WICED HCI Firmware Download

ModusToolbox

About this document

Scope and purpose

The firmware upgrade feature provided in ModusToolbox® allows an external device to install a newer firmware version on devices equipped with Cypress WICED® Bluetooth chips. This document describes the process used to download new firmware images containing ModusToolbox sample applications and firmware patches.

Intended audience

This document is intended for application developers using a ModusToolbox Bluetooth Software Development Kit (SDK) to create and test designs based on WICED Bluetooth devices.

Table of contents

1 Introduction

The WICED firmware download process updates the embedded program, patch, and configuration for a device. The process described uses the Host Controller Interface (HCI) UART to perform the download. The MCU host control, in addition to HCI Tx, Rx, CTS, and RTS signals, must also control the device RESET.

A few files are mentioned in this document. The .btp file contains many parameters that customize the download for a given device. For example, `DLMaxWriteSize` is the limit for the download data payload size. This file is located in the *wiced_btsdk/dev-kit/baselib/<device>/platforms* folder. This folder also contains the minidriver hex file. Finally, the IntelHexToBin tool is located in *wiced_btsdk/dev-kitbtsdk-tools.*

1.1 IoT Resources and Technical Support

The wealth of data available **here** will help you to select the right IoT device for your design, and quickly and effectively integrate the device into your design. You can access a wide range of information, including technical documentation, schematic diagrams, product bill of materials, PCB layout information, and software updates. You can acquire technical documentation and software from the **Support Community website**.

2 Download Scenarios

This document will cover two scenarios: a complete reprogramming of the device after a flash chip erase, and a reprogramming of only the firmware portion of the device. The host operating the HCI will be referred to as the MCU.

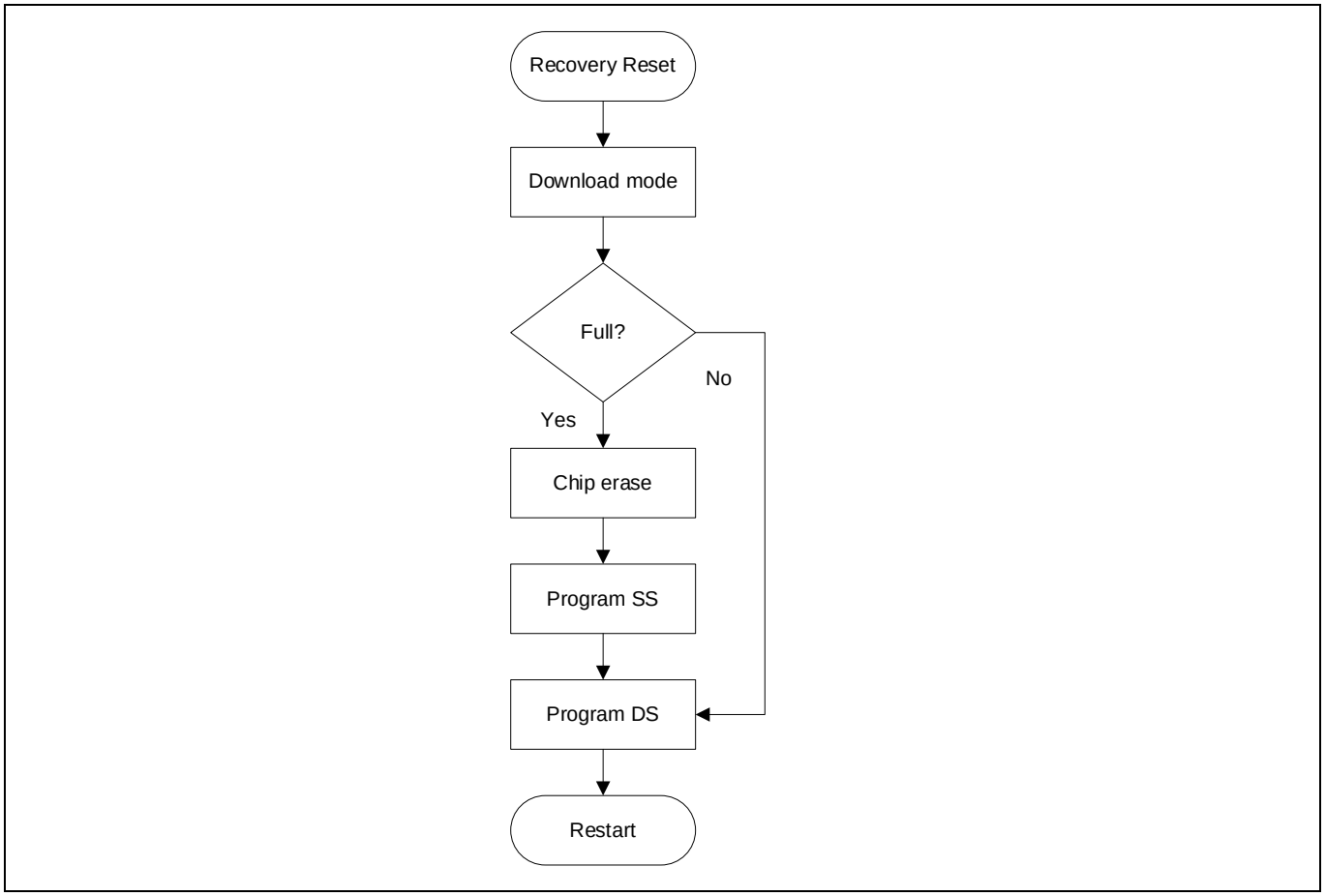
The first scenario is straightforward. The non-volatile memory is wiped, then reprogrammed. This scenario would be used for initial factory programming, for example, or during program development. This technique is used by the ModusToolbox “make … program” operation. In this document we will call this method “Full Download”.

The second scenario would be used for field upgrades of the firmware. This method would preserve key parts of non-volatile memory and only update the portions used for program execution. Any long-term identifying data such as Bluetooth Device Address, cryptographic keys, etc., would be preserved using this technique. We will refer to this method as “Upgrade Download”.

The non-volatile memory used by Bluetooth devices is managed in three logical partitions: Static Section (SS), Volatile Section (VS), and Data Section (DS). The SS contains data that is programmed at the factory and is intended to remain in place for the lifetime of the device. The VS contains data records that may be added, modified, or removed during run-time, but are maintained across power cycles. The DS contains code, data, and configuration records that are processed during the normal device boot (not Recovery Reset).

The Upgrade Download replaces only DS, while the Full Download replaces SS, VS, and DS.

The methods used for Full and Upgrade downloads overlap. This document will describe all the methods and their relations.

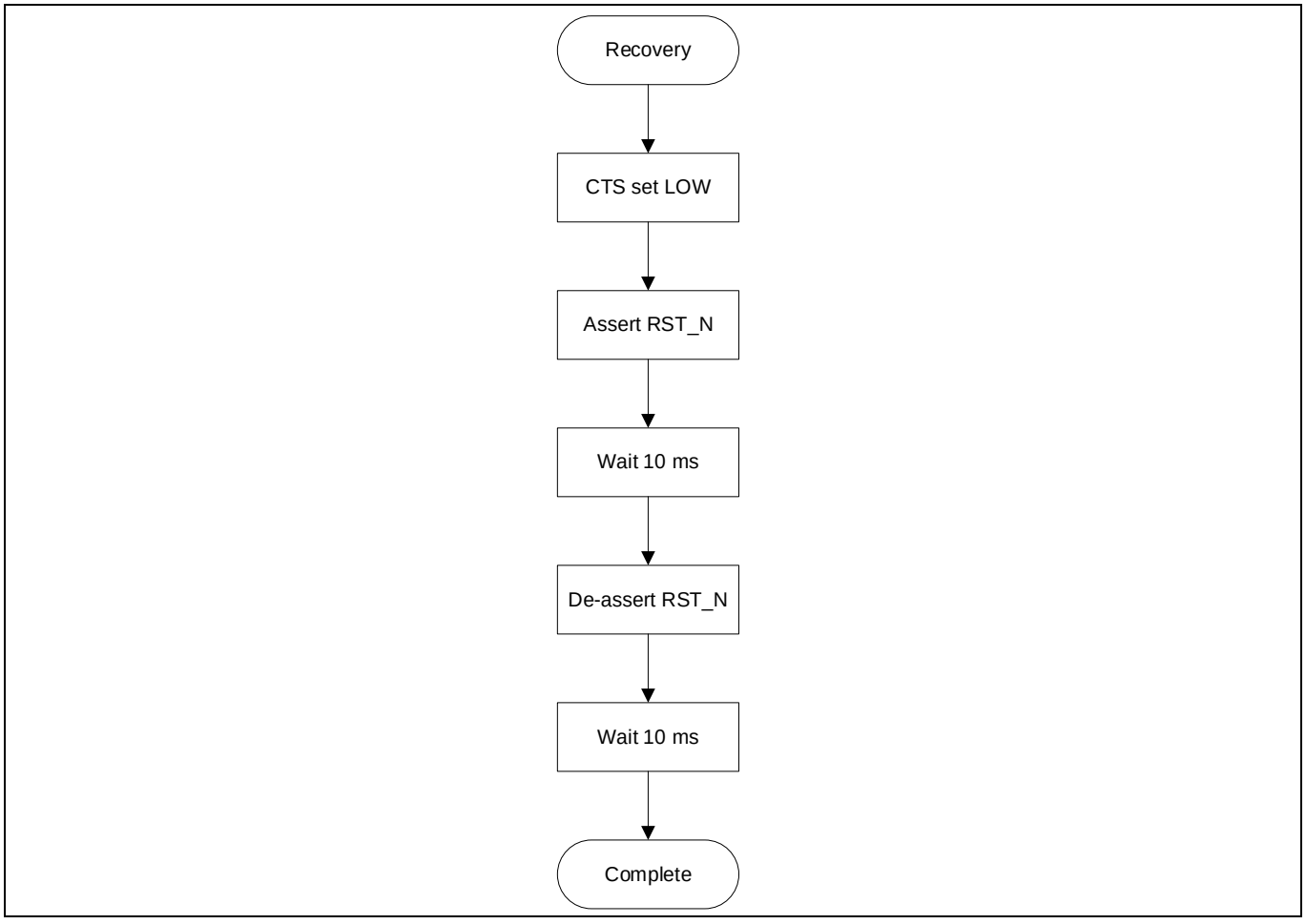


3 Entering Download Mode

3.1 Recovery Reset

Bluetooth devices supported by ModusToolbox enter download mode using Recovery Reset, described as follows.

When a Bluetooth device supported by ModusToolbox is initially powered on, the boot code will attempt to identify the hardware interface to be used for HCI communication. When using HCI via UART, the device behavior depends on the state of CTS (MCU side RTS) when RST_N is de-asserted. If CTS is LOW at this time, the device enters the autobaud state (download mode). If CTS is HIGH after reset, the device will check NVRAM and apply any stored configuration, typically ending in a mode ready to accept all HCI commands at a default baud rate. If no configuration is available, the device will also enter autobaud mode.



Summary:

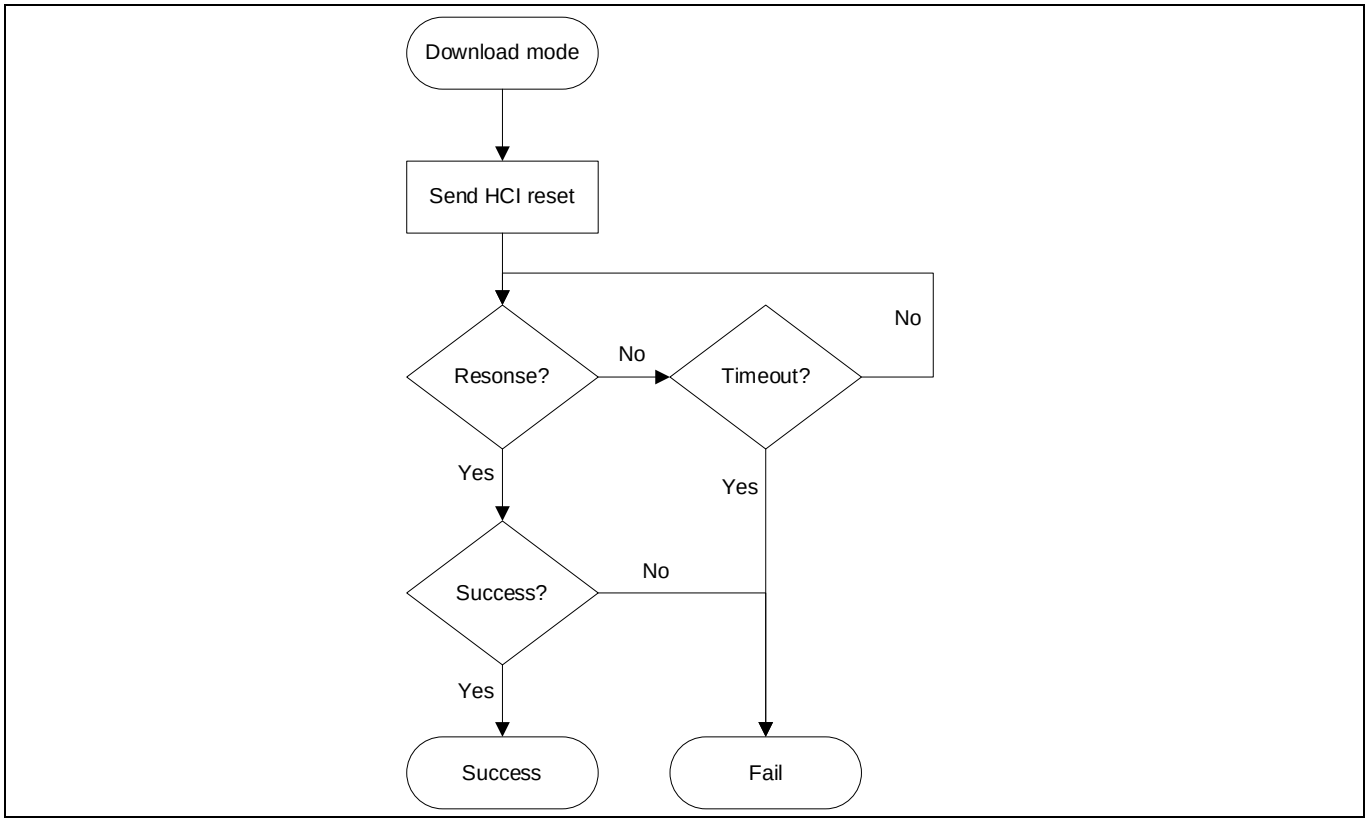
1. Set Chip CTS LOW (== host RTS).
2. Set chip reset LOW.
3. Wait 10 ms (could be a lot less but this is a good approximation).
4. Set chip reset HIGH.
5. Wait 10 ms.
6. Set chip CTS HIGH (idle).

3.2 Download Mode

When in download mode, the device checks the UART Rx for particular bit patterns and will adapt the baud rate for the best match. Reliable serial communication is obtained with an initial baud rate of 115200. To enter download mode, the MCU and device will exchange the following messages:

1. The MCU issues the following standard Bluetooth `HCI_RESET` command:
   `01 03 0C 00`
2. The following response is expected from the CYW20xxx device within 100 ms:
   `04 0E 04 01 03 0C 00`

After HCI reset and response, the device baud rate is set and communication can continue. The device is in download mode and can accept read, write, and launch commands.



3.3 Minidriver

The minidriver is a RAM-resident stand-alone program designed to download the code and data efficiently to the non-volatile memory. The minidriver supports some additional commands over the ROM download mode: chip erase, change baud rate, and validate with CRC.

The minidriver is provided as a .hex file in the Intel hex format (https://en.wikipedia.org/wiki/Intel_HEX). This can be converted to binary with the following command:

```
IntelHexToBin.exe minidriver.hex minidriver.bin
```

To determine the minidriver load address, check the hex file. For the example given, the minidriver should load to 0x00220000. The format consists of records delimited by ASCII carriage return (CR) and line feed (LF): 0xd, 0xa, but each record also has a start indicator, as described below.

1. Start code ":", ASCII 0x3a.
2. Byte count in record payload as two hexadecimal digits; for example, 'FF' is a count of 255.
3. Address as four hexadecimal digits; for example, '1000' would represent 0x1000 or 4096.
4. Record type as two hexadecimal digits. The record types used for download images are:
   a) '00' for data record, where the address field represents the low 16-bits of image destination.
   b) '01' for end of file (last record), the payload is zero bytes in length, and the address field is not used and set as '0000'.
   c) '04' for extended address (high 16 bits of the subsequent data record addresses).
   d) '05' for a 32-bit address. The record address field is left at '0000', the length is '04' bytes, and the eight hexadecimal data digits are interpreted as a 32-bit address. For example, '00220001' would be the address 0x220001. This record is often used to indicate a LAUNCH_RAM destination described below.
5. The record payload data represented as hexadecimal ASCII digits; two digits per byte and extending for the number of bytes indicated by the record's byte count.
6. Checksum of the entire preceding record data represented as two hexadecimal ASCII digits.

The first record of the minidriver hex file specifies the upper 16-bits of 32-bit address (type 02, Extended Segment Address). The following data records (type 00, Data) specify the lower 16-bits of address where the payload data is intended to occupy. The second to final record is a launch address to be used with the HCI Launch command (type 05, Start Linear Address). The last record is type 01, End Of File.

For example, the 20719B2 minidriver file begins with the following records, specifying upper 16-bits of address as 0x0022, followed by a 16-byte data record starting at 0x00220000, followed by a 16-byte data record starting at 0x00220010:

```
:020000040022D8
:10000000002DE9F0412C480024264684602B4800F05E
:10001000005FF82B4D6E626E6300F03DFE01F05CFBFD
```

The MCU writes the minidriver to the CYW20xxx device by sending `WRITE_RAM` commands corresponding to the data and address of the hex file data record payloads. The payloads can be accumulated into larger blocks, but the limit for the particular device as specified by `DLMaxWriteSize`. This is 240 (0xF0) bytes for the 20719B2, for example.

The following `WRITE_RAM` command is an example:

```
01 4C FC nn xx xx xx xx yy yy yy …
```

In the above `WRITE_RAM` command:

- `nn` is 4 + N, which represents four address bytes plus N payload bytes.
- `xx xx xx xx` is the 4-byte, mapped address for serial flash offset.
- `yy yy yy …` are the N payload bytes to be loaded into the mapped address. The following response to each `WRITE_RAM` command is expected within 200 ms:

  ```
  04 0E 04 01 4C FC 00
  ```

For the 20719B2, the minidriver, the first download command would look like the following.

```
01 4C FC F4 00 00 22 00 … 240 data bytes
```

Followed by the response

```
04 0E 04 01 4C FC 00
```

This pattern repeats until all the data records have been written. The minidriver program is then executed by the `LAUNCH_RAM` command:

```
01 4E FC 04 xx xx xx xx
```

In the above command, the `xx xx xx xx` bytes represent the destination address for the CPU branch as read from the hex file type 05, Start Linear Address record.

The following response to the `LAUNCH_RAM` command is expected within 200 ms:

```
04 0E 04 01 4E FC 00
```

At this point, the minidriver is executing and ready for commands to download the firmware image.

To summarize, the minidriver is downloaded with the following procedure:

1. `WRITE_RAM` 240 bytes to 0x00220000.
2. `WRITE_RAM` next 240 bytes to 0x00220000 + 0xF0.
3. Repeat until all data is transferred, incrementing destination address by 0xF0 each time.
4. `LAUNCH_RAM` to 0x00220000.
5. Wait 10 ms for minidriver to initialize before attempting to communicate.

4 Download Using Minidriver

The minidriver supports some additional commands to the ROM beyond read, write, and launch. The minidriver also supports chip erase, change baud rate, and verify CRC. In addition, the minidriver tracks the non-volatile memory sector usage, and will automatically erase a sector before writing to it for the first time. These additional commands are designed to improve the download speed.

4.1 Download Preparation

When performing the Full Download or the Upgrade Download, the first operation is typically to change the baud rate.

To speed up application downloading, the MCU host commands the CYW20xxx device to communicate at a new, higher rate by issuing the following vendor-specific `UPDATE_BAUDRATE` command:

```
01 18 FC 06 00 00 xx xx xx xx
```

In the above command, the `xx xx xx xx` bytes specify the 32-bit value of the new rate in bits per second. For example, 115200 is represented as `00 C2 01 00`.

The following response to the `UPDATE_BAUDRATE` command is expected within 100 ms:

```
04 0E 04 01 18 FC 00
```

The host switches to the new baud rate after receiving the response at the old baud rate. For 20719B2, the `UPDATE_BAUDRATE` command to 3000000 baud is, for example:

```
01 18 FC 06 00 00 C0 C6 2D 00
```

With a response within 20 ms as follows:

```
04 0E 04 01 18 FC 00
```

Then the process continues at a baud rate of 3000000.

When performing the Full Download, the next operation is `CHIP_ERASE`. If it is desirable to preserve some data in flash and only erase sectors that will be written, this step may be skipped.

The following `CHIP_ERASE` command is an example:

```
01 CE FF 04 xx xx xx xx
```

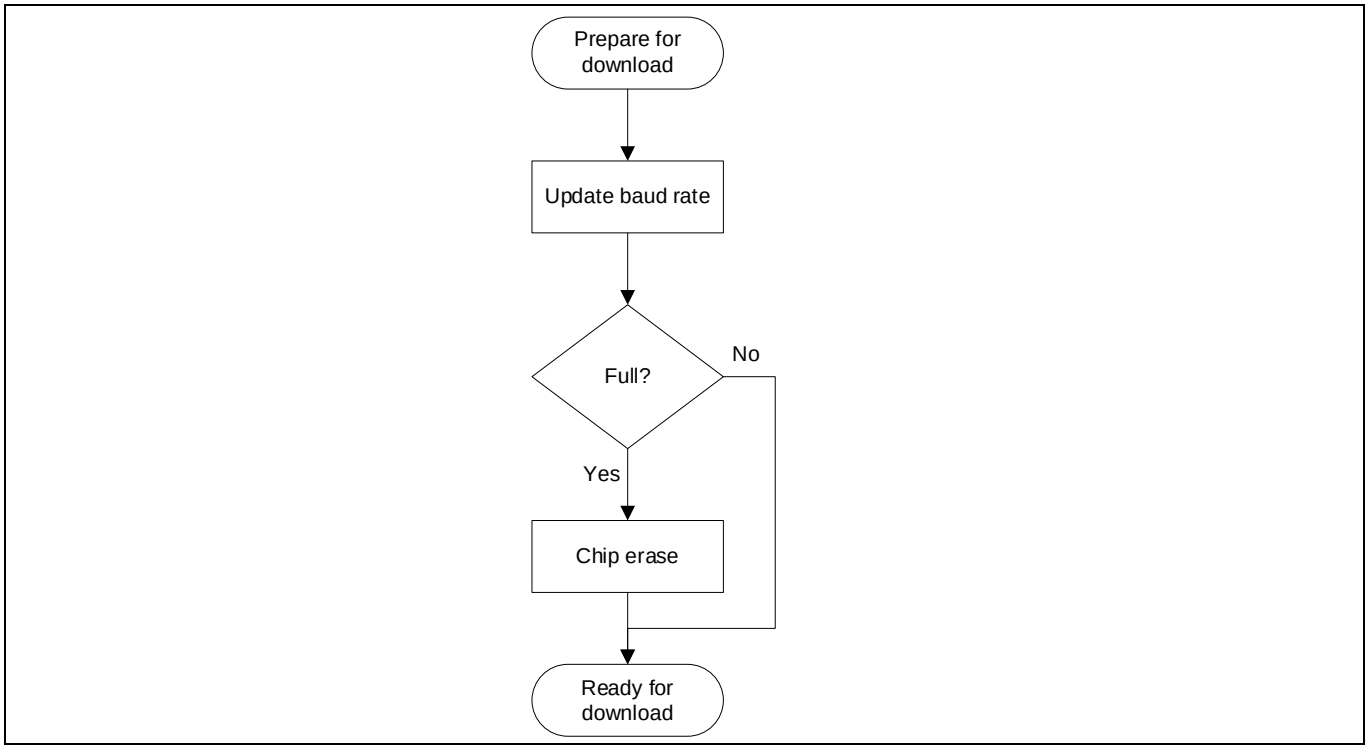
With a response following (timing depends on flash specifications, for 20719B2 within 300 ms):

```
04 0E 04 01 CE FF 00
```

In the above `CHIP_ERASE` command, `xx xx xx xx` is the 4-byte address indicating the range of the non-volatile memory to be erased. A special value of `EF EE BE FC` (0xFCBEEEEF) is used to signal “use the lowest valid non-volatile memory range”. Otherwise, the device to be erased is determined from the value when compared to valid ranges, where 0x500000 would be the start of on-chip flash when supported and 0xFF000000 would be the start of off-chip flash.

For the 20719B2, the `CHIP_ERASE` command is, for example:

```
01 CE FF 04 EF EE BE FC
```



4.2 Download

The download process using the minidriver continues with `WRITE_RAM` commands, similar to the process of downloading the minidriver. In order to validate the download, the MCU should accumulate the CRC-32 for the data transferred. After each section of data (SS, DS) is transferred, the CRC of the section in non-volatile memory can be checked by the minidriver as described below.

The download operation can be guided by the download data file. For example, if a Full Download is being performed, the download file should have data blocks for SS and DS sections. If an Upgrade Download is being performed, the download file should only have a DS data block. The download files are created with input from the .btp file, where SS, VS, and DS non-volatile memory addresses are specified. For example, the 20719B2 .btp file has:

```
DLConfigSSLocation = 0x500000
DLConfigVSLocation = 0x501000
ConfigDSLocation = 0x503000
```

This indicates that contiguous data in the download file that starts at 0x500000 will be the SS data. Contiguous data starting at 0x503000 will be the DS data. Using this information, the **_download.hex* file can be used to generate separate binary data files for SS and DS using the IntelHexToBin tool provided with ModusToolbox.

For example, the 20719B2 *_download.hex can be used in this manner:

```
IntelHexToBin.exe -l 0x503000 BLE_Beacon_download.hex BLE_Beacon_download_DS.bin
IntelHexToBin.exe -u 0x501000 BLE_Beacon_download.hex BLE_Beacon_download_SS.bin
```

These binary data files can be read in 240 byte chunks to form `WRITE_RAM` commands to download the SS and DS data.

An example of data download for SS for the 20719B2 is:

```
01 4C FC 46 00 00 50 00 … 66 data bytes in SS
```

After a data section is downloaded, the non-volatile memory image can be validate using CRC-32. The minidriver is given the `VerifyCRC` command:

```
01 CC FC 08 xx xx xx xx yy yy yy yy
```

In the above command, the `xx xx xx xx` bytes specify the 32-bit value of the mapped address for the serial flash offset and the `yy yy yy yy` bytes specify the 32-bit value of the number of bytes to be read from the serial flash for the CRC calculation.

The following response is expected after the `VerifyCRC` command:

```
04 0E 08 01 CC FC 00 xx xx xx xx
```

In the above response, the xx bytes are the 32-bit CRC-32 value calculated by reading the data bytes from the flash starting at the virtual address given in the CRC command, and continuing for the number of bytes provided in the CRC command.

An example of the CRC verification of the SS data block for the 20719B2 is:

```
01 CC FC 08 00 00 50 00 42 00 00 00
```

With a successful response expected within 300 ms of:

```
04 0E 08 01 CC FC 00 35 78 A9 35
```

A similar process can be applied to the DS section. The Full Download would write both SS and DS sections after a `CHIP_ERASE`. The Upgrade Download would only write the DS section and would not use a `CHIP_ERASE`. Other than that, the processes are similar.

4.3 Download Completion

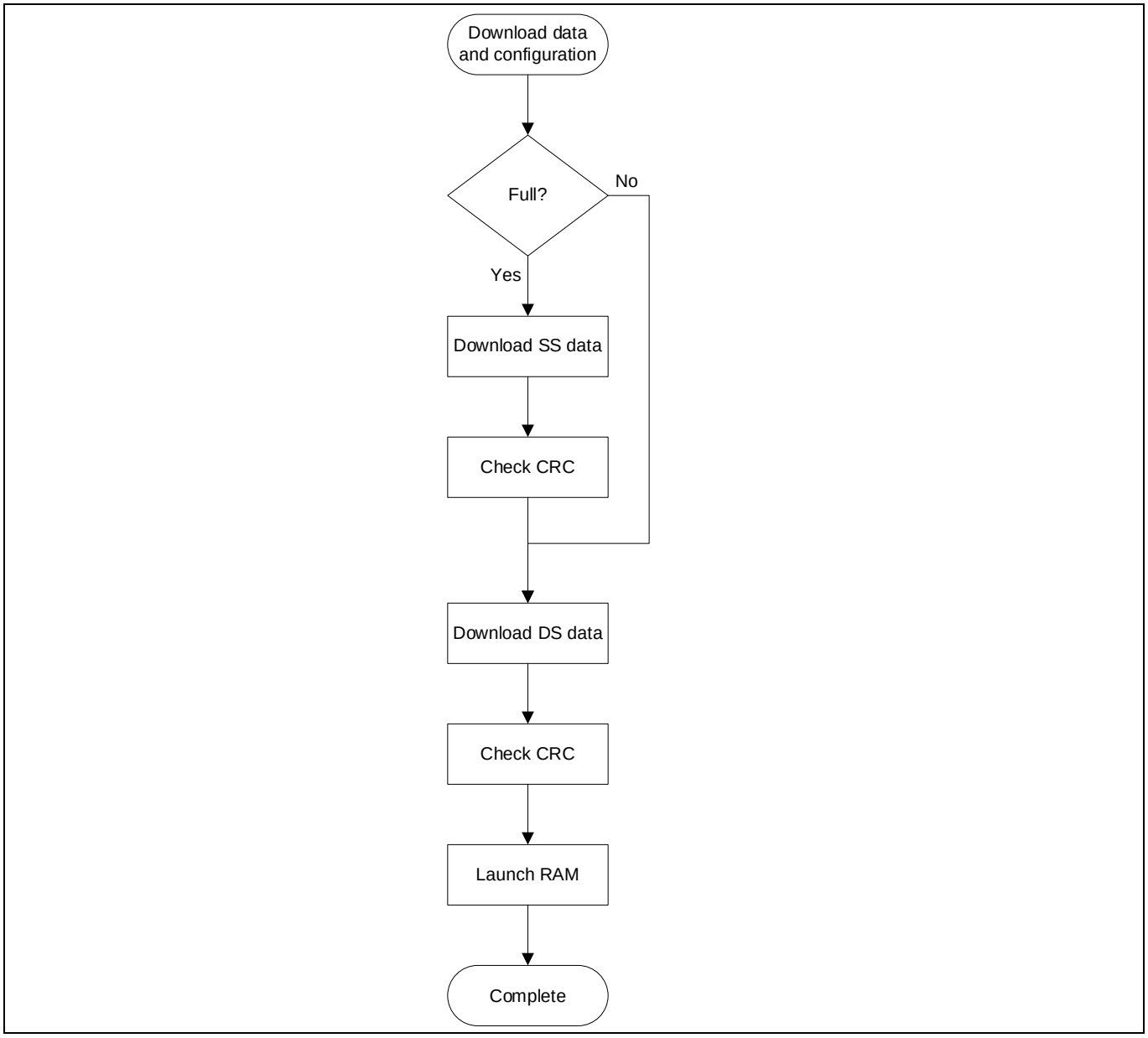
After download and verification using the minidriver, the device can be rebooted for completion of the process. This is done with the `LAUNCH_RAM` command, but using a magic address of 0x0000000.

An example of download completing reset for the 20719B2 is:

```
01 4E FC 04 00 00 00 00
```

With a successful response expected within 10 ms of:

```
04 0E 04 01 4E FC 00
```


Download data and configuration
Full?
No
Yes
Download SS data
Check CRC
Download DS data
Check CRC
Launch RAM
Complete

Revision history

Document version	Date of release	Description of changes
**	2021-01-15	Initial release

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

Edition 2021-01-15

Published by

Infineon Technologies AG

81726 Munich, Germany

Do you have a question about this document?

Go to www.cypress.com/support

Document reference

002-32056 Rev. **

IMPORTANT NOTICE

The information given in this document shall in no event be regarded as a guarantee of conditions or characteristics (“Beschaffenheitsgarantie”) .

With respect to any examples, hints or any typical values stated herein and/or any information regarding the application of the product, 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.

In addition, any information given in this document is subject to customer’s compliance with its obligations stated in this document and any applicable legal requirements, norms and standards concerning customer’s products and any use of the product of Infineon Technologies in customer’s applications.

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.