Programming and Duplicating the BL233B Internal EEProm

Introduction

This does not explain how to use BL233 to read and write I2C and SPI serial EEProms! It explains how to configure the BL233 Internal EEProm and Settings

The BL233B has an internal 128 byte eeprom which contains configuration and default settings, and can optionally contain stored macros.

The BL233B datasheet has full details of EEProm configuration, macros and programming commands. The datasheet is the primary command reference. Always get the latest version of the datasheet from the website.

Some users have experienced problems with the timing of eeprom writes, and this document provides more specific details about programming.

You may be trying to get a new adaptor going in an old system, where the eeprom settings are long forgotten. This will tell you how to copy an eeprom from a good sample to a new unit.

Changing Baud Rate

More information about changing the Baud Rate is at: BL233B_Setting_Baud_Rate

Contents

1 Web Configuration Tool

There is now a web tool to help you create and change EEProm settings and macros.

www.i2cchip.com/i2c_front_panels/BL233_EEPROM_Configurator.html
It will generate the command strings to send to the BL233. You can copy the command strings it generates into Realterm's send tab. Alternatively you can copy to a command file to send during configuration or for some other purpose.

1.1 Direct Control using Realterm

- Only from Internet Explorer
- Install Realterm 2.0.70X or higher, as this has an extra activeX to support IE control.
- You will need to make www.i2cchip.com a Trusted Site in the internet security options.

1.2 Copying Commands To Realterm

Copy the strings for the configuration changes you want to make into a text file. (by default we use ".i2c" as an extension for these command files). In Realterm, set the character delay to 10, and send the command file.

2 Programming

Programming VDD range: 3.0 - 5.5V

Any byte in the EEProm can be written. Writes to each byte take 10ms. During this period, the BL233B cannot process characters and will simply lose any that are sent to it. This can easily corrupt the programming sequence. Each byte is written immediately the second hex data character is received.
Most EEProm settings such as baud rate, only have an effect at power on or reset
The EEWr_Protect bit in fSerial will prevent any further eeprom writes, unless "special pins mode" is used.

Timing Problem Symptoms

If you can program a single byte interactively from the keyboard, but multi-byte sequences are unreliable, this is likely a timing problem. If the BL233 appears dead after programming it you may need to use Special Pins Mode to reprogram it. See also EEProm Data Corruption

There are several approaches to reliable programming:

- Using Character Delay and a Programming File
- Using Pause ":" command
- Program as Single Bytes

2.1 Using Character Delay and a Programming File

The easiest (and recommended) approach is just to pad every character sent with a 10ms delay. This guarantees no problems. Handshaking does not need to be working, and you can write the whole eeprom array at once if you want, without worrying about the buffer size.

- Put all the eeprom programming commands into a text file
- Send from Realterms "Send" tab.
- You can set the delays from the command line, and even send the file directly from the command line.

```
Realterm chardly=10 flow=2 sendfile=eedata.txt
```

2.2 Using Pause ":" command

The pause ":" command makes the BL233B wait until it receives a linefeed before actioning the characters in the buffer. Precede the programming sequence with colon, ":" . Programming does not begin until a linefeed is received.

```
:V 60D0 454647484950<LF> ~<60ms delay before next char>
```

- Programming sequence must fit in the buffer (Buffer is 48 chars, so 16 bytes max per sequence)
- After the linefeed, you must not send any characters for 10ms per byte programmed

This is ideal for programming interactively from the keyboard or "send string" in realterm. For programming in production the character delay method is preferred.

2.3 Program as Single Bytes

If you are setting the eeprom from a computer program, then rather than sending a single V, then a block of bytes for the eeprom, and trying to somehow delay the characters as they are sent, you may find it is better to just send a complete program command for each individual byte, with a delay between. This has the advantage that any delay corruption only affects that single byte of eeprom, and doesn't corrupt every subsequent byte.

3 Dumping and Verifying

The U command dumps the *whole* EEProm as hex. Dumps start at F7 (fserial), not at address 0. You can capture the dump manually from the "Capture" tab in Realterm.
After capturing the file *DO NOT OPEN IT*. As soon as you open it with an editor it will be changed.
Rename the file, eg "eedump_ok.txt"
Now make a copy of this file, and edit this copy. Use the original, unedited file for doing file compares (below)

3.1 Programming and Verifying automatically with Realterm

If the programming file has a U command at the end then realterm can capture the dumped data. Realterm can do both automatically.

```
realterm chardly=10 flow=2 sendfile=eedata.txt capsecs=2 capquit=eedump.txt
```

Now eedump.txt has the whole eeprom data. By comparing this file with one captured from a known good unit you can verify. If you try to use the dos command FC (file compare) you will discover that it is useless in batch files as it does not return a result code. Get cmp.exe in the "unxutils" package instead. Note as above that if you edit (or even open with some Microsoft programs) "eedump_ok.txt" will be changed (reformatted, EOF added etc) and it will not compare correctly.

```
cmp.exe -c eedump.txt eedump_ok.txt
```

Putting these two together into a batch file will make a quick programming tool.

4 Duplicating and Production Programming

If you are changing eeprom configurations or . Sometimes you will find yourself trying to get an existing system working, where the original engineer has gone, and knowledge of how the I2C2PC adaptors was configured has been lost.

Production programming consists of

- Dump the contents of the eeprom from a good sample unit (seeDumping and Verifying) into a file.
- Copy this file, and edit the copy to add the required simple programming commands to the beginning.
- Program (seeProgramming) this into all new BL233's or I2C2PC's.

4.1 Editing the Dumped File

When you have captured the eeprom dump you can edit a copy to add

```
    V F708
```

to the start, and

```
    U
```

to the end of the file. See "bl233b_factory_eeprom_settings.dat"

The file is just plain text. The BL233 will ignore spaces and line breaks. Just ensure that your editor does not put any line breaks where they would split a pair of hex digits.(ie you should have an even number of hex chars on each line). Note warning above not to open or edit the dump file itself, but only a copy. You need the virginal dump file for doing file compares.

Be careful using Notepad, and don't try to use Word or Wordpad. It is best to use a programmers editor.

4.2 Production Programming Software

There are many ways to make a programming system. All they have to do is dump the file with a 10ms per character delay.
It is simple to use the commands above in a batch file to achieve this. An example production batch file is given in bl233b_factory_eeprom_settings_examples.zip. You will need to edit it for your file names, paths, and com port number. Some utilities are included, but you may need to get up to date versions from the net.

5 Special Pins Mode

Special Pins Mode may be able to rescue you from:

- Baud Rate set to something you can't talk to
- Start-up macro has and endless loop or crash preventing BL233 from ever reaching command mode.
- EEWr_Protect bit of fSerial has been set, preventing any further writes of eeprom.

In Special Pins Mode the baud rate and fSerial are in their factory state (57,600bd), and the start-up macro does not run. You will not get the hello message, as this is the start up macro. *Special Pins Mode does not restore the default eeprom settings, it only allows you to regain control so you can rewrite them from file.*

5.1 Entering Special Pins Mode

For BL233B it is only necessary to pull either Pin 5 (INT) to 0V or All of Pins1,2,17,18 to 0. Future versions will behave as described below. On I2C2PC, join pins 3 and 5 of I2C Bus#1.

To enter SPM, force all of pins 1,2,17,18, and 6 to 0V, during and 500ms after reset / power on. These pins should be pulled up if unused, to prevent accidental entry to SPM. For surface mount devices you may like to ensure that there is some way to access these pins if SPM is needed, eg pcb test points. This can be done through the I2C connectors on the I2C2PC adaptor. These pins are SDA1, SCL1, SDA2, SCL2, IRQ on the I2C2PC adaptor. On the I2C2PC adaptor you can make a special cable or use a hub connector that plugs into I2C bus#1 and I2CBus#2 and shorts the I2C cable pins 1,3,4,5 together.

6 EEProm Data Corruption

There have been a couple of reports of corrupted eeprom data. To avoid this we recommend:

- Make sure you have proper and adequate power supply bypassing especially the 100nF capacitor *at* the power pins.
- PCB: Crystal tracks are short, and the ground tracks from crystal capacitor is short and clean.

If your system is doing eeprom writes during its normal operational life (ie not a one-time factory configuration) .

- Make sure the 10ms/byte timing is *actually* being complied with. Otherwise corruption is expected.
- Don't write unless you need to. Dump the eeprom, check if changes are needed and only write if needed.
- After writing is done, park the EEProm data pointer to an unused location with the command "`VA05FM`". Note that the dUmp command leaves the pointer parked.

If your system is not doing eeprom writes during its normal operational life, and you experience eeprom corruption:

- At startup, park the EEProm data pointer to an unused location with an eeprom dump or the command "`VA05FM`"

7 Restoring the EEProm to Factory values

The file "bl233b_factory_eeprom_settings.dat" contains the data and commands to rewrite the whole eeprom to the factory state, then dump the eeprom. As described above, you may need to be in "special pins mode". The following command will send the reprogram the bl233b, and save a dump of the eeprom afterwards, all on one line. (Set the port number for your system)

```
"c:\program files\realterm\realterm" port=1 chardly=10 flow=2
     sendfile=bl233b_factory_eeprom_settings.dat capsecs=2 capture=bl233b_eeprom_dump.dat
```

"bl233b_factory_eeprom_settings.dat" contains:

```
V F708 210F660E08000D726E54343834393230343933323433323037363331333233313041 3C5432313C4E
4A30303053353431323334504A303030315335343132333450543231 3C53353431323334543438
3439575432313C54FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF54
36393C5437373C3C
U
```

8 Unique Identifiers, Serial Numbers, and Calibration Coefficients

The EEProm are be used to implement Unique Identifiers, Serial Numbers, Calibration Coefficients, and other data storage. There are two different approached to doing this:

- Use a macro with the typeback “T” command to send the data back. This is best for small amounts of data.
- Use the eeprom dump “U” command to dump the entire contents of the eeprom. The data is simply stored raw in free eeprom space. This will give better data density for a larger block.

Note that one significant advantage of using the dump approach, is that if an factory fresh adaptor is plugged in, you will not be trying to call random macro commands.

9 Macro Programming

The BL233B has the ability to perform simple macros. These can be used to

- macros for power-on, interrupt and watchdog actions
- Speed up commands by reducing long command sequences that must be sent to the adaptor to a 3 character call

The mechanism is that command strings are stored directly in eeprom. Note that when writing the eeprom using V, you must send each byte of eeprom as a 2 digit hex value. So if you want “`S40P`” in eeprom at 0x20, you will send “`V20DF 53343050`”

9.1 Macro Specific Commands

These commands are important for Macro use. See BL233B datasheet.

Char	Function	Explanation
:	Force return to RS232	Execution returns to RS232 buffer *unconditionally*
;	Return if Chars	Execution returns to RS232 buffer if there are chars in buffer, otherwise continues in EE
<	Return	Return to calling Macro or RS232 buffer
>	Jump to Address	Used to call a macro, or to loop and jump within macros
V	Write to EEProm	buffer
U	Dump whole EEProm	buffer

It would be good practice to fill unused eeprom with either non executing commands (eg 0x00), or *return* “<”

9.2 Looping Macros

Macros may be an endless loop using “>” to jump back to the beginning. It is recommended to use the “;” command within the loop, so it can be interrupted by sending any character.

9.3 Subroutines within Macros

9.3.1 Calling Depth

The call depth is only 1 deep. A macro can call a subroutine macro. A second return will always return to the RS232 buffer. A chain of macros can call each other without adverse effect, but the return stack is only 1 deep. It the case below, 20 will call 30; 30 calls 40; 40 returns to 30; but 30 will return to the RS232 buffer, not the calling macro 20 as the stack depth was too great.

```
0x20: >30<
0x30: >40<
0x40: <
```

9.3.2 Examples

We want to make a macro to that reads all 8 channels of a MAX127, with commas between values. To read channel 0 we need this command:

```
S5080PR02P,
```

And to for each subsequent channel we change the channel select byte from 80 to 90,A0,B0 etc. Each channel takes 11 bytes of eeprom. So reading 8 channels will take 88 bytes. We could make this better by using W for subsequent reads:

```
S5080PR02P, W90PR02P, WA0PR02P  ~... etc
```

So this reduces it to 74 bytes. But lets break this up.

```
PR02P,<
```

is common to all channels, so make it a subroutine macro at 0x20. Now the main macro (at 0x30) will call it with >20

```
S5080>20 W90>20 WA0>20 ~... etc
```

Now we have reduced it to 50 bytes. We have also sped it up. To read the whole ADC required 74 bytes to be sent previously, now a 3 character call is all that is required.

As an endless loop this will read 4 channels on a line, comma separated, with a 1 second pause.

```
S5080>20 W90>20 WA0>20 WA0>20 L0200.;>30
```

9.4 Standard Macros

As supplied the eeprom has some macros stored in it. The power on macro is always located at 00. The interrupt and watchdog macros have vectors, so they can be moved. Always ensure that there is a valid macro at the vector, a return “<” is the minimum.

Address	Macro		Function
00	Power On		Shows welcome message
1A			types “!”
1E			
6F	Default Interrupt	T69<	types “i”
73	Default Watchdog	T77<	type “w”
76	End	<	should always be <

9.4.1 Getting the Version string

>00 will make the power on macro run, and display the welcome and version string. Note that if you have overwritten the power on macro with something different, then it cannot run.

9.5 EEProm Addresses

These are the Address/Complement pairs for various addresses

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
F0								F708	F807	F906	FA05	FB04	FC03	FD02	FE01	FF00
00	00FF	01FE	02FD	03FC	04FB	05FA	06F9								0EF1	
10	10EF														1EE1	
20	20DF															
40																
70																

10 BL233 Versions

BL233B has welcome string with “V118”

10.1 BL233C

When released the BL233C will have the following improvements to EEProm operation:

- Receive characters during eeprom byte programming.. 10ms delays no longer needed. However the procedure described here will always work correctly with all parts, regardless of handshaking and buffer size, so it is to be recommended.
- Special Pins mode works as originally described, so needs to be done properly.
- Data pointer parking to reduce chances of spontaneous corruption, using pointer parking will be pointless.

Note that there is no need or advantage to changing anything that works on BL233B.

11 Revision History

Rev	Date	
0	28 June 06	
1	31 May 09	
2	8 Feb 12	Added information about duplicating eeprom from one BL233 to another.
3	14 Oct 12	Correct CHARDLY parameter in example command lines. Add Web configurator link
4	2 Nov 13	Corrected Special pins mode I2C pin# for interrupt
5	21 Apr 15	Add programming VDD range. Add note about single byte programming, and avoiding corruption
6	14 Jan 17	SignPDFS, colorise BL233 code