iMOTION™ 2.0 Device Programming

Initial programming and updating of the Motion Control Engine

About this document

Scope and purpose

This document describes the programming and updating the firmware of iMOTION™ devices. This includes the programming of the Motion Control Engine (MCE rev. 2.0) itself as well as the handling of parameter sets and scripts.

All new iMOTION™ products with integrated Motion Control Engine (MCE rev. 2.0) use a secure bootloader mechanism for production (end-of-line) device programming as well as for updating the MCE to a new revision, for example during the development stage. The software images for the individual iMOTION™ devices are made available on the Infineon Technologies website (www.infineon.com/imotion-software).

The firmware package for a specific iMOTION™ device is delivered as an encrypted file and can be used only in combination with the respective chip type and encryption key. Therefore these software images can only be installed into specific device types with a matching key, assuring the compatibility of the MCE with the respective device.

The parameter sets for the configuration of system, motor and power factor correction (PFC) are managed in unencrypted form.

The same applies to scripts for the integrated scripting engine.

Intended audience

This document primarily targets providers of end-of-line programming equipment (gang programmer).

Since the programming of the iMOTION™ devices of the generation 2 only requires a UART as a physical interface, this documentation can also be used by customers for implementing the resp. programming.

References

[1] Product documentation and software can be downloaded from http://www.infineon.com/iMOTION

Table of Contents

1 Overview

This document describes the programming and update management for the iMOTION™ products using the Motion Control Engine revision 2 (MCE 2.0). Devices of this generation like the IMC100 series are shipped without any software or data programmed.

The following steps have to be performed in order to achieve a usable motor controller.

1. Programming of the Motion Control Engine (MCE)
2. Programming of parameter sets for system, motor and power factor correction (PFC – optional)
3. Programming of any customer scripts (optional)
4. Programming of the combined file for an integrated system

Programming of the MCE is done via a secure boot loader using the encrypted images provided by Infineon. This secure loader is part of the device and cannot be changed or erased.

The parameter and script programming is done with the unencrypted data. The respective parameter/script loader is a part of the MCE image meaning that the MCE firmware has to be programmed first.

Reprogramming of the MCE, e.g. in order to upgrade to a newer release is done by entering the secure loader mode. Reprogramming the MCE via secure loader will erase the complete internal Flash memory, including all parameter sets and scripts.

The communication protocol for programming the MCE, parameter sets and scripts is almost identical with slight differences in the commands supported.

1.1 Loader protocol for download tool support

Configuration

The loader uses the UART protocol for communication between the host and the device, with the configuration:

- 8 data bits.
- 1 stop bit.
- No parity.
- LSB first.
- Channel selection based on which RxD pin the first Start and Header bytes are received.

Note: The default UART port is assigned to RX0 and TX0 for software updating.

Operation

Once the download tool has established the communication channel and baud rate, it uses the loader command set (see section 1.2 Loader command set) to carry out the necessary actions on the device. The detailed download flow, including baud rate negotiation, is described in section 3.3 “Detailed description of MCE download flow”.

1.2 Loader command set

Loader commands are used to program MCE, Motor/PFC parameter and script code into the target device and read status. The loader protocol is as follows:

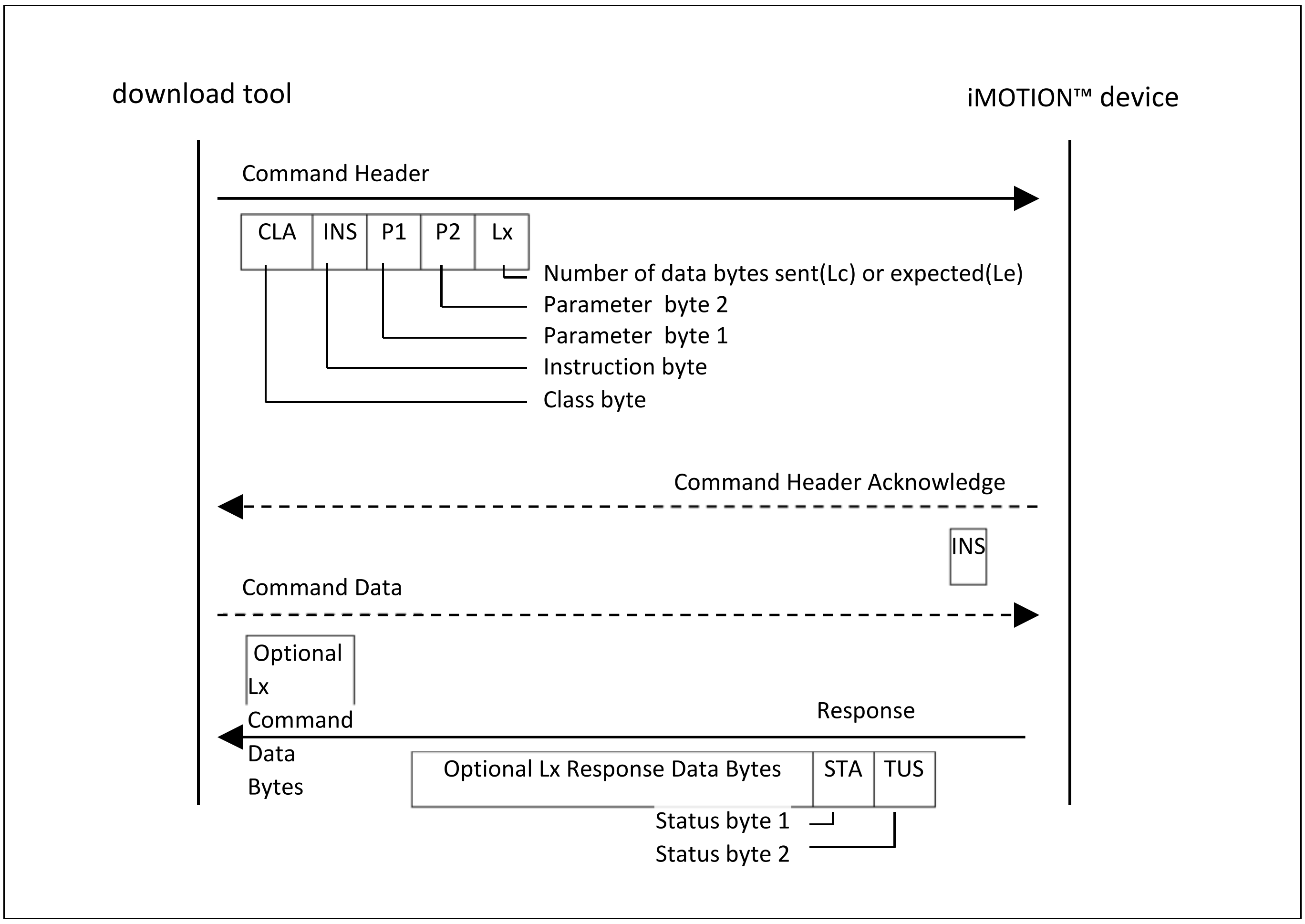


Figure 1 **Data flowchart for the SBSL Loader protocol**

When ‘Lx’ is not equal to zero, the SBSL device will return the received ‘INS’ byte as ‘header acknowledge’. For example for FLASH_CHIP_RESET command (chapter 3.2.3 FLASH_CHIP_RESET), the L_c is 0x00, so there is no ‘header acknowledge’ from the device.

If the loader requires more time, it sends one or more 'Waiting Time Extension Requests' as illustrated by **Figure 2** for the SBSL Loader command ' FLASH_LOAD_CHECK_SIGNATURE ' ('A0 21 00 00 00').

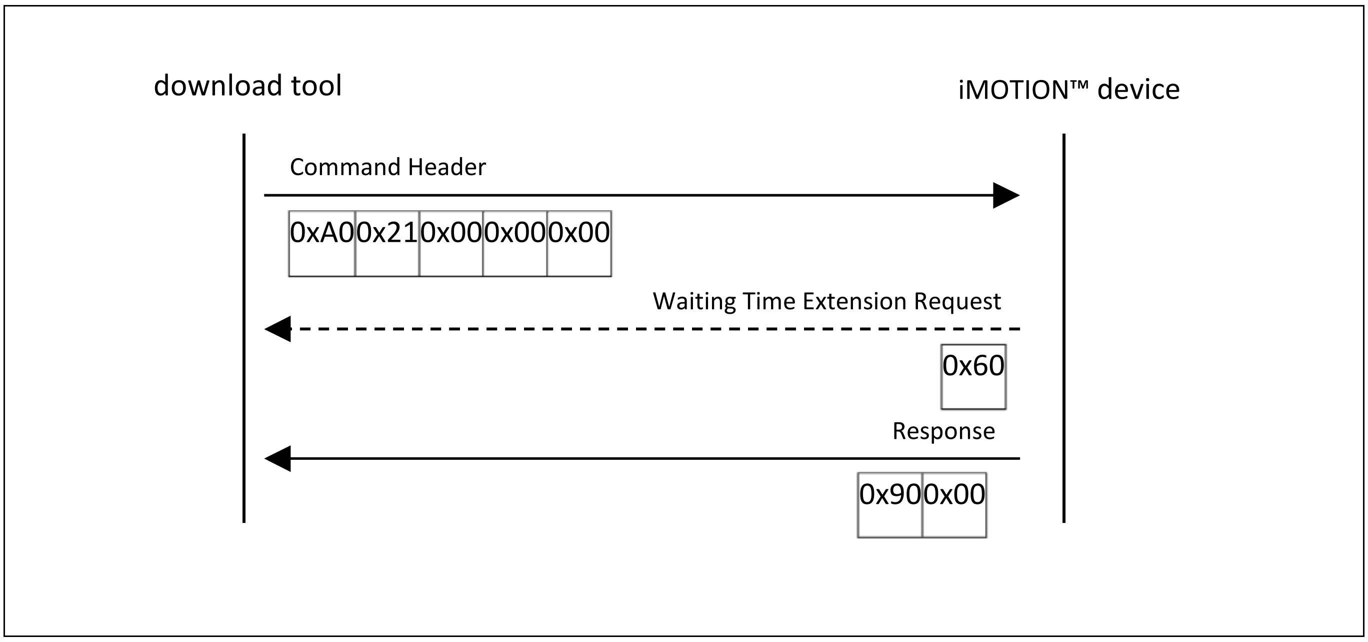


Figure 2 **Data flowchart for Waiting Time Extension Request**

About loader commands

A loader command is identified by two 8-bit integers representing the command class CLA and command instruction INS. It also contains:

- Two 8-bit command parameters P1 and P2.
- An 8-bit field L_c, indicating the number of bytes of the following command data.
- N_c bytes of command data.
- An 8-bit field L_e, indicating the maximum number of response bytes expected.

Commands are designed to transport payload data only in one direction; i.e. to the loader in the command's data field, or from the loader in the response's data field, but not in both at the same time.

1.3 MCE command set

MCE commands are used to communicate with device when device is in Application Mode. The MCE protocol is as follows:

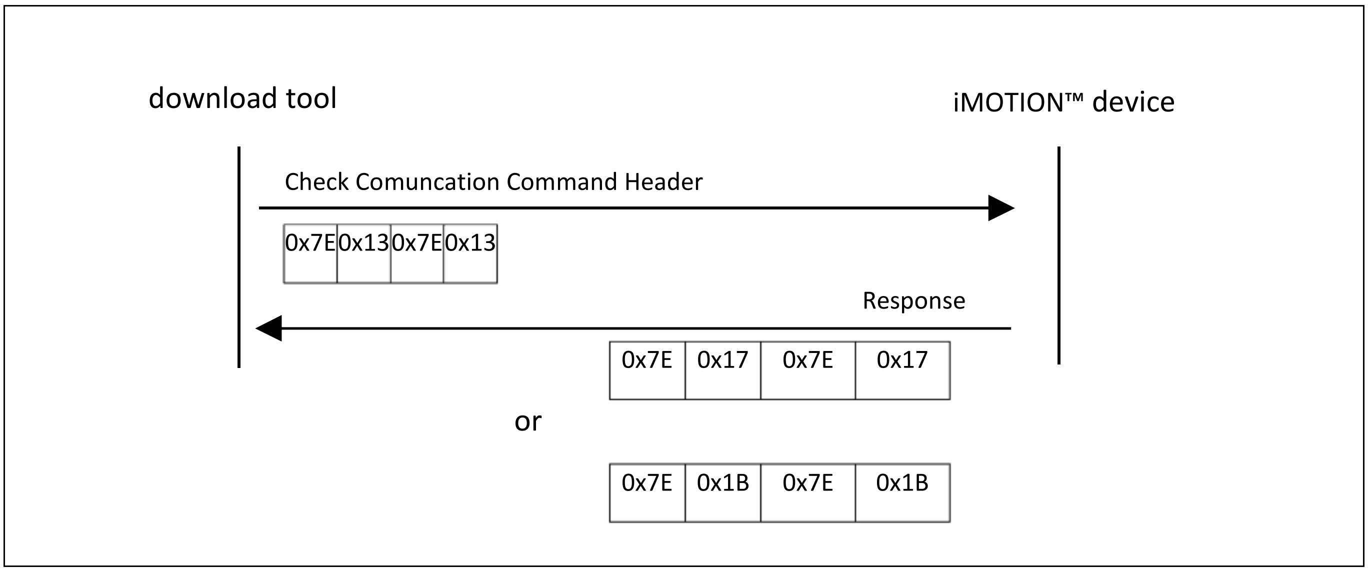


Figure 3 **Data flowchart for the MCE communication check protocol**

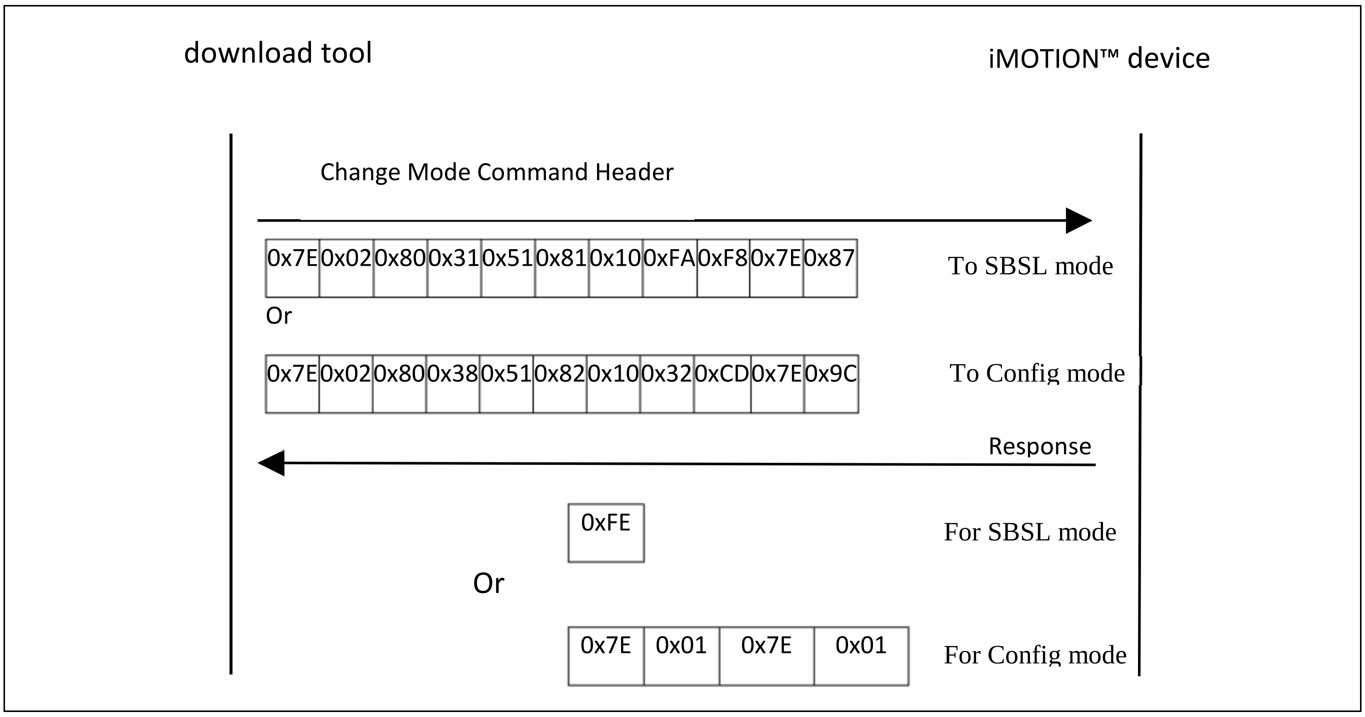

download tool
iMOTION™ device
Change Mode Command Header
0x7E 0x02 0x80 0x31 0x51 0x81 0x10 0xFA 0xF8 0x7E 0x87
To SBSL mode
Or
0x7E 0x02 0x80 0x38 0x51 0x82 0x10 0x32 0xCD 0x7E 0x9C
To Config mode
Response
0xFE
For SBSL mode
Or
0x7E 0x01 0x7E 0x01
For Config mode

Figure 4 **Data flowchart for MCE command to change boot mode**

When MCE firmware and parameters are programed, for update process, we need change the boot mode to Config or SBSL mode. For this operation, we can use MCE command for boot mode change. **Error! Reference source not found.** depicts the dataflow when changing the boot more by using the MCE command.

About MCE commands

An MCE command is identified by special format.

- Connection baud rate should be 115200 bps.
- This command only happens when the MCE code with parameters file programed to device.
- Commutation connection check data frame and mode change frame are fixed type.
- Mode change is handled by MCE firmware command parser with the parameters configuration with command interface.

1.4 Catch at start-up method for changing the boot mode

The device mode catches at start-up is as follows:

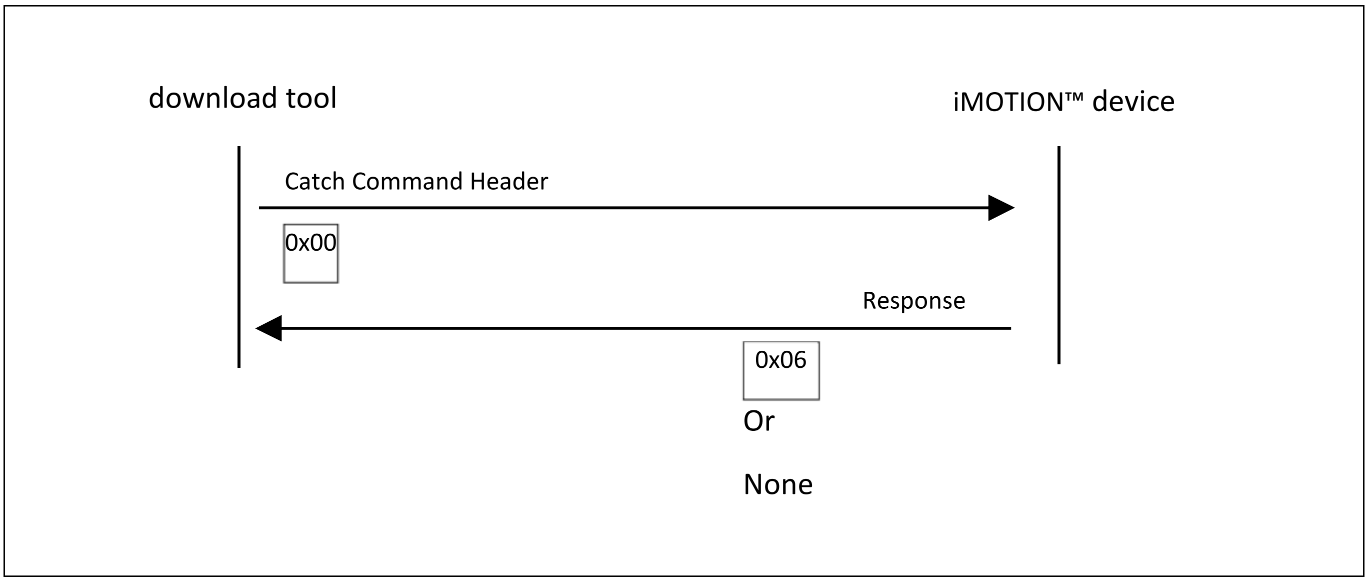


Figure 5 Data flowchart for the catch at start up protocol

When MCE firmware or parameters are programed, for update process, we need to change the boot mode to Config mode. For this operation, we can use catch procedure at start up for boot mode change. Config mode can be entered by sending low pulses with 155us width to pin RXD0 at power-up. This is a 0x00 at 57.600 baud. The system responds with a 0x06 at pin TXD0. Then you can connect normally with CONNECT command.

About catch procedure for mode change:

A 0x00 for low 155us is identified by special format.

- Connection baud rate should be 57600 bps. And a 0x00 will send to device.
- This procedure only works when the MCE code with parameters file programed to device, the device is powered up and the 0x00 is sent to the device.
- Commutation connection check data frame and mode change frame to config are fixed type.
- Mode change is then handled by MCE firmware start up function.

2 Device boot modes description

iMOTION™ devices which are intended for use with the Motion Control Engine (a.k.a. 'turnkey' devices) are shipped from the factory with Secure Bootstrap Loader (SBSL) as the default start-up mode. The SBSL supports the secure transfer of the encrypted MCE software image into the device Flash and assures the compatibility of the software with the device type. Here is the overview of the available device modes.

1) Secure Bootstrap Loader Mode

This mode is default factory setting. After MCE firmware is programed, the device will change to config mode automatically.

2) Config Mode

After programming the MCE firmware, the device switches to this mode automatically, and waits for the parameters to be programmed. It can be switched back to SBSL mode by mode the change command. (CHANGE_BOOT_MODE to SBSL mode) in case the MCE firmware needs to be re-programmed, or it can go to Fail Safe Mode automatically if Class B self-test routines detect the failure. Typically, from this mode the next step will be the change to Application Mode after programmer parameters file.

3) Application Mode

After MCE firmware and parameters files are programed into the device, then boot mode will set to Application Mode automatically. From this mode, in order to change mode to Config mode or SBSL mode, a set of low pulses needs to be applied at power-up (see Section 1.4Catch at start-up Catch at start-up) or a special MCE communication command needs to be sent(see Section 1.3 MCE command set).

4) Fail Safe Mode

The device enters the "Failsafe Mode" as result of ClassB fault. In this mode the user can use Get_Status and Chip Reset commands. Re-programming the firmware requires the command CHANGE_BOOT_MODE to SBSL mode.

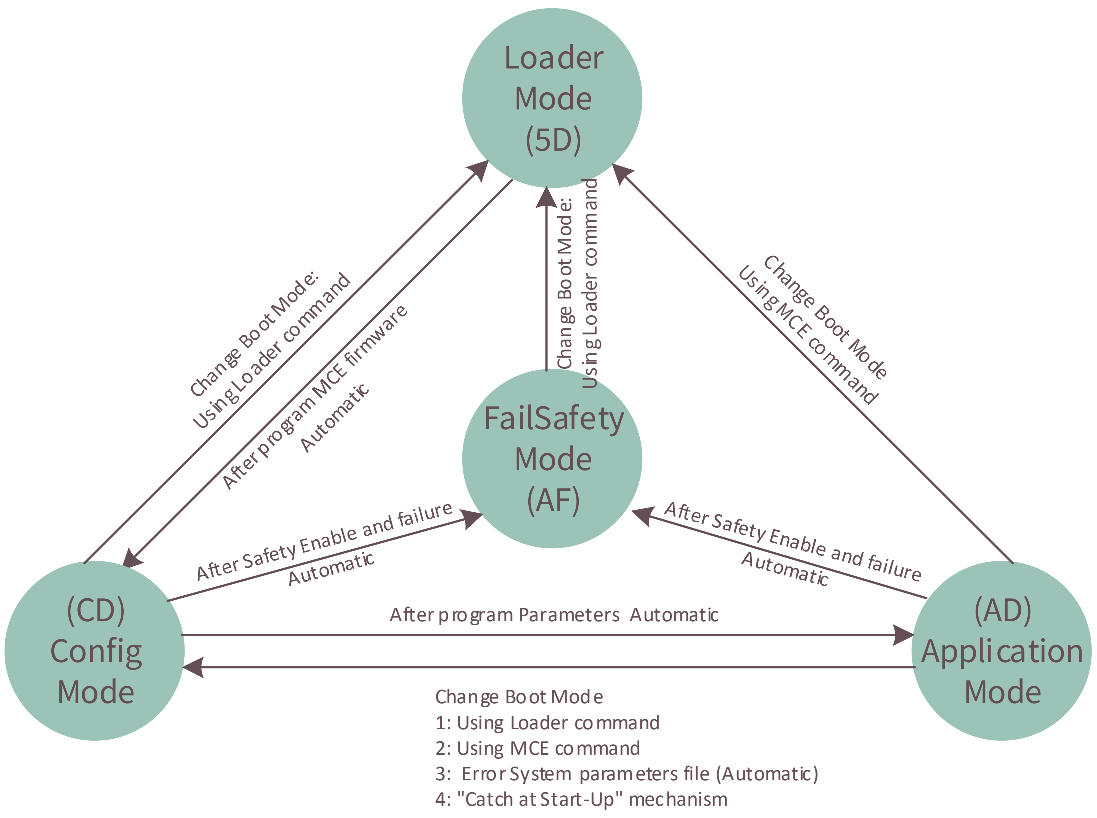

Loader Mode (5D)
Change Boot Mode: Using Loader command
After program MCE firmware Automatic
Change Boot Mode: Using Loader command
Change Boot Mode Using MCE command
FailSafety Mode (AF)
After Safety Enable and failure Automatic
After Safety Enable and failure Automatic
(CD) Config Mode
After program Parameters Automatic
(AD) Application Mode
Change Boot Mode
1: Using Loader command
2: Using MCE command
3: Error System parameters file (Automatic)
4: "Catch at Start-Up" mechanism

Figure 6 **Relation flowchart for device mode**

3 Programming the Motion Control Engine (MCE)

iMOTION™ “turnkey” devices are shipped with Secure Bootstrap Loader (SBSL) as the default start-up mode. The SBSL supports the secure transfer of the encrypted MCE software image into the device Flash and assures the compatibility of the software with the device type. Encryption is based on the Advanced Encryption Standard (AES) with the key size of 128 bits. Following chapter describes the protocol and the commands used for programming iMOTION™ “turnkey” devices and the procedure that user has to follow.

3.1 Prerequisites and basics of usage

The following prerequisites have to be met before starting the programming cycle.

- Encrypted MCE software image needs to be downloaded from www.infineon.com/imotion-software to user’s PC:
  e.g. `IMC101T-T038_A_V1.00.00.ldf` – MCE rev. 1.0 for device IMC101T-T038
- UART connection from PC to the device needs to be available

The MCE software image is provided on the Infineon Technologies website packaged in a *.zip file. In addition to the actual firmware the package contains the license agreement, the release notes and a short readme describing the usage.

3.2 Protocol-specific commands for MCE firmware

This section provides the commands used for communication between the iMOTION™ device and the download tool which are relevant to relevant to device programming. First 2 chapters provide an overview of all commands used and the rest of the section provides detailed information about all the available commands.

3.2.1 Supported Commands

Table 1 below lists the commands supported by the secure loader for programming the Motion Control software image.

Table 1 SBSL commands

CLA	INS	Name	Description
0xA0	0x00	FLASH_CHIP_RESET	Triggers chip reset
0xA0	0x10	FLASH_GET_SBSL_STATUS	Retrieves the 39-byte SBSL status information, *rSbslStatus*
0xA0	0x12	FLASH_CHANGE_KEY	Updates the IP Key, K_{IP} and its label, L_{IP}
0xA0	0x20	FLASH_LOAD_DATA	Loads data to Flash memory
0xA0	0x21	FLASH_LOAD_CHECK_SIGNATURE	Verifies checksum of downloaded data

3.2.2 General command status response

The SBSL always returns a two-byte status word, SW1 and SW2, and data (if applicable) in response to a SBSL command.

Table 2 lists the general command status response values. Any additional command-specific responses are listed in the respective command description.

Table 2 Command status response values SW1-SW2

SW1	SW2	Meaning	Processing status
0x90	0x00	Success	Normal
0x64	0x00	Execution error: NVM unchanged	Execution error
0x65	0x00	Execution error: NVM changed	
0x65	0x81	NVM is changed; memory failure	
0x67	0x00	Wrong length (L_c or L_e)	Checking error
0x69	0x82	Insufficient security state	
0x69	0x83	Authentication method blocked	
0x69	0x84	Reference data not usable	
0x69	0x85	Conditions of use not fulfilled	
0x6A	0x00	Wrong parameters P1 and P2	
0x6A	0x86	Wrong parameters P1 and P2	
0x6C	L'_e	Wrong length L_e, SW2 indicates the expected length L'_e	
0x6D	0x00	Invalid instruction byte (INS)	
0x6E	0x00	Invalid class byte (CLA)	

3.2.3 FLASH_CHIP_RESET

This command triggers a chip reset. The response is returned and the chip reset takes place.

Security

None

Parameters

None

Syntax

Table 3 FLASH_CHIP_RESET syntax

CLA	INS	P1	P2	L_c	Data field	L_e
0xA0	0x00	0x00	0x00	0x00	-	-

Response

Table 4 FLASH_CHIP_RESET response

Data Field	SW1	SW2	Status
-	0x90	0x00	Success

Return value

The command always reports success.

Typical reset happened time after Reset command send

The response is returned and the chip reset takes place after 100ms.

3.2.4 FLASH_GET_SBSL_STATUS

This command retrieves the 39-byte SBSL status information, *rSbslStatus*, from the chip. *rSbslStatus* is useful to determine further steps for handling the SBSL or for comparison to the expected state during personalization.

During SBSL status preparation, the SBSL executes the erase flash procedure, if the SBSL has been previously re-activated and the user flash area is therefore scheduled for erasure. Waiting Time Extension (WTX) requests are sent during flash erase to obey protocol timing. A byte of value 0x60 is sent for each WTX request.

Security

None.

Parameters

None.

Syntax

Table 5 FLASH_GET_SBSL_STATUS syntax

CLA	INS	P1	P2	L_c	Data Field	L_e
0xA0	0x10	0x00	0x00	-	-	0x27

Response

Table 6 FLASH_GET_SBSL_STATUS response

Data field	SW1	SW2	Status
rSbslStatus	0x90	0x00	Success
rSbslStatus	0x65	0x81	Erase procedure failure
-	0x67	0x00	Wrong L_e

Return value

This command returns *rSbslStatus*.

Table 7 rSbslStatus structure

Offset	Bytes	Value	Description
0	4	“SBSL”	Magic name identifying structure
4	1	0xC0	SBSL version tag
5	1	0x04	Length of following data
6	1	0x06	iMOTION™
7	3	vr rb bb	Software version (v), revision (r), build (b)
10	1	0xC1	SBSL patch version tag
11	1	0x03	Length of following data
12	3	vr rb bb	Patch version (v), revision (r), build (b)
15	1	0xC2	SBSL state tag
16	1	0x04	Length of following data

Offset	Bytes	Value	Description
17	1	ULC	**SBSL Unified Life Cycle**
18	1	V	V.0 bit: 0 -> SBSL is not valid; 1 -> SBSL is valid V.1 bit: 0 -> K_{IP} is not valid or set; 1 -> K_{IP} is valid Others: reserved
19	1	0x00	Reserved
20	1	FDTC	**Flash Download Trial Counter** Indicates the current remaining number of download attempts. Every start of a download sequence decreases the value of FDTC by one, upon receiving the first 'FLASH_LOAD_DATA' command. If the download ended successfully (verified by a checksum calculation, see section 3.4.2), FDTC is reset to its initial start value. If the download failed, FDTC remains on its decreased value. In case FDTC has reached 0, further flash downloads are irreversibly blocked and the affected chip needs to be replaced with a new one.
21	1	0xC3	SBSL ID tag
22	1	0x10	Length of following data
23	16	SBSL ID	SBSL ID

3.2.5 FLASH_LOAD_DATA

This command delivers a configurable length of encrypted SBSL download data to the chip. This data is handed over to the decryption module of the SBSL.

After decryption, the data is decoded and complete pages are flashed into the Flash memory. A checksum is computed over all data blocks and validated with the FLASH_LOAD_CHECK_SIGNATURE command.

Note: The flash download sequence must be explicitly finished with a following FLASH_LOAD_CHECK_SIGNATURE command.

Security

None.

Parameters

D_l is the encrypted SBSL data of l bytes.

Syntax

Table 8 FLASH_LOAD_DATA syntax

CLA	INS	P1	P2	L_c	Data field	L_e
0xA0	0x20	0x00	0x00	l	D_l	-

Response

Table 9 FLASH_LOAD_DATA response

Data Field	SW1	SW2	Status
-	0x90	0x00	Success
-	0x64	0x00	Fatal No fab-out state.
-	0x65	0x00	Error updating the SBSL state; for example FDTC.
-	0x65	0x01	Fatal Write to Flash outside user Flash range was suppressed, chip enters sleep mode.
-	0x65	0x81	Fatal NVM programming error occurred, chip enters sleep mode.
-	0x69	0x82	Insufficient security state. No download trials are left for example
-	0x69	0x84	Error in download stream was found.
-	0x69	0x85	Error interpreting a download record.

Return value

The command either returns success or an error status value.

In case the “no fab-out” state (0x64 0x00) is returned, the following causes may apply:

- The chip is not in a fab-out state anymore; i.e. data has already been flashed into the Flash memory using the SBSL. After re-activation of the SBSL, the FLASH_GET_SBSL_STATUS command described in section 3.2.4 must be run to erase the user flash.

In case of a fatal error during execution, the SBSL restarts the chip immediately after sending the command response.

3.2.6 FLASH_LOAD_CHECK_SIGNATURE

This command verifies the checksum computed over all data blocks and finishes the flash download procedure.

The actual download signature verification is performed within the download stream interpretation. Its result is kept in memory until it is retrieved with this command.

Immediately after reporting the status the Secure Boot Strap Loader activates the downloaded user flash software in case of success, or otherwise restarts the chip.

Security

None.

Parameters

None.

Syntax

Table 10 FLASH_LOAD_DATA syntax

CLA	INS	P1	P2	L_c	Data field	L_e
0xA0	0x21	0x00	0x00	0x00	-	-

Response

Table 11 FLASH_LOAD_DATA response

Data Field	SW1	SW2	Status
-	0x90	0x00	Success
-	0x65	0x00	Wrong hash value
-	0x65	0x81	Flash programming error
-	0x69	0x82	No download started

Return Value

The command either returns success or an error status value.

3.3 Detailed description of MCE download flow

The following sections explain the download flow for the motion control engine in more detail.

The download of application software and data to an iMOTION™ device can be performed with any tool implementing UART communication and the SBSL command set.

The following operations are executed during a download:

1. A baudrate for communication between the iMOTION™ chip and the download tool has to be negotiated.
2. The SBSL ID's stored on-chip are used as the base for the decryption of the MCE image, and are have to match the key used for the encryption of the image.
   In case of a mismatch, the download operation will be aborted, and the device will stay in the SBSL mode with the flash content erased.
3. The SBSL ID specific code and data have to be downloaded.

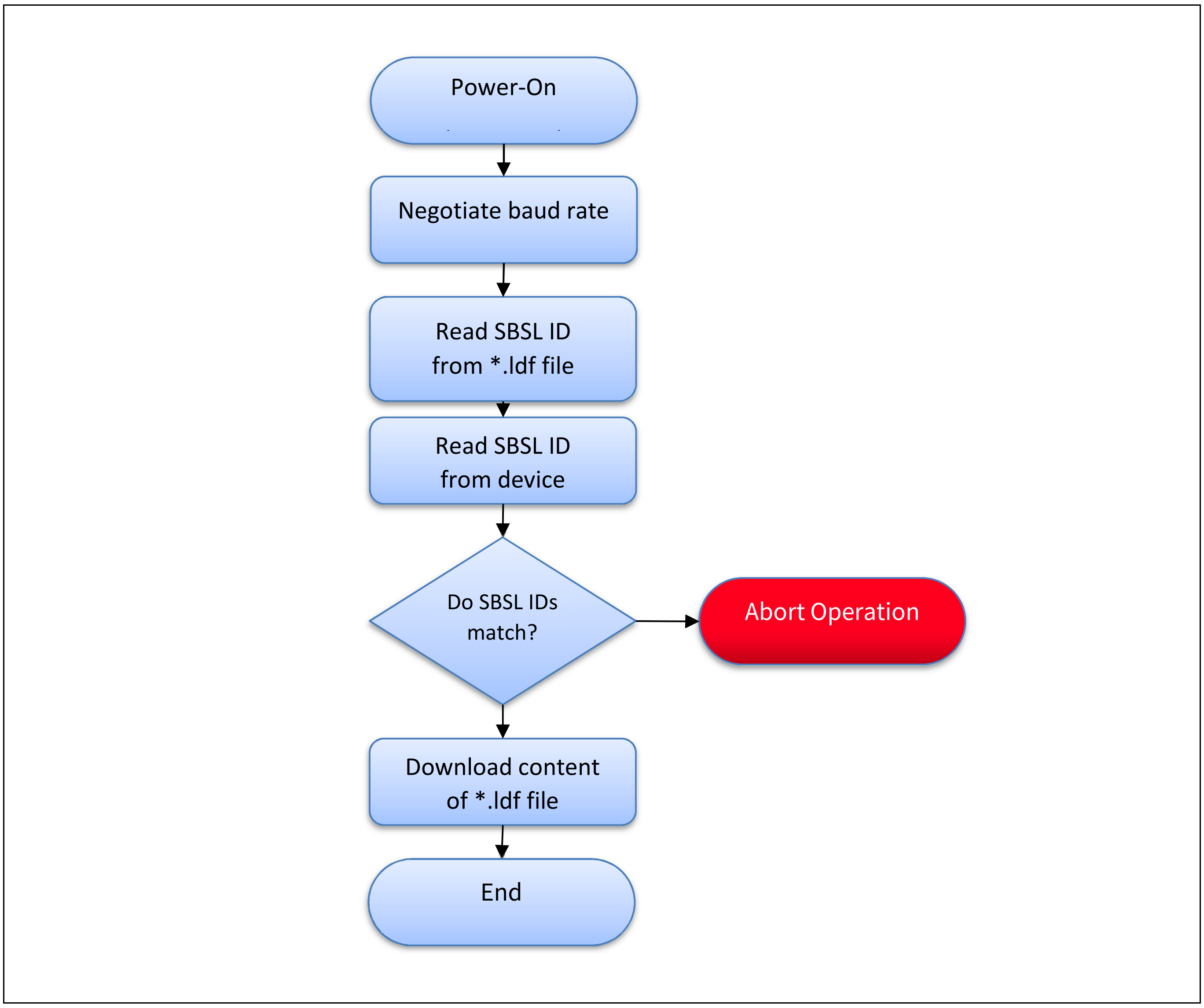


Figure 7 Simplified download flow

3.3.1 Programming pin

For most iMOTION™ devices two UARTs are made available. For programming the iMOTION™ device the first UART 'UART0' has to be used. This is true for both the MCE as well as for parameter and script programming.

UART1 is dedicated to user communication e.g. to send and receive commands to control the speed and report back the status of the drive.

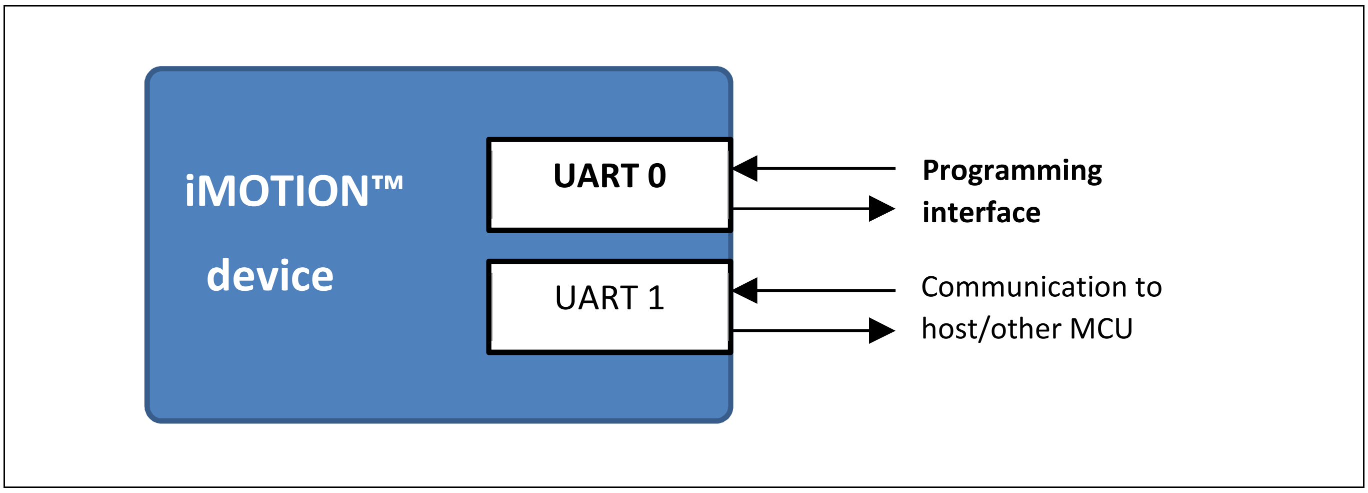


Figure 8 Pins UART0 used for iMOTION™ device programming

3.3.2 Baudrate negotiation

iMOTION™ devices are equipped with baudrate recognition logic working in the range between 300 and 115,200 Baud, depending on the internal MCLK (see **Table 12**).

After a reset, iMOTION™ devices can operate in different baudrate modes:

- Standard Baudrate mode
  - In this mode the device always operates with a fixed baudrate that is determined by the first bytes arriving on UART's Rx line.
- Enhanced Baudrate mode
  - In this mode the device starts with an initial baudrate that is also set by the first bytes arriving on UART's Rx line.
  - Devices initial baudrate can be modified by the download tool to a 'working baudrate' up to 3,996,000 Baud depending on the internal MCLK (see **Table 12**).

3.3.3 Standard Baudrate mode

The "Standard Baudrate mode" is activated by the download tool with the '0x00 0x6C' command. In response, the iMOTION™ device answers with '0x5D'. If a different reply is returned, the device failed to recognize the baudrate used by the download tool.

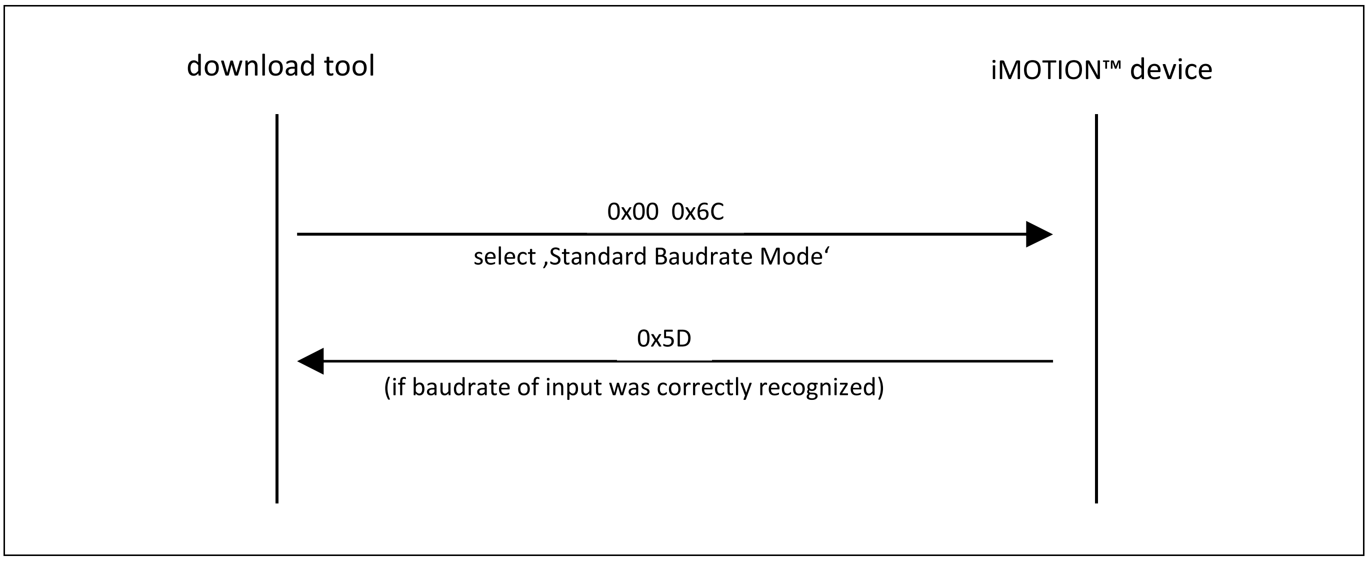

download tool
iMOTION™ device
0x00 0x6C
select ‚Standard Baudrate Mode'
0x5D
(if baudrate of input was correctly recognized)

Figure 9 Protocol flow in Standard Baudrate mode

3.3.4 Enhanced Baudrate mode

The Enhanced Baudrate mode is activated by the download tool via the '0x00 0x93' command.

The device answers with '0xA2' and a 'PDIV' value in the same baudrate.

The download tool analyses the received 'PDIV' value and uses it for the calculation of the 'STEP' value defining the requested 'final baudrate'.

After transmission of '0xF0' by both iMOTION™ device and the download tool, the switch to the 'final baudrate' has been successfully finished.

It should be noted, that the '0xF0' Acknowledge from the iMOTION™ device is transmitted in the initial baudrate, and the download tool uses the 'final baudrate' as shown in Figure 10.

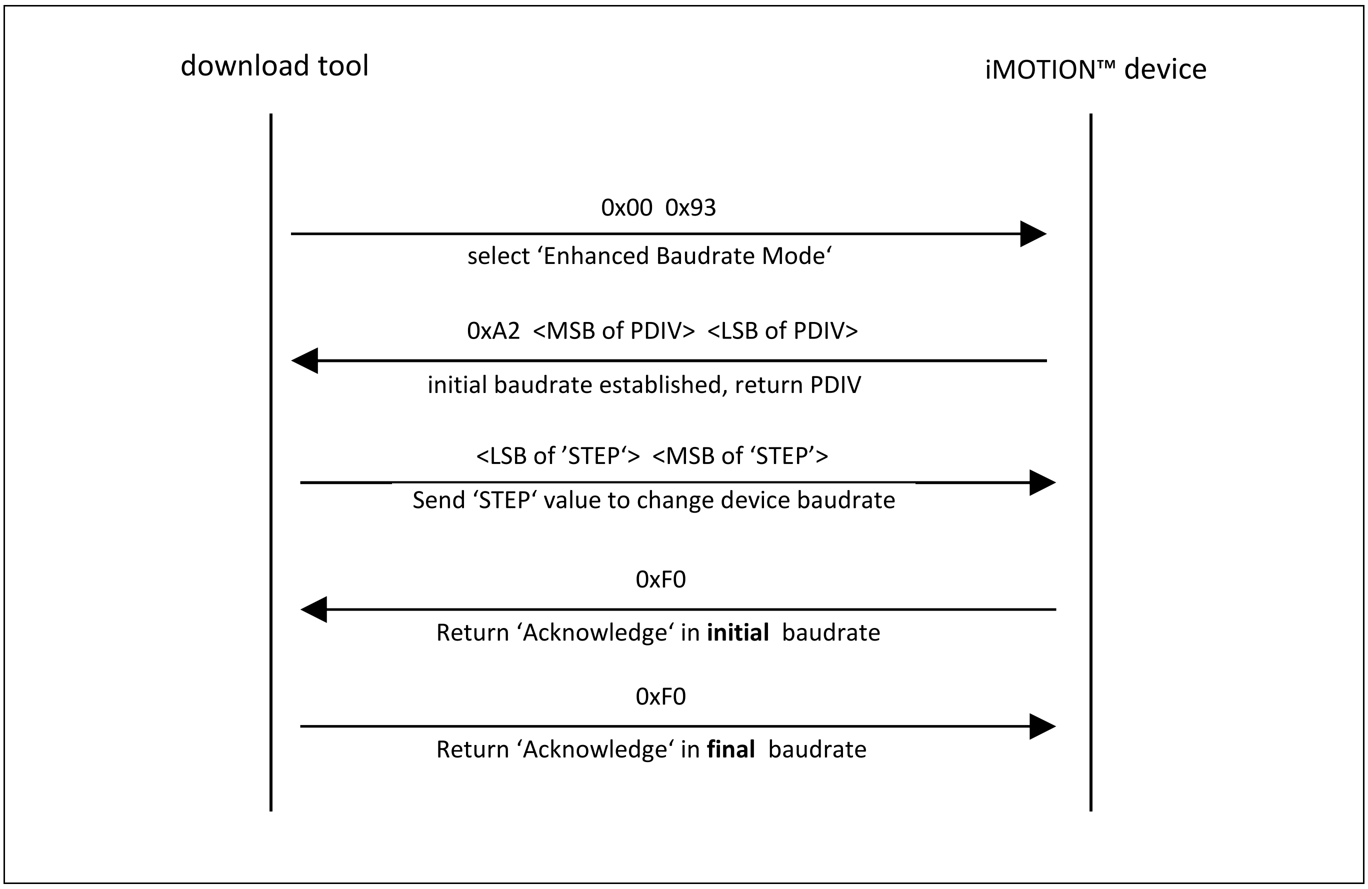

download tool
iMOTION™ device
0x00 0x93
select 'Enhanced Baudrate Mode'
0xA2 <MSB of PDIV> <LSB of PDIV>
initial baudrate established, return PDIV
<LSB of 'STEP'> <MSB of 'STEP'>
Send 'STEP' value to change device baudrate
0xF0
Return 'Acknowledge' in **initial** baudrate
0xF0
Return 'Acknowledge' in **final** baudrate

Figure 10 Protocol flow in Enhanced Baudrate Mode

3.3.4.1 Analysis of PDIV value

The returned PDIV value is used to calculate the Master Clock Frequency (MCLK) in the iMOTION™ device according to following formula:

MCLK = Initial Baudrate x (PDIV + 1) x 8

Example

- Initial Baudrate = 9,600 Baud = 9,600 Hz
- PDIV = 0x00 0x67 = 0x0067 = 103

Results in: MCLK = 9,600 Hz x 104 x 8 = 7.9872 MHz ~ 8 MHz.

As indicated by **Table 12**, there are different minimum and maximum baudrates that are allowed depending on the current MCLK value.

Table 12 Supported Baudrates

MCLK	Minimum Initial Baudrate (Baud)	Maximum Initial Baudrate (Baud)	Maximum Final Baudrate (Baud)
2 MHz (min)	300	7200	249750
8 MHz	1200	28800	999000
16 MHz	2400	57600	1998000
32 MHz	4800	115200	3996000

3.3.4.2 Calculation of STEP value

The STEP value is used to adjust the final baudrate according to the formula:

STEP = 1024 x (Target Baudrate / Initial Baudrate) / (PDIV + 1)

Example

- Initial Baudrate = 9,600 Baud
- Target Baudrate = 115,200 Baud
- PDIV = 0x00 0x67 = 0x0067 = 103

STEP = 1024 x (115,200 / 9,600) / 104 = 118.16 ~ 118 = 0x0076 = 0x00 0x76.

3.4 Examples

3.4.1 Reading SBSL ID out of the iMOTION™ device

The SBSL ID and further information are read via the Flash Get SBSL Status command from the device.

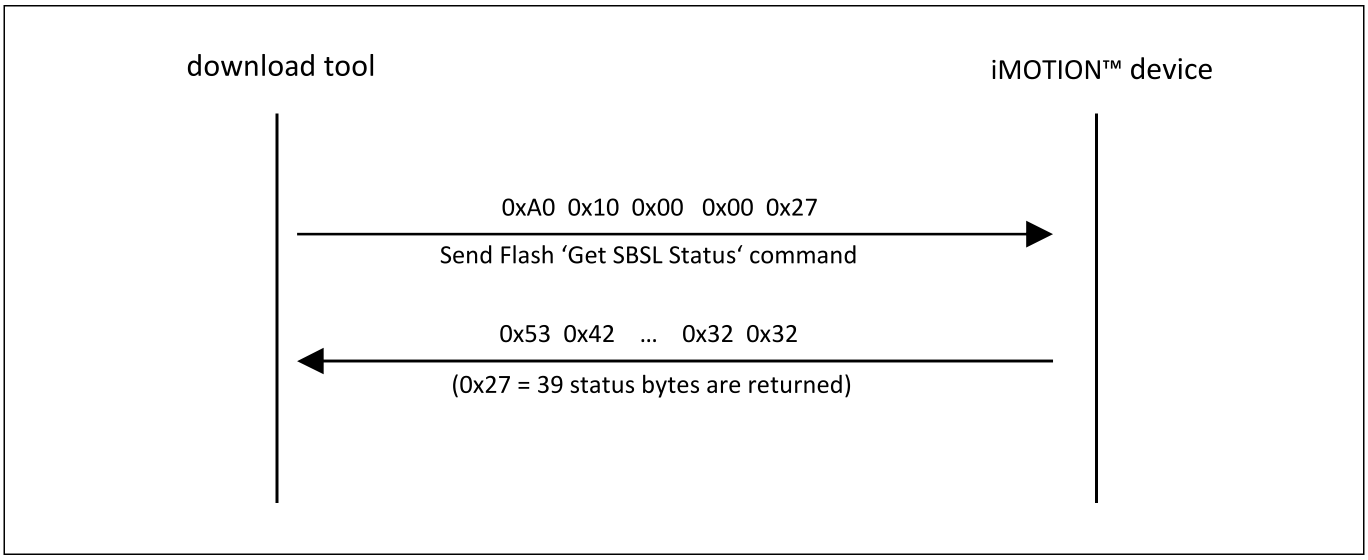


Figure 11 Protocol flow for SBSL Id evaluation

Note: For the layout of the Flash Get SBSL Status reply, please refer to ***Table 7****.*

3.4.2 Download of MCE code and data

File <design>_ip.ldf contains the encrypted application code and data, which have to be downloaded to the iMOTION™ device. The file layout is as follows:

```
# DEVICE:  IMC101T-T038
# RELEASE: A_V1.00.00
# DATE:    31.01.2018
# LIP: 70A0BD00
A0 20 00 00 82  A0 00 00 00  ...  18 1A
A0 20 00 00 82  12 C5 68 80  ...  A8 3B
.
.                         .
A0 20 00 00 44  3F 98 2D 82  ...  F4 CE
A0 21 00 00 00
```

Note: Lines starting with '#' hold comments and must not be sent to the device.

The following lines contain:

- a command APDU
  - 'A0 20 00 00 xx' announcing 'xx' following code and/or data bytes.
  - 'A0 21 00 00 00' initiating a checksum calculation over all downloaded bytes.
- the affiliated code and/or data bytes

When the iMOTION™ device processes successfully the new APDU and <code and/or data bytes> string, it returns '90 00'. Otherwise, a 2 byte 'error SW1 SW2' (e.g. '67 xx') is returned.

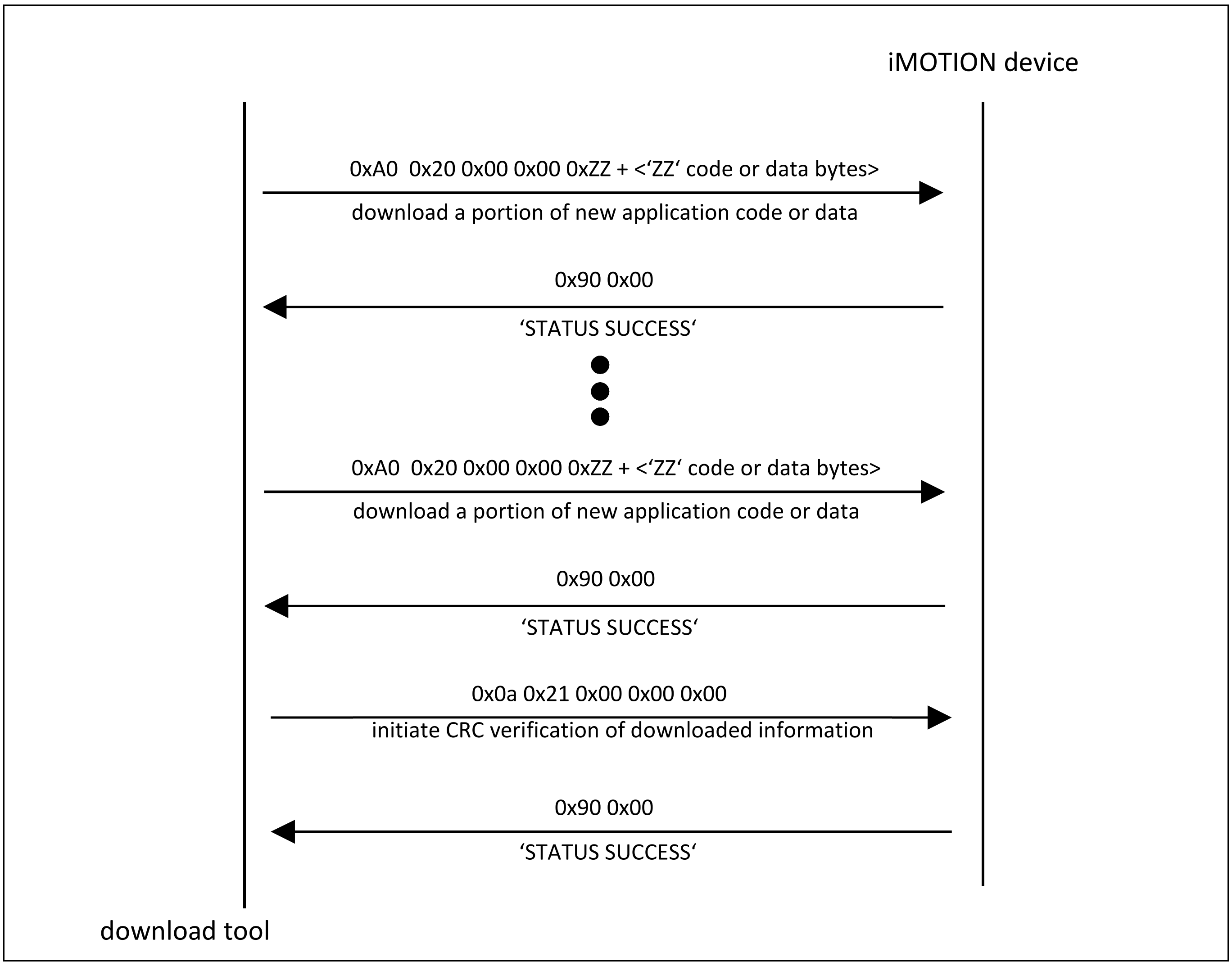

iMOTION device
0xA0 0x20 0x00 0x00 0xZZ + <‘ZZ‘ code or data bytes>
download a portion of new application code or data
0x90 0x00
‘STATUS SUCCESS‘
0xA0 0x20 0x00 0x00 0xZZ + <‘ZZ‘ code or data bytes>
download a portion of new application code or data
0x90 0x00
‘STATUS SUCCESS‘
0x0a 0x21 0x00 0x00 0x00
initiate CRC verification of downloaded information
0x90 0x00
‘STATUS SUCCESS‘
download tool

Figure 12 Protocol flow for download of new application code and data

4 Parameter and script programming

Programming of parameter sets as well as scripts is done without encryption and uses a loader incorporated within the MCE image. Therefore it is mandatory to have the MCE be programmed into the device.

The loader runs in a special mode in which the motor (or PFC) control application is not running.

The communication protocol used for parameter and script programming is the same as for the secure loader protocol as described above.

It should be noted that there are two distinct commands to download data into the RAM and to verify and program to Flash.

4.1 Prerequisites and basics of usage

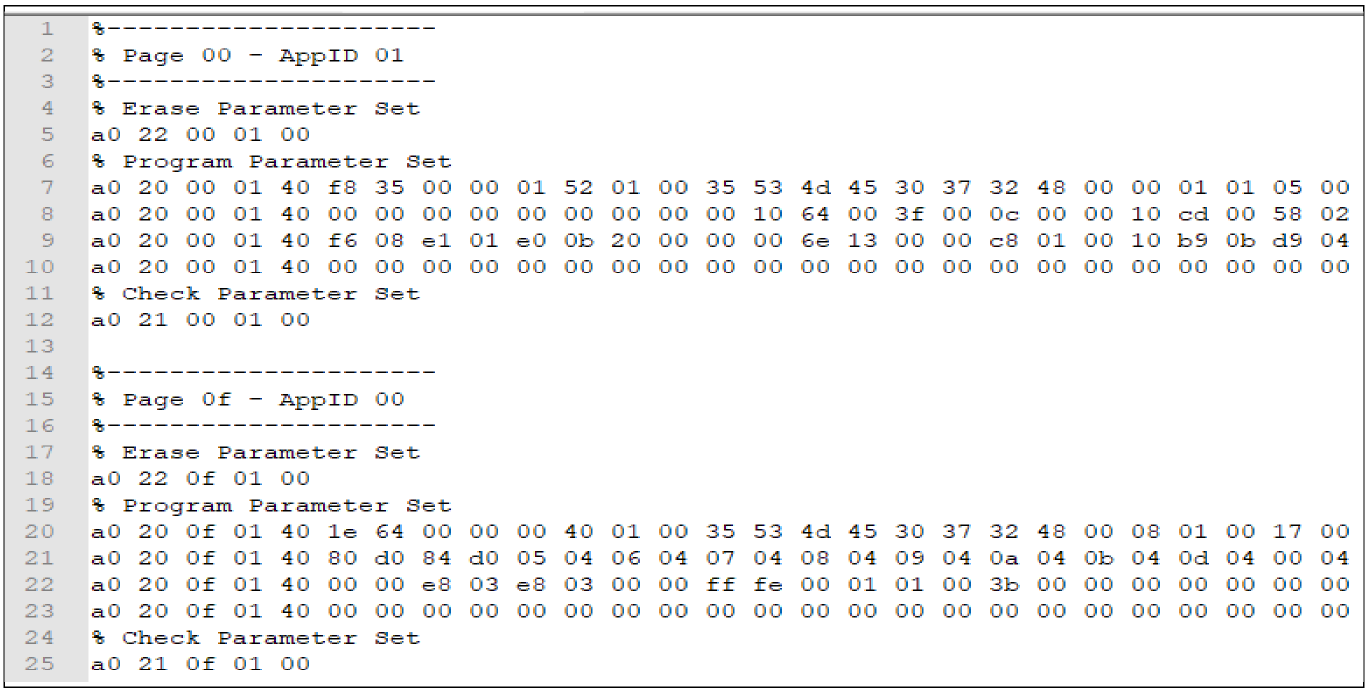
The following prerequisites have to be met before starting the programming cycle.

- file with parameter set or script to be programmed
  e.g. `IMC101T-T038_Parameter.ldf` – parameter set file for device IMC101T-T038
  `or/and 05_Test#1_Fw_Rw_Control.ldf` – Script code file for application
- UART connection to the device
- Download tool with loader protocol support

Attention: *This parameters file always relatives with custom design, the name depends on custom project and their habits.*

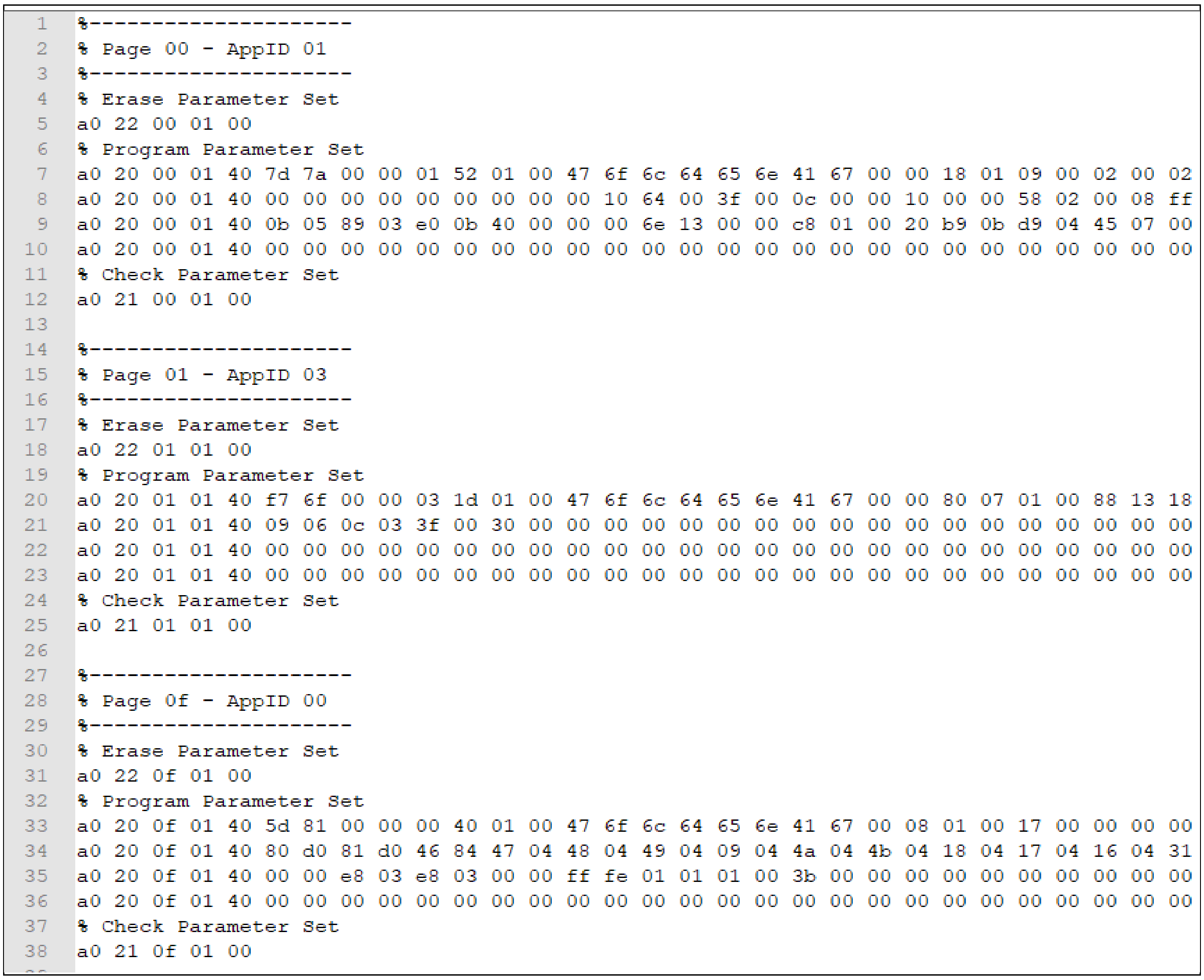
4.2 Parameters set file format description

It should be noted that there are three parameters blocks for system which contains PFC application. Otherwise, there are two blocks need program.



```
%----------------------
% Page 00 - AppID 01
%----------------------
% Erase Parameter Set
a0 22 00 01 00
% Program Parameter Set
a0 20 00 01 40 f8 35 00 00 01 52 01 00 35 53 4d 45 30 37 32 48 00 00 01 01 05 00
a0 20 00 01 40 00 00 00 00 00 00 00 00 00 10 64 00 3f 00 0c 00 00 10 cd 00 58 02
a0 20 00 01 40 f6 08 e1 01 e0 0b 20 00 00 00 6e 13 00 00 c8 01 00 10 b9 0b d9 04
a0 20 00 01 40 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
% Check Parameter Set
a0 21 00 01 00

%----------------------
% Page 0f - AppID 00
%----------------------
% Erase Parameter Set
a0 22 0f 01 00
% Program Parameter Set
a0 20 0f 01 40 1e 64 00 00 00 40 01 00 35 53 4d 45 30 37 32 48 00 08 01 00 17 00
a0 20 0f 01 40 80 d0 84 d0 05 04 06 04 07 04 08 04 09 04 0a 04 0b 04 0d 04 00 04
a0 20 0f 01 40 00 00 e8 03 e8 03 00 00 ff fe 00 01 01 00 3b 00 00 00 00 00 00 00
a0 20 0f 01 40 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
% Check Parameter Set
a0 21 0f 01 00
```

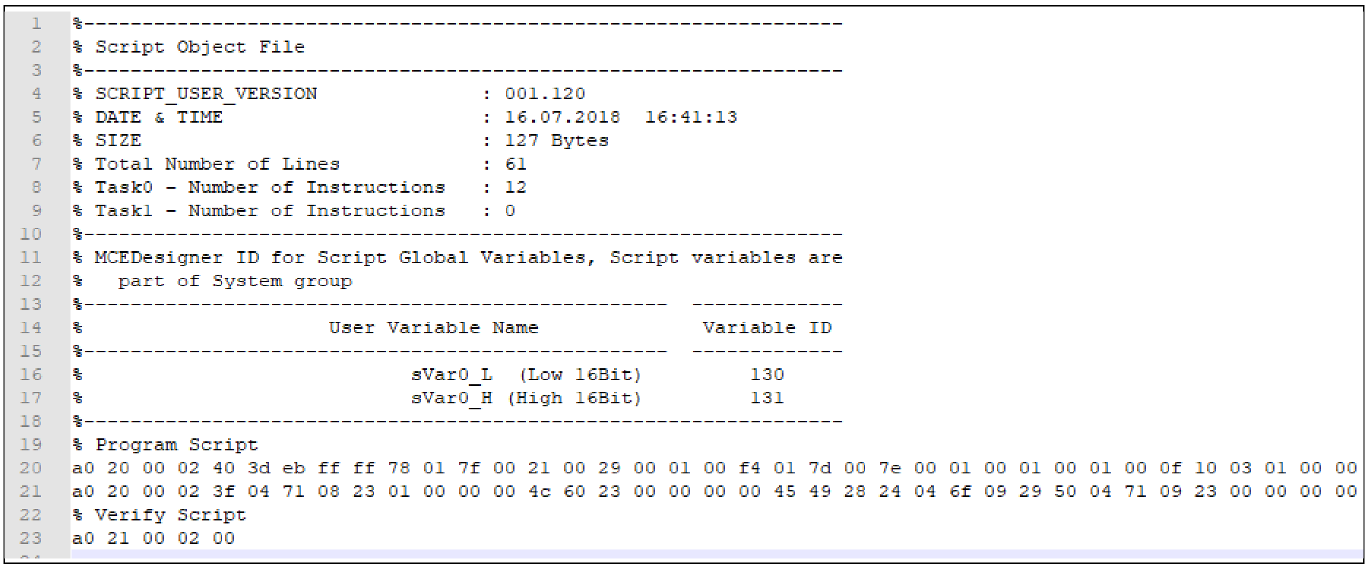
Figure 13 Parameters set file without PFC parameters block.



```
%----------------------
% Page 00 - AppID 01
%----------------------
% Erase Parameter Set
a0 22 00 01 00
% Program Parameter Set
a0 20 00 01 40 7d 7a 00 00 01 52 01 00 47 6f 6c 64 65 6e 41 67 00 00 18 01 09 00 02 00 02
a0 20 00 01 40 00 00 00 00 00 00 00 00 00 10 64 00 3f 00 0c 00 00 10 00 00 58 02 00 08 ff
a0 20 00 01 40 0b 05 89 03 e0 0b 40 00 00 00 6e 13 00 00 c8 01 00 20 b9 0b d9 04 45 07 00
a0 20 00 01 40 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
% Check Parameter Set
a0 21 00 01 00

%----------------------
% Page 01 - AppID 03
%----------------------
% Erase Parameter Set
a0 22 01 01 00
% Program Parameter Set
a0 20 01 01 40 f7 6f 00 00 03 1d 01 00 47 6f 6c 64 65 6e 41 67 00 00 80 07 01 00 88 13 18
a0 20 01 01 40 09 06 0c 03 3f 00 30 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
a0 20 01 01 40 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
a0 20 01 01 40 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
% Check Parameter Set
a0 21 01 01 00

%----------------------
% Page 0f - AppID 00
%----------------------
% Erase Parameter Set
a0 22 0f 01 00
% Program Parameter Set
a0 20 0f 01 40 5d 81 00 00 00 40 01 00 47 6f 6c 64 65 6e 41 67 00 08 01 00 17 00 00 00 00
a0 20 0f 01 40 80 d0 81 d0 46 84 47 04 48 04 49 04 09 04 4a 04 4b 04 18 04 17 04 16 04 31
a0 20 0f 01 40 00 00 e8 03 e8 03 00 00 ff fe 01 01 01 00 3b 00 00 00 00 00 00 00 00 00 00
a0 20 0f 01 40 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
% Check Parameter Set
a0 21 0f 01 00
```

Figure 14 **Parameters set file with PFC parameters block.**

4.3 Script set file format description



```
%------------------------------------------------------------------
% Script Object File
%------------------------------------------------------------------
% SCRIPT_USER_VERSION               : 001.120
% DATE & TIME                       : 16.07.2018  16:41:13
% SIZE                              : 127 Bytes
% Total Number of Lines             : 61
% Task0 - Number of Instructions    : 12
% Task1 - Number of Instructions    : 0
%------------------------------------------------------------------
% MCEDesigner ID for Script Global Variables, Script variables are
%   part of System group
%--------------------------------------------------  -------------
%                    User Variable Name              Variable ID
%--------------------------------------------------  -------------
%                           sVar0_L  (Low 16Bit)          130
%                           sVar0_H (High 16Bit)          131
%------------------------------------------------------------------
% Program Script
a0 20 00 02 40 3d eb ff ff 78 01 7f 00 21 00 29 00 01 00 f4 01 7d 00 7e 00 01 00 01 00 01 00 0f 10 03 01 00 00
a0 20 00 02 3f 04 71 08 23 01 00 00 00 4c 60 23 00 00 00 00 45 49 28 24 04 6f 09 29 50 04 71 09 23 00 00 00 00
% Verify Script
a0 21 00 02 00
```

Figure 15 **Script file for device.**

4.4 Protocol-specific commands for parameter and script

4.4.1 Supported Commands

Table 1 below lists the commands supported by the loader for programming parameter sets and scripts.

Table 13 parameter loader commands

CLA	INS	Name	Description
0x00	0x6C	CONNECT	Connect with auto baudrate detection
0xA0	0x00	CHIP_RESET	Trigger chip reset
0xA0	0x10	GET_STATUS	Provide configuration status of the device
0xA0	0x11	GET_PARAMETER_SET_NAME	Provide meta data of selected parameter page
0xA0	0x12	GET_PARAMETER_SET_VALUES	Provide list of all parameters of the selected page
0xA0	0x14	GET_SYSTEM_CONFIG	Provide the system configuration page
0xA0	0x18	CHANGE_BOOT_MODE	Request to change the boot mode
0xA0	0x20	DOWNLOAD_PARAMETER	Download parameter into RAM
0xA0	0x21	CHECK_PARAMETER	Verify the checksum and program the page into Flash
0xA0	0x22	CLEAR_PARAMETER	Clear Parameter Page

4.4.2 CONNECT

This command connects to the chip with auto baudrate detection. The first byte is used to capture the baudrate; the second byte is an ID byte. The connect command is the same for all three protocols and the response byte is unique for each mode. Hence the host can identify the current mode by the response byte to a connect command

Security

None

Parameters

None

Syntax

Table 14 CONNECT syntax

CLA	INS	P1	P2	L_c	Data field	L_e
0x00	0x6C	-	-	-	-	-

Response

Table 15 CONNECT response

Data Field	SW1	SW2	Status
-	0xCD	-	Success, Config Mode

Return value

None.

4.4.3 CHIP_RESET

This command triggers a chip reset. The response is returned and the chip reset takes place after 100ms.

Security

None

Parameters

None

Syntax

Table 16 CHIP_RESET syntax

CLA	INS	P1	P2	L_c	Data field	L_e
0xA0	0x00	0x00	0x00	0x00	-	-

Response

Table 17 CHIP_RESET response

Data Field	SW1	SW2	Status
-	0x90	0x00	Success

Return value

The command always reports success.

4.4.4 GET_STATUS

The get status command shows the chip ID, software version number as well as parameter page usage.

Security

None.

Parameters

None.

Syntax

Table 18 GET _STATUS syntax

CLA	INS	P1	P2	L_c	Data Field	L_e
0xA0	0x10	0x00	0x00	-	-	0x1F

Response

Table 19 GET _STATUS response

Data field	SW1	SW2	Status
rConfStatus	0x90	0x00	Success
rConfStatus	0x65	0x81	Erase procedure failure
-	0x67	0x00	Wrong L_e

Return value

This command returns *rConfStatus*.

Table 20 rConfStatus structure

Offset	Bytes	Value	Description
0	4	"CONF"	Magic name identifying structure
4	1	0xC0	SBSL version tag
5	1	0x08	Length of following data
6	4	Chip ID	iMOTION™ Chip ID
10	4	vr rb bb	Hardware Version: version, build
14	1	0xC1	Parameter Page Usage
15	1	0x0F	Length of following data
16	1	table0	Table Type of page 0
17	1	table1	Table Type of page 1
18	1	table2	Table Type of page 2
19	1	table3	Table Type of page 3
20	1	table4	Table Type of page 4
21	1	table5	Table Type of page 5
22	1	table6	Table Type of page 6
23	1	table7	Table Type of page 7
24	1	table8	Table Type of page 8
25	1	table9	Table Type of page 9
26	1	table10	Table Type of page 10
27	1	table11	Table Type of page 11
28	1	table12	Table Type of page 12
29	1	table13	Table Type of page 13
30	1	table14	Table Type of page 14

4.4.5 GET_PARAMETER_SET_NAME

The get parameter set name command provides the Meta data of the selected parameter page. This contains the page number, the table type, and the number of parameters and the name of the page.

Security

None.

Parameters

None.

Syntax

Table 21 GET_PARAMETER_SET_NAME syntax

CLA	INS	P1	P2	L_c	Data Field	L_e
0xA0	0x11	Page	0x00	-	-	0x13

Response

This commands returns *rParsStatus*.

Table 22 GET_PARAMETER_SET_NAME response

Data field	SW1	SW2	Status
rParsStatus	0x90	0x00	Success
	0x65	0x80	Checksum error
-	0x67	0x00	Wrong *length*

Return value

This command returns *rParsStatus*.

Table 23 rParsStatus structure

Offset	Bytes	Value	Description
0	4	"PARS"	Magic name identifying the structure
4	1	0xC2	Parameter Set Name tag
6	1	page	Parameter Page
7	1	table	Table Type of Page
8	1	number	Number of Parameter
9	10	name	Name of Page

4.4.6 GET_PARAMETER_SET_VALUE

The get parameter set values command provides the content of the parameter set.

Security

None.

Parameters

None.

Syntax

Table 24 GET_PARAMETER_SET_VALUE syntax

CLA	INS	P1	P2	L_c	Data Field	L_e
0xA0	0x12	Page	0x00	0x00	-	-

Response

This commands returns *rParvStatus*.

Table 25 GET_PARAMETER_SET_VALUE response

Data field	SW1	SW2	Status
	0x6A	0x86	Forbidden page number

Return value

This command returns *rParvStatus*.

Table 26 rParvStatus structure

Offset	Bytes	Value	Description
0	4	"PARV"	Magic name identifying the structure
4	1	0xC3	Parameter Set Name tag
5	2	len	length
for all function blocks:			
	1	page#	
	1	table	

4.4.7 CHANGE_BOOT_MODE

The change boot mode command is a request to change the boot mode. First the response is transmitted, second in case of a successful command, the boot mode is changed and third the device restarts automatically in the selected boot mode.

Security

None.

Parameters

None.

Syntax

Table 27 CHANGE_BOOT_MODE syntax

CLA	INS	P1	P2	L_c	Data Field	L_e
0xA0	0x18	mode	~mode	-	-	0x00

Table 28 Boot modes

Mode	~mode	Description
0x5D	0xA2	Boot Loader Mode (SBSL)
0xCD	0x32	Config Mode (CONF)
0xAD	0x52	Application Mode (MCEDesigner + User COM)
0xAF	0x50	Failsafe Mode (ClassB fault)

5D:

- Program the MCE

CD:

- Reprogram parameters and script, because integrity check might have failed
- if reprogram firmware: use command CHANGE_BOOT_MODE to SBSL mode.

AD:

- Config mode can be entered by sending low pulses with 155us width to pin RXD0 at power-up. This is a 0x00 at 57.600 baud. The system responds with a 0x06 at pin TXD0. Then you can connect normally with 0x00, 0x6C

AF:

- This is “Failsafe Mode” as result of ClassB fault
- User can use Get_Status and Chip Reset

Response

Table 29 CHANGE_BOOT_MODE response

Data field	SW1	SW2	Status
rParsStatus	0x90	0x00	Success, than change to new mode
	0x65	0x81	NVM Programming Error
-	0x69	0x84	Mismatch between mode and ~mode
	0x6A	0x86	Wrong boot mode

Return value

This commands returns *rParvStatus*.

4.4.8 DOWNLOAD_PARAMETER

The download parameters command sends the data packets to the devices where they are stored into a buffer. The actual programming is triggered by command check parameter.

Security

None.

Parameters

None.

Syntax

Table 30 DOWNLOAD_PARAMETER syntax

CLA	INS	P1	P2	L_c	Data Field	L_e
0xA0	0x20	Page	0x00	length	-	data_length

Response

Table 31 DOWNLOAD_PARAMETER response

Data field	SW1	SW2	Status
	0x90	0x00	Success

Return value

None.

4.4.9 CHECK_PARAMETER

The check parameter command calculates the checksum of the data in the buffer and in case of valid data, it programs the selected parameter page.

Security

None.

Parameters

None.

Syntax

Table 32 CHECK_PARAMETER syntax

CLA	INS	P1	P2	L_c	Data Field	L_e
0xA0	0x21	page	0x00	0x00	-	-

Response

Table 33 CHECK _PARAMETER response

Data field	SW1	SW2	Status
	0x90	0x00	Success
	0x65	0x80	Checksum Error in RAM, not programmed
	0x65	0x81	NVM Programming Error
	0x65	0x82	Page is not empty, not programmed

Return value

None.

4.4.10 CLEAR_PARAMETER

The clear parameter command erases the selected parameter page.

Security

None.

Parameters

None.

Syntax

Table 34 CHECK_PARAMETER syntax

CLA	INS	P1	P2	L_c	Data Field	L_e
0xA0	0x22	page	0x00	0x00	-	-

Response

Table 35 CHECK _PARAMETER response

Data field	SW1	SW2	Status
	0x90	0x00	Success
	0x69	0x84	NVM erase failed

Return value

None.

4.5 Protocol-specific commands for safety failure mode

4.5.1 Supported Commands

Table 1 below lists the commands supported by the loader for programming parameter sets and scripts.

Table 36 parameter loader commands

CLA	INS	Name	Description
0xA0	0x00	CHIP_RESET	Trigger chip reset
0xA0	0x10	GET_STATUS	Provide safety failure status of the device
0xA0	0x18	CHANGE_BOOT_MODE	Request to change the boot mode

4.5.2 CHIP_RESET

This command triggers a chip reset. The response is returned and the chip reset takes place after 100ms.

Security

None

Parameters

None

Syntax

Table 37 CHIP_RESET syntax

CLA	INS	P1	P2	L_c	Data field	L_e
0xA0	0x00	0x00	0x00	0x00	-	-

Response

Table 38 CHIP_RESET response

Data Field	SW1	SW2	Status
-	0x90	0x00	Success

Return value

The command always reports success.

4.5.3 GET_STATUS

The get status command shows the chip ID, software version number as well as safety fault reset state.

Security

None.

Parameters

None.

Syntax

Table 39 GET _STATUS syntax

CLA	INS	P1	P2	L_c	Data Field	L_e
0xA0	0x10	0x00	0x00	-	-	0x18

Response

Table 40 GET _STATUS response

Data field	SW1	SW2	Status
rConfStatus	0x90	0x00	Success
rConfStatus	0x67	0x00	Wrong L_e

Return value

This command returns *rConfStatus*.

Table 41 rConfStatus structure

Offset	Bytes	Value	Description
0	4	"FSMD"	Magic name identifying structure
4	1	0xF0	SBSL version tag
5	1	0x0C	Length of following data
6	4	Chip ID	iMOTION™ Chip ID
10	4	vr rb bb	Hardware Version: version, build
14	4	0xC1	Feature ID
18	1	0xF1	Safety version tag
19	1	0x 04	Length of following data
20	4		Failure reset state

4.5.4 CHANGE_BOOT_MODE

The change boot mode command is a request to change the boot mode. First the response is transmitted, second in case of a successful command, the boot mode is changed and third the device restarts automatically in the selected boot mode.

Security

None.

Parameters

None.

Syntax

Table 42 CHANGE_BOOT_MODE syntax

CLA	INS	P1	P2	L_c	Data Field	L_e
0xA0	0x18	mode	~mode	-	-	0x00

Table 43 Boot modes

Mode	~mode	Description
0x5D	0xA2	Boot Loader Mode (SBSL)
0xCD	0x32	Config Mode (CONF)
0xAD	0x52	Application Mode (MCEDesigner + User COM)
0xAF	0x50	Failsafe Mode (ClassB fault)

Response

Table 44 CHANGE_BOOT_MODE response

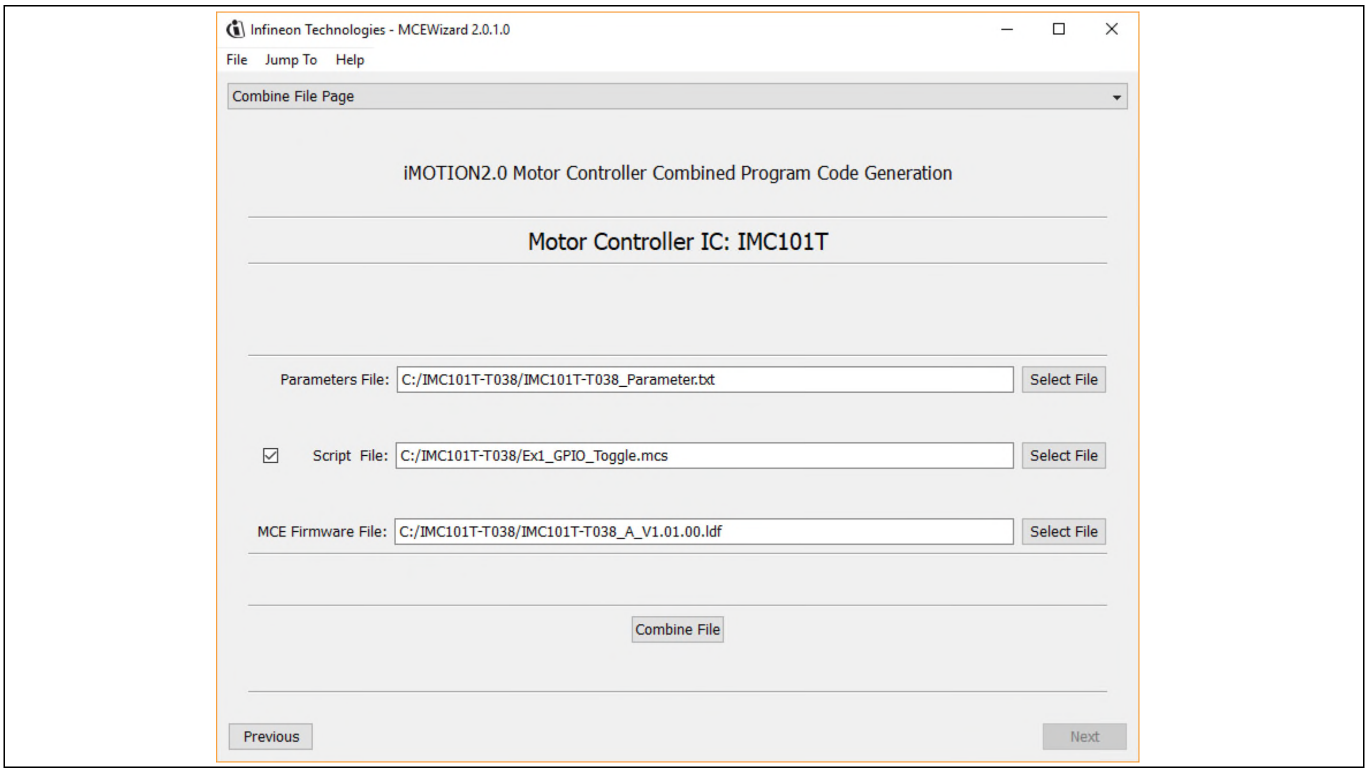
Data field	SW1	SW2	Status
rParsStatus	0x90	0x00	Success, than change to new mode
-	0x69	0x84	Mismatch between mode and ~mode
	0x6A	0x86	Wrong boot mode

Return value

5 Combined file format description

5.1 Generating a combined output file with MCEWizard

After the system tuning, all 3 relevant files (MCE firmware, parameters file and Script file (if used)) can be combined to one for end-of-line programming during the production.


Infineon Technologies - MCEWizard 2.0.1.0
File
Jump To
Help
Combine File Page
iMOTION2.0 Motor Controller Combined Program Code Generation
Motor Controller IC: IMC101T
Select File
Select File
Select File
Combine File
Previous
Next

Parameters File: C:/IMC101T-T038/IMC101T-T038_Parameter.txt

☑ Script File: C:/IMC101T-T038/Ex1_GPIO_Toggle.mcs

MCE Firmware File: C:/IMC101T-T038/IMC101T-T038_A_V1.01.00.ldf

Figure 16 Combined file using MCEWizard for production

Note: Script file is one option, if user uses this function, this file is needed.

5.2 Combined file format

There are two/three sections in generated .ldf file.

- ➢ For MCE section, it will appearance in the beginning with the following flag:

  %:Firmware Data Section Begin

  ……………..(Comments and Data)

  %:Firmware Data Section End

- ➢ For Parameters section, it will appearance in the middle (if script data exist)/end with the following flag:

  %:Parameters Data Section Begin

  ………………. (Comments and Data Block)

  %:Parameters Data Section End

- ➢ For Script section, it will appearance in the end (if script data exist) with the following flag:

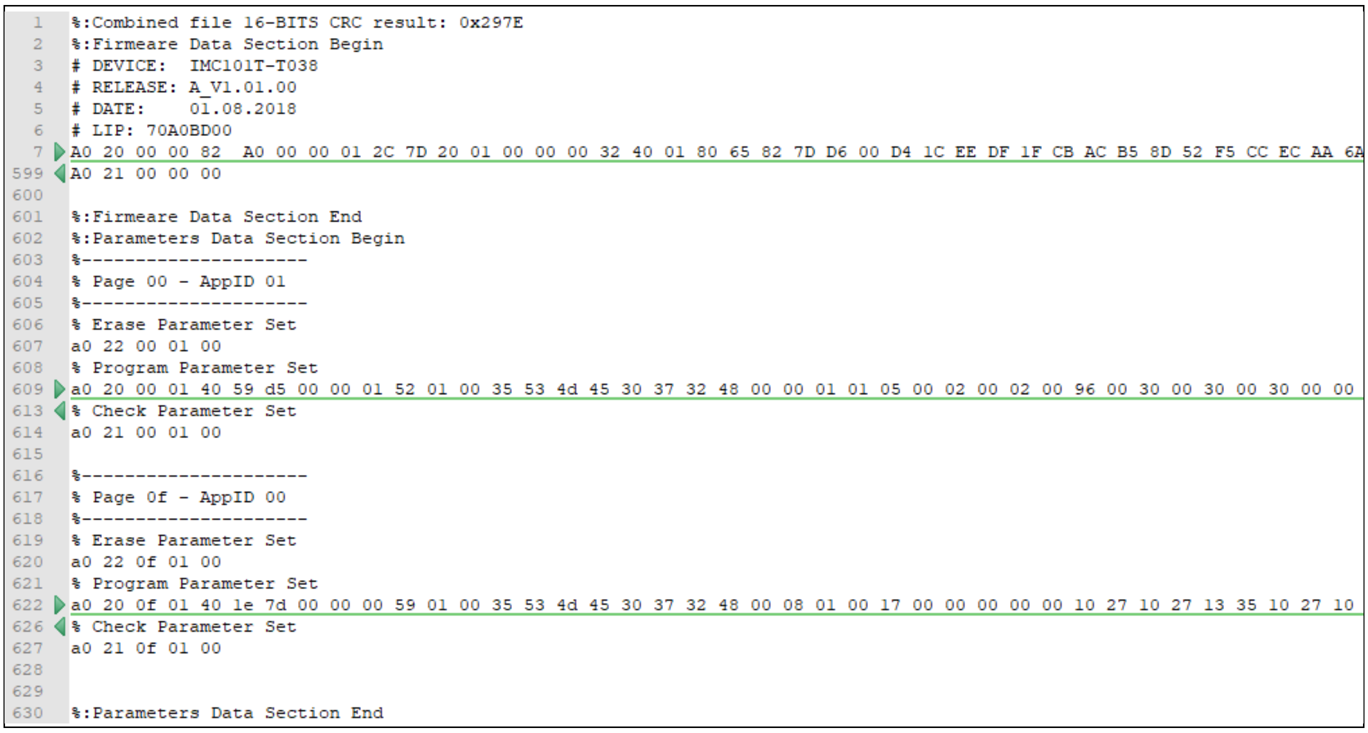
  %:Script Data Section Begin

  ………………. (Comments and Data)

  %: Script Data Section *End*

1) For System without PFC:

If script function is not used by custom, it will combine MCE firmware and parameters two sections into one file. For example:

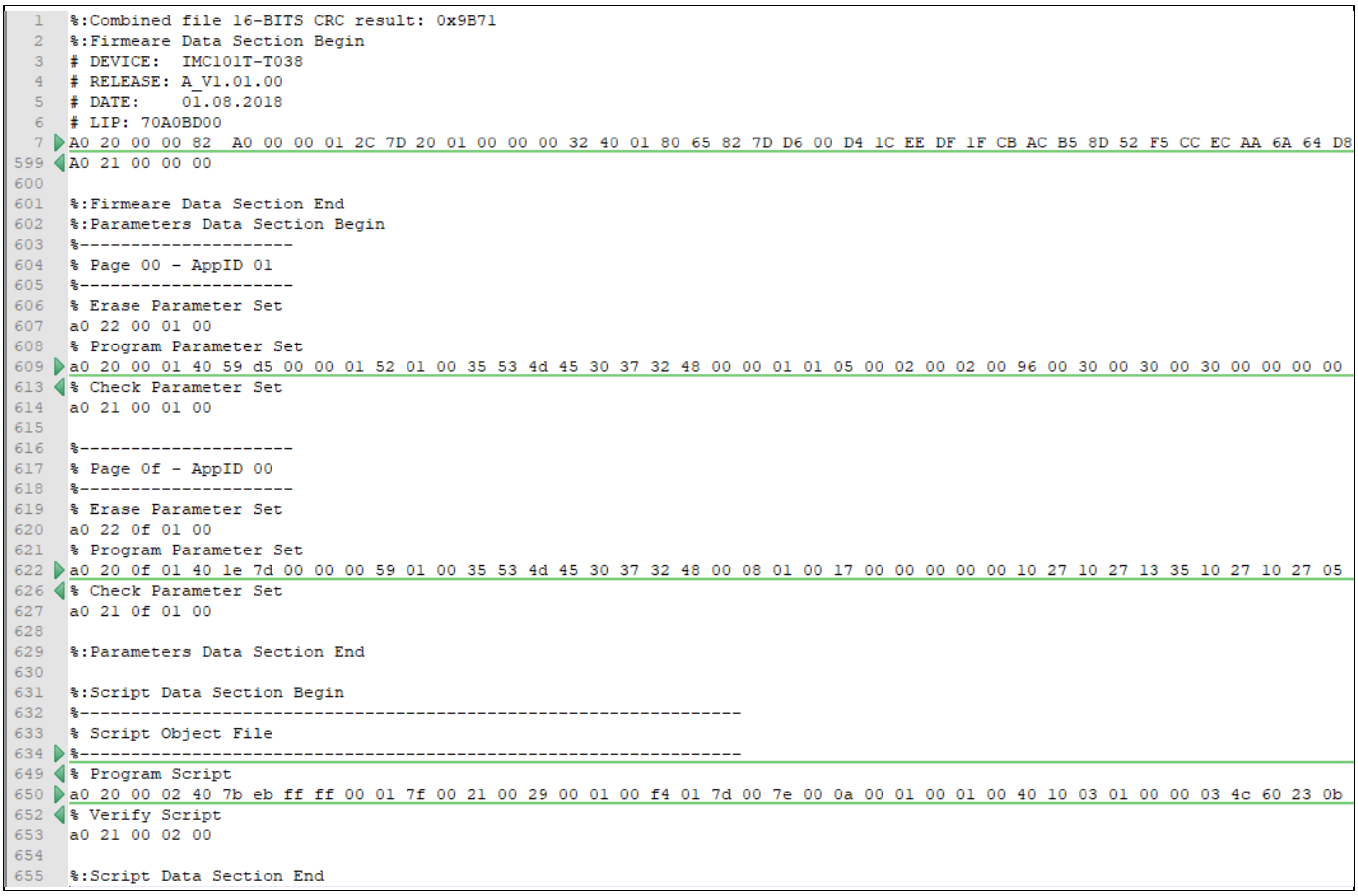


```
%:Combined file 16-BITS CRC result: 0x297E
%:Firmeare Data Section Begin
# DEVICE:  IMC101T-T038
# RELEASE: A_V1.01.00
# DATE:    01.08.2018
# LIP: 70A0BD00
A0 20 00 00 82  A0 00 00 01 2C 7D 20 01 00 00 00 32 40 01 80 65 82 7D D6 00 D4 1C EE DF 1F CB AC B5 8D 52 F5 CC EC AA 6A
A0 21 00 00 00

%:Firmeare Data Section End
%:Parameters Data Section Begin
%---------------------
% Page 00 - AppID 01
%---------------------
% Erase Parameter Set
a0 22 00 01 00
% Program Parameter Set
a0 20 00 01 40 59 d5 00 00 01 52 01 00 35 53 4d 45 30 37 32 48 00 00 01 01 05 00 02 00 02 00 96 00 30 00 30 00 30 00 00
% Check Parameter Set
a0 21 00 01 00

%---------------------
% Page 0f - AppID 00
%---------------------
% Erase Parameter Set
a0 22 0f 01 00
% Program Parameter Set
a0 20 0f 01 40 1e 7d 00 00 00 59 01 00 35 53 4d 45 30 37 32 48 00 08 01 00 17 00 00 00 00 00 10 27 10 27 13 35 10 27 10
% Check Parameter Set
a0 21 0f 01 00


%:Parameters Data Section End
```

Figure 17 Combined file without Script file without PFC parameters.

Combined file format description

If script function is used by custom, it will combine MCE firmware, parameters and script three sections into one file. For example:

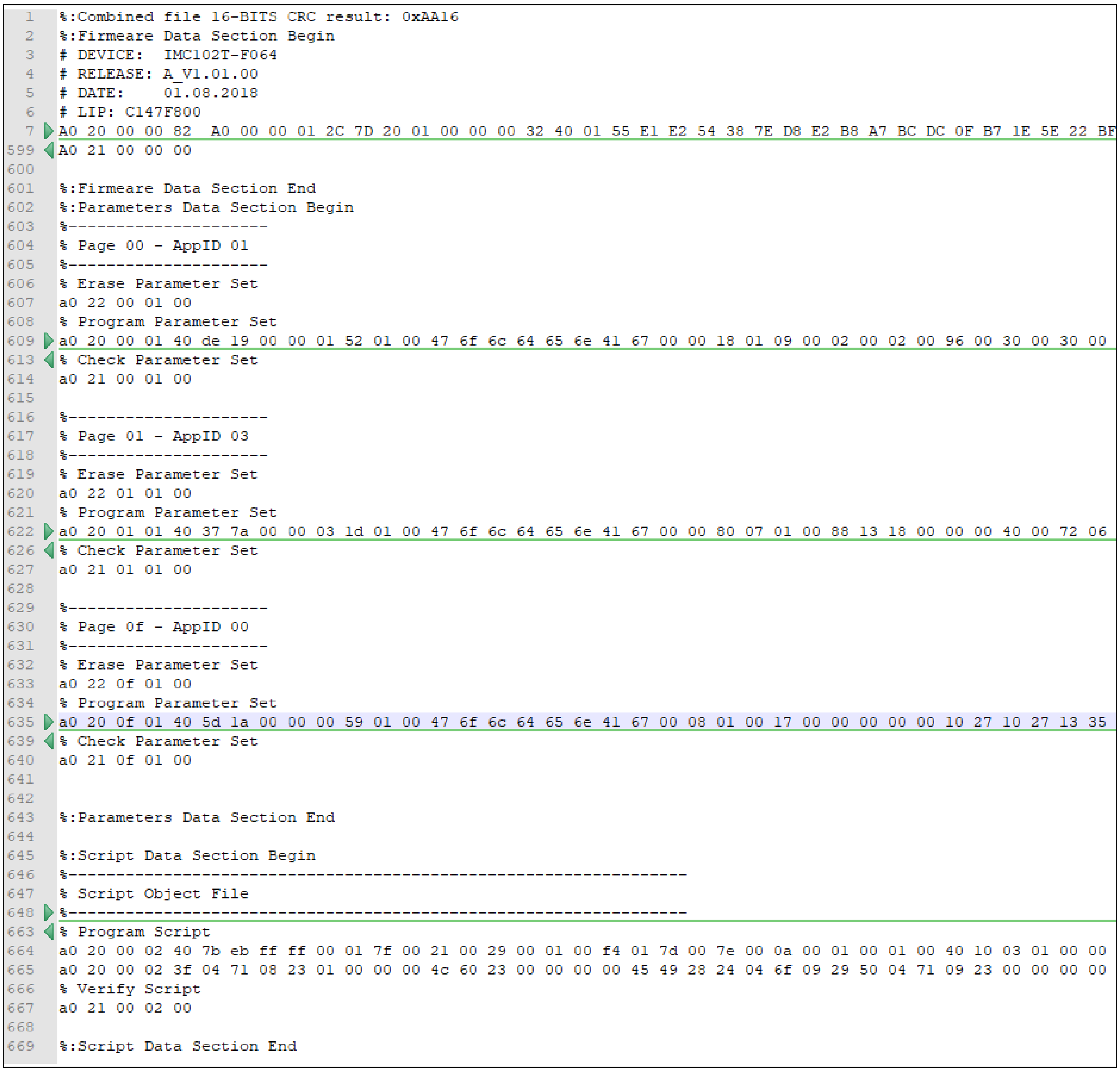


```
%:Combined file 16-BITS CRC result: 0x9B71
%:Firmeare Data Section Begin
# DEVICE:  IMC101T-T038
# RELEASE: A_V1.01.00
# DATE:    01.08.2018
# LIP: 70A0BD00
A0 20 00 00 82  A0 00 00 01 2C 7D 20 01 00 00 00 32 40 01 80 65 82 7D D6 00 D4 1C EE DF 1F CB AC B5 8D 52 F5 CC EC AA 6A 64 D8
A0 21 00 00 00

%:Firmeare Data Section End
%:Parameters Data Section Begin
%---------------------
% Page 00 - AppID 01
%---------------------
% Erase Parameter Set
a0 22 00 01 00
% Program Parameter Set
a0 20 00 01 40 59 d5 00 00 01 52 01 00 35 53 4d 45 30 37 32 48 00 00 01 01 05 00 02 00 02 00 96 00 30 00 30 00 30 00 00 00 00
% Check Parameter Set
a0 21 00 01 00

%---------------------
% Page 0f - AppID 00
%---------------------
% Erase Parameter Set
a0 22 0f 01 00
% Program Parameter Set
a0 20 0f 01 40 1e 7d 00 00 00 59 01 00 35 53 4d 45 30 37 32 48 00 08 01 00 17 00 00 00 00 00 10 27 10 27 13 35 10 27 10 27 05
% Check Parameter Set
a0 21 0f 01 00

%:Parameters Data Section End

%:Script Data Section Begin
%------------------------------------------------------------------
% Script Object File
%------------------------------------------------------------------
% Program Script
a0 20 00 02 40 7b eb ff ff 00 01 7f 00 21 00 29 00 01 00 f4 01 7d 00 7e 00 0a 00 01 00 01 00 40 10 03 01 00 00 03 4c 60 23 0b
% Verify Script
a0 21 00 02 00

%:Script Data Section End
```

Figure 18 Combined file with Script file without PFC parameters.

2) For System with PFC Parameters:

If there is a PFC control used for system, then parameters section will have 3 blocks for programming.



```
%:Combined file 16-BITS CRC result: 0xAA16
%:Firmeare Data Section Begin
# DEVICE:  IMC102T-F064
# RELEASE: A_V1.01.00
# DATE:    01.08.2018
# LIP: C147F800
A0 20 00 00 82  A0 00 00 01 2C 7D 20 01 00 00 00 32 40 01 55 E1 E2 54 38 7E D8 E2 B8 A7 BC DC 0F B7 1E 5E 22 BF
A0 21 00 00 00

%:Firmeare Data Section End
%:Parameters Data Section Begin
%---------------------
% Page 00 - AppID 01
%---------------------
% Erase Parameter Set
a0 22 00 01 00
% Program Parameter Set
a0 20 00 01 40 de 19 00 00 01 52 01 00 47 6f 6c 64 65 6e 41 67 00 00 18 01 09 00 02 00 02 00 96 00 30 00 30 00
% Check Parameter Set
a0 21 00 01 00

%---------------------
% Page 01 - AppID 03
%---------------------
% Erase Parameter Set
a0 22 01 01 00
% Program Parameter Set
a0 20 01 01 40 37 7a 00 00 03 1d 01 00 47 6f 6c 64 65 6e 41 67 00 00 80 07 01 00 88 13 18 00 00 00 40 00 72 06
% Check Parameter Set
a0 21 01 01 00

%---------------------
% Page 0f - AppID 00
%---------------------
% Erase Parameter Set
a0 22 0f 01 00
% Program Parameter Set
a0 20 0f 01 40 5d 1a 00 00 00 59 01 00 47 6f 6c 64 65 6e 41 67 00 08 01 00 17 00 00 00 00 00 10 27 10 27 13 35
% Check Parameter Set
a0 21 0f 01 00


%:Parameters Data Section End

%:Script Data Section Begin
%-------------------------------------------------------------------
% Script Object File
%-------------------------------------------------------------------
% Program Script
a0 20 00 02 40 7b eb ff ff 00 01 7f 00 21 00 29 00 01 00 f4 01 7d 00 7e 00 0a 00 01 00 01 00 40 10 03 01 00 00
a0 20 00 02 3f 04 71 08 23 01 00 00 00 4c 60 23 00 00 00 00 45 49 28 24 04 6f 09 29 50 04 71 09 23 00 00 00 00
% Verify Script
a0 21 00 02 00

%:Script Data Section End
```

Figure 19 Combined file with Script file with PFC parameters.

6 Example for programing and re-programing device of iMOTION™

6.1 How to program FLASH with an empty device

1) Set UART configuration before communication with device.

- The UART is configured as 8 bits for byte size, no parity, and one stop bit. And 115200 bps band rate is recommended.

Note: before device powered on, RXD/TXD pins at idle state should be set to high level.

2) Check mode state when connection to device.

- Send *0x00 0x6C* to device.
- Wait 50ms.
- Read one byte from device.
- If read value = 0x5D then we are in the right mode (SBSL).

3) Read SBSL ID from device

- Send *0xA0 0x10 0x00 0x00 0x27* to device.
- Wait 1000ms.
- Read 0x28 bytes from device. Get the SBSL_ID at the right bytes address. (See Table 7)
- Wait 50ms.
- Read 0x02 bytes from device. This is the state of reading status process.

SBSL_ID *example values*:

IMC101T: 0x02 0x27 0x0F 0x1F 0xCC 0xDF 0x57 0xC3 0x33 0xD3 0x1A 0xBD 0x78 0xF9 0x60 0xB0

IMC102T: 0x02 0x89 0x42 0x6D 0xAA 0x14 0x29 0x3A 0xB3 0x18 0x28 0xD8 0x34 0x1A 0xD4 0xEF

4) Send MCE firmware to device:

- Ignore lines with comments or empty lines.
- Read data line and check FLASH_LOAD_DATA format.
- Send each line and get feedback from device.
- After program, send the verify command.
- Send *0xA0 0x21 0x00 0x00 0x00* to device.

- Then wait a certain time to read feedback and finish verify process.

5) Check mode state after programing MCE firmware.

- Send 0x00 0x6C to device.
- Wait 50ms.
- Read one byte from device.
- If read value = 0xCD (Config Mode), then we get the right mode from programming parameters/script file.

6) Send parameters and script file(if included) to device:

- Ignore lines with comments or empty lines.
- Read data line and check CLEAR_PARAMETER format.
- Send command for erasing each page by erase demand.
- Read data line and check DOWNLOAD_PARAMETER format.
- Send each line and get feedback from device.

7) Change boot mode to 0xAD (Application Mode) using CHANGE_BOOT_MODE command.

6.2 How to program FLASH with a programed device

1) Set UART configuration before communication with device.

- The UART is configured as 8 bits for byte size, no parity, and one stop bit. And 115200 bps band rate is demanded.

Note: before device is power on, RXD/TXD pins at idle state should be set to high level.

2) Check mode state when connection to device.

- Send *0x00 0x6C* to device.
- Wait 50ms.
- Read one byte from device.
- If read value is one of the following values: 0xFF (same as 0xAD mode), 0x5D, 0xCD, 0xAD, 0xAF, then we are properly connected to the iMOTION™ device, else we need to repower on device and send the CONNECT command to recheck device mode.

3) Change mode to SBSL (0x05D) or to Config Mode (0xCD) in order to program MCE firmware or parameters/script file.

- If programming MCE firmware, boot mode should in SBSL mode (0x5D). If programing parameters/script file, boot mode should be in Config mode (0xCD).
- If device is in the Application mode (0xAD), the download tool needs send an MCE command to change mode to SBSL mode or Config mode. (See 1.3 MCE command set) or needs to initiate the catch procedure at start-up period to change to Config mode (See 1.4 catch at start-up method).

Note: if user use catch at start-up method to enter the Config mode, the UART band rate should change to 57600 before send 0x00 to device.

- If device in 0xCD/0xAF mode, the download tool needs to send the CHANGE_BOOT_MODE command to SBSL mode in order to program MCE firmware. (See 4.4.7 Change boot mode).

4) After device mode is set to SBSL (0x5D), the standard procedure for programming the empty device (see 6.1 How to program FLASH with an empty device) can be followed.

7 Revision History

Major changes since the last revision

Document version	Date of release	Description of changes
1.0	2018-09-26	First Release
1.1	2018-11-28	Plus small modifications

Edition 2018-11-28

Published by

Infineon Technologies AG

81726 Munich, Germany

© 2018 Infineon Technologies AG.
All Rights Reserved.

Do you have a question about this document?

Email: erratum@infineon.com

Document reference
AN2018-33