XMC4700 / 4800 USB Mass Storage Device Bootloader and In Application Programming Example

Getting Started V1.0

1. Overview and Requirements
2. Setup
3. Short Overview – Boot modes
4. Implementing the bootloader executable
5. Implementing the application executable
6. How to test

Copyright © Infineon Technologies AG 2016. All rights reserved.

1 Overview and Requirements

2 Setup

3 Short Overview – Boot modes

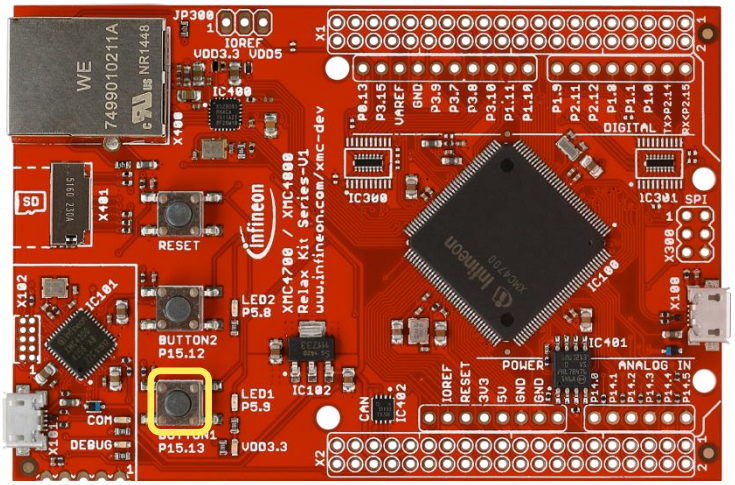
4 Implementing the bootloader executable

5 Implementing the application executable

6 How to test

Copyright © Infineon Technologies AG 2016. All rights reserved.

Overview



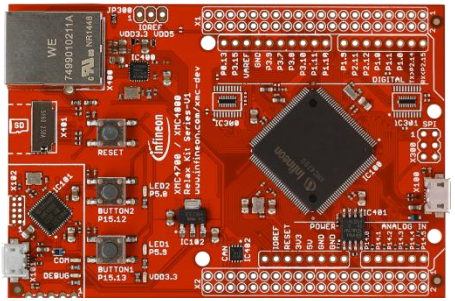
This example demonstrates the implementation of a bootloader for „in application programming" via USB Mass Storage Device. It is based on the LUFA implementation which is ported to XMC4000 family and provided within this example.

Within this documentation you will learn to configure XMC4700 having two independent flash images installed. The bootloader image implements the USB mass storage device. The bootloader executable is then used to update the application image. For this you just need to drag & drop the binary image of the application executable to the USB mass storage device on host side.

By default, after every reset XMC4700 is executing „application" image. Pushing BUTTON1 during startup forces XMC4700 to execute the „bootloader" image.

Copyright © Infineon Technologies AG 2016. All rights reserved.

Requirements – hardware



XMC 4700 Relax Kit

or optional

XMC 4800 Relax EtherCAT Kit



Windows Laptop installed

- DAVE v4 (Version4.1.4 or higher)



2x Micro USB Cable (Debugger and USB mass storage connector)

Copyright © Infineon Technologies AG 2016. All rights reserved.

Requirements - free software downloads



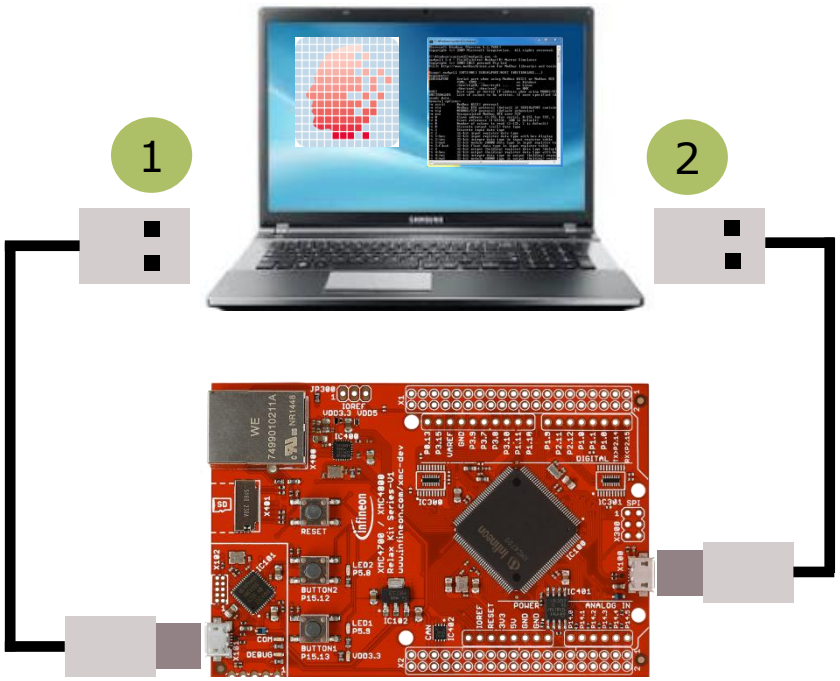
DAVE™ (v4.1.4 or higher)
Link: Download DAVE (Version 4)

Copyright © Infineon Technologies AG 2016. All rights reserved.

1. Overview and Requirements
2. Setup
3. Short Overview – Boot modes
4. Implementing the bootloader executable
5. Implementing the application executable
6. How to test

Copyright © Infineon Technologies AG 2016. All rights reserved.

Setup – Hardware



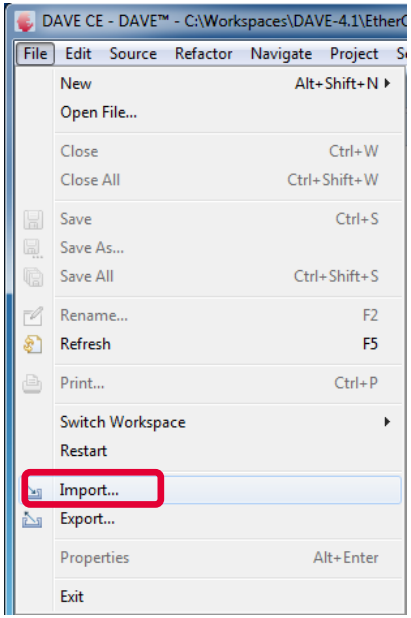
1. Micro USB cable connected to X101 connector (debugger)

2. Micro USB cable connected to X100 connector (XMC4700 USB mass storage)

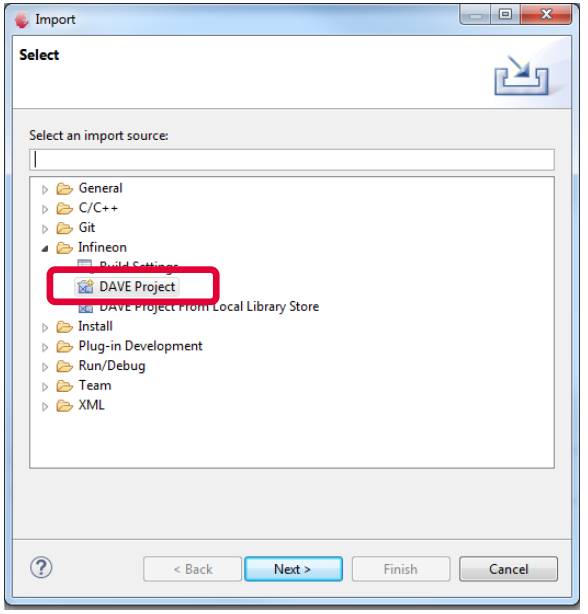
Copyright © Infineon Technologies AG 2016. All rights reserved.

Setup – Import example project into DAVE™

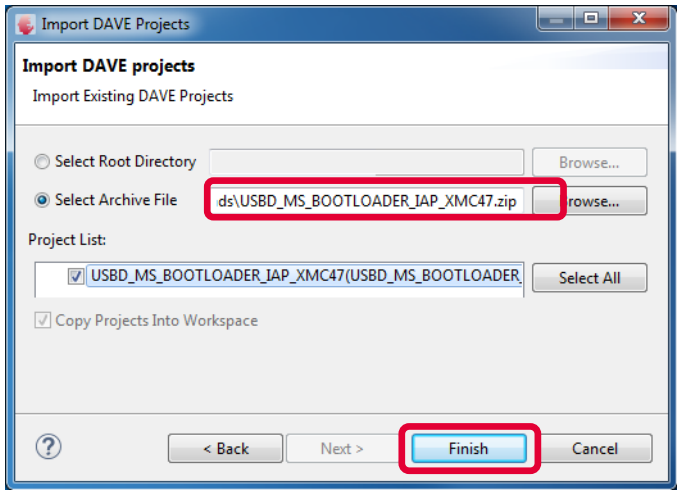
1



2



3



Copyright © Infineon Technologies AG 2016. All rights reserved.

1 Overview and Requirements

2 Setup

3 Short Overview – Boot modes

4 Implementing the bootloader executable

5 Implementing the application executable

6 How to test

Copyright © Infineon Technologies AG 2016. All rights reserved.

Boot modes- overview

Table 27-2 System reset boot modes

SWCON[3:0]	Boot mode
0000_B	Normal
0001_B	ASC BSL
0010_B	BMI
0011_B	CAN BSL
0100_B	PSRAM boot
1000_B	ABM-0
1100_B	ABM-1
1110_B	Fallback ABM

XMC4700 supports different boot modes.

1. Driving the boot mode pins (JTAG TCK TMS) with appropriate logic levels and issuing a power on reset, these boot modes can be selected: Normal, ASC BSL (UART bootstrap), CAN BSL (CAN bootstrap), BMI

2. By preparing a header information and issuing a system reset (via software) these boot modes can be used in addition. These boot modes cannot be entered with power on reset:
Alternative Boot modes (ABM-0/ABM-1), PSRAM boot, FallbackABM

Copyright © Infineon Technologies AG 2016. All rights reserved.

Boot modes – normal and alternative boot mode 0

Table 27-2 System reset boot modes

SWCON[3:0]	Boot mode
0000_B	Normal
0001_B	ASC BSL
0010_B	BMI
0011_B	CAN BSL
0100_B	PSRAM boot
1000_B	ABM-0
1100_B	ABM-1
1110_B	Fallback ABM

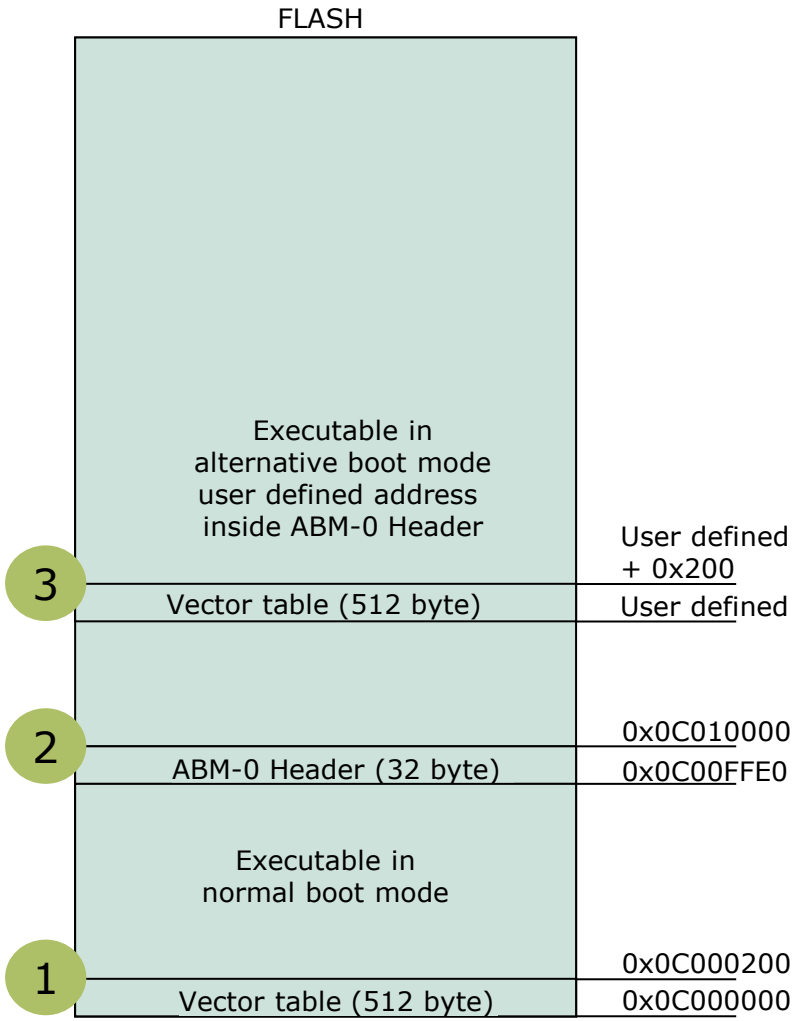
For the example these boot modes are used:

Normal boot mode (after power on reset)
An application located at start of the flash is executed.

Alternative boot mode 0 (after system reset)
An application located at a user defined address inside flash is executed. The address is defined inside the „ABM Header".
ABM Header must be located at last 32bytes (0x0C00FEE0) of the first 64kB physical sector inside flash.

Copyright © Infineon Technologies AG 2016. All rights reserved.

Boot modes - memory outline

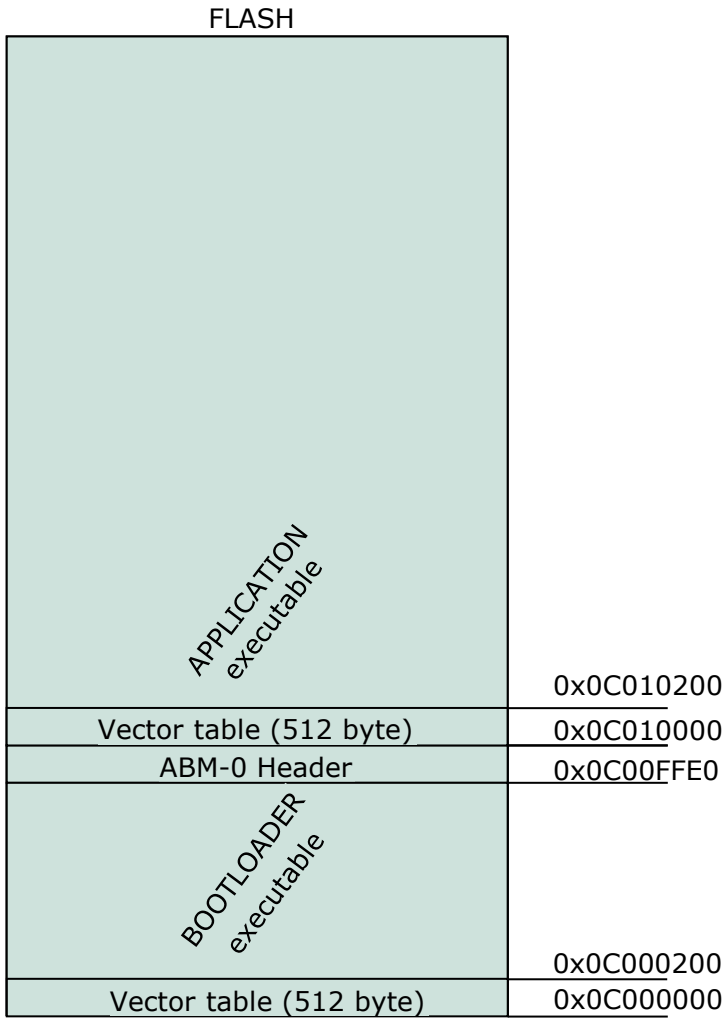


1. In **normal boot mode** execution starts at address 0x0C000000
2. **ABM-0 header** is located at fixed address 0x0C00FFE0
3. In **ABM-0 boot mode** execution starts at address defined by user inside ABM-0 header

ATTENTION: Typically the user defined start address inside ABM0-header is set to 0x0C010000. With this you avoid an unnecessary gap in your the memory outline.

Copyright © Infineon Technologies AG 2016. All rights reserved.

Boot modes – typical memory outline of application and bootloader executables



The application executable is located at flash address 0x0C010000:

- Application is never executed directly after power on reset

- Application execution can only be triggered by a system reset e.g. from bootloader executable

The bootloader executable is located at flash address 0x0C000000:

- Bootloader execution is triggered by power on reset. Bootloader starts the application executable or proceeds with bootloader functionality.

Copyright © Infineon Technologies AG 2016. All rights reserved.

1. Overview and Requirements
2. Setup
3. Short Overview – Boot modes
4. Implementing the bootloader executable
5. Implementing the application executable
6. How to test

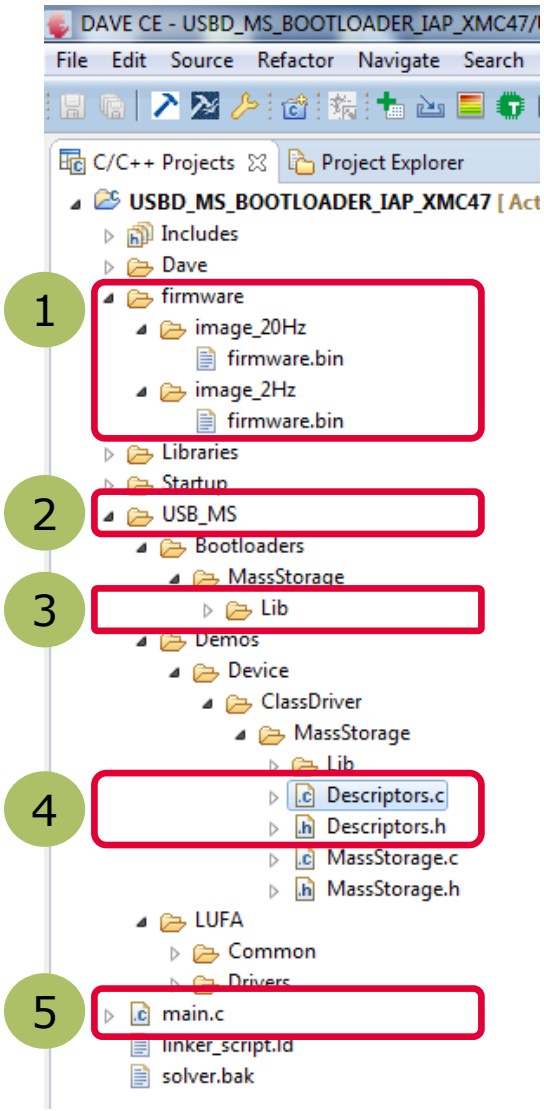
Copyright © Infineon Technologies AG 2016. All rights reserved.

Bootloader – Functional overview
USB MS device: In Application programming

- After power on reset always bootloader executable is started first
- Bootloader executable decides if application executable shall be started by checking state of BUTTON1 after power on reset
- If BUTTON1 is not pushed during startup, bootloader proceeds with system reset to start the application executable
- If BUTTON1 is pushed during startup, USB mass storage device and virtual file system is initialized
- When the binary file is copied by host to the USB MS device, the binary content is programmed to flash address 0x0C010000 of application executable
- Once the host disconnects from USB mass storage device, bootloader proceeds with system reset to start the application executable

Copyright © Infineon Technologies AG 2016. All rights reserved.

Bootloader – DAVE™ project structure

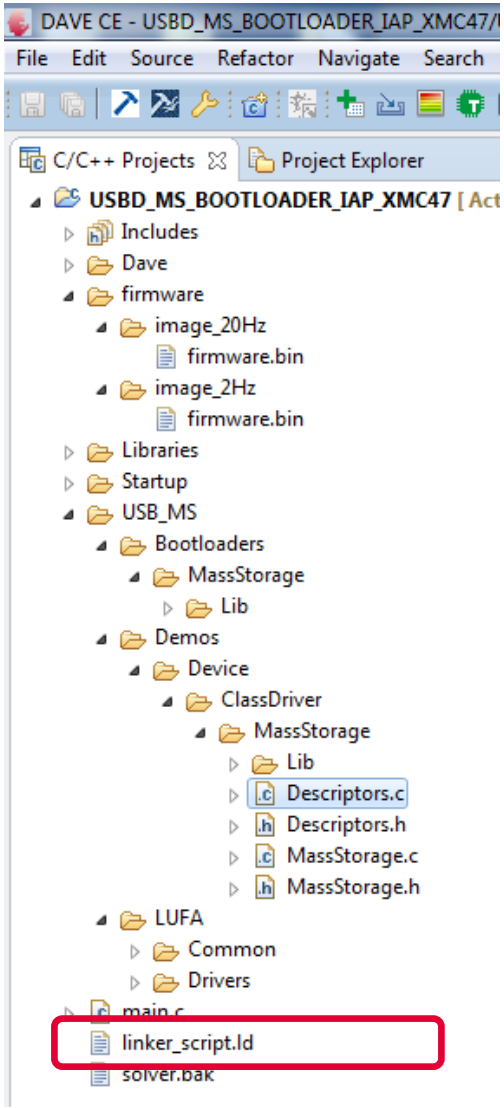


The folder structure of the example project:
Beside standard DAVE project items you will find these USB MS related items:

1. Binary application images used for testing of this example
2. USB mass storage device driver implementation based on LUFA device class drivers
3. Virtual FAT filesystem and Flash programming implementation
4. USB Mass Storage device description files
5. The main file implementing the main application

Copyright © Infineon Technologies AG 2016. All rights reserved.

Bootloader – Limit bootloader flash size inside linker file



The bootloader executable is limited to the first 64KB inside flash.

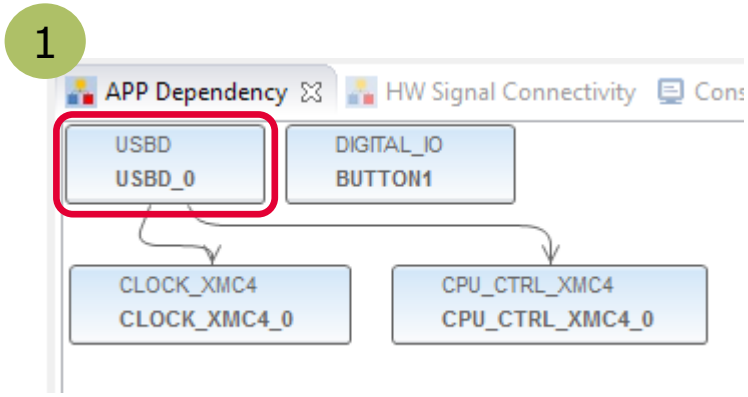
To make build aware of this limitation, the linker file has to be adapted:

linker_script.ld

```
60 MEMORY
61 {
62     FLASH_1_cached(RX) : ORIGIN = 0x08000000, LENGTH = 0x00010000
63     FLASH_1_uncached(RX) : ORIGIN = 0x0C000000, LENGTH = 0x00010000
64     PSRAM_1(!RX) : ORIGIN = 0x1FFE8000, LENGTH = 0x18000
65     DSRAM_1_system(!RX) : ORIGIN = 0x20000000, LENGTH = 0x20000
66     DSRAM_2_comm(!RX) : ORIGIN = 0x20020000, LENGTH = 0x20000
67     SRAM_combined(!RX) : ORIGIN = 0x1FFE8000, LENGTH = 0x00058000
68 }
```


Copyright © Infineon Technologies AG 2016. All rights reserved.

Bootloader – DAVE™ APP structure



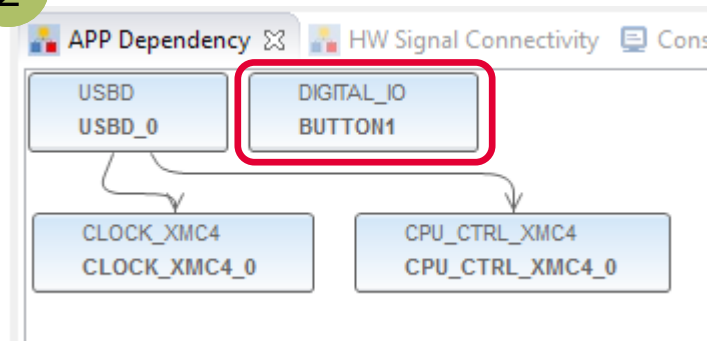
1 USBD APP is used to implement USB LUFA stack and access the USB low level drivers. CLOCK_XMC4 and CPU_CTRL_XMC4 APPs are automatically included by DAVE™ when the dependencies are resolved.

You do not need to assign pins to the USBD APP, because inside hardware dedicated pins are assigned without secondary funtionality.

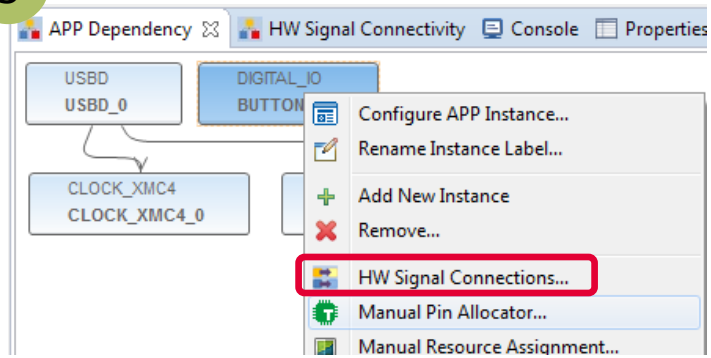
Copyright © Infineon Technologies AG 2016. All rights reserved.

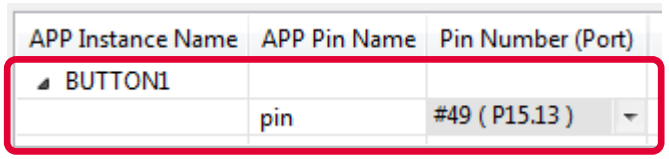
Bootloader – DAVE™ APP structure

2



3





2 DIGITAL_IO APPs are used to access the input/output ports. Here only one port is used to read the state of BUTTON1 of XMC 4700 Relax Kit.

3 Right click on DIGITAL_IO APP. From the context menu select „Manual Pin Allocator" to open the pin allocation. Assign BUTTON1 input port of the XMC4700 Relax Kit to the DIGITAL_IO APP.

Copyright © Infineon Technologies AG 2016. All rights reserved.

Bootloader – implementing main()

main.c

```
120 int32_t main(void)
121 {
122   /*Initialize DAVE */
123   if(DAVE_Init() == DAVE_STATUS_FAILURE)
124   {
125     /* Placeholder for error handler code.*/
126     XMC_DEBUG(("DAVE APPs initialization failed\n"));
127     while(1U)
128     {
129       /* do nothing */
130     }
131   }
132
133   if (DIGITAL_IO_GetInput(&BUTTON1))
134   {
135     /* Button1 is not pushed, restart the device
136      * in upper flash code */
137
138     /* Restart device in ABM0 */
139     BL_FlashABM0_Restart();
140   }
141   else
142   {
143     /* Button1 is pushed, device is operating as a USB MS device
144      * to flash upper flash code
145
146      * Initialize the virtual filesystem */
147     VFAT_Init();
148
149     /* Initialize USB MS device */
150     USB_Init();
151
152     /* Process of the USB MS device stack. */
153     for( ;; )
154     {
155       MS_Device_USBTask(&Disk_MS_Interface);
156     }
157   }
158   return 0;
159 }
160
161 void EVENT_USB_Device_Disconnect() {
162   /* When the usb is disconnected, restart the device
163    * in upper flash code */
164   BL_FlashABM0_Restart();
165 }
```

1. Initialize DAVE™ system including all APPs
2. If BUTTON1 is not pushed during power on reset, restart device in alternative boot mode 0
3. If BUTTON1 is pushed during power on reset, initialize virtual file system and USB mass storage device. Process USB class driver and device stack.
4. If USB mass storage device is disconnected, restart device in alternative boot mode 0

Copyright © Infineon Technologies AG 2016. All rights reserved.

Bootloader – triggering systemreset and alternative boot mode 0 execution

main.c

```
void BL_FlashABM0_Restart(void)
{
  /* Restart in alternative bootmode 0 */
  /* Clear the reset source field */
  XMC_SCU_RESET_ClearDeviceResetReason();
  /* Set ABM0 as boot mode */
  XMC_SCU_SetBootMode(XMC_SCU_BOOTMODE_ABM0);
  /* Trigger system reset */
  PPB->AIRCR = 1 << PPB_AIRCR_SYSRESETREQ_Pos |
                    0x5FA<<PPB_AIRCR_VECTKEY_Pos |
                    0x1 << PPB_AIRCR_PRIGROUP_Pos;
}
```

To restart in alternative boot mode 0 the following actions are needed:

Clear the reset source of the last reset

Set boot mode to alternative boot mode 0

Issue system reset (software reset)

Copyright © Infineon Technologies AG 2016. All rights reserved.

Bootloader – defining ABM0-header

main.c

```
#define ABM_HEADER_MAGIC_KEY 0xA5C3E10F

/* \brief Alternative Boot Mode structure */
typedef struct ABM_Header {
    uint32_t MagicKey; /**< Magic key. Always 0xA5C3E10F */
    uint32_t StartAddress; /**< Start address of the programm to load */
    uint32_t Length; /**< Length of the programm to load. */
    uint32_t ApplicationCRC32; /**< CRC32 Sum of the complete application code */
    uint32_t HeaderCRC32; /**< CRC32 Sum of the four fields before */
} ABM_Header_t;

/*****************************************************
 * LOCAL DATA
 *****************************************************/
static const ABM_Header_t __attribute__((section(".flash_abm")))
ABM0_Header = {
    .MagicKey = ABM_HEADER_MAGIC_KEY,
    .StartAddress =  0x08010000,  /* Start Flash Physical Sector 1 */
    .Length = 0xFFFFFFFF,
    .ApplicationCRC32 = 0xFFFFFFFF,
    .HeaderCRC32 =  0xE176A4E6
};
```

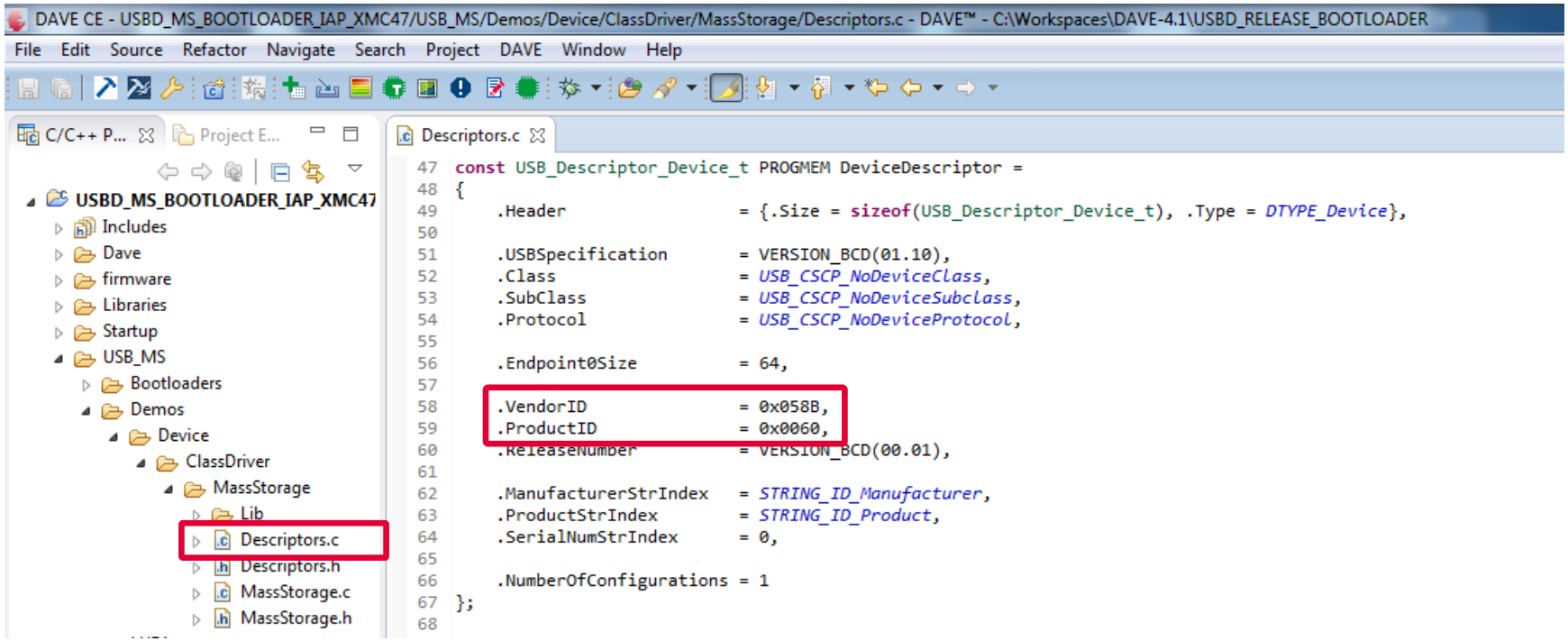
linker_script.ld

```
        *(.ARM.extab* .gnu.linkonce.armextab.*)
    } > FLASH_1_cached AT > FLASH_1_uncached

    __exidx_start = .;
    .ARM.exidx :
    {
        *(.ARM.exidx* .gnu.linkonce.armexidx.*)
    } > FLASH_1_cached AT > FLASH_1_uncached
    __exidx_end = .;

    /* http://mcuoneclipse.com/2012/11/01/defining-variables */
    .abm ABSOLUTE(0x0800FFE0): AT(0x0800FFE0 | 0x04000000)
    {
       KEEP(*(.flash_abm))
    } > FLASH_1_cached
```

1. Define magic key of ABM header
2. Define ABM header structure
3. Define content of ABM header and location inside flash. Beside magic key and CRC the start adress of the flash is the only mandatory information needed.
4. Fix flash address of ABM header inside linker script to last 32byte of the first 64KB: 0x0C00FFE0

Copyright © Infineon Technologies AG 2016. All rights reserved.

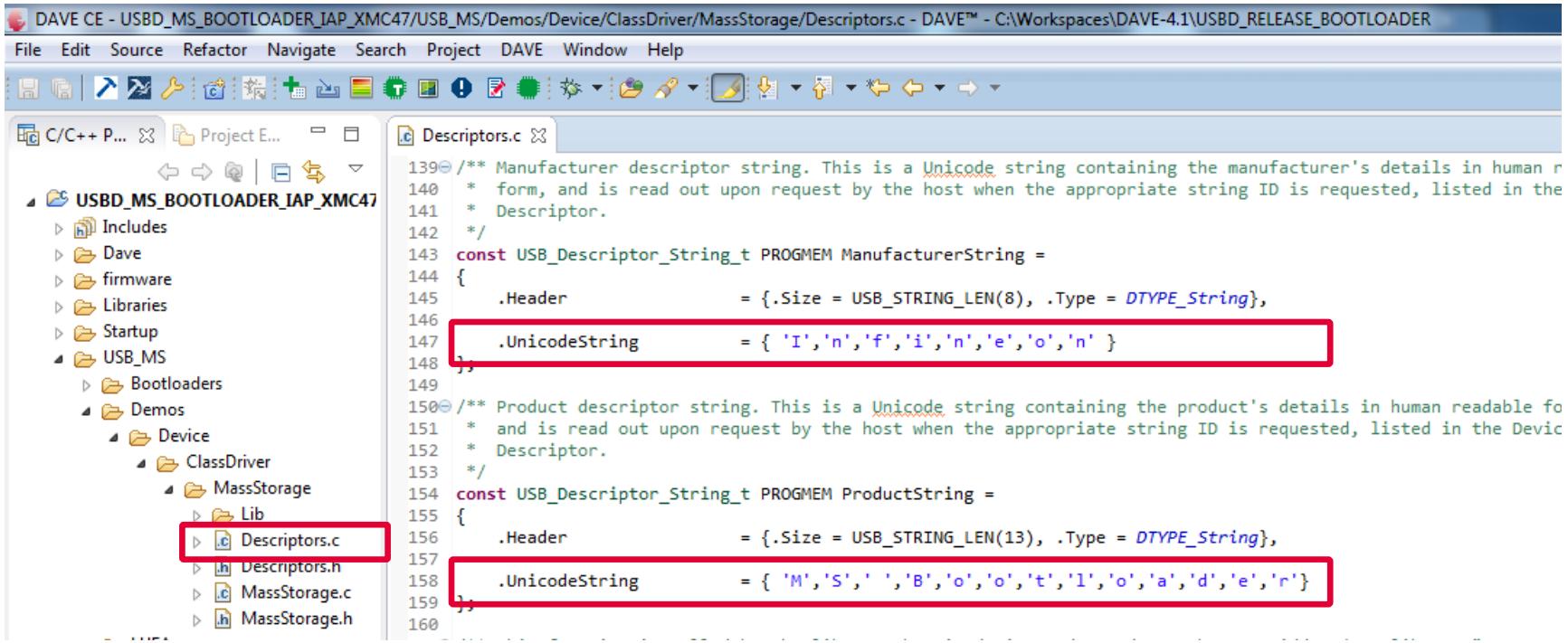
Bootloader – USB device description VendorID, ProductID



Define your companies VendorID and ProductID inside the device description file.

Copyright © Infineon Technologies AG 2016. All rights reserved.

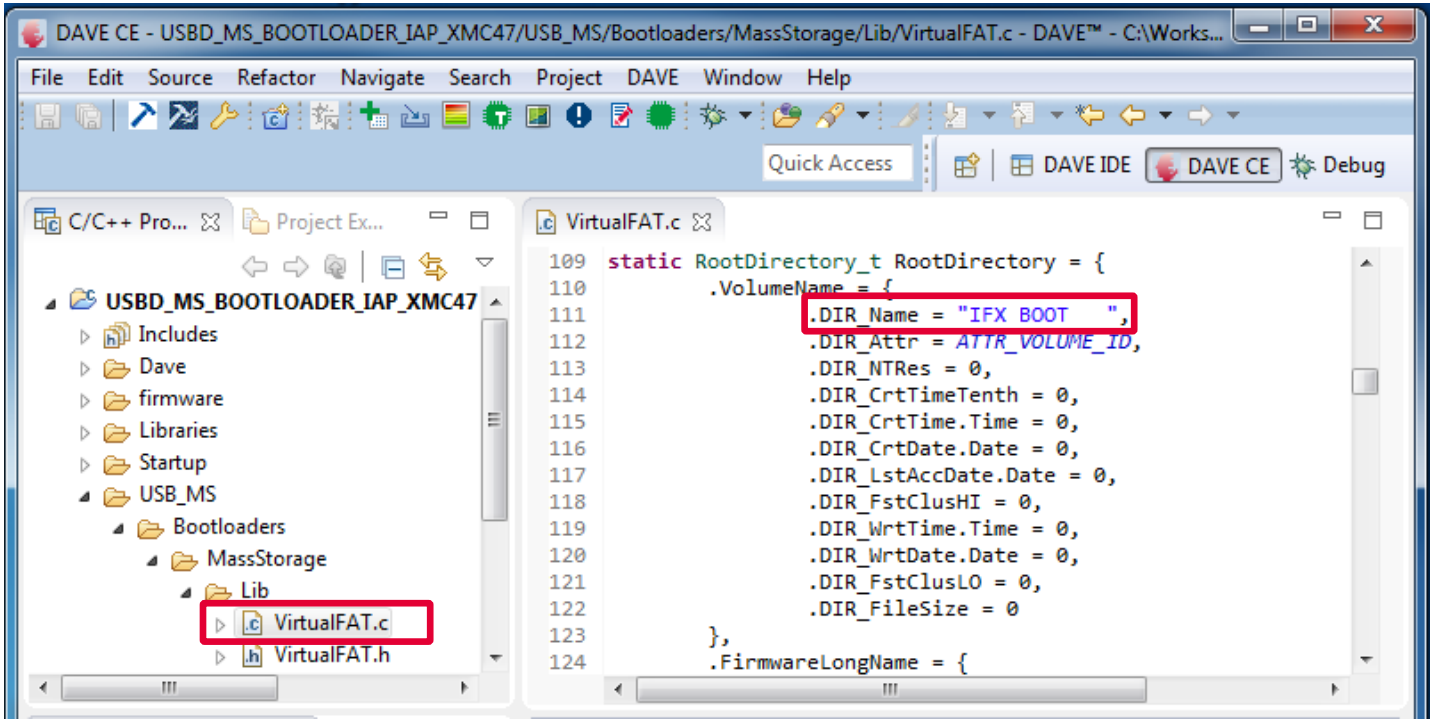
Bootloader – USB device description VendorID, ProductID



Define your own manufacturer and product descriptor string here.

Copyright © Infineon Technologies AG 2016. All rights reserved.

Bootloader – Virtual Filesystem Volume name



Define your own mass storage volume name here.

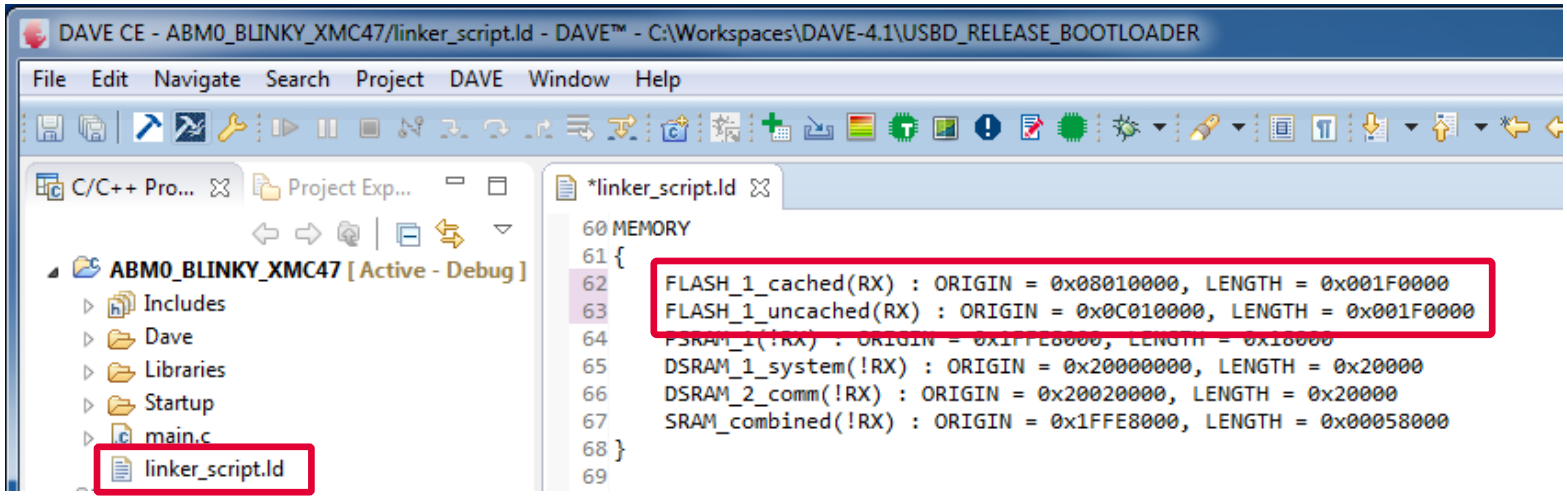
Copyright © Infineon Technologies AG 2016. All rights reserved.

1. Overview and Requirements
2. Setup
3. Short Overview – Boot modes
4. Implementing the bootloader executable
5. Implementing the application executable
6. How to test

Copyright © Infineon Technologies AG 2016. All rights reserved.

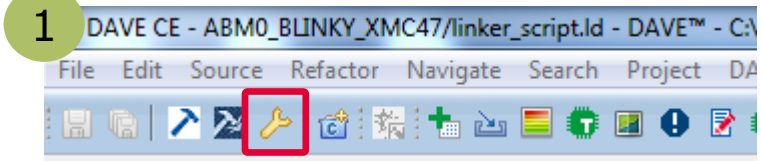
Application – modifying the linker file

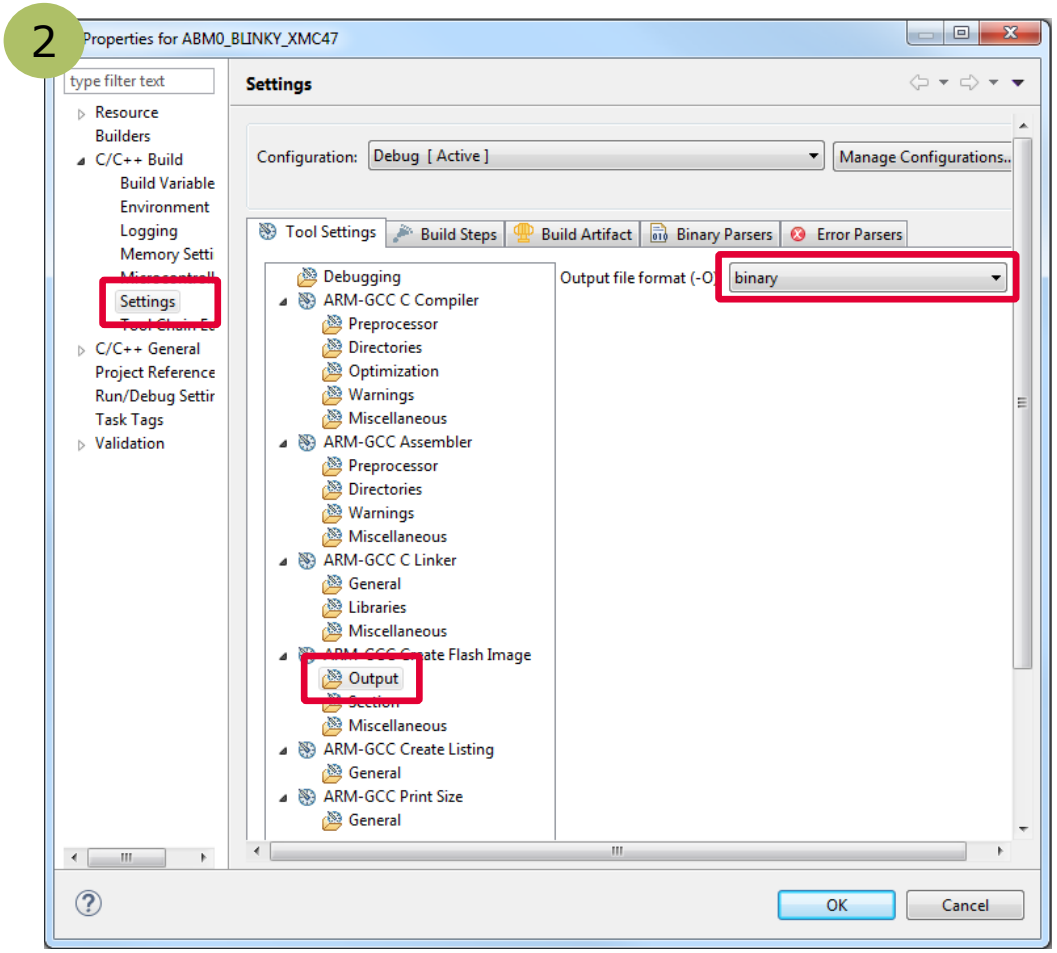
You can turn any application which is executable in normal boot mode into an application which is executable in alternative boot mode 0. For this, you just need to modify the flash start address to 0x08010000/0x0C010000 inside linker file and fix the LENGTH accordingly:



Copyright © Infineon Technologies AG 2016. All rights reserved.

Application – generating binary output file

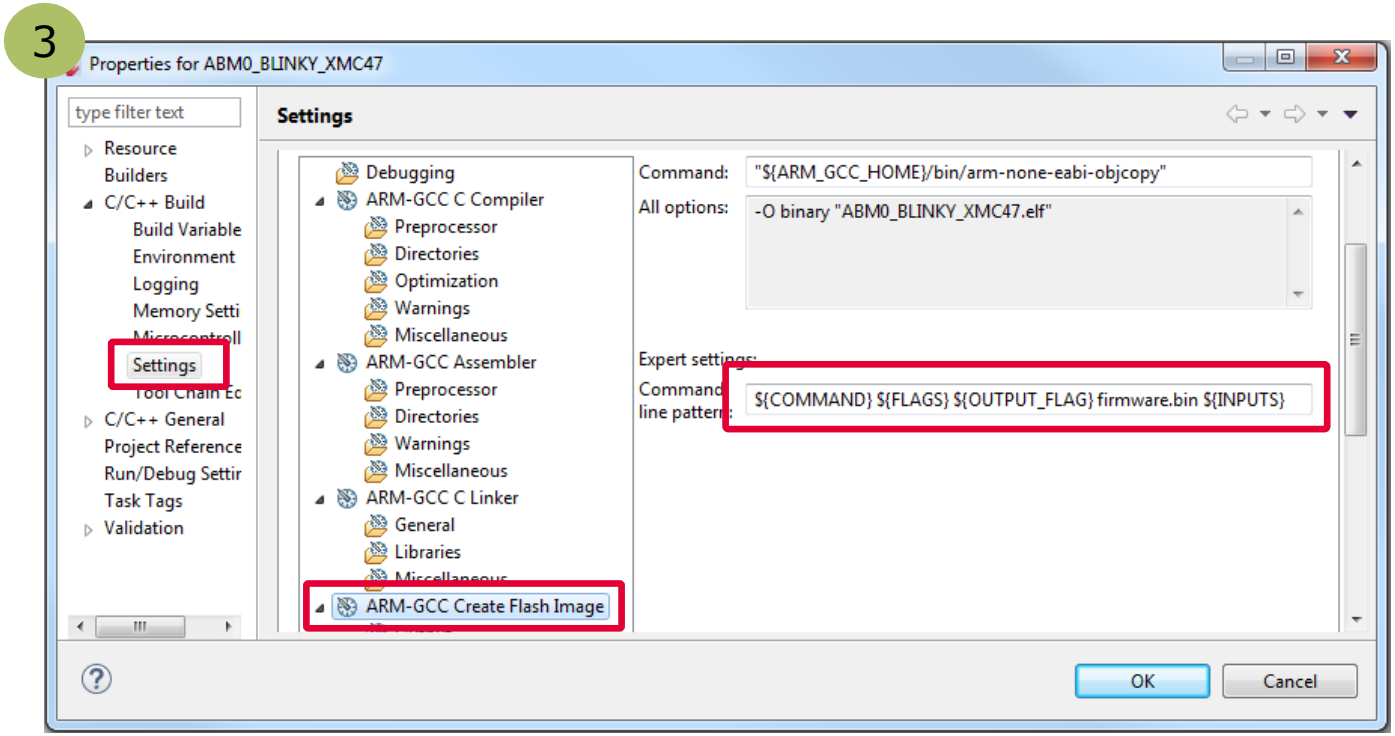




1. Open project properties dialog
2. Set output format to binary format

Copyright © Infineon Technologies AG 2016. All rights reserved.

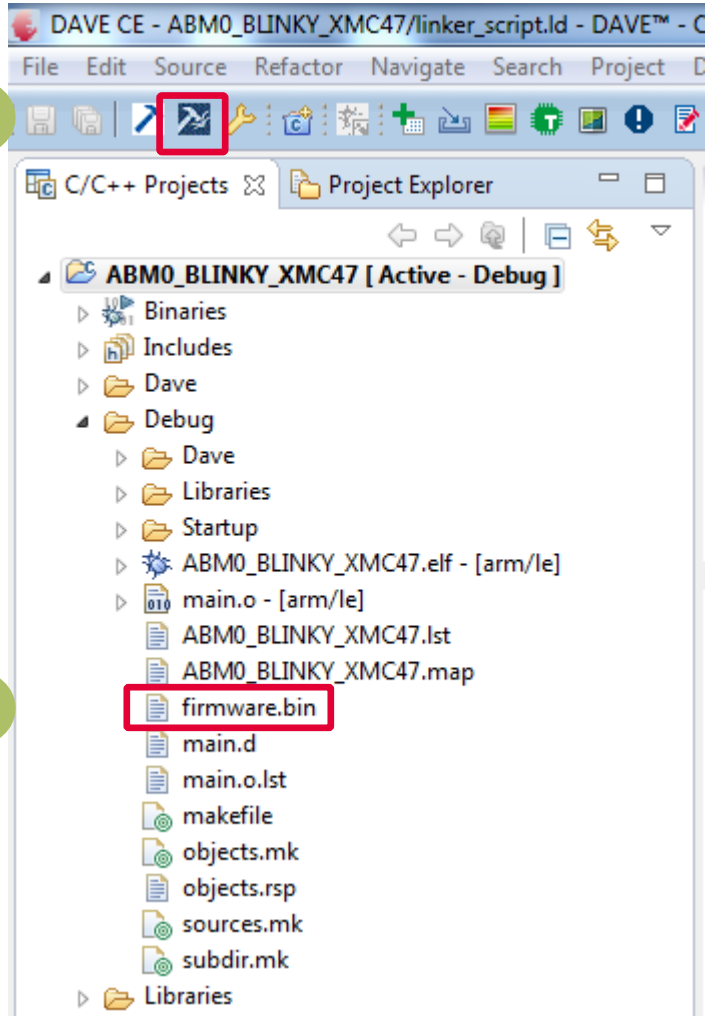
Application – generating binary output file



3 Define the name of the binary flash image as „firmware.bin“

Copyright © Infineon Technologies AG 2016. All rights reserved.

Application – generating binary output file



4 Rebuild your project

5 Pickup the „firmware.bin“ file from your debug folder.

Copyright © Infineon Technologies AG 2016. All rights reserved.

1. Overview and Requirements
2. Setup
3. Short Overview – Boot modes
4. Implementing the bootloader executable
5. Implementing the application executable
6. How to test

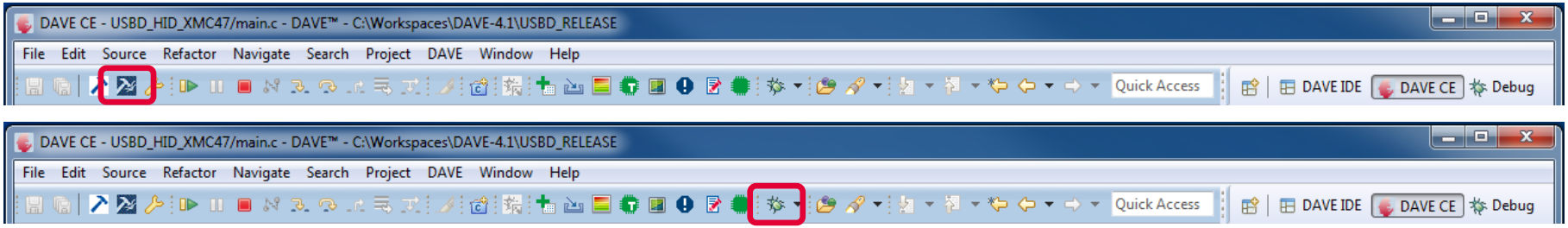
Copyright © Infineon Technologies AG 2016. All rights reserved.

How to test – Build, debug and start bootloader executable

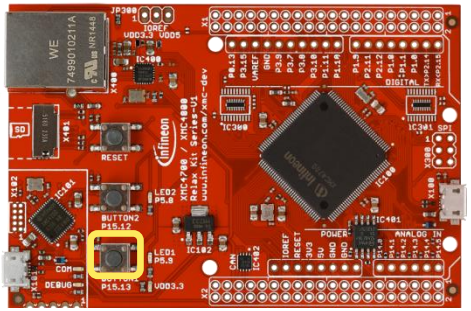


ACTIONS

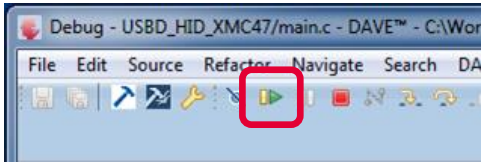
1. Build and download the bootloader example to the XMC4700 / XMC4800 Relax Kit and start the debugger



2. Push BUTTON1 on the XMC4700 / XMC4800 Relax Kit and keep it pushed



3. Start the software by the run button



4. Release BUTTON1 on the XMC4700 / 4800 Relax Kit

Copyright © Infineon Technologies AG 2016. All rights reserved.

How to test – Confirm bootloader executable is running

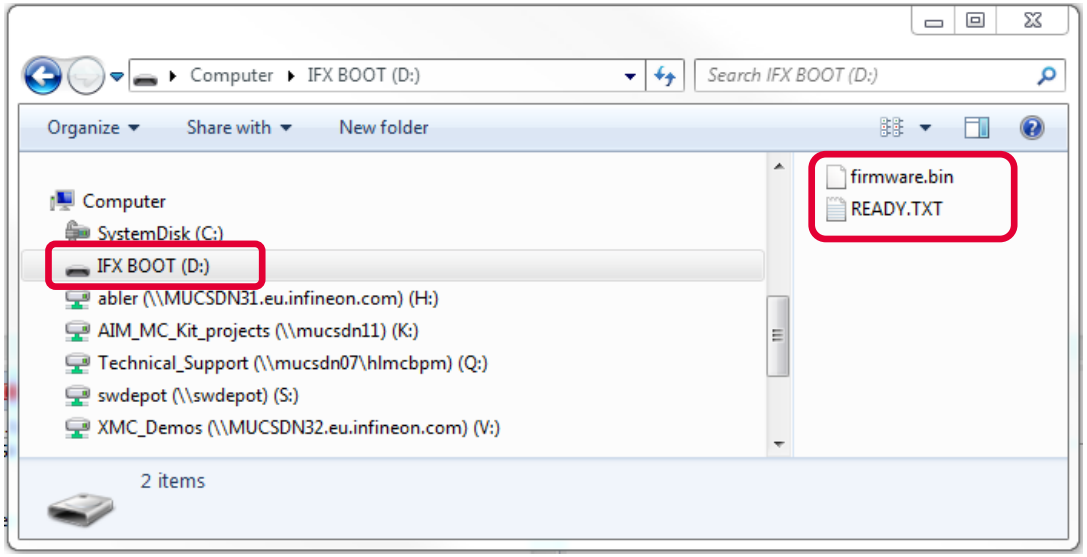
OBSERVATIONS

The USB MS appears inside your explorer window.

You find two files on the USB MS device:

firmware.bin – File representing the flash content

READY.txt – File indicating the status of the device



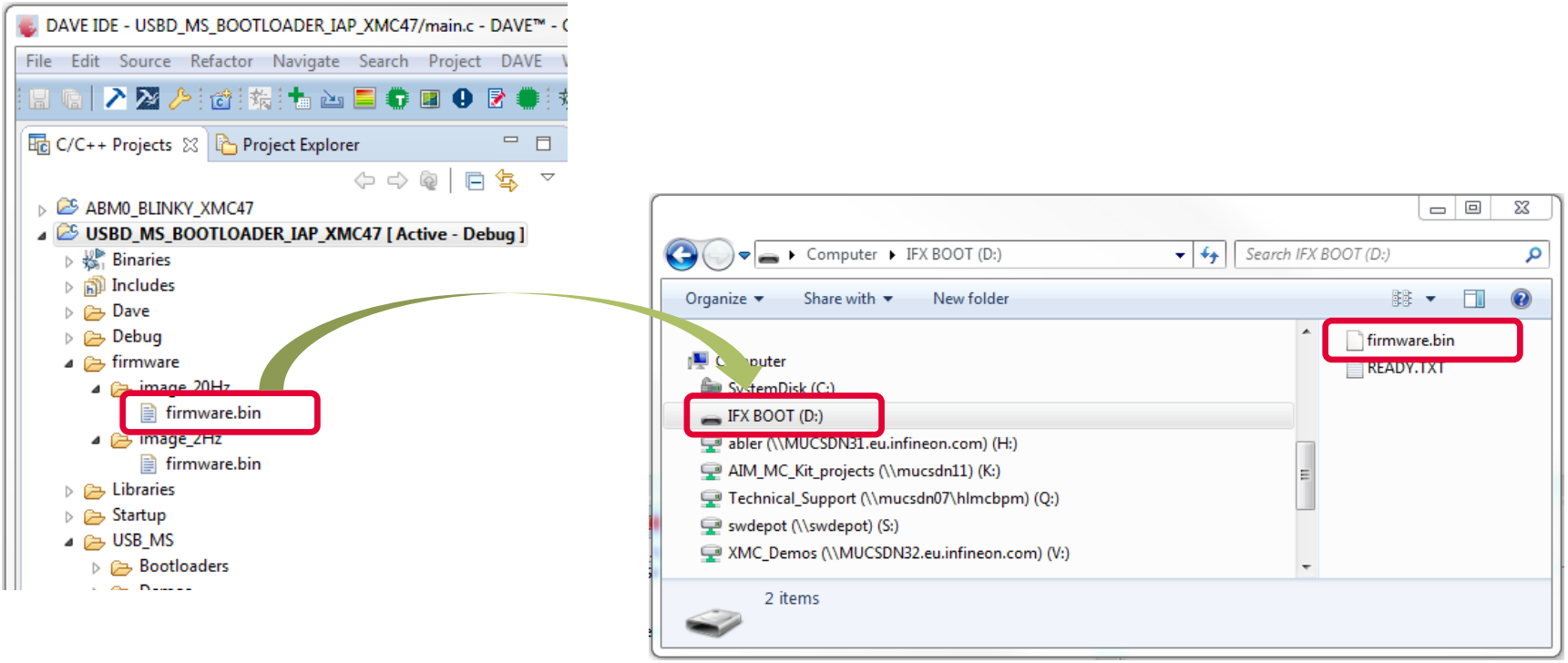
Copyright © Infineon Technologies AG 2016. All rights reserved.

How to test – Program flash with new binary file



ACTIONS

1. Program the flash content for the application executable by overwriting the firmware.bin file. For this drag & drop the 20 Hz firmware image from the example project to the USB MS device:



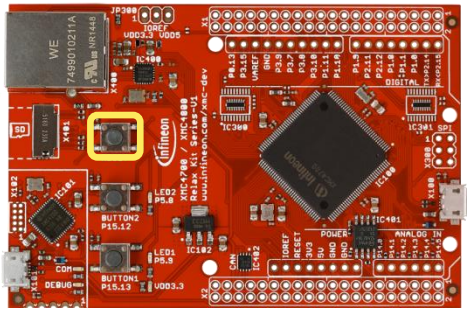
Copyright © Infineon Technologies AG 2016. All rights reserved.

How to test - Start application executable



ACTIONS

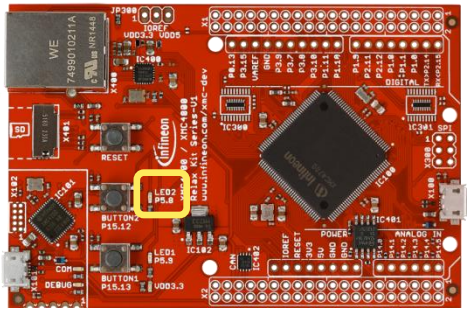
2. Push reset button to restart bootloader and to trigger start of application executable.





OBSERVATIONS

Application executable is running. This is confirmed by LED2 flashing with a frequency of 20 hertz.



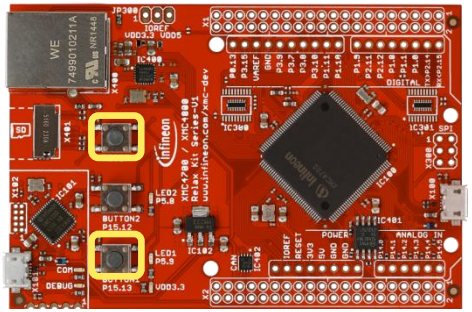
Copyright © Infineon Technologies AG 2016. All rights reserved.

How to test - Start bootloader executable and program flash with new binary

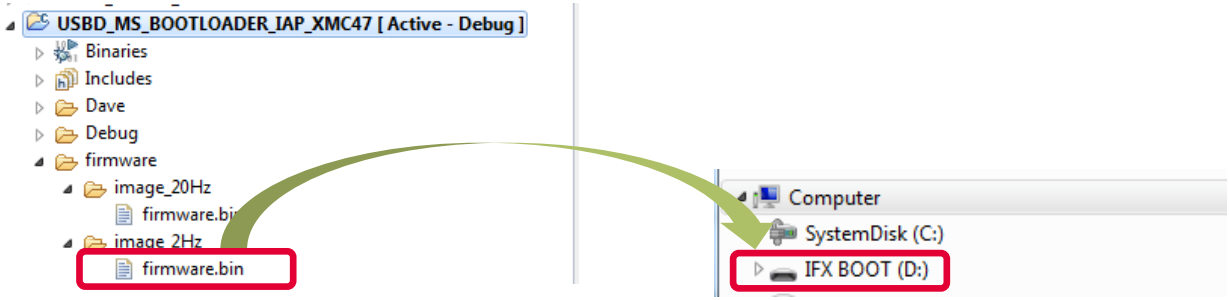


ACTIONS

1. Push reset button while you keep BUTTON1 pressed to restart bootloader executable.



2. Re-program the flash content of application executable by overwriting the firmware.bin file. This time choose the two hertz firmware image:



Copyright © Infineon Technologies AG 2016. All rights reserved.

How to test - Start application executable from bootloader executable



ACTIONS

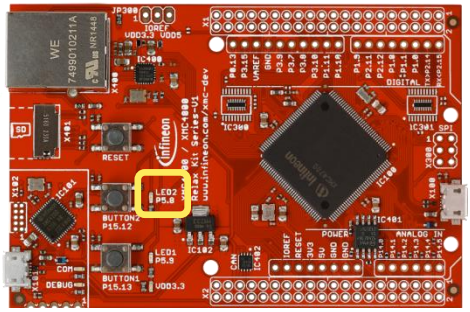
3. Remove USB connector X100 to break USB MS connection. This triggers bootloader executable to start application executable.

Optionally, you can also push reset button.



OBSERVATIONS

Application executable is running. This is confirmed by LED2 flashing with a frequency of two hertz.



Copyright © Infineon Technologies AG 2016. All rights reserved.



Part of your life. Part of tomorrow.

Infineon