that the contents of the device will be changed and prompted to continue.

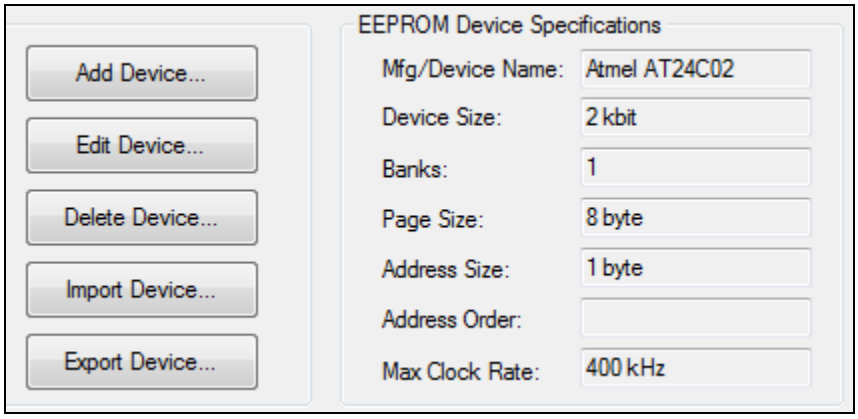
Logs

iBurner has two logging systems. While operations are in progress they write information to the *Log* tab indicating success or failure and progress. The *Log* tab also provides the *Cancel* button, which halts the current operation, and a *Copy to clipboard* button that will place the contents of the log on the clipboard for pasting into another application. The *Clear* button clears the on-screen display.

In addition to the *Log* tab, all iBurner content is logged to files. Files are named with the date and the ".log" extension, as in "yyyymmdd.log". The log file to which data is being stored is indicated in the log itself and can be seen in the on-screen display. To view a log file, click *View log file…* in the *File* menu, and select the desired file. A *Copy to clipboard* option is provided in the resulting window to copy the contents of an entire day's operations.

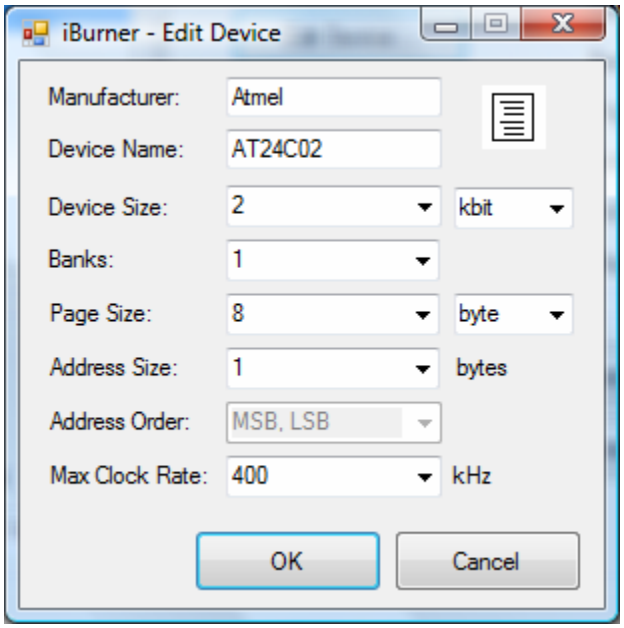
All operations that are processed in iBurner are logged, both those initiated with the GUI and the scripted interface. For more information on the logging of scripts, see the "Automation" section.

The device library



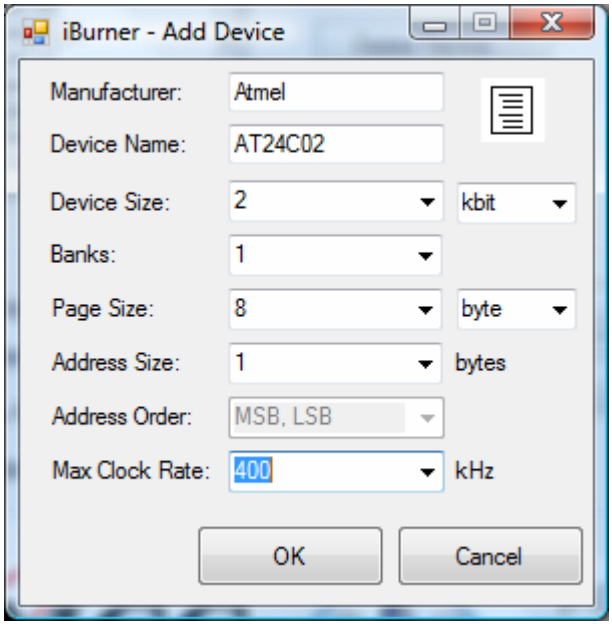
The device library is a collection of EEPROM devices, by manufacturer, for which iBurner contains predefined settings. The device library is loaded when iBurner starts and can be modified in a number of ways.

Editing an existing device



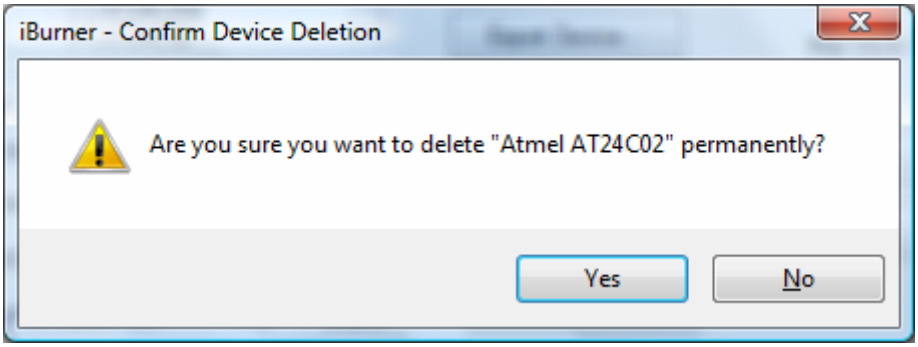
Existing devices can be edited by selecting the manufacturer and device and clicking the *Edit Device…* button, or *Edit device…* from the *Device Library* submenu of the *EEPROM* menu. This is useful for correcting improperly added devices or changing a device definition so it is used differently (with a smaller page size, with a lower bus speed, etc).

Adding a new device



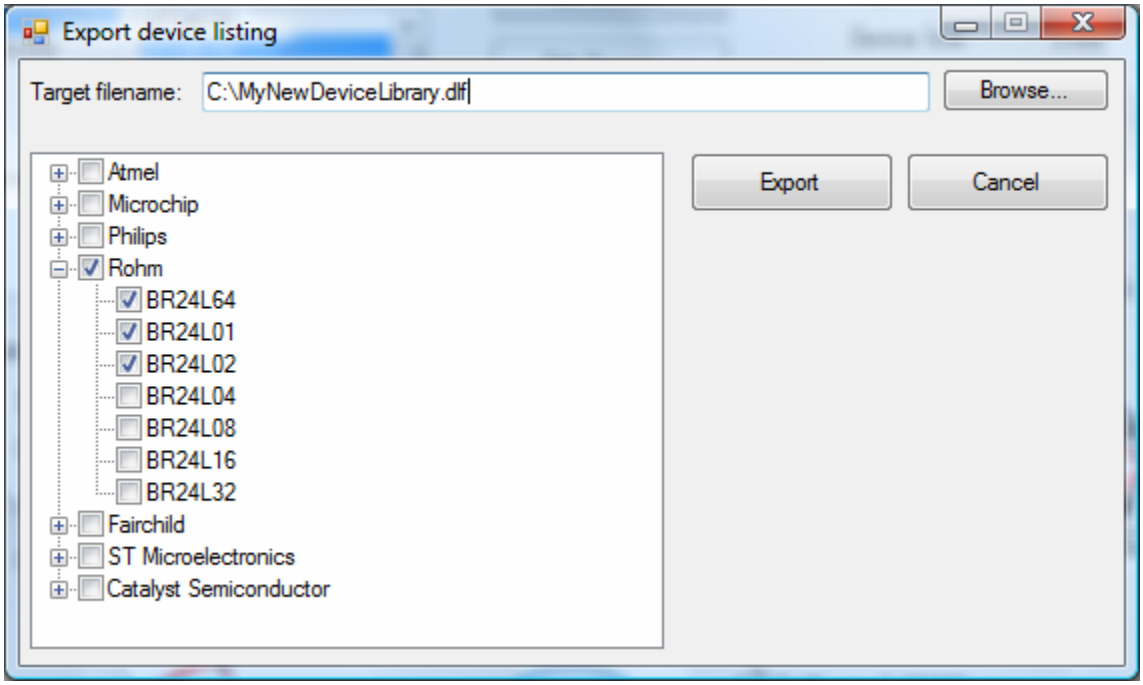
New device descriptions can be added to the device library by clicking the Add *Device…* button, or *New device…* from the *Device Library* submenu of the *EEPROM* menu. See the "I2C EEPROMs" section for information on determining the proper parameters to describe a particular device.

Removing an existing device



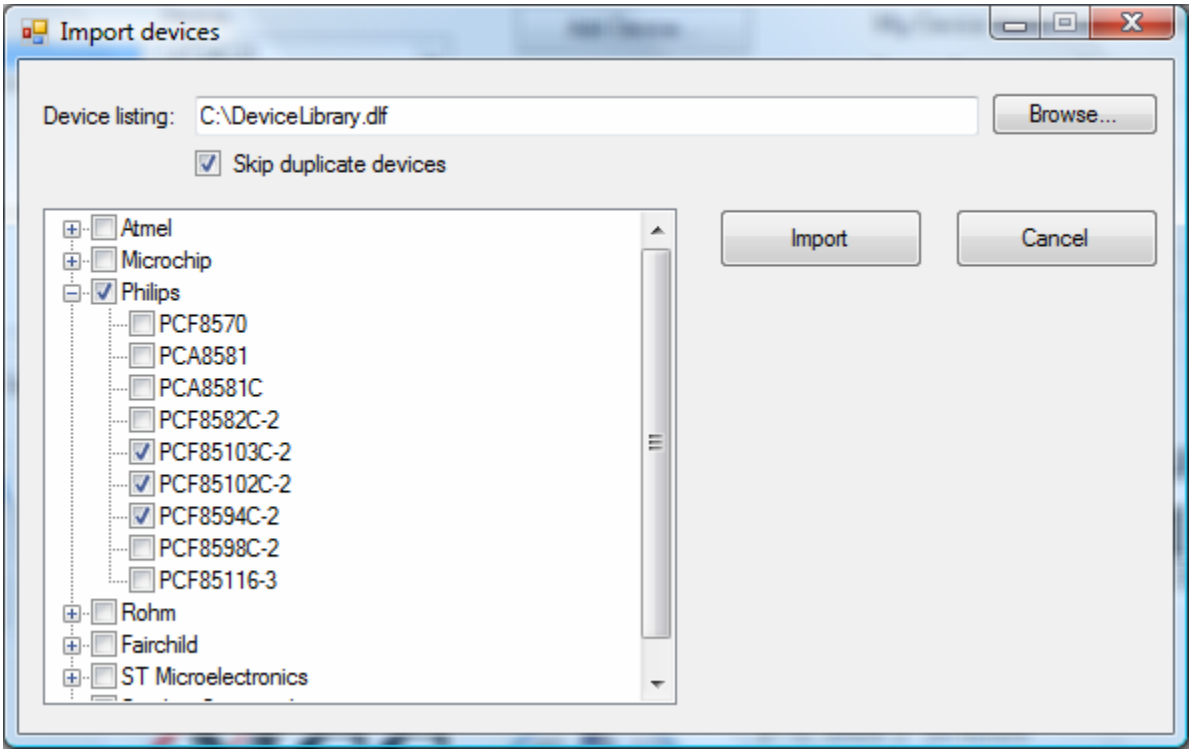
Devices can be removed from the device library by selecting the manufacturer and device and clicking the *Delete Device* button, or *Delete device* from the *Device Library* submenu of the *EEPROM* menu. A dialog box is displayed to confirm the removal.

Exporting devices



The complete device library or any selection of devices within it can be exported for use in another installation of iBurner by clicking the *Export Device…* button, or *Export device listing…* from the *Device Library* submenu of the *EEPROM* menu. Devices can be checked and unchecked to denote whether or not they will be exported. By default, no devices are checked. A device manufacturer can be checked or unchecked to check or uncheck all of the devices belonging to that manufacturer.

Importing devices



A complete device library that was previously exported or any selection of devices within it can be imported into iBurner by clicking the *Import Device…* button, or *Import device listing…* from the *Device Library* submenu of the *EEPROM* menu. Devices can be checked or unchecked to denote whether or not they will be imported. By default, no devices are checked. A device manufacturer can be checked or unchecked to check or uncheck all of the devices belonging to that manufacturer.

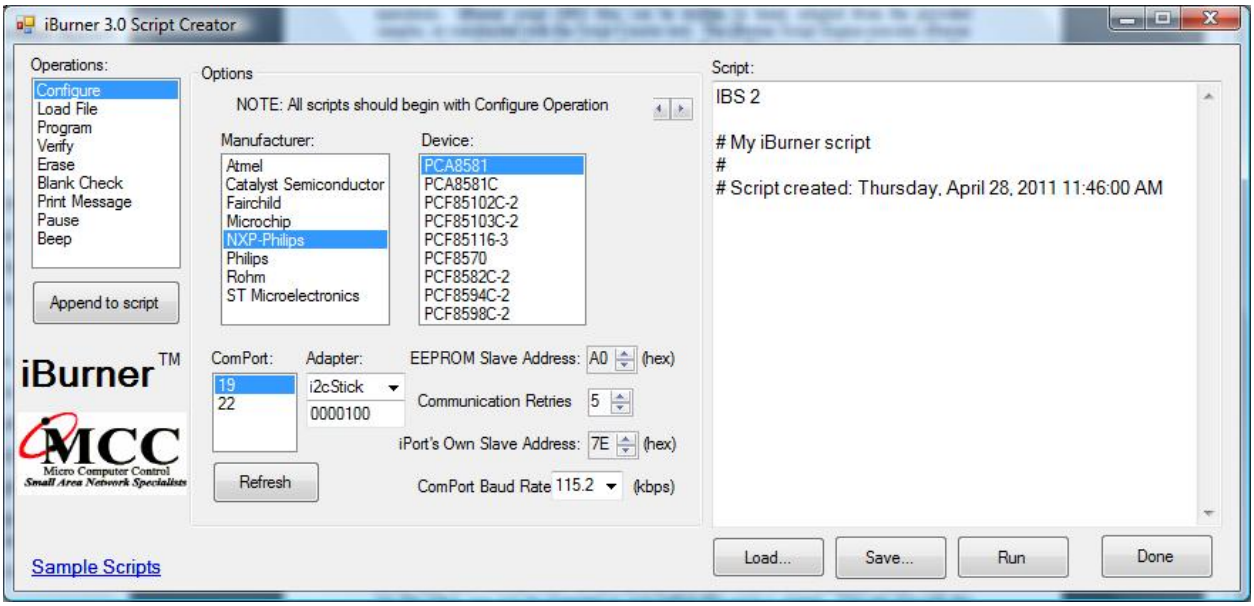
Automation and scripting

iBurner provides a powerful script language that you can use to automate many EEPROM operations. iBurner script (IBS) files can be written by hand, adapted from the provided samples, or constructed with the Script Creator tool. The iBurner Script Engine executes iBurner scripts, and can be run from the Script Creator tool, from a Windows shortcut, from a batch file, or from another executable application via a Windows API call.

Starting with iBurner 3.0, the Script Engine returns an exit code (0=Success, -1=Fail) that can be used to determine successful completion of the script. Specific details of script performance can be found in the iBurner log file.

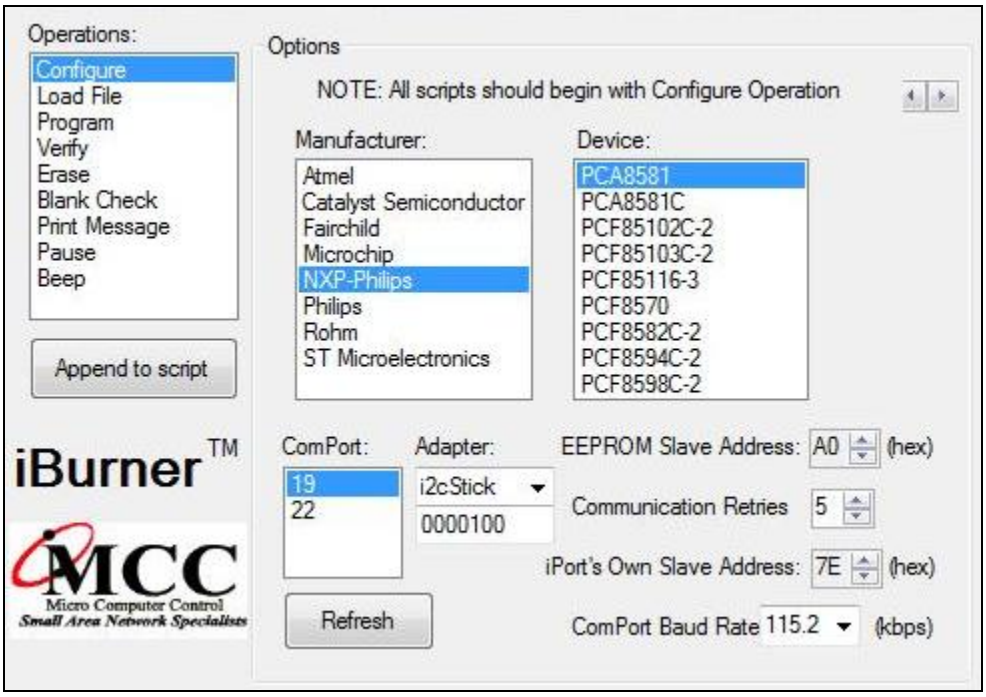
Script Creator Tool

The Script Creator tool provides a GUI for writing IBS files by allowing the user to select the desired operations, enter the required parameters, and automatically construct the script. The Script Creator can be started from the iBurner folder on the Windows Start menu.



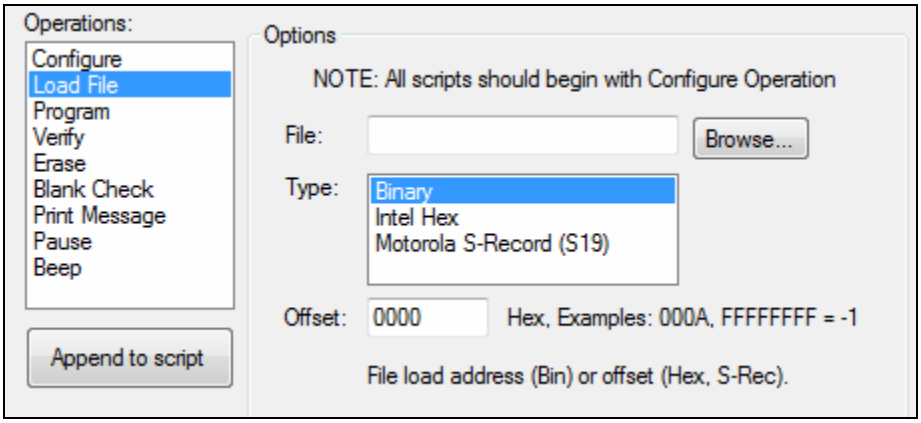
To create a script, select operations in the list on the left, enter the proper options into the center panel, and click "Add to script" to add the necessary IBS code to the script file. Click "Run" to run the script; you will be prompted to save before the script is started. You can also edit the script manually in the on-screen display.

Configure



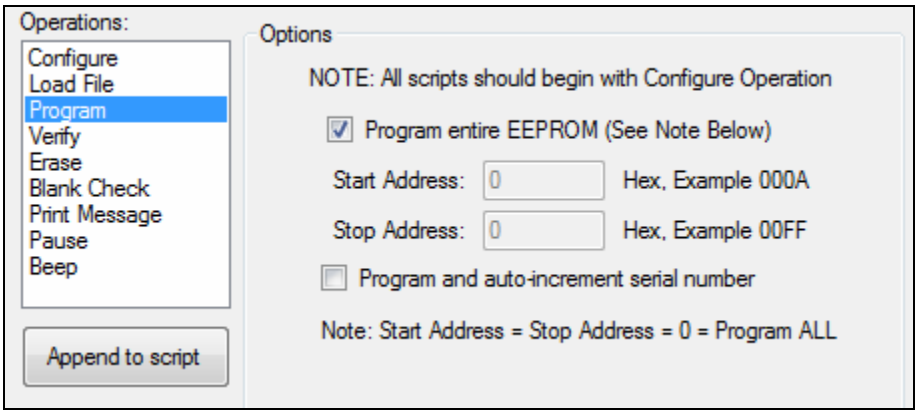
The "Configure" operation should be the first operation added to the script. Select the manufacturer and device name from the device library (if your device is not present you can add it in the main iBurner application as detailed earlier in this guide). Select the slave address of the EEPROM device, the ComPort, and the ComPort baud rate you wish the application to use, along with the number Communication Retries on failure, and click "Append to script". A "Configure" operation consists of a number of individual commands, which will be added automatically, along with comments and error detection.

Load file



The "Load file" operation allows you to load a file into the buffer at a particular offset. The Script Creator supports the same files as the iBurner application. Click "Browse…" to select the file, choose the proper file type and offset, and click "Add to script".

Program



The "Program" operation programs a selected region of the EEPROM with the buffer, and optionally inserts a serial number into the buffer (and auto-increments it upon successful programming). The serial number configuration (size, initial value, and offset within the buffer) can be set in the main iBurner application.

Verify

The "Verify" operation performs a byte-by-byte comparison of the EEPROM device and the buffer. If they do not match, or an error interrupts the process, the script terminates with a message; otherwise script operation continues. There are no configuration options for the "Verify" operation (you can edit the script manually if you want to change the message or operation behavior).

Erase

The "Erase" operation writes blank (0xFF) bytes to the EEPROM device. There are no configuration options for the "Erase" operation.

Blank Check

The "Blank Ccheck" operation verifies that every byte in the EEPROM device is blank (0xFF). If a non-blank byte is found, or an error interrupts the process, the script terminates with a message; otherwise script operation continues. There are no configuration options for the "Blank check" operation (you can edit the script manually if you want to change the message or operation behavior).

Print message

The "Print message" operation displays a message when executed. Enter the desired message and click "Add to script".

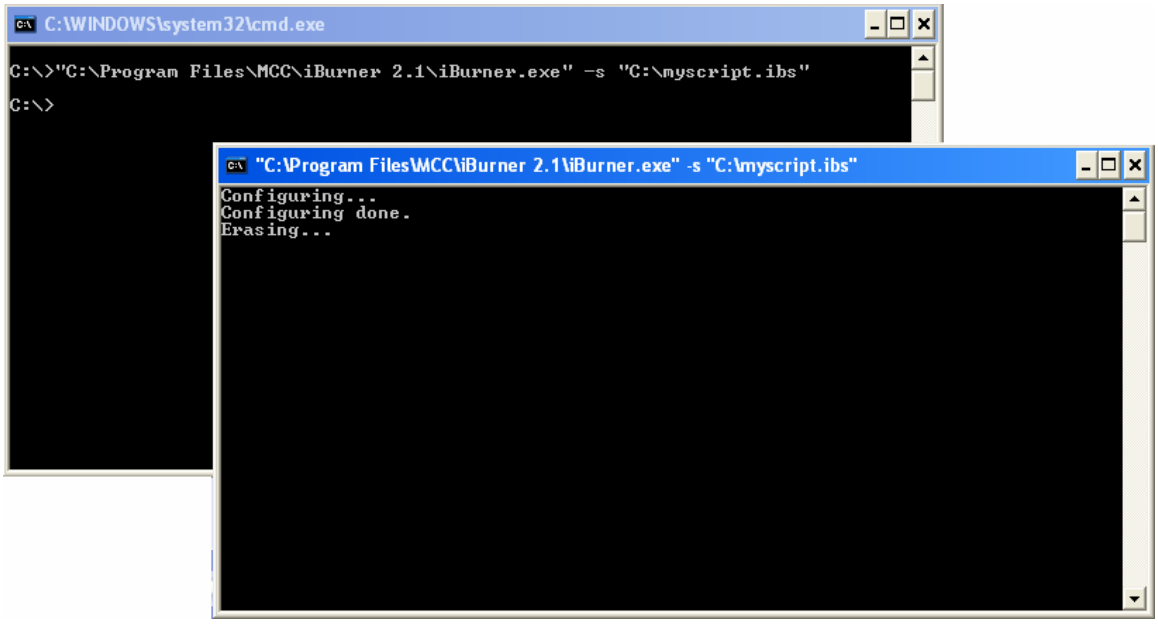
Pause

The "Pause" operation pauses script execution until the user presses a key. If the user presses the escape key, script operation terminates (without a message).

Beep

The "Beep" operation generates a PC speaker beep sound that can be used as an operator alert.

Running scripts



IBS scripts can be run from the command line, from shortcuts, from batch files, or another program, with the following command.

```
"C:\Program Files\MCC\iBurner 3.0\iBurner.exe" -s
"C:\The\Path\To\My\Script.ibs"
```

The "-s" option indicates that the program should run in scripted mode. Alternatively you could specify "-m" which will interpret a script in "managed" mode, meaning that the script will wait for the user to press a key before exiting. This is useful for temporary command line windows, such as that created when running scripts with Windows shortcuts.

You can start the Script Creator itself from the command line, as well.

```
"C:\Program Files\MCC\iBurner 3.0\iBurner.exe" -c
```

Script Engine Automation Examples can be found on the iBurner Help menu.

Script logging

Scripts log the same information as operations performed manually in the main iBurner application, but this information is not displayed to the screen (only script outputs are displayed on the command line). The logged information is stored to the same log files as used in the main iBurner application, and can be retrieved by clicking "File", "View log files…" or by navigating to the "Logs" folder of the program installation directory.

Script language specification

The Script Creator provides a handy way to generate scripts; if you require advanced control over the script or wish to edit an existing script, this section will document the IBS format in its entirety.

This is version 2 of the IBS file format.

IBS signature

Every IBS file begins with an IBS signature, a single line containing "IBS X", where X is the version of the IBS format. Attempting to execute scripts of newer versions than that supported by the iBurner software will yield an error, causing script termination.

The stack

iBurner scripts operate on a stack (LIFO structure). The IBS file is processed line-by-line, with the first character on the line determining how the contents of the line interact with the stack. Blank lines are ignored.

Any line that begins with # is a comment. The line is ignored and script execution moves on to the next line.

Any line that begins with * is a command. There is a fixed set of available commands, documented below. Commands pop some number of arguments off the stack, operate on them, and then push some number of return arguments onto the stack.

All other lines are assumed to be data. Data is pushed onto the stack.

The following script illustrates the difference between comments, data, and commands, and how commands operate on the stack.

```
IBS 2

# The "5" below is neither a comment nor a command,
# so it is pushed onto the stack.
5

# The "port" command below pops one argument off the
# stack and if it is a valid number sets it as the
# ComPort on which further operations will occur.
*port

# The "port" command pushes true or false, indicating
# whether or not the provided number was valid. If the
# "port" command failed, the script automatically
# terminates with a –1 exit code.

# If the port number was bad, the script won't get
# this far, it will display our message and quit.
# Otherwise the script will just keep running.
```

Script Commands

The following lists all commands available to iBurner scripts. The arguments are listed in the order they should be pushed.

Command	Arguments	Return arguments	Description
msg	*text*	none	Displays *text* to the screen.
pause	none	none	Waits for the user to press a key. If the key is ESC, the script exits immediately.
quit	*flag*, *text*	none	Legacy command, ignored. Starting with iBurner 2.2, scripts automatically terminate when an error is detected.
port	*port*	*flag*	*port* is established as the desired COM port. *flag* indicates success or failure. *port* should be a decimal number (“1”, “14”, etc).
baud	*rate*	*flag*	*rate* is established as the serial link speed. *flag* indicates success or failure. *rate* should be “19.2”, “57.6”, or “115.2”.
haddress	*addr*	*flag*	*addr* is established as the host adapter I2C address. *flag* indicates success or failure. *addr* should be a two digit hexadecimal value (“A0”, “AE”, etc).
eaddress	*addr*	*flag*	*addr* is established as the target EEPROM I2C address. *flag* indicates success or failure. *addr* should be a two digit hexadecimal value (“A0”, “AE”, etc).
retry	*count*	*flag*	*count* is established as the retry count. *flag* indicates success or failure. *count* should be a decimal number (“0”, “5”, etc).
eeprom	*mfg*, *prod*	*flag*	The EEPROM with manufacturer *mfg* and product name *prod* is established as the target EEPROM. *flag* indicates success or failure.
load	*fname*, *form*, *offset*	*flag*	The file *fname* (of format *form*) is loaded into memory at *offset*. *flag* indicates success or failure. *form* should be “bin”, “hex”, or “s19”. *offset* should be a hexadecimal number (“00A5”, “FF0”, etc).
program	*saddr*, *eaddr*	*flag*	The previously configured EEPROM is programmed from address *saddr* to *eaddr*. *flag* indicates success or failure. *saddr* and *eaddr* should be hexadecimal numbers (“00A5”, “FF0”, etc).
programwsn	saddr, eaddr	*flag*	Identical to *program*, but writes the configured serial number to the buffer (configured in the GUI) and auto-increments it after a successful program.
verify	none	*flag*	Verifies that the buffer and device are equivalent. *flag* indicates this.
erase	none	*flag*	Erases the device. *flag* indicates success or failure.
blank	none	*flag*	Verifies whether or not the device is blank. *flag* indicates this.

Sample scripts

A number of sample scripts that can be modified as needed are available in the Script Creator “Sample Scripts” link and Load button.