Application Note, V 1.0, August 2008

AP08083

XC878 Flash Download Using Bootstrap Loader

Microcontrollers

Never stop thinking

Edition 2008-08

Published by Infineon Technologies AG,
St.-Martin-Strasse 53,
81669 München, Germany

Information

For further information on technology, delivery terms and conditions and prices please contact your nearest Infineon Technologies Office (**www.infineon.com**).

Warnings

Due to technical requirements components may contain dangerous substances. For information on the types in question please contact your nearest Infineon Technologies Office.

Infineon Technologies Components may only be used in life-support devices or systems with the express written approval of Infineon Technologies. Life support devices or systems are intended to be implanted in the human body, or to support and/or maintain and sustain and/or protect human life. If they fail, it is reasonable to assume that the health of the user or other persons may be endangered.

Revision History: **2008-08** V 1.0

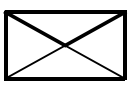
Previous Version: none

Version	Subjects (major changes since last revision)
V 1.0	Initial release for XC878

We Listen to Your Comments

Any information within this document that you feel is wrong, unclear or missing at all? Your feedback will help us to continuously improve the quality of this document. Please send your proposal (including a reference to this document) to:

mcdocu.comments@infineon.com



1 Introduction

A built-in bootstrap loader (BSL) is implemented in the XC800 family to provide a mechanism to load data / code into the internal memory of the device (XRAM or FLASH) via a UART interface. In variants that support external memory, it is also possible to load data / code to the external memory.

The protocol used in the XC800 family is standard, although there might be a slight variation due to different structures of the flash memory.

1.1 Overview

This document provides a cookbook on the BSL mode for XC878. It will show detailed steps on flash downloading.

In general, XC878 devices have 2 types of variants for BSL mode:

- LIN Variant (e.g: XC878L)
- UART Variant (e.g: XC878)

The access modes for these 2 variants are different.

The user's manual contains detailed information about the BSL mode protocol.

This example code is additionally provided for reference. Although the code has been tested, there is no warranty provided.

There are four files containing the example code:

xc800_bsl.cpp	: code that contains the API for BSL mode
xc800_bsl.h	: corresponding header file
xc800.cpp	: main example code on how to use the API
xc800_verify.h	: verification code to be loaded into XRAM

These files can be compiled in Visual C 6.0.

Please note that the example code also provides routines for other devices of the XC800 family. Thus routines dedicated to the XC878 device are indicated by the "_xc878" suffix.

In order to run the example code, the PC host needs to be connected to a starterkit with a UART cable.

The following files may also be included for tips and recommendations:

verify.a51	: assembly code to verify the flash content (UART).
verify.hex	: hex file generated from the above assembly code.
verify_lin.a51	: assembly code to verify the flash content (LIN).
verify_lin.hex	: hex file generated from the above assembly code.

2 Supported BSL Mode in XC878

In order to gain access to the **BSL mode**, the chip must be **reset with MBC and TMS pins pulled low externally**.

The following hardware pins are used:

P1.0: Used as RXD (IN)

P1.1: Used as TXD (OUT)

Note: Please refer to chapter 7.2.3 in the user's manual for detailed information on the booting scheme and entering BSL mode.

2.1 UART Mode

For UART variants the following steps have to be done by the PC Host:

1. Send the first byte: **0x80** for the auto baud rate recognition.

   Note: Because of the accuracy, it is recommended to use only a range of baud rate from **9600** to **57600** bit/second.
2. Wait for 1 byte **acknowledgement**: **0x55**
3. Send the **bsl_erase_flash_xc878()** instruction to erase the flash memory.

   The function uses the option for mass erase. All Program Flash[1)] (P-Flash) and Data Flash[1)] (D-Flash) memory is erased at once.

   Flash erase will commence before the acknowledgement is sent from the microcontroller. Hence, there is a waiting time for about **100 ms**.

   Wait for 1 byte acknowledgement: 0x55 (embedded inside the function).
4. Send the **bsl_uart_header()** instruction.

   Parameter to be used:

   bslHeader.mode = 2

   bslHeader.dataLength = 32[2)]

   bslHeader.startAddr[3)] = <Starting Address - must by 32-byte aligned>

   Wait for 1 byte acknowledgement: 0x55 (embedded inside the function).

   Details about the header block and the type of acknowledgement is described in **Section 5.4** or chapter 19 in the user's manual.

1) Refer to for **Section 5.2** for further information on the flash memory organization.

2) The maximum data length for D-Flash programming is 32 bytes. If only P-Flash is programmed, the data maximum length can be set to 64 bytes. No crossing of a wordline boundary is allowed (see **Section 5.2**).

3) The Starting Address is set to 0x0000 by default.

5. Send the **bsl_uart_data()** instruction.

   Parameter to be used:

   bslData.dataLength = 32

   bslData.cDataArray = <Pointer to the Data Array>

   It is recommended to send **only 32 bytes** of data for flash programming at once.

   Flash programming will commence before the acknowledgement is sent from the microcontroller. Hence, there is a waiting time for about **2 ms**.

   Wait for 1 byte acknowledgement: 0x55 (embedded inside the function).

6. Repeat the data sending until all data is sent. The flash memory will be programmed continuously.
7. Send the **bsl_uart_eot()** instruction.

   This routine must have an empty data buffer (when this API is used for flash download).

   Parameter to be used:

   bslEOT.dataLength = 32

   bslEOT.lastCodeLength = 0[1)]

   Wait for 1 byte acknowledgement: 0x55 (embedded inside the function).

Note: Step 3 to Step 7 are performed in the function bsl_file_download_xc878() (see API in chapter 4).

2.2 LIN Mode

For LIN variants, the following steps have to be done by the PC Host:

1. Send the **bsl_lin_header()** instruction to initialize LIN communication.

   Parameter to be used:

   bslHeader.NAD = 0x7F

   bslHeader.dwBaudrate = <baud rate>

   bslHeader.startAddr = 0xF000

   bslHeader.mode = 0

   bslHeader.fastLin = 1 (to enter the Fast LIN mode)

   Wait for 9 bytes acknowledgement (embedded inside the function).

   The routine will check that the **second byte** has a value of **0x55** to indicate a success.

1) In this example code all the program code bytes are transmitted in data blocks.

Note: Because of the accuracy, it is recommended to use only a range of baud rate from **9600** to **57600** bit/second.

Fast LIN BSL is an enhanced feature in XC878 devices, supporting higher baud rates of up to 57.6[1] kbit/second.

When Fast LIN BSL mode is entered, the microcontroller will **switch to the BSL UART protocol** at the calculated baud rate. The microcontroller will stay at FAST LIN BSL and the **communication structure will be the same as in UART BSL mode**.

2. Send the **bsl_erase_flash_xc878()** instruction to erase the flash memory.

   The function uses the option for mass erase. All Program Flash (P-Flash) and Data Flash (D-Flash) memory is erased at once.

   Flash erase will commence before the acknowledgement is sent from the microcontroller. Hence, there is a waiting time for about **100 ms**.

   Wait for 1 byte acknowledgement: 0x55 (embedded inside the function).

3. Send the **bsl_uart_header()** instruction.

   Parameter to be used:

bslHeader.mode	= 2
bslHeader.dataLength	= 32[2]
bslHeader.startAddr[3]	= <Starting Address - must by 32-byte aligned>

   Wait for 1 byte acknowledgement: 0x55 (embedded inside the function).

   Details about the header block and the type of acknowledgement is described in **Section 5.4** or chapter 19 in the user's manual.

4. Send the **bsl_uart_data()** instruction.

   Parameter to be used:

bslData.dataLength	= 32
bslData.cDataArray	= <Pointer to the Data Array>

   It is recommended to send **only 32 bytes** of data for flash programming at once.

   Flash programming will commence before the acknowledgement is sent from the microcontroller. Hence, there is a waiting time for about **2 ms**.

   Wait for 1 byte acknowledgement: 0x55 (embedded inside the function).

5. Repeat data sending until all data is sent. The flash memory will be programmed continuously.

---

1) This is higher than the Standard LIN, which supports only baud rates of up to 20 kbit/second.

2) The maximum data length for D-Flash programming is 32 bytes. If only P-Flash is programmed, the data maximum length can be set to 64 bytes. No crossing of a wordline boundary is allowed (see **Section 5.2**).

3) The Starting Address is set to 0x0000 by default.

6. Send the **bsl_uart_eot()** instruction.

   This routine must have an empty data buffer (when this API is used for flash download).

   Parameter to be used:

   bslEOT.dataLength = 32

   bslEOT.lastCodeLength = 0[1)]

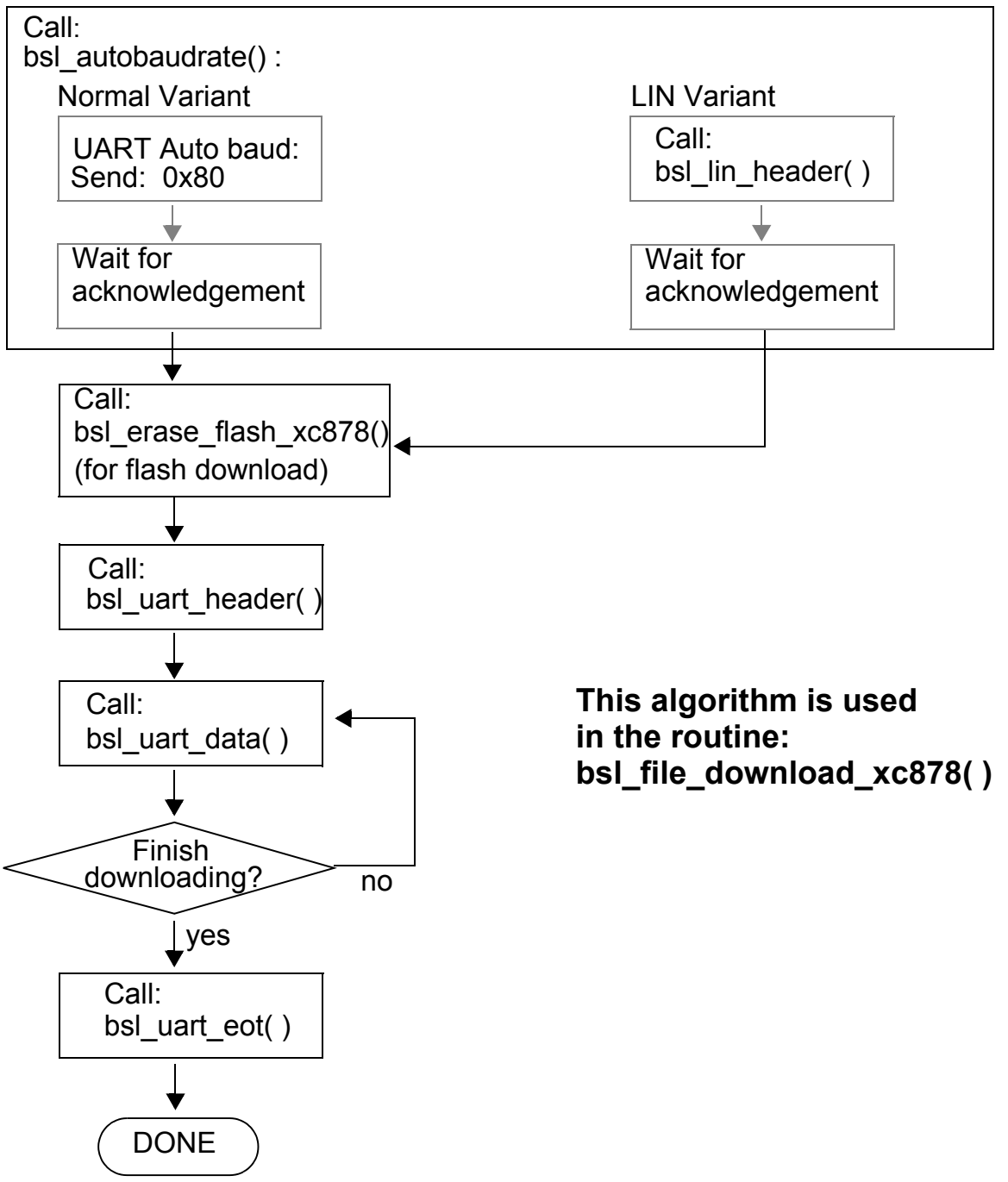
Wait for 1 byte acknowledgement: 0x55 (embedded inside the function).

Note: Step 2 to Step 6 are performed in the function bsl_file_download_xc878() (see API in chapter 4).

Note: For the LIN variant in single wire mode, the HOST will receive its sent bytes as an echo from the device immediately after sending. This echo must be taken care of.

1) In this example code all the program code bytes are transmitted in data blocks..

2.3 BSL Diagram for Code Download



This algorithm is used in the routine: bsl_file_download_xc878( )

3 Flash Protection and Unprotection

XC878 allows the flash to be protected by password after downloading the program code.

After the flash protection is done, no further flash downloading will be possible. External access to the device, including the flash, will be blocked.

To protect the flash, the following instruction is sent:

1. Send **bsl_uart_header()** instruction.

   Parameter to be used:

   bslHeader.mode = 6

   bslHeader.password = <2-bytes password>

When flash is not protected yet, the microcontroller will enable the flash protection scheme. Protection mode will be activated and the password will be stored by the device.

When flash is already protected, the microcontroller will deactivate all flash protection, if the user-password matches the stored password. The protected **P-Flash block will be automatically erased** and the stored password will be reset. The D-Flash erasure depends on bit 12 of the user-password as descibed in chapter 3.3.1 in the user's manual.

The flash unprotection will be valid at the next power-up or hardware reset.

Please refer to chapter 3.3.1 of the user's manual for further information on the flash memory prtotection.

4 API Description of XC800_BSL

4.1 Data Type Structure in XC800_BSL

```
typedef struct BSL_HEADER {
    unsigned char NAD;              //  NAD for LIN.
    unsigned char mode;             //  0  = Download to XRAM
                                        1  = Run from XRAM (0xF000)
                                        2  = Download to FLASH
                                        3  = Run from FLASH (0x0000)
                                        6  = Password protect/unprotect
                                        16 = Erase Flash
    unsigned int  startAddr;        //  Starting Address for mode 0 and 2.
    unsigned char dataLength;       //  UART Mode:
                                         Data Byte Length to be
                                         written in a subsequent
                                         DATA Block.
                                        LIN Mode:
                                         The number of subsequent data
                                         blocks that will be sent.
    unsigned char fastLIN;          //  Only for LIN mode (0 = Normal
                                        LIN, 1 = Fast LIN)
    unsigned short password;        //  1 byte password to protect
                                        or unprotect Flash (Mode 6 only)
    DWORD             dwBaudrate;   //  Baud rate (Only necessary for
                                        LIN Mode)
    unsigned char singlewire;       //  Set to 1 if single wire connection
                                        is used.
    unsigned char waitNoResponse;   //  Only for LIN mode.
                                        If set to 1, it will not wait for
                                        the acknowledge
    unsigned char ucDeviceType;     //  Specify the device type
} BSL_HEADER;


typedef struct BSL_DATA {
    unsigned char NAD;              //  NAD for LIN.
    unsigned char *cDataArray;      //  Pointer to the data to be loaded.
    unsigned char dataLength;       //  Data Byte Length to be loaded (MUST
                                        be the same value as in BSL HEADER)
    DWORD         dwBaudrate;       //  Baud rate (Only necessary for LIN
                                        Mode)
    unsigned char singlewire;       //  Set to 1 if single wire connection
                                        is used.
} BSL_DATA;
```

```
typedef struct BSL_EOT {
    unsigned char NAD;              //  NAD for LIN.
    unsigned char *cDataArray;      //  Pointer to the data to be loaded.
    unsigned char lastCodeLength;   //  Data Byte Length to be loaded
                                        (lastCodeLength < dataLength).
    unsigned char dataLength;       //  Data Byte Length as stated in BSL
                                        HEADER
    DWORD         dwBaudrate;       //  Baud rate (Only necessary for LIN
                                        Mode)
    unsigned char singlewire;       //  Set to 1 if single wire connection
                                        is used.
} BSL_EOT;


typedef struct BSL_DOWNLOAD {
    char *hexFileName;              //  Hex File Name.
    unsigned eraseFlash;            //  0 = Flash will NOT be erased before
                                            downloading
                                    //  1 = Flash  will  be  erased  before
                                            downloading
    unsigned eraseOnly;             //  0 = Flash will be downloaded after
                                            erasing.
                                    //  1 = Flash  will  NOT  be  downloaded
                                            after erasing.
    unsigned verbose;               //  0 = No message will be displayed.
                                    //  1 = (Default)  Message  will  be
                                            displayed.
    unsigned *xram_valid;           //  0 = No  Download  to  externally
                                            mapped XRAM is done
                                    //  1 = Download to externally mapped
                                            XRAM is done
    unsigned char singleWire;       //  Set to 1 if single wire connection
                                        is used.
    unsigned char ucDeviceType;     //  Specify the device type
} BSL_DOWNLOAD;


typedef struct BSL_ERASE {
    unsigned bankNumber;            //  Bank Number to indicate the bank to
                                        be erased.
    unsigned sectorNumber;          //  Sector Number to indicate the
                                        sector to be erased.
    unsigned char option;           //  Not used for XC878
    unsigned char singleWire;       //  Set to 1 if single wire connection
                                        is used.
    unsigned char ucDeviceType;     //  Specify the device type
} BSL_ERASE;
```

```
typedef struct AUTO_BAUDRATE {
    unsigned char detection;       //  Detection Mode:
                                   //  0 = UART Mode (No Auto Detection)
                                       1 = LIN  Mode (No Auto Detection)
                                       2 = Auto Detection Mode
                                        Byte 0x80 will be send first
                                        and wait for 10ms.
                                        If no response, then proceed
                                        with LIN auto detection.
    unsigned char mode;            //  Same as BSL HEADER
    unsigned char fastLIN;         //  Only for LIN mode (0 = Normal LIN,
                                       1 = Fast LIN)
    DWORD         dwBaudrate;      //  Baud rate
    unsigned char singleWire;      //  Set to 1 if single wire connection
                                       is used.
    unsigned char waitNoResponse;  //  Only for LIN mode.
                                       If set to 1, it will not wait for
                                       the acknowledge
    unsigned char nac;             //  no activity count
    unsigned char nad;             //  NAD for LIN mode
    bool bLIN;                     //  Set to “true“ if LIN device
    bool bReset;                   //  enable reset by bsl_autobaudrate()
    unsigned char ucDeviceType;    //  Specify the device type
} AUTO_BAUDRATE;
```

4.2 Function Prototypes in XC800_BSL

```
/*-------------------------------------------------------------------
Function Name          : bsl_init_uart()
Description            : Responsible to initialize the UART Port (COM
                         PORT)
                         Following actions are done:
                         -) Initialize the chosen COM PORT with the
                         selected baud rate
                         -) Return the hComm handle.
                       The parameter *uiError will be updated accordingly.
Function Called        : None
Input Parameter        : *cPortName     => Port Name (e.g: "COM1", "COM2"
                                            etc)
                         dwBaudrate     => Baud rate
Output Parameter       : *hComm         => Valid Communication Handle
                         *uiError       => Error Code
Return Value           : None
-------------------------------------------------------------------*/
```

```
/*---------------------------------------------------------------------
Function Name         : bsl_autobaudrate()
Description           : Sending the first byte for auto baud rate
                        detection.
                        If UART or LIN auto detection is enabled, the
                        function will first send byte: 0x80 to detect if
                        it is UART or not.  If there is no responds within
                        the specified time-out value, then send the first
                        LIN header (the chip must be reset again!!)
Function Called       : bsl_lin_header()
Input Parameter       : *hComm        => Communication Handle.
                        *autobaud     => AUTO_BAUDRATE structure.
Output Parameter      : *autobaud     => AUTO_BAUDRATE structure.
                        *uiError      => Error Code.
Return Value          : None
---------------------------------------------------------------------*/

/*---------------------------------------------------------------------
Function Name         : bsl_lin_header()
Description           : Sending the LIN Header Block.
Function Called       : None
Input Parameter       : *hComm        => Communication Handle.
                        bslHeader     => BSL_HEADER structure.
Output Parameter      : *uiError      => Error Code
Return Value          : None
---------------------------------------------------------------------*/

/*---------------------------------------------------------------------
Function Name         : bsl_erase_flash_xc878()
Description           : Responsible to erase the Flash memory
Input Parameter       : *hComm        => Communication Handle.
                        bslErase      => BSL_ERASE Structure
Output Parameter      : *uiError      => Error Code
Return Value          : None
---------------------------------------------------------------------*/

/*---------------------------------------------------------------------
Function Name         : bsl_uart_header()
Description           : Sending the UART Header Block.
Function Called       : None
Input Parameter       : *hComm        => Communication Handle.
                        bslHeader     => BSL_HEADER structure.
Output Parameter      : *uiError      => Error Code
Return Value          : None
---------------------------------------------------------------------*/
```

```
/*---------------------------------------------------------------------
Function Name          : bsl_uart_data()
Description            : Sending the UART Data Block.
Function Called        : None
Input Parameter        : *hComm         => Communication Handle.
                         bslData        => BSL_DATA structure.
Output Parameter       : *uiError       => Error Code
Return Value           : None
---------------------------------------------------------------------*/

/*---------------------------------------------------------------------
Function Name          : bsl_uart_eot()
Description            : Sending the UART EOT Block.
Function Called        : None
Input Parameter        : *hComm         => Communication Handle.
                         bslEOT         => BSL_EOT structure.
Output Parameter       : *uiError       => Error Code
Return Value           : None
---------------------------------------------------------------------*/

/*---------------------------------------------------------------------
Function Name          : close_interface()
Description            : Responsible to close all of the communication
                         channel (UART or JTAG)
Function Called        : None
Input Parameter        : *hComm         => Communication Handle.
Output Parameter       : *uiError       => Error Code
Return Value           : None
---------------------------------------------------------------------*/

/*---------------------------------------------------------------------
Function Name          : file_download_xc878()
Description            : The function to download the hex file.
Function Called        : bsl_erase_flash_xc878(),bsl_uart_header()
                         bsl_uart_data(), bsl_uart_eot()
Input Parameter        : *hComm         => Communication Handle.
                         bslDownload    => BSL_DOWNLOAD structure.
                          .eraseFlash   => 0: Flash is not erased before
                                             downloading.
                                           1: Flash is erased before
                                            downloading.
                          .eraseOnly    => 0: Continue with Flash download.
                                           1: Do not continue with Flash
                                            download after erase.
Output Parameter       : *uiError       => Error Code
Return Value           : None
---------------------------------------------------------------------*/
```

```
/*-------------------------------------------------------------------
Function Name          : bsl_prepare_verification()
Description            : Loads the verification code to XRAM and
                         executes it.
Function Called        : bsl_uart_header(),bsl_uart_data(),
                         bsl_uart_eot()
Input Parameter        : *hComm         => Communication Handle.
                         autobaudrate   => AUTO_BAUDRATE structure.
Output Parameter       : *uiError       => Error Code
Return Value           : None
-------------------------------------------------------------------*/
```

5 Tips and Recommendations

5.1 Flash Verification

Since it is not possible to use BSL mode to read out the memory content, there is a need to download a verification program into XRAM and run it in order to verify the flash content. For this purpose, the file **verify.a51** (**verify.hex**) containing the verification code, is provided. In case of a LIN device the file **verify_lin.a51** (**verify_lin.hex**) contains the respective verification code.

After downloading the hex file content into XRAM, the code will do the following:

1. Auto baud rate detection.

   HOST has to send bsl_autobaudrate() command.
2. Wait for the bsl_uart_header() from HOST.

   The header block contains the starting address for the verification.

   HOST has to send bsl_uart_header() instruction with MODE=2
3. Wait for the bsl_uart_data() from HOST.

   HOST will send data blocks containing the code that was downloaded to the flash before. The data bytes received from the HOST will be compared with the data bytes stored in the flash.
5. Step 3 will be repeated until all code / data is verified. In case of a mismatch of received and stored data byte the microcontroller will send a verification error response as described below for each byte.
6. HOST has to send the bsl_uart_eot() instruction. **Important: the parameter lastCodeLength has to be 0.**
7. After completing the verification, HOST will reset the device and reinitialize the BSL mode.

This procedure can be performed for both UART and LIN (using Fast LIN BSL) devices.

In case of mismatch of received and stored data byte, the following response will be sent for each error byte:

```
Verification Error Byte    : 0xFC
High Byte Address          : 0xXX
Low Byte Address           : 0xXX
Actual Data Byte           : 0xXX
Expected Data Byte         : 0xXX
```

5.2 Flash / XRAM Memory Mapping

The XC878 family offers flash devices with 64 kbytes or 52 kbytes of embedded flash memory. Each flash device consists of a P-Flash and D-Flash block.

A 64-kbyte flash device consists of 60 kbytes of P-Flash and 4 kbytes of D-Flash; a 52-kbyte device consists of 48 kbytes of P-Flash and 4 kbytes of D-Flash.

The following tables show the address mappings for P-Flash and D-Flash of the two XC878 devices.

For the 64-kbyte flash device, P-Flash has 120 pages, each page has 8 wordlines with 64 bytes per wordline. D-Flash has 64 pages, each page has 2 wordlines with 32 bytes per wordline. Both flash types can be used for code and data storage. The flash can be either erased page by page or mass erased.

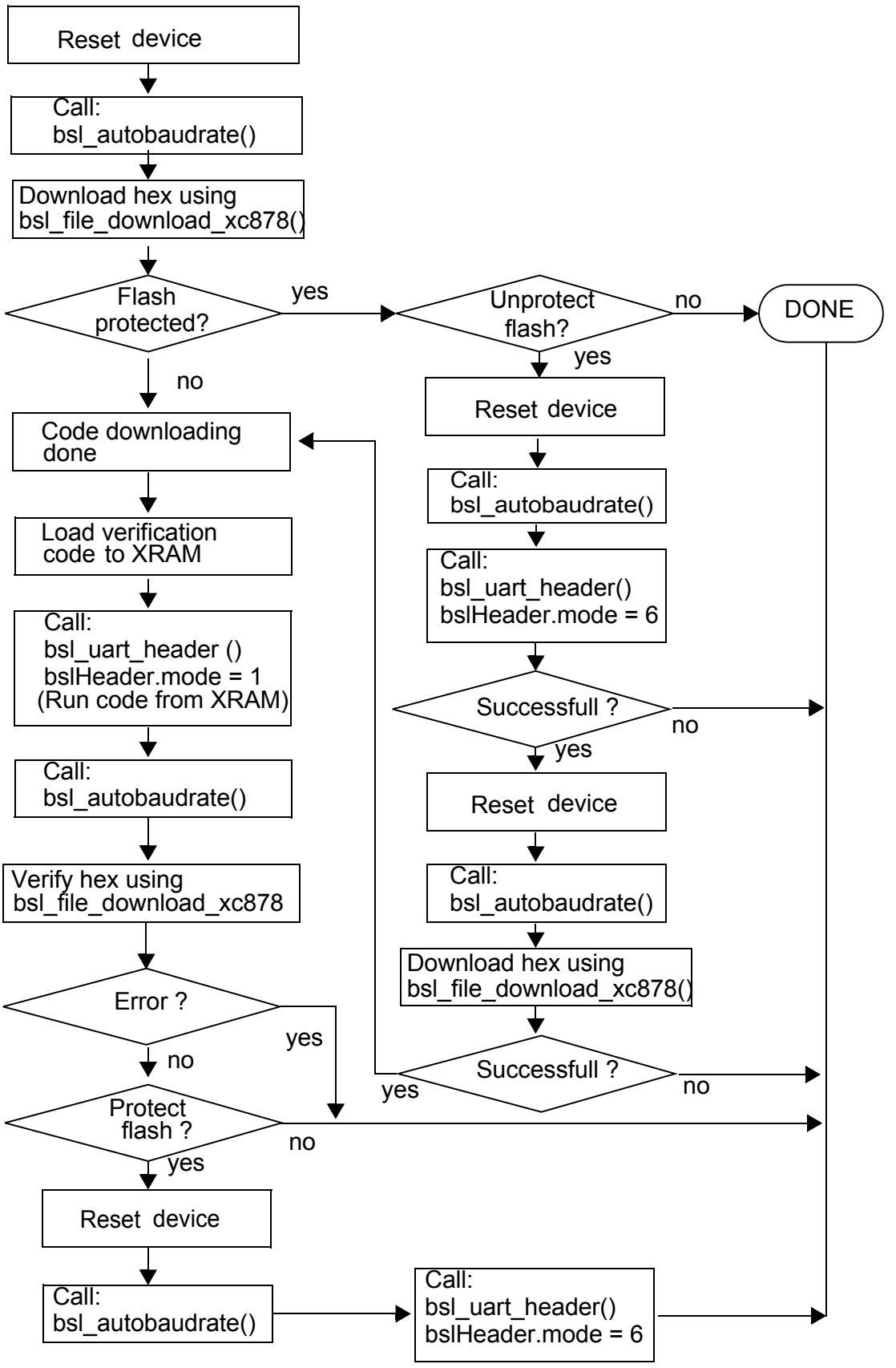
64 kbytes of flash program memory:

Bank	Type	Address Range	Size
0	P-Flash	0x0000 - 0xEFFF	60 kbytes
0	D-Flash	0xF000 - 0xFFFF	4 kbytes
2	XRAM	0xF000 - 0xFBFF	3 kbytes

52 kbytes of flash program memory:

Bank	Type	Address Range	Size
0	P-Flash	0x0000 - 0xBFFF	48 kbyte
0	D-Flash	0xE000 - 0xEFFF	4 kbytes
0	XRAM	0xF000 - 0xFBFF	3 kbytes

5.3 Flow Chart of the Example Code



5.4 BSL Protocol Used

5.4.1 Flash / XRAM Code Download / Verification Using UART

Header Block

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
TYPE (0)	**MODE**	**Start Address High**	**Start Address Low**	**Block Length**	**Not Used**	**Not Used**	**Checksum**

Data Block

Byte 0	Byte 1 - (Block_Length - 2)	Byte (Block_Length - 1)
TYPE (1)	**Program Codes ((Block_Length - 2) byte)**	**Checksum**

EOT Block

Byte 0	Byte 1	Byte 2 - ...	...	...
TYPE (2)	**Last_Codelength**	**Program Codes (Last_Codelength) byte**	**Not Used (Block_Length-3-Last_Codelength) byte**	**Checksum**

5.4.2 Flash Erasing Using UART

Header Block

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
TYPE (0)	**MODE (0x10)**	**Start-Addr High**	**Start-Addr Low**	**Erase/ MassErase**	**Not Used**	**Not Used**	**Checksum**

5.4.3 Flash Protection / Unprotection Using UART

Header Block

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
TYPE (0)	**MODE (0x06)**	**Password High**	**Password Low**	**Not Used**	**Not Used**	**Not Used**	**Checksum**

	Description
TYPE	0 = Header Block 1 = Data Block 2 = EOT Block
MODE	0 = Download to XRAM 1 = Execute Code from XRAM (leave BSL mode) 2 = Download to FLASH 3 = Execute Code from flash (leave BSL mode) 6 = Flash Protection / Unprotection (leave BSL mode) 16=Erase Flash
Start Address High / Low	16 bit starting address for the programming.
Block Length	The number of bytes for the subsequent Data and EOT blocks (including Type and Checksum)
Program Codes	The bytes to be programmed into the memory.
Last_CodeLength	The number of bytes of the next program codes to be programmed into the memory.
Password	The password for the protection / unprotection of flash (2 bytes).
Erase/MassErase	0 = Erase single page 1 = Erase all P-Flash and D-Flash at once
Response (Acknowledge Frame)	0x55 = OK 0xFE = Checksum Error 0xFD = Flash is protected 0xFC = Verification Error 0xFF = Block Error

5.4.4 LIN Auto Baud Rate Detection

Master Request Header + Command

Byte 0
SYN Break: 0x0 (Half baud rate)

==> Wait 700 µs (maximum)

Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7	Byte8	Byte9	Byte10
SYN Char (0x55)	**ID (0x3C)**	**NAD (0x7F)**	**0x00**	**0x00**	**0x00**	**0x00**	**0x00**	**0x00**	**Fast LIN 0x01**	**Check sum (0x80)**

==> Wait 10 ms

Slave Response Header

Byte 0
SYN Break: 0x0 (Half baud rate)

==> Wait 700 µs (maximum)

Byte 0	Byte 1
SYN Char (0x55)	**ID (0x7D)**

==> Wait 40 ms

==> Wait for Acknowledgement Frame (9 Bytes)

An **Acknowledgement Frame** can be always requested sending a **Slave Response Header** to the device.

LIN Acknowledgement Frame:

Byte0	Byte1	Byte2 - Byte7	Byte8
NAD	**Response**	**Not Used (6 bytes)**	**Checksum**

It is recommended to request an acknowledgement after each **Master Request Header** sent.

As described in the following sections, all commands to the device are sent via the **Master Request Header**.

5.4.5 Flash / XRAM Code Download / Verification Using LIN

Header Block

Byte 0

SYN Break: 0x0 (Half baud rate)

==> Wait 700 µs (maximum)

Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7	Byte8	Byte9	Byte10
SYN Char (0x55)	ID (0x3C)	NAD (0x7F)	TYPE (0x00)	MODE	Start Addr High	Start Addr Low	Data Count	Not Used	Fast LIN 0x01	Check sum

Data Block

Byte 0

SYN Break: 0x0 (Half baud rate)

==> Wait 700 µs (maximum)

Byte0	Byte1	Byte2	Byte3	Byte4 - Byte9	Byte10
SYN Char (0x55)	ID (0x3C)	NAD (0x7F)	TYPE (0x01)	Program Code (6 bytes)	Check sum

EOT Block

Byte 0

SYN Break: 0x0 (Half baud rate)

==> Wait 700 µs (maximum)

Byte0	Byte1	Byte2	Byte3	Byte4	Byte5 - Byte8	Byte9	Byte10
SYN Char (0x55)	ID (0x3C)	NAD (0x7F)	TYPE (0x02)	Last Code Length	Program Code	Not Used	Check sum

5.4.6 Flash Erasing Using LIN

Header Block

Byte 0

SYN Break: 0x0 (Half baud rate)

==> Wait 700 µs (maximum)

Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7	Byte8	Byte9	Byte10
SYN Char (0x55)	ID (0x3C)	NAD (0x7F)	TYPE (0x00)	MODE (0x10)	Start Addr High	Start Addr Low	Erase/ Mass Erase	Not Used	Not Used	Check sum

5.4.7 Flash Protection / Unprotection Using LIN

Header Block

Byte 0

SYN Break: 0x0 (Half baud rate)

==> Wait 700 µs (maximum)

Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7	Byte8	Byte9	Byte10
SYN Char (0x55)	ID (0x3C)	NAD (0x7F)	TYPE (0x00)	MODE (0x06)	Pass-word High	Pass-word Low	Not Used	Not Used	Not Used	Check sum

***Note:** A Slave Response Header can be sent after each Master Request Block in order to request an Acknowledgement Frame from the device (see Section 5.4.4).*

***Note:** The only LIN frame used in this example code is the LIN "Auto baud rate Detection" frame (see Section 5.4.4) in order to enter Fast LIN BSL mode. The microcontroller will switch to the BSL UART protocol and the communication structure will be the same as in UART BSL mode.*

	Description
TYPE	0 = Header Block 1 = Data Block 2 = EOT Block
MODE	0 = Download to XRAM 1 = Execute Code from XRAM (leave BSL mode) 2 = Download to FLASH 3 = Execute Code from flash (leave BSL mode) 6 = Flash Protection / Unprotection (leave BSL mode) 16=Erase Flash
Fast LIN	0 = Disable Fast LIN BSL 1 = Enable and enter Fast LIN BSL mode (used in this example code)
Start Address High / Low	16 bit starting address for the programming.
Data Count	The number of data blocks to be subsequently sent during programming[1)].
Program Codes	The bytes to be programmed into the memory.
Last_CodeLength	The number of bytes of the next program codes to be programmed into the memory.
Password	The password for the protection / unprotection of flash (2 bytes).
Erase/MassErase	0 = Erase single page 1 = Erase all P-Flash and D-Flash at once
Response (Acknowledge Frame)	0x55 = OK 0xFE = Checksum Error 0xFD = Flash is protected 0xFC = Verification Error 0xFF = Block Error

1) The length of program code in the data block is fixed to 6 bytes.

w w w . i n f i n e o n . c o m

Published by Infineon Technologies AG