



BOSCH  CC/ESM2	Hex File Handling in Gen9 projects	Edition 1.16.2	Page 1/32	Date 14.07.2016
	Documentation	Author Heiko Eckert		Phone 07062/911-4332

Documentation

HexFile Handling in Gen9 projects

BOSCH  CC/ESM2	Hex File Handling in Gen9 projects	Edition 1.16.2	Page 2/32	Date 14.07.2016
	Documentation	Author Heiko Eckert		Phone 07062/911-4332

Project name: HexFile Handling in Gen9 projects

Author: CC/EVD8 Heiko Eckert

Relevant for: Make Tool Chain
Subordinate tools (HawCC, XFlash, PMSD, DLM-Composer)
HSW component of ECU projects




BOSCH  CC/ESM2	Hex File Handling in Gen9 projects	Edition 1.16.2	Page 3/32	Date 14.07.2016
	Documentation	Author Heiko Eckert		Phone 07062/911-4332

Table of Contents:

1. REVISION HISTORY.....	5
2. INTRODUCTION.....	7
2.1. SCOPE / PURPOSE.....	7
2.2. TERMS / ABBREVIATIONS	7
3. CONCEPTS.....	8
3.1. BASE CONCEPT	8
3.2. MEMORY LAYOUT	9
3.3. HEXFILE IDENTIFICATION CODE (HexID).....	9
3.3.1. VALID HEXFILE IDENTIFICATION CODES	9
3.3.2. MARKING HEXFILES AS INVALID FOR THE PLANT	9
3.3.3. SPECIFIC (INVALID) HEXFILE IDENTIFICATION CODES:	10
3.4. FILLING EMPTY HEXAREAS WITH PSEUDO RANDOM FILL PATTERN	10
3.5. MULTI-BLOCK HEXFILE HANDLING.....	10
3.6. HANDLING OF DOWNLOAD MODULES	10
4. SUPPORTED DEVICES.....	11
4.1. SUPPORTED DEVICES F035	11
4.2. SUPPORTED DEVICES F021	12
4.3. DETAILED INFORMATION ON HEXFILE BLOCKS	13
4.3.1. GENERAL.....	13
4.3.2. REFERENCE POINTERS.....	14
4.3.3. HEXFILE HEADER.....	15
5. SW HEXINFO BLOCK DEFINITIONS	16
5.1. VALID HexINFO STRUCTURE IDS	17
5.1.1. STRUCTURE IDS 0x09/0x89.....	17
5.2. CODING OF HEXINFO ENTRIES.....	18
5.2.1. MTC VERSION.....	18
5.2.2. BASE VERSION	18
5.2.3. BUILD NO.....	18
5.2.4. BUILD SAMPLE	19
5.2.5. DATE & TIME	19
5.2.6. CODING OF APPLICATION INFORMATION.....	19
5.2.7. CONSISTENCY CHECKSUM	20
5.2.8. LOGISTIC DATA	20
5.2.9. MTC CONFIGURATION	21
5.2.10. APPLICATION DATA	21
6. CHECK BLOCK CODING	22
6.1.1. CHECK BLOCK LOCATION	22
6.2. CALCULATION OF CHECK WORDS	22
6.3. VALID CHECKSUM ALGORITHM IDS	23
6.3.1. CHECKSUM ALGORITHM 0x83.....	23
6.3.2. CHECKSUM ALGORITHM 0x84.....	23
6.4. CONSISTENCY CHECKS FOR HEXFILE BLOCKS	24


BOSCH  CC/ESM2	Hex File Handling in Gen9 projects	Edition 1.16.2	Page 4/32	Date 14.07.2016
	Documentation	Author Heiko Eckert		Phone 07062/911-4332

7.	DEVICE DESCRIPTION FILE TI_GALAXY.DDF	26
7.1.	DEFINITION	26
7.2.	EXAMPLE	27
8.	EXTERNAL FLASH SECURITY TOOLS	28
8.1.	CONFIGURATION / ARCHITECTURE	28
8.2.	FLASH PARITY	29
8.3.	ECC	29
9.	HSW	30
9.1.	LCF FILE HANDLING	30
9.1.1.	DEVICEID	30
9.1.2.	MEMORY REGIONS	30
9.1.3.	IMPORTANT LINKS	30
9.2.	LCF09_COMMON	31
9.2.1.	HEXFILE FLAGS	31
9.2.2.	BLOCK DESIGN	32


BOSCH  CC/ESM2	Hex File Handling in Gen9 projects	Edition 1.16.2	Page 5/32	Date 14.07.2016
	Documentation	Author Heiko Eckert		Phone 07062/911-4332

1. Revision History

Document version	Date	Name	Comment
V0.1	16.02.07	CC/ESM2-Eckert	First draft version.
V0.2	19.02.07	CC/ESM2-Eckert	Converted to Gen9 requirements CHECK location always directly coded in hexfile now DevID shifted to 0x0000002C
V0.3	28.02.07	CC/ESM2-Eckert	Removed CHECK from device graphics Using universe device for DDF example now
V0.4	20.03.07	CC/ESM2-Eckert	Corrections
V1.0	04.04.07	CC/ESM2-Eckert	First version
V1.1	19.04.07	CC/ESM2-Eckert	Added parity chapter 8
V1.2	13.07.07	CC/ESM2-Eckert	Inserted hexfile header description
V1.3	06.09.07	CC/ESM2-Eckert	Initial gen9 hexinfo structures: 0x89/0x09
V1.3.1	17.09.07	CC/ESM2-Eckert	Minor changes in gen9 hexinfo description
V1.4	23.10.07	CC/ESM2-Eckert	Repaired links to external documents Added pointer to DLM in graphics Updated consistency structure
V1.4.1	04.01.08	CC/ESM2-Eckert	some minor changes / corrections
V1.4.2	14.01.08	CC/ESM2-Eckert	corrected findings from HIT-excelsheet
V1.5	18.02.08	CC/ESM2-Eckert	reworked chapters 8/9 (added ECC tool, added Mira+ device)
V1.6	10.03.08	CC/ESM2-Eckert	reworked chapter 9, added WegaPlusRevA device
V1.7	09.06.08	CC/ESM2-Eckert	Added device Sirius_F035 Added description for new BaseVersion coding.
V1.8	09.02.09	CC/EMT5-Eckert	Added device Capella_F035 Added MTC9 configuration concept

BOSCH  CC/ESM2	Hex File Handling in Gen9 projects		Edition 1.16.2	Page 6/32	Date 14.07.2016
	Documentation		Author Heiko Eckert		Phone 07062/911-4332

V1.8.1	17.02.09	CC/EMT5-Eckert	corrected findings from HIT-excelsheet
V1.9	23.02.09	CC/EMT5-Eckert	Random fill pattern for ePSW
V1.10	09.04.09	CC/EMT5-Eckert	Fixed address offsets in chapter 5.1.1 Added ProcyonRevC_F035 Added device WegaPlusRevC_F035
V1.10.1	15.06.09	CC/EMT5-Eckert	Updated MTC configuration concept
V1.11	10.07.09	CC/EMT5-Eckert	Added MiraPlusRevC, CapellaRevA, SiriusRevA, updated description of consistency check concept.
V1.11.1	08.10.09	CC/EMT5-Eckert	Updated picture in chapter 3.5.4
V1.12	11.03.10	CC/EVD8-Eckert	Added BlockInterfaceRef, corrected reserved bytes at <blockstart>+0x20
V1.13	07.10.10	CC/EVD8-Eckert	Added chapter 5.2.10, modified chapters 3.3.2 and 3.5.2.2
V1.14	04.05.11	CC/EVD8-Eckert	Adapted DLM handling documentation Chapter 4.2: Added F021 devices Random fillpattern
V1.14.1	06.07.11	CC/EVD8-Eckert	Added F021 devices to table in chapter 8
V1.14.2	21.10.11	CC/EVD8-Eckert	Added F021 devices: CarinaRevB, GladiatorRevB, CoronaRev0
V1.15	28.03.12	CC/ESM2-Eckert	Moved content of chapter 3.5/3.6 to other documents Added RigelRev0 Added new blocktype: partial CAL
V1.15.1	25.04.12	CC/ESM2-Eckert	Corrected chapter 4.3.3
V1.15.2	10.05.12	CC/ESM2-Eckert	Corrected chapter 3.3.2 and 3.3.3
V1.16	18.09.12	CC/ESM2-Eckert	Added F021 devices: CapellaRevA, CapellaRevB
V1.16.1	12.12.12	CC/ESM2-Eckert	Added F021 devices: GladiatorRevC, minor bugfixes
V1.16.2	14.07.16	CC/ESM2-Eckert	Corrected configuration numbering in hexinfo

BOSCH  CC/ESM2	Hex File Handling in Gen9 projects	Edition 1.16.2	Page 7/32	Date 14.07.2016
	Documentation	Author Heiko Eckert		Phone 07062/911-4332


2. Introduction

2.1. Scope / Purpose

The purpose of this specification is the definition of the handling of hexfiles during the make process for Gen9 SW.

2.2. Terms / Abbreviations

Term / Abbreviation	Explanation
MTC	Make Tool Chain
TI	Texas Instruments
TMS470	ECU microcontroller device generation
HEX	Here: Intel Hexfile Format
Hexinfo	Structure containing project specific data like BBNr, BuildNr., BuildTime,...
Hexfile block	Defined part of a hexfile A hexfile block contains 0-n complete memory segments of current device
HawCC	Hexfile Alteration Wizard for CC. MTC tool to fill in checksum and patch hexfile information
DLM-Composer	EMT2-Tool to patch download modules to MTC generated hexfile.
Multi-block-concept	Handling of more than one hexfile blocks within current hexfile
PMSD	Parameter modifying system (application tool)
DDF	Device description file. Actually the file ti_galaxy.ddf, which can be found in C:\TSDST\CHK
EOL	end-of the line (reprogramming feature)
DLM	Download modules
DLMT	Download module table
CALPART	A calblock that needs not to start and end on HW section border, but is not flashable

BOSCH  CC/ESM2	Hex File Handling in Gen9 projects	Edition 1.16.2	Page 8/32	Date 14.07.2016
	Documentation	Author Heiko Eckert		Phone 07062/911-4332

3. Concepts

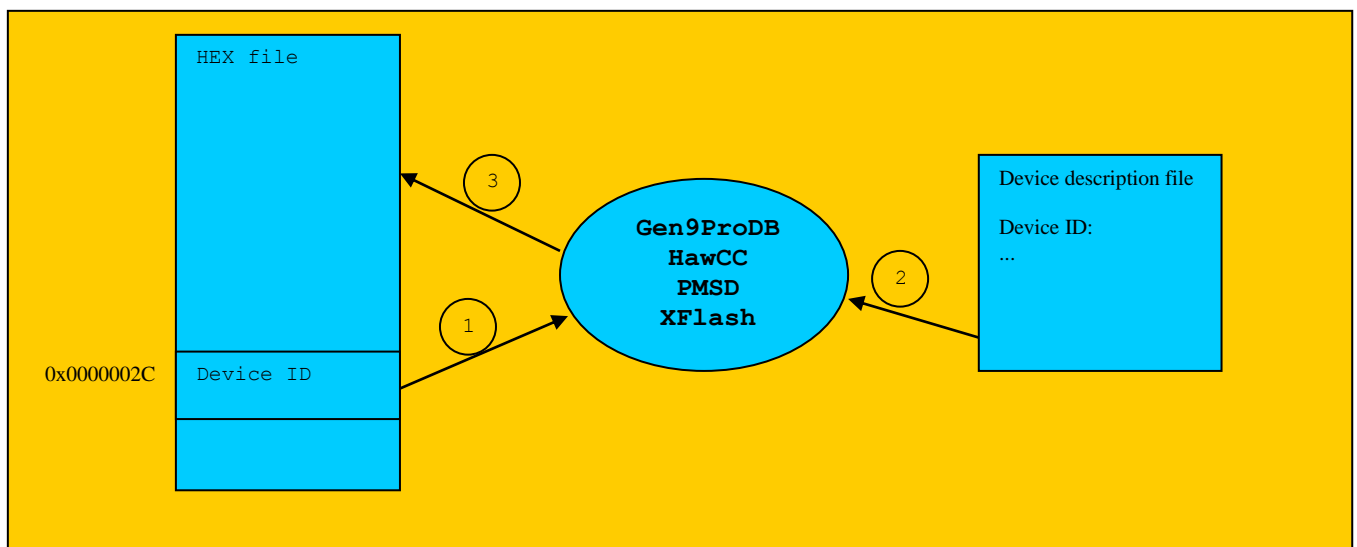
3.1. Base concept

Actual MTC versions support the handling of one RAW hexfile (generated by linker). This hexfile can be split to several block hexfiles by Gen9ProDB tool within MTC.

The members of the TI Gen9 microcontroller family have different memory layouts. To handle these different layouts when manipulating or interpreting hexfiles it is necessary to have a device description for each member. The complete device description is too big to program it directly into the hexfile thus it is just the device identifier which is programmed into the hexfile.

As the program, which interprets the hexfile does not know in advance which device the hexfile is created for, it is necessary to write this device ID at a fix address. The device ID address is determined to 0x0000002C.


Any tool is now able to read the device identifier from that fix address and refer to a device description file, which provides all necessary information about the device.



Step1: The tool reads the device identifier from the fix address 0x0000002C.

Step2: The device description file provides the device specific data for the tool.

Step3: The tool is now able to manipulate/interpret the given hexfile.

BOSCH  CC/ESM2	Hex File Handling in Gen9 projects	Edition 1.16.2	Page 9/32	Date 14.07.2016
	Documentation	Author Heiko Eckert		Phone 07062/911-4332

3.2. Memory layout

To enable flashing of split code and data sections these sections must be allocated at flash section boundaries. It is the task of the ECU SW project to set the correct definitions inside the linker command file (LCF). A side effect of this is that for each hexfile block up to one flash sector may be left unused.

3.3. Hexfile identification code (HexID)

The hexfile identification code ensures backwards compatibility for MTC-external tools like HawCC, XFlash and PMSD.

All Gen9 hexfiles must have valid identification code at address 0x00000030.

3.3.1. Valid hexfile identification codes

Valid identification codes are all values within the following range:
0x00000003-0x000000FF

The Hexfile identification code for Gen9 hexfiles starts at 0x00000003 and will be increased in HSW every time a major change in global mainpath hexfile structure is done.


Gen9 MTC will throw a fatal error if a RawHexfile containing an invalid identification value at 0x00000030 is detected.

3.3.2. Marking hexfiles as invalid for the plant

External tools like PMSD and HawCC must set the hexfile identification code to a specific numeric value after a hexfile has been manipulated.

MTC (Gen9ProDB) will set this identifier as well if one of the following issues occur:

- Not a version in TCM (BuildType in EPK must be "V" for valid hexfiles)
- RELEASE_CONFIGURATION flag in MTC9 is not set
- Build has been started via SharCC (for plant deliveries a CLI build is required)
- MTC version in mk9.cmd differs from the currently used MTC version

BOSCH  CC/ESM2	Hex File Handling in Gen9 projects	Edition 1.16.2	Page 10/32	Date 14.07.2016
	Documentation	Author Heiko Eckert		Phone 07062/911-4332

3.3.3. Specific (invalid) Hexfile identification codes:

MTC9:	0xEEEEEEEEEC
HawCC:	0xEEEEEEEEEF
PMS/D:	0xEEEEEEEEED
CANAPE:	0xEEEEEEEEEA (set via RomDrv.dll)
INCA:	0xEEEEEEEEEB (set via RomDrv.dll)

The RomTrans tool must decline all hexfiles marked with one of these specific (invalid) HexIDs for production.

3.4. Filling empty hexareas with Pseudo random fill pattern

All gaps in RawHex must be filled by MTC. MTC generated hexfiles must not have any gaps.

The standard fill pattern will be patched by Gen9ProDB on every run of the MTC. For gen9 devices we will use a so called "pseudo random pattern".

To make hexfiles from different builds comparable we will create the fill pattern for a given address out of the address value itself. This means for a given address in two hexfiles, the fill pattern always stays the same.

We will use a CRC32 algorithm (polynomial 0x04C11DB7) to calculate the fill pattern for a given address:

FillPattern(current hexaddress) = Σ (valid 32-bit words of address range) Address range: [(current file base address) ... (current address)]


For all MTC versions < 9.7 the fix fill pattern 0xB6 is used instead of the random fill pattern.

3.5. Multi-block hexfile handling

See document: [Multi block hexfile handling.pdf](#)

3.6. Handling of download modules


See document: [Handling download modules.pdf](#)

BOSCH  CC/ESM2	Hex File Handling in Gen9 projects	Edition 1.16.2	Page 11/32	Date 14.07.2016
	Documentation	Author Heiko Eckert		Phone 07062/911-4332

4. Supported devices


4.1. Supported devices F035

Device/ Address	Wega F035 WegaPlusRev0 F035 WegaPlusRevA F035 WegaPlusRevC F035	Mira F035 MiraPlusRev0 F035 MiraPlusRevB F035 MiraPlusRevC F035	SiriusRev0 F035 SiriusRevA F035	CapellaRev0 F035 CapellaRevA F035	ProcyonRevC F035
0x001FFFF0		Fdoc			
0x0013FFF0				Fdoc	
0x00100000					
0x000BFFF0	Fdoc				
0x0005FFF0					Fdoc
0x0003FFF0			Fdoc		
0x00010000					
0x00000038	PubKeyRef	PubKeyRef	PubKeyRef	PubKeyRef	PubKeyRef
0x00000034	CheckRef	CheckRef	CheckRef	CheckRef	CheckRef
0x00000030	Hexfile ID	Hexfile ID	Hexfile ID	Hexfile ID	Hexfile ID
0x0000002C	Device ID	Device ID	Device ID	Device ID	Device ID
0x0000002A	Block Size	Block Size	Block Size	Block Size	Block Size
0x00000028	Hexfile Flags	Hexfile Flags	Hexfile Flags	Hexfile Flags	Hexfile Flags
0x00000024	DLMTRef	DLMTRef	DLMTRef	DLMTRef	DLMTRef
0x00000000					
supported bootblock size (bootsize)	32k (0x08000) 64k (0x10000)	32k (0x08000) 64k (0x10000)	16k (0x04000), 32k (0x08000), 64k (0x10000)	32k (0x08000) 64k (0x10000)	32k (0x08000) 64k (0x10000)

BOSCH  CC/ESM2	Hex File Handling in Gen9 projects	Edition 1.16.2	Page 12/32	Date 14.07.2016
	Documentation	Author Heiko Eckert		Phone 07062/911-4332

4.2. Supported devices F021

Device/ Address	CapellaRev0 F021 CapellaRevA F021 CapellaRevB F021	CarinaRevA F021 CarinaRevB F021	GladiatorRevA F021 GladiatorRevB F021 GladiatorRevC F021	CoronaRev0 F021	RigelRev0 F021
0x003FFFF0					Fdoc
0x002FFFF0			Fdoc		
0x0013FFF0	Fdoc	Fdoc			
0x00100000					
0x000BFFF0					
0x0005FFF0				Fdoc	
0x0003FFF0					
0x00010000					
0x00000038	PubKeyRef	PubKeyRef	PubKeyRef	PubKeyRef	PubKeyRef
0x00000034	CheckRef	CheckRef	CheckRef	CheckRef	CheckRef
0x00000030	Hexfile ID	Hexfile ID	Hexfile ID	Hexfile ID	Hexfile ID
0x0000002C	Device ID	Device ID	Device ID	Device ID	Device ID
0x0000002A	Block Size	Block Size	Block Size	Block Size	Block Size
0x00000028	Hexfile Flags	Hexfile Flags	Hexfile Flags	Hexfile Flags	Hexfile Flags
0x00000024	DLMTRef	DLMTRef	DLMTRef	DLMTRef	DLMTRef
0x00000000					
supported bootblock size (bootsize)	16k (0x04000) 32k (0x08000) 64k (0x10000)	32k (0x08000) 64k (0x10000)	32k (0x08000) 64k (0x10000)	32k (0x08000) 64k (0x10000)	32k (0x08000) 64k (0x10000)

BOSCH  CC/ESM2	Hex File Handling in Gen9 projects		Edition 1.16.2	Page 13/32	Date 14.07.2016
	Documentation		Author Heiko Eckert		Phone 07062/911-4332


4.3. Detailed information on Hexfile blocks

Generation 9 devices do not contain any FPROT/FPROTEMUL areas.

Address of CHECK area will be coded directly in hexfile for all blocks.


4.3.1. General

		rel. addr.	content		
Fdoc	Flash documentation data	0x00	Fdoc word 0		
overall length	0x10	0x04	Fdoc word 1		
start address	defined in DDF	0x08	Fdoc word 2		
	once per file	0x0c	Fdoc word 3		
				initialized by ECU SW	
CHECK	Checksum and CRC	0x00	HexInfo Start	&c_SwHexInfo_ST	32 bit pointer to SW hex info block
overall length	0x10	0x04	Check algorithm ID	0xFF	algorithm required by ECU SW
start address	defined in CheckRref	0x05	reserved	0x00	must be zero
	once per block	0x06	reserved	0x00	must be zero
		0x07	HexInfo Length	0x54	used for EOL consistency checksum calculation
		0x08	64-bit checksum	0xFFFFFFFFFFFFFFFF	initialized by ECU SW, patched by GenProDB
Device ID	Device Identifier	Must be at the fix address 0x2C for all devices !!!			
overall length	0x04	alias name	product name	Device ID	
start address	0x0000002C	WegaPlusRev0_F035	TMS470PSF761A	0x001E6B05	
	once per block	WegaPlusRevA_F035	TMS470PSF761A	0x001E6B0D	
		WegaPlusRevC_F035	TMS470PSF761A	0x001E6B1D	
		Mira_F035	TMS570PSFC61	0x80126B05	
		MiraPlusRev0_F035	TMS570PSFC61A	0x801C6D05	
		MiraPlusRevB_F035	TMS570PSFC61A	0x801C6D15	
		MiraPlusRevC_F035	TMS570PSFC61A	0x801C6D1D	
		SiriusRev0_F035	TMS470PSF261	0x00106B05	
		SiriusRevA_F035	TMS470PSF261	0x00106B0D	
		CapellaRev0_F035	TMS570PSFB61	0x802A6D05	
		CapellaRevA_F035	TMS570PSFB61	0x802A6D0D	
		ProcyonRevC_F035	TMS470PSF361A	0x00266B05	virtual device ID (modified Capella device)
		GladiatorRevA_F021	TMS570LS30336	0x802AAD05	
		GladiatorRevB_F021	TMS570LS30336	0x802AAD15	
		GladiatorRevC_F021	TMS570LS30336	0x802AAD1D	
		CarinaRevA_F021	TMS570LS12004	0x804AAD05	virtual device ID (modified Gladiator device)
		CarinaRevB_F021	TMS570LS12004	0x804AAD15	virtual device ID (modified Gladiator device)
		CapellaRev0_F021	TMS570LS12005	0x804CAD05	virtual device ID (modified Gladiator device)
		CapellaRevA_F021	TMS570LS12005	0x804CAD0D	virtual device ID (modified Gladiator device)
		CapellaRevB_F021	TMS570LS12005	0x804CAD15	virtual device ID (modified Gladiator device)
		CoronaRev0_F021	TMS570LS04005	0x8048AD05	
		RigelRev0_F021	TMS570LC4047	0x8044AD05	

BOSCH  CC/ESM2	Hex File Handling in Gen9 projects		Edition 1.16.2	Page 14/32	Date 14.07.2016
	Documentation		Author Heiko Eckert		Phone 07062/911-4332

4.3.2. Reference Pointers

CheckRef	reference to CHECK block	0x00	Check Block Ptr	32 bit pointer to CHECK block	
overall length	0x04				
start address	0x00000034				
SignatureRef	reference to Signature Area	0x00	Signature Ptr	32 bit pointer to memory location where signatures are located	
overall length	0x04				
start address	0x00000038				
PubKeyRef	reference to Public Key Area	0x00	Public Key Ptr	32 bit pointer to memory location where public keys are located	
overall length	0x04				
start address	0x00000038				
ConsistencyRef	reference to Consistency structure	0x00	Consistency Ptr	32 bit pointer to memory location where consistency structure is located	
overall length	0x04				
start address	0x0000003C				
BlockInterfaceRef	reference to block interface	0x00	BlockInt Ptr	32 bit pointer to memory location where the block interface structure is located	
overall length	0x04				
start address	0x00000040				
DLMTRef	reference to DLM table	0x00	DLMT Ptr	32 bit pointer to memory location where Download module table (DLMT) is located	
overall length	0x04			0x00000000	no space for DMT found
start address	0x00000024			0x00000001-0x00000029	valid DMT ID
				0x00000030-0xffffffff	valid DMT pointer


BOSCH  CC/ESM2	Hex File Handling in Gen9 projects	Edition 1.16.2	Page 15/32	Date 14.07.2016
	Documentation	Author Heiko Eckert		Phone 07062/911-4332

4.3.3. Hexfile Header

HEX file flags	Must be at the fix address 0x00000028 for all devices !!!		
	Bit 15	x	RESERVED - must be zero
	Bit 14	0	no Public Key / Signature area defined for current block
		1	Public Key / Signature area for current block defined by pointer at offset 0x38
	Bit 13	0	no Consistency Check performed for current block
		1	Consistency address of current block determined by pointer at offset 0x3C
	Bit 12	0	current block is NO CAL block
		1	current block is CAL block
	Bit 11	0	current block does NOT contain boot data
		1	current block contains boot data
	Bit 10	0	current block is NO FSW/code block
		1	current block is FSW/code block
	Bit 9	0	current block is NO partial CAL block
		1	current block is partial CAL block
	Bit 8-0	xxxxxxxx	RESERVED - must be zero
Block size	Must be at the fix address 0x0000002A for all devices !!!		
Boot,Code,Cal	Bit 15-0	xxxxxxxxxxxxxx	current block size (1bit = 8kByte = 0x2000) for example: 0100 (bin) == 4 (dez) := 0x8000 == 32kbytes
partial CAL	Bit 15-0	xxxxxxxxxxxxxx	current block size (1bit = 16Byte = 0x10) for example: 0100 (bin) == 4 (dez) := 0x0040 == 64bytes
Hexfile ID	Must be at the fix address 0x00000030 for all devices !!! Must be in range of 0x00000003-0x000000FF		

Caution: partial CALblocks are not supported on current HSW mainpath. This feature can only be used by GM_D2XX on project specific HSW sidepath.

Address	Size	Name	Content	Filled by
0x00000020	4 Bytes	reserved	hex:0xB6	Gen9ProDB
0x00000024	4 Bytes	DLM-Pointer	hex:<0x00000000>-<0xFFFFFFFF>	Gen9ProDB
0x00000028	2 Bytes	Hexfile Flags	bitwise:<0xxxxx0000000000>	HSW
0x0000002A	2 Bytes	Blocksize	hex:<0x00000000>-<0x001FFFFFFF>	HSW
0x0000002C	4 Bytes	Device ID	hex:<0x00000000>-<0xFFFFFFFF>	HSW
0x00000030	4 Bytes	Hexfile ID	hex:<0x00000003>-<0x0000FFFF> hex:<0xEEEEEEEEA> hex:<0xEEEEEEEEC> hex:<0xEEEEEEEEED> hex:<0xEEEEEEEEEF>	HSW RomDRV.dll Gen9ProDB PMSD HawCC
0x00000034	16 Bytes	Reference Pointers	hex:<0x00000000>-<0xFFFFFFFF>	HSW

BOSCH  CC/ESM2	Hex File Handling in Gen9 projects	Edition 1.16.2	Page 16/32	Date 14.07.2016
	Documentation	Author Heiko Eckert		Phone 07062/911-4332

5. SW Hexinfo block definitions

The HSW/PSW development has to provide a const structure inside the hexfile as placeholder for the MTC hexfile handling tools. The first byte of that block is an identifier describing the subsequent block structure enabling the tool to write the appropriate data into it. The identifier will be incremented if changes in the data structure are made. A pointer to the base address of the SW hex info block has to be programmed into the CHECK block at offset 0x00.

The following chapters describe the currently defined block structures.


All structures are coded in big endian.

The Hexfile information is written to a location inside every specified block of the current hexfile.

Following conclusion has been found concerning multi-block concept support(split code/data sections):

- An additional consistency checksum will be calculated and placed within hexinfo structure. The original ROM checksum will include hexinfo area, while consistency checksum will exclude this area from calculation.

It is a requirement from CC/ECC FMEA to include hexinfo structure into ROM checksum calculation!

BOSCH  CC/ESM2	Hex File Handling in Gen9 projects	Edition 1.16.2	Page 17/32	Date 14.07.2016
	Documentation	Author Heiko Eckert		Phone 07062/911-4332


5.1. Valid HexInfo structure IDs

5.1.1. Structure IDs 0x09/0x89

Hexinfo structure IDs used for all Gen9 devices.
The colored Fields are present within CAL blocks (ID 0x89) and FSW (ID 0xC9) only.

0x09: Bootblock/FSW/SingleBlockFile (default area -> white)
0x89: EOL CAL blocks (area colored in red in addition to default area)

Rel. Address	Field name	Size	C# Type	Example	Tool	Comment
0x00	Block structure ID	1 Byte	byte (Byte)	0x89	HSW/PSW	0x01 - 0xFE
0x01	BB number	5 Byte	string (ASCII)	"25409"	Gen9ProDB EPK	nnnnn mit n=[0-9]
0x06	Version Type	1 Byte	char (ASCII)	"E"	Gen9ProDB EPK	V: Version, E: Edition, T: Testversion
0x07	Base Version	4 Byte	string (ASCII)	"0200" "B799"	Gen9ProDB EPK	= 2.0 (1.0 - 359.99) = 117.99
0x0B	Branch 1	4 Byte	string (ASCII)	"0101"	Gen9ProDB EPK	= 1.1 (1.0 - 99.99)
0x0F	Branch 2	4 Byte	string (ASCII)	"0000"	Gen9ProDB EPK	= no branch (1.0 – 99.99)
0x13	Build No.	3 Byte	string (ASCII)	"097"	Gen9ProDB EPK	000-999
0x16	Build Sample	3 Byte	string (ASCII)	"XCB"	Gen9ProDB EPK	Make Mode: X C P E Target: M C P Sample: A B C ...
0x19	Configuration Number	2 Byte	string (ASCII)	"02"	Gen9ProDB EPK	Defines a valid MTC9 configuration
0x1B	---	1 Byte	---	---	---	reserved
0x1C	MTC version	4 Byte	uint (UInt32)	0x07000800	Gen9ProDB	MTC version without "."
0x20	Date	10 Byte	string (ASCII)	"2007-05-05"	Gen9ProDB	see chapter 5.2.5 Date & Time
0x2A	Time	8 Byte	string (ASCII)	"15:34:07"	Gen9ProDB	see chapter 5.2.5 Date & Time
0x32	Developer	8 Byte	string (ASCII)	"ECK2SI "	Gen9ProDB	<Win2000 user>
0x3A	Application version	2 Byte	uint (UInt16)	0x0012	Application tool	see chapter 5.2.10 Application Data
0x3C	Application date	10 Byte	string (ASCII)	"2000-11-06"	Application tool	"yyyy-mm-dd"
0x46	Application time	8 Byte	string (ASCII)	"17:33:05"	Application tool	"hh:mm:ss"
0x4E	Application engineer	8 Byte	string (ASCII)	"ECK2SI "	Application tool	<NT user>
0x56	---	2 Byte	---	---	---	reserved
0x58	Consistency Checksum	8 Byte	ulong (UInt64)	0xBF234569ABC DEF12	Gen9ProDB	see chapter 5.2.6 Consistency Checksum
0x60	Logistic Number (CAL-BB)	5 Byte	string (ASCII)	"12345"	Gen9ProDB	nnnnn mit n=[0-9]
0x65	---	1 Byte	---	---	Gen9ProDB	Reserved (0x00)
0x66	Default Variant	1 Byte	byte (Byte)	0x02	Gen9ProDB	= 01-99
0x67	Customer Data	32 Byte	string (ASCII)	"Opel X4400"	Gen9ProDB	32byte freetext

BOSCH  CC/ESM2	Hex File Handling in Gen9 projects	Edition 1.16.2	Page 18/32	Date 14.07.2016
	Documentation	Author Heiko Eckert		Phone 07062/911-4332

5.2. Coding of HexInfo entries

5.2.1. MTC version

MTC version used to generate current hexfile.

For every significant number one byte is used for coding.

Thus a range from MTC 1.0.0.0 to MTC 99.99.99.99 can be coded.

Development versions are coded by "D" in the last part of the number.

```
Example:  MTC 7.0.8.0          ->  0x07000800
          MTC 9.2.11.0         ->  0x09021100
          MTC 9.3 DevLatest    ->  0x090300D0
```

5.2.2. Base Version

The Base Version is used to identify the version of the used ECU-SW.

To extend the normal range for 4-byte ASCII coding (1.0-99.99) we use the characters from A to Z.

```
Example:  1.0      ->  0100      ->  0x30313030
          75.5     ->  7505      ->  0x37353035
          108.37   ->  A837      ->  0x41383337
          350.99   ->  Z099      ->  0x5A303939
```


The range for SW Base Numbers is extended this way to 1.0-359.99.

The Base Version is written to TFL by TCM. The SharCC tool will read this data and transfer it to EPK (which is written to hexfile by Gen9ProDB).

5.2.3. Build No.

The **Build No.** field contains the current build number if the field „Version Type“ == „E“ (Edition): 000-999.

For Versions from TCM the **Build No.** field is empty: 000

BOSCH  CC/ESM2	Hex File Handling in Gen9 projects	Edition 1.16.2	Page 19/32	Date 14.07.2016
	Documentation	Author Heiko Eckert		Phone 07062/911-4332

5.2.4. Build sample

1st character: SW type = [P|X|C|E] (P: Platform SW, X: XPASS SW, C: Complete SW, E: ePSW)

2nd character: HW platform = [M|C|P]

M:	Emulator (currently unused -> no Emulator SW in gen9)
C:	ECU (default)
P:	Prototype ECU (currently unused)

3rd character: HW variant = [A-E] (A: variant A, B: variant B, ... , E: variant E)

Examples:	
XCA:	XPASS SW for HW variant A
PCC:	Platform SW for HW variant C
CCD:	Complete SW for HW variant D

5.2.5. Date & Time

Date & Time Information is patched by Gen9ProDB into the hexfile.

Coding of **Date & Time** if field „Version Type“ == „E“ (Edition) or „Version Type“ == „T“ (Testversion):

Actual system time. The Make Tool Chain may not pass Date & Time parameters.

Coding of **Date & Time** if field „Version Type“ == „V“ (Version):

Date & Time of the project version. This information is written into the TFL file list, when the configuration management system TCM creates it.

5.2.6. Coding of application information


Application version, application date/time and application engineer are patched by application tool (e.g. PMSD, INCA, Canape).

Application tool patches actual date/time to all hexinfo structures within current file.

Application tool patches application engineer to all hexinfo structures within current file.

Application version will be increased by one for every new hexfile version written by application tool.

External application tools (INCA/CANAPE) will use RomDRV.dll to patch this CC specific data to the output hexfile.

BOSCH  CC/ESM2	Hex File Handling in Gen9 projects	Edition 1.16.2	Page 20/32	Date 14.07.2016
	Documentation	Author Heiko Eckert		Phone 07062/911-4332

5.2.7. Consistency checksum

Consistency checksum is used to ensure that all hexfile parts (BOOT/CODE/CAL) flashed to ECU are from the same MTC build.

Consistency checksum is calculated and patched by Gen9ProDB to all hexfile blocks with consistency check turned on.

Consistency checksum uses same algorithm than standard ROM checksum calculation but not same range (Hexinfo area is excluded from consistency checksum calculation).

Start of hexinfo area (hexinfo_start) is coded at address CHECK + 0x00.

Length of hexinfo area (hexinfo_length) is coded at address CHECK + 0x07.

5.2.8. Logistic data

Logistic data will be available in hexfile information for variant calibration blocks only.


Logistic data will be read out of file ParameterAllocation_BB<xxxxxx>.xml and stored to hexinfo by Gen9ProDB.

Logistic data must contain the following information:

- **DEFAULT_VARIANT:** This variant will be patched to complete hexfile generated by MTC.
- **LOGISTIC_NUMBER:** BB-Number of actual variant CAL block (or a project specific number if a separate Cal-BB is not required).
- **CUSTOMER_DATA:** special comment / customer specific number

The current variant is no longer written to hexinfo for MTC>=9.9 (background is that there is the possibility to write more than one variant into one calblock, thus an explicit variant number is not available).

For additional information on configuring the Hexinfo content via ParameterAllocation_BB<xxxxxx>.xml for EOL projects see the following document: [Multi block hexfile handling.pdf](#)

BOSCH  CC/ESM2	Hex File Handling in Gen9 projects	Edition 1.16.2	Page 21/32	Date 14.07.2016
	Documentation	Author Heiko Eckert		Phone 07062/911-4332

5.2.9. MTC Configuration

Every SW generated with MTC9 gets a configuration name which can be specified by user. This configuration name will be coded in the name of every MTC generated output file.

Example for a valid configuration name: CSW_ECB_NotFinalized

Example for the naming of the corresponding MTC generated hexfile:

PRJ_Hexfile_BB90058_V07010100_ECB_ **CSW_ECB_NotFinalized**.hex

MTC9 will reserve a unique number for every configuration (starting with "1") which will be coded in hexinfo as an ASCII string.

Example:

CSW_ECB_NotFinalized	1
CSW_ECB_Finalized	2
PSW_ECB	3
PSW_ECC	4

The MTC configuration number will be added to EPK since MTC 9.3.


5.2.10. Application Data

All application tools modifying a MTC generated hexfile must fill in the following values in hexinfo:

Application Date:	Date of modification
Application Time:	Time of modification
Application Engineer:	The UserID of the person, which did the modification
Application Version:	BuildNumber of the current used version of application tool

The BuildNumber must be a unique number which identifies the version of the used application tool. It must be provided by the application tool itself.

Additionally the name of the currently used application tool can be extracted out of the HexID at address 0x00000030 of the current hexfile.

BOSCH  CC/ESM2	Hex File Handling in Gen9 projects	Edition 1.16.2	Page 22/32	Date 14.07.2016
	Documentation	Author Heiko Eckert		Phone 07062/911-4332

6. CHECK block coding

6.1.1. CHECK block location

The CHECK block location is defined directly in hexfile. The CheckRef pointer at address <blockstart>+0x00000034 points to the beginning of the CHECK structure in code of current block.

The checksum algorithm is executed for every block of the hexfile. This means every block has its own checksum within its CHECK block. The range for the checksum calculation is ("BlockStart") - („BlockSize"-1). The block size is coded at <blockstart>+0x0000002A in Raw hexfile.

6.2. Calculation of check words


Gen9ProDB has to calculate a 64 bit checksum and patch the result into offset 0x08 of the CHECK block. This checksum will be verified by the ECU SW at run time.

The formula for the checksum calculation is defined by the identifier at offset 0x04/0x05 of the CHECK block.

Valid checksum data is all data except for the following areas:

- CHECK
- all data between CHECK and the end of current block.

The address of the CHECK block is defined at address <block start> + 0x34 in hexfile.

BOSCH  CC/ESM2	Hex File Handling in Gen9 projects	Edition 1.16.2	Page 23/32	Date 14.07.2016
	Documentation	Author Heiko Eckert		Phone 07062/911-4332

6.3. Valid checksum algorithm IDs

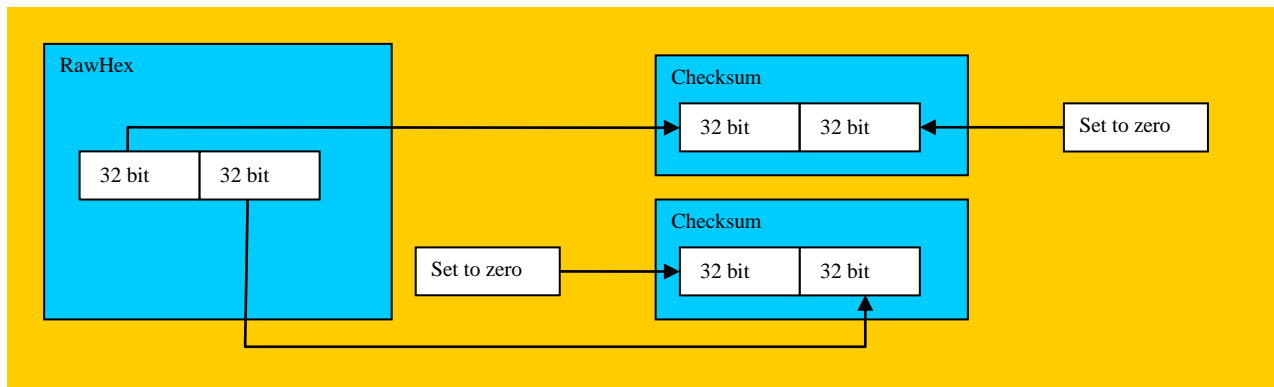
6.3.1. Checksum algorithm 0x83

A 64-bit checksum will be patched to address CHECK+0x08

$CHECK(0x83) = CRC64(\text{valid 32-bit words of address range})$

Address range:

$[(\text{current block's base address}) \dots (\text{current CHECK block address} - 1)]$



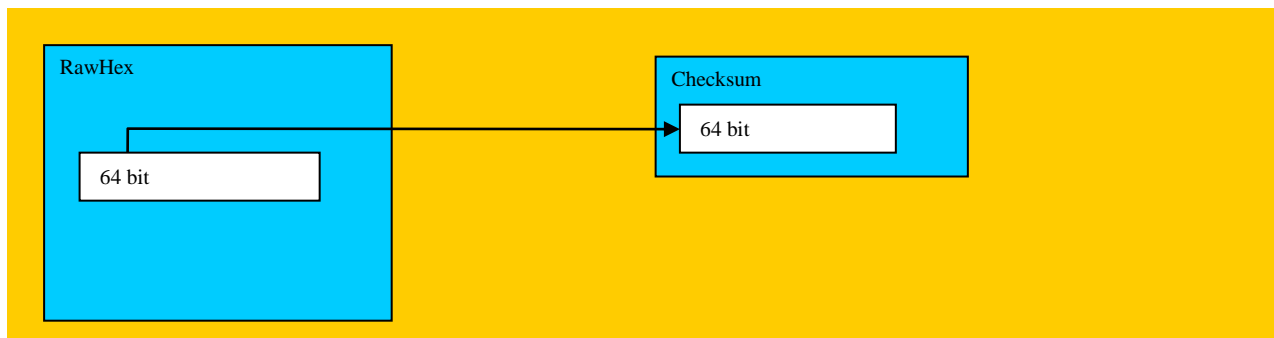
6.3.2. Checksum algorithm 0x84


A 64-bit checksum will be patched to address CHECK+0x08

$CHECK(0x84) = CRC64(\text{valid 64-bit words of address range})$

Address range:

$[(\text{current block's base address}) \dots (\text{current CHECK block address} - 1)]$



BOSCH  CC/ESM2	Hex File Handling in Gen9 projects	Edition 1.16.2	Page 24/32	Date 14.07.2016
	Documentation	Author Heiko Eckert		Phone 07062/911-4332

6.4. Consistency checks for hexfile blocks

The verification that hexfile blocks are consistent is mandatory for the correct behavior of the ECU SW. The following checks shall be implemented:


- Hexfile block vs. hexfile block
This shall be an additional functionality of the HawCC tool.
- Hexfile block vs. ECU block
This shall be an additional functionality of Reprog tools.
- ECU block vs. ECU block
This shall be an additional functionality of boot block and/or function SW. It is recommended to lock ECU functionality unless this check is successfully done.

The consistency information is created by the Gen9ProDB tool and patched during build into the different SW blocks. **The data structure containing the consistency check data must be located in an area, which is not regarded for checksum calculation**, i.e. AFTER the CHECK block itself. If a consistency block shall be created bit 13 of hexfile flags must be set and at offset 0x3C a pointer to the consistency check area must be initialized accordingly.

The consistency check block record contains a pointer to data in other blocks. It is the task of the ECU project to set up this pointer correctly. The check tool will - after calculation of all checksums - copy the data described by the pointers to the appropriate data fields. Thus any consistency check between any hexfile blocks could be implemented.

The structure of the consistency check area is described here:

Rel. Address	Field name	Size	Type	Example	Set by	Comment
0x00	Block structure ID	2 Byte	uint16	0x02	PSW	0x01 - 0xFE (currently the one and only valid ID is 0x02)
0x02	reserved	2 Byte	---	0xFF	PSW	0xFF
0x04	Pointer	4 Byte	uint32*	0x001FFF00	PSW	
0x08	Data 0	8 Byte	uint64	0x0123456789ABCDEF	Gen9ProDB	Consistency checksum

BOSCH  CC/ESM2	Hex File Handling in Gen9 projects	Edition 1.16.2	Page 25/32	Date 14.07.2016
	Documentation	Author Heiko Eckert		Phone 07062/911-4332


The block structure ID gives information about the internal structure of the consistency structure itself.

Currently the only valid consistency block structure ID is 0x02.

In former times 0x01 was used as well (gen8).

Typically consistency checks are used in projects which take part in code/data separation.

Bit 13 of hexfile flags and ConsistencyRef are set in CAL block then. The pointer in CAL consistency structure points typically to consistency checksum in hexinfo of the codeblock. Gen9ProDB will copy the code consistency checksum to consistency structure of the CAL block then.

BOSCH  CC/ESM2	Hex File Handling in Gen9 projects	Edition 1.16.2	Page 26/32	Date 14.07.2016
	Documentation	Author Heiko Eckert		Phone 07062/911-4332

7. Device Description file ti_galaxy.ddf

The device description file (DDF) contains all data that is necessary to manipulate / interpret a hexfile for a given device.

7.1. Definition


The DDF has the Windows INI file format, with the following entries:

```
[DevID_0x<hhhhhhhh>]
ProductName=<identifier>
AliasName=<identifier>
Memory_Start=<start_address>
Memory_Length=<length>
RAM_Start=<start_address>
RAM_Length=<length>

FlashDocumentation_Start=<start_address>
FlashDocumentation_Length=<length>
FlashProtection_Start=0x00000000
FlashProtection_Length=0x00000000
FlashSafetyTool=<name_of_executable>
FlashSafetyPath=<path of executable>
FlashSafetyVersion=<version of executable>
FlashSafetyParams=<calling parameters for flash security tool>
FlashSafety_Start=<start_address>
FlashSafety_Length=<length>
FlashSafety_Start<n>=<start_address>
FlashSafety_Length<n>=<length>
SD_Start<n>=<start_address>
SD_Length<n>=<length>
SD_SectorSize<n>=<length>
```


*) All entries must occur only once per device, except the FlashSafety_Start, FlashSafety_Length, SD_Start, SD_Length and SD_SectorSize entries.

```
<identifier>:      name
<start_address>:   hexadecimal number (8 digits)
<length>:          hexadecimal number (8 digits)
<n>:               1..n, number starting with 1 following fields with
                  incremented number
<name_of_executable>: par470.exe|ecc470.exe
```

BOSCH  CC/ESM2	Hex File Handling in Gen9 projects	Edition 1.16.2	Page 27/32	Date 14.07.2016
	Documentation	Author Heiko Eckert		Phone 07062/911-4332

7.2. Example

```
[DevID_0x000C6B05]
ProductName=TMS470PSF761
AliasName=Wega_F035
Memory_Start=0x00000000
Memory_Length=0x000C0000
RAM_Start=0x08000000
RAM_Length=0x0000C000
FlashDocumentation_Start=0x000BFFF0
FlashDocumentation_Length=0x00000010
FlashProtection_Start=0x00000000
FlashProtection_Length=0x00000000
FlashProtEmulation_Start=0x00000000
FlashProtEmulation_Length=0x00000000
FlashSafetyTool=par470.exe
FlashSafetyPath=c:\gen9base\hexfilehandling\par470
FlashSafetyVersion=V_1_12
FlashSafetyParams=-a -odd -lf035
FlashSafety_Start=<0x02000000>
FlashSafety_Length=<0x00020000>
FlashSafety_Start=0x02000000
FlashSafety_Length=0x00018000
FlashSafety_Start1=0x02000000
FlashSafety_Length1=0x00001000
FlashSafety_Start2=0x02001000
...
SD_Start1=0x00000000
SD_Length1=0x00018000
SD_SectorSize1=0x00008000
SD_Start2=0x00018000
SD_Length2=0x00004000
SD_SectorSize2=0x00002000
SD_Start3=0x0001C000
SD_Length3=0x00004000
SD_SectorSize3=0x00004000
SD_Start4=0x00020000
SD_Length4=0x00020000
SD_SectorSize4=0x00010000
...
```

BOSCH  CC/ESM2	Hex File Handling in Gen9 projects	Edition 1.16.2	Page 28/32	Date 14.07.2016
	Documentation	Author Heiko Eckert		Phone 07062/911-4332


8. External Flash Security Tools

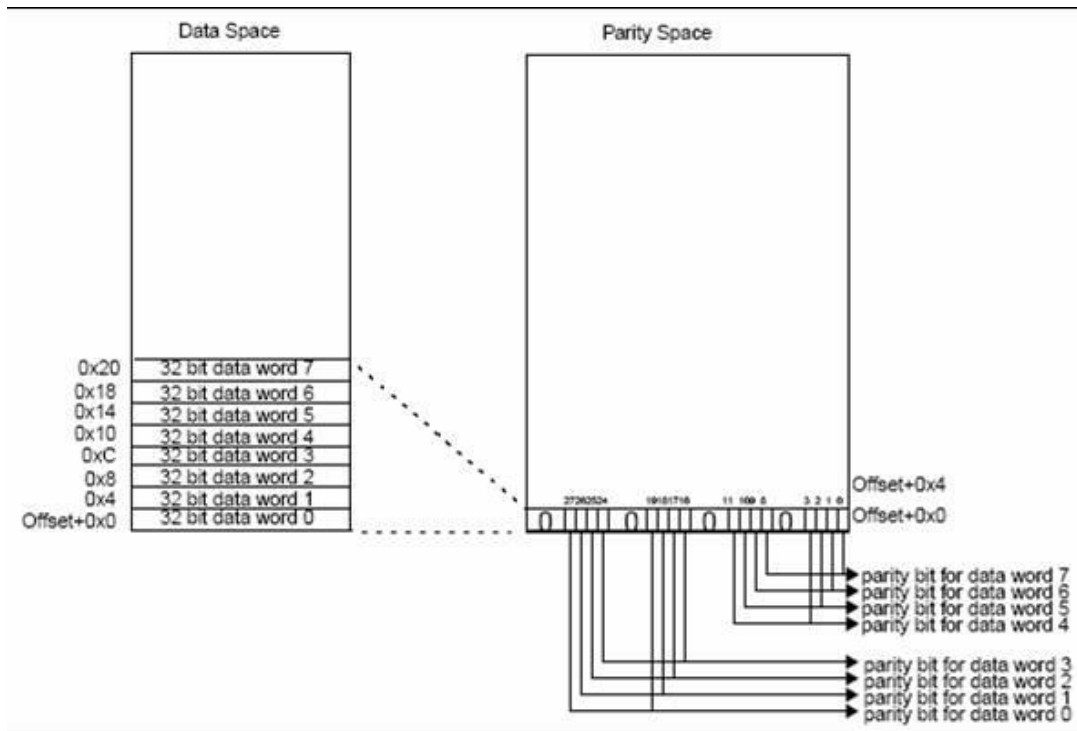
8.1. Configuration / Architecture

Device	Tool	Option	Address Offset
Wega F035	par470.exe	lf035	0x02000000
Wega+ F035	par470.exe	f035	0x00400000
Mira F035	par470.exe	f035	0x00400000
Mira+ F035	ecc470.exe	f035	0x00400000
Sirius F035	par470.exe	f035	0x00400000
Capella F035	ecc470.exe	f035	0x00400000
Capella F021	ecc470.exe	F021	0xF0400000
Procyon F035	par470.exe	f035	0x00400000
Gladiator F021	ecc470.exe	F021	0xF0400000
Carina F021	ecc470.exe	F021	0xF0400000
Corona F021	ecc470.exe	F021	0xF0400000
Rigel F021	Ecc470.exe	F021	0xF0400000

The safety tools will be configured via DDF file. Thus a new entry has been defined:

```
FlashSafetyTool=<par470.exe|ecc470.exe>
FlashSafetyParams=<-a -odd -f035|-a -odd -lf035|-a -r4 -f035|...>
FlashSafety_Start=<start address of parity/ECC region>
FlashSafety_Length=<lenghtgth of parity/ECC region>
```

BOSCH  CC/ESM2	Hex File Handling in Gen9 projects	Edition 1.16.2	Page 29/32	Date 14.07.2016
	Documentation	Author Heiko Eckert		Phone 07062/911-4332



8.2. Flash Parity

To insert parity information into the created hexfiles an external tool must be called.

Example: `par470.exe -i<input file> -a -odd -f035`

For additional information see:

[http://bosch.com/dfsrb/DfsDE/DIV/CS/DE_CS\\$/Tech/uC/TI_NEMICS/Tools/Flash Tools/Flash Parity/par470/par470_userguide.pdf](http://bosch.com/dfsrb/DfsDE/DIV/CS/DE_CS$/Tech/uC/TI_NEMICS/Tools/Flash Tools/Flash Parity/par470/par470_userguide.pdf)


8.3. ECC

To insert ECC information into the created hexfiles an external tool must be called.

Example: `ecc470.exe -i<input file> -a -r4 -f035`

For additional information see:

[http://bosch.com/dfsrb/DfsDE/DIV/CS/DE_CS\\$/Tech/uC/TI_NEMICS/Tools/Flash Tools/Flash ECC/ecc470_tool_user_guide.pdf](http://bosch.com/dfsrb/DfsDE/DIV/CS/DE_CS$/Tech/uC/TI_NEMICS/Tools/Flash Tools/Flash ECC/ecc470_tool_user_guide.pdf)

BOSCH  CC/ESM2	Hex File Handling in Gen9 projects	Edition 1.16.2	Page 30/32	Date 14.07.2016
	Documentation	Author Heiko Eckert		Phone 07062/911-4332

9. HSW

9.1. LCF file handling

9.1.1. DeviceID

The device ID can be found in LCF within a comment at the beginning of the file. The device ID is read by Gen9ProDB tool.

Gen9ProDB checks if the device ID read from LCF file fits together with the device ID read directly from RawHexfile.

9.1.2. Memory regions

The memory regions are defined in LCF file as well. They are read by Gen9ProDB and are written to Gen9ProDB XML file.

The memory regions are used by the linker and by INCA application tool.


9.1.3. Important Links

LCF file handling is described in an external HSW related document:

[\\bosch.com\dfsrb\DfsDE\DIV\CS\DE_CS\\$\Tech\HSW_Doc\Gen09\SW-Groups\LCF09_COMMON\Documentation\SWDesignSpec_LCF09_COMMON_doc.tcm](\\bosch.com\dfsrb\DfsDE\DIV\CS\DE_CS$\Tech\HSW_Doc\Gen09\SW-Groups\LCF09_COMMON\Documentation\SWDesignSpec_LCF09_COMMON_doc.tcm)

Linker / Assembler description can be found here:

[\\bosch.com\dfsrb\DfsDE\DIV\CS\DE_CS\\$\Tech\HSW_Doc\Gen09\SW-Groups\LCF09_COMMON\Documentation\TI_Documents\V4_4_x\SPNU118G.pdf](\\bosch.com\dfsrb\DfsDE\DIV\CS\DE_CS$\Tech\HSW_Doc\Gen09\SW-Groups\LCF09_COMMON\Documentation\TI_Documents\V4_4_x\SPNU118G.pdf)

BOSCH  CC/ESM2	Hex File Handling in Gen9 projects	Edition 1.16.2	Page 31/32	Date 14.07.2016
	Documentation	Author Heiko Eckert		Phone 07062/911-4332

9.2. LCF09_COMMON

9.2.1. Hexfile Flags

Allocation of hexfile flags is defined in

..\Psw\HSW\CMHSW_Config\SRC\LCF09_HEXBLOCKS_ALLOCATION_BBxxxxx.C

Example1 (single block project):

```
#define S_HexFlagSignature_UL      (0<<14)
#define S_HexFlagConsistency_UL   (0<<13)
#define S_HexFlagFswBlock_UL      (1<<10)

#define S_FswHexFlag_UL ((UL)(( S_HexFlagSignature_UL
                               | S_HexFlagFswBlock_UL) <<16)
                               | S_FswBlockSize_UL)
```


Example2 (multi-block project):

```
#define S_HexFlagSignature_UL      (1<<14)
#define S_HexFlagConsistency_UL   (1<<13)
#define S_HexFlagCalBlock_UL      (1<<12)
#define S_HexFlagBootBlock_UL     (1<<11)
#define S_HexFlagFswBlock_UL      (1<<10)

#define S_BootHexFlag_UL ((UL)(( S_HexFlagSignature_UL
                               | S_HexFlagBootBlock_UL) <<16)
                               | S_BootBlockSize_UL)

#define S_FswHexFlag_UL ((UL)(( S_HexFlagSignature_UL
                               | S_HexFlagFswBlock_UL) <<16)
                               | S_FswBlockSize_UL)

#define S_CalHexFlag_UL ((UL)(( S_HexFlagSignature_UL
                               | S_HexFlagConsistency_UL
                               | S_HexFlagCalBlock_UL) <<16)
                               | S_CalBlockSize_UL)
```

BOSCH  CC/ESM2	Hex File Handling in Gen9 projects	Edition 1.16.2	Page 32/32	Date 14.07.2016
	Documentation	Author Heiko Eckert		Phone 07062/911-4332

9.2.2. Block design

Size and type and order of blocks are defined in module

..\Psw\CMPSW_Config\HDR\PSW\CM_PSWHexblocksConfig_<BBxxxxx>.h

Example1 (bootblock):

```
#define FS_BootblockSize_0KB 1
/*DE|BootblockSize is 0 KB |*/
#define FS_BootblockSize_16KB 2
/*DE|BootblockSize is 16 KB |*/
#define FS_BootblockSize_32KB 3
/*DE|BootblockSize is 32 KB |*/
#define FS_BootblockSize_64KB 4
/*DE|BootblockSize is 64 KB |*/
#define FS_BootblockSize FS_BootblockSize_32KB /*DU|200607Eck|*/
```

Example2 (calblock):

```
#define FS_CalblockSize_0KB 1
/*DE|size of the calibration block (parameters) is 0kB |*/
#define FS_CalblockSize_8KB 2
/*DE|size of the calibration block (parameters) is 8kB |*/
#define FS_CalblockSize_16KB 3
/*DE|size of the calibration block (parameters) is 16kB |*/
#define FS_CalblockSize FS_CalblockSize_0KB /*DU|130905Eck|*/
```

Example3 (blockorder):

```
#define FS_HexBlksOrder_Fsw 1
/*DE|Project has only one HEX block(default) |*/
#define FS_HexBlksOrder_BootFsw 2
/*DE|Boot Block Project with no Calibration block |*/
#define FS_HexBlksOrder_BootFswCal 3
/*DE|Boot Block Project with Calibration block behind Fsw |*/
#define FS_HexBlksOrder_BootCalFsw 4
/*DE|Boot Block Project with Calibration block before Fsw |*/
#define FS_HexBlksOrder_CustomizedOrder 5
/*DE|Project with Hex blocks order other than the above |*/
#define FS_HexBlksOrder FS_HexBlksOrder_BootFsw /*DU|200607Eck|*/
```