iMOTION™ 2.0 Device Programming

Programming of the Motion Control Engine, parameters and scripts

About this document

Scope and purpose

This document describes the programming and updating the firmware of second generation iMOTION™ devices (iMOTION™ 2.0). 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 decryption 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.

Note: *If iMOTION products like the IMC300 series contain a microcontroller based on an Arm® Cortex® core they use the standardized SWD interface as specified by Arm®. The debug controller and register structure is documented in the reference manual of the resp. product. References are given at the end of the document.*

Intended audience

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

Since the programming of the iMOTION™ 2.0 devices only requires a UART as a physical interface, this documentation can also be used by other users.

References

Product documentation and software can be downloaded from http://www.infineon.com/iMOTION. Please also refer to the references given at the end of this document.

Table of Contents

Table of Contents

1 Overview

This document describes programming and updating iMOTION™ 2.0 products that are based on the Motion Control Engine (MCE revision 1.0 and above).

iMOTION™ 2.0 products are shipped without software, so the following steps have to be performed at the end of the production line:

1. Programming of the Motion Control Engine (MCE)
2. Programming of parameter sets for system, motor and power factor correction (PFC is optional)
3. Programming of customer scripts (optional)

All communication and programming is done via a UART interface using the protocol definitions given in the following chapters.

Communication flow is based on sending and receiving bytes, which are given in hexadecimal notation in this application note.

In order to select the right protocol it is important to identify the current state of the device as described in 1.1 Device modes description. Depending on the device state the following programming steps can be performed.

- Programming of the MCE via the Secure Bootstrap Loader (SBSL) using the encrypted firmware images provided by Infineon. This secure loader is part of the device and cannot be changed or erased. The SBSL supports the use of high baudrates to allow short programming times.
- Reprogramming of the MCE, e.g. when upgrading to a newer firmware release, is done by re-entering the secure loader mode. Reprogramming the MCE via secure loader will erase the complete internal Flash memory, including all parameter sets and scripts.
- Parameter and script programming is done with unencrypted data. The parameter and script loaders are a part of the MCE firmware thus the MCE firmware has to be programmed first.

This overview chapter gives an introduction into the programming of the iMOTION devices and also provides the basic hardware interface settings.

The following chapters provide the definitions of the communication protocols that are to be used in the different product modes.

Finally examples are provided for initial (end-of-line) programming as well as for updating parameter sets or scripts.

1.1 Device modes description

iMOTION devices can be in several modes which also define the baud rate and protocol supported. The device will report its state automatically in reply to a connection request on the UART and the state can also be requested by using the command GET_STATUS. The reported value is given below and in Figure 1.
Transitions between the modes can be initiated via commands or be done automatically as given in Figure 1.

1. **Secure Bootstrap Loader Mode 0x5D**

All devices which are based on the Motion Control Engine (MCE) are shipped from the factory with an embedded Secure Bootstrap Loader (SBSL) which is active at device start-up. 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. After MCE firmware is programmed, the device will change to Config mode automatically.

2. **Config Mode 0xCD**

This mode is entered after (re-)programming the MCE firmware or if requested with the resp. command. Config mode is used for programming of parameters and/or scripts.

After a valid parameter set has been programmed the device can transition to Application Mode.

3. **Application Mode 0xAD**

Application Mode is the normal operational state of the device, running the motor and the optional PFC. In case a valid parameter set has been programmed this mode is entered after power up.

The CHANGE_BOOT_MODE command can be used to enter the Config Mode for programming new parameters or scripts or to SBSL mode if the MCE itself should be updated.

4. **Fail Safe Mode 0xAF**

The device enters the “Failsafe Mode” as result of a Class B 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.

Changing from one mode to another has the following effects.

- Changing into SBSL mode will automatically erase the complete flash of the device, i.e. the MCE itself as well as the parameter sets and scripts regardless of the previous state.
- Changing into Config mode will not erase the existing parameter sets and scripts. Instead parameter sets can be erased and written selectively. The Script is overwritten, i.e. it is automatically erased when a new script programming is done.
- A transition into fault state does not change any flash contents.

Overview

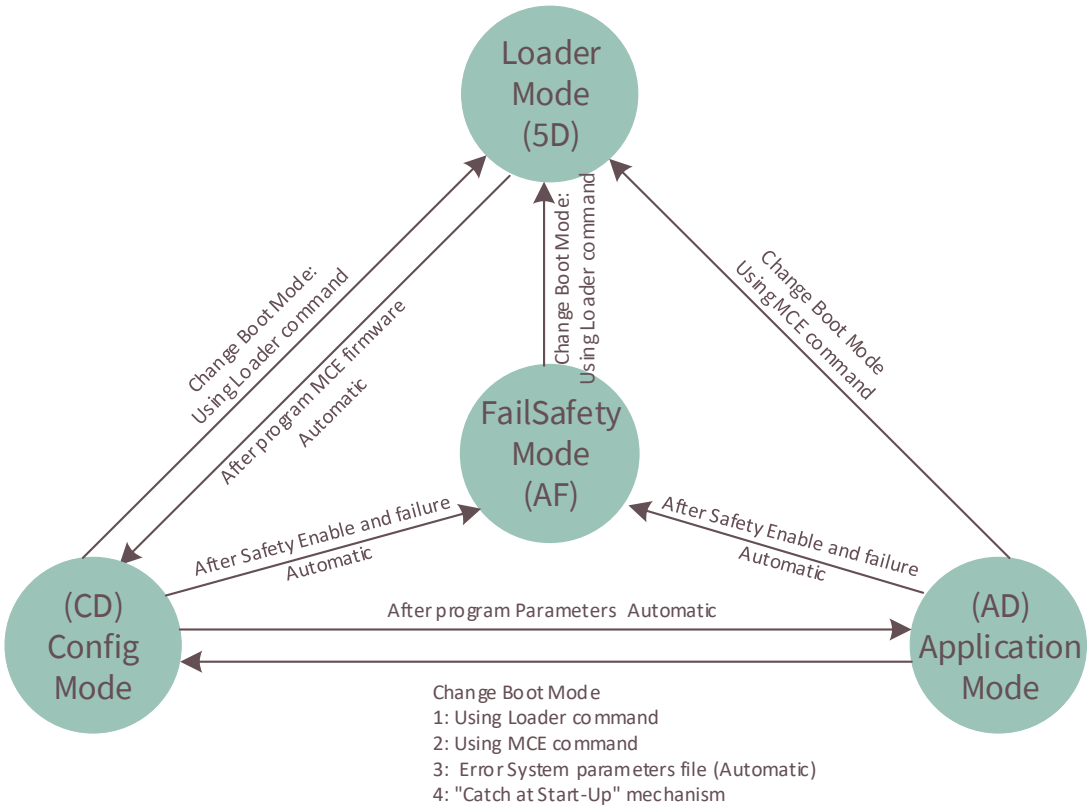

Figure 1 Flowchart for various device modes

1.2 UART configuration

For devices that provide more than one UART the port which is marked for ‘device programming’ in the respective data sheet must be used 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. For more information on the UART interfaces see AN2020-07 Interfacing with iMOTION™ products.

iMOTION devices use the following configuration of the programming port:

- 8 data bits
- 1 stop bit
- No parity
- LSB first

Automatic baudrate detection

iMOTION devices implement an automatic baudrate detection in the range between 300 and 115,200 Baud.

The baudrate detection is activated by the download tool sending the CONNECT command ‘0x00 0x6C’. On correct detection of the baudrate, the iMOTION™ device answers with the current mode as given in 1.1 Device modes description. In case of a new device this would be the SBSL mode ‘0x5D’. If a reply other than device state is returned, the device failed to recognize the baudrate used by the download tool.

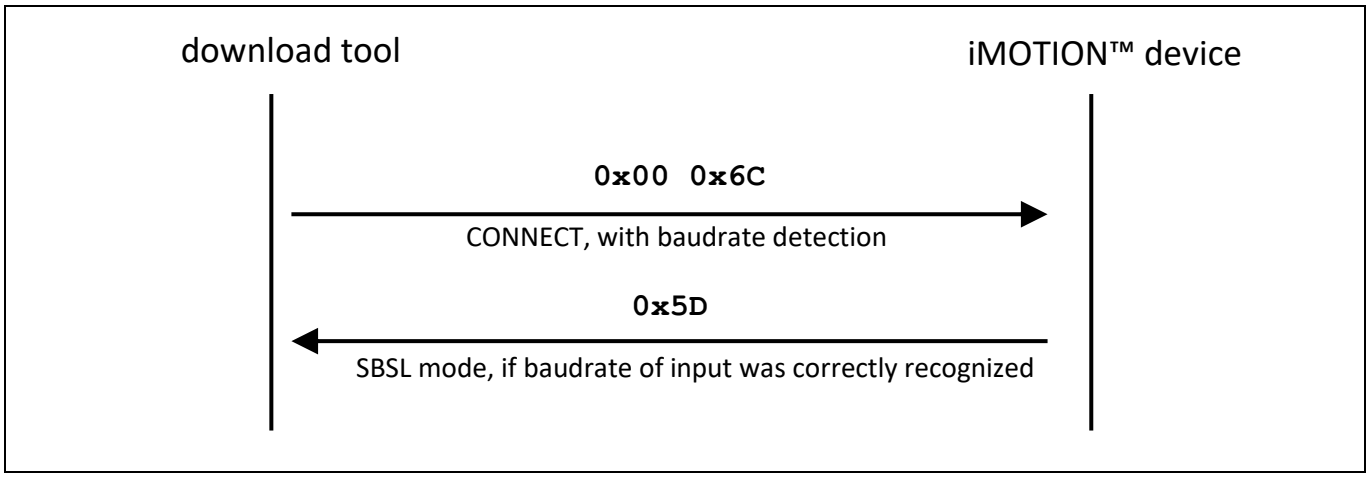

Figure 2 Protocol flow with baudrate detection

Enhanced Baudrate mode

In order to allow a fast end-of-line programming in manufacturing, the secure boot loader (SBSL) additionally supports an enhanced baudrate mode. This mode also implements auto baudrate detection and allows the use of baudrates up to 5.994 MHz. The flow to enter the enhanced baudrate mode is as described below and shown in Figure 3 Protocol flow in Enhanced Baudrate Mode.

The Enhanced Baudrate mode is requested by the download tool by sending the ‘0x00 0x93’ CONNECT command.

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

Overview

The download tool uses the 'PDIV' value for the calculation of the 'STEP' value defining the requested 'final baudrate' and sends the STEP value to the device. (STEP is sent with MSB first)

The iMOTION™ device acknowledges with '0xF0' in the initial baudrate and switches to the target baudrate.

After changing to the target baudrate the download tool also sends the acknowledge '0xF0'.

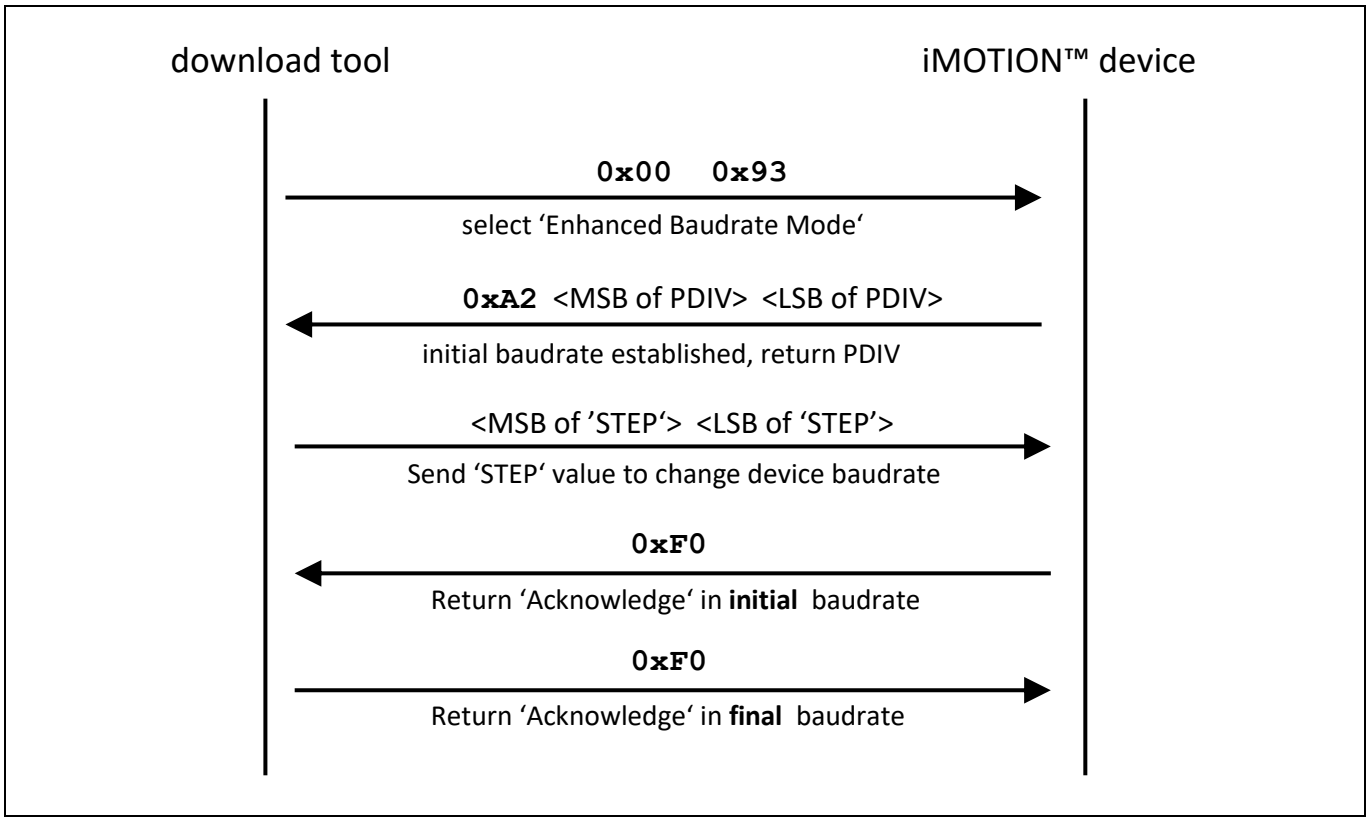

Figure 3 Protocol flow in Enhanced Baudrate Mode

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

Changing the baudrate from initial 115,200 to 1 Mbaud.

- Initial Baudrate = 115,200 Baud
- Target Baudrate = 1,000,000 Baud
- PDIV = 0x00 0x34 = 0x0034 = 52

STEP = 1024 x (1,000,000 / 115,200) / (52+1) = 167.7 ~ 168 = 0x00A8 = 0x00 0xA8.

Note: Using the enhanced baudrates is only recommended for dedicated tools like gang programmers. Typical USB to UART ('VCOM') interfaces for PCs might have problems to achieve these high baudrates reliably.

1.3 Loader file format description

Infineon provides the Motion Control Engine in loader files with the file extension *.ldf on the website.

These files contain the respective encrypted firmware images in the form of an ASCII file with the individual data bytes given in hexadecimal representation.

This loader file is provided 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 basic usage.

Example: Infineon-MCE_IMC101T-T038-Firmware-v01_03-EN.zip

- IMC101T-T038_A_V1.03.03.ldf - MCE rev. 1.3 for the device IMC101T-T038
- IMC101T-T038_Parameter.txt - default parameter file (text format)
- IMC101T_V1.03.03_Default.map - default map file
- IMC101T_V1.03.03.irc - default *.irc file
- IMC101T-T038_V1.03.03_ReleaseNotes.pdf - MCE version release notes
- iMOTION_License_Agreement.pdf - iMOTION usage license agreement
- Readme.txt - brief description on usage

Attention: ***The encrypted loader files are type specific. Trying to program the wrong file to a device will decrease the Flash Download Trial Counter (FDTC). If the FDTC reaches zero the device will be rendered unusable. (see 2.5)***

For programming only the loader file is required. An abbreviated example of a loader file is given below.

Table 1 Examples of a MCE loader file (abbreviated)

`# DEVICE: IMC101T-T038`
`# RELEASE: A_V1.03.03`
`# DATE: 24.07.2020`
`# LIP: 70A0BD00`
	file header,
	lines starting with '#' are comments

`A0 20 00 00 82 A0 00 00 01 5C 67 20 … A1 9E`
`A0 20 00 00 82 98 B1 34 BC 47 C9 BD … 5D E0`
…
`A0 20 00 00 82 FB D5 66 EA 4C BA C3 … 4D 64`
	MCE image,
	each line starts with the FLASH_LOAD_DATA command including the length byte (0x82 – 130 bytes)
	followed by the payload (MCE)

`A0 20 00 00 12 D3 69 49 41 F6 10 76 … BC D2`	last line is typically shorter, here 0x12 – 18 bytes
`A0 21 00 00 00`	FLASH_LOAD_CHECK_SIGNATURE command required to finalize the flash programming

1.4 Parameter set loader file format

Parameter files to be used for download have the file extension *.ldf and use the same format as the MCE loader file with additional header information. (Parameter files with a human readable text presentation have the extension *.txt.)

Since parameters are programmed using the loader integrated in the MCE the commands are as given in 3 Config Mode - Parameter and script programming.

There is one parameter block for each motor and one for PFC parameters, so typically there are multiple parameter blocks to be programmed.

Table 2 Examples of a parameter loader file (abbreviated)

`%:Parameters Data Section Begin`
`%---------------------`
`% Page 00 - AppID 01`
`%---------------------`
	parameter set header,
	lines starting with '%' are comments

`% Erase Parameter Set` `a0 22 00 01 00`	CLEAR_PARAMETER command for erasing the parameter set

`% Program Parameter Set`
`a0 20 00 01 40 99 9b 00 00 01 … 33 03`
`a0 20 00 01 40 00 00 00 00 33 … 3f 00`
	parameter data image,
	each line starts with the DOWNLOAD_PARAMETER command including the length byte (0x40 – 64 bytes)
	followed by the payload

`% Check Parameter Set` `a0 21 00 01 00`	CHECK_PARAMETER command required for checksum calculation

1.5 Script loader file format

Scripts are compiled into a byte code which is loaded to the device and executed by the integrated script engine. The files to be used for download have the file extension *.ldf and use the same format as the parameter loader files with differing header information. (Script source files use the extension *.mcs.)

Like parameters scripts are programmed using the loader integrated in the MCE the commands are as given in 3 Config Mode - Parameter and script programming.

Table 3 Examples of a script loader file (abbreviated)

%---------- % Script Compiler Version : V1.0.14 %---------- % Script Object File %---------- % SCRIPT_USER_VERSION : 001.000 % SIZE : 453 Bytes % Total Number of Lines : 107	script code header, lines starting with '%' are comments summary of version, size and total number of lines
% Task0 - Number of Instructions : 20 % Task1 - Number of Instructions : 13 %----------	number of instructions per task, can be used for partitioning of the script task execution (see [3])
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 %----------	list of global variables and their ID, global variables can be monitored by an external tool in parallel to all other MCE parameters
% Program Script a0 20 00 02 40 72 aa ff ff 00 … 88 13 a0 20 00 02 40 00 00 29 29 47 … 3e 4a …	script byte code image, each line starts with the DOWNLOAD_PARAMETER command including the length byte (0x40 – 64 bytes) followed by the payload
% Verify Script a0 21 00 02 00	verify script command for checksum calculation

1.6 Combined loader file

For ease of use the three data sections as described above (MCE firmware, parameters and script) can be combined into a single loader file (*.ldf).

The three sections in the loader file are separated with markers for the respective sections.

Table 4 Examples of a combined loader file (abbreviated)

%:Combined file 16-BITS CRC result: 0xA97F **%:Firmware Data Section Begin** # DEVICE: IMC101T-T038 … A0 20 00 00 12 00 80 AE 9A 6B 55 … B5 24 A0 21 00 00 00 **%:Firmware Data Section End**	file header with description and checksum begin of MCE firmware section end of MCE section
%:Parameters Data Section Begin %--------------------- % Page 00 - AppID 01 %--------------------- … % Check Parameter Set a0 21 0f 01 00 **%:Parameters Data Section End**	begin of parameters section (note that the parameters section can contain multiple parameters sets) end of parameters section
%:Script Data Section Begin %--------------------------------- % Script Compiler Version : V1.0.14 %--------------------------------- … % Verify Script a0 21 00 02 00 **%:Script Data Section End**	begin of script section end of script section

2 Secure boot loader (SBSL) – MCE programming

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.

2.1 Secure loader command set

Secure loader commands are used to program the firmware, parameter and script code into the target device and read status. The loader protocol is using ISO7816 T=0 protocol as shown below:

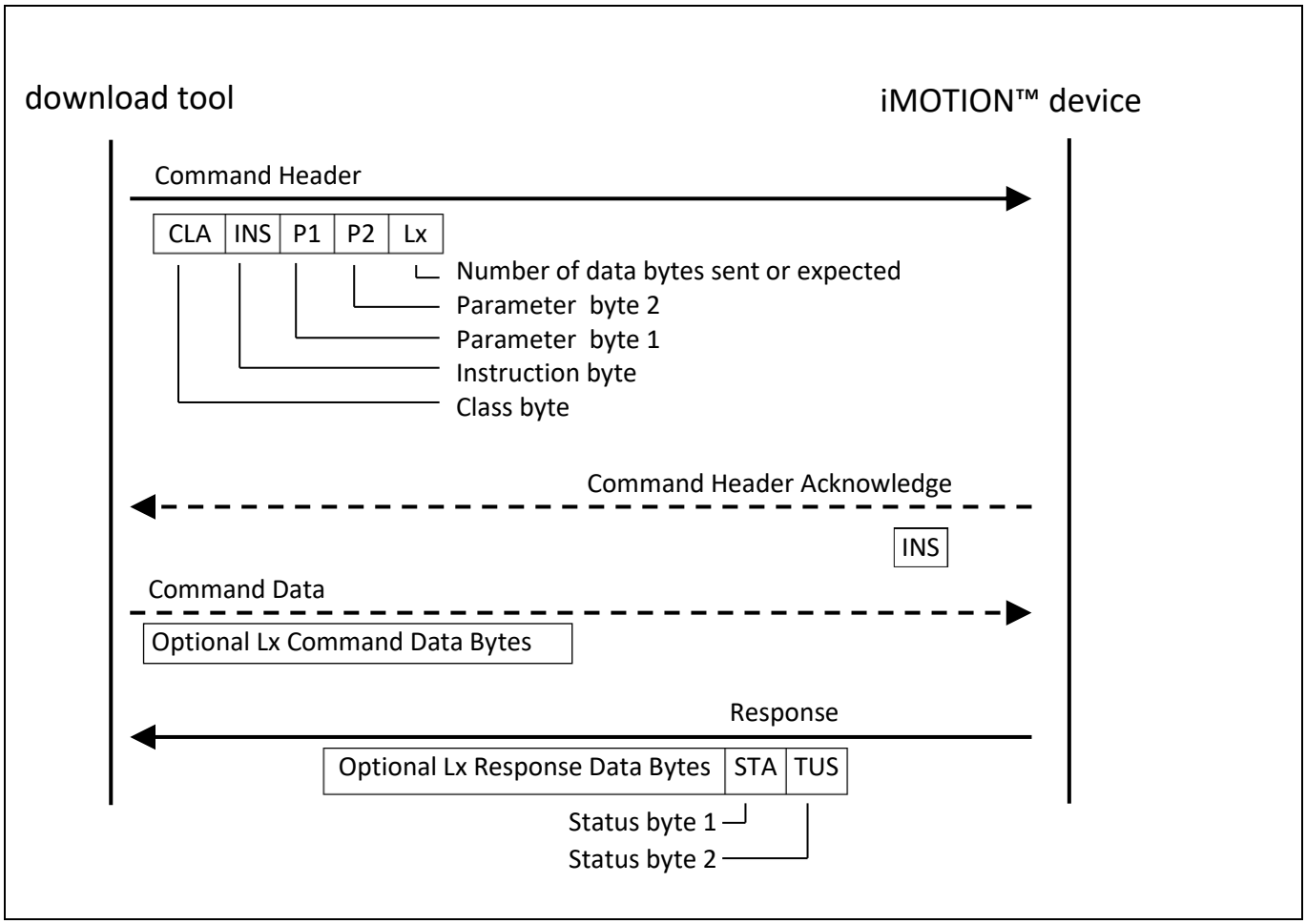

Figure 4 Data flowchart for the secure loader protocol

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 0

FLASH_CHIP_RESET), the L_c is 0x00, so there is no ‘header acknowledge’ from the device.

If the secure loader requires more time, it sends one or more 'Waiting Time Extension Requests' as illustrated by **Figure 5** for the secure loader command ' FLASH_LOAD_CHECK_SIGNATURE ' ('A0 21 00 00 00').

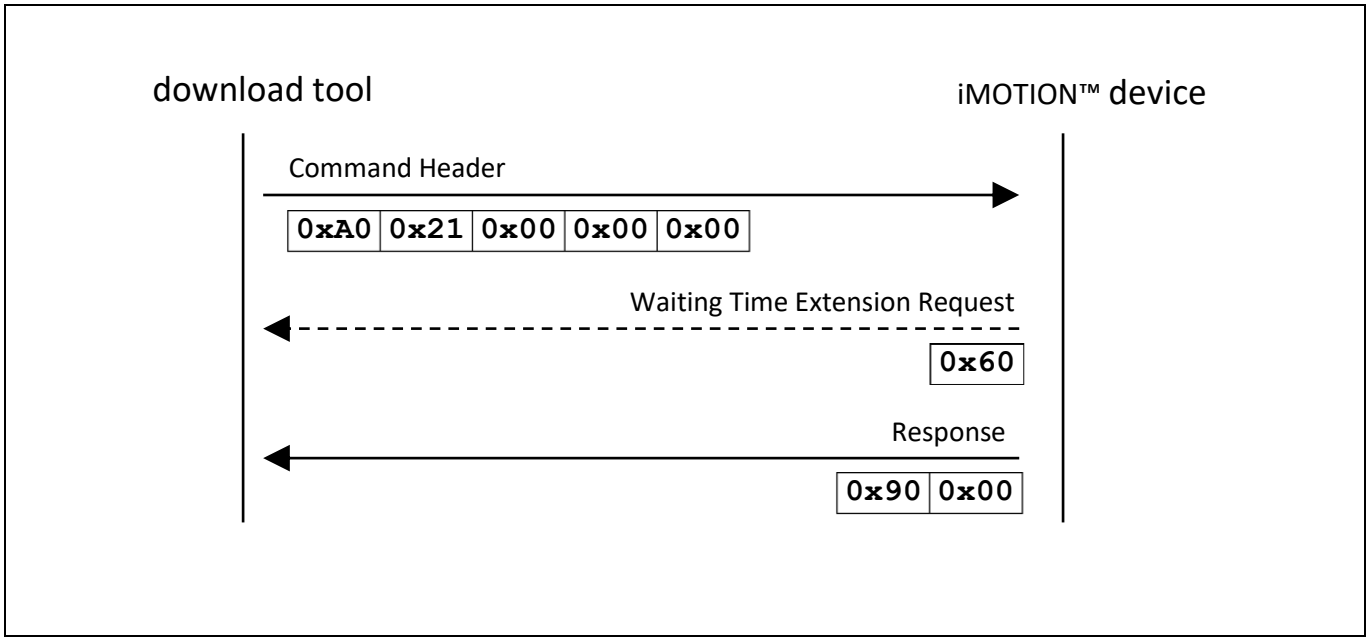

Figure 5 Data flowchart for Waiting Time Extension Request

About secure loader commands

A secure 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.

2.2 Supported Commands

The table below lists the commands supported by the secure loader for programming the Motion Control software image.

Table 5 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

2.3 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.

The table below lists the general command status response values. Any additional command-specific responses are listed in the respective command description.

Table 6 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)	

2.4 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 7 FLASH_CHIP_RESET syntax

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

Response

Table 8 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.

2.5 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.

Attention: ***The download tool should check the status of the Flash Download Trial Counter (FDTC, Byte 20). This counter initially has a value of 0x10 = 16 and is decreased on unsuccessful programming attempts, e.g. due to wrong keys or file used. As soon as the value reaches zero the device is no longer usable.***

Security

None.

Parameters

None.

Syntax

Table 9 FLASH_GET_SBSL_STATUS syntax

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

Response

Table 10 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 11 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
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.

Offset	Bytes	Value	Description
			If the download ended successfully (verified by a checksum calculation, see section 2.7 FLASH_LOAD_CHECK_SIGNATURE), FDTC is reset to its initial start value of 0x10. 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

2.6 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 12 FLASH_LOAD_DATA syntax

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

Response

Table 13 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

Data Field	SW1	SW2	Status
			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 chapter 2.5 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.

2.7 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 14 FLASH_LOAD_DATA syntax

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

Response

Table 15 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 Config Mode - Parameter and script programming

Parameter sets and scripts are not encrypted and uses a loader incorporated in the Motion Control Engine (MCE). Therefore it is required that the MCE has been programmed into the device first.

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 commands to download data (either parameter sets or script) must be followed by the respective verify command.

The file formats used by the Infineon tools for parameter sets as well as scripts are described at the beginning of the application note. Gang programmer can use different formats.

3.1 Supported Commands

The table below lists the commands supported by the loader for programming parameter sets and scripts.

Table 16 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 or script into RAM
0xA0	0x21	CHECK_PARAMETER	Verify the checksum and program the page into Flash used for parameters and scripts
0xA0	0x22	CLEAR_PARAMETER	Clear Parameter Page

3.2 CONNECT

This command connects to the chip with auto baudrate detection. 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 17 CONNECT syntax

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

Response

Table 18 CONNECT response

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

Return value

None.

3.3 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 19 CHIP_RESET syntax

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

Response

Table 20 CHIP_RESET response

Data Field	SW1	SW2	Status
-	0x90	0x00	Success

Return value

The command always reports success.

3.4 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 21 GET _STATUS syntax

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

Response

Table 22 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 23 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

Config Mode - Parameter and script programming

Offset	Bytes	Value	Description
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

3.5 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 24 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 25 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 26 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

3.6 GET_PARAMETER_SET_VALUE

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

Security

None.

Parameters

None.

Syntax

Table 27 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 28 GET_PARAMETER_SET_VALUE response

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

Return value

This command returns *rParvStatus*.

Table 29 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	

3.7 CHANGE_BOOT_MODE

The change boot mode command is used to change into one of the other device modes. 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 30 CHANGE_BOOT_MODE syntax

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

Table 31 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 (Class B fault)

0x5D:

- SBSL mode for programming the MCE

0xCD:

- Config mode for (re)programming of parameters and script

0xAD:

- Application mode for running the device

0xAF:

- Failsafe Mode as a result of Class B safety fault

Response

Table 32 CHANGE_BOOT_MODE response

Data field	SW1	SW2	Status
rParsStatus	0x90	0x00	Success, than change to new mode
	0x65	0x81	NVM Programming Error

Data field	SW1	SW2	Status
-	0x69	0x84	Mismatch between mode and ~mode
	0x6A	0x86	Wrong boot mode

Return value

This commands returns *rParvStatus*.

3.8 DOWNLOAD_PARAMETER (also used for script)

The download parameters command sends the data packets to the devices where they are stored into a buffer. After the download it is mandatory to call the check parameter command in order to calculate the checksum and finalize the flash writing.

Security

None.

Parameters

None.

Syntax

Table 33 DOWNLOAD_PARAMETER syntax (parameter sets)

CLA	INS	P1	P2	L_c	Data Field	L_e
0xA0	0x20	Page	0x01	length	-	data_length

Table 34 DOWNLOAD_PARAMETER syntax (script)

CLA	INS	P1	P2	L_c	Data Field	L_e
0xA0	0x20	0x00	0x02	length	-	data_length

Response

Table 35 DOWNLOAD_PARAMETER response

Data field	SW1	SW2	Status
	0x90	0x00	Success

Return value

None.

3.9 CHECK_PARAMETER (also used for script)

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 36 CHECK_PARAMETER syntax (parameter sets)

CLA	INS	P1	P2	L_c	Data Field	L_e
0xA0	0x21	page	0x01	0x00	-	-

Table 37 CHECK_PARAMETER syntax (script)

CLA	INS	P1	P2	L_c	Data Field	L_e
0xA0	0x21	0x00	0x02	0x00	-	-

Response

Table 38 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.

3.10 CLEAR_PARAMETER

The clear parameter command erases the selected parameter page.

Security

None.

Parameters

None.

Syntax

Table 39 CHECK_PARAMETER syntax

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

Response

Table 40 CHECK _PARAMETER response

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

Return value

None.

4 Safety failure mode

4.1 Supported Commands

Table 5 table below lists the commands supported by the loader for programming parameter sets and scripts.

Table 41 List of supported 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

4.2 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 42 CHIP_RESET syntax

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

Response

Table 43 CHIP_RESET response

Data Field	SW1	SW2	Status
-	0x90	0x00	Success

Return value

The command always reports success.

4.3 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 44 GET _STATUS syntax

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

Response

Table 45 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 46 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

4.4 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 47 CHANGE_BOOT_MODE syntax

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

Table 48 Boot modes

Mode	~mode	Description
0x5D	0xA2	Boot Loader Mode (SBSL)
0xCD	0x32	Config Mode (CONF)
0xAD	0x52	Application Mode (MCE running)
0xAF	0x50	Failsafe Mode (Class B fault)

Response

Table 49 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

None

5 Application Mode - MCE command set

MCE commands are used to communicate with the device when it is in Application Mode with MCE and at least one valid parameter set programmed.

The MCE application mode is primarily targeting in-application communication and is therefore different from the other communication protocols. For programming purposes only the CHANGE_STATUS command is relevant. For updates of parameter and/or scripts a change to Config mode is needed and for an update of the MCE itself a change to SBSL mode.

About MCE commands

An MCE command uses a special format.

- Connection baud rate is fixed at 115200 bps
- The connection check data frame should be used to establish communication

5.1 Supported Commands

Table 5 table below lists the commands supported by the application mode of the MCE. Only commands relevant for programming purposes are listed here.

Table 50 MCE commands

Name	Description
CHECK_COMMUNICATION	Check communication and device state
CHANGE_BOOT_MODE	Change boot mode to SBSL or Config mode

5.2 CHECK_COMMUNICATION

This command is used to check the status of the device.

Security

None.

Parameters

None.

Syntax

Table 51 CHECK_COMMUNICATION syntax

Byte 1	Byte 2	Byte 3	Byte 4			
0x7E	0x13	0x7E	0x13	-	-	-

Response

Table 52 CHECK_COMMUNICATION response

Byte 1	Byte 2	Byte 3	Byte 4	
0x7E	0x17	0x73	0x17	
0x7E	0x1B	0x73	0x1B	

Return value

None

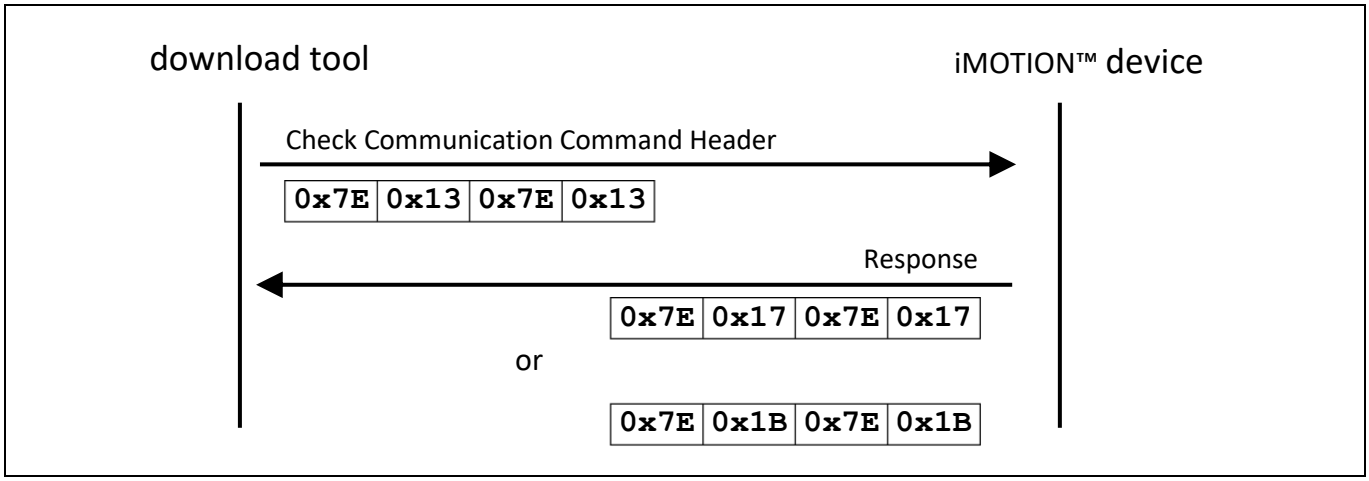

Figure 6 MCE communication protocol check

5.3 CHANGE_BOOT_MODE

The change boot mode command is a request to change to another device state. 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 53 CHANGE_BOOT_MODE syntax

Byte sequence	Description
0x7E 0x02 0x80 0x31 0x51 0x81 0x10 0xFA 0xF8 0x7E 0x87	change to SBSL mode
0x7E 0x02 0x80 0x38 0x51 0x82 0x10 0x32 0xCD 0x7E 0x9C	change to Config mode

Response

Table 54 CHANGE_BOOT_MODE response

Application Mode - MCE command set

Response	Status
0xFE	Success, than change to SBSL mode
0x7E 0x01 0x7E 0x01	Success, than change to Config mode

Return value

None

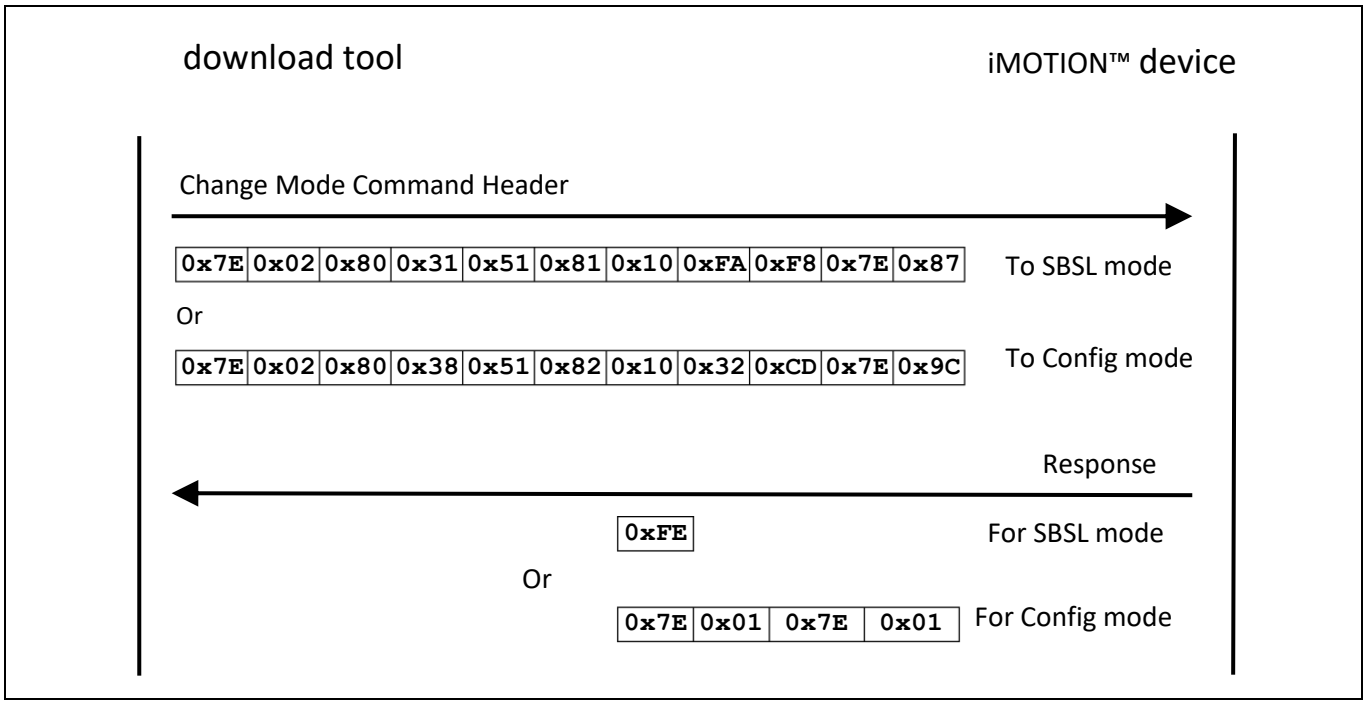

Figure 7 MCE command to change boot mode

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

When MCE firmware or parameters are programmed, 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 programmed 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.

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

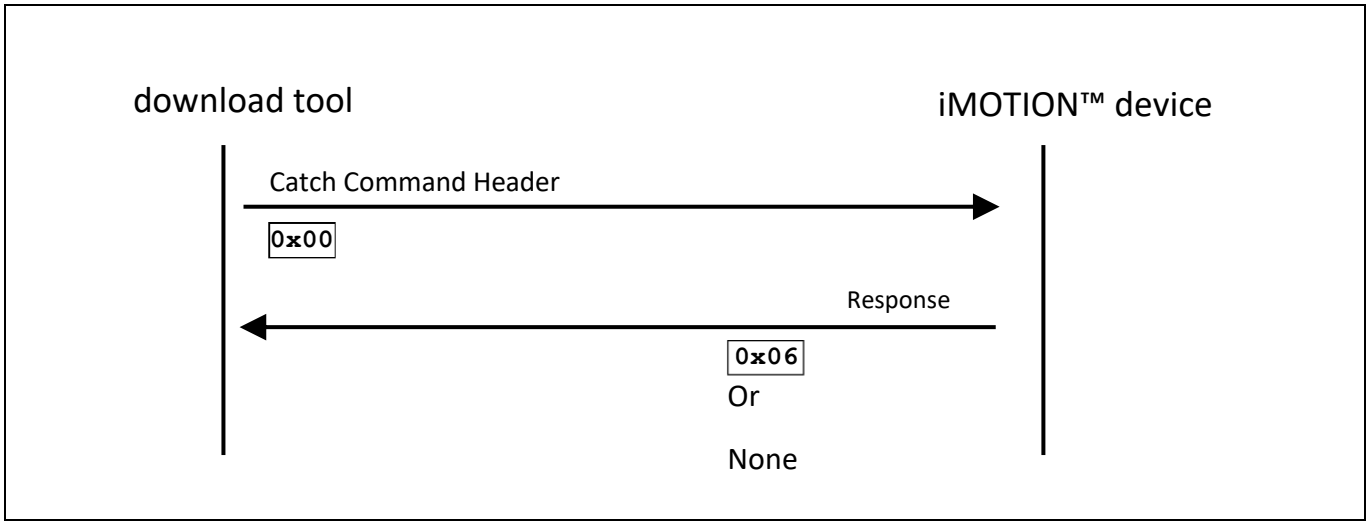

Figure 8 Data flowchart for the catch at start up protocol

6 Examples

6.1 Reading SBSL status of a device

The SBSL Status that includes the SBSL ID and further information can be read from the device as follows.

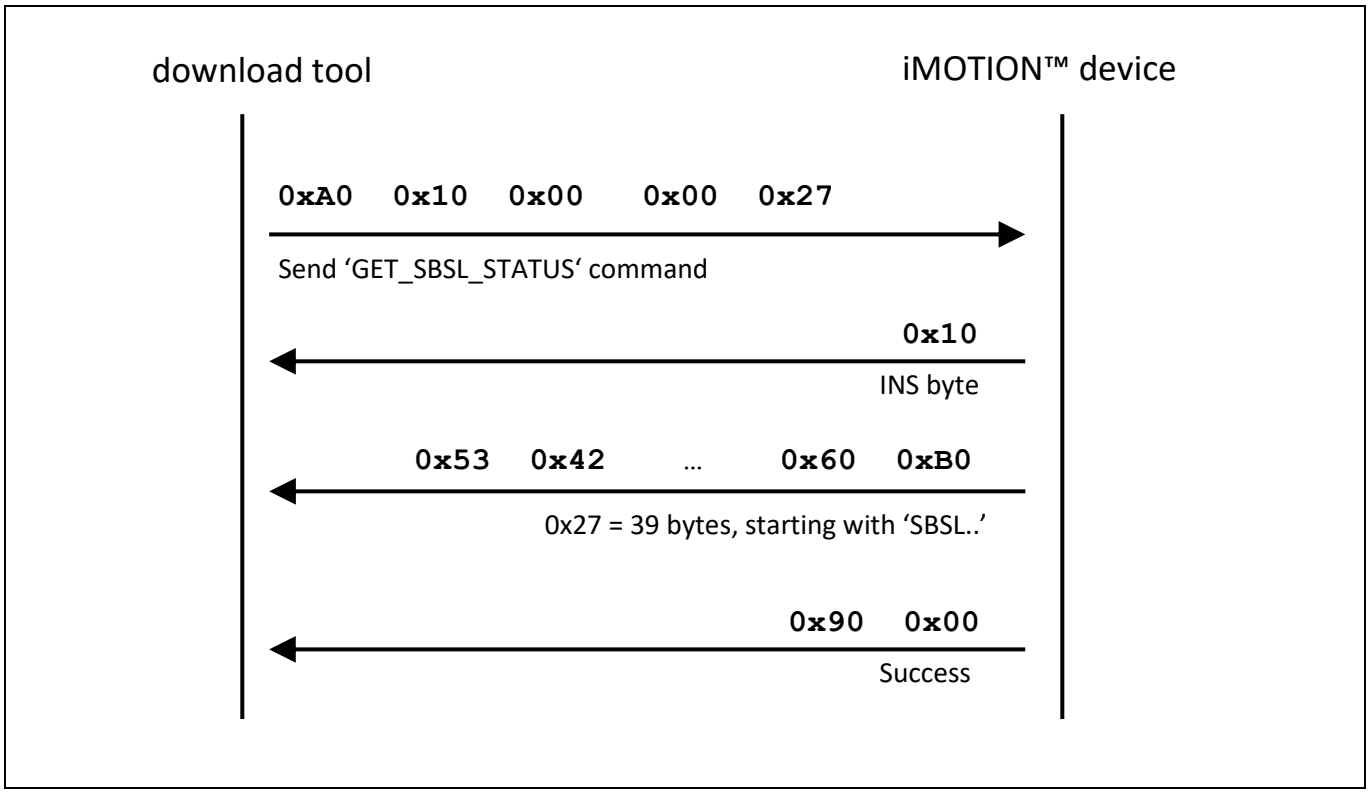

Figure 9 Protocol flow for SBSL Id evaluation

For the layout of the complete reply, please refer to **Table 11**.

The reply contains the Flash Download Trial counter (FDTC) as byte 20.

The SBSL-ID returned at the end of the SBSL status is stated in the respective device datasheets.

Note: In a productive programming environment it is recommended to use this command before the actual programming step in order to verify the correct device type and check the FDTC in order to avoid wrong programming.
For a previously programmed device this use of this command is mandatory before starting programming.

6.2 Programming at the end of manufacturing line

The following gives the recommended flow to follow in order to program an iMOTION device in manufacturing.

The recommendation given here is a balanced approach in terms of programming speed vs. robustness. It is in the responsibility of the manufacturer to implement and possibly further optimize this.

It is strongly recommended to use a professional programming tool that can reliably achieve high baudrates. Use of the development tool iMOTION Link is not recommended for this purpose.

Most of the commands provide a response or a ‘waiting time extension request’ (0x60) as documented in the resp. chapters. These checks are not given for each of the steps below.

The following steps are recommended:

1. Power up the device and negotiate the enhanced baudrate using the procedure described in chapter 1.2 UART configuration (0x00 0x93). The achievable baudrate depends on the programmer as well as the programming environment and should be validated carefully.
2. Use the GET_SBSL_STATUS command to verify that the device is in SBSL state, that the device type is correct (SBSL ID) and that the FDTC is at the initial value. (see 6.1 Reading SBSL status of a device)
3. The SBSL ID’s stored on-chip are used as the base for the decryption of the MCE image, and 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.
   An aborted download will decrease the FDTC.
4. In the next step download the MCE firmware to the device one line at a time. The load command is specified above (2.6 FLASH_LOAD_DATA) and also included in the loader file made available (1.3 Loader file format description).
5. Programming is finalized by sending the FLASH_LOAD_CHECK_SIGNATURE command. This is also given in the loader file (*.ldf). Since this command calculates a checksum over the complete flash programmed it may need more time to respond and therefore send a 'Waiting Time Extension Request' (see Figure 5). After signature check was successful the device sends ‘Success’ (0x90 0x00) and performs a reset transitioning into Config mode.
6. The download tool should reconnect to the device using standard baudrate mode (0x00 06C) and check for the device being in Config mode (reply 0xCD).
7. Now the parameter sets can be downloaded using the commands given in chapter 3 Config Mode - Parameter and script programming. If a (combined) loader file (*.ldf) is used these commands are included in the file. Note that for parameters and scripts they are loaded into RAM first and require the CHECK_PARAMETER command to permanently store them into flash.
8. Send a CHECK_PARAMETER command to calculate the checksum and program into flash.
9. Programming a script is the final step and uses the same commands as used for parameter sets.
10. Send a CHECK_PARAMETER command to calculate the checksum and program into flash.

The device is now ready for use.

6.3 Performing an update of the MCE

Following the transition to the SBSL mode, updating the Motion Control Engine is done in the same way as the initial programming.

The only difference is that in the case of an update the use of the **GET_SBSL_STATUS** command (Step 2. above) is **mandatory**. In case of a previously programmed device this command will initiate the erase of the flash and prepare the actual programming.

6.4 Performing a parameter or script update

If an update for a parameter set or a script is required this is done as follows:

- The device should be brought into Config mode. From a running MCE application this is done with the respective CHANGE_BOOT_MODE command as given under 5 Application Mode - MCE command set.
- In Config mode parameter sets can be selectively erased and written using the commands CLEAR_PARAMETER, DOWNLOAD_PARAMETER, etc. (3 Config Mode - Parameter and script programming)
- After each block of downloaded parameter sets the CHECK_PARAMETER command needs to be sent in order to finalize the programming cycle.
- Updating the script is done using the respective variant of the DOWNLOAD_PARAMETER command.
- After the script is downloaded the CHECK_PARAMETER (script) needs to be requested.

Note: The commands DOWNLOAD_PARAMETER and CHECK_PARAMETER have a different syntax in terms of the parameters P1 and P2. (see 3.8 3.9)

If the loader file (*.ldf) has been generated with the iMOTION tools, it already contains the correct sequence of the respective commands (erase – program – check) for both parameter sets and scripts.

References

[1] iMOTION™ web site, www.infineon.com/iMOTION

[2] AN2020-07 Interfacing with iMOTION™ products, Application Note

[3] iMOTION™ Motion Control Engine, Software Reference Manual

[4] iMOTION™ IMC300 motor controller with additional MCU, www.infineon.com/IMC300

[5] Arm® documentation on SWD, https://developer.arm.com

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	Small modifications
1.2	2020-07-24	Minor wording corrections, fixed missing figure link, added reference to new AN2020-07, updated closing page with latest verbiage
1.3	2021-02-02	Major update

Trademarks
All referenced product or service names and trademarks are the property of their respective owners.

Edition 2021-02-02

Published by

Infineon Technologies AG

81726 Munich, Germany

Do you have a question about this document?

Email: erratum@infineon.com

Document reference
AN2018-33

IMPORTANT NOTICE

The information contained in this application note is given as a hint for the implementation of the product only and shall in no event be regarded as a description or warranty of a certain functionality, condition or quality of the product. Before implementation of the product, the recipient of this application note must verify any function and other technical information given herein in the real application. Infineon Technologies hereby disclaims any and all warranties and liabilities of any kind (including without limitation warranties of non-infringement of intellectual property rights of any third party) with respect to any and all information given in this application note.

The data contained in this document is exclusively intended for technically trained staff. It is the responsibility of customer's technical departments to evaluate the suitability of the product for the intended application and the completeness of the product information given in this document with respect to such application.

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

WARNINGS

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

Except as otherwise explicitly approved by Infineon Technologies in a written document signed by authorized representatives of Infineon Technologies, Infineon Technologies' products may not be used in any applications where a failure of the product or any consequences of the use thereof can reasonably be expected to result in personal injury.