GM Bootloader Updater

Technical Reference

GM Specifics

Version 1.0

Authors	Sebastian Loos
Status	Released

Document Information

History

Author	Date	Version	Remarks
Sebastian Loos	2016-07-27	0.1	Initial Version
Sebastian Loos	2016-12-13	1.0	Release

Reference Documents

No.	Source	Title	Version
[1]	Vector	TechnicalReference_FBL_Updater.pdf	1.00.00
[2]	Vector	TechnicalReference_FBL_<HW>.pdf	

Contents

Illustrations

Tables

1 Introduction

This document covers the GM-specific particularities of the Bootloader Updater. You will find a general description of the Bootloader Updater in [1]. Please check also for a document describing the hardware specifics (if available) [2].

2 Theory of Operation

The Updater is a component that is downloaded to the ECU like a regular application. The FBL does not have any knowledge about the Updater so that from the FBL’s point of view this process is completely transparent. There is no specific handling implemented in the FBL. I.e. the Updater has to pass the usual validation process like for any application download.

For production purpose, the updater requires the typical GM specific container including GM signed header just as any application would. For a detailed description of the update process see [1]. In Figure 2-1 the typical update process of a GM FBL is shown.

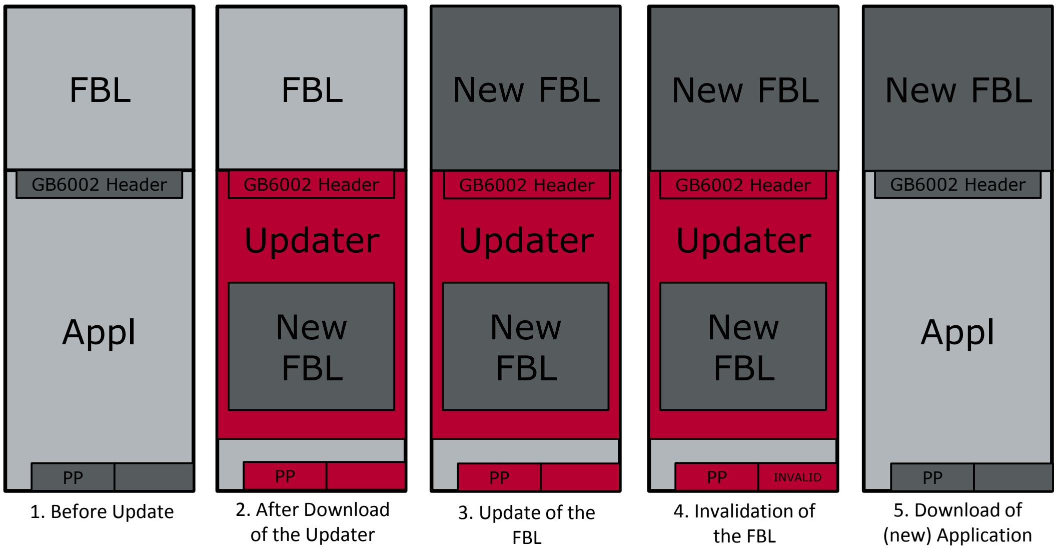


Figure 2-1 Download Process with Invalidation of the Updater

The GM Updater does not transmit or receive messages during the update process.

3 Getting Started

3.1 Integration

The OEM-specific files may have to be customized for your application. The files are found in the FblUpd_Template folder. You should copy these files to your Bootloader Updater project folder, and rename them, removing the leading underscore from the filename.

File	Description
_upd_oem_ap.c	Callbacks for invalidating the Presence Pattern (Programmed State Indicator) after a successful update.
_upd_oem_ap.h	
_upd_oem_cfg.h	Configuration values

Table 3-1 GM-specific files

3.2 Mapping

On some hardware platforms, and if the Updater shall be reset safe, there may be some restrictions to the mapping.

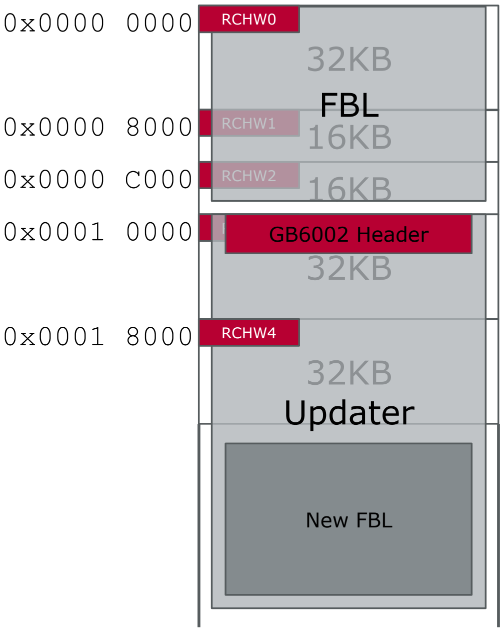


Figure 3-1 Example Mapping for a MPC derivative

It is required by the GB6002 that the Plain Header is located in the beginning of the application. Unfortunately this is often the region where some magic words are expected by the hardware. These magic words are usually located at the start of a flash block and interfere with the location of the Plain Header. Mostly at least one magic word is necessary for a reset safe updater.

Affected platforms are e.g.:

- Tricore Aurix (BMHD)
- MPC (RCHW)
- …

Check [2] for some details.

In Figure 3-1 an example for such restrictions on a MPC platform are shown: Because the FBL usually covers most of the flash area below address 0x10000, the application starts from 0x10000. It is required that the Plain Header is placed at the beginning of the application flash area. This leaves only the RCHW at 0x18000 for use by the updater.

3.3 Configuration

If you use multiple memory drivers, make sure that the right value (the same as is used in the bootloader project) for `FBL_PP_SEGMENT_SIZE` is set.

If you define more than one logical block in the logical block table of the updater, you have to adapt `FBL_UPD_LBT_NR_OF_UPDATER`.

Note
Please check the files for “TODO by customer” tags.

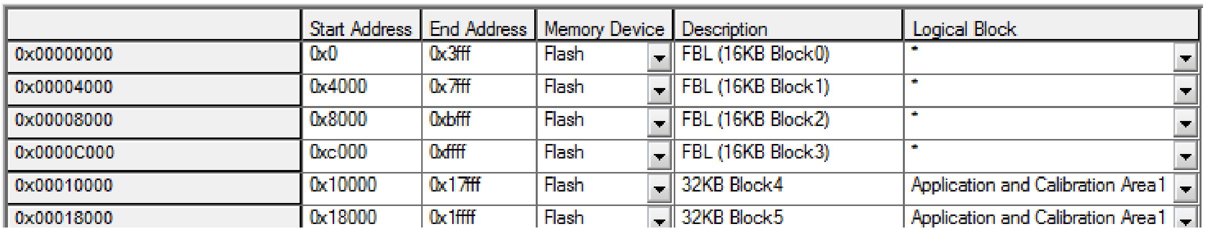
Depending on the used configuration tool, please follow the appropriate sub-paragraph:

3.3.1 GENy as Generator

In the flash block table, (fbl_apfb.c), the flash blocks are specified in which the (new) Bootloader resides. In the logical block table (fbl_mtab.c) the location of the presence pattern of the updater is defined.

Create a GENy Configuration, comparable to the Bootloaders configuration. If you re-use the Bootloaders configuration, you have to make these changes:

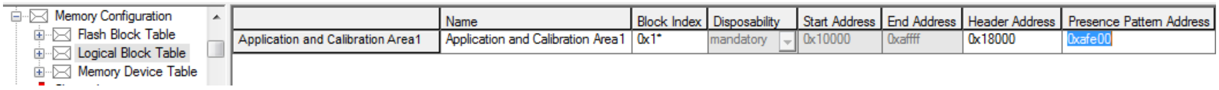
1. Change the settings of the flash blocks in which the Bootloader resides from “Protected” to “Flash” (see Figure 3-2).



	Start Address	End Address	Memory Device	Description	Logical Block
0x00000000	0x0	0x3fff	Flash	FBL (16KB Block0)	*
0x00004000	0x4000	0x7fff	Flash	FBL (16KB Block1)	*
0x00008000	0x8000	0xbfff	Flash	FBL (16KB Block2)	*
0x0000C000	0xc000	0xffff	Flash	FBL (16KB Block3)	*
0x00010000	0x10000	0x17fff	Flash	32KB Block4	Application and Calibration Area1
0x00018000	0x18000	0x1ffff	Flash	32KB Block5	Application and Calibration Area1

Figure 3-2 Flash Block Table adaptions

2. Make sure, the same address for the Presence Pattern as used in the Bootloader Project is set (see Figure 3-3).



	Name	Block Index	Disposability	Start Address	End Address	Header Address	Presence Pattern Address
Application and Calibration Area1	Application and Calibration Area1	0x1*	mandatory	0x10000	0xaffff	0x18000	0xafe00

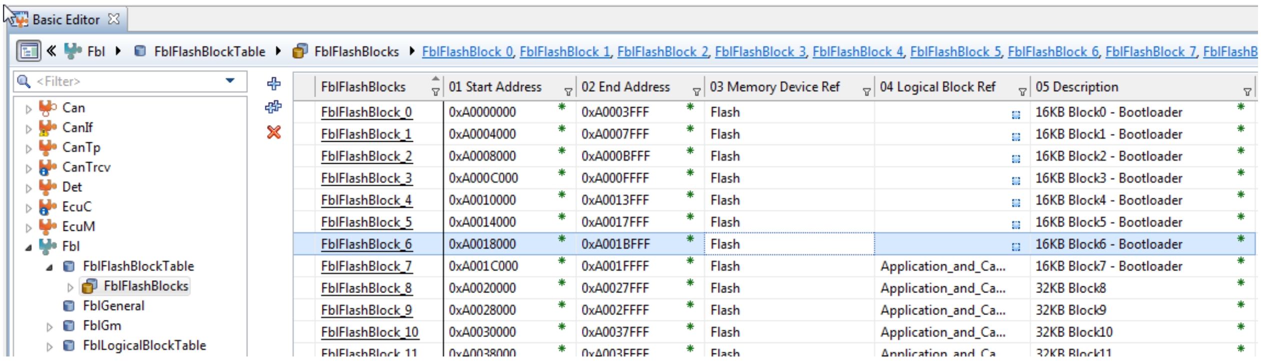
Figure 3-3 Presence Pattern Address

3.3.2 Da Vinci Configurator 5

In the flash block table, (Fbl_Fbt.c), the flash blocks are specified in which the (new) Bootloader resides. In the logical block table (Fbl_Lbt.c) the location of the presence pattern of the updater is defined.

Create a GENy Configuration, comparable to the Bootloaders configuration. If you re-use the Bootloaders configuration, you have to make these changes:

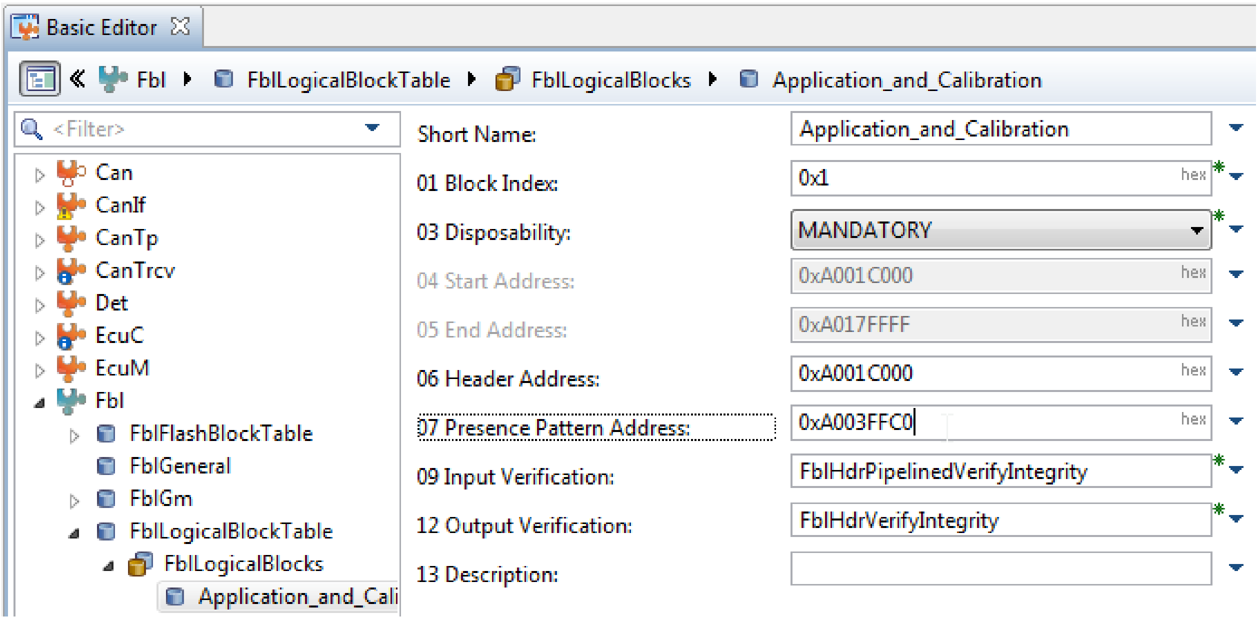
Change the settings of the flash blocks in which the Bootloader resides from “Protected” to “Flash” (see Figure 3-4).



FblFlashBlocks	01 Start Address	02 End Address	03 Memory Device Ref	04 Logical Block Ref	05 Description
FblFlashBlock_0	0xA0000000	0xA0003FFF	Flash		16KB Block0 - Bootloader
FblFlashBlock_1	0xA0004000	0xA0007FFF	Flash		16KB Block1 - Bootloader
FblFlashBlock_2	0xA0008000	0xA000BFFF	Flash		16KB Block2 - Bootloader
FblFlashBlock_3	0xA000C000	0xA000FFFF	Flash		16KB Block3 - Bootloader
FblFlashBlock_4	0xA0010000	0xA0013FFF	Flash		16KB Block4 - Bootloader
FblFlashBlock_5	0xA0014000	0xA0017FFF	Flash		16KB Block5 - Bootloader
FblFlashBlock_6	0xA0018000	0xA001BFFF	Flash		16KB Block6 - Bootloader
FblFlashBlock_7	0xA001C000	0xA001FFFF	Flash	Application_and_Ca...	16KB Block7 - Bootloader
FblFlashBlock_8	0xA0020000	0xA0027FFF	Flash	Application_and_Ca...	32KB Block8
FblFlashBlock_9	0xA0028000	0xA002FFFF	Flash	Application_and_Ca...	32KB Block9
FblFlashBlock_10	0xA0030000	0xA0037FFF	Flash	Application_and_Ca...	32KB Block10
FblFlashBlock_11	0xA0038000	0xA003FFFF	Flash	Application_and_Ca	32KB Block11

Figure 3-4 Flash Block Table adaptions

Make sure, the same address for the Presence Pattern as used in the Bootloader Project is set (see Figure 3-5).



Short Name: Application_and_Calibration

01 Block Index: 0x1 hex

03 Disposability: MANDATORY

04 Start Address: 0xA001C000 hex

05 End Address: 0xA017FFFF hex

06 Header Address: 0xA001C000 hex

07 Presence Pattern Address: 0xA003FFC0 hex

09 Input Verification: FblHdrPipelinedVerifyIntegrity

12 Output Verification: FblHdrVerifyIntegrity

13 Description:

Figure 3-5 Presence Pattern Address

4 Glossary and Abbreviations

4.1 Glossary

Term	Description

4.2 Abbreviations

Abbreviation	Description

5 Contact

Visit our website for more information on

- News
- Products
- Demo software
- Support
- Training data
- Addresses

www.vector.com