Flash Bootloader OEM

Technical Reference

CANfbl GM compression interface

Version 1.0

Authors	Andreas Wenckebach
Status	Released

Document Information

History

Author	Date	Version	Remarks

Table 1-1 History of the Document

Reference Documents

No.	Source	Title	Document No.	Version
[1]	GM	GB6002 Bootloader specification	GB6002	V1.1 (Oct 21 2014)
[2]	GM	Global-A Secure Bootloader Specification		V3.1 (July 29 2013)

Table 1-2 References Documents

Contents

Illustrations

Tables

1 Introduction

In specifications [1] applicable for Global B programs and [2] applicable for Global A programs General Motors (GM) introduces the possibility to use compression for the ECU flash programming process. Both specifications allow for different compression algorithms to be used. The default algorithm GM specifies is the ARLE (Adaptive Run Length Encoding) compression which is variant of RLE (Run Length Encoding).

GM will create the Signed- and Compressed data containers, therefore their tooling need to be able the required compression.

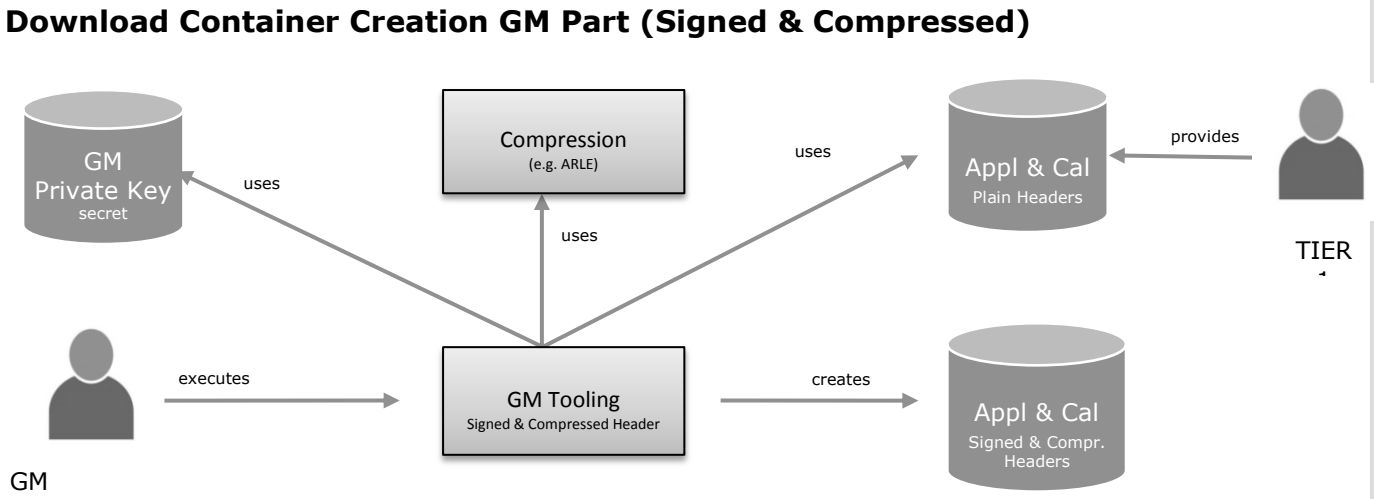


Figure 1-1 Gm creates signed- and compressed container format out of provided plain format.

2 Vector compression related products and services

Within our deliveries since 2015 (one in late 2014) we support compression in our GM SLP5 and SLP6 products as defined in [1] or [2].

As per January 2015 you can choose between these options:

1. Fully integrated support of the GM specified ARLE Algorithm
2. Compression Interface, to be used to implement user specific compression as required by GM

This document shall explain the used API for 1. and provide the required information to implement a user specific compression for 2.

Note
The Vector Standard compression (LZSS based) is currently not available for the GM compression use case. It is currently under examination if GM can use it in their tooling to create containers. Please contact us to get latest information on this.

3 The Vector Data Processing API

Every compression to be implemented will have to be below the Data Processing API found in fbl_ap.c. Each function is called with a pointer to a tProcParam structure with the following members; some are input and some are output, dataLength is both input and output:

Struct tProcParam	
dataBuffer	**Input**: data buffer with to be decompressed byte stream.
dataLength	**Input**: Length of provided input data, **Output**: Length of consumed input data
dataOutBuffer	**Output**: Buffer provided for output data. Only modify contents, but not pointer
dataOutLength	**Output**: Length of produced output data
dataOutMaxLength	**Input**: Maximum length of output data (size of buffer provided for output data)
wdTriggerFct	**Input**: Watchdog trigger function / Polling function; to be called at least every 1ms within decompression.
mode	

Table 3-1 tProcParam members and their function

Prototype	
`tFblResult ApplFblInitDataProcessing( tProcParam * procParam )`	
Parameter	
`tProcParam * procParam`	Check member Table 3-1. Relevant parameter for Init is procParam->mode.
Return code	
`tFblResult`	kFblOk/kFblFailed upon function execution success.
Functional Description	
Call Compression initialization function if applicable (default on any compression: (MODULE_DF_COMPR == (procParam->mode & MODULE_DF_COMPR)).	
Particularities and Limitations	
> Already prepared for Compression use case both for ARLE and user specific compression	
Call context	
> Called from fbl_hdr.c FblHdrCheckEnvelopesExtractSignedHeader during envelope extraction > Called from fbl_mem.c FblMemSegmentStartIndication before programming data Segment	

Table 3-2 Function ApplFblInitDataProcessing

Prototype	
`tFblResult ApplFblDataProcessing( tProcParam * procParam )`	
Parameter	
`tProcParam * procParam`	Check member table Table 3-1.All members are used
Return code	
`tFblResult`	kFblOk/kFblFailed depending on function success.
Functional Description	
Prepares and calls Decompress functionality, if compression is applicable through `procParam->mode` - Redefines procParam->dataOutMaxLength depending on left bytes in segment in case of decompression of programmed data. - Consumes Segment size in case of decompression of programmed data	
Particularities and Limitations	
> Already prepared for Compression use both for ARLE and user specific compression	
Call context	
Called from fbl_hdr.c FblHdrCheckEnvelopesExtractSignedHeader during envelope extraction Called from fbl_mem.c FblMemProcessJob during programming.	

Table 3-3 Function ApplFblDataProcessing

Prototype	
`tFblResult ApplFblDeinitDataProcessing( tProcParam * procParam )`	
Parameter	
`tProcParam * procParam`	Check member table in Table 3-1. All members are used
Return code	
`tFblResult`	kFblOk/kFblFailed depending on function success.
Functional Description	
Particularities and Limitations	
> May call Decompression Deinit if applicable (not required in case of ARLE, might be helpful in case of user decompresssion)	
Call context	
> Blind text, blind text	

Table 3-4 Function ApplFblDeinitDataProcessing

4 Particularities on the GM use case

4.1 Compression indicator

GM does not use the UDS commonly used DFI of service $34 to determine if decompression is applicable.

Instead, on the first $36 received for a given module, the module's envelope is analyzed to determine the availability of a compression envelope. If such an envelope is found the decompression is applicable for this module (compare [1] or [2], chapter "Assigned Data Types" and "Signed and Compessed Application / Calibration File").

If decompression is applicable, decompression need to be performed two times:

> First decompressing the signed header to FblRamHeader. This allows for analyzing the signed header envelope in RAM

> Decompressing the to be programmed content behind the signed header, starting with the plain header envelope. To do this the decompression need to start again from the first compressed byte, but shall produce output bytes only when the first to be programmed byte is reached.

4.2 The mode flag

The procParam->mode flag can be used to querry the context of the current decompression action mentioned above:

condition	Decompression context
procParam->mode == MODULE_DF_COMPR_HDR	Decompression of the signed header to FblRamHeader
procParam->mode == MODULE_DF_COMPR	Decompression the to be programmed content
((procParam->mode & MODULE_DF_COMPR) == MODULE_DF_COMPR)	Any Decompression (Signed header or programmed content)

Table 4-1 Two different Decompression states: `procParam->mode`

4.3 Decompress Signed header to Ram

In fbl_hdr.c the function FblHdrCheckEnvelopesExtractSignedHeader will call the whole Data Processing Api once (ApplFblInitDataProcessing/ ApplFblInitDataProcessing / ApplFblDeinitDataProcessing) in order to decompress enough bytes to be able to extract the Signed Header envelope completely.

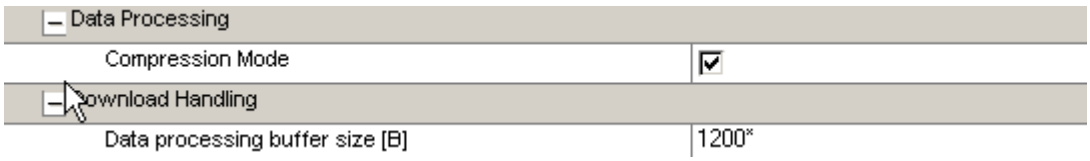
4.4 Decompression of Programmed Data

Initiated from fbl_hdr.c FblHdrTransferDataProcess->FblMemDataIndication() (fbl_mem.c) the programmed data is decompressed. The decompression once again start from the very first compressed byte. However, the decompression will receive a **threshold** value, in order to decide which are the first bytes to produce as output (the first to be programmed bytes after the signed header envelope and the plain header data type bytes).

4.5 GENy configuration

Be sure enable “Compression Mode” in GENy FblDrCan_XX module. Data processing size need to be configured to be large enough to hold the complete header (fbl_hdr.h HDR_MODULE_MAX_RAW_LEN) plus the largest possible produced size when requesting decompression of a single byte (FBL_MAX_COMPR_PRODUCED_CHUNK, detailed in 5.3). For ARLE this is 1188 bytes when allowing a maximum number of data regions (Maximum Number of Segments, GENy configured) of 10.

The Software will check the configured number of bytes to be large enough.



5 The compression module (user compression: to be created)

In the case you receive a fully integrated compression this is already provided. In case you want to implement a user specific compression you have to create this module.

5.1 Files

File Name	Description
cmpr.c	Decompression source code
cmpr.h	Decompression header file

Table 5-1 Files (Provided with ARLE, to be created for user specific compression)

5.2 Configuration

No configuration is required in case of the provided ARLE modules. There are no special requirements for your user specific compression configurations.

5.3 The API

Find below the functions and Global required fort he compression interface. The interface is already implemented if you ordered a fully integration compression. If you want to implement a user specific compression, you need to implement the functions for your compression algorithm.

In fbl:ap.c the required API is mapped to the used naming scheme, adapt the names to your required scheme (CmprXXDecompress, CmprXXInit, CmprXXReadCmprHeader):

```
#if defined( FBL_ENABLE_COMPRESSION_MODE )
/* The below functions are defined if you ordered the Vector Compression interface,
 * the interface has to be implemented by you else.
 */
# include "cmpr.h"
# define ApplFblDecompress CmprArleDecompress
# define ApplFblCmprInit CmprArleInit
# define ApplFblCmprReadHeader CmprArleReadCmprHeader
#endif
```

Globals (cmpr.h)	Function
FBL_MAX_COMPR_PRODUCED_CHUNK	Macro to return the maximum number of bytes that may be produced when just a single output byte is requested to be produced. This can be 1 for an “intelligent” implementation that ignores/buffers further output bytes itself. Example: With ARLE the largest amount of bytes that can be produced when decompression 1byte is (4*63), which would be a 4 byte repeating pattern that can repeat 63 times at maximum (compare [1] or [2]).

Table 5-2 Required Globals

Prototype	
`tFblResult CmprXXInit(void)`	
Parameter	
–	-
Return code	
`tFblResult`	kFblOk/kFblFailed depending on function success.
Functional Description	
Initialize compression module.	
Particularities and Limitations	
> None	
Call context	
Called from fbl_ap.c data processing interface, `ApplFblInitDataProcessing()`	

Table 5-3 Function CmprXXInit

Prototype	
`tFblResult CmprXXDecompress( tProcParam* procParam, vuint16 outThreshold);`	
Parameter	
`tProcParam* procParam`	Check member table Table 3-1.All members are used (compare Particularities and Limititations).
`outThreshold`	Put data to procParam->dataOutBuffer only after a certain amount of decompresed bytes. A value of "0" means no threshold is used.
Return code	
–	-

Functional Description
Decompress data stream according to either Gm defined ARLE or user specific compression (in accordance with GM specification).
Particularities and Limitations
> May be called with less input bytes than required. May need to backup partly decompressed bytes from current decompression action waiting for required compressed input bytes. > The Compression module needs to call `procParam ->wdTriggerFct` at least every 1ms (Recommendation in all loops) to guarantee correct handling of timer related functionality in the Fbl (despite the member name it is more the only watchdog)
Caution In earlier version (2014 releases) a 3rd parameter “vuint16 maxOutLen” was available. If this is available it is to be used instead of the “`procParam`->dataOutMaxLength” param
Call context
Called from fbl_ap.c data processing interface, `ApplFblDataProcessing()`

Table 5-4 Function CmprXXDecompress

Prototype	
`tFblResult CmprXXReadCmprHeader( tFblLength* comprLength, const vuint8 * cmprBuffer, vuint16 * cmprDataOffset)`	
Parameter	
`tFblLength* comprLength`	Output: the compressed data size (return 0 if not applicable for user specific DataInfo)
`const vuint8 * cmprBuffer`	Input: Pointer to compressed data
`vuint16 * cmprDataOffset`	Output: Index to start of compressed data stream (0 if no DataInfo field used)
Return code	
–	-
Functional Description	
Read Compression header DataInfo field/format (compare [1] or [2] “Details of a Compressed and Signed file”) . In case of ARLE this header includes information of the compressed data length.	
Particularities and Limitations	
> Check [1] or [2] on Compression header requirements.	

Call context
> Called from FblHdrCheckEnvelopesExtractSignedHeader when compressed envelope has been identified.

Table 5-5 Function CmprXXReadCmprHeader

6 Glossary and Abbreviations

6.1 Glossary

Term	Description
GM	General Motors Company
ECU	Electronic Control Unit
FBL	Flash Boot Loader
LZSS Algorithm	Lempel-Ziv-Storer-Szymanski-Algorithm
UDS	Unified Diagnostic Services
GMLAN	General Motor Local Area Network
DFI	Data Format Identifier
$34	UDS/GMLAN request download service
$36	UDS/GMLAN Transfer Data service

6.2 Abbreviations

Abbreviation	Description
ARLE	Adaptive Run Length Encoding
RLE	Run Length Encoding

7 Contact

Visit our website for more information on

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

www.vector.com