RN4678, BM78 Firmware Upgrade using Host MCU

Table of Contents

1   Overview

The IS1678 chip contained within the BM78 or RN4678 module has a code memory (Flash) and configuration memory(EEPROM). The code memory stored the firmware of BM78. There are 5 banks composing the flash and each bank is in 64Kbytes size.

All the data are aligned in big endian in this document.

1.1   Memory programming

To perform any memory programming operation, the user needs to follow the process flow shown in the flowchart in Figure 1 with respect to the command protocols. The steps are explained below:

- **Enter memory programming Mode**

  Set the module into the 'Write Flash' mode by setting P2_0/P2_4/EAN to ON as the TABLE 2-4 in the BM78 EVB user guide and then issue an hardware reset by controlling RESET pin.

TABLE 2-4:   BM78 EVB MODE SWITCH POSITIONS

Mode		Switch Positions	PIN Definition		
			1/P2_0	2/P2_4	3/EAN
Flash	Write Flash		ON	ON	ON
ROM	Test (Write EEPROM)		ON	OFF	ON
	Application (default)		OFF	OFF	ON
Flash	Test (Write EEPROM)		ON	OFF	OFF
	Application (default)		OFF	OFF	OFF

- **Connect to the Flash**

use CSEQ Get Flash ID command connect to the Flash in the BM78 and a flash ID is returned for verification.

- **Flash operations**

CSEQ Flash Erase/Bank Switch/Flash Write command can be use to implement device firmware upgrade(DFU).

- **Disconnect from Flash**

  After the DFU, host MCU needs set BM78 into 'Application Mode' by setting P2_0/P2_4/EAN to OFF as the TABLE 2-4 in the BM78 EVB user guide and then issue an hardware reset by controlling RESET pin.
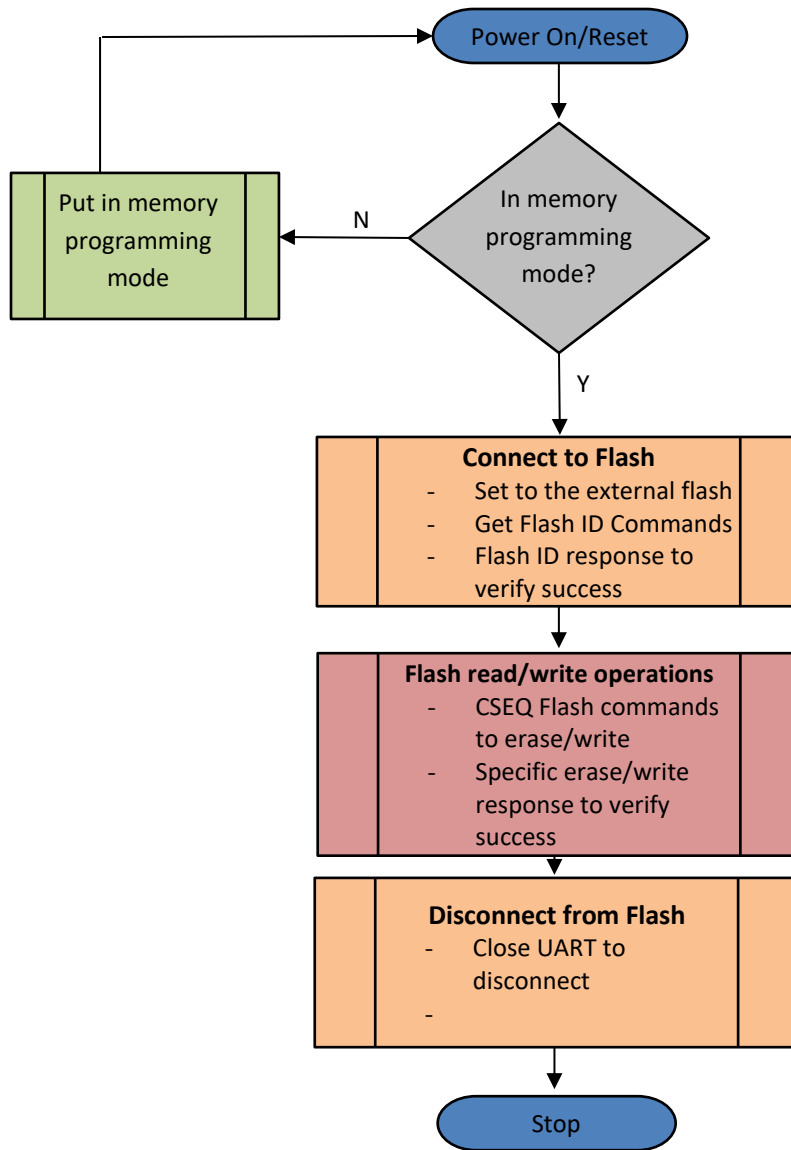
Figure 1. Overview of memory programming process

2 Device Firmware Upgrade Process

2.1 Entering Flash Programming Mode

Pin P2_0, P2_4 and EAN should be set to 'Write Flash' mode according TABL 2-1 in the datasheet at https://ww1.microchip.com/downloads/en/DeviceDoc/60001380C.pdf.

TABLE 2-1: SYSTEM CONFIGURATION SETTINGS

Module	P2_0	P2_4	EAN	Operational Mode
BM78SPPx5NC2 (ROM Variant)	Low	High	High	Test mode (Write EEPROM)
	High	High	High	APP mode (Normal operation)
BM78SPPx5MC2 (Flash Variant)	Low	Low	High	Write Flash
	Low	High	Low	Test mode (Write EEPROM)
	High	High	Low	APP mode (Normal operation)

2.2 Connect to the Flash

To connect to the Flash via UART in memory programming, CSEQ commands 0x70 and 0x50 are used.

- Set Flash (Opcode: 0x70)
  Set the flash as the external flash by CSEQ Set Flash command. This is the first flash operation command sent to BM78 during DFU.

- Get Flash ID (Opcode: 0x50)
  Read the flash ID from BM78 by CSEQ Get Flash ID command. The flash ID is returned with this command. Host MCU doing DFU might check it for confirmation. It must be sent after 'Set Flash' command.

2.3 Flash operations

After connecting to the flash, the host may carry flash operations like Erase Flash, Switch Bank, and Write Flash as below.

- Flash Erase command (Opcode: 0x51)
  This command erases entire external flash of the BM78.

- Switch Bank Command (Opcode: 0x63)
  Before Write Flash, this command is used to switch to the target bank.

- Write Flash command (Opcode: 0x45)
  Use this command to write current bank of flash with data specified in this command. The max data is limit to 240Bytes.

2.4 Disconnect

Once the DFU is done, a hardware reset is needed after setting P2_0, P2_4 and EAN to APP mode.

3 Firmware image manipulation

There are 5 banks of image files composing the entire firmware. Inside each image file, there are many records indicating which type of hex record it presents. Typically, there are 3 types of hex records including bank start record, data/firmware record and bank end record. The firmware is resided in data type record.

Normally, each bank of image file stands for a 64K Bytes of firmware. And the data/firmware records are aligned from the offset 0x0000 of each bank and each data/firmware record is aligned in address ascendant order. And there is no hole in-between the records.

	Start	Data Length	Starting address	Category	Data	Checksum	End
Nibble#	0	1-2	3-6	7-8	9..n	n + 1	n+2
Value	:	0xNN	0x0000 – 0xFFE0	0x04: Start of bank 0x00: Data 0x01: End of Bank	Data; can be up to 32 bytes	0xXX	'\n'

But there always are exceptions. For example the offset might not always start from 0x0000 and there might be empty hole in- between the data/firmware records. To ease this problem, isUpdate Tool can be used to Rehex the 5 banks of image into one large hex file in which each data/firmware record is placed in sequence. Each empty hole and each non-zero offset are filled with empty record of 0xFF so that the host MCU can easily download the firmware data in sequence from 0x0000 of bank0 to the end of bank4. At the same time, a binary file with pure firmware is also generated. In this reference design, we turn the binary file into a large constant C array which resides in the flash memory of the host MCU.
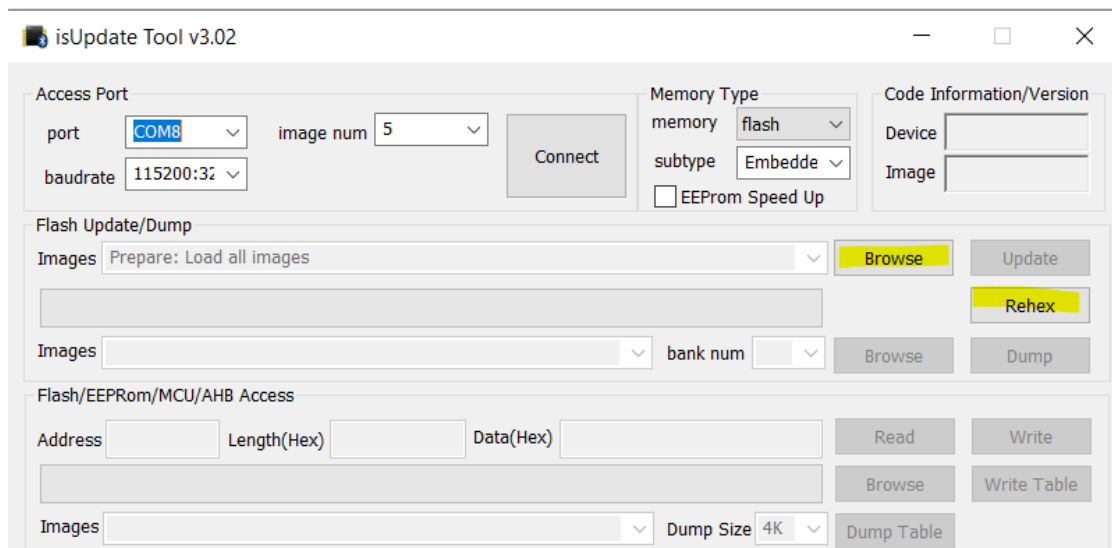
4 CSEQ Request, Response and Event

Requests are sent from the host and the corresponding responses are expected from BM78. There might be events from BM78 and the expected responses are from the host. All data fields mentioned below are aligned in big endian.

4.1 Format of CSEQ Request (Host --> BM78)

The request is always sent from the host MCU. The CSEQ payload includes OPCODE, parameter and CRC fields. Any 0x3c in payload should be escaped with 0x3C so that 0x3c 0x3c sequence represents the 0x3c in the payload.

The CRC is calculated against the raw data of OPCODE and parameter without escaping with 0x3c.

	CSEQ_DEL (Delimiter)	CSEQ_ BEGIN	CSEQ_ OPCODE	CSEQ parameter	CRC	CSEQ_DEL (Delimiter)	CSEQ_END
BYTE NO.	0	1	2	3 - N	N+1 - N+2	N+3	N+4
Size (BYTES)	1	1	1	3…n	2	1	1
VALUE	0x3C	0x66	0xXX	0xXXXXXX	0xXXXX	0x3C	0x5A
	Packet Indicator		CSEQ payload			Packet end	

4.2 Format of CSEQ Event (BM78 --> Host)

On receiving a request, an event might be returned from BM78. The event packet shares the same format of the CSEQ Request. As it does to CSEQ Request, a response must be sent on receiving the event.

4.3 Format of CSEQ Response (BM78<--> Host)

There are 2 types of response: ACK and NAK. There is no CRC in response packet. A response must be given to the peer on receiving a request or event.

	CSEQ_DEL (Delimiter)	CSEQ_
BYTE NO.	0	1
Size (BYTES)	1	1
VALUE	0x3C	0x33(ACK) or 0x55(NAK)
	Packet Indicator	

5 CSEQ Flash Request overview

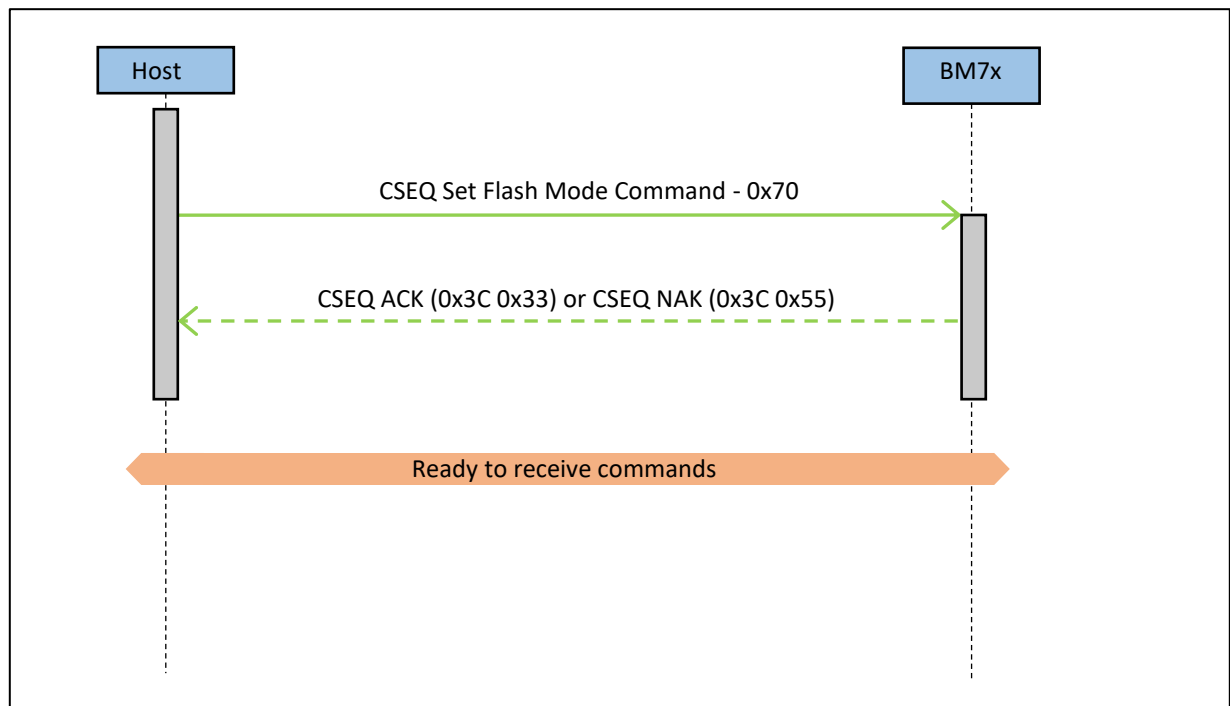Totally there are 5 CSEQ requests to manipulate the flash as below.

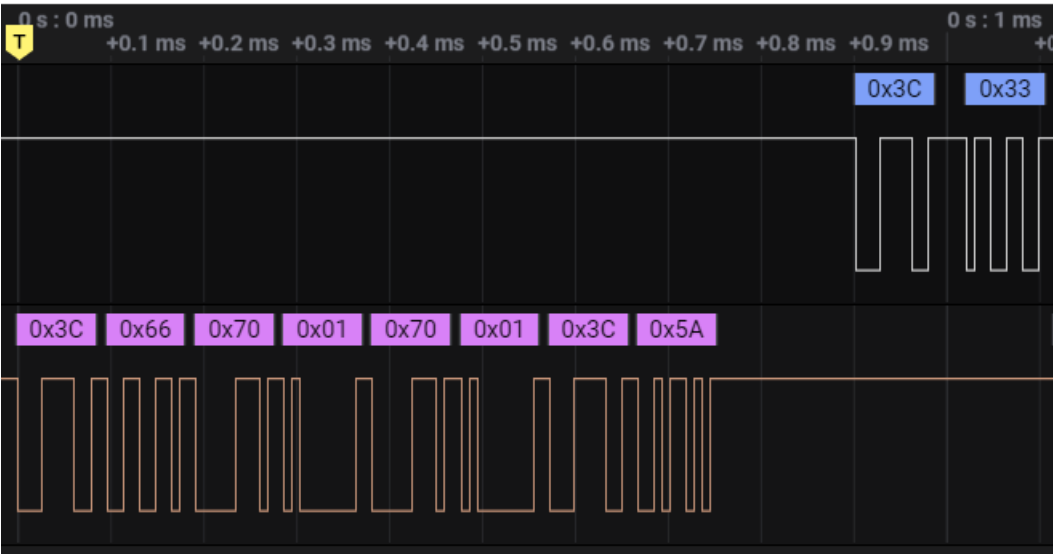| OpCode		Parameter Description
0x45	Write Flash	Write to program space
0x50	Get Flash ID	Read flash id
0x51	Erase Flash	Erase entire external flash
0x63	Switch Bank	Switch flash bank
0x70	Set Flash Mode	Set to external flash

5.1 Set Flash Request

The parameter is fixed to 0x01.

	CSEQ_DEL	CSEQ_BEGIN	CSEQ OPCODE	CSEQ parameter	CSEQ CRC	CSEQ_DEL	CSEQ_END
BYTE NO.	0	1	2	3	4-5	6	7
Size (BYTES)	1	1	1	1	2	1	1
VALUE	0x3C	0x66	0x70	0x01	0xXXXX	0x3C	0x5A

On receiving the 'Set Flash Request', an ACK or NAK might be returned.

Host   BM7x

CSEQ Set Flash Mode Command - 0x70

CSEQ ACK (0x3C 0x33) or CSEQ NAK (0x3C 0x55)

Ready to receive commands

The example data:

5.2 Get Flash ID Request

There is no parameter to this request.

	CSEQ_DEL	CSEQ_BEGIN	CSEQ OPCODE	CSEQ CRC	CSEQ_DEL	CSEQ_END
BYTE NO.	0	1	2	3-4	5	6
Size (BYTES)	1	1	1	2	1	1
VALUE	0x3C	0x66	0x50	0xXXXX	0x3C	0x5A

The flash ID is returned in the Event.
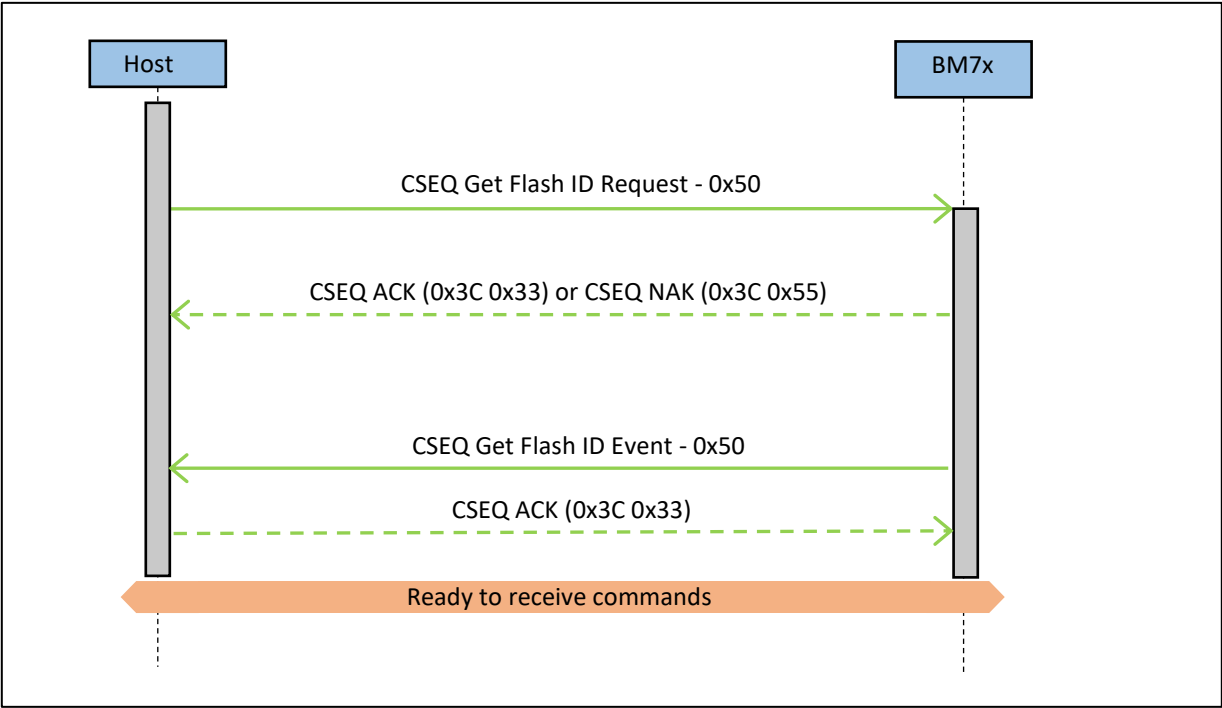
Example data: the request and response
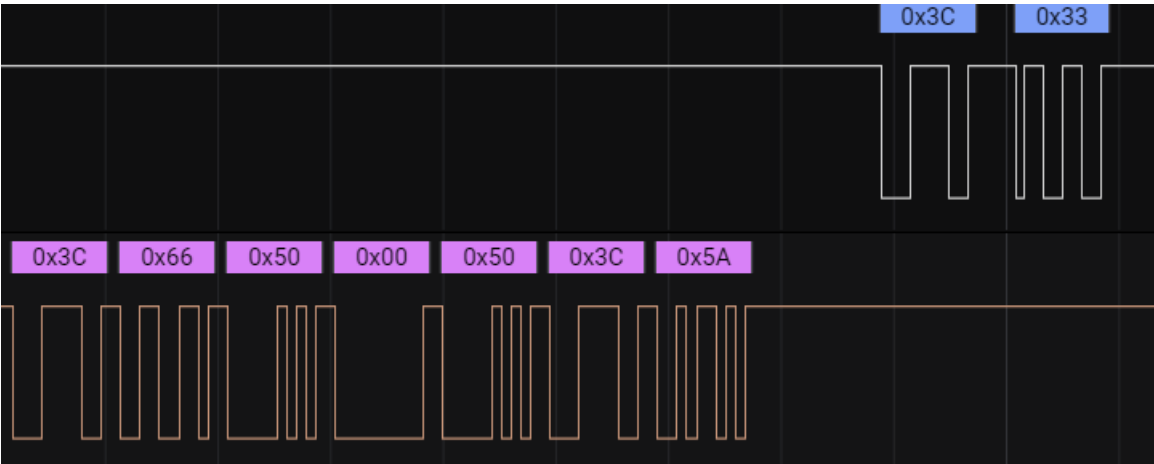
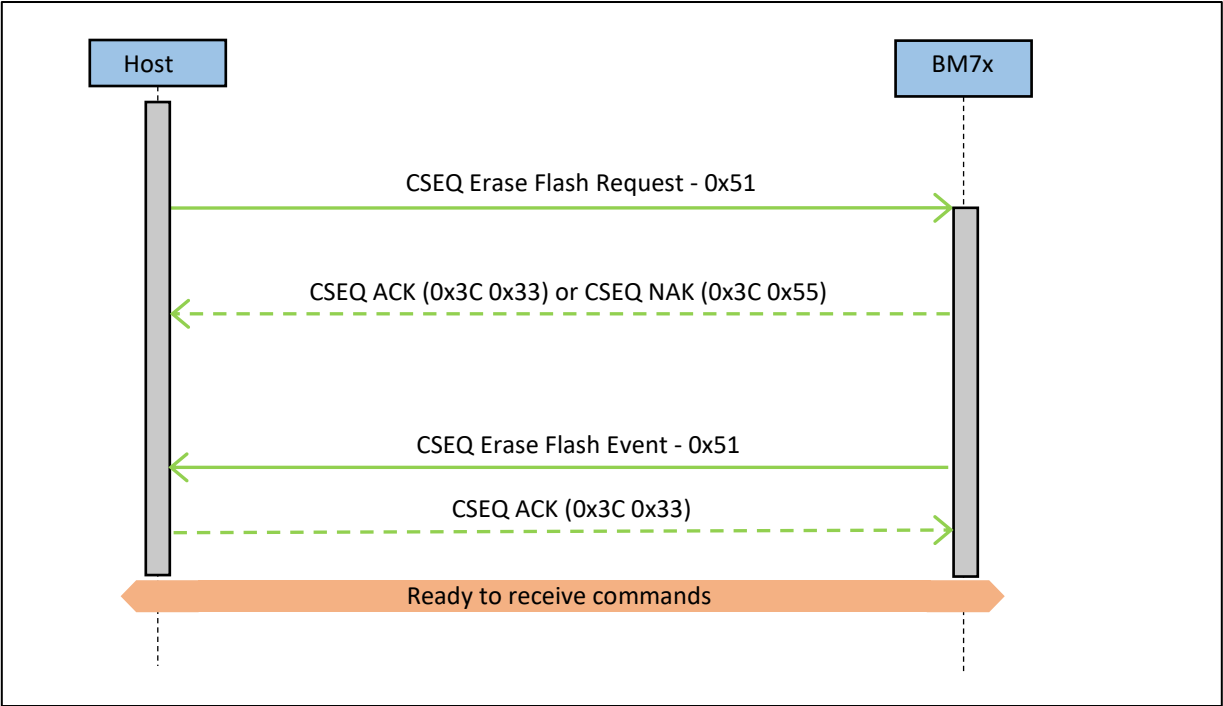Example data: the event and response. The flash ID 0x1CBA is returned in the event.

5.3 Erase Flash Request

There is no parameter to this request.

	CSEQ_DEL	CSEQ_BEGIN	CSEQ OPCODE	CSEQ CRC	CSEQ_DEL	CSEQ_END
BYTE NO.	0	1	2	3-4	5	6
Size (BYTES)	1	1	1	2	1	1
VALUE	0x3C	0x66	0x51	0xXXXX	0x3C	0x5A

An ACK or NAK is returned to the request. It takes time to send the erase status event to the host MCU.

Example data: the request and response

Example data: the erase status event and the response

5.4 Switch Bank Request

Before Write Flash, the Switch Bank must be sent to set target bank. The bank number is set in the parameter field.

	CSEQ_DEL	CSEQ_BEGIN	CSEQ OPCODE	CSEQ parameter	CSEQ CRC	CSEQ_DEL	CSEQ_END
BYTE NO.	0	1	2	3	4-5	6	7
Size (BYTES)	1	1	1	1	2	1	1
VALUE	0x3C	0x66	0x63	Bank number	0xXXXX	0x3C	0x5A

On receiving the request, an ACK or NAK will be returned. After that, the Switch Bank Event is sent from BM78 with the same format as the Switch Bank Request.
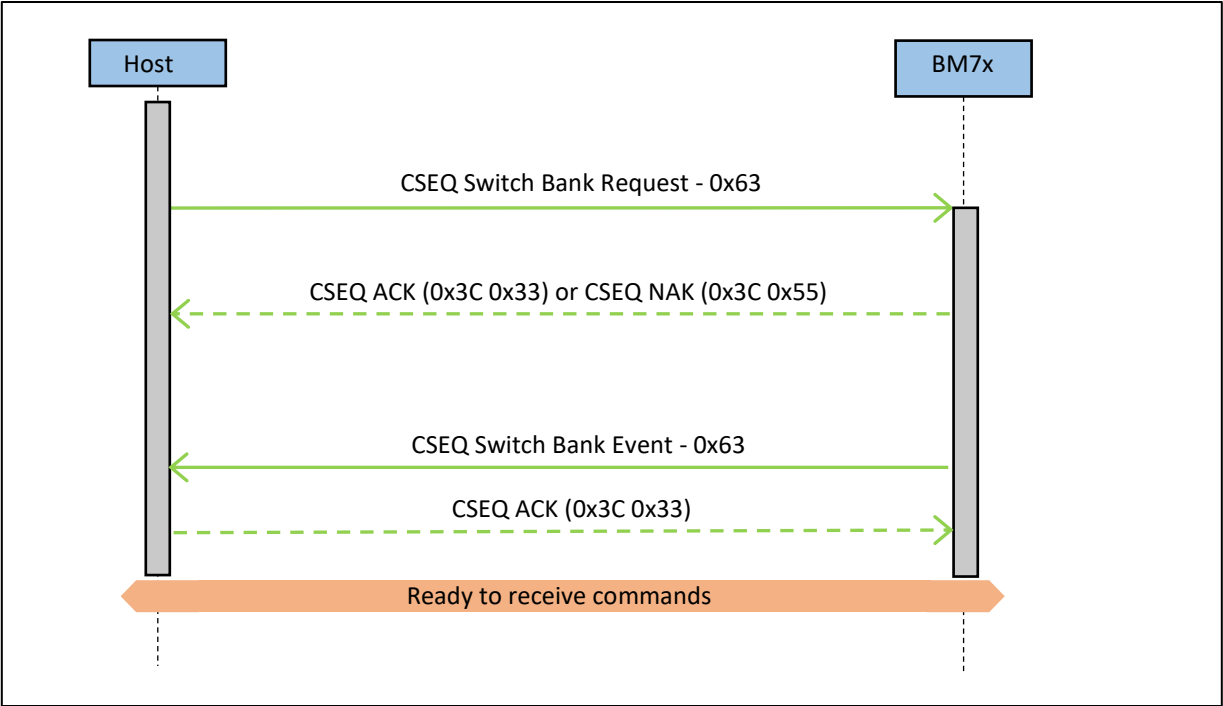
Sample data: the request and response
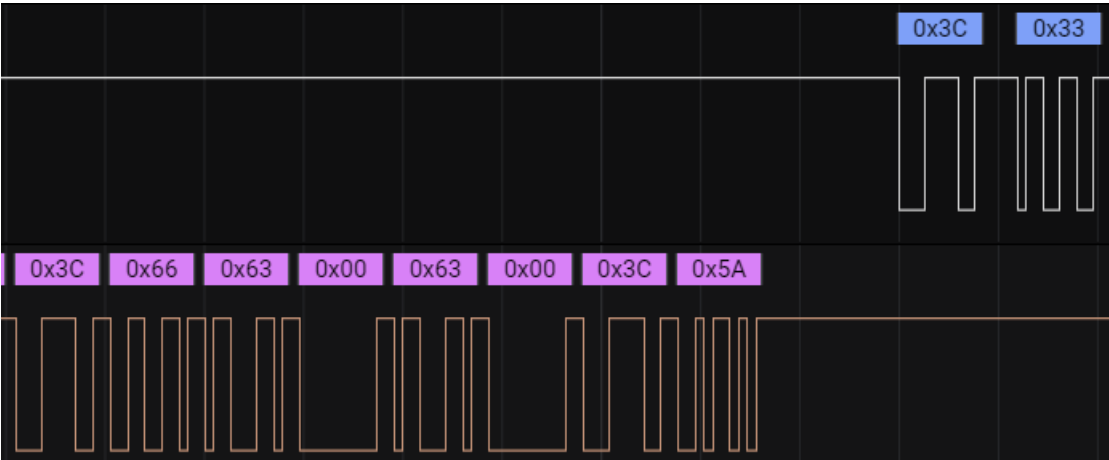
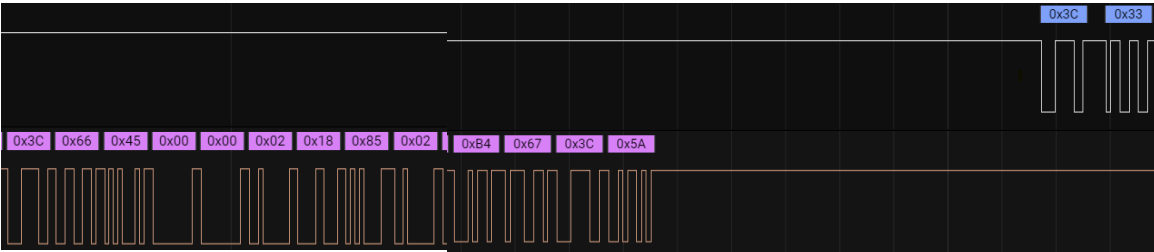Sample data: the switch bank event and response

5.5 Write Flash Request

Two parameters are needed for this request. One is the offset of the firmware data to send.
Another is the firmware data itself. The max firmware data is limit to 240 bytes. After the
firmware is flashed by BM78, an event is generated and returned with the firmware offset and
the firmware data length.

	CSEQ_ DEL	CSEQ_ BEGIN	CSEQ_ OPCODE	CSEQ_ OFFSET	CSEQ_DATA	CSEQ_CRC	CSEQ_ DEL	CSEQ_ END
BYTE NO.	0	1	2	3 - 4	5-n	n+1-n+2	n+3	n+4
Size (BYTES)	1	1	1	2	n	2	1	1
VALUE	0x3C	0x66	0x45	0xXXXX	0xXXXX…	0xXXXX	0x3C	0x5A

Sample data: the write flash request and response. In this reference design, the firmware data
length is set to 32 and the snapshot is just a part of the total packet.

Sample data: the write flash Event and response. In the event, 0x0000 is the previous firmware data offset and 0x20 is the data length of the firmware chunk.

5.6 CRC Calculation

CRC is calculated against with the raw payload without escaping with 0x3c. The algorithm is provided in the reference design. And it is also escaped with 0x3c if including 0x3c in the CRC result.

6 Setup the demo

This reference design uses SAMG55 XPLAINED PRO evaluation kit and BM78-PICTAIL. The SAMG55 will control the whole process of firmware DFU to BM78 using UART.

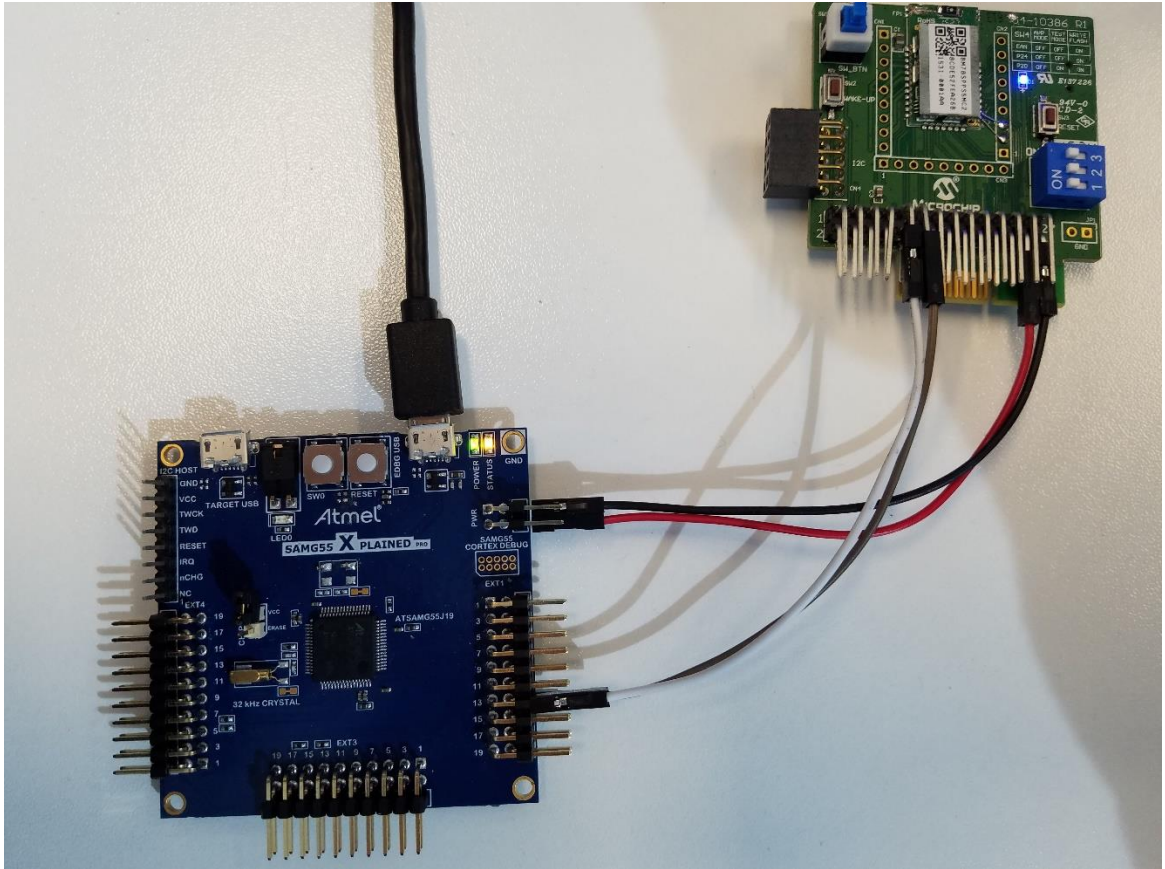SAMG55 XPLAINED PRO evaluation kit refers to:

https://www.microchip.com/DevelopmentTools/ProductDetails/PartNO/ATSAMG55-XPRO

BM78-PICTAIL refers to:

https://www.microchip.com/DevelopmentTools/ProductDetails/PartNO/BM-78-PICTAIL

BM78 module refers to:

https://www.microchip.com/wwwproducts/en/BM78

6.1 Hardware Setup

- Connect the EDEBUG USB of SAMG55 EVB to PC
- Open the Terra Term to this EDEBUG USB on PC
- Connect UART0 on EXT1 of SAMG55 EVB to BT_UART on J1 of BM78 EVB

	SAMG55	BM78
UART Connection	UART_RX Pin13	BT_UART_TX Pin9
	UART_TX Pin14	BT_UART_RX Pin11

- Connect 3v3 on J100 of SAMG55 EVB to EXT_3v3 on J1 of BM78

	SAMG55	BM78
Power connection	3v3 Pin4	EXT_3v3 Pin26
	GND Pin2	GND Pin28

- To ease the connection setup, P2_0, P2_4, EAN and RESET Pin are not connected to SAMG55 and they are controlled by the BM78 EVB.
Use SW4 to set BM78 in 'WRITE FLASH' mode with all pin set to ON. And then do a hardware reset by pressing SW3 the reset button. Then the blue LED will be blinking and BM78 is set to 'WRITE FLASH' mode.
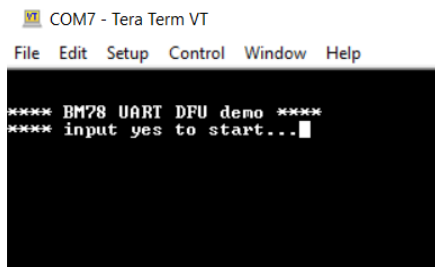
6.2   Firmware Setup

6.2.1   Open the MPLAB-X project
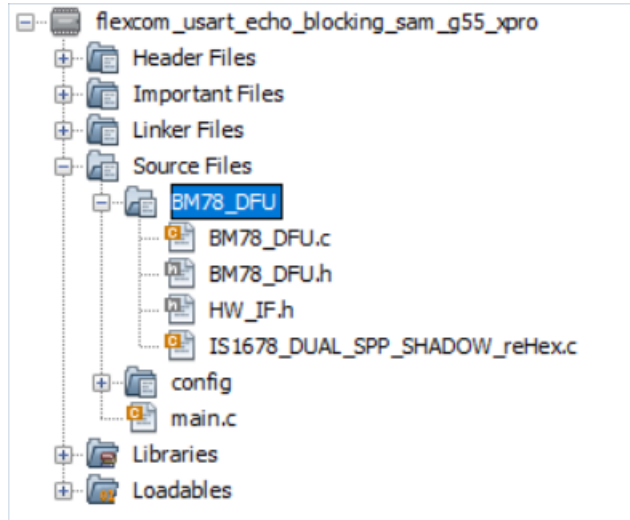
The SAMG55 firmware is designed with MPLAB-X and Harmony latest revision. Open the project and one may see the BM78_DFU which includes the DFU code and a processed BM78 1v38 firmware which is stored in a C array. The only one exploded API to application level is BM78_DFU( ).

6.2.2   Run the SAMG55 firmware

- Put BM78 in 'WRITE FLASH' mode manually as section 6.1
- Open a Tera Term on PC
- Compile and download the SAM55
- On the Tera term, user is prompted to input yes to start the DFU
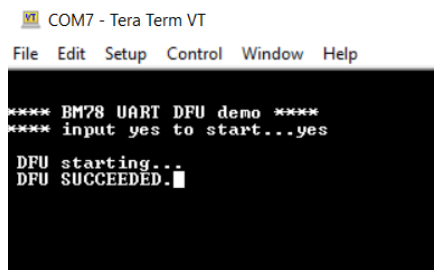
- Input 'yes' to start the DFU

- After around 2 minutes, the DFU is done with LED0 of SAMG55 on indicating a success DFU

7 Port the code

The reference code is designed for easy porting to other MCU and its framework if customer is not using SAMG55 and Harmony framework.

As shown below, the hardware relevant API is declared in the HW_IF header file. There are 9 macro needed modification according to the user host MCU architecture and its framework if any. The 9 APIs are categorized into 3 types: GPIO control, UART interface and an optional delay API if more responsiveness is required.


```
73   #define RESET_BM78( )            do{}while(0)
74   #define SET_BM78_FLASH_MODE( )   do{}while(0)
75   #define SET_BM78_EEPROM_MODE( )  do{}while(0)
76   #define SET_BM78_APP_MODE( )     do{}while(0)
77
78   #define UART_INIT()              FLEXCOM0_USART_Initialize()
79   #define UART_RX_READY()          FLEXCOM0_USART_ReceiverIsReady()
80   #define UART_RX( buffer, size )  FLEXCOM0_USART_Read( buffer, size )
81   #define UART_TX( buffer, size )  FLEXCOM0_USART_Write( buffer, size )
82   #define UART_DELAY_10ms( )       do{}while(0)
```

8 Appendix

A UART log file, BM78-1V38-DFU-log-32Byte.sal, in saleae Logic2 format is provided for reference.