

F²MC-16FX FAMILY
16-BIT MICROCONTROLLER
ALL SERIES

**MCU FLASH PROGRAMMING
AND BOOT ROM PROTOCOL**

APPLICATION NOTE

Revision History

Date	Issue
2006-08-22	V1.0: First Version; MWi
2007-06-25	V1.1; Release version; MWi
2007-07-17	V1.2; Corrections done, synchronous protocol added; MWi
2007-07-18	V1.3; Synchronous Dial-up sequence notes added; MWi
2007-10-24	V1.4; update Related Documents; add supported USART of MCU Series; PHu
2007-11-09	V1.5; clarify serial baudrates; PHu
2007-12-05	V1.6; typos corrected, checksum calculation notes to <code>WRITE OFF</code> command added; MWi
2009-04-08	V1.7; typos corrected, table in chapter 7 updated; MWi
2009-09-08	V1.8; Checksum calculation changed to Nassi-Shneiderman diagram; MWi
2009-10-29	V1.9; Checksum diagram updated again; MWi

This document contains 22 pages.

Warranty and Disclaimer

To the maximum extent permitted by applicable law, Fujitsu Microelectronics Europe GmbH restricts its warranties and its liability for **all products delivered free of charge** (eg. software include or header files, application examples, target boards, evaluation boards, engineering samples of IC's etc.), its performance and any consequential damages, on the use of the Product in accordance with (i) the terms of the License Agreement and the Sale and Purchase Agreement under which agreements the Product has been delivered, (ii) the technical descriptions and (iii) all accompanying written materials. In addition, to the maximum extent permitted by applicable law, Fujitsu Microelectronics Europe GmbH disclaims all warranties and liabilities for the performance of the Product and any consequential damages in cases of unauthorised decompiling and/or reverse engineering and/or disassembling. **Note, all these products are intended and must only be used in an evaluation laboratory environment.**

1. Fujitsu Microelectronics Europe GmbH warrants that the Product will perform substantially in accordance with the accompanying written materials for a period of 90 days from the date of receipt by the customer. Concerning the hardware components of the Product, Fujitsu Microelectronics Europe GmbH warrants that the Product will be free from defects in material and workmanship under use and service as specified in the accompanying written materials for a duration of 1 year from the date of receipt by the customer.
2. Should a Product turn out to be defect, Fujitsu Microelectronics Europe GmbH's entire liability and the customer's exclusive remedy shall be, at Fujitsu Microelectronics Europe GmbH's sole discretion, either return of the purchase price and the license fee, or replacement of the Product or parts thereof, if the Product is returned to Fujitsu Microelectronics Europe GmbH in original packing and without further defects resulting from the customer's use or the transport. However, this warranty is excluded if the defect has resulted from an accident not attributable to Fujitsu Microelectronics Europe GmbH, or abuse or misapplication attributable to the customer or any other third party not relating to Fujitsu Microelectronics Europe GmbH.
3. To the maximum extent permitted by applicable law Fujitsu Microelectronics Europe GmbH disclaims all other warranties, whether expressed or implied, in particular, but not limited to, warranties of merchantability and fitness for a particular purpose for which the Product is not designated.
4. To the maximum extent permitted by applicable law, Fujitsu Microelectronics Europe GmbH's and its suppliers' liability is restricted to intention and gross negligence.

NO LIABILITY FOR CONSEQUENTIAL DAMAGES

To the maximum extent permitted by applicable law, in no event shall Fujitsu Microelectronics Europe GmbH and its suppliers be liable for any damages whatsoever (including but without limitation, consequential and/or indirect damages for personal injury, assets of substantial value, loss of profits, interruption of business operation, loss of information, or any other monetary or pecuniary loss) arising from the use of the Product.

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

Contents

REVISION HISTORY	2
WARRANTY AND DISCLAIMER	3
CONTENTS	4
1 INTRODUCTION	6
2 MCU CONNECTION	7
2.1 Serial Interface	7
3 USING FUJITSU STARTER KITS	8
3.1 Starter Kits	8
3.1.1 FLASH-CAN-100P-340 V1.1, V2.0	8
3.1.1.1 Jumpers	8
3.1.1.2 Mode Pins (S2)	8
3.1.2 SK-96380-120PMT V1.0	9
3.1.2.1 Mode Pins (S1)	9
4 FUJITSU F²MC-16FX FLASH PROGRAMMER	10
4.1 Introduction	10
4.2 Usage	10
5 EXTERNAL CLOCK FREQUENCIES AND BAUD RATES	12
5.1 External Clock	12
5.2 Asynchronous Baud Rate Configuration Tables	12
5.2.1 MB96F32x/MB96F34x/MB96F35x/MB96F38x	12
6 FLASH MCU BOOT ROM PROTOCOL	13
6.1 Introduction	13
6.2 Boot ROM Sequence for asynchronous (Duplex) Communication	14
6.2.1 Connection	15
6.2.2 Calibrate off	15
6.2.3 Check Flash Security	16
6.2.4 Locking Flash Memory	16
6.2.5 Download Kernel	17
6.2.6 Run (Jump to RAM)	17
6.2.7 Checksum	18
6.3 Boot ROM Sequence for Synchronous (Duplex) Communication	18
6.3.1 Synchronous Communication Preface	18
6.3.2 Boot ROM Sequence for Synchronous (Duplex) Communication	19

6.3.3	Connection	19
6.3.4	Command Sequences.....	20
7	SERIAL PROGRAMMING USART.....	21
8	APPENDIX.....	22
8.1	Related Documents.....	22

1 Introduction

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

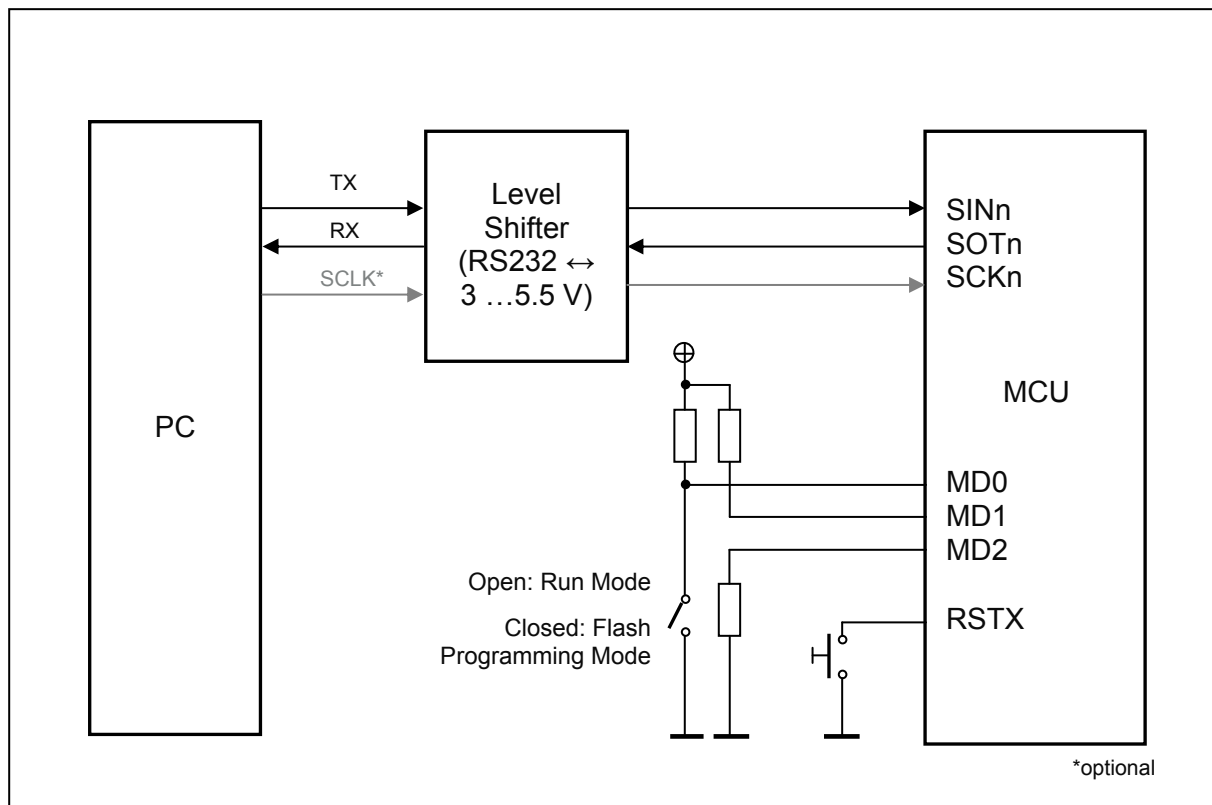


Figure 2-1: Flash Programming Connection Block Diagram

Please see application note *AN-390223-16FX_HW_SETUP* 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 Fujitsu Starter Kits

JUMPER SETTINGS OF FUJITSU STARTER KIT BOARDS

3.1 Starter Kits

For almost each 16FX device a Fujitsu Starter Kit board exists. These boards have at least one connector 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.

3.1.1.1 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

3.1.1.2 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

3.1.2.1 Mode Pins (S1)

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

4 Fujitsu F²MC-16FX Flash Programmer

HOW TO USE THE F²MC-16FX FLASH PROGRAMMER

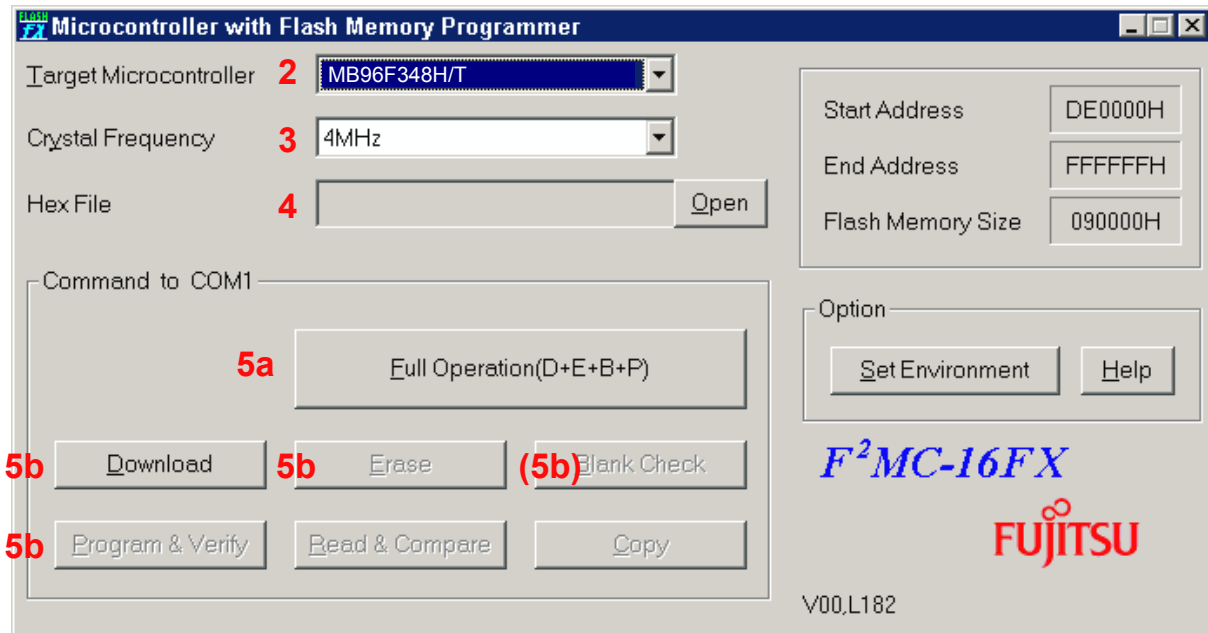
4.1 Introduction

The Fujitsu 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.

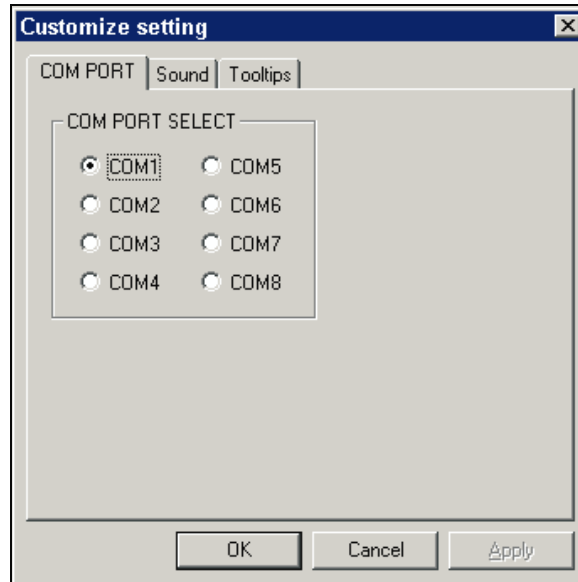
5a. Full Operation: This performs all steps for programming the MCU. It includes “Download”, “Erase”, “Blank Check”, and “Program & Verify”.

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.



5 External Clock Frequencies and Baud Rates

CONFIGURATION OF CLOCK FREQUENCY AND COMMUNICATION BAUD RATE

5.1 External Clock

Using a 16FX 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 MB96F32x/MB96F34x/MB96F35x/MB96F38x

External Frequency	Minimum Baud Rate	Maximum Baud Rate
3.5 MHz	4800	19200
4 MHz	4800	38400
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 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 6-1).

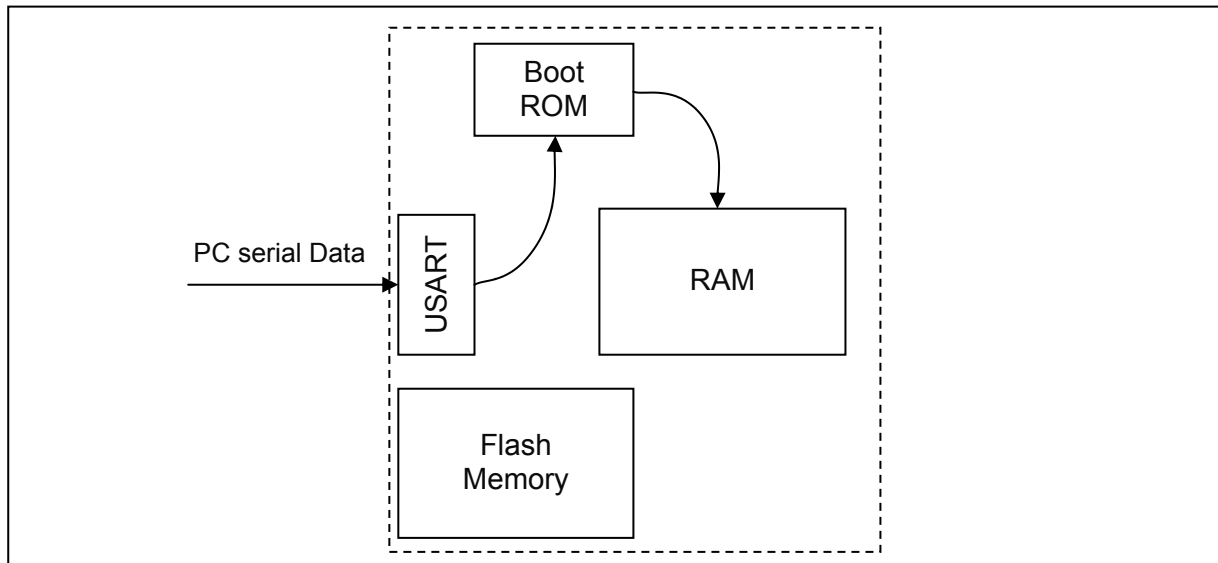


Figure 6-1: Download of User Program to RAM

The user program is then able to manipulate the on-chip Flash Memory as required (Figure 6-2). It communicates directly via the USART to the connected PC, which sends the commands.

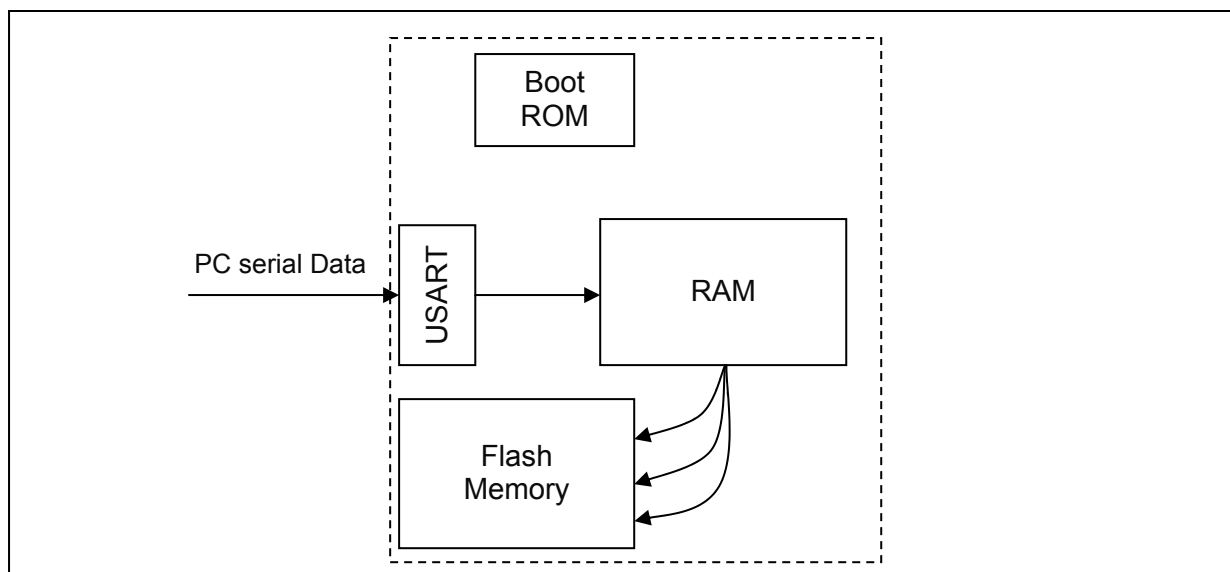
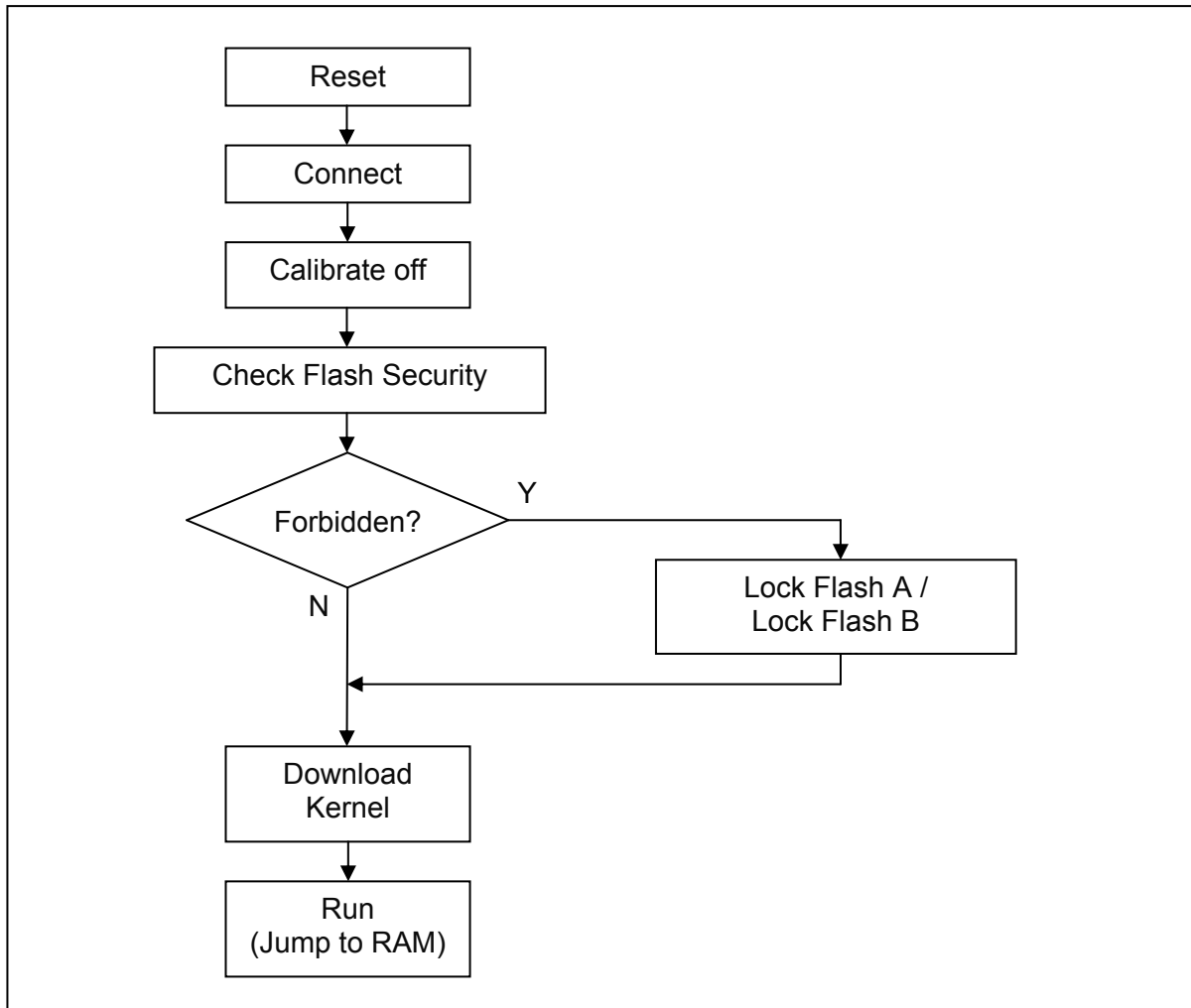


Figure 6-2: Programming Flash Memory from RAM

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 Fujitsu 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.

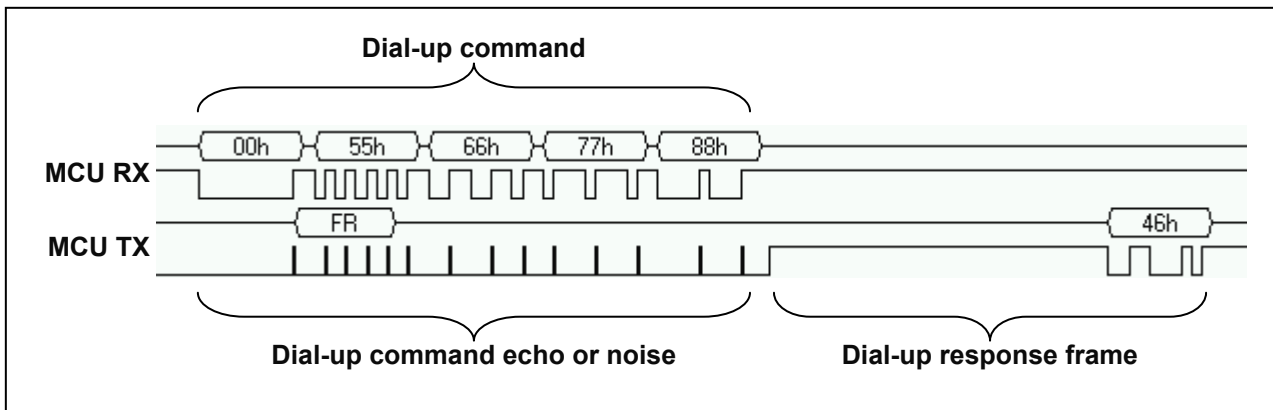


Figure 6-3: Example of asynchronous dial-up command, command echo (intermediate framing error) and final response code

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 *mcu-an-300213-e-16fx_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`.

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

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

Note that 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
<code>0x69</code>
MCU → PC

6.2.6 Run (Jump to RAM)

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

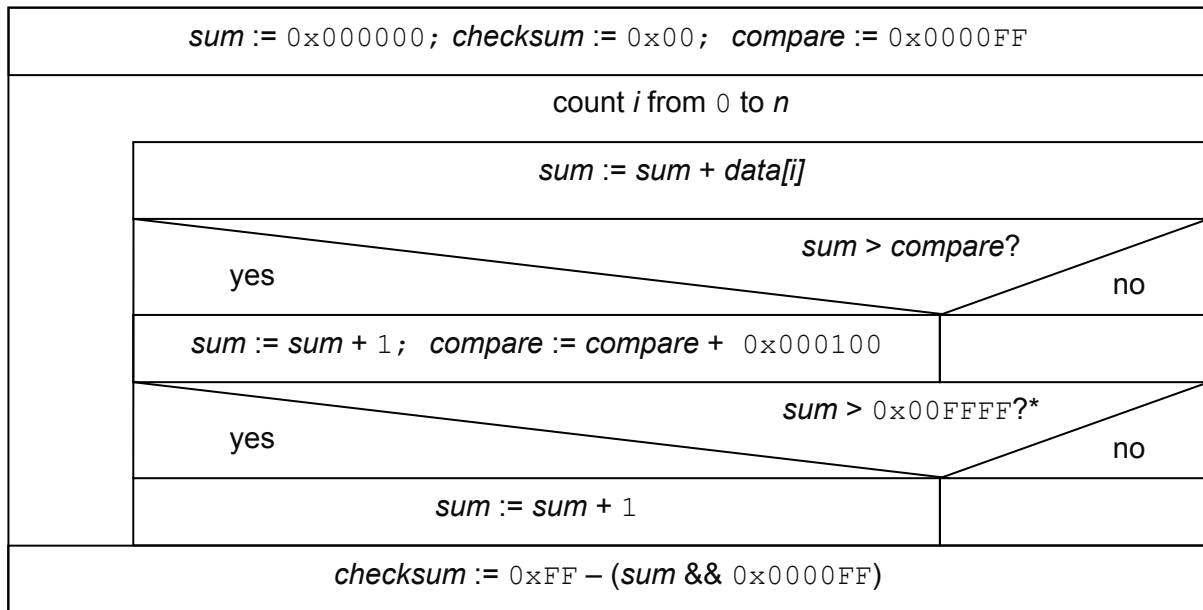
Header	Payload			Checksum
Cmd	Addr0	Addr1	Addr2	
<code>0x9F</code>	<code>0xXX</code>	<code>0xYY</code>	<code>0xZZ</code>	<code>0xWW</code>
PC → MCU				

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

Response
<code>0x69</code>
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 $n \sim 255$ 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, up stream and down stream.

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

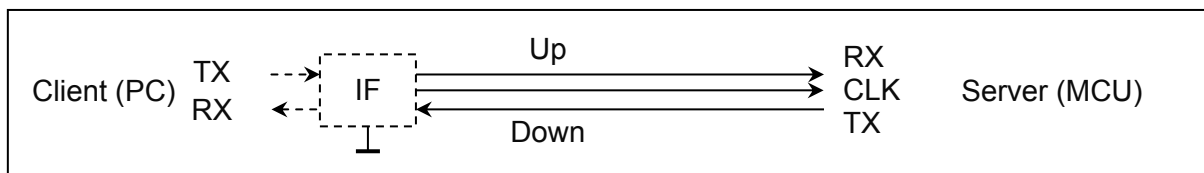


Figure 6-4: Separated synchronous up/down stream wires

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.

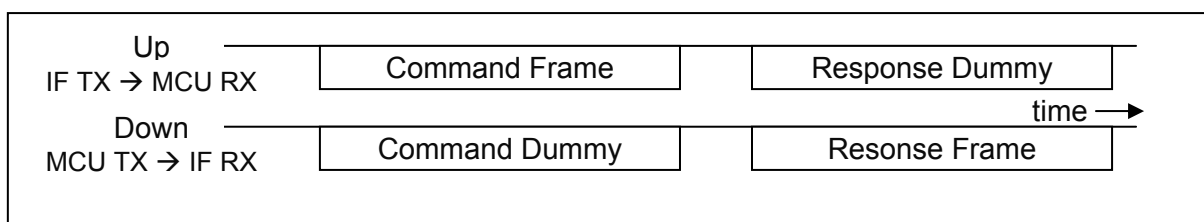


Figure 6-5: Separated synchronous up/down stream wires

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.

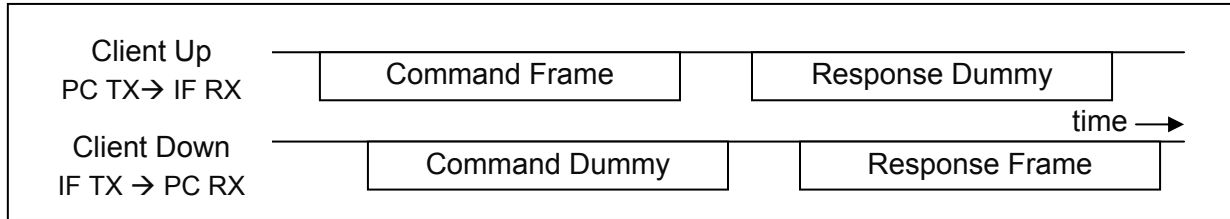


Figure 6-6: Delayed Command Dummy of interface converter

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.

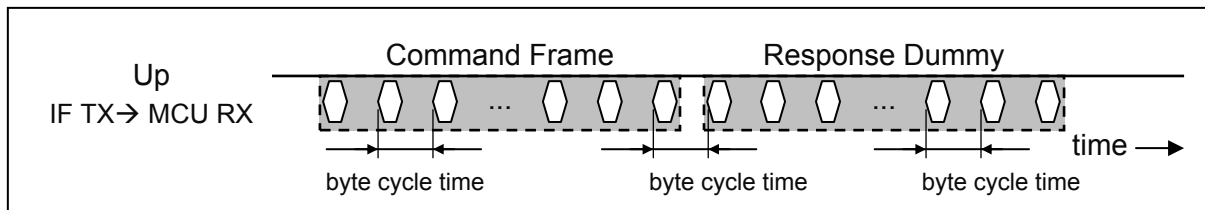


Figure 6-7: byte-cycle-time-wise transmission of synchronous frames

Note, that 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.

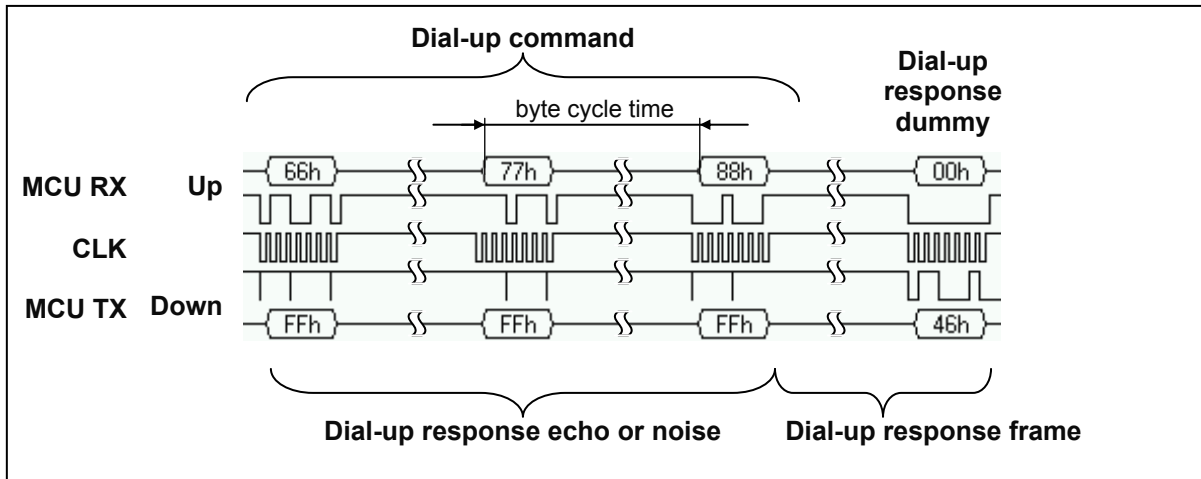


Figure 6-8: Example of synchronous dial-up

Note, that 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.

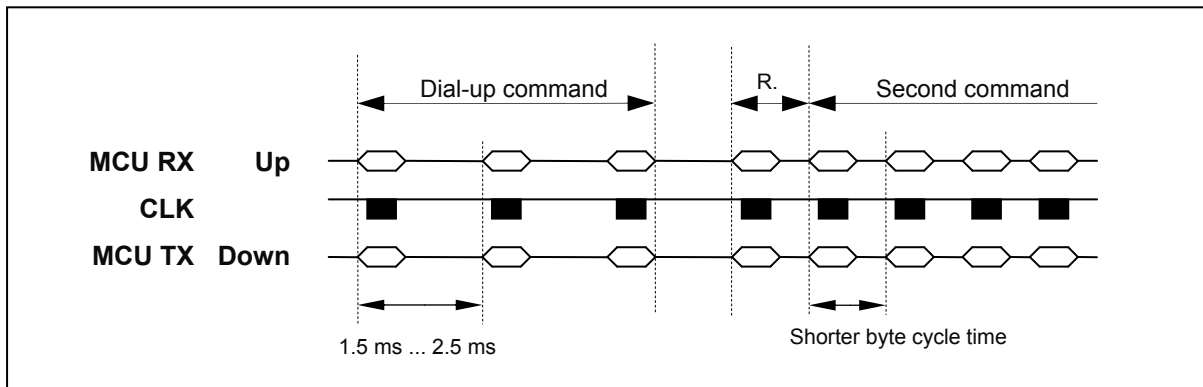


Figure 6-9: Example of byte cycle times

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

- *mcu-an-300213-e-16fx_flash_security*
This application note shows the Flash Security mechanism.
- *mcu-an-300218-e-16fx_flash*
This application note shows the architecture of the Flash memory and how to program it.
- *mcu-an-300223-e-16fx_hw_setup*
This application note describes a minimum 16FX hardware system and gives further hardware design recommendations.