Please note that Cypress is an Infineon Technologies Company.
The document following this cover page is marked as “Cypress” document as this is the company that originally developed the product. Please note that Infineon will continue to offer the product to new and existing customers as part of the Infineon product portfolio.

Continuity of document content
The fact that Infineon offers the following product as part of the Infineon product portfolio does not lead to any changes to this document. Future revisions will occur when appropriate, and any changes will be set out on the document history page.

Continuity of ordering part numbers
Infineon continues to support existing part numbers. Please continue to use the ordering part numbers listed in the datasheet for ordering.



THIS SPEC IS OBSOLETE

Spec No: 001-70193

Spec Title: EZ-USB(R) FX3 SPI BOOT OPTION - AN70193

Sunset Owner: Hasib Mannil (hbm)

Replaced By: 001-76405

EZ-USB® FX3 SPI Boot Option

AN70193

Author: Shruti Maheshwari
Associated Project: No
Associated Part Family: EZ-USB® FX3
Software Version: None
Associated Application Notes: None

OBSOLETE

Application Note Abstract

The features of Cypress EZ-USB® FX3 SPI boot option is described in this Application Note. It also covers the details of selecting SPI boot option, devices supported by the bootloader, and format of boot Image to be loaded onto the EEPROM.

Introduction

Cypress EZ-USB FX3 is the next generation USB 3.0 peripheral controller that provides highly integrated and flexible features that enable developers to add USB 3.0 functionality to any system.

EZ-USB FX3 has a fully configurable, parallel, general programmable interface called GPIF II that can connect to an external processor, ASIC, or FPGA. The GPIF II is an enhanced version of the GPIF in FX2LP, Cypress's flagship USB 2.0 product. It provides easy and glue less connectivity to popular interfaces such as asynchronous SRAM, asynchronous and synchronous address data multiplexed interface, and many others.

FX3 supports many boot options including booting over I^2C, SPI, and USB interfaces. This application note discusses booting from SPI memories.

FX3 Boot Options

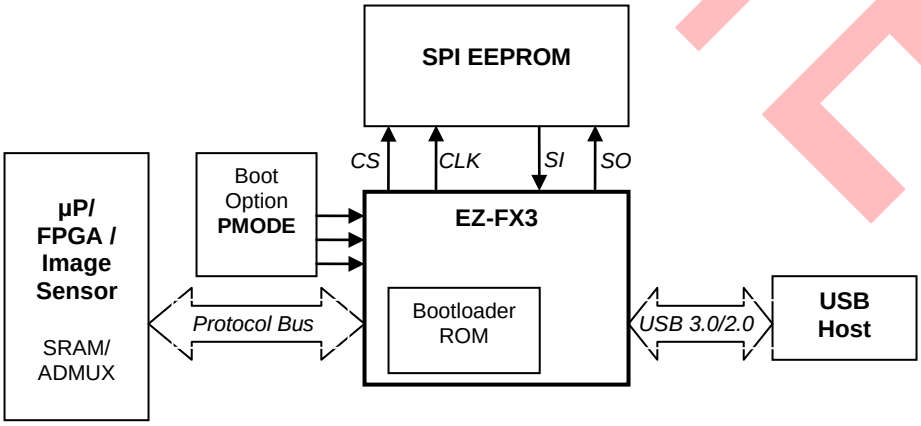
FX3 comes with a bootloader, which is the startup code for the ARM9 CPU that resides in masked ROM. The function of the bootloader is to download the FX3 firmware image from various interfaces such as I^2C EEPROM, SPI EEPROM, GPIF II ASYNC SRAM and ASYNC/SYNC ADMUX interfaces, and USB.

The boot options supported by the bootloader in EZ-USB FX3 are listed in Table 1.

Table 1. Boot Option Selection for FX3

PMODE Pins	Boot Option	USB Fallback
F00	Sync ADMUX (16-bit)	No
F01	Async ADMUX (16-bit)	No
F11	USB Boot	Yes
F0F	SRAM (16-bit)	No
1FF	I^2C	No
F1F	I^2C => USB	Yes
0F1	SPI => USB	Yes
Other combinations are reserved.		

Figure 1. High-Level Interface for FX3 Bootloader



PMODE Pins: The FX3 bootloader uses three pins of FX3 to determine the booting options.

As shown in Table 1, the three pins available can be driven high (1) or low (0) or no connection (F - Float).

The float detection capability of bootloader provides 3^3 = 27 possible combinations that can be used as boot options.

This application note discusses the boot options highlighted in Table 1.

SPI EEPROM Boot

PMODE Pins: 0F1

Features

FX3 boots from SPI EEPROM devices through a 4-wire SPI interface.

- EEPROM device sizes supported: 1 Kbit to 32 MBit
- Manufacturers supported: ATMEL, Microchip, and Numonyx.

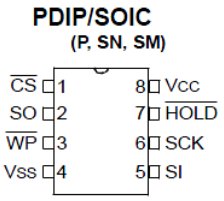
No other sizes are supported.

- Supported boot frequencies are ~10 MHz, ~20 MHz, and ~30 MHz:
  - SPI Speeds might be varied due to the round off on the SPI clock divider and clock input. When crystal clock is 26 MHz and 52 MHz, the internal PLL will be running at 416 MHz. When crystal clock is 19.2 MHz and 38.4 MHz, the internal PLL will be running at 384 MHz.
  - SPI Speeds supported for PLL_CLK = 416 MHz will be: 10.4 MHz, 20.8 MHz, and 34.66 MHz.
  - SPI Speeds supported for PLL_CLK = 384 MHz will be: 9.6 MHz, 19.2 MHz, and 32 MHz.
- Operating Voltage supported: 1.8 V, 2.5 V, and 3.3 V.
- Only one firmware image is stored on SPI EEPROM. No redundant Image allowed.
- Supports USB Fallback used for storing new VID/PID for USB booting. See the SPI EEPROM Boot with USB Fallback section in this application note for more information.

Storing Image on EEPROM

The FX3 bootloader supports a master SPI interface for external serial SPI EPROM devices. The serial SPI EEPROM can be used to store application specific code and data. The following diagram shows the pinout of a typical SPI EEPROM.

Figure 2. Pinout of a Typical SPI EEPROM



The SPI EEPROM interface consists of four active wires:

1. CS\: Chip Select
2. SO: Serial Data Output (Master In Slave Out)
3. SI: Serial Data Input (Master Out Slave In)
4. SCK: Serial Clock input
5. HOLD\ signal should tied to VCC while booting/ reading from EEPROM.

WP\ and HOLD\ signals should tied to VCC while writing the image onto EEPROM.

OBSOLETE

Boot Image Format

The firmware Image should be stored on to EEPROM as follows:

Binary Image Header	Length (16-bit)	Description
wSignature	1	Signature 2 bytes initialize with “CY” ASCII text
bImageCTL;	½	Bit0 = 0: execution binary file; 1: data file type Bit3:1 Not use when booting in SPI EEPROM Bit5:4(SPI speed): - 00: 10 MHz - 01: 20 MHz - 10: 30 MHz - 11: 40MHz (reserved). **Note**: Bootloader power-up default is set to 10 MHz and it will adjust the SPI speed if needed. The FX3 SPI hardware can only run up to 33 MHz, so the mode “11:40 MHz” is reserved. Bit7:6: Reserved should be set to zero
bImageType;	½	bImageType = 0xB0: normal FW binary image with checksum bImageType = 0xB1: Reserved for security image type bImageType = 0xB2: SPI boot with new VID and PID
dLength 0	2	1st section length, in long words (32-bit) When bImageType = 0xB2, the dLength 0 will contain PID and VID. Bootloader ignores the rest of the any following data.
dAddress 0	2	1st section address of Program Code. Note: Internal ARM address is byte addressable, so the address for each section should be 32-bit align
dData[dLength 0]	dLength 0*2	Image Code/Data must be 32-bit align.
…		More sections
dLength N	2	0x00000000 (Last record: termination section)
dAddress N	2	Should contain valid Program Entry (Normally, it should be the Startup code i.e. the RESET Vector) **Note:** if bImageCTL.bit0 = 1, the bootloader will not transfer the execution to this Program Entry. If bImageCTL.bit0 = 0, the bootloader will transfer the execution to this Program Entry: This address should be in ITCM area or SYSTEM RAM area Bootloader does not validate the Program Entry
dCheckSum	2	32-bit unsigned little endian checksum data will start from the 1st sections to termination section. The checksum will not include the dLength, dAddress, and Image Header

OBSOLETE

Example: The binary image file is stored in the SPI EEPROM in the following order:

Byte0: “C”
Byte1: “Y”
Byte2: bImageCTL
Byte3: bImageType
…..

Byte N: Checksum of Image

Important Points to Note:

- Bootloader default boot speed = 10 MHz; to change the speed from 10 MHz to 20 MHz the bImageCTL<5:4> should be set to 01.

Checksum Calculation

The bootloader computes the checksum when loading the binary image SPI EEPROM. If the checksum does not match the one in the Image, the bootloader will not transfer execution to the program entry.

The bootloader operates in little endian mode; for this reason, the checksum must also be computed in little endian mode.

32-bit unsigned little endian checksum data starts from the first sections to the termination section. The checksum will not include the dLength, dAddress, and Image Header.

OBSOLETE

First Example boot image is as follows:

Following Image is stored only at one section in System RAM of FX3 at location 0x40008000:

Location1: 0xB0 0x10 ’Y’ ’C’	//CY Signature, 20 MHz, 0xB0 Image
Location2: 0x00000004	//Image length = 4
Location3: 0x40008000	//1st section stored in SYSMEM RAM at 0x40008000
Location4: 0x12345678	//Image starts
Location5: 0x9ABCDEF1	
Location6: 0x23456789	
Location7: 0xABCDEF12	
Location8: 0x00000000	//Termination of Image
Location9: 0x40008000	//Jump to 0x40008000 on FX3 System RAM
Location 10: 0x7C048C04	//Checksum (0x12345678 + 0x9ABCDEF1 + 0x23456789 + 0xABCDEF12)

Second Example boot image is as follows:

Following Image is stored at two sections in System RAM of FX3 at location 0x40008000 and 0x40009000:

Location1: 0xB0 0x10 ’Y’ ’C’	//CY Signature, 20MHz, 0xB0 Image
Location2: 0x00000004	//Image length of section 1 = 4
Location3: 0x40008000	//1st section stored in SYSMEM RAM at 0x40008000
Location4: 0x12345678	//Image starts (Section1)
Location5: 0x9ABCDEF1	
Location6: 0x23456789	
Location7: 0xABCDEF12	//Section 1 ends
Location8: 0x00000002	//Image length of section 2 = 2
Location9: 0x40009000	//2nd section stored in SYSMEM RAM at 0x40009000
Location10: 0xDDCCBBAA	//Section 2 starts
Location11: 0x11223344	
Location12: 0x00000000	//Termination of Image
Location13: 0x40008000	//Jump to 0x40008000 on FX3 System RAM
Location 14: 0x6AF37AF2	//Checksum (0x12345678 + 0x9ABCDEF1 + 0x23456789 + 0xABCDEF12+
0xDDCCBBAA +0x11223344)	

Similarly you can have N sections of an Image stored using one boot image.

The following section shows the checksum sample code:

```
// Checksum sample code
DWORD dCheckSum, dExpectedCheckSum;
WORD wSignature, wLen;
DWORD dAddress, i;
DWORD dImageBuf[512*1024];

fread(&wSignature,1,2,input_file); // read signature bytes
if (wSignature != 0x5943)          // check 'CY' signature byte
{
   printf("Invalid image");
   return fail;
}
fread(&i, 2, 1, input_file);       // skip 2 dummy bytes
dCheckSum = 0;
while (1)
{
   fread(&dLength,4,1,input_file);  // read dLength
   fread(&dAddress,4,1,input_file); // read dAddress
   if (dLength==0) break;           // done
   // read sections
   fread(dImageBuf, 4, dLength,  input_file);
   for (i=0; i<dLength; i++) dCheckSum += dImageBuf[i];
}

// read pre-computed checksum data
fread(&dExpectedChecksum, 4, 1, input_file);

if  (dCheckSum != dExpectedCheckSum)
{
   printf("Fail to boot due to checksum error\n");
   return fail;
}
```

SPI EEPROM Boot with USB Fallback

In all USB Fallback ("=>USB"), USB is enumerated if 0xB2 boot is selected or an error occurs. After USB is enumerated, the external USB host can boot FX3 using USB Boot. SPI EEPROM boot with USB Fallback (SPI => USB) is also used to store Vendor Identification (VID) and Product Identification (PID) for USB Boot.

SPI EEPROM boot fails under the following conditions:

- SPI address cycle or data cycle error.
- Invalid signature on FX3 firmware. Invalid image type.
- A special image type is used to denote that instead of the FX3 firmware image, data on EEPROM is the VID and PID for USB boot. This helps in having a new VID and PID for USB Boot.

Note:

- On USB Boot, Bootloader supports ONLY USB High Speed and USB Full Speed. USB 3.0 Super speed is not supported.
- In 0xB2 boot option, the USB descriptor will use the customer defined VID and PID as part of the 0xB2 Image from SPI EEPROM.
- In case USB Fallback when any error occurs during the I^2C Boot, the USB descriptor uses the VID = 0x04B4 and PID = 0x00F3.
- The USB Device Descriptor will be reported as BUS-power that will consume around 200 mA. The FX3 chip itself consumes around 100 mA.

Example Image for boot with VID and PID

Location1: 0xB2 0x10 'Y' 'C'	//CY Signature, 20 MHz, 0xB2 Image
Location2: 0x04B40008	//VID = 0x04B4 | PID = 0x0008

Summary

The details of SPI Boot option supported by FX3 bootloader has been discussed in this Application Note. It also enables you to select an appropriate EEPROM device, storing boot image onto EEPROM, and booting FX3 over SPI interface.

OBSOLETE

About the Author

Name: Shruti Maheshwari
Title: Systems Engineer Senior
Contact: svrm@cypress.com

Document History

Document Title: EZ-USB® FX3 SPI Boot Option – AN70193

Document Number: 001-70193

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	3284799	SVRM	06/15/2011	New application note.
*A	3288571	SVRM	06/21/2011	Updated Document History
*B	3333553	SVRM	08/10/2011	Changed Figure 1, "I^2C EEPROM to SPI EEPROM" There was a typo in Second Example Boot Image. Fixed it to "2nd section stored in SYSMEM RAM at 0x40009000."
*C	4035072	HBM	06/20/2013	Obsolete spec. Content of this spec merged to Spec: 001-76405 (AN76405).

OBSOLETE

EZ-USB is a registered trademark of Cypress Semiconductor Corporation. All other trademarks or registered trademarks referenced herein are the property of their respective owners.

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709
Phone: 408-943-2600
Fax: 408-943-4730
http://www.cypress.com/

© Cypress Semiconductor Corporation, 2011. The information contained herein is subject to change without notice. Cypress Semiconductor Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in a Cypress product. Nor does it convey or imply any license under patent or other rights. Cypress products are not warranted nor intended to be used for medical, life support, life saving, critical control or safety applications, unless pursuant to an express written agreement with Cypress. Furthermore, Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress products in life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

This Source Code (software and/or firmware) is owned by Cypress Semiconductor Corporation (Cypress) and is protected by and subject to worldwide patent protection (United States and foreign), United States copyright laws and international treaty provisions. Cypress hereby grants to licensee a personal, non-exclusive, non-transferable license to copy, use, modify, create derivative works of, and compile the Cypress Source Code and derivative works for the sole purpose of creating custom software and or firmware in support of licensee product to be used only in conjunction with a Cypress integrated circuit as specified in the applicable agreement. Any reproduction, modification, translation, compilation, or representation of this Source Code except as specified above is prohibited without the express written permission of Cypress.

Disclaimer: CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Cypress reserves the right to make changes without further notice to the materials described herein. Cypress does not assume any liability arising out of the application or use of any product or circuit described herein. Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress' product in a life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

Use may be limited by and subject to the applicable Cypress software license agreement.