AN204773

F²MC-16FX, MCU Flash Programming and Boot ROM Protocol

This application note describes how to program a 16FX MB96300 series MCU with the serial interface and shows the Boot ROM communication protocol, so that the user can create his own programmer software.

Contents

1 Introduction

This application note describes how to program a 16FX MB96300 series MCU with the serial interface and shows the Boot ROM communication protocol, so that the user can create the own programmer software.

2 MCU Connection

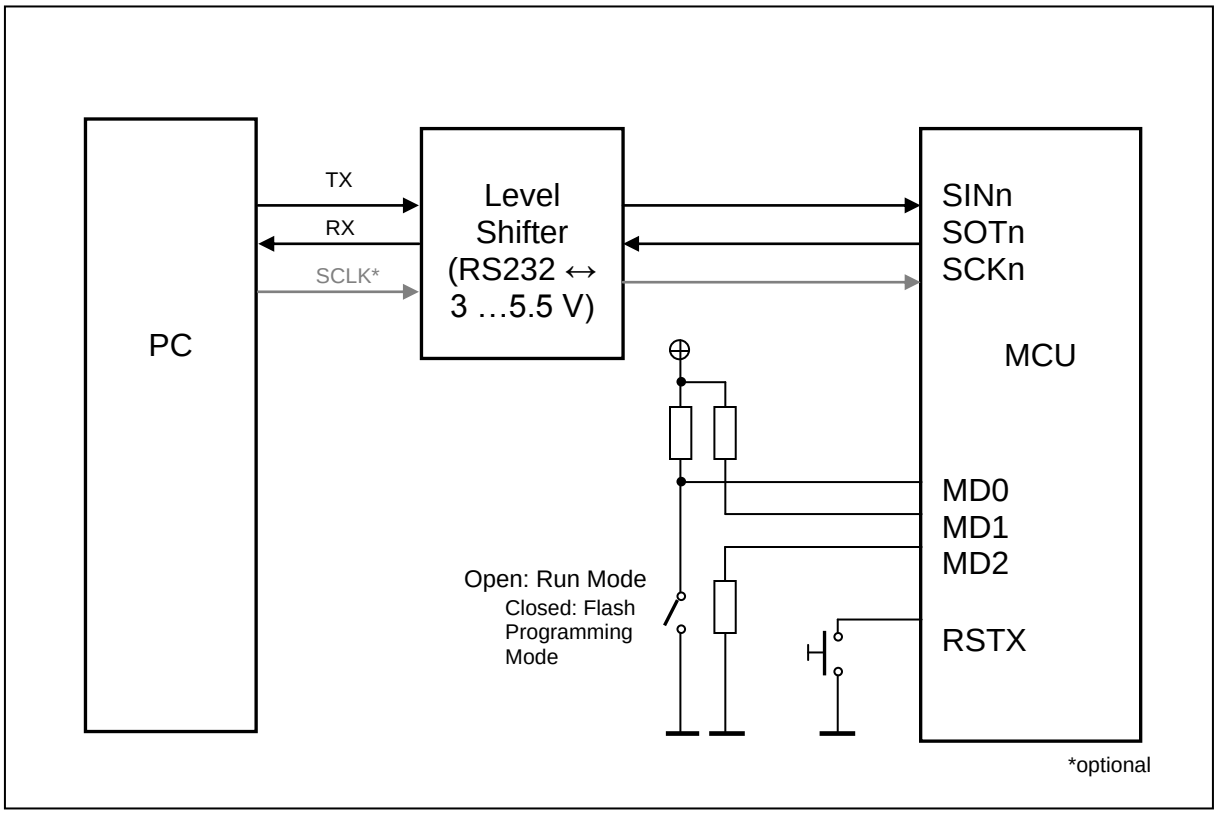
The Connection to the MCU

2.1 Serial Interface

For MCU Flash Programming a serial interface (USART) is used. The embedded Boot ROM allows the user to program the MCU in asynchronous and synchronous serial communication. The MCU has to be switched to its embedded Boot ROM serial programming mode for this.

The asynchronous programming can be done by a standard PC COM interface. The following Block Diagram shows how to connect a PC to a 16FX MB96300 series MCU.

Figure 1. Flash Programming Connection Block Diagram



Please see application note 002-04772 (AN204772 - F²MC-16FX Family, Hardware Set Up) for details of a minimum MCU system and hardware recommendations.

Multiple USARTs are usable for serial Flash programming. The Boot ROM scans the USARTs after Reset in serial Flash programming mode and tries to establish communication. See Datasheet for device specific USART channels. In most cases USART0 to USART3 are supported.

3 Using Cypress Starter Kits

Jumper Settings of Cypress Starter Kit Boards

3.1 Starter Kits

For almost each MB96300 series device a Cypress Starter Kit board exists. These boards have at least one connecter to a USART of the supported MCU. The following tables shows, how to configure the jumpers and set the mode pins of several boards for serial Flash Programming communication.

3.1.1 FLASH-CAN-100P-340 V1.1, V2.0

This board is applicable for the MB96F34x series in QFP-100 package.

Jumpers

Jumper	USART0	USART2
JP1a	closed	x
JP1b	open	x
JP1c	open	x
JP2a	closed	x
JP2b	open	x
JP2c	open	x
JP44	closed	x
JP31*	open	x
JP8a	x	closed
JP8b	x	open
JP8c	x	open
JP6a	x	closed
JP6b	x	open
JP6c	x	open
JP32*	x	open

* When not using RTS/CTS for MCU-Reset

Mode Pins (S2)

Operation Mode	1	2	3	4	5	6	7	8
Normal Run	OFF	OFF	ON	x	x	x	x	x
Flash Programming	ON	OFF	ON	x	x	x	x	x

3.1.2 SK-96380-120PMT V1.0

This board is applicable for the MB96F38x series in LQFP-120 package.

Jumper	USART0	USART2
JP21	1-2	x
JP23*	open	x
JP25	1-2	x
JP27	1-2	x
JP28	open	x
JP37	x	1-2
JP38*	x	open
JP39	x	1-2
JP40	x	1-2
JP42	x	open
JP33*	open	open

* When not using RTS/CTS for MCU-Reset

Mode Pins (S1)

Operation Mode	1	2	3	4
Normal Run	OFF	OFF	ON	x
Flash Programming	ON	OFF	ON	x

3.1.3 Other Cypress Starter Kits

Please refer to the corresponding user guides for USART jumper settings.

4 Cypress F²MC-16FX Flash Programmer

How to Use the F²MC-16FX Flash Programmer

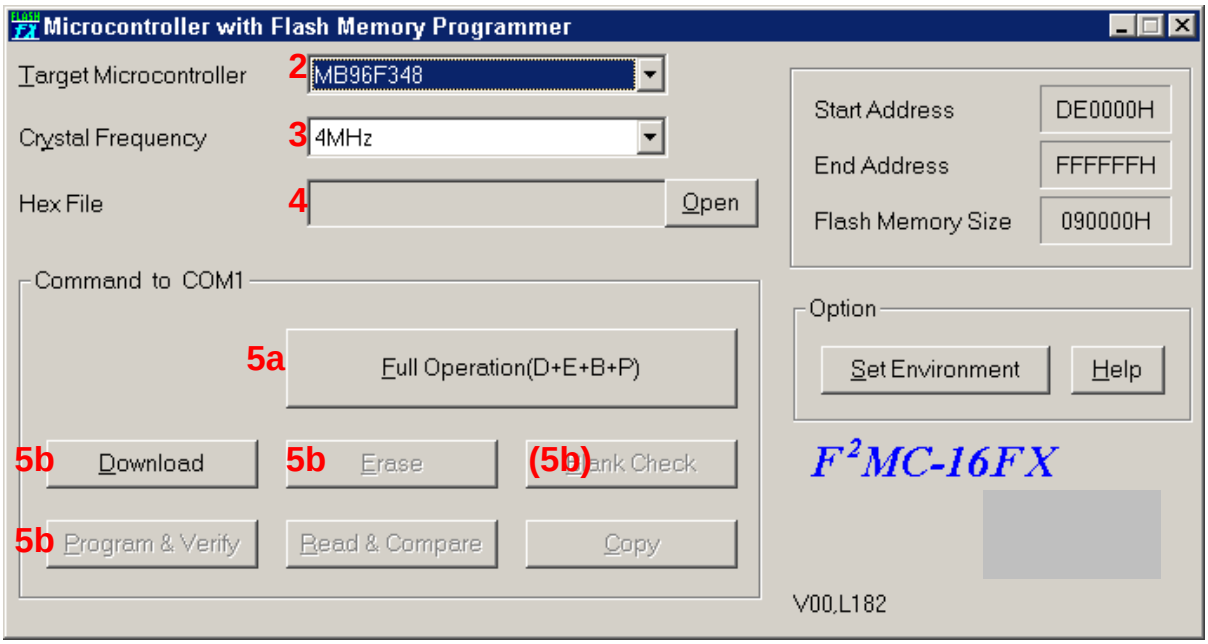
4.1 Introduction

The Cypress F²MC-16FX Flash Programmer is a freeware tool for programming 16FX Flash MCU. It uses standard COM (RS232) communication ports of the PC.

> Please note that the F²MC-16FX Flash Programmer Software is for developing and evaluation purposes only – it is not approved for mass production.

4.2 Usage

After starting the Flash Programmer its user interface will look like the following:



The programming is done by a programming kernel, which is downloaded to the RAM of the MCU by the embedded Boot ROM. This kernel performs the further steps of the programming and communicates with the PC. This is described in more detail in chapter 6.

For programming a Flash MCU the user should proceed as follows.

1. Set the MCU to flash programming mode (3) and perform a Reset.
2. Select “Target Microcontroller” to choose the MCU.
3. Enter used Crystal Frequency on the target board.
4. Browse to the Hex-File (*project_name*.mhx) of the project.

At this step the user has two different possibilities to proceed.

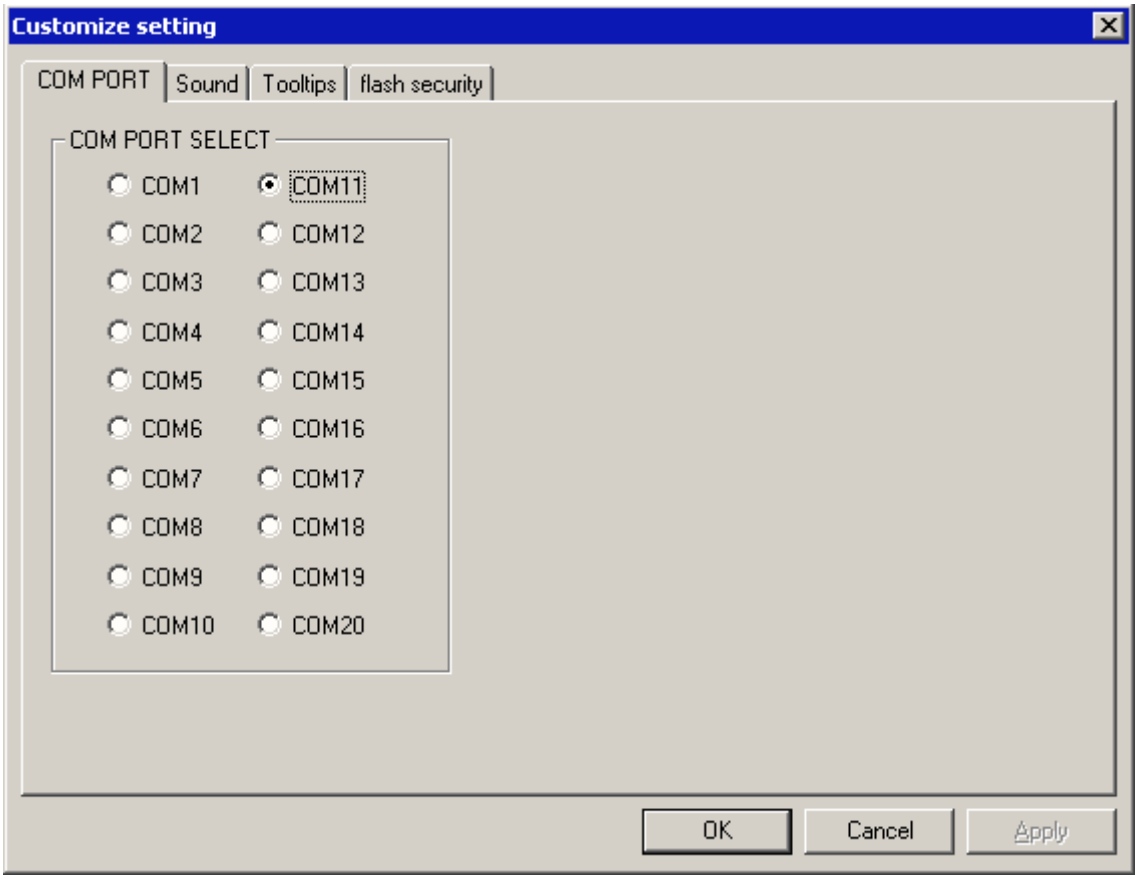
5a. Full Operation: This performs all steps for programming the MCU. It includes “Download”, “Erase”, “Blank Check”, and “Program & Verify”.

5b. Single Steps: “Download”, “Erase” (“Blank Check” optionally), and “Program & Verify” manually.

A *chip erase* command is performed regardless of using method of 5a or 5b.

After successful programming, the MCU should be switched to normal RUN mode via the mode pins (3) and a reset will start the programmed software.

With “Set Environment” several settings can be done. The most important is the options for COM port selection.



Due to the rising number of virtual COM ports using USB-R232 converters, the number of available COM ports of the Flash Programmer now was increased to 20.

5 External Clock Frequencies and Baud Rates

Configuration of Clock Frequency and Communication Baud Rate.

5.1 External Clock

Using a 16FX MB96300 series MCU the used baud rate for serial Flash Programming can vary in a wide range by using a fix external clock frequency. The Boot ROM uses a protocol, which measures the incoming baud rate and adjusts its own with it.

Without external clock, Boot ROM uses the internal RC clock.

5.2 Asynchronous Baud Rate Configuration Tables

The following table gives an overview of several baud rate ranges for several external frequencies depending on the used MCU. The shown baud rates are used only for establishing the communication. Further communication may use higher baud rates. This depends on the user application.

5.2.1 MB9631x/MB96F32x/MB9633x/MB96F34x/MB96F35x/MB9637x/MB96F38x/MB9639x

External Frequency	Minimum Baud Rate	Maximum Baud Rate
3.5 MHz	4800	19200
4 MHz	4800	19200
5 MHz	4800	38400
6 MHz	4800	38400
8 MHz	9600	76800
10 MHz	9600	115200
12 MHz	9600	115200
16 MHz	19200	153600

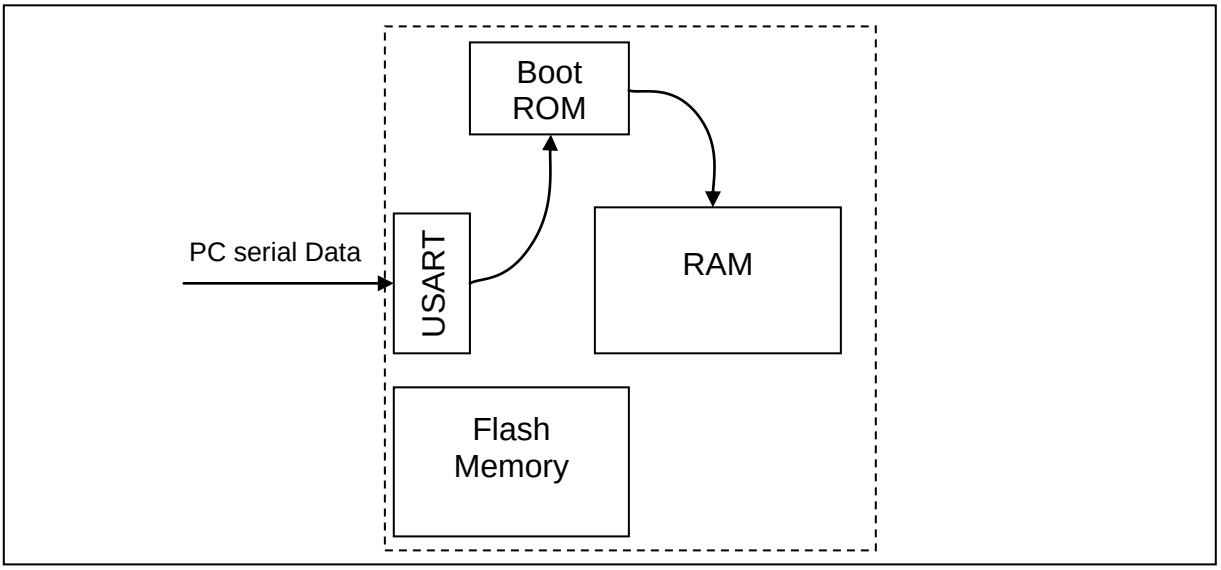
6 Flash MCU Boot ROM Protocol

Serial Communication to Program Flash Memory

6.1 Introduction

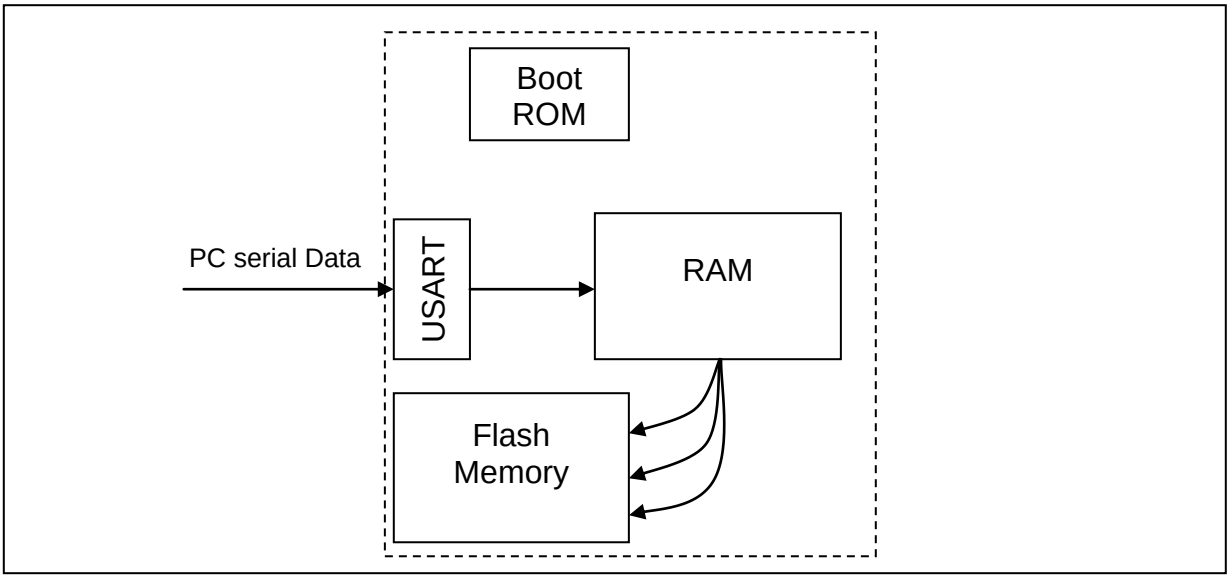
The 16FX MB96300 series Flash MCUs contain a fixed firmware (Boot ROM) program that supports a proprietary protocol to allow download of a user program (kernel) to on-chip RAM memory (Figure 2).

Figure 2. Download of User Program to RAM



The user program is then able to manipulate the on-chip Flash Memory as required (Figure 3). It communicates directly via the USART to the connected PC, which sends the commands.

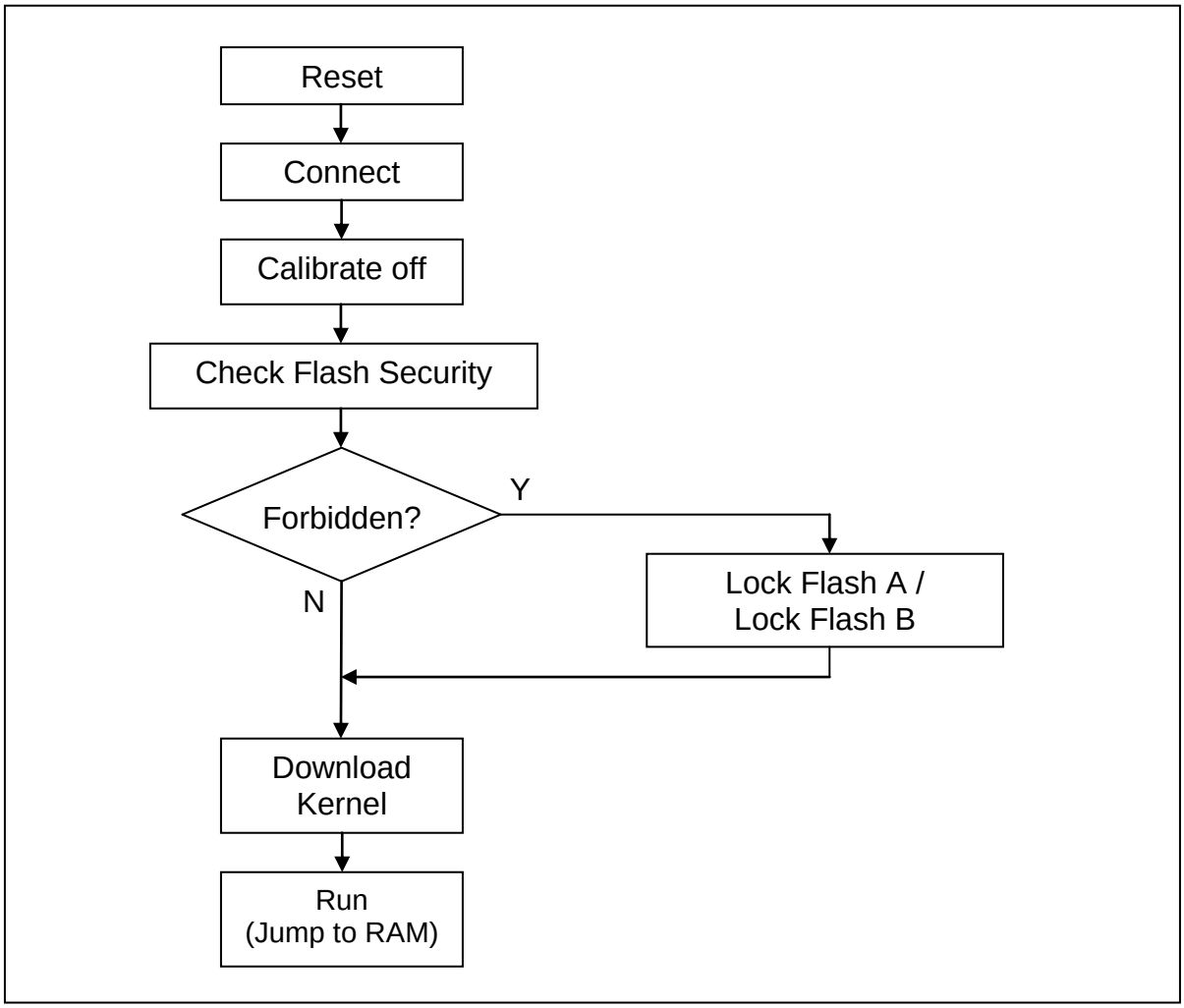
Figure 3. Programming Flash Memory from RAM



The Boot ROM supports asynchronous (Duplex and K-Line) and synchronous serial communication. Except the auto calibration phase for the asynchronous mode(s) at the beginning, the ROM protocol is identical for both communication types.

6.2 Boot ROM Sequence for asynchronous (Duplex) Communication

The following flow chart shows the communication from a PC to the Boot ROM. Please note, that the data format is 8N2 for PC and 8N1 for MCU, when using asynchronous communication. For the baud rate to use, please refer to chapter 5.2



The PC has to send 8N2 format at “connect” and MCU sends back 8N1 format. The PC can switch to 8N1 then, if there are problems receiving 8N1 format.

If the Flash Memory is secured, the user program (Kernel) should perform a chip erase and then await an external Reset. After this a new connection is performed. This is also the procedure using the Cypress Flash Programmer software.

6.2.1 Connection

In the connection phase the PC sends a sequence to the MCU. This sequence should use the following data.

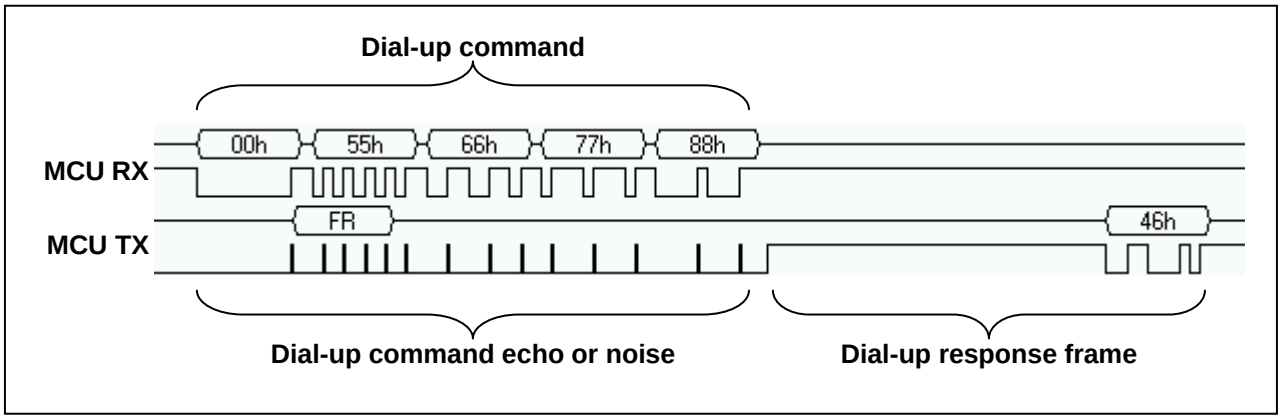
Calibration Header		Dial-up Pattern		
Synch Break	**Synch Byte**	**Dial-up ID 0**	**Dial-up ID 1**	**Dial-up ID 2**
0x00	0x55	0x66	0x77	0x88
PC → MCU				

If the connection was successful, the MCU set its USART-SOT pin from Hi-Z to “1” and sends a response byte:

Response
0x46
MCU → PC

Note: Until 32 bit times before 0x46 is sent by MCU, PC may receive random data.

Figure 4. Example of asynchronous dial-up command, command echo (intermediate framing error) and final response code



6.2.2 Calibrate off

On crystal systems the calibration header is not needed for each command, so the calibration header can be switched off. The switch off command itself has to be sent with the calibration header of course.

Calibration Header		Header	Payload	Checksum
Synch Break	**Synch Byte**	**Calibrate off Command**	**Mode 0**	
0x00	0x55	0x87	0x00	0x78
PC → MCU				

The MCU sends back a response byte.

Response
0x69
MCU → PC

6.2.3 Check Flash Security

The Flash Security is read by the `READ8` command to dummy address `0xFF0000`.

Header					Checksum
Cmd	Addr0	Addr1	Addr2	Count	
0x90	0x00	0x00	0xFF	0x01	0x6E
PC → MCU					

If the Flash memory is secured (“Forbidden”), the response of the MCU will be:

Response
0x96
MCU → PC

In the unsecured case, the response will be:

Header	Payload	Checksum
Response	Data	
0x69	*0xXX*	*0xYY*
PC → MCU		

Please note that *0xXX* represents the (random) content of the Flash address `0xFF0000` and *0xYY* the accordant checksum.

For further details please refer to application note 002-05551 (AN205551 - F²MC-16FX Family, Flash Security)

6.2.4 Locking Flash Memory

In case of secured Flash memory every memory command is forbidden. The PC has to send the `LOCK` command to allow read and write access to RAM. The flash memory reading is not possible by hardware. Once the Flash memory is locked, the contents cannot be unlocked anymore. However, instead of `LOCK`, the `UNLOCK` (`0x0A`) command grants access to Flash, if the key is correct.

Header		Checksum
Cmd	Select	
0x0C	0xFF	*0xYY*
PC → MCU		

The command is accepted, if the MCU sends the following response:

Response
0x69
MCU → PC

6.2.5 Download Kernel

The Kernel is downloaded by the `WRITE OFF` command to the MCU RAM. The start address depends on the size of the device RAM. Please see the datasheet for details. By default it should be start at the upper 2K bytes of the RAM: `0x007A20`.

Header					Premature Check-sum	Payload			Check-sum
Cmd	Addr0	Addr1	Addr2	Count N		Data0	. . .	Data N-1	
0x12	*0xXX*	*0xYY*	0x00	*0xZZ*	*0xTT*	*0xUU*	. . .	*0xVV*	*0xWW*
PC → MCU									

Note: The payload (N) can contain a maximum of 256 bytes. In this case *Count N* has to be set to `0x00`.

Note: The checksum (*0xWW*) is calculated over all bytes of the `WRITE OFF` command and the *premature* checksum (*0xTT*) is calculated over the 5 previous header bytes only.

Also note that the checksum (`0xWW`) can also be calculated over only the payload bytes, but starting with `0xFF` as initial value.

In case of no errors the MCU will respond with the usual single byte response:

Response
0x69
MCU → PC

6.2.6 Run (Jump to RAM)

After downloading the kernel, it can be executed by the RUN command.

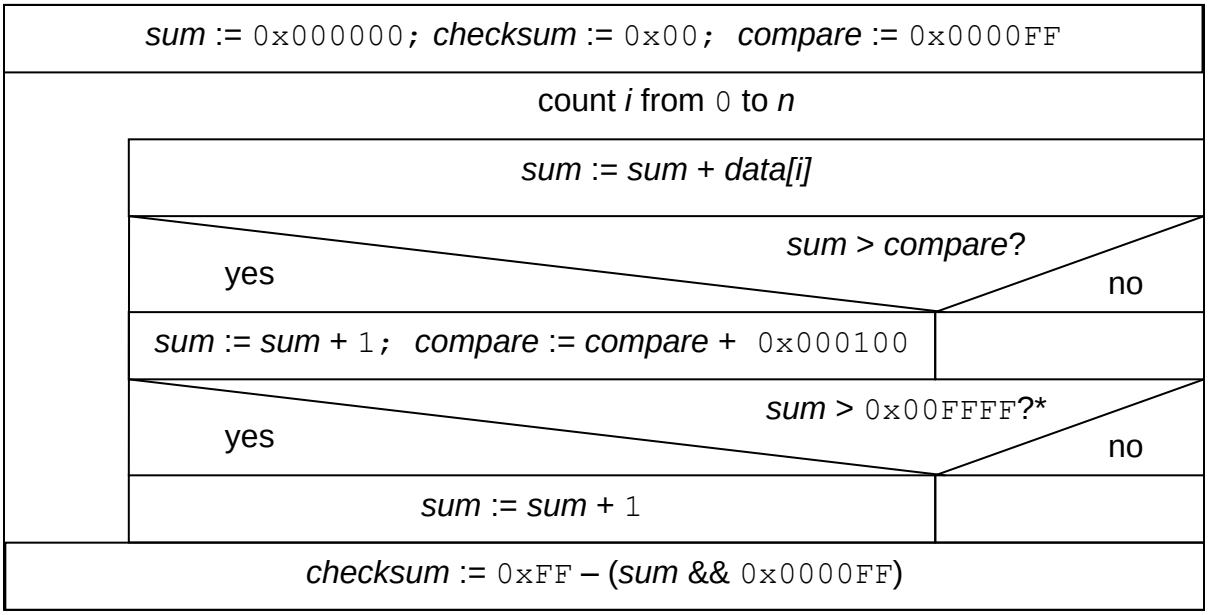
Header	Payload			Checksum
Cmd	Addr0	Addr1	Addr2	
0x9F	*0xXX*	*0xYY*	*0xZZ*	*0xWW*
PC → MCU				

The MCU responds with the usual single byte response before performing the jump to the denoted address.

Response
0x69
MCU → PC

6.2.7 Checksum

The checksum is calculated with the following program flow. It includes all data bytes (`0`…*n*) including command header, but except the calibration header (`0x00`, `0x55`):



* A 2nd compare counter is not needed because 64K overflow is only possible once for n ~ 255 and all data = 0xFF plus header.

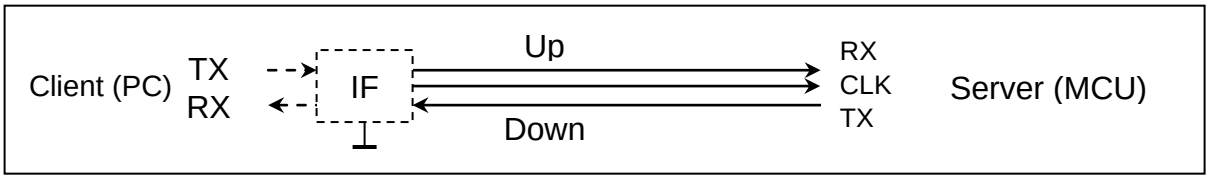
6.3 Boot ROM Sequence for Synchronous (Duplex) Communication

6.3.1 Synchronous Communication Preface

A synchronous connection uses common timing by common clock but separate media. The clock is mastered by client. An active clock shifts bytes to both, upstream and downstream.

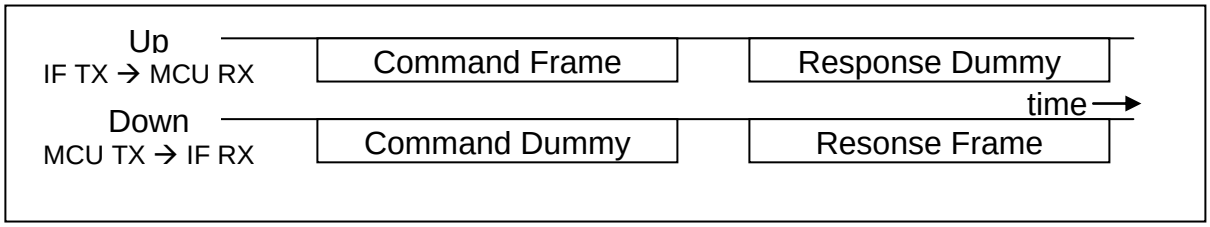
If the client is not capable of mastering the connection, an interface converter has to be used.

Figure 5. Separated synchronous up/down stream wires



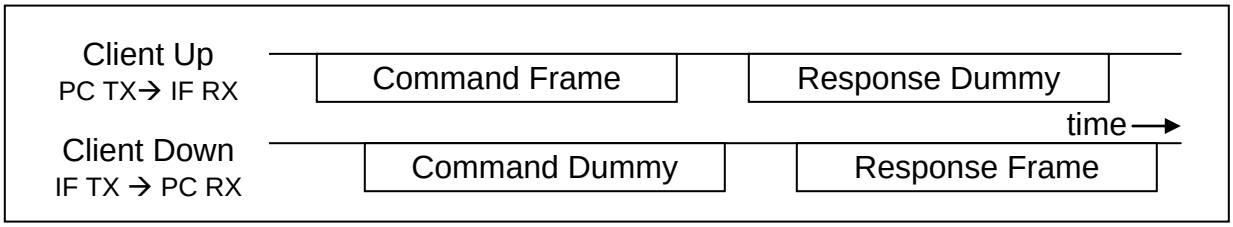
In order to receive the response from server (clock slave), the client (clock master) needs to send dummy bytes according to the expected number of response bytes. Vice versa, the server (MCU) will send dummy bytes (`NULL`), when it receives the command frame. The client needs to consider the reception of those dummy bytes. If the server does not explicitly send command dummy and response frame, the client receives random data.

Figure 6. Separated synchronous up/down stream wires



A client, which uses an asynchronous-to-synchronous converter, also needs to consider delayed reception. Command dummy and response will be delayed due to converter and different throughput on busses.

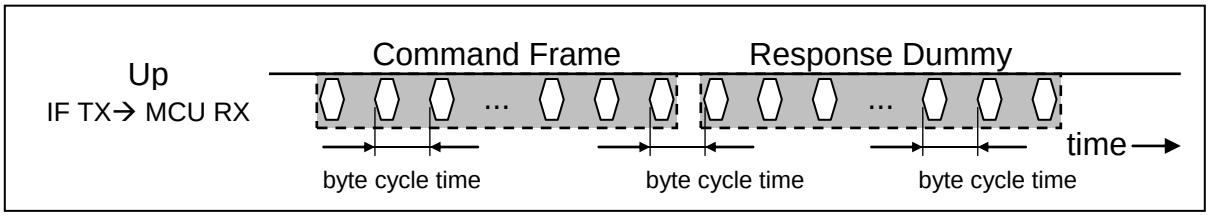
Figure 7. Delayed Command Dummy of interface converter



Since the start-up performance of the MCU is limited by slow system clock, the throughput of synchronous connection must be limited too. Instead of sending continuous bytes, a byte cycle time is applied (e.g. by interface converter). It might be dynamically adapted to the frequency of core bus clock `CLKB`. However, during dial-up, a worst-case default shall be used.

The byte cycle time is also sufficient to separate command frame and response dummy.

Figure 8. Byte-cycle-time-wise transmission of synchronous frames



Note: The minimum byte cycle time should not be less than the minimum asynchronous data frame time after dial-up connection. During Dial-up the byte cycle time should be between 1.5 ms and 2.5 ms.

6.3.2 Boot ROM Sequence for Synchronous (Duplex) Communication

The sequence is the same described for asynchronous communication (6.2).

6.3.3 Connection

In the connection phase the PC sends a sequence to the MCU. This sequence should use the following data. It is the same pattern like in asynchronous communication but without `0x00` and `0x55` data at the beginning.

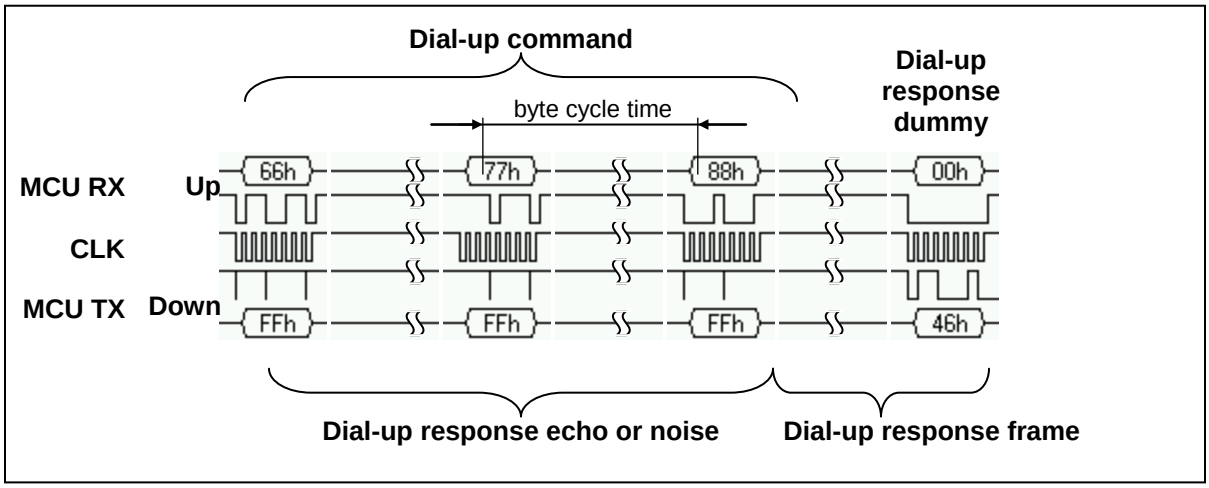
Dial-up Pattern		
Dial-up ID 0	**Dial-up ID 1**	**Dial-up ID 2**
`0x66`	`0x77`	`0x88`
PC → MCU		

If the connection was successful, the MCU sends a response byte. Please be aware, that for any response the PC has to send a dummy byte.

Response
`0x46`
MCU → PC

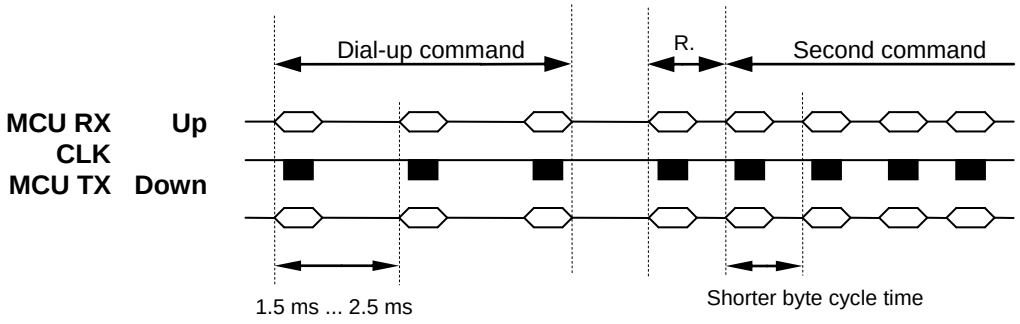
Note: At any dial-up data byte sent by the PC before `0x46` is responded by MCU, PC may receive random data.

Figure 9. Example of synchronous dial-up



Note: The byte cycle time during dial-up command should be between 1.5 ms and 2.5 ms unless of the used baud rate. After successful connection the byte cycle time can be shorter but should not undercut the minimum frame time of the maximum (asynchronous) baud rate.

Figure 10. Example of byte cycle times



6.3.4 Command Sequences

The command sequences are the same like for asynchronous communication. Please refer to the descriptions above (start from 6.2.2).

7 Serial Programming USART

This sections shows the USARTs supported for serial communication mode

For each 16FX MCU series, one of up to 4 USARTs can be used to establish a connection in serial communication mode. Sometimes, a USART is offered on different pins, the default one and the relocated one. The relocated ones show the suffix R at the pin names. Example: USART 2 uses pins SIN2, SOT2, and SCK2. When relocation is enabled, the pins SIN2_R, SOT2_R, and SCK2_R are used. Please note that relocated USARTs will be shown separately.

Series	USART
MB96310	2, 7R, 8R
MB96320	2, 3, 7R, 8R
MB96330	0, 1, 2, 3
MB96340	0, 1, 2, 3
MB96350	2, 3, 7R, 8R
MB96370	0, 1, 2, 3
MB96380	0, 1, 2
MB96390	0, 1, 2

8 Appendix

Further Information

8.1 Related Documents

- 002-05551(AN205551-F²MC-16FXFamily,FlashSecurity)
  This application note shows the Flash Security mechanism.
- 002-05556(AN205556-F²MC-16FXFamily,FlashMemory)
  This application note shows the architecture of the Flash memory and how to program it.
- 002-04772 (AN204772 - F²MC-16FX Family, Hardware Set Up)

  This application note describes a minimum 16FX hardware system and gives further hardware design recommendations.

9 Document History

Document Title: AN204773 - F²MC-16FX, MCU Flash Programming and Boot ROM Protocol

Document Number: 002-04773

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	-	NOFL	08/22/2006	Initial release
			06/25/2007	Release version
			07/17/2007	Corrections done, synchronous protocol added
			07/18/2007	Synchronous Dial-up sequence notes added
			10/24/2007	Updated Related Documents; added supported USART of MCU Series
			11/09/2007	Clarify serial baud rates
			12/05/2007	Typos corrected, checksum calculation notes to WRITE OFF command added
			04/08/2009	Typos corrected, table in chapter 7 updated
			09/08/2009	Checksum calculation changed to Nassi-Shneiderman diagram
			10/29/2009	Checksum diagram updated again
			06/16/2010	16FX family baud rates updated, new screenshot of programmer
			04/03/2012	Baud rate table 5.2.1 corrected (19200 Baud @ 4 MHz)
*A	5080508	NOFL	04/06/2016	Migrated Spansion Application Note MCU-AN-300224-E-V21 to Cypress format
*B	5821285	AESATMP8	07/17/2017	Updated logo and Copyright.

Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturer's representatives, and distributors. To find the office closest to you, visit us at Cypress Locations.

Products

ARM® Cortex® Microcontrollers	cypress.com/arm
Automotive	cypress.com/automotive
Clocks & Buffers	cypress.com/clocks
Interface	cypress.com/interface
Internet of Things	cypress.com/iot
Memory	cypress.com/memory
Microcontrollers	cypress.com/mcu
PSoC	cypress.com/psoc
Power Management ICs	cypress.com/pmic
Touch Sensing	cypress.com/touch
USB Controllers	cypress.com/usb
Wireless Connectivity	cypress.com/wireless

PSoC® Solutions

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP | PSoC 6

Cypress Developer Community

Forums | WICED IOT Forums | Projects | Videos | Blogs | Training | Components

Technical Support

cypress.com/support

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709