

User's Guide



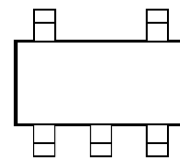
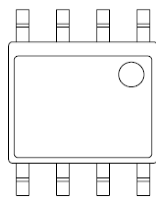
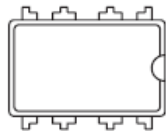
I²C Bus Serial EEPROM Programmer Software for MCC ASCII Interface I²C Bus Host Adapters

Version 1.1

10101000

I²C Bus

01001010



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 2000, XP, or higher PC, 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), and iPort/USB (#MIIC-204).

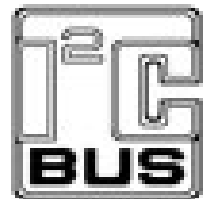
This user's guide describes the operation of the iBurner software.

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 Philips components and are licensed to use the I²C Bus.

“Purchase of Philips I²C components conveys a license under the Philips' I²C patent to use the components of the I²C system, provided the system conforms to the I²C specifications defined by Philips.”

I²C is a trademark of Philips Corporation.



15-MAY-07

Revision Report

15-MAY-07 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.

16-AUG-06 Initial Release V1.0

Copyright© 2007 by Micro Computer Control Corporation. All rights are reserved. No part of this publication may be reproduced by any means without the prior written permission of Micro Computer Control Corporation, PO Box 275, Hopewell, New Jersey 08525 USA.

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

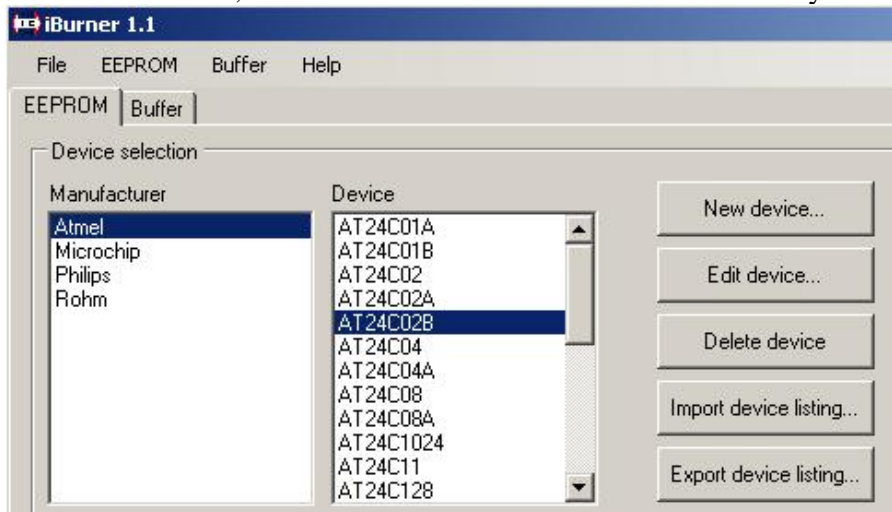
iBurner 1.1 User's Guide

© 2007 Micro Computer Control Corporation

Introduction to the iBurner Software	2
System Requirements.....	2
Using the EEPROM View	3
Selecting a Target Device	3
The Device Library	4
Adding new devices to the device library.....	4
Editing devices.....	5
Deleting devices from the device library	5
Importing a device listing	6
Exporting a device listing	6
Communication Settings.....	6
Selecting a host adapter	6
Choosing a COM port.....	7
Adapter and chip addresses.....	7
Serial link speed and I2C clock speed	8
Retry count.....	8
Using the Buffer View	8
Loading Files Into the Buffer.....	9
Loading binary files	9
Loading Intel ASCII Hex files.....	10
Loading files with non-conventional filenames.....	10
Saving the Buffer to a File	10
Modifying the buffer.....	11
The “fill” operation	11
The “copy” operation.....	11
EEPROM Operations.....	12
Programming.....	12
Reading	13
Erasing	14
Verifying.....	14
Byte-by-byte	14
Blank check.....	14
Checksumming	14
Device checksum	15
Buffer checksum	15
Command Line Operation.....	15
Troubleshooting	16
Application related issues	16
EEPROM related issues.....	17
Appendix A – Alternate Multi-Bank Chip Use	18

Introduction to the iBurner Software

iBurner is an I2C based EEPROM programmer, developed to interface with the iPort line of I2C bus host adapters. It can program, read, erase, and verify data stored on devices, as well as compute checksums, load and save data in two file formats, and manipulate a buffer of data in a variety of ways. The software is provided with a number of EEPROM device definitions, and allows the user to add more as necessary.



iBurner has two tabs, each focusing on a particular aspect of serial EEPROM programming. The EEPROM tab provides configuration options, such as device and I2C host adapter selection, and the major operations that can be performed, including programming, reading, erasing, and verifying. The Buffer tab provides a view of the iBurner's memory space, and allows manipulation of the data within this memory space. In addition, all available functionality can be accessed from either tab by using the menu.

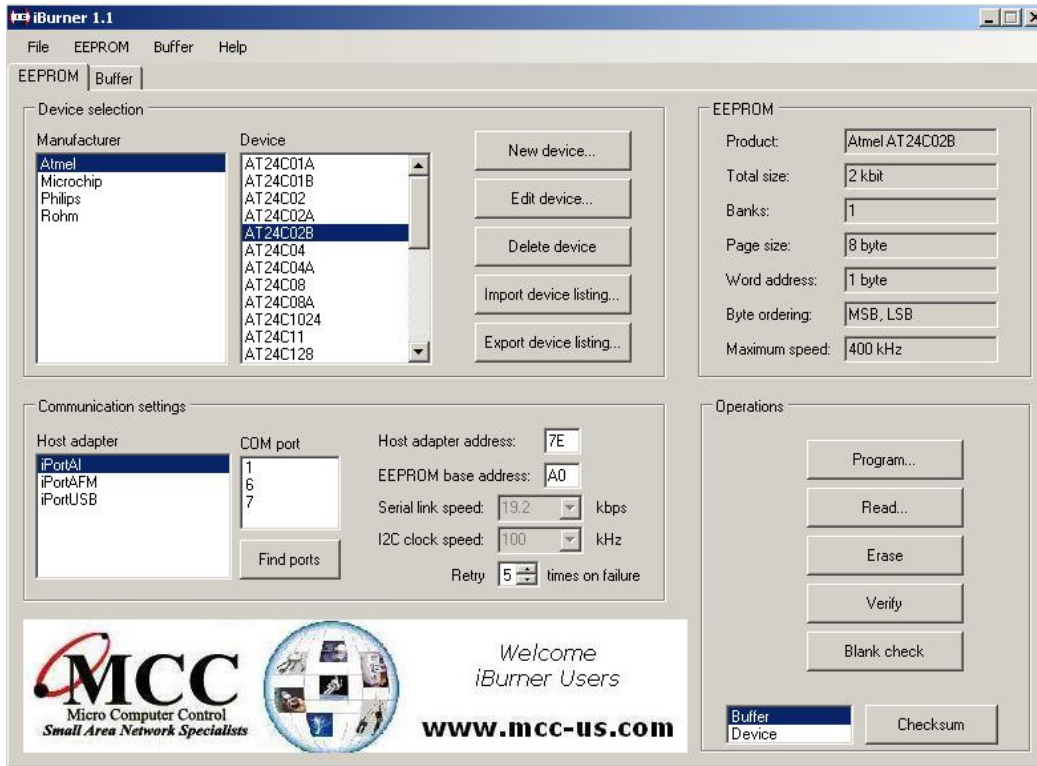
System Requirements

In order to use the iBurner software you must meet the following system requirements:

1. Microsoft Windows operating system
 - Windows 98 SE
 - Windows 2000
 - Windows XP
2. I2C bus host adapter
 - iPort/AI (#MIIC-202)
 - iPort/AFM (#MIIC-203)
 - iPort/USB (#MIIC-204)

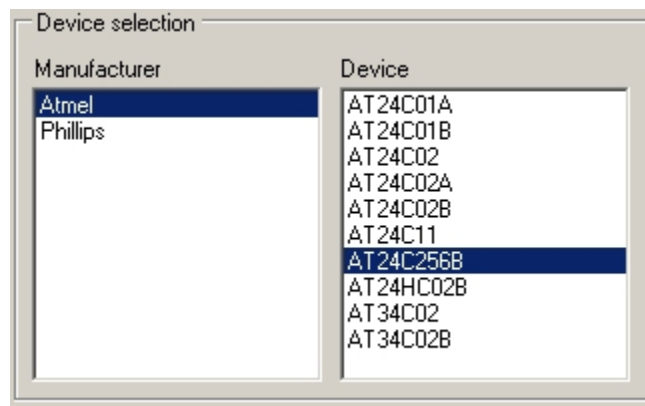
Using the EEPROM View

The EEPROM view provides configuration and device specific options, as well as the major operations that can be performed.



Selecting a Target Device

A target device can be selected from the Device Library by first choosing the manufacturer name. This will bring up (in the "Device" box) a list of supported devices. Selecting a device will load that device's specifications into the information boxes on the far right of the screen.



Every time you select a device, the buffer (see [Using the Buffer View](#)) will be updated to reflect the new device. If the selected device has less memory than the previous selected device, you will be prompted to continue, as some data might be lost as the buffer becomes smaller.

Product:	Microchip 24AA024
Total size:	2 kbit
Banks:	1
Page size:	16 byte
Word address:	1 byte
Byte ordering:	MSB, LSB
Maximum speed:	400 kHz

The Device Library

The device library is a collection of device specifications that tells the iBurner software how to access a given EEPROM. You can interact with the library in several ways.

Adding new devices to the device library

Manufacturer:	Phillips	
Device name:	PCF8570	
Total size:	128	byte
Number of banks:	1	
Page size:	1	byte
Word address:		bytes
Byte ordering:	MSB, LSB	
Maximum speed:	100	kHz

OK Cancel

It is possible that your target device is not currently in the device library. In this case you can add your device manually, by clicking the “New device...” button. This will bring up a window in which you can enter the defining characteristics of the device. The parameters are described below.

Manufacturer: this is the name of the company that produces the device. If the manufacturer is already present in the device library, the new device will be added to its list.

- Device name: this is the product name, or any identifying string, that you wish to use to reference the device.
- Total size: This is the complete amount of memory available on the chip. Make sure you specify the correct units.
- Number of banks: Some EEPROMs span several I2C addresses; instead of being accessed as one contiguous block of memory, they must be accessed via separate I2C addresses. Each of these blocks of memory is called a “bank” (some documentation refers to them directly as “blocks”). The iBurner software must know how many memory banks the target chip has. NOTE: these banks must be organized sequentially. Very rarely, a chip will have multiple blocks, but the blocks will not be organized sequentially... in other words, the several I2C addresses that the chip spans are not contiguous. These chips must be treated specially... see [Appendix A](#) for more information.
- Page size: This is the number of bytes that can be written to in a single operation.
- Address width: Devices with more than 256 bytes of memory per bank require more than one address byte. Note that this is the internal word address, not the device I2C address.
- Byte ordering: For devices with more than one address byte, the iBurner software must know the byte order. By far the most common configuration is “MSB, LSB”, or most-significant-byte first. This setting has no effect on chips that use only a single byte for addressing.
- Maximum speed: All I2C devices have a maximum bus clock speed at which they can operate. All devices can work at the base I2C speed of 100 kHz, so if in doubt, 100 is a safe choice.
-
- Each bank in a multi-bank chip can also be accessed individually. See [Appendix A](#) for more information.

Editing devices

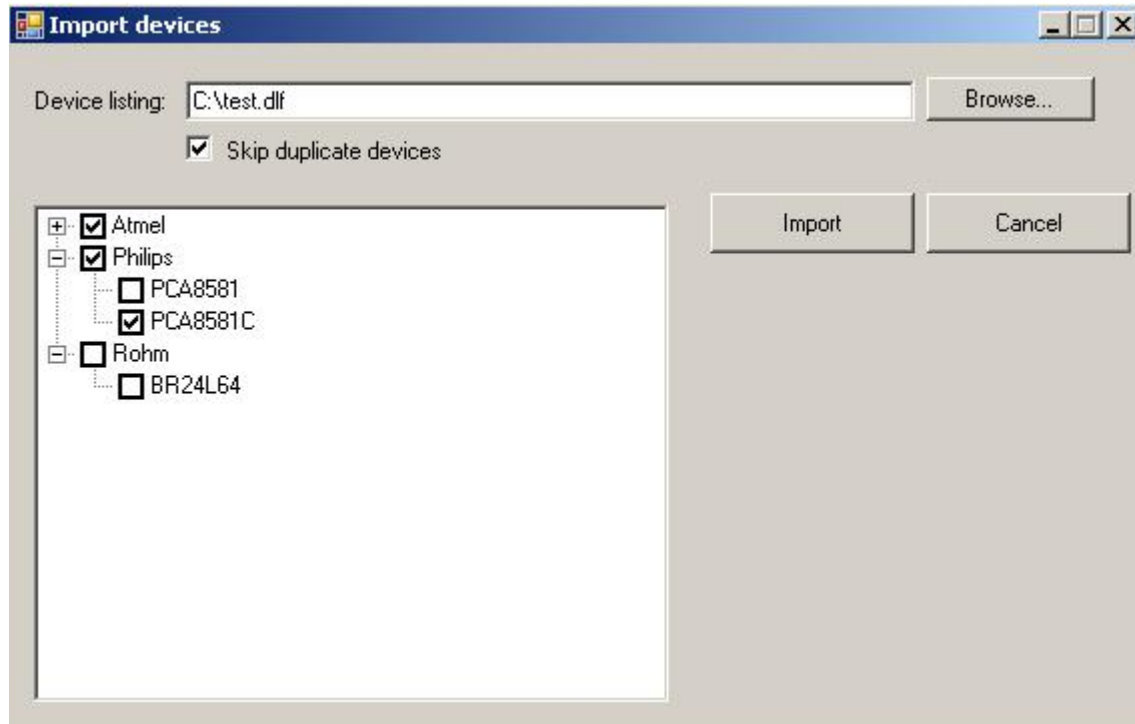
To edit an existing device description, select it and click "Edit device...". A form similar to the "New device" dialog will appear, in which you can change the current device's parameters. When you click "OK", the chip description will be updated and the buffer adjusted to match the device.

Deleting devices from the device library

To delete a device permanently from the library, select it and click the “Delete device” button. You will be prompted for verification before the change occurs.

Importing a device listing

Over time, companies (including MCC) may release Device Listing Files (DLFs) that provide specifications for more serial EEPROMs that the iBurner software can support. Rather than add each device to the library by hand, you can import all devices in the DLF file automatically, by clicking “Import device listing...”. Click “Browse” in the resulting window to choose the desired DLF file to import. Unchecking the “Skip duplicate devices” option will overwrite old devices with new devices of the same name, if duplicate devices are found. Check the devices (or manufacturer to include all devices by this manufacturer) that you wish to import.



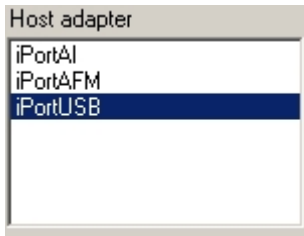
Exporting a device listing

You may also wish to distribute your own DLF file. To do this, click the “Export device listing...” file. You may then choose (via the “Browse” button) a file location to which the DLF file will be saved, and select the devices you wish to export. This file can then be distributed to other iBurner users.

Communication Settings

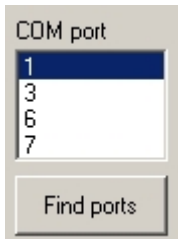
Selecting a host adapter

To select your host adapter, click the name of the adapter from the list. All devices are backwards compatible with the iPortAI, so selecting “iPortAI” is always a safe choice.



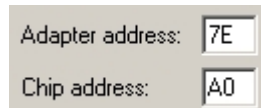
Choosing a COM port

The COM port is the number of the serial port to which your iPort is attached. For iPortUSB devices, this serial port may be “virtual” (as the device is connected to the PC in a fashion other than over a serial cable). Click “Find ports” to have the iBurner software scan for any COM ports on your system. The resulting ports will be listed in the box just above the button. Click on a port to select it.



Adapter and chip addresses

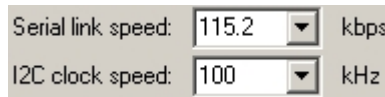
Both the I2C host adapter (your iPort) and the target device have an I2C slave address, an even two-digit hexadecimal number. The adapter’s address should be set to avoid conflicts with other slave devices on the bus. In most situations the default value of 7E will suffice. However, if you are writing to an active bus (ie. Other devices are communicating across the I2C bus at the same time you are intending to operate on the EEPROM), you may wish to adjust this value so it does not coincide with the address of an existing device on your bus.



The target device's slave address is set via hardware pins on the chip. You must then provide this address to the iBurner software so it can communicate with the device over the I2C bus. See the device data sheet more device-specific information on slave addressing.

Serial link speed and I2C clock speed

The serial link speed is the speed at which data is transferred between the host adapter (the iPort) and the PC that the iBurner software is running. The I2C block speed is the speed at which data is transferred across the I2C bus. The iPortAFM and iPortUSB devices can operate at a variety of link and clock speeds, while the iPortAI is fixed at 19.2 kbps and 100 kHz.



Serial link speed: 115.2 kbps
I2C clock speed: 100 kHz

Retry count

It is possible (especially on an active bus) that one of the write or read operations that make up the program, read, erase, and verify routines could be interrupted, for a variety of reasons. Raising the retry count allows the iBurner software to repeat the operation a certain number of times.



Retry 5 times on failure

Note that this count is for each individual write or read. If you were programming a device, and the first 8 bytes to be written failed to write properly, you would have a certain number of retries (set in the retry count box). Once the write succeeded, you would have the same number of attempts to perform the second 8 bytes, and so on.

This is particularly important if the target EEPROM has a burn cycle of longer than 10 ms, as it will not be able to respond while it is burning, and may cause the next write to fail. There is an 100 ms pause between each retry, so five retries (the default setting) should be more than enough for any EEPROM.

Using the Buffer View

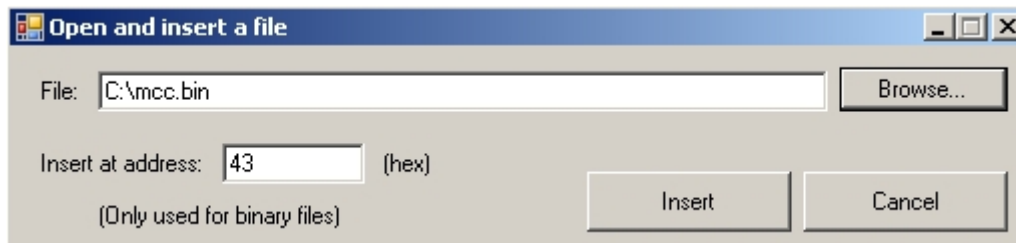
The buffer view is accessed by clicking the “Buffer” tab on the main screen. If you have not yet selected a target device the buffer will be empty, otherwise it will contain as many bytes as can fit within the selected device. On the far left is a column labeled “Address”. This column holds the address of the first byte in each row. The grid in the center holds a hexadecimal representation of the data, and the grid on the right holds an ASCII display. Note that the ASCII display does not included extended ASCII characters.

There are a number of operations you can perform upon the buffer.

EEPROM Buffer																																			
Address	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F			
0	4D	43	43	20	20	4D	69	63	72	6F	20	43	6F	6D	70	75	M	C	C			M	i	c	r	e	o	C	o	m	p	u			
10	74	65	72	20	43	6F	6E	74	72	6F	6C	20	43	6F	72	70	t	e	r	C	o	n	t	r	o	l	C	o	r	p					
20	6F	72	61	74	69	6F	6E	20	20	77	77	77	2E	6D	63	63	o	r	a	t	i	o	n		w	w	w	.	m	o	e				
30	2D	75	73	2E	63	6F	6D	20	20	61	62	63	64	65	66	67	-	u	s	.	c	o	m		a	b	c	d	e	f	g				
40	68	69	6A	68	6C	6D	6E	6F	70	71	72	73	74	75	76	77	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w			
50	78	79	7A	31	32	33	34	35	36	37	38	39	30	41	42	43	x	y	z	1	2	3	4	5	6	7	8	9	0	A	B	C			
60	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F	50	51	52	53	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S			
70	54	55	56	57	58	59	5A	20	20	4D	43	43	20	20	4D	69	T	U	V	W	X	Y	Z		M	C	C	M	i						
80	63	72	6F	20	43	6F	6D	70	75	74	65	72	20	43	6F	6E	e	r	o	C	o	m	p	u	t	e	r	C	o	n					
90	74	72	6F	6C	20	43	6F	72	70	6F	72	61	74	69	6F	6E	t	r	o	C	o	r	p	o	r	a	t	i	o	n					
A0	20	20	77	77	77	2E	6D	63	63	2D	75	73	2E	63	6F	6D			w	w	w	.	m	o	e	-	u	s	.	c	o	m			
B0	20	20	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E			a	b	c	d	e	f	g	h	i	j	k	l	m	n			
C0	6F	70	71	72	73	74	75	76	77	78	79	7A	31	32	33	34	e	p	q	r	s	t	u	v	w	x	y	z	1	2	3	4			
D0	35	36	37	38	39	30	41	42	43	44	45	46	47	48	49	4A	5	6	7	8	9	0	A	B	C	D	E	F	G	H	I	J			
E0	4B	4C	4D	4E	4F	50	51	52	53	54	55	56	57	58	59	5A	x	i	m	e	n	d	r	e	s	t	u	v	w	x	y	z			

Loading Files Into the Buffer

The iBurner software supports three file formats: “flat” binary images, Intel ASCII Hex files, and Motorola S-Record files.



Binary images are byte-for-byte representations of a contiguous block of memory. Intel ASCII Hex and Motorola S-Record files consist of records, each describing a small section of memory. For instance, a binary image might contain the 1024 bytes to be written somewhere (the file does not specify) to a device, while an Intel ASCII Hex file or Motorola S-Record file might contain the first 8 bytes, the last 12 bytes, and any number of small sections to be placed at a specific position within the buffer.

Loading binary files

All files are loaded by clicking “Open...” under the “File” menu. Click “Browse” to select a file to load. For binary files, you may also set the address at which the file will be inserted into the buffer by entering it into the provided box. For example, you may wish to load a binary file containing 512 bytes of data. Because the binary file does not specify where this data belongs within the buffer, you must enter the address of the first byte into the box. Clicking “Insert” will then load the file and write its contents to the desired location.

Loading Intel ASCII Hex files

Hex files are loaded the same way as binary files, except that the “Insert at address” box has no effect. Hex files specify where the data belongs in the file, and cannot be positioned manually.

Loading Motorola S-Record files

S-Record files are loaded identically to ASCII Hex files.

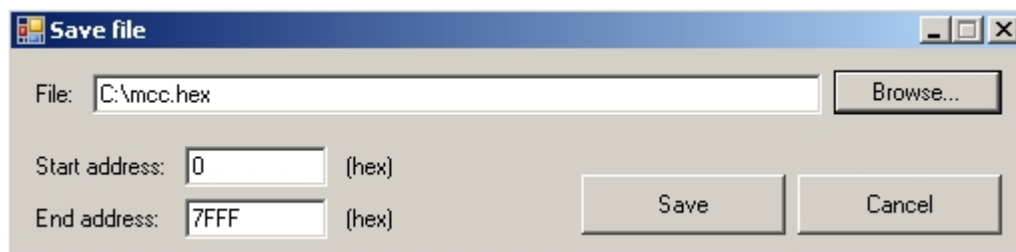
Loading files with non-conventional filenames

The iBurner software expects binary files to have the extension “.bin”, hex files to have the extension “.hex”, and S-Record files to have the extension “.s19”. If your file does not have one of these extensions, but you know that its format is either binary, Intel ASCII Hex, or Motorola S-Record, you may still open it with iBurner. Click “Open” under the “File” menu as before, and click “Browse” to select the file. Set the file filter to display all files, and choose your file. When you click “Insert” you will be prompted to choose what format of file you are opening.

Saving the Buffer to a File

You may wish to save your buffer to a file to keep a record of the data you have written to a device. To do this, click “Save as...” under the “File” menu. You can then click “Browse” to select a file type and location to save the file to. If you don’t want to save the entire buffer you can adjust the “Start address” and “End address” boxes. Finally, click “Save” to write the file.

iBurner will determine the correct format in which to save based on the extension of the filename. A “.bin” filename will be saved as a binary file, a “.hex” filename as an Intel ASCII Hex file, and a “.s19” filename as a Motorola S-Record file. If any other file extension is used, a dialog box will appear in which the correct file-type (binary, hex, or S-Record) can be selected.



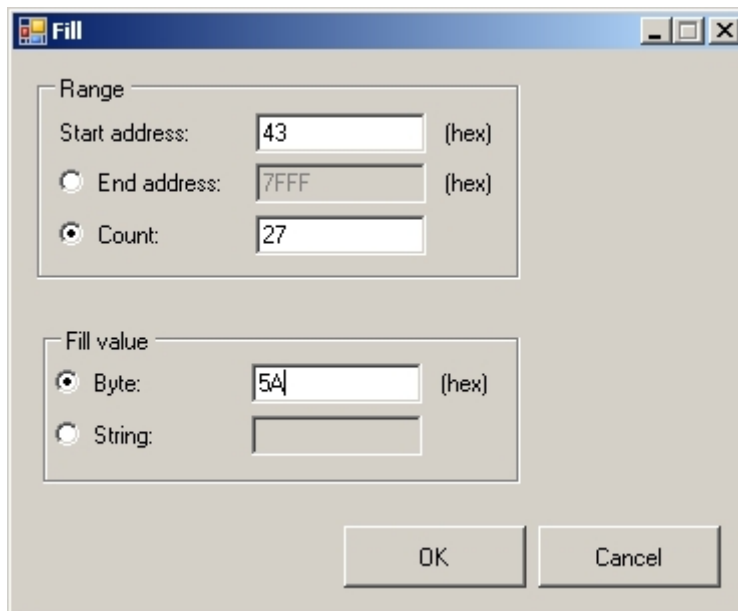
The default range for saving a file will always be the entire file. When you exit the iBurner software you will be prompted whether or not you want to save the buffer to a file.

Modifying the buffer

You can edit any byte in the buffer by clicking or double clicking on it, and then entering the value you wish to replace it with. Editing a byte can be canceled by pressing the escape key. You can move directly to a byte by clicking “Go to address...” under the “Buffer” menu and entering the desired address into the resulting dialog.

The “fill” operation

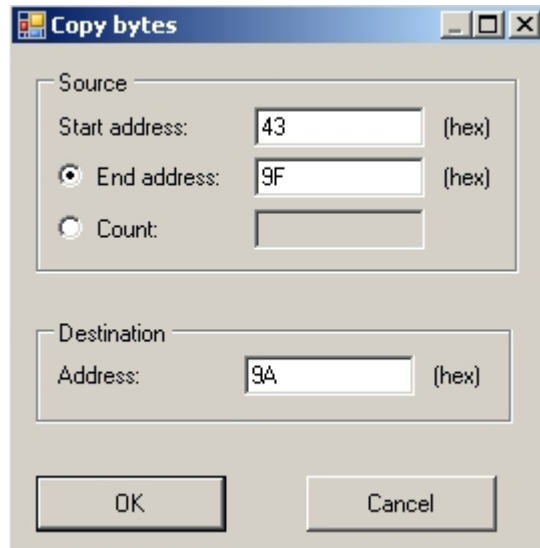
Filling entails taking a particular byte of data or ASCII string and replacing some block of memory with that byte or string, repeated as many times as necessary. You can perform a fill by clicking “Fill...” under the “Buffer” menu. Enter the desired range, either in terms of a start and end address, or a start address and a number of bytes to fill. The default start address will always be the currently selected byte, while the default ending address and count will be the end of the buffer.



In the “Fill value” section, choose “Byte” or “String”, and enter the appropriate data. Clicking “OK” will then fill the desired range with the supplied data.

The “copy” operation

A copy entails taking some range of data from the buffer and duplicating it at some other address. A copy can be performed by clicking “Copy...” under the “Buffer” menu. You will need to supply a source range, from which the data to be copied will come. You can do this in terms of a start address and an ending address, or a start address and a number of bytes to copy. The default start address will always be the currently selected byte.

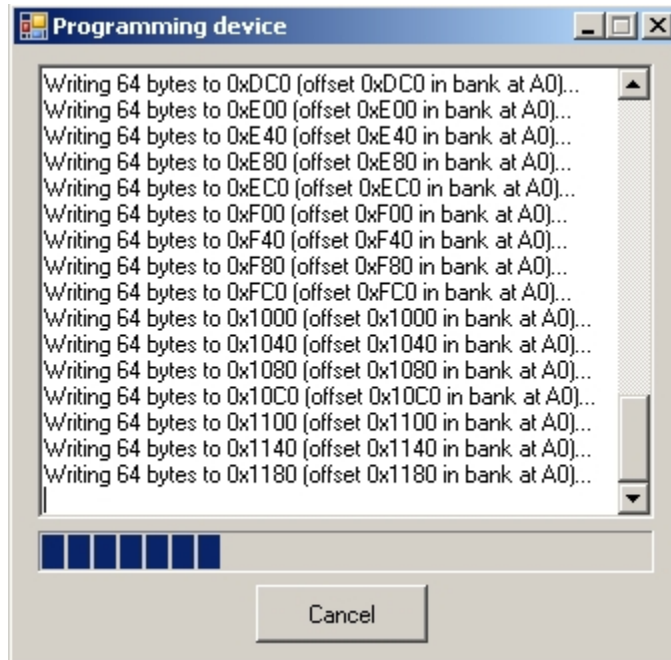


You must also enter a destination address. Overwriting is allowed, meaning that the destination address may fall inside the source range (the entire source range will copied, then written to the destination afterwards).

EEPROM Operations

Programming

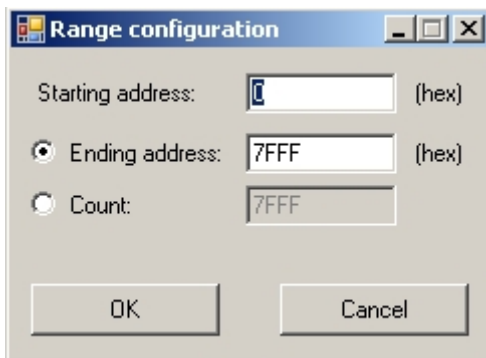
The “program” operation writes the memory buffer to the device. To do this, click “Program...” under the “EEPROM” menu, or click the “Program...” button in the EEPROM view. A window will appear that will let you choose the range to write, either by specifying a starting address and an ending address, or a starting address and a number of bytes to write. The default starting address will always be 0, and the default ending address and count will always be the end of the buffer.



Click “OK” to start the program procedure. You will be prompted to verify the overwriting of any data that is currently on the device, and a window will appear to inform you of the progress of the write.

Reading

The “read” operation reads data from the device into the memory buffer. This can be performed by clicking “Read...” under the “EEPROM” menu, or by clicking the “Read...” button in the EEPROM view. Like with a program operation you will be able to specify the range you wish to read. The default starting address will always be 0, and the default ending address and count will be the end of the device’s memory.



Click “OK” to start the read procedure. You will be asked to verify the operation (as the read will overwrite the specified range in the current buffer), and a window will appear to monitor the progress of the operation.

Erasing

The “erase” operation replaces all data on the target device with the byte value 0xFF. This can be performed by clicking “Erase” under the “EEPROM” menu, or clicking the “Erase” button in the EEPROM view. You will be prompted to verify the operation before it begins, and a window will appear to monitor the progress.

Verifying

The purpose of verification is to ensure that the expected data is present in the device’s memory. Verification can either be performed from the “Verify” submenu under the “EEPROM” menu, or from the EEPROM view. There are two types of verification.

Byte-by-byte

Byte-by-byte verification involves reading the entirety of the chip’s memory and comparing it to what is currently in the buffer. When a non-matching byte is found, the relevant information is displayed, and you will be prompted to continue searching or stop.

Blank check

Blank check verification entails reading the device’s memory, and ensuring that each byte is set equal to 0xFF. If all bytes in the device are set to 0xFF, the device is said to be blank. You can verify that an erase operation completed successfully by running a blank check immediately afterwards.

Checksumming

A checksum is a handy number that can be recorded outside of the iBurner software, and then compared at a later time against another checksum. For example, a file might be created, and the checksum calculated and recorded. To make sure that this file has not been modified at a later date, you could compute the checksum again, and make sure that it compares to the recorded value. This can be done for both the buffer and the device’s memory.

As an additional form of verification, the checksum the buffer and the checksum for the device could be computed immediately after programming the device, and compared to make sure that the write completed successfully.

The iBurner software uses an unsigned 4 byte integer sum of all the bytes in the buffer or memory to calculate the checksum value. All bytes are summed up in an unsigned 4 bytes integer variable, allowing overflows, and the final value is reported to the user.

Device checksum

To compute the checksum of the memory inside the device, click “Checksum” under the “EEPROM” menu. The iBurner software will need to read the device’s memory first, and will then provide the checksum value.

Buffer checksum

To compute the checksum of the buffer, click “Checksum” under the “Buffer” menu. This value will be computed very quickly and presented to the user.

Command Line Operation

Major EEPROM operations can be performed from the command line. To take advantage of this, the user could create a Windows Shortcut with the desired command line arguments, and the software would automatically perform the desired operations or apply the desired settings upon starting up. Command line arguments are each separated by a space.

The first command line argument (if any are used) must be the iPort selection. Capitalization does not matter. Parameters in square brackets ([,]) are optional.

Command	Syntax	Examples
iPort selection	iPort/xxx	iPort/AI iPort/AFM iPort/USB
Serial port selection	COMxxx	COM1 COM2
Baud rate	BAUDxxxx	BAUD19200 BAUD57600 BAUD115200
Clock speed	CLOCKxxxx	CLOCK23k CLOCK86k CLOCK100k CLOCK400k
Host adapter address	hostaddressXX	hostaddress7E
Target EEPROM address	targetaddressXX	targetaddressA0
Number of retries allowed	retryX	retry4
EEPROM selection	EEPROM:manufacturer,product	EEPROM:Microchip,24FC128
Load a file into the buffer	open:filename[,loadaddress]	open:c:\mcc.bin,60 open:c:\mcc.hex
Erase the device	erase	
Program the device	program[:startaddress,endaddress]	program

		program:0,AF4
Blank check verification	verify:blank	
Byte-by-byte verification	verify:bytes	

The following example selects an iPortUSB on the third serial port, sets the maximum baud rate and clock speeds, chooses and configures a device, loads a few files, then performs an erase, followed by a blank check (to make sure it was actually erased), followed by a program, followed by a byte-by-byte verify.

```
iBurner iPort/USB COM3 BAUD115200 CLOCK400k hostaddress7E
targetaddressA0 retry4 EEPROM:Microchip,24FC128
open:c:\mcc.bin,60 open:c:\MyRoms\mcc.hex erase verify:blank program
verify:bytes
```

The following example sets up a configuration for an iPortAI and an EEPROM, but does not perform any actions. Note that an iPortAI can only operate at 19.2 kbps and 100 kHz, so no configuration those parameters is necessary.

```
iBurner iPort/AI COM1 hostaddress60 targetaddressAE retry3
EEPROM:Atmel,AT24C04A open:c:\MoreRoms\ThatRom\mcc.hex
```

Troubleshooting

Application related issues

- 1. The application fails to start – instead, an error is displayed mentioning .Net version 2.0.**

This application requires the Microsoft .Net framework, version 2.0 or newer. The .Net framework is free, and can be obtained from www.microsoft.com/net

- 2. Selecting devices takes a long time; the application stops responding for a long time after I select a device.**

Larger EEPROMs require a larger amount of memory to properly represent. Adjusting the buffer for a new device typically takes less than a second, but if the host PC is running many applications (in particular, many other .Net based applications) it can slow down dramatically.

The best way to solve this issue is simply to shut down and restart the software, and shut down any unnecessary applications that may be running.

- 3. Windows displays messages stating that virtual memory is being increased.**

Similar to #2, Windows may determine that more “virtual memory” is needed to maintain system stability. This can happen if you have been running many applications for a long time without restarting your computer.

The most effective solution is to shut down any running software and restart your computer. Note that no action is necessary, but you should at least wait a few minutes to allow Windows to finish allocating more virtual memory before using any running programs.

4. The application stops responding for short amounts of time. Other applications running at the same time stop responding.

This can happen when the .Net framework stops the application to perform maintenance operations, and can last anywhere from a second or two to 10 seconds or more (on a heavily loaded or older system). The best response is just to wait for the application to begin responding again.

If this happens frequently, restart your computer.

EEPROM related issues

1. While connecting to an EEPROM to perform an operation, I receive “Time out” errors.

Make sure that you have selected the correct bus host adapter and serial port, and that the iPort is powered (iPortUSB devices do not need external power). Make sure that you have selected the correct EEPROM.

Verify that you can communicate with the iPort using MCC’s Message Center software.

2. Operations fail immediately after starting.

Make certain that you have selected the correct EEPROM base address. Most EEPROMs allow you to configure the address via pins on the chip, but not all devices use the same pin logic (for instance, some Atmel EEPROMs invert the state of one of the pins). Most EEPROMs fall in the A0 – AE range of addresses; you can try each if necessary.

Is the device currently being used by another master on the I2C bus?

Disconnect power from the I2C bus to clear any busy conditions that other masters may have induced, then try the operation again.

3. Operations fail at some point while they are running.

How long does the EEPROM's read or burn cycle take? Burn cycle's longer than 10 ms may cause the iPort to report an error (as the device is not ready when the next write command occurs). Make sure that the retry count is set sufficiently high (5 retries should be more than enough, there is an 100 ms pause between each retry).

Is the I2C bus active? If another master begins communicating with the EEPROM while you are trying to program or read it, failures may occur. To ensure proper operation, make sure that other devices on the I2C bus are not communicating with the EEPROM while the operation is underway.

4. My EEPROM is not featured in the device library.

Check with the device's manufacturer to see if they provide a DLF file (which can be imported into iBurner) containing information on their components.

Check MCC's website (www.mcc-us.com) to see if a new DLF file has been released.

You can add your [device](#) to the library manually, using the EEPROMs documentation to provide the necessary information. To do this, click "New device..." on the "EEPROM" view.

I can program an EEPROM, but verifying its contents or reading the data back indicates that nothing was actually written.

Verify that the EEPROM is not write-protected.

Check your EEPROM's data sheet for errata. Verify that there are no limitations on the slave addresses that your device may be configured to use.

5. I accidentally deleted an EEPROM from the device library, can I get it back?

- 6. The device description cannot be automatically recovered.** MCC provides the original DLF file packaged with the software online (www.mcc-us.com). If you have the EEPROM's documentation you can add the device to the library manually, by clicking "New device..." on the "EEPROM" view.

Appendix A – Alternate Multi-Bank Chip Use

Note: this material only applies to EEPROMs which have multiple banks of memory (ie. devices that span several slave addresses). Also note that this method is not required for devices with contiguous banks, but may be desired. See "Number of banks" under [Adding new devices to the device library](#) for more information.

Given a multi-bank chip, you may wish to access each bank individually. In other words, you would have a device description in the device library for MyChipBank0, MyChipBank1, MyChipBank2, etc. To do this, you need to add a new device to the library for each bank.

The values of the fields in the "New Device" dialog are slightly different when accessing a device in this manner.

- **Manufacturer:** this is the name of the company that produces the device. The same manufacturer name should be used for each bank description.
- **Device name:** this is the product name, followed by some identifier to indicate the bank, such as "Bank0".
- **Total size:** This is the size of a single bank of memory in the device.
- **Number of banks:** This should always be set to 1.
- **Page size:** This is the number of bytes that can be written to in a single operation.
- **Address width:** This is the number of bytes in the word address.
- **Byte ordering:** For devices with more than one address byte, the iBurner software must know what order the bytes should be placed in. By far the most common configuration is "MSB, LSB", or most-significant-byte first. This setting has no effect on chips that use only a single byte for addressing.
- **Maximum speed:** All I2C devices have a maximum bus clock speed at which they can operate. All devices can work at the base I2C speed of 100 kHz, so if in doubt, 100 is a safe choice.

Once a device description is added for each bank, you are ready to begin. To access a particular bank, select it as you would a device. You must then set the EEPROM base slave address to be the slave address for the selected bank. The relation between the bank and the slave address differs between devices, and can be determined by examining the device's documentation.

Any operations that occur from this point on will operate on the desired bank in the device, instead of on the device as a whole.