XMC4500 USB Mass Storage Host 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

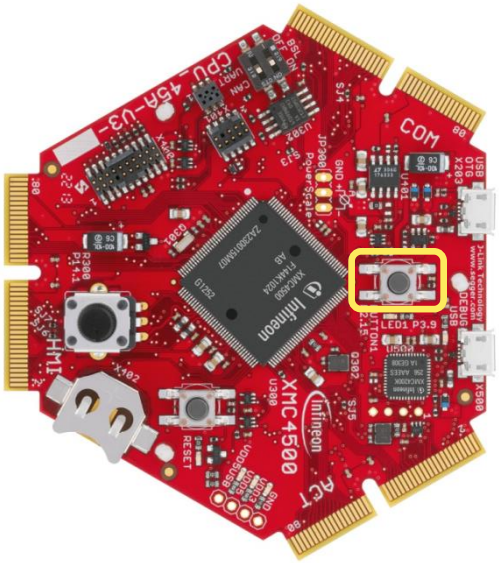
4 Implementing the bootloader executable

5 Implementing the application executable

6 How to test

Copyright © Infineon Technologies AG 2016. All rights reserved.

Overview



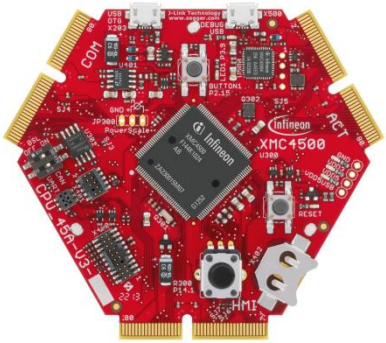
This example demonstrates the implementation of a bootloader for firmware update via USB Mass Storage Host. It is based on the LUFA implementation which is ported to XMC4000 family and provided within this example.

Within this documentation you will learn to configure XMC4500 having two independent flash images installed. The bootloader image implements the USB mass storage host. The bootloader image can be used, to update the firmware „application image". For this, you just need to connect an USB flash drive which contains the firmware binary file to the USB port.

By default, after every reset XMC4500 is executing „application" image. Pushing BUTTON1 during startup forces XMC4500 to remain executing the „bootloader" image and waiting for a USB flash drive to connect.

Copyright © Infineon Technologies AG 2016. All rights reserved.

Requirements – hardware



XMC4500 general purpose board CPU_45A-V3



USB cable micro-A plug to standard-A receptacle for host example



USB cable micro-B plug to standard-A plug for debugging



USB flash drive

Copyright © Infineon Technologies AG 2016. All rights reserved.

Requirements - free software downloads



- Windows laptop
- DAVE™ installed

- DAVE™ (v4.1.4 or Higher)
- Download DAVE™ free of charge

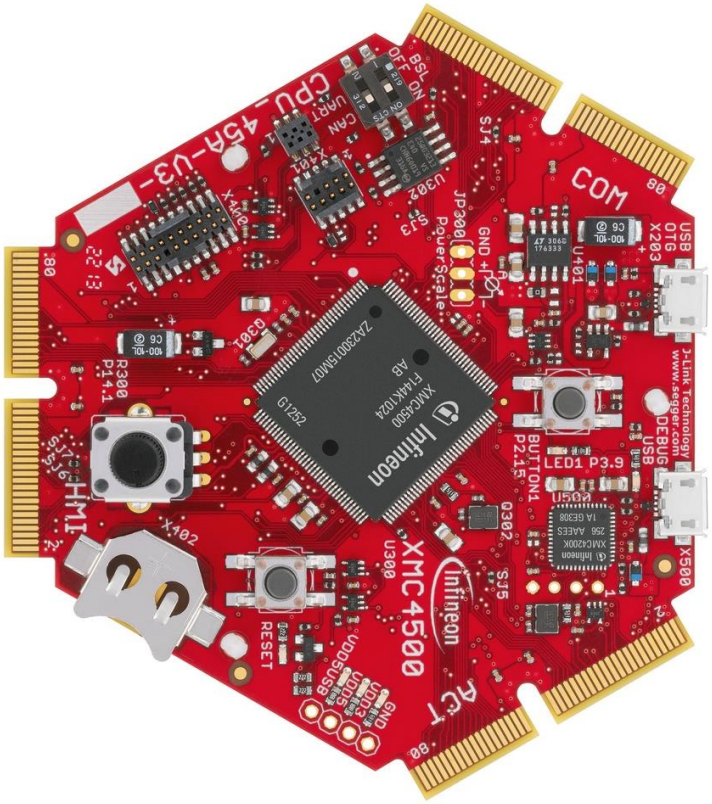
  Link : DAVE™ v4 Download

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



Micro-A plug

Micro-B plug

Std-A receptacle

Flash Drive

Std-A plug

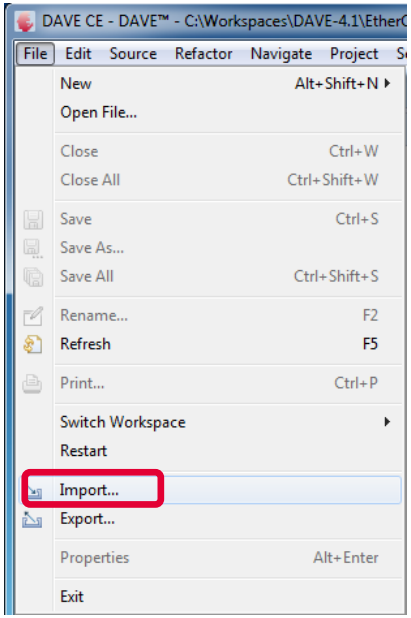
Windows laptop running DAVE™



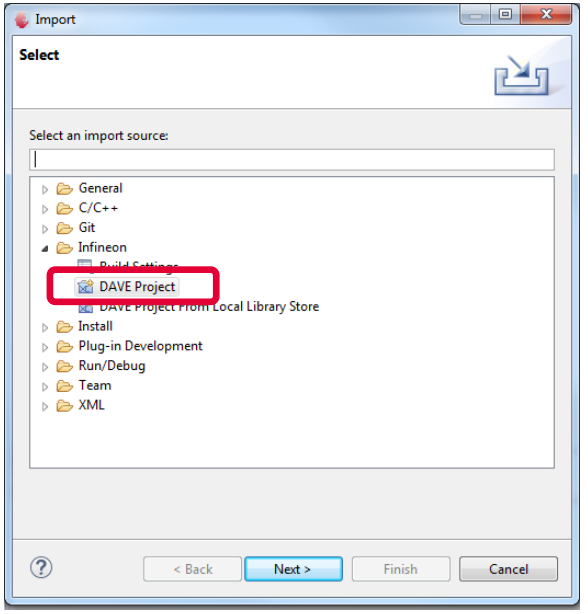
Copyright © Infineon Technologies AG 2016. All rights reserved.

Setup – Import example project into DAVE™

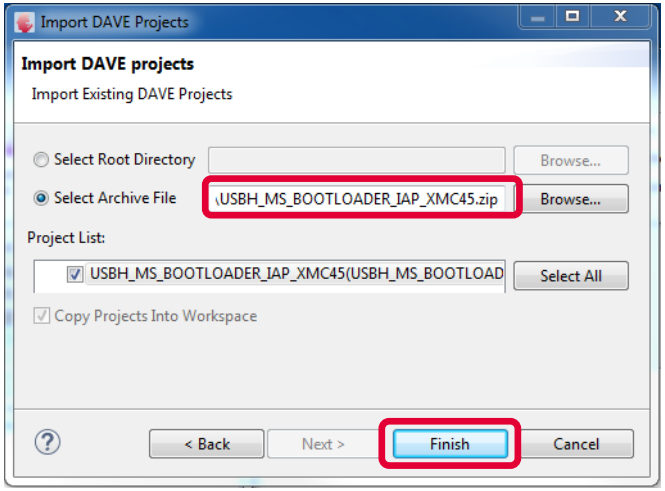
1



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

XMC4500 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 inside flash 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

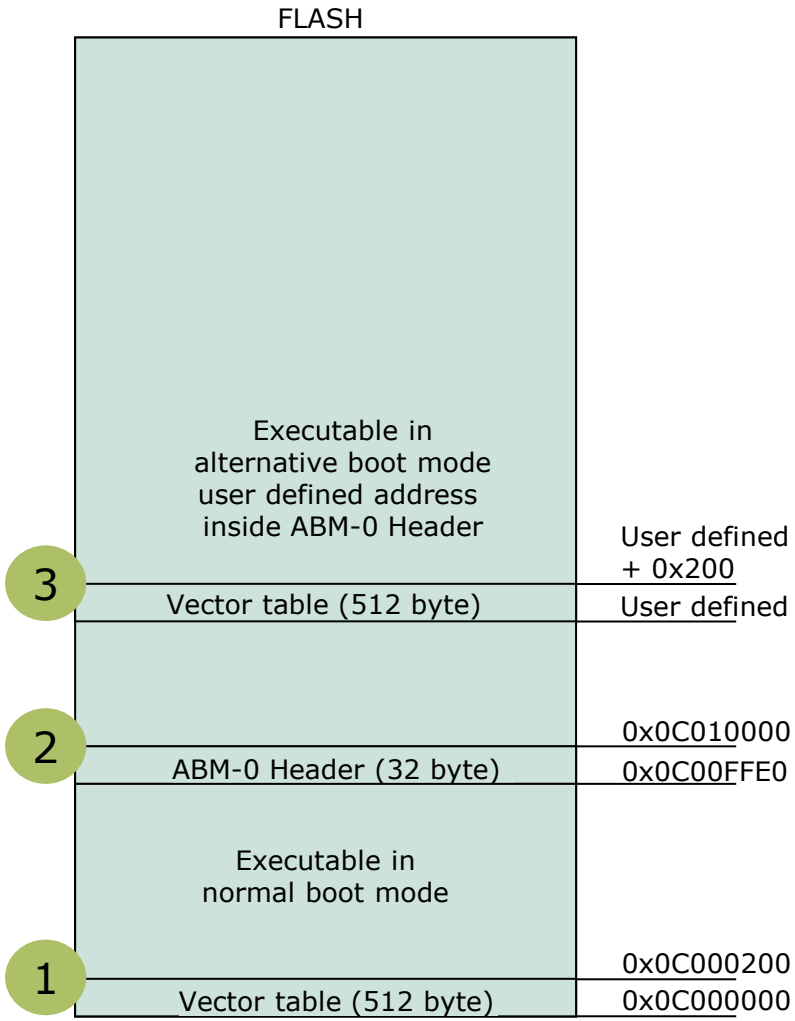
For the example these boot modes are used:

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

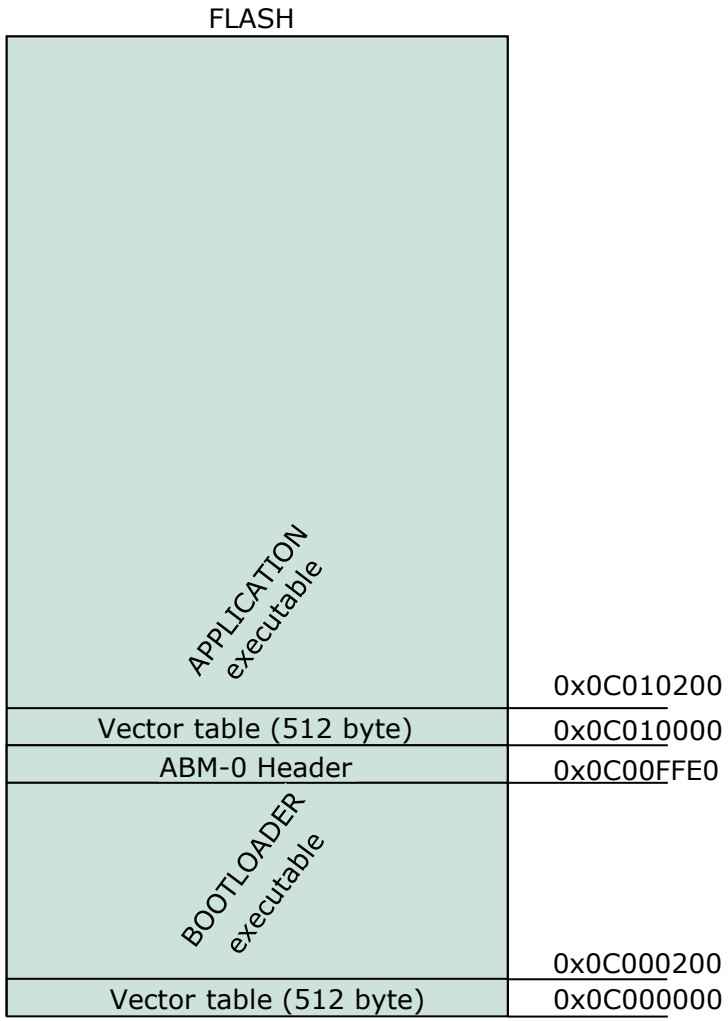


1. In **normal boot mode** execution starts at address 0x0C000000
2. **ABM-0 header** is located at fixed address 0x0C00FFE0
3. In **ABM-0 boot mode** execution starts at address defined by user inside ABM-0 header

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

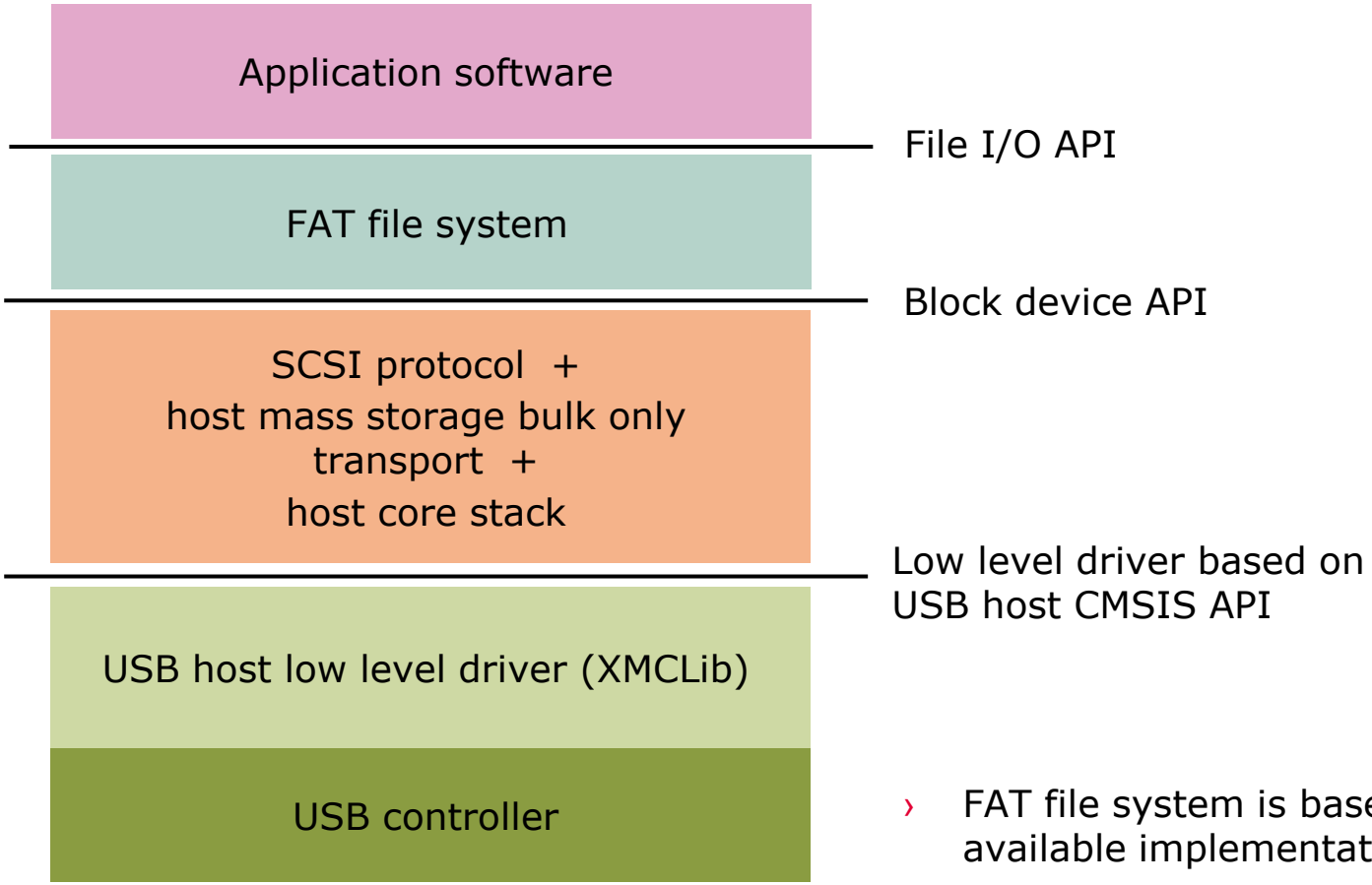
Copyright © Infineon Technologies AG 2016. All rights reserved.

Bootloader – Functional overview
USB MS host: 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 host is initialized. Bootloader mode is indicated to user by turning LED1 on.
- If an USB mass storage device with a file named „firmware.bin“ is found on USB port, the binary content of the file is programmed to flash address 0x0C010000 of application executable
- When programming of the application executable is finished, the device is restarted in ABM0 mode: application executable is running

Copyright © Infineon Technologies AG 2016. All rights reserved.

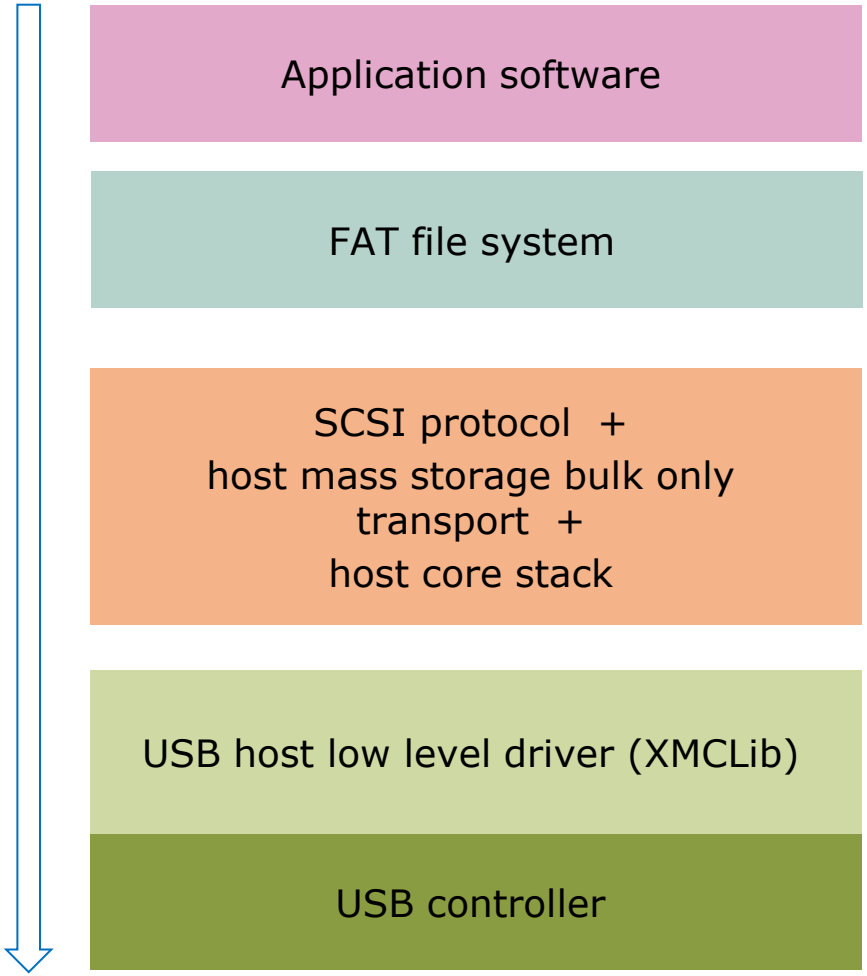
Bootloader – USB MS host architecture



- FAT file system is based on a freely available implementation
- USB stack and mass storage class implementation is based on a freely available stack known as LUFA

Copyright © Infineon Technologies AG 2016. All rights reserved.

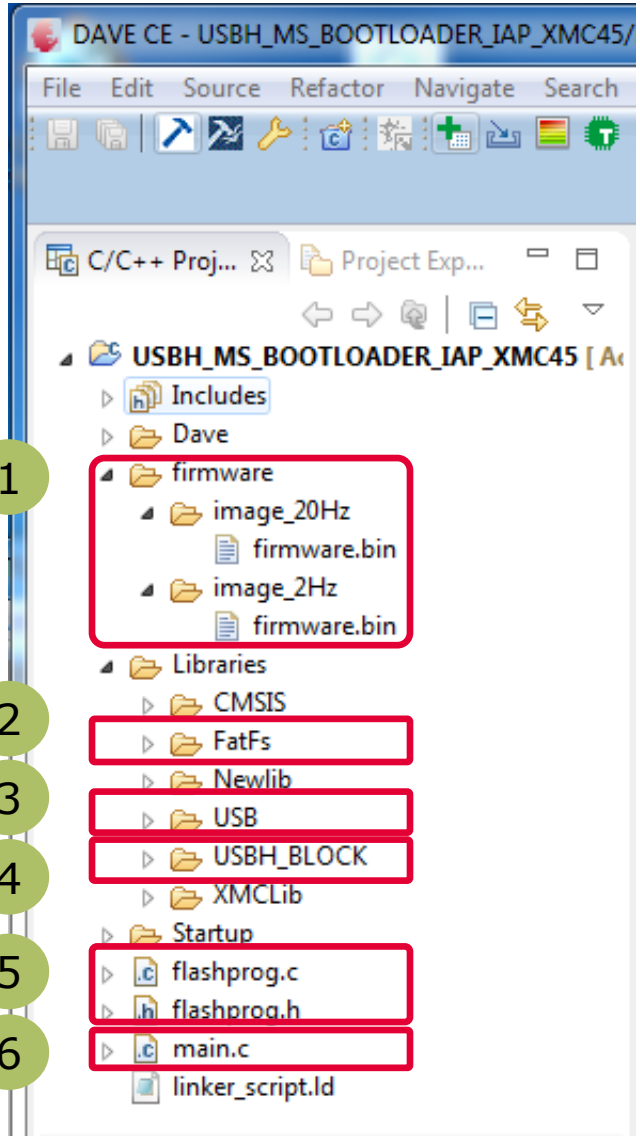
Bootloader – USB MS host data flow



- Application software launches a file IO request using FAT file IO API
- FAT file system converts the file IO calls in to block device calls.
- Blocks are read and written using the standard SCSI commands and the mass storage bulk only transport protocol state machine
- Data is transferred to/from USB mass storage device using low level driver APIs

Copyright © Infineon Technologies AG 2016. All rights reserved.

Bootloader – DAVE™ project structure

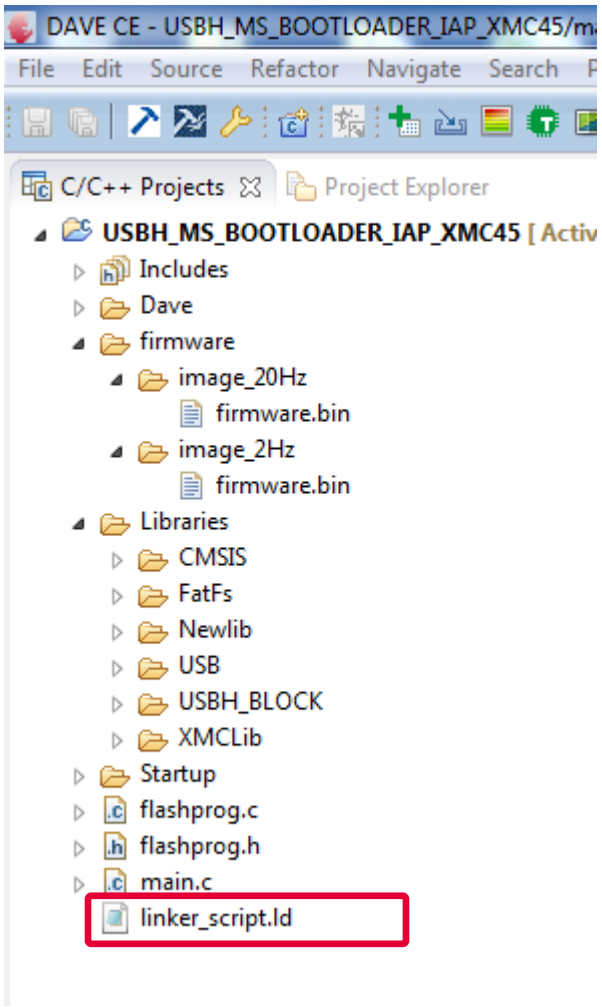


The folder structure of the bootloader example project: Beside standard DAVE project items you will find these USB MS host related items:

1. Binary application images used for testing of this example
2. Free FatFs code ported to XMC4000.
3. LUFA USB host stack and mass storage class ported to XMC4000
4. Block device implementation used by FAT filesystem implementation
5. Flash programming helper functions
6. The main file implementing the bootloader

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.

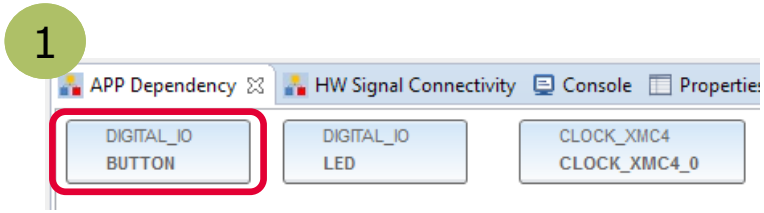
To make build aware of this limitation, the linker file has to be adapted:

linker_script.ld

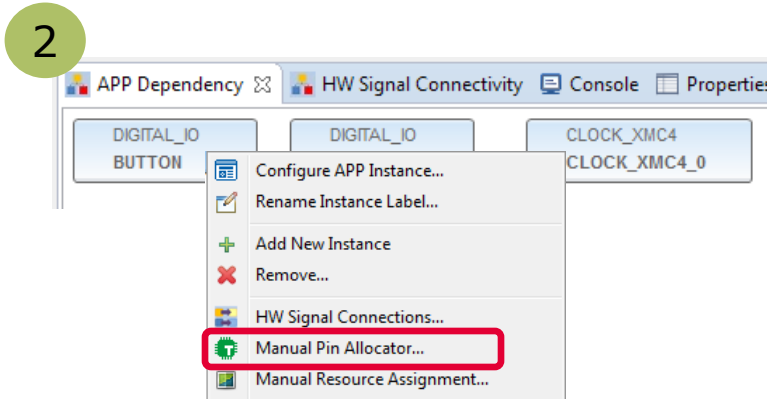
```
61 MEMORY
62 {
63     FLASH_1_cached(RX) : ORIGIN = 0x08000000, LENGTH = 0x010000
64     FLASH_1_uncached(RX) : ORIGIN = 0x0C000000, LENGTH = 0x010000
65     PSRAM_1(!RX) : ORIGIN = 0x10000000, LENGTH = 0x10000
66     DSRAM_1_system(!RX) : ORIGIN = 0x20000000, LENGTH = 0x10000
67     DSRAM_2_comm(!RX) : ORIGIN = 0x30000000, LENGTH = 0x8000
68 }
```


Copyright © Infineon Technologies AG 2016. All rights reserved.

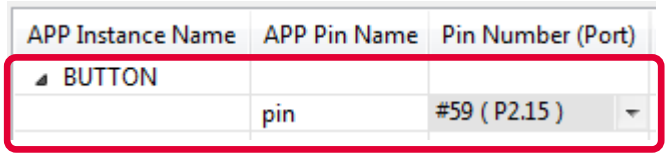
Bootloader – DAVE™ APP structure



1 DIGITAL_IO APPs are used to access the input/output ports. Here one port is used to read the state of BUTTON1 on XMC4500 general purpose board.



2 Right click on DIGITAL_IO APP instance „BUTTON". From the context menu select „Manual Pin Allocator" to open the pin allocation. Assign BUTTON1 input port of the XMC4500 general purpose board to the DIGITAL_IO APP.



Copyright © Infineon Technologies AG 2016. All rights reserved.

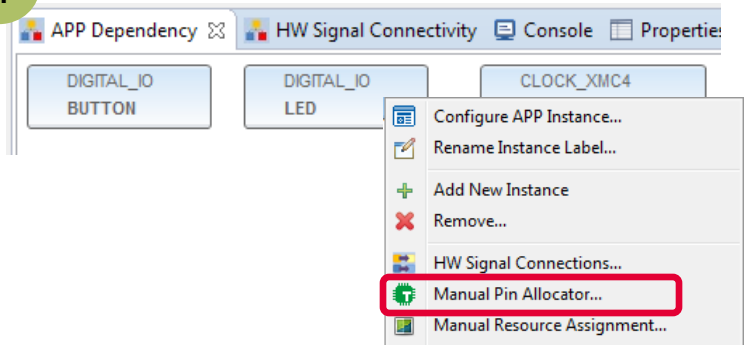
Bootloader – DAVE™ APP structure

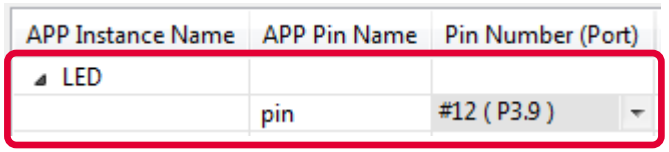
3



3 DIGITAL_IO APPs are used to access the input/output ports. Here one port is used to set the state of the LED1 on XMC4500 general purpose board. By turning LED1 into on state, it is indicated bootloader is active.

4



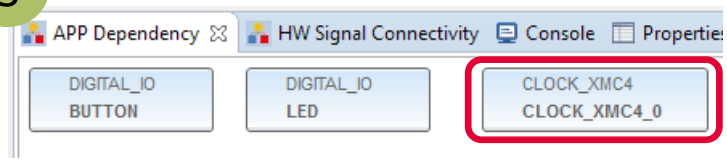


4 Right click on DIGITAL_IO APP instance „LED“. From the context menu select „Manual Pin Allocator“ to open the pin allocation. Assign LED1 output port of the XMC4500 general purpose board to the DIGITAL_IO APP.

Copyright © Infineon Technologies AG 2016. All rights reserved.

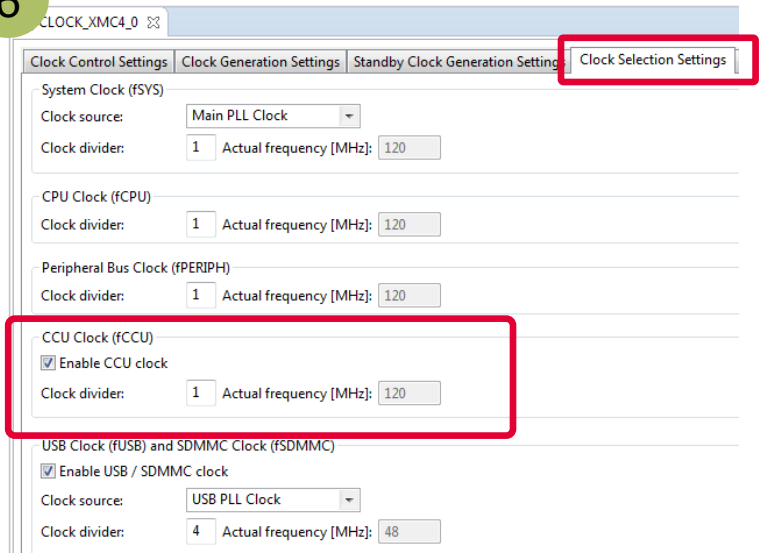
Bootloader – DAVE™ APP structure

5



5 CLOCK_XMC4 APP is used to configure the clock system.
In the context of this example, this APP is important, to configure the clock for the USB subsystem.

6



6 Once this APP is added to your project, USB clock is enabled by default

Copyright © Infineon Technologies AG 2016. All rights reserved.

Bootloader – implementing main()

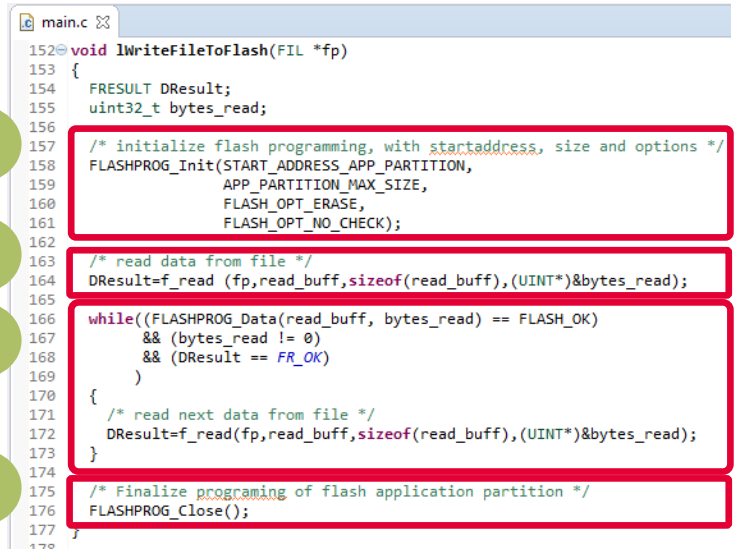
main.c

```
186 int32_t main(void)
187 {
188   FRESULT DResult;      /* Result of file operations */
189   FIL fp ;              /* File pointer */
190   FATFS myfsObject;     /* File System Object Structure variable */
191   uint8_t USBHostState; /* Variable to access the current USB host state */
192
193   /*Initialize DAVE */
194   if(DAVE_Init() == DAVE_STATUS_FAILURE)
195   {
196     /* Placeholder for error handler code.*/
197     XMC_DEBUG(("DAVE APPs initialization failed\n"));
198     while(1)
199     {
200       /* do nothing */
201     }
202   }
203
204   if(DIGITAL_IO_GetInput(&BUTTON))
205   {
206     /* button1 is not pushed -> no firmware update */
207     BL_FlashABM0_Restart(); /* restart device in ABM0 */
208   }
209   else
210   {
211     /* button1 is pushed -> firmware update */
212     /* Indicate by turning LED static on, that bootloader is active */
213     DIGITAL_IO_SetOutputLow(&LED);
214     /* Initialize the USB host driver. */
215     USB_Init_Host(&MS_CB);
216     XMC_USBH_Select_VBUS(XMC_GPIO_PORT3, 2U); /* Select P3.2 as VBUS pin*/
217
218     /* Process USB MS host stack and program flash */
219     for(;;)
220     {
221       /* Get USB Host status. */
222       USB_GetHostState(&USBHostState);
223       /* Enumerate the attached mass storage device */
224       USBBL_Process();
225       /* Mass storage device configured ?*/
226       if(USBHostState == HOST_STATE_Configured)
227       {
228         /* mount file system and check availability of firmware binary */
229         f_mount(&myfsObject, "0:", 0);
230         DResult = f_open( &fp, "firmware.bin", (FA_READ));
231         /* firmware binary file available ?*/
232         if (DResult == FR_OK)
233         {
234           /* Opening firmware binary file succeeded
235            * now program content of firmware binary file into flash */
236           lWriteFileToFlash(&fp);
237           /* Restart Device in ABM0 */
238           BL_FlashABM0_Restart();
239         }
240         f_close(&fp);
241       }
242       /* Keep calling this function for USB management.
243        * This function handles the USB host states till it gets
244        * addressed */
245       USB_USBTask();
246     }
247   }
248   return 0;
249 }
```

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 USB mass storage host
   - wait for successful enumeration
   - mount the filesystem
   - check for the firmware binary on the USB flash drive
   - if firmware binary is available, program the XMC flash.

Copyright © Infineon Technologies AG 2016. All rights reserved.

Bootloader – programming the content of binary from USB flash drive to XMC flash



1. Initialize programming of XMC flash with start address, size and options (erase flash before write, no checking)
2. Read first data chunk from file
3. Write data chunk to XMC flash and proceed, as long as data is available inside file
4. Finalize programming of XMC flash

Copyright © Infineon Technologies AG 2016. All rights reserved.

Bootloader – defining ABM0-header

main.c

```
52 /*****************************************************************
53  * MACROS AND DEFINES
54  *****************************************************************/
55 /** MagicKey value for \ref ABM_Header_t - see */
56 #define ABM_HEADER_MAGIC_KEY          0xA5C3E10F
57 /* start address of application partition */
58 #define START_ADDRESS_APP_PARTITION   (uint32_t*)0x0C010000
59 /* size of application partition; first 64KByte of flash occupied by bootlader */
60 #define APP_PARTITION_MAX_SIZE        (2 * 1024 * 1024 - 64 * 1024)
61
63  * \brief Alternative Boot Mode structure
83 typedef struct ABM_Header {
84   uint32_t MagicKey; /**< Magic key. Always 0xA5C3E10F */
85   uint32_t StartAddress; /**< Start address of the programm to load */
86   uint32_t Length; /**< Length of the programm to load. */
87   uint32_t ApplicationCRC32; /**< CRC32 Sum of the complete application code */
88   uint32_t HeaderCRC32; /**< CRC32 Sum of the four fields before */
89 } ABM_Header_t;
90
91 /*****************************************************************
92  * LOCAL DATA
93  *****************************************************************/
94 static const ABM_Header_t  __attribute__((section(".flash_abm")))
95 ABM0_Header = {
96     .MagicKey = ABM_HEADER_MAGIC_KEY,
97     .StartAddress =   0x08010000,  /* Start Flash Physical Sector 1 */
98     .Length = 0xFFFFFFFF,
99     .ApplicationCRC32 = 0xFFFFFFFF,
100    .HeaderCRC32 =  0xE176A4E6
101 };
```

linker_script.ld

```
117     } > FLASH_1_cached AT > FLASH_1_uncached
118     __exidx_end = .;
119
120     /* http://mcuoneclipse.com/2012/11/01/defining-variables-at-absolute-add
121     .abm ABSOLUTE(0x0800FFE0): AT(0x0800FFE0 | 0x04000000)
122     {
123        KEEP(*(.flash_abm))
124     } > FLASH_1_cached
125
126     /* DSRAM layout (Lowest to highest)*/
127
128     Stack (NOLOAD) :
129     {
```

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 – triggering systemreset and alternative boot mode 0 execution

main.c

```
118 void BL_FlashABM0_Restart(void)
119 {
120   /* Restart in alternative bootmode 0 */
121   /* Clear the reset source field */
122   XMC_SCU_RESET_ClearDeviceResetReason();
123   /* Set ABM0 as boot mode */
124   XMC_SCU_SetBootMode(XMC_SCU_BOOTMODE_ABM0);
125   /* Trigger power on reset */
126   PPB->AIRCR = 1 << PPB_AIRCR_SYSRESETREQ_Pos |
127                0x5FA<<PPB_AIRCR_VECTKEY_Pos |
128                0x1 << PPB_AIRCR_PRIGROUP_Pos;
129 }
130
131
```

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

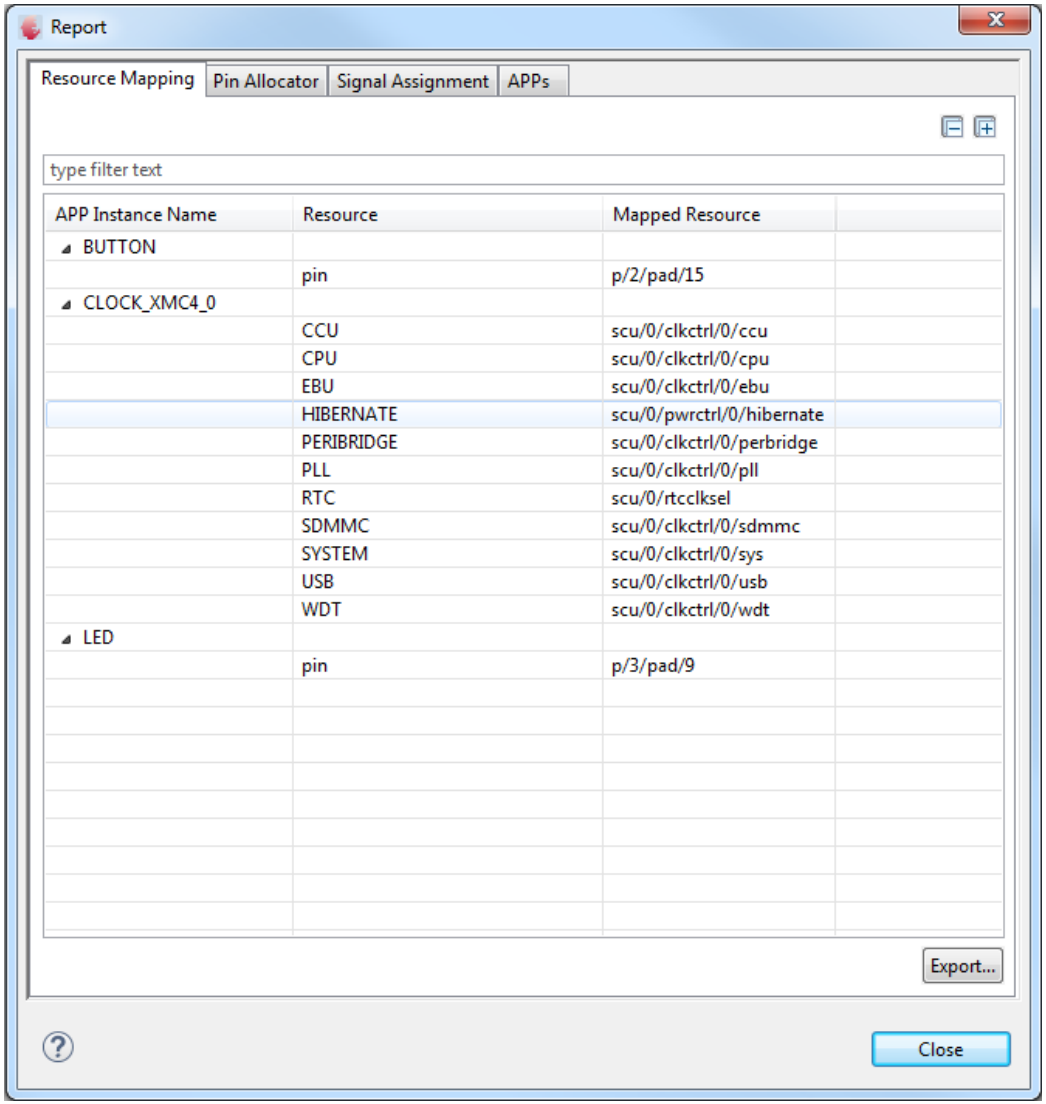
Clear the reset source of the last reset

Set boot mode to alternative boot mode 0

Issue system reset (software reset)

Copyright © Infineon Technologies AG 2016. All rights reserved.

Bootloader – Overview on used HW resources



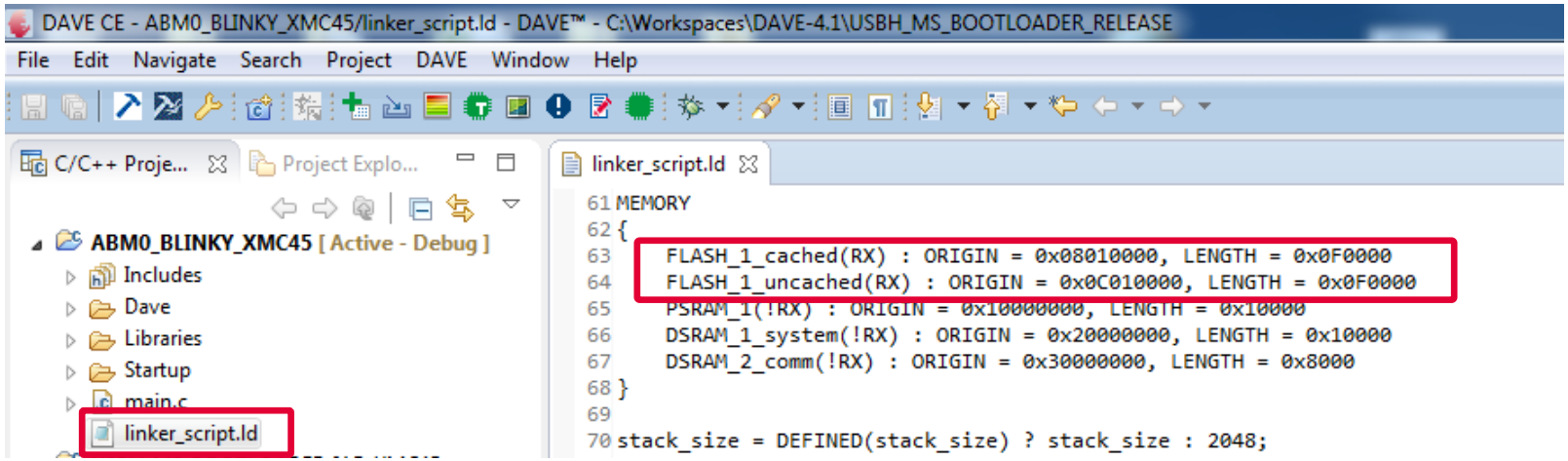
Copyright © Infineon Technologies AG 2016. All rights reserved.

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.

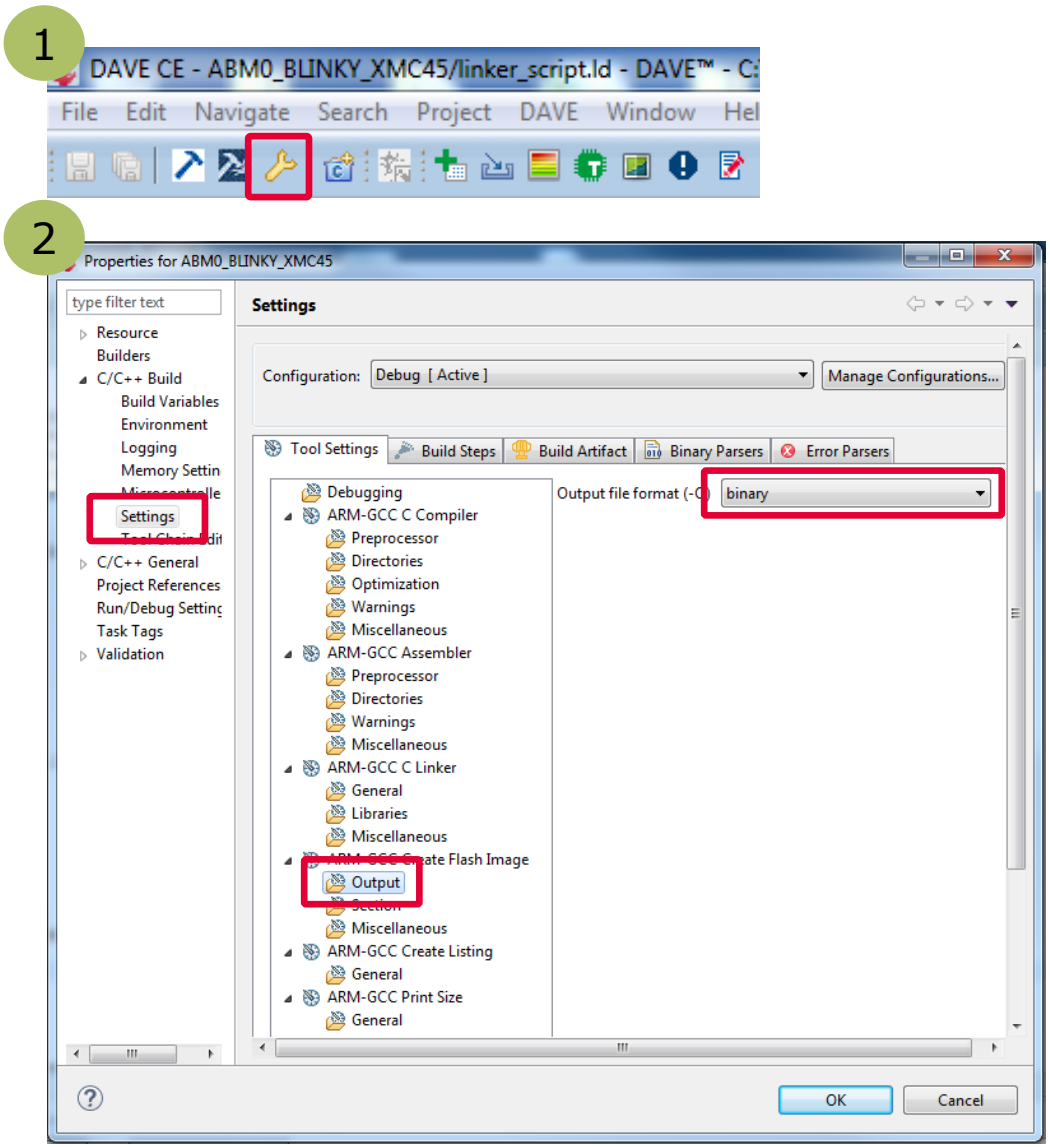
Application – modifying the linker file

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 the linker file and fix the LENGTH accordingly:



Copyright © Infineon Technologies AG 2016. All rights reserved.

Application – generating binary output file

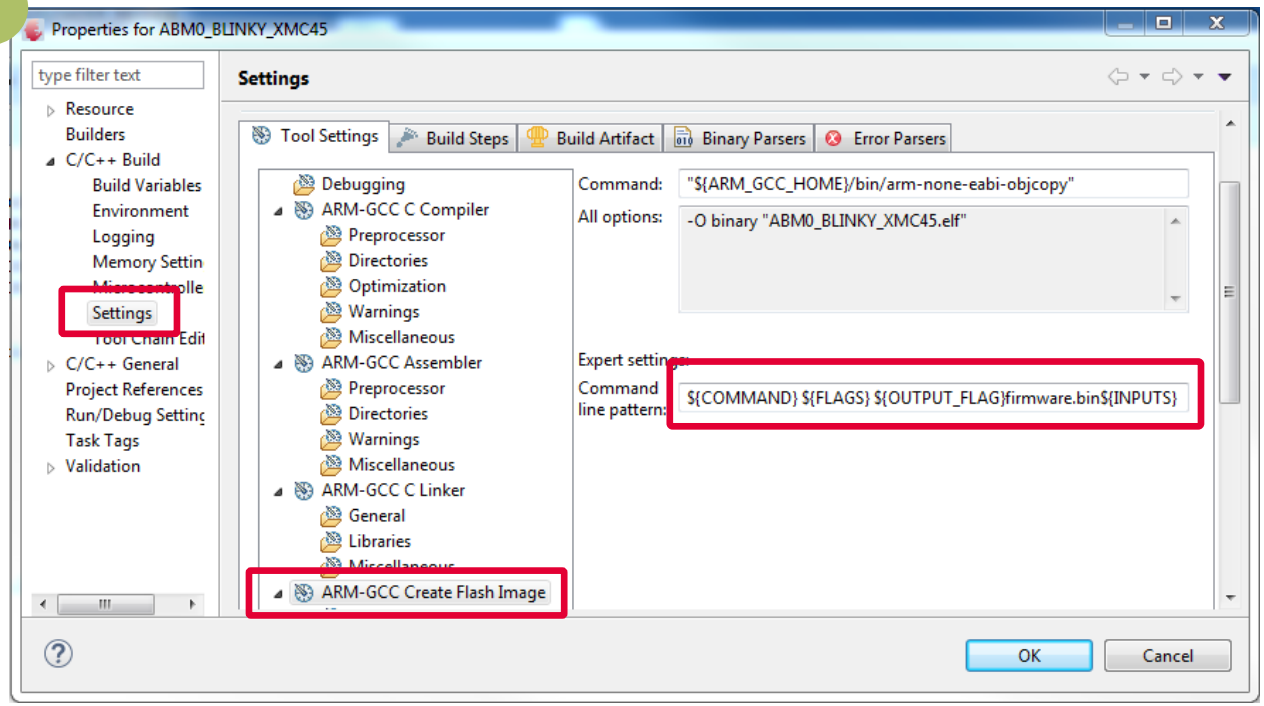


1. Open project properties dialog
2. Set output format to binary format

Copyright © Infineon Technologies AG 2016. All rights reserved.

Application – generating binary output file

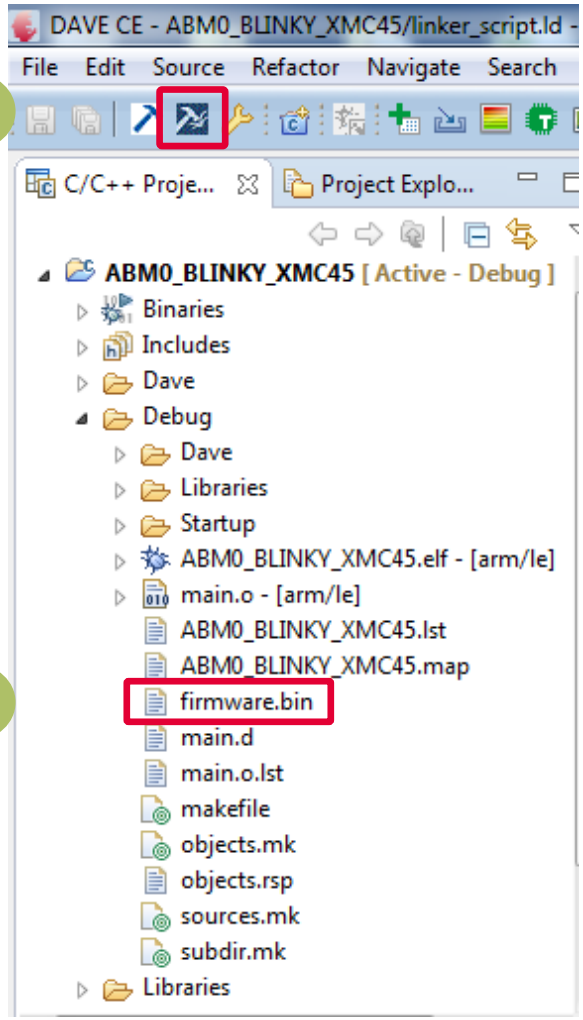
3



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

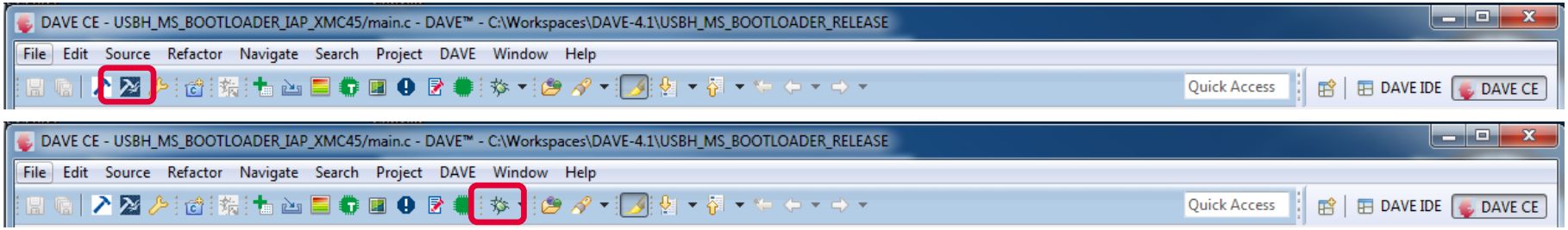
Copyright © Infineon Technologies AG 2016. All rights reserved.

How to test – Build, debug and start bootloader executable

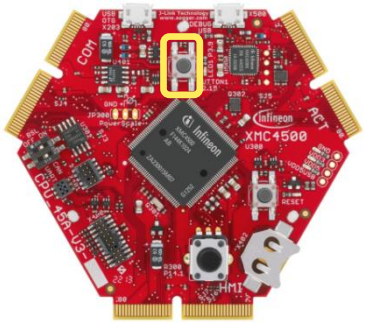


ACTIONS

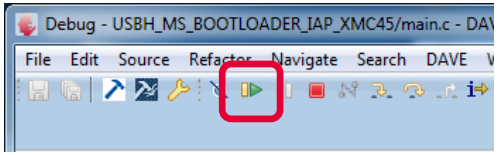
1. Build and download the bootloader example to the XMC4500 general purpose board and start the debugger



2. Push BUTTON1 on the XMC4500 general purpose board



3. Start the software by the run button



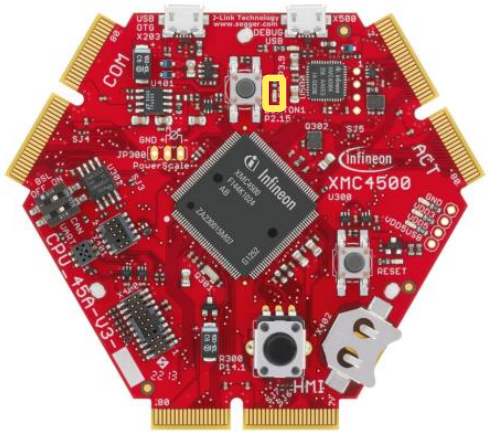
4. Release BUTTON1 on the XMC4500 general purpose board

Copyright © Infineon Technologies AG 2016. All rights reserved.

How to test – Confirm bootloader executable is running

OBSERVATIONS

LED1 on your XMC4500 general purpose board is static turned on. This confirms, the device is executing bootloader and waiting for an USB flash drive to connect.



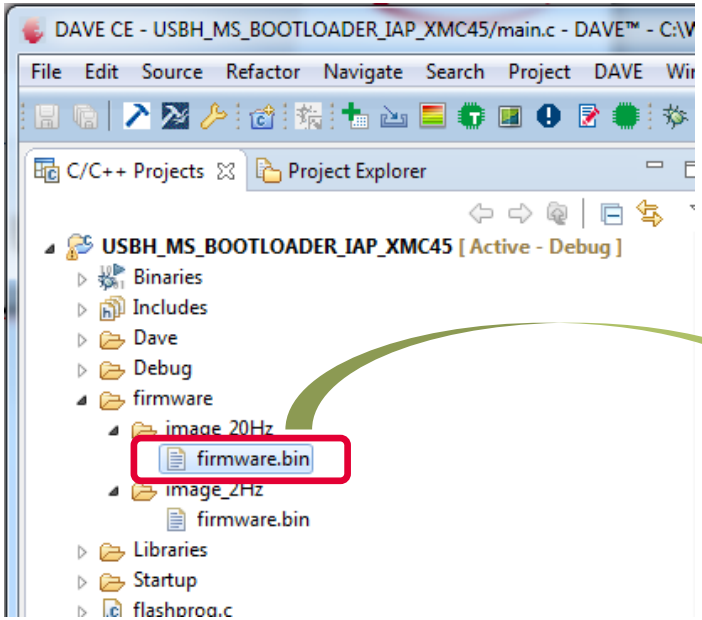
Copyright © Infineon Technologies AG 2016. All rights reserved.

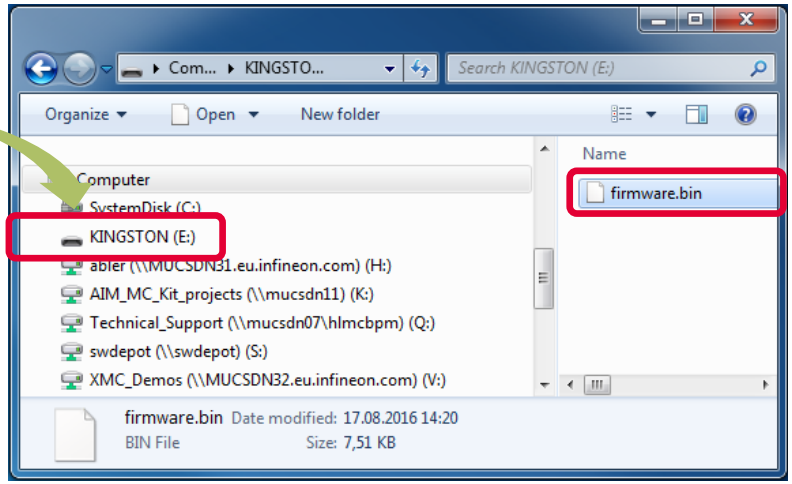
How to test – Firmware update with new binary file (20 Hz version) [1/2]



ACTIONS

1. Prepare your USB flash drive. For this, connect the USB flash drive to your laptop. Copy the 20 Hz application executable „firmware.bin" file to your USB flash drive:





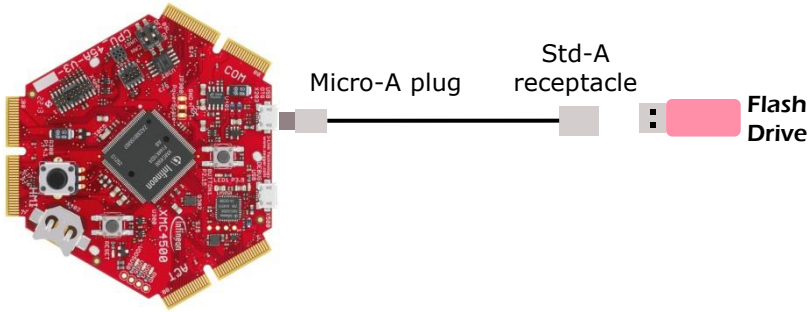
Copyright © Infineon Technologies AG 2016. All rights reserved.

How to test – Firmware update with new binary file (20 Hz version) [2/2]



ACTIONS

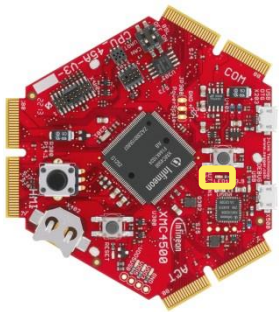
2. Disconnect safely the USB flash drive from the laptop and connect it your XMC4500 general purpose board on connector X203.





OBSERVATIONS

After connecting the USB flash drive, bootloader will enumerate it and start programming the firmware binary.
When finished (~3 seconds), the application binary will be executed.
LED1 will flash with a frequency of 20 Hz.



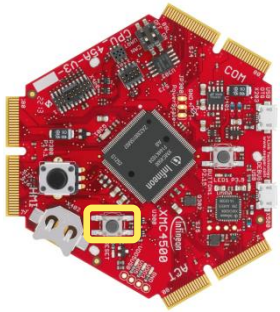
Copyright © Infineon Technologies AG 2016. All rights reserved.

How to test – Start 20 Hz application binary after power on reset



ACTIONS

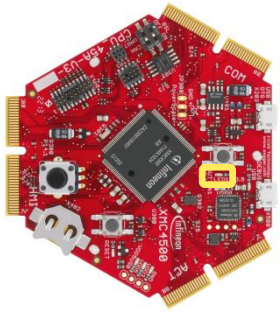
Push reset button without pressing BUTTON1. Bootloader will immediately restart the application binary (in Alternative Boot Mode 0)





OBSERVATIONS

After reset LED1 will flash with a frequency of 20 Hz.



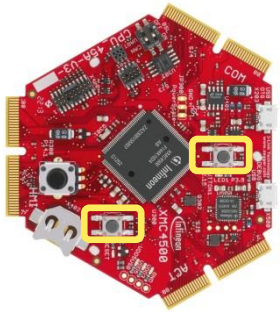
Copyright © Infineon Technologies AG 2016. All rights reserved.

How to test - Restart bootloader executable and confirm bootloader executable is running



ACTIONS

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





OBSERVATIONS

LED1 on your XMC4500 general purpose board is static turned on. This confirms, the device is executing bootloader and waiting for an USB flash drive to connect.

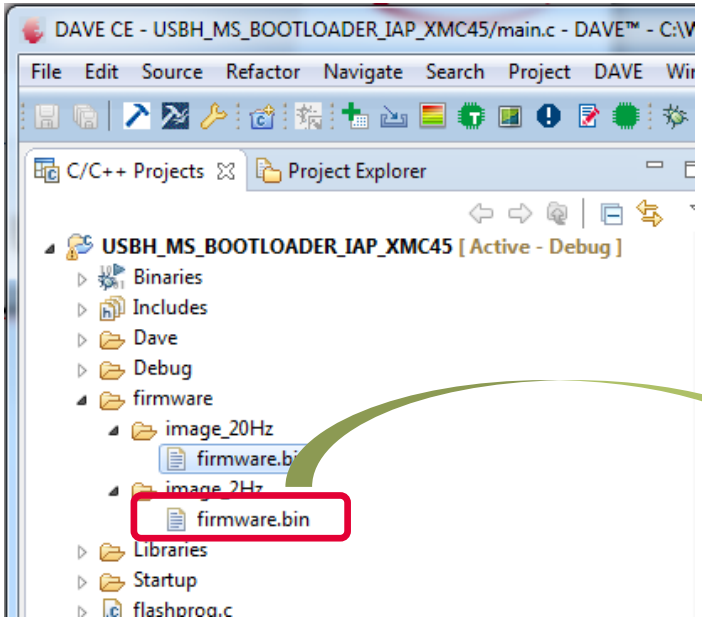
Copyright © Infineon Technologies AG 2016. All rights reserved.

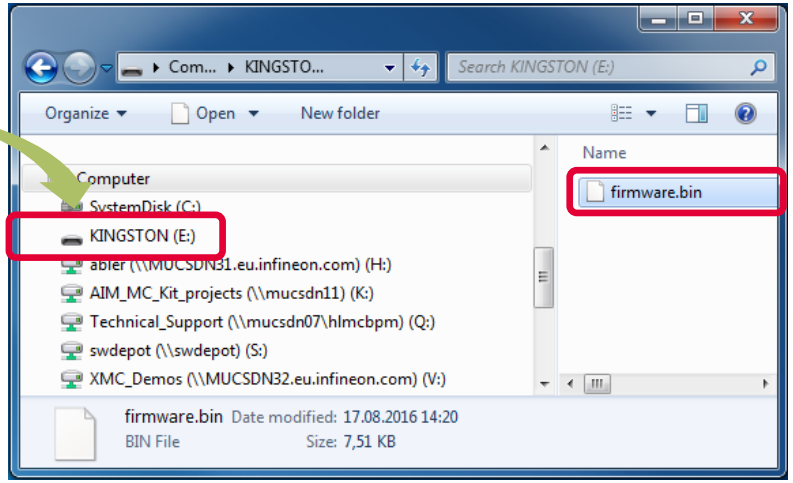
How to test – Firmware update with new binary file (2 Hz version) [1/2]



ACTIONS

1. Prepare your USB flash drive. For this, connect the USB flash drive to your laptop. Copy the 2 Hz application executable „firmware.bin" file to your USB flash drive:





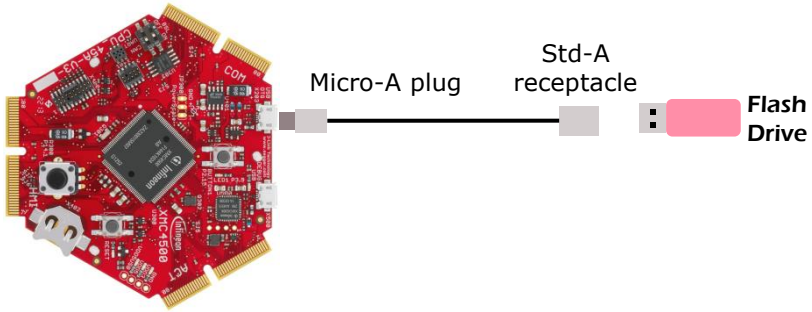
Copyright © Infineon Technologies AG 2016. All rights reserved.

How to test – Firmware update with new binary file [2/2]



ACTIONS

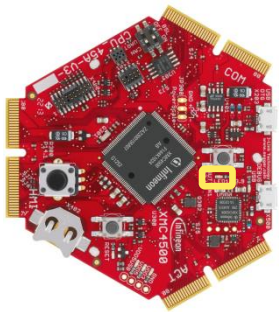
2. Disconnect safely the USB flash drive from the laptop and connect it your XMC4500 general purpose board on connector X203.





OBSERVATIONS

After connecting the USB flash drive, bootloader will enumerate it and start programming the firmware binary.
When finished (~3 seconds), the application binary will be executed.
Now, LED1 will flash with a frequency of 2 Hz.



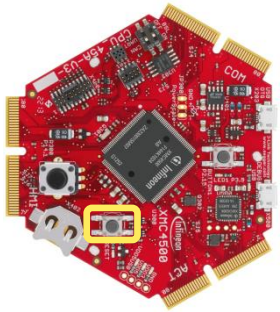
Copyright © Infineon Technologies AG 2016. All rights reserved.

How to test – Start 2Hz application binary after power on reset



ACTIONS

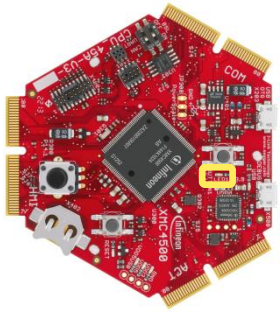
Push reset button without pressing BUTTON1. Bootloader will immediately restart the application binary (in Alternative Boot Mode 0)





OBSERVATIONS

After reset LED1 will flash with a frequency of 2 Hz.



Copyright © Infineon Technologies AG 2016. All rights reserved.



Part of your life. Part of tomorrow.

Infineon