BitCloud Software 1.2

Serial Bootloader User's Guide

Doc. P-ZBN-451~02 v.1.2 September 2008

Disclaimer

MeshNetics believes that all information is correct and accurate at the time of issue. MeshNetics reserves the right to make changes to this product without prior notice. Please visit MeshNetics website for the latest available version.

MeshNetics does not assume any responsibility for the use of the described product or convey any license under its patent rights.

MeshNetics warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with MeshNetics standard warranty. Testing and other quality control techniques are used to the extent MeshNetics deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

Trademarks

MeshNetics®, ZigBit, BitCloud, SensiLink, as well as MeshNetics and ZigBit logos are trademarks of MeshNetics Ltd.

All other product names, trade names, trademarks, logos or service names are the property of their respective owners.

Technical Support

Technical support is provided by MeshNetics.

E-mail: support@meshnetics.com

Please refer to Support Terms and Conditions for full details.

Contact Information

MeshNetics

EMEA Office
Am Brauhaus 12
01099, Dresden, Germany
Tel: +49 351 8134 228
Office hours: 8:00am - 5:00pm (Central European Time)
Fax: +49 351 8134 200

US Office
5110 N. 44th St., Suite L200
Phoenix, AZ 85018 USA
Tel: (602) 343-8244
Office hours: 9:00am - 6:00pm (Mountain Standard Time)
Fax: (602) 343-8245

Russia Office
9 Dmitrovskoye Shosse, Moscow 127434, Russia
Tel: +7 (495) 725 8125
Office hours: 8:00am - 5:00pm (Central European Time)
Fax: +7 (495) 725 8116

E-mail: info@meshnetics.com

www.meshnetics.com

Table of Contents

1. Introduction

Serial Bootloader is a part of BitCloud™ software [1] that consists of two parts: embedded bootstrap code, running on the ATmega1281 (or similar) MCU of the ZigBit™ module [2] and PC (or similar platform)-based application. PC application sends data to the embedded bootstrap over serial link. Embedded application uses the received data to program the internal flash memory and/or EEPROM of the MCU. A simple communication protocol is used to ensure proper programming. Motorola S-record (SREC) format files are used as source images for the serial bootloader PC part.

Related documents:

[1] BitCloud™ Product Datasheet. MeshNetics Doc. M -252~08

[2] ZigBit™ OEM Modules. Product Datasheet. MeshNetics Doc. M-251~01

[3] ZigBit™ Development Kit User's Guide. MeshNetics Doc. S-ZDK-451

[4] ZigBit™ Amp Development Kit User's Guide. MeshNetics Doc. S-ZDK-451~02

[5] ZigBit™ 900 Development Kit User's Guide. MeshNetics Doc. S-ZDK-451~03

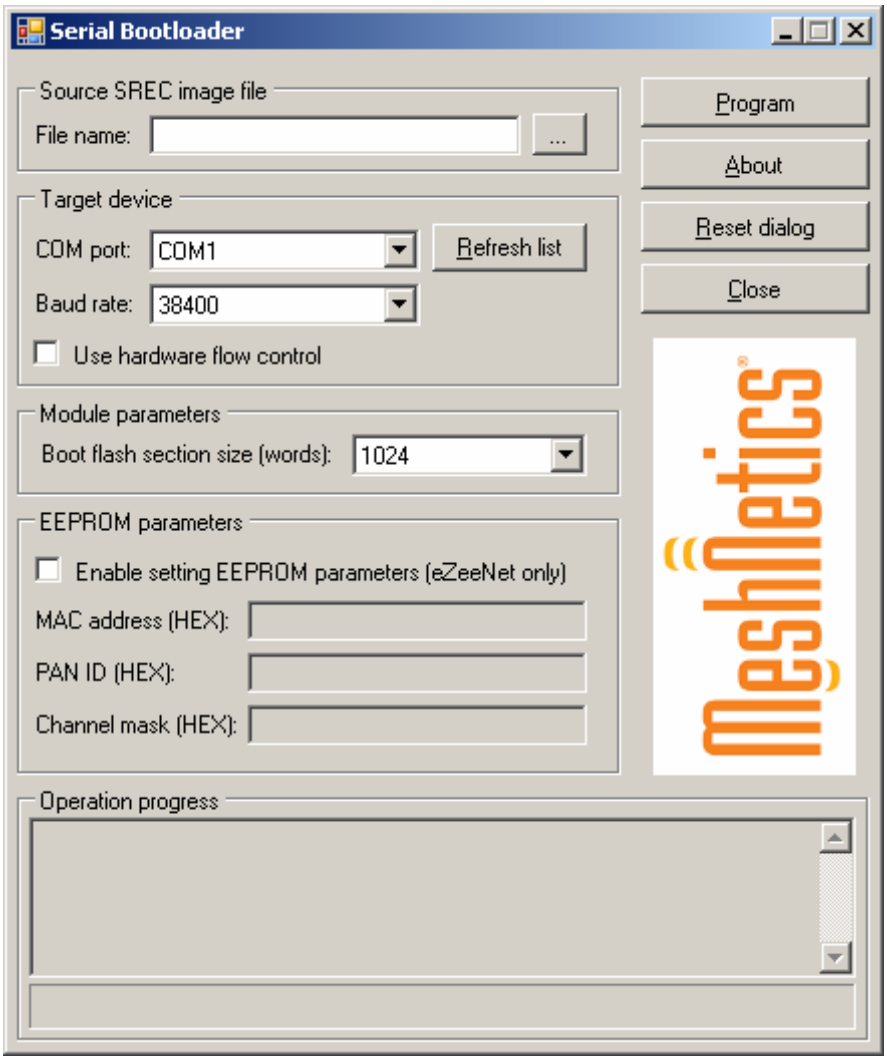
[6] CP210x USB to UART Bridge VCP Drivers. http://www.silabs.com/tgwWebApp/public/web_content/products/Microcontrollers/USB/en/mcu_vcp.htm

[7] AVR Studio User Guide. Available in HTML Help with the product. http://www.atmel.com/dyn/products/tools_card.asp?tool_id=2725

[8] JTAGICE mkII Quick Start Guide. http://www.atmel.com/dyn/resources/prod_documents/doc2562.pdf

2. Application Description

Serial Bootloader is a stand-alone utility included in the ZigBit™ Development Kit [3], ZigBit™ Amp Development Kit [4] and ZigBit™ 900 Development Kit [5]. Its PC portion is supplied in both GUI and console versions. The console version does not require any special installation and may be just copied from your distribution set to the desired location on your PC. The GUI version (pictured below) must be installed prior to use.



The firmware part of Serial Bootloader may be uploaded to the device using Atmel JTAGICE mkII emulator [8].

There are some minor restrictions on software downloadable by serial booting process. Serial Bootloader cannot rewrite the upper 2 KB of memory starting from `0xFC00` address, because the bootstrap code resides in that area.

3. Input File Format

Serial Bootloader recognizes image files in Motorola S-record hexadecimal format, also known as SREC or S19 format. Such file names have the `.srec` extension. Motorola SREC files for Serial Bootloader contain both flash memory and EEPROM images.

A user's application developed with AVR Studio can be converted into SREC format using the AVR-`objcopy` utility so it becomes downloadable via serial booting process.

The SREC format is an ASCII encoding for binary data that has several advantages over binary formats. ASCII encoding allows the files to be edited with a text editor. Also, each record contains a checksum to identify data that has been corrupted in transmission.

A SREC file consists of a series of ASCII records. All hexadecimal (hex) numbers are big-endian. The records have the following structure:

1. Start code (one character): "S".
2. Record type (one digit, 0 to 9) defining the type of the data field.
3. Byte count (two hex digits), indicating the number of bytes (pairs of hex digits) that follow in the rest of the record (in the address, data and checksum fields).
4. Address (four, six, or eight hex digits, as determined by the record type), indicating the position in memory for the data.
5. Data (2n hex digits) – n bytes of the data.
6. Checksum (two hex digits) – the complement of bytewise sum of byte count, address, and data fields.

There are eight record types, listed below:

Record	Description	Address bytes	Data sequence
S0	Block header	2	Yes
S1	Data sequence	2	Yes
S2	Data sequence	3	Yes
S3	Data sequence	4	Yes
S5	Record count	2	No
S7	End of block	4	No
S8	End of block	3	No
S9	End of block	2	No

The S0 record data sequence contains vendor specific data rather than program data. The record count in the S5 record is stored in the 2-byte address field. The address field of the S7, S8, or S9 records may contain a starting address for the program.

Example SREC file:

```
S00F0000068656C6C6F202020202000003C
S11F00007C0802A6900100049421FFF07C6C1B787C8C23783C6000003863000026
S11F001C4BFFFFE5398000007D83637880010014382100107C0803A64E800020E9
S111003848656C6C6F20776F726C642E0A0042
S5030003F9
S9030000FC
```

Start code | Record type | Byte count | Address | Data | Checksum

4. Serial Bootloader Programming Process

Here's a brief description of the serial bootloading process:

On ZigBit side:

1. After module is reset (eg. by pressing the "Reset" button on the MeshBean board) bootstrap code waits 500ms for HANDSHAKE_REQ data sequence. If no HANDSHAKE_REQ is received, bootstrap jumps to the entry point of the program in flash memory (if any).
2. If a HANDSHAKE_REQ sequence is received, bootstrap code sends a HANDSHAKE_CONF sequence and starts receiving SREC records.
3. For every valid SREC record bootstrap responds with an ACK data sequence.
4. In case of any error during loading process, bootstrap code sends a NACK data sequence, then proceeds to (1).

On PC side:

1. PC bootloader sends HANDSHAKE_REQ data sequence for 30 seconds with 200ms interval, waiting for HANDSHAKE_CONF data sequence between sends. Any reply except HANDSHAKE_CONF is ignored.
2. If HANDSHAKE_CONF is received, PC bootloader starts sending data from the SREC file via serial link. Each record from the SREC file is converted to binary representation before sending.
3. For every record sent an ACK is expected to be received over the serial link in return. If a NACK sequence is received or a timeout occurs, PC bootloader aborts.

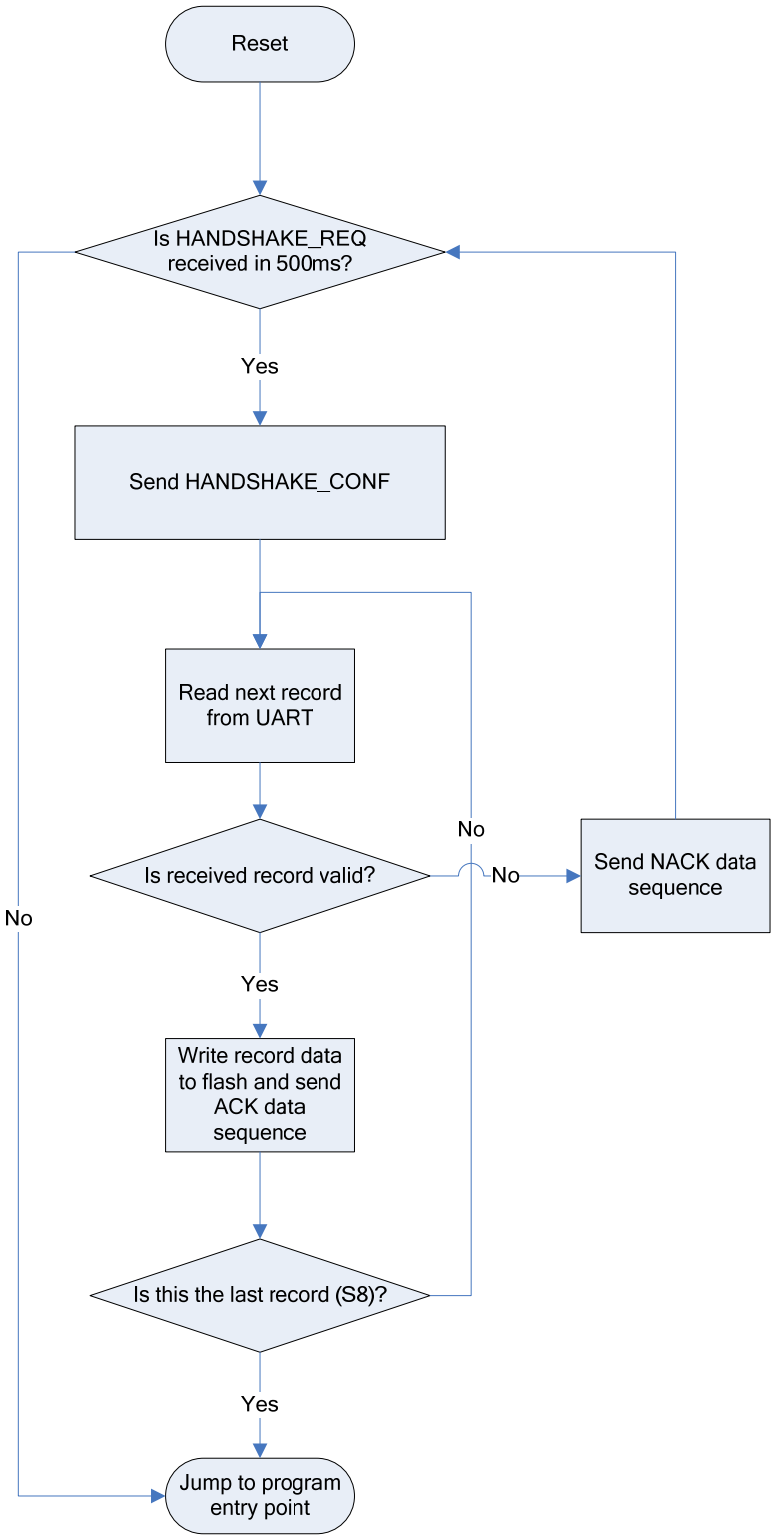
Data sequences used:

- HANDSHAKE_REQ: 0xB2, 0xA5, 0x65, 0x4B
- HANDSHAKE_CONF: 0x69, 0xD3, 0xD2, 0x26
- ACK: 0x4D, 0x5A, 0x9A, 0xB4
- NACK: 0x2D, 0x59, 0x5A, 0xB2

Binary representation of the SREC record:

Field name	Length (bytes)
Record type	2
Address length	1
Address	4
Record size	1
Record data	Variable
Boot section size	1024

Here's a flowchart of the serial bootloader programming process:



5. Using the Serial Bootloader

In order to program a wireless device using Serial Bootloader, do the following:

1. Connect a device to your PC via USB or RS-232 port following the operating instructions. In order to use USB port for serial connection between a device and PC you need to install a virtual COM port driver from Silicon Laboratories [6].

NOTE:

Once Windows detects new hardware, the driver installation wizard appears. Follow the on-screen instructions. When installation is completed, make sure the driver is installed successfully and the new COM port is present in the hardware list. For that purpose invoke the Device Manager: Start → Control Panel → System → Hardware → Device Manager and check 'Ports (COM&LPT)' item.

2. Run Serial Bootloader (either console or GUI version), specifying the image file, COM port and other options (if required).
3. Press reset button on the device
4. Release reset button on the device. Serial Bootloader will be waiting for approximately 30 seconds for the button to be released. If this does not happen, programming would be aborted.

Serial Bootloader indicates the operation progress. Once loading is finished successfully, the device would be restarted automatically. If loading fails, Serial Bootloader would indicate the reason. In rare cases, loading process could fail due to communication errors between the device and the PC. If this happens, try to repeat programming or try to use normal RS-232 port instead of USB. If loading still fails, the previous code programmed into the device could be corrupted, and the device should be reprogrammed again.

Setting Fuse Bits

To enable programming of a node by Serial Bootloader fuse bits are to be set up for ZigBit as follows: **`0xFF, 0x9C, 0x62`**.

In order to ensure these fuse bits are set, check ON the following options in Fuses Tab using AVR Studio [7]:

```
Brown-out detection disabled; [BODLEVEL=111]
JTAG Interface Enabled; [JTAGEN=0]
Serial program downloading (SPI) enabled; [SPIEN=0]
Boot Flash section size=1024 words Boot start
address=$FC00;[BOOTSZ=10]
Divide clock by 8 internally; [CKDIV8=0]
Int. RC Osc.; Start-up time: 6 CK + 65 ms; [CKSEL=0010
SUT=01]
Boot Reset vector Enabled (default address=$0000);
[BOOTRST=0]
```

Uncheck the rest of options, and write the fuse bits to device. Make sure the above hex value string appears at the bottom of Fuses Tab.

It is recommended to use Atmel JTAGICE mkII emulator [8] together with AVR Studio to set up fuse bits.

NOTE:

Be careful using JTAG for setting up the fuse bits. If you set wrong fuse bits by JTAG, your device would not work.

Setting MAC Address

To communicate within WSN network each node must be identified with a unique MAC address. In general, MAC address can be specified for a node in the following way:

1) by hardware pre-configuration

2) by loading an image file containing MAC address to a node

3) by means of Serial Bootloader command options.

Command Options

The console version of Serial Bootloader accepts the following command-line options (may be applied in any order):

```
bootloader –p port_number [–f file_name] [-b baud_rate] [-h]
[-s bootstrap_size] [-M MAC address] [-P PANID]
[-C Channel mask]
```

Any of the options except `-p` can be omitted.

The GUI version contains interface controls for the same set of options. The description of the options is given in the table below.

Option	GUI control	Description	Default
`-f`	`File name`	Name of Motorola SREC file	
`-p`	`COM port`	COM port	
`-b`	`Baud rate`	Baud rate in bits per second (1200, 2400, 4800, 9600, 19200, 38400, 57600, or 115200)	38400
`-h`	`Use hardware flow control`	Hardware flow control, if used	None
`-s`	`Boot flash section size`	The size of bootstrap code, in words (512, 1024, 2048, or 4096)	1024
`-M`	`MAC address (HEX)`	MAC address in HEX format to be assigned to the node	
`-P`	`PAN ID (HEX)`	PANID in HEX format to be assigned to the network	
`-C`	`Channel mask (HEX)`	Channel mask in HEX format to be assigned to the network	

NOTE:

Serial Bootloader is designed so that, if `-M`, `-C` or `-P` option is present in the command line, the corresponding parameter stored in the EEPROM would be overwritten. Furthermore, the relevant value(s) which was set inside the downloaded image file will be ignored.

If –f option is not specified (i.e. without image file download) you can use any other command option to change the EEPROM settings of the node without interfering with the application code downloaded before.

Usage Examples

```
bootloader –f wsndemo.srec –p COM5 –M 1 –C 100000 –P 5320
```

The above command demonstrates how to load the WSN Demo image into a node connected to PC via COM5. The following parameters are assigned:

MAC address = `0x1`
Channel mask = `0x100000`
PANID = `0x5320.`

Serial Bootloader can be used apart from downloading any image:

```
bootloader –p COM5 –M 2 –C 100000 –P 5320
```

The above command is used to assign the following parameters to a node without affecting an image:

MAC address = `0x2`
Channel mask = `0x100000`
PANID = `0x5320.`

To set baud rate, flow control mode or the bootstrap code size to the default value, omit the corresponding option in command line.