Spansion® Analog and Microcontroller Products

The following document contains information on Spansion analog and microcontroller products. Although the document is marked with the name "Fujitsu", the company that originally developed the specification, Spansion will continue to offer these products to new and existing customers.

Continuity of Specifications

There is no change to this document as a result of offering the device as a Spansion product. Any changes that have been made are the result of normal document improvements and are noted in the document revision summary, where supported. Future routine revisions will occur when appropriate, and changes will be noted in a revision summary.

Continuity of Ordering Part Numbers

Spansion continues to support existing part numbers beginning with "MB". To order these products, please use only the Ordering Part Numbers listed in this document.

For More Information

Please contact your local sales office for additional information about Spansion memory, analog, and microcontroller products and solutions.

Colophon

The products described in this document are designed, developed and manufactured as contemplated for general use, including without limitation, ordinary industrial use, general office use, personal use, and household use, but are not designed, developed and manufactured as contemplated (1) for any use that includes fatal risks or dangers that, unless extremely high safety is secured, could have a serious effect to the public, and could lead directly to death, personal injury, severe physical damage or other loss (i.e., nuclear reaction control in nuclear facility, aircraft flight control, air traffic control, mass transport control, medical life support system, missile launch control in weapon system), or (2) for any use where chance of failure is intolerable (i.e., submersible repeater and artificial satellite). Please note that Spansion will not be liable to you and/or any third party for any claims or damages arising in connection with above-mentioned uses of the products. Any semiconductor devices have an inherent chance of failure. You must protect against injury, damage or loss from such failures by incorporating safety design measures into your facility and equipment such as redundancy, fire protection, and prevention of over-current levels and other abnormal operating conditions. If any products described in this document represent goods or technologies subject to certain restrictions on export under the Foreign Exchange and Foreign Trade Law of Japan, the US Export Administration Regulations or the applicable laws of any other country, the prior authorization by the respective government entity will be required for export of those products.

Trademarks and Notice

The contents of this document are subject to change without notice. This document may contain information on a Spansion product under development by Spansion. Spansion reserves the right to change or discontinue work on any product without notice. The information in this document is provided as is without warranty or guarantee of any kind as to its accuracy, completeness, operability, fitness for particular purpose, merchantability, non-infringement of third-party rights, or any other warranty, express, implied, or statutory. Spansion assumes no liability for any damages of any kind arising out of the use of the information in this document.

F²MC-FM3 FAMILY

32-BIT MICROCONTROLLER

ALL FM3 SERIES WITH USB

USB HOST MASS STORAGE BOOTLOADER

APPLICATION NOTE

FUJITSU

Revision History

Date	Issue
2012-03-20	MSc, V10, First Version

This document contains 18 pages.

Warranty and Disclaimer

The use of the deliverables (e.g. software, application examples, target boards, evaluation boards, starter kits, schematics, engineering samples of IC's etc.) is subject to the conditions of Fujitsu Semiconductor Europe GmbH ("FSEU") as set out in (i) the terms of the License Agreement and/or the Sale and Purchase Agreement under which agreements the Product has been delivered, (ii) the technical descriptions and (iii) all accompanying written materials.

Please note that the deliverables are intended for and must only be used for reference in an evaluation laboratory environment.

The software deliverables are provided on an as-is basis without charge and are subject to alterations. It is the user's obligation to fully test the software in its environment and to ensure proper functionality, qualification and compliance with component specifications.

Regarding hardware deliverables, FSEU warrants that they will be free from defects in material and workmanship under use and service as specified in the accompanying written materials for a duration of 1 year from the date of receipt by the customer.

Should a hardware deliverable turn out to be defect, FSEU's entire liability and the customer's exclusive remedy shall be, at FSEU´s sole discretion, either return of the purchase price and the license fee, or replacement of the hardware deliverable or parts thereof, if the deliverable is returned to FSEU in original packing and without further defects resulting from the customer's use or the transport. However, this warranty is excluded if the defect has resulted from an accident not attributable to FSEU, or abuse or misapplication attributable to the customer or any other third party not relating to FSEU or to unauthorised decompiling and/or reverse engineering and/or disassembling.

FSEU does not warrant that the deliverables do not infringe any third party intellectual property right (IPR). In the event that the deliverables infringe a third party IPR it is the sole responsibility of the customer to obtain necessary licenses to continue the usage of the deliverable.

In the event the software deliverables include the use of open source components, the provisions of the governing open source license agreement shall apply with respect to such software deliverables.

To the maximum extent permitted by applicable law FSEU disclaims all other warranties, whether express or implied, in particular, but not limited to, warranties of merchantability and fitness for a particular purpose for which the deliverables are not designated.

To the maximum extent permitted by applicable law, FSEU's liability is restricted to intention and gross negligence. FSEU is not liable for consequential damages.

Should one of the above stipulations be or become invalid and/or unenforceable, the remaining stipulations shall stay in full effect.

The contents of this document are subject to change without a prior notice, thus contact FSEU about the latest one.

Contents

1 Introduction

1.1 Overview

This user guide describes the implementation of a bootloader which uses USB mass storage support driver to read a Motorola S-Record file named *app.mhx* in root directory of the storage device and flash it as main application. Supported file systems are FAT, FAT16 and FAT32. An optional debug information is displayed via UART 0 (115200,N,1).

The software is developed for the Fujitsu FM3 series USB microcontroller. This example is designed primarily for Fujitsu microcontroller with existing evaluation boards.

Also in this application note a FAT file system is used which can be downloaded from the developer’s website:
http://elm-chan.org/fsw/ff/00index_e.html

1.2 Features

- Supports USB stick / USB card reader (tested with max. 320GB USB hard drive)
- Supports FAT, FAT16 and FAT32
- Supports reset vector check (If not valid, load bootloader)
- Starts bootloader only at power on (except no valid Reset Vector)
- Needs max. 16KByte Flash
- Debug Information via UART 0

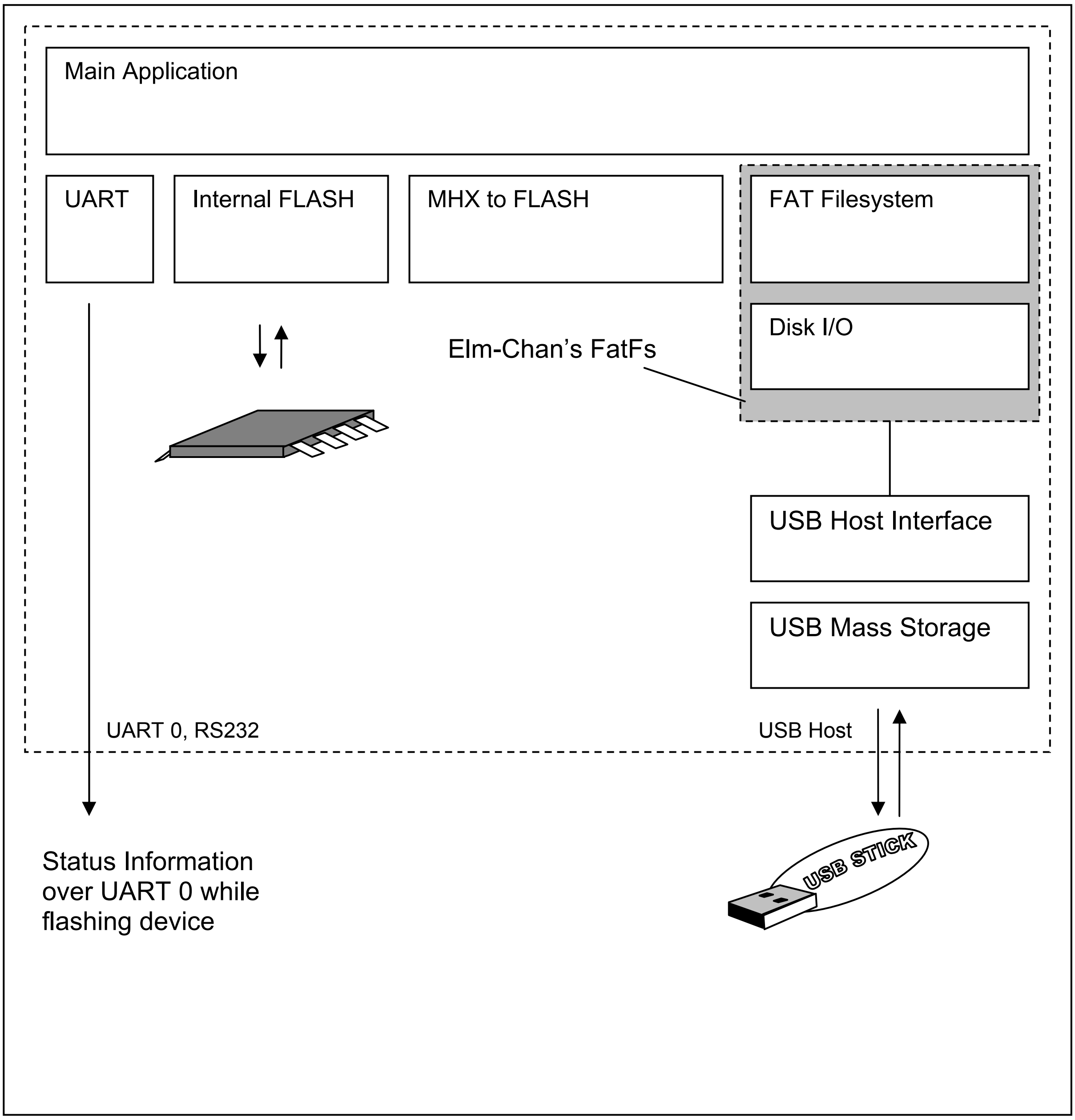
2 Overview

OVERVIEW OF THE USB/SD CARD BOOTLOADER

2.1 Project Parts

- USB Host & USB Mass Storage – Disk I/O Driver
- FatFs Module (FAT, FAT16, FAT32 driver) from Elm-Chan
- Internal Flash Routines
- Main Application (bootloader) and MHX/S-Record File Converter
- UART for Debug Information

2.1.1 Project Block Diagram



2.2 Quick Start using SK-FM3-100PMC

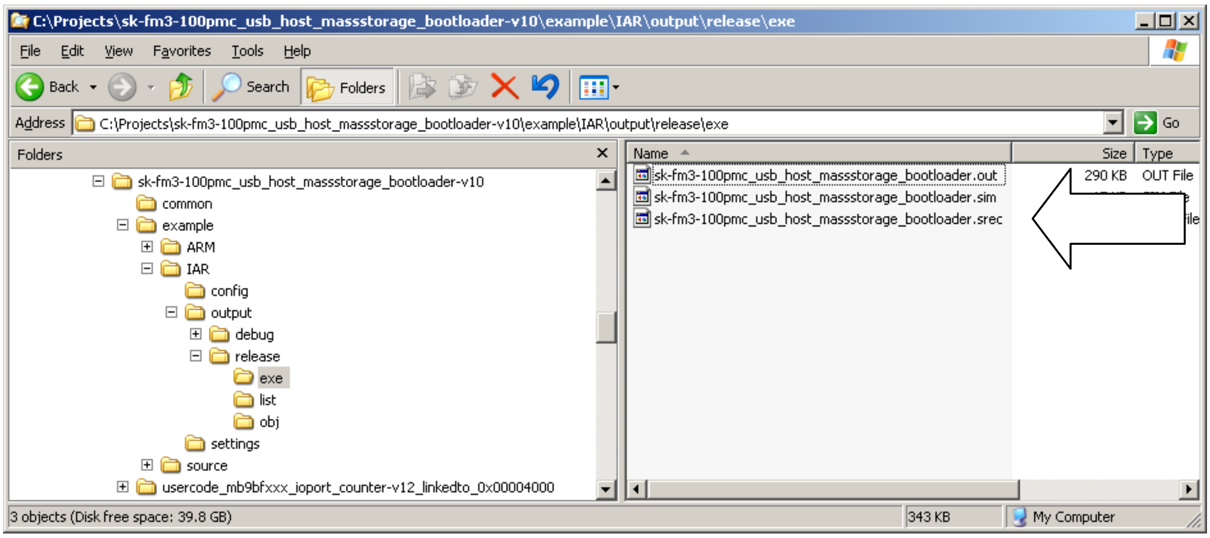
Choose sk-fm3-100pmc_usb_host_massstorage_bootloader-vXX.zip located at the evaluation board website:

http://mcu.emea.fujitsu.com/mcu_tool/detail/SK-FM3-100PMC.htm

or located in all software examples:

http://mcu.emea.fujitsu.com/mcu_product/mcu_all_software.htm

Extract the zip file and navigate to *example\IAR\output\release\exe*:



C:\Projects\sk-fm3-100pmc_usb_host_massstorage_bootloader-v10\example\IAR\output\release\exe

Address: C:\Projects\sk-fm3-100pmc_usb_host_massstorage_bootloader-v10\example\IAR\output\release\exe

```
sk-fm3-100pmc_usb_host_massstorage_bootloader-v10
    common
    example
        ARM
        IAR
            config
            output
                debug
                release
                    exe
                    list
                    obj
            settings
        source
    usercode_mb9bfxxx_ioport_counter-v12_linkedto_0x00004000
```

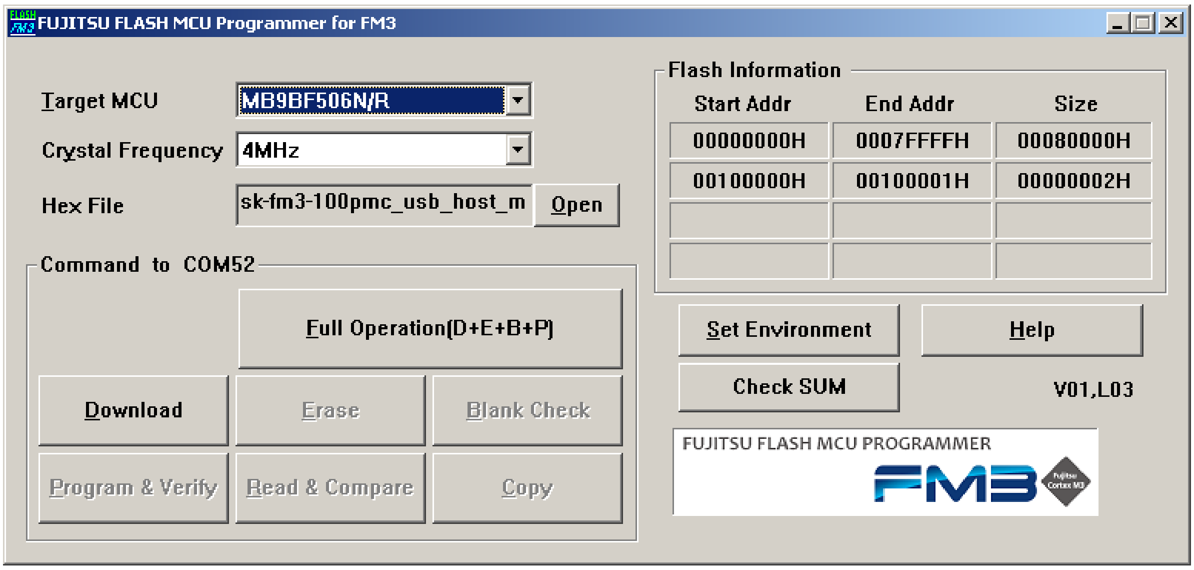
Name	Size	Type
sk-fm3-100pmc_usb_host_massstorage_bootloader.out	290 KB	OUT File
sk-fm3-100pmc_usb_host_massstorage_bootloader.sim		
sk-fm3-100pmc_usb_host_massstorage_bootloader.srec		

3 objects (Disk free space: 39.8 GB) 343 KB My Computer

Here a *.srec file should be located:

sk-fm3-100pmc_usb_host_massstorage_bootloader.srec

This file must be flashed using the Fujitsu USB Direct Programmer (via USB) or the Fujitsu FLASH MCU Programmer for FM3 (via UART). In this case the Fujitsu FLASH MCU Programmer for FM3 will be used.



FUJITSU FLASH MCU Programmer for FM3

Target MCU: MB9BF506N/R
Crystal Frequency: 4MHz
Hex File: sk-fm3-100pmc_usb_host_m

Flash Information

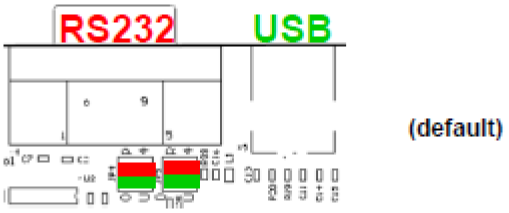
Start Addr	End Addr	Size
00000000H	0007FFFFH	00080000H
00100000H	00100001H	00000002H

Command to COM52

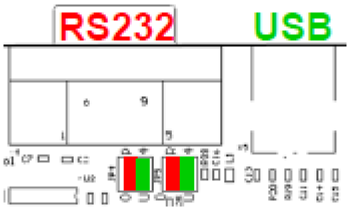
V01.L03

Via “Set Environment” the COM port for programming the evaluation board can be chosen. Programming via UART only works with UART 0. There are two different options: Using RS232 or USB to UART converter for use with UART 0 / 4.

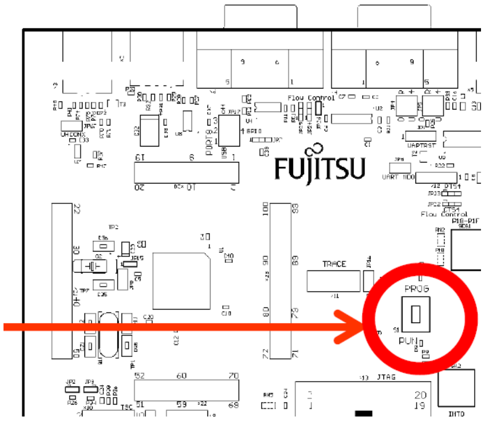
- UART0 = USB-connector (X5), UART4 = Sub-D9 (X4) (default)
  - Setting of Jumper JP4 and JP5: U-0 / R-4



- UART0 = Sub-D9 (X4), UART4 = USB-connector (X5)
  - Setting of Jumper JP4 and JP5: U-4 / R-0



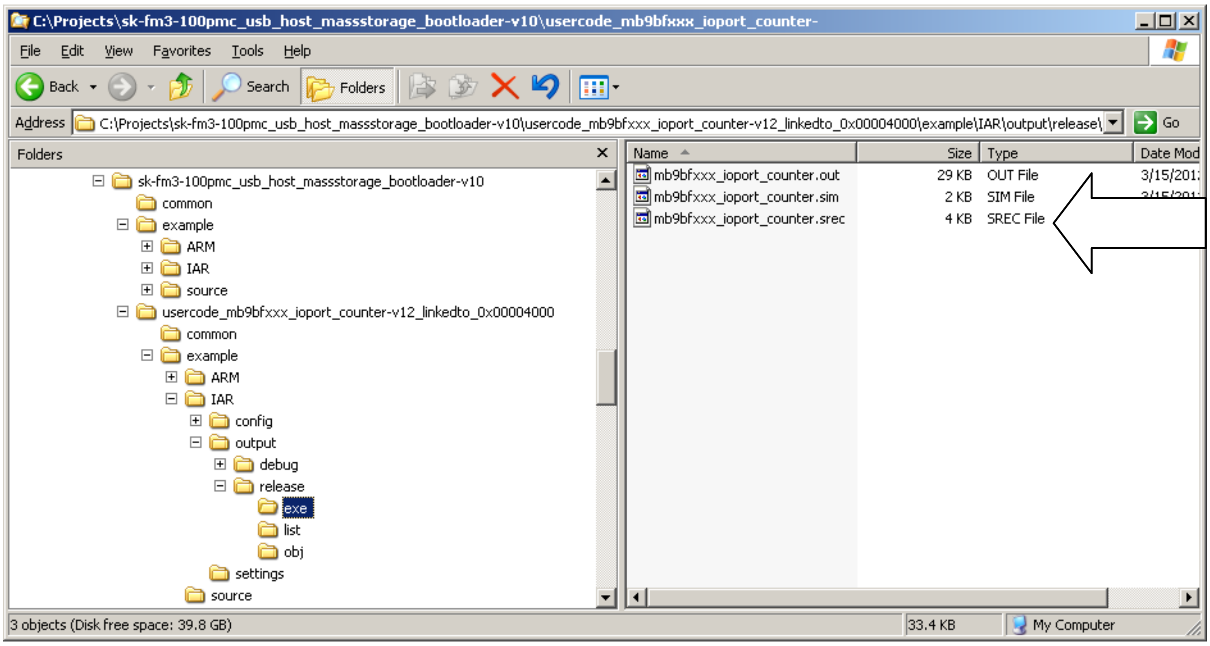
Switch the board to programming mode and press “Full Operation” and following the instructions.



After programming is completed, switch back to run mode and press reset on the evaluation board. The bootloader starts only at power on or if no user code application exists. Because no user application code exists, the bootloader will start. Plug in an USB stick into the PC and copy an application code named to *app.mhx* to root directory. Now this USB stick can be used with the evaluation board. To re-flash later an application, the USB stick must be plugged in before powering the evaluation board.

An example linked to `0x0000_4000` can be found in the example folder:

usercode_mb9bfxxx_ioport_counter-v12_linkedto_0x00004000\example\IAR\output\release\exe



C:\Projects\sk-fm3-100pmc_usb_host_massstorage_bootloader-v10\usercode_mb9bfxxx_ioport_counter-

Address C:\Projects\sk-fm3-100pmc_usb_host_massstorage_bootloader-v10\usercode_mb9bfxxx_ioport_counter-v12_linkedto_0x00004000\example\IAR\output\release\

Name	Size	Type	Date Mod
mb9bfxxx_ioport_counter.out	29 KB	OUT File	3/15/201:
mb9bfxxx_ioport_counter.sim	2 KB	SIM File	3/15/201:
mb9bfxxx_ioport_counter.srec	4 KB	SREC File	

3 objects (Disk free space: 39.8 GB) 33.4 KB My Computer

File Edit View Favorites Tools Help
Back
Search
Folders
Go
Folders
sk-fm3-100pmc_usb_host_massstorage_bootloader-v10
common
example
ARM
IAR
source
usercode_mb9bfxxx_ioport_counter-v12_linkedto_0x00004000
common
example
ARM
IAR
config
output
debug
release
exe
list
obj
settings
source

mb9bfxxx_ioport_counter.srec must be renamed to *app.mhx* and must be placed at the usb stick. After plugging in the USB stick into the evaluation board, the bootloader can start to flash the MCU.

For change linker settings in own applications, refer chapter 3.6.1.

The seven segment display shows following status:

01: Bootoader started

02: USB Stick detected

03: User Code found

04: Start erasing flash

05: Start flashing

00: Operation completed successfully, remove USB Stick

11: Error while erasing, System halt

12: Error CRC, System halt

If the seven segment reaches “00” the program was flashed and the USB stick can be removed to restart in user application code.

3 Bootloader Operation

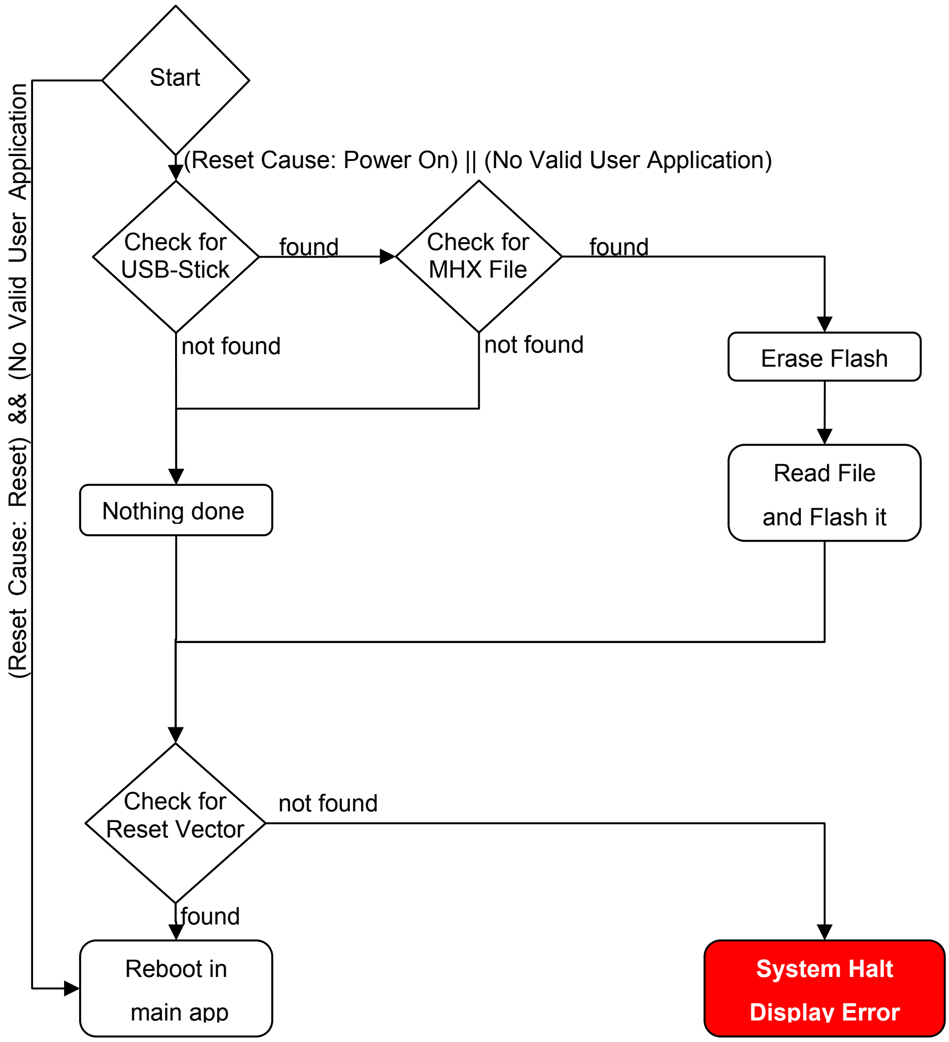
Bootloader Process

3.1 Bootloader operation flow-chart

The bootloader starts with some initialisations: Before it handles its own start code, it checks for a valid user application code and checks the power on state. If no valid user application code exists within the microcontrollers flash memory, the microcontroller would execute not wanted undefined code. So if it does not exist, the bootloader will be started instead of the main application. If the "Power On" state was a reset, the main application will be started instead no valid boot vector exists. If the reset cause was power on, the bootloader starts.

After USB-initialisation the bootloader enumerates the USB stick - if it is available - and tries to mount it. After a timeout of about 500ms the function gives up to find a valid USB mass storage. If no USB stick was found, the bootloader restarts in user application code. If a USB stick was found, it will be checked whether a file *app.mhx* exists in root directory. If the file was found, the bootloader begins to erase all Flash sections except the bootloader Flash sections and flashes the main application from *app.mhx*.

Note: The Power-On Flag is reset by reading it by the bootloader. It is not available for the main application anymore.



3.2 Memory Map

As described in the *Technical Reference Manual* of ARM for Cortex-M3 core, ARM Cortex M3 microcontrollers have its flash memory starting at `0x0000_0000`, RAM memory starting at `0x2000_0000` and so on.

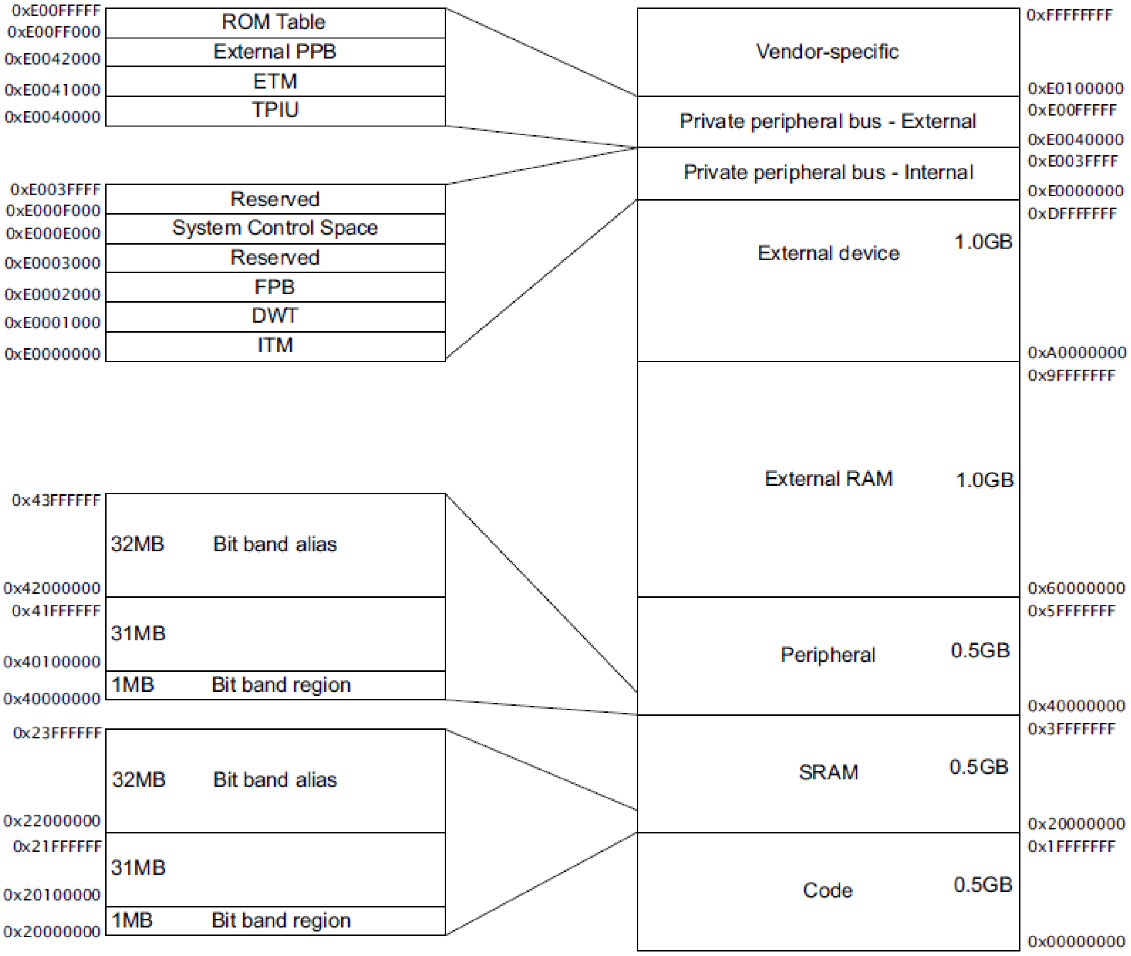
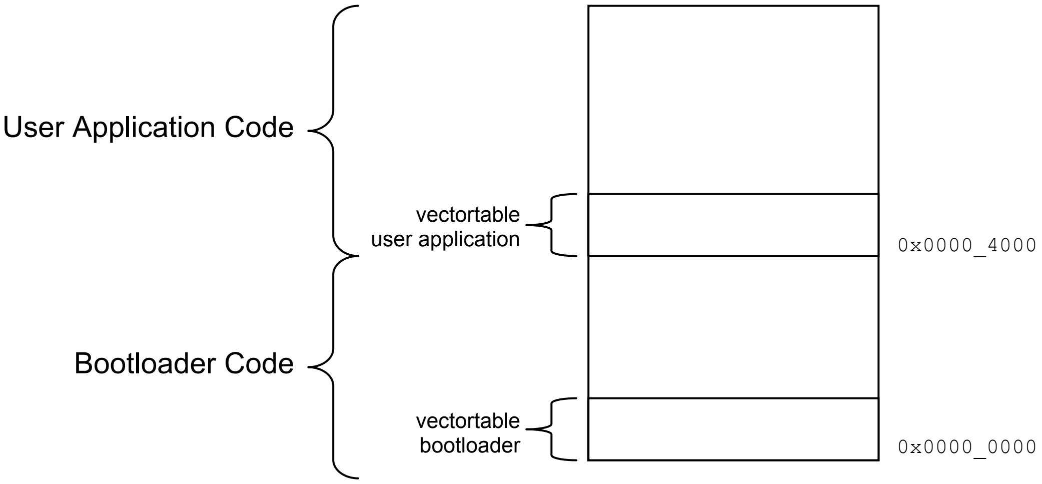


Figure 3-1: Processor memory map (ARM Cortex-M3 Technical Reference Manual r2p0, 4-1)

As defined for ARM Cortex-M3 MCUs, the vector table starts with the stack pointer followed by the reset vector and so on. In case of using a bootloader, the vector table of the bootloader is fixed to `0x0000_0000` and the user code must be linked to upper areas of the flash memories. In case of Fujitsu bootloader, the user code starts `0x0000_4000`.



3.3 Jump into user application

To start the user application code, the stack pointer and vector table must be updated. Following procedure is called, to start the user application code, linked to `0x0000_4000`:

```
#define USER_FLASH_START 0x00004000

int32_t main(void)
{
    BootloaderAPI_JumpBoot(USER_FLASH_START);
}

/**
 ******************************************************************************
 ** \brief  Jump to user code application
 **
 ** \param  u32Address Address of user code
 ** \return none
 ******************************************************************************/
#ifdef __ICCARM__
  void BootloaderAPI_JumpBoot(uint32_t u32Address)
  {
     __asm("LDR SP, [R0]");        //Load new stack pointer address
     __asm("LDR PC, [R0, #4]");    //Load new program counter address
  }
#elif  __CC_ARM
  __asm void BootloaderAPI_JumpBoot(uint32_t u32Address)
  {
     LDR SP, [R0]                   ;Load new stack pointer address
     LDR PC, [R0, #4]               ;Load new program counter address
  }
#else
  #error Please check compiler and linker settings
#endif

/**
 ******************************************************************************
 ** \brief  Execute main application
 **
 ** \param  none
 ** \return none
 ******************************************************************************/
void BootloaderAPI_ExecuteUserApplication(void)
{
    //Change the Vector Table to the USER_FLASH_START
    SCB->VTOR = USER_FLASH_START & 0x1FFFFF80;
    BootloaderAPI_JumpBoot(USER_FLASH_START);
}
```

The code relocates the vector table, loads the new stack pointer (starting at the user code vector table) and sets the program counter to the user code application start.

3.4 Bootloader start condition

Fujitsu FM3 series MCUs have a reset cause register. The reset cause register recognizes which kind of reset was executed. The bootloader starts only at power on. All other resets (software reset, hardware reset, watchdog reset, etc.) will execute the user code application.

```
/* If reset cause was not power on, start user application, if it is valid */
if ((bFM3_CRG_RST_STR_PONR == 0) && (BootloaderAPI_UserCodeValid() == TRUE))
{
    BootloaderAPI_ExecuteUserApplication();
}
```

3.5 MHX Format Conversion

3.5.1 S-Record Structure

The MHX Format is a Motorola-S-Record file, which has the ability to flash only used sections and leave sections, which are not used untouched. Each line represents a record. There are 10 different types of records (S0 – S9); however the mhx converter in the bootloader handles only the S1..3 type. The line is structured as followed:

3.5.1.1 Example: MHX Record

Data in S-Record:
`S209F8B0788C7625096640`

Start Code	Record Type	Byte Count	Address	Data	CRC
S	2	09	F8B078	8C76250966	40

Start-Code: Is every record line the first character and contains every time an “S”

Record-Type: Record Type (0-9), only 1..3 is supported

Byte-Count: Number of Bytes of record (Address + Data + CRC)

Address: Flash-Address (3 Bytes for S2 record)

Data: Data to write at the specified address

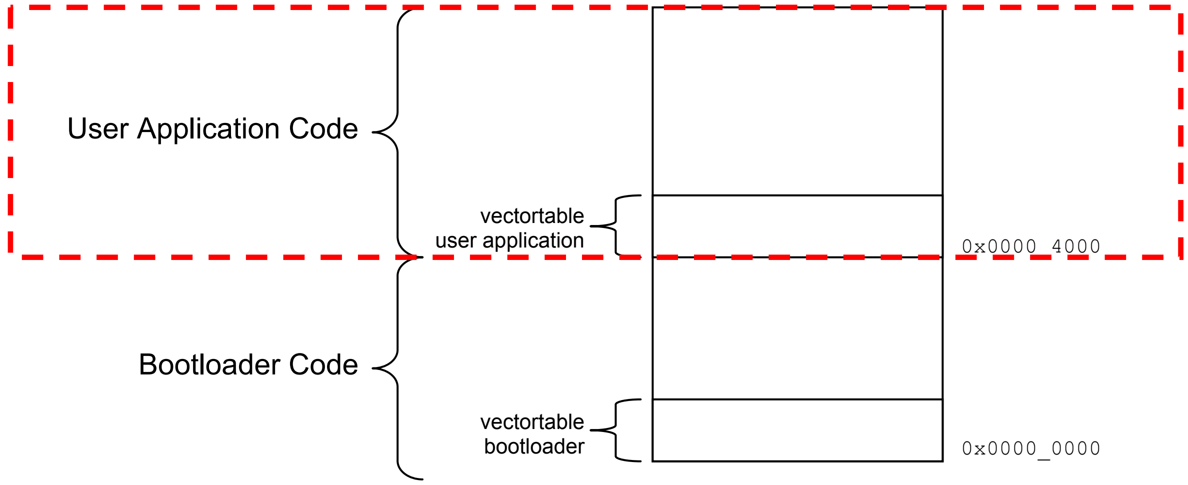
CRC: Checksum

3.6 Linker settings

For bootloader usage, linker settings for the bootloader and the user application must be changed.

3.6.1 Linker Settings User Code

For using the user application code with the bootloader in parallel, the user application must start from 0x0000_4000 instead of 0x0000_0000.

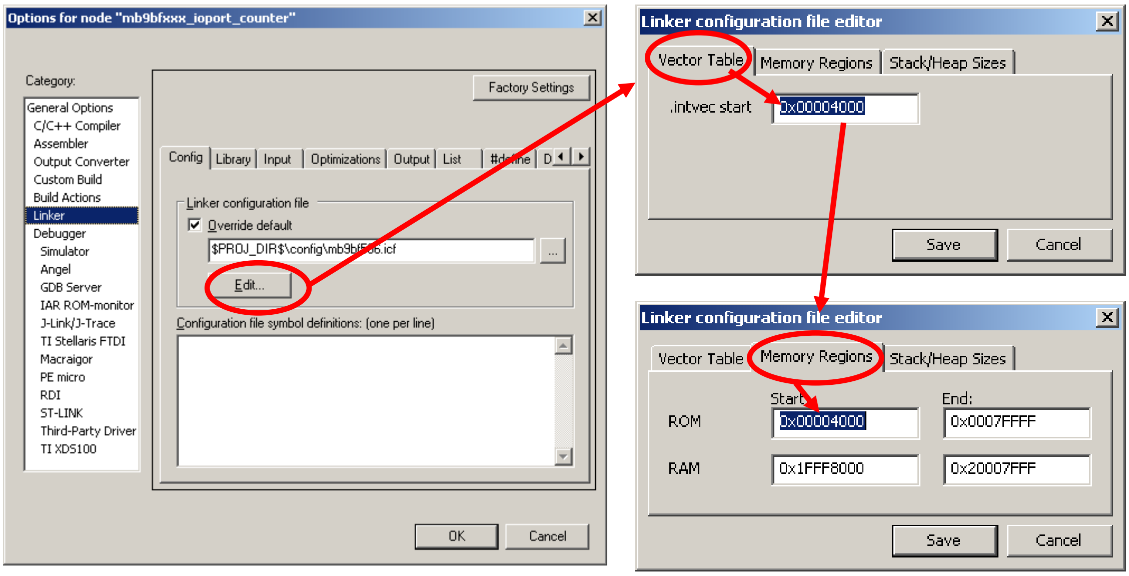


3.6.1.1 IAR user code linker settings

The following lines were changed in the standard FM3 template linker files (*.icf):

```
/*-Specials-*/
define symbol __ICFEDIT_intvec_start__ = 0x00004000;
/*-Memory Regions-*/
define symbol __ICFEDIT_region_ROM_start__ = 0x00004000;
```

This can be done also via Project -> Options -> Linker. Here .intvec start and ROM start can be specified as well.



Linker configuration file
☑ Override default
$PROJ_DIR$\config\mb9bf506.icf
Edit...
Configuration file symbol definitions: (one per line)

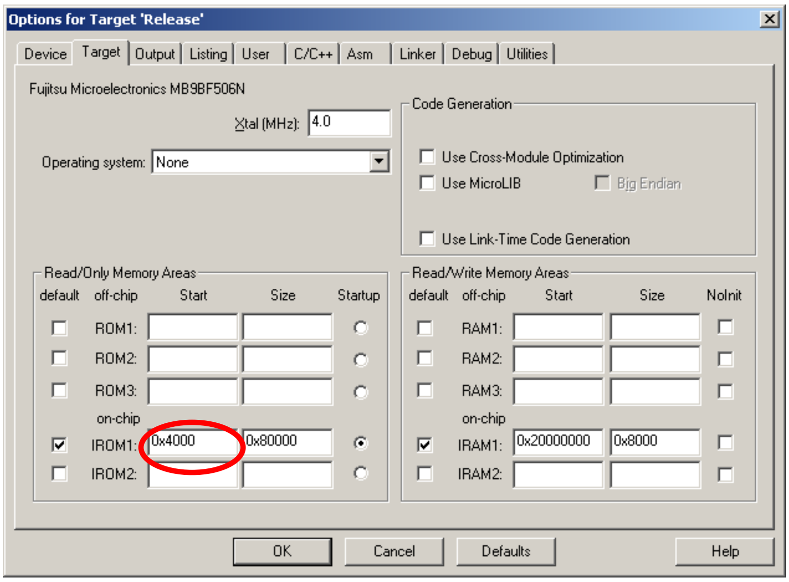
.intvec start 0x00004000

	Start:	End:
ROM	0x00004000	0x0007FFFF
RAM	0x1FFF8000	0x20007FFF

3.6.1.2 Keil user code linker settings

Project -> Options for Target 'Release' -> Target:

IROM1 - Start must be changed from 0x0 to 0x00004000



Fujitsu Microelectronics MB9BF506N

Xtal (MHz): 4.0

Operating system: None

Code Generation

□ Use Cross-Module Optimization

□ Use MicroLIB □ Big Endian

□ Use Link-Time Code Generation

Read/Only Memory Areas

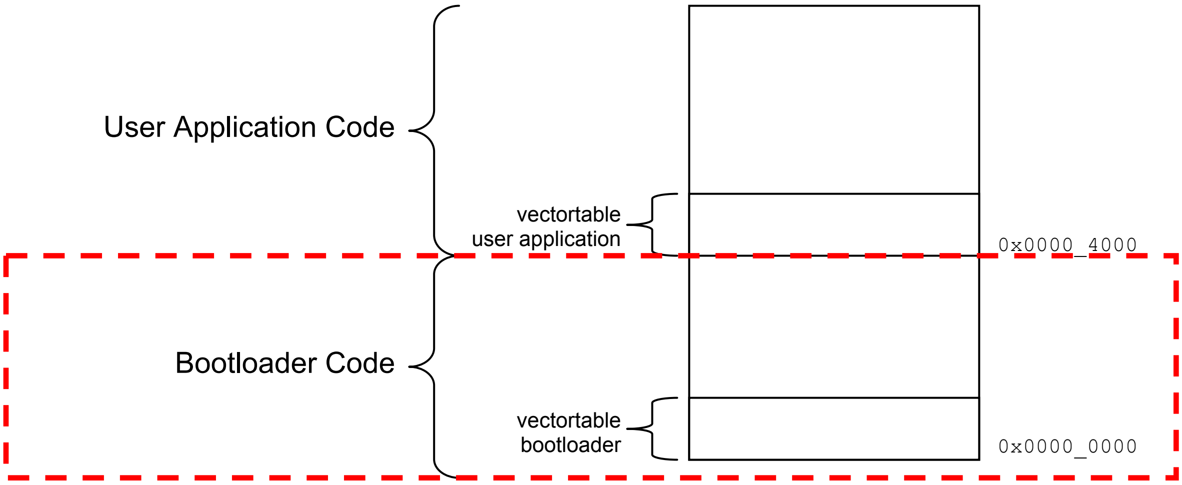
default	off-chip	Start	Size	Startup
□	ROM1:			○
□	ROM2:			○
□	ROM3:			○
	on-chip			
☑	IROM1:	0x4000	0x80000	●
□	IROM2:			○

Read/Write Memory Areas

default	off-chip	Start	Size	NoInit
□	RAM1:			□
□	RAM2:			□
□	RAM3:			□
	on-chip			
☑	IRAM1:	0x20000000	0x8000	□
□	IRAM2:			□

3.6.2 Linker Settings Bootloader

For writing flash, flash routines must be copied into RAM. This can be configured in the linker files and is normally done by the IDE startup code. To be on the safe side, the ROM (flash) area should be set to use only `0x0000_0000` to `0x0000_3FFF`.



3.6.2.1 IAR bootloader linker settings

For using RAM code for Flash erase/programming routines copied to RAM automatically by start-up and executed later in RAM, the following lines were added to the standard FM3 template linker files (*.icf):

```
define symbol __RAM_func_start__   = 0x20000000;
define symbol __RAM_func_end__     = 0x20007FFF;
define   region   RAM_func_region   =        mem:[from   __RAM_func_start__   to
__RAM_func_end__];

define block RamCode {section .flash_ram_code};
place in RAM_func_region  { block RamCode };
```

3.6.2.2 Keil bootloader linker settings

For using RAM code for Flash erase/programming routines copied to RAM automatically by start-up and executed later in RAM, the following adjustment was done:

Options for flash.c -> Properties -> Memory Assignment -> Code / Const:

IRAM (0x2000000-0x20007FFF)

Read application note mcu-an-300401-e-fm3_flash_programming for further details.

4 Appendix

4.1 Bootloader Download

Please download the newest version from
http://mcu.emea.fujitsu.com/mcu_product/mcu_all_software.htm

4.2 FatFs Module (FAT, FAT16, FAT32 driver)

For more information refer visit the developer’s website:
http://elm-chan.org/fsw/ff/00index_e.html