

# PMA71xx/PMA51xx

SmartLEWIS™ MCU

RF Transmitter FSK/ASK 315/434/868/915 MHz  
Embedded 8051 Microcontroller with 10 bit ADC  
Embedded 125 kHz ASK LF Receiver

## Application Note

I<sup>2</sup>C protocol for programming the PMA71xx/PMA51xx  
Revision 1.1, 2010-04-23

**Edition 2010-04-23**

**Published by  
Infineon Technologies AG  
81726 Munich, Germany**

**© 2010 Infineon Technologies AG  
All Rights Reserved.**

### **Legal Disclaimer**

The information given in this document shall in no event be regarded as a guarantee of conditions or characteristics. With respect to any examples or hints given herein, any typical values stated herein and/or any information regarding the application of the device, Infineon Technologies hereby disclaims any and all warranties and liabilities of any kind, including without limitation, warranties of non-infringement of intellectual property rights of any third party.

### **Information**

For further information on technology, delivery terms and conditions and prices, please contact the nearest Infineon Technologies Office ([www.infineon.com](http://www.infineon.com)).

### **Warnings**

Due to technical requirements, components may contain dangerous substances. For information on the types in question, please contact the nearest Infineon Technologies Office.

Infineon Technologies components may be used in life-support devices or systems only with the express written approval of Infineon Technologies, if a failure of such components can reasonably be expected to cause the failure of that life-support device or system or to affect the safety or effectiveness of that device or system. Life support devices or systems are intended to be implanted in the human body or to support and/or maintain and sustain and/or protect human life. If they fail, it is reasonable to assume that the health of the user or other persons may be endangered.

## Table of Contents

	<b>Table of Contents</b>	3
	<b>List of Tables</b>	5
<b>1</b>	<b>Introduction</b>	6
<b>2</b>	<b>General information</b>	6
2.1	PMA71xx/PMA51xx PROGRAMMING Mode selection	6
2.2	Setup for I <sup>2</sup> C communication	7
2.3	I <sup>2</sup> C protocol	7
2.4	FLASH memory organization	8
<b>3</b>	<b>PROGRAMMING Mode commands</b>	8
3.1	FLASH Write Line	9
3.2	FLASH Read Line	9
3.3	FLASH Erase	9
3.4	FLASH Check Erase Status	10
3.5	FLASH Set Code Lock (Lockbyte 2)	11
3.6	FLASH Set User Data Sector Lock (Lockbyte 3)	11
3.7	Read Status	11
<b>4</b>	<b>Programming sequence for PMA71xx/PMA51xx code sector</b>	12
<b>5</b>	<b>Estimation of device programming time</b>	14
5.1	FLASH command transaction times	14
5.2	FLASH programming scenario 1	17
5.3	FLASH programming scenario 2	17
<b>6</b>	<b>Application example (PMA Starterkit as programmer)</b>	18
	<b>List of Figures</b>	19
	<b>References</b>	20



## List of Tables

Table 1	Electrical parameters for I <sup>2</sup> C communication	9
Table 2	FLASH Erase: Sector byte	12
Table 3	FLASH Check Erase Status: Sector byte	12
Table 4	FLASH Check Erase Status: Status byte	13
Table 5	Read Status: Status byte	14
Table 6	Assumptions for the calculation of FLASH command transaction times	16
Table 7	FLASH Check Erase Status	16
Table 8	FLASH Erase	17
Table 9	FLASH Write Line	17
Table 10	FLASH Read Line	17
Table 11	FLASH Read Status	18
Table 12	FLASH Set User Data Sector Lock	18
Table 13	Programming time estimation with status verification every 256 bytes	19
Table 14	Programming time estimation with verification of each byte	19

## List of Figures

Figure 1	Operating mode selection of the PMA71xx/PMA51xx after Reset . . . . .	8
Figure 2	PMA71xx/PMA51xx hardware setup for I <sup>2</sup> C communication . . . . .	9
Figure 3	I <sup>2</sup> C protocol . . . . .	10
Figure 4	PMA71xx/PMA51xx FLASH memory organization . . . . .	10
Figure 5	Nomenclature for I <sup>2</sup> C communication . . . . .	11
Figure 6	FLASH Write Line command . . . . .	11
Figure 7	FLASH Read Line command . . . . .	11
Figure 8	FLASH Erase command . . . . .	12
Figure 9	FLASH Erase: Sector byte . . . . .	12
Figure 10	FLASH Check Erase Status command . . . . .	12
Figure 11	FLASH Check Erase Status: Sector byte . . . . .	12
Figure 12	FLASH Check Erase Status: Status byte . . . . .	13
Figure 13	FLASH Set Lockbyte 3 command . . . . .	13
Figure 14	Read Status command . . . . .	13
Figure 15	Read Status: Status byte . . . . .	13
Figure 16	Possible programming sequence for PMA71xx/PMA51xx code sector . . . . .	15
Figure 17	Usage of PMA Starterkit as programmer . . . . .	20

## Revision History: 2010-04-23, Revision 1.1

### Previous Revision: 1.0

Page	Subjects (major changes since last revision)
	Initial Version
	Restructured document: Table of Contents, List of Tables and List of Figures were moved to the beginning of the document.
11	Description of FLASH Write Line command changed: A read back verification of the written line is executed within each FLASH Write Line instruction.

### Trademarks of Infineon Technologies AG

APOXI™, BlueMoon™, COMNEON™, CONVERGATE™, COSIC™, C166™, CROSSAVE™, CanPAK™, CIPOST™, CoolMOS™, CoolSET™, CORECONTROL™, DAVE™, EasyPIM™, EconoBRIDGE™, EconoDUAL™, EconoPACK™, EconoPIM™, EiceDRIVER™, EUPEC™, FCOS™, FALC™, GEMINAX™, GOLDMOS™, HITFET™, HybridPACK™, ISAC™, ISOFACE™, IsoPACK™, my-d™, MIPAQ™, ModSTACK™, NovalithIC™, OmniTune™, OmniVia™, OPTIVERSE™, OptiMOS™, ORIGAT™, PROFET™, PRO-SIL™, PrimePACK™, RASIC™, ReverSave™, SCEPTRE™, SEROCCO™, SICOFI™, SMARTi™, SMINT™, SOCRATES™, SatRIC™, SensoNor™, SINDRION™, SmartLEWIS™, SIEGET™, TrueNTRY™, TEMPFET™, TriCore™, thinQ!™, TRENCHSTOP™, VINAX™, VINETIC™, X-GOLD™, XMM™, X-PMU™, XPOSYS™, XWAY™.

### Other Trademarks

AMBA™, ARM™, MULTI-ICE™, PRIMECELL™, REALVIEW™, THUMB™ of ARM Limited, UK. AUTOSAR™ is licensed by AUTOSAR development partnership. Bluetooth™ of Bluetooth SIG Inc. CAT-iq™ of DECT Forum. COLOSSUS™, FirstGPS™ of Trimble Navigation Ltd. EMV™ of EMVCo, LLC (Visa Holdings Inc.). EPCOS™ of Epcos AG. FLEXGO™ of Microsoft Corporation. FlexRay™ is licensed by FlexRay Consortium. HYPERTERMINAL™ of Hilgraeve Incorporated. IEC™ of Commission Electrotechnique Internationale. IrDA™ of Infrared Data Association Corporation. ISO™ of INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. MATLAB™ of MathWorks, Inc. MAXIM™ of Maxim Integrated Products, Inc. MICROTEC™, NUCLEUS™ of Mentor Graphics Corporation. Mifare™ of NXP. MIPI™ of MIPI Alliance, Inc. MIPS™ of MIPS Technologies, Inc., USA. muRata™ of MURATA MANUFACTURING CO. OmniVision™ of OmniVision Technologies, Inc. Openwave™ Openwave Systems Inc. RED HAT™ Red Hat, Inc. RFMD™ RF Micro Devices, Inc. SIRIUS™ of Sirius Sattelite Radio Inc. SOLARIS™ of Sun Microsystems, Inc. SPANSION™ of Spansion LLC Ltd. Symbian™ of Symbian Software Limited. TAIYO YUDEN™ of Taiyo Yuden Co. TEAKLITE™ of CEVA, Inc. TEKTRONIX™ of Tektronix Inc. TOKO™ of TOKO KABUSHIKI KAISHA TA. UNIX™ of X/Open Company Limited. VERILOG™, PALLADIUM™ of Cadence Design Systems, Inc. VLYNQ™ of Texas Instruments Incorporated. VXWORKS™, WIND RIVER™ of WIND RIVER SYSTEMS, INC. ZETEX™ of Diodes Zetex Limited.

Last Trademarks Update 2009-05-27



## 1 Introduction

This document describes how the PMA71xx/PMA51xx can be programmed via the I<sup>2</sup>C interface. In [Chapter 2](#) the PROGRAMMING Mode selection, the setup for I<sup>2</sup>C communication, the I<sup>2</sup>C protocol and the FLASH memory organization are explained. [Chapter 3](#) shows the structure of the I<sup>2</sup>C commands that can be used in the PROGRAMMING Mode. A possible procedure how data can be loaded into the code sector, verified and protected against undesired read outs is illustrated in [Chapter 4](#). An estimation of the device programming time is given in [Chapter 5](#). Finally [Chapter 6](#) describes how the PMA71xx/PMA51xx Starterkit can be used as programmer.

## 2 General information

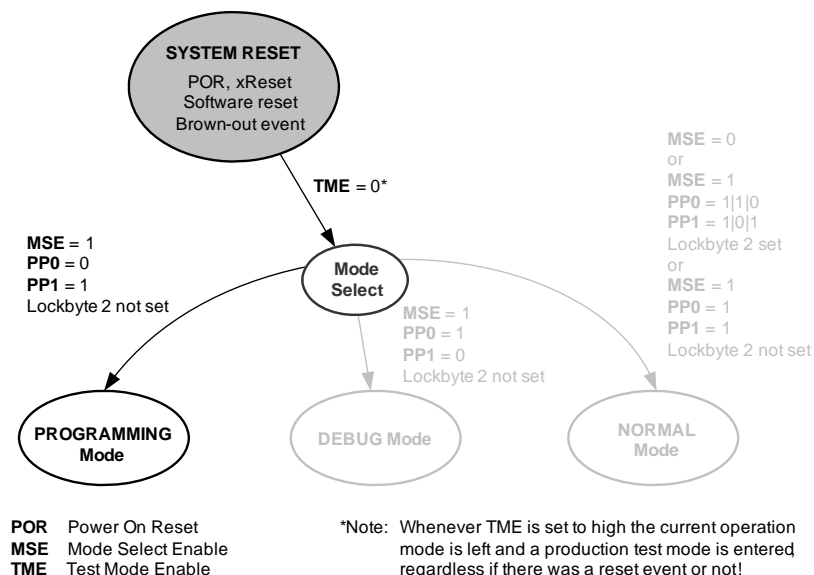
This chapter describes how the PROGRAMMING Mode of the PMA71xx/PMA51xx can be entered. Furthermore the setup for I<sup>2</sup>C communication, the I<sup>2</sup>C protocol which is used for communication with the programmer and the FLASH organization of the PMA71xx/PMA51xx is shown.

### 2.1 PMA71xx/PMA51xx PROGRAMMING Mode selection

The PMA71xx/PMA51xx can be operated in three different operating modes

- PROGRAMMING Mode
- NORMAL Mode
- DEBUG Mode

The Mode Select is entered after the System Reset expires. The levels on the I/O pins PP0 and PP1 are latched by the system controller and read by the operating system to determine the mode of operation of the device. [Figure 1](#) shows the settings for the selection of the three different operating modes. The MSE, PP0 and PP1 levels must not change after reset release during the whole  $t_{MODE}$  (defined in [\[1\]](#)) period. Furthermore, the available operating modes are also dependent upon the state of the FLASH code lock (Lockbyte 2), which is used to secure the FLASH code sector against undesired read outs.

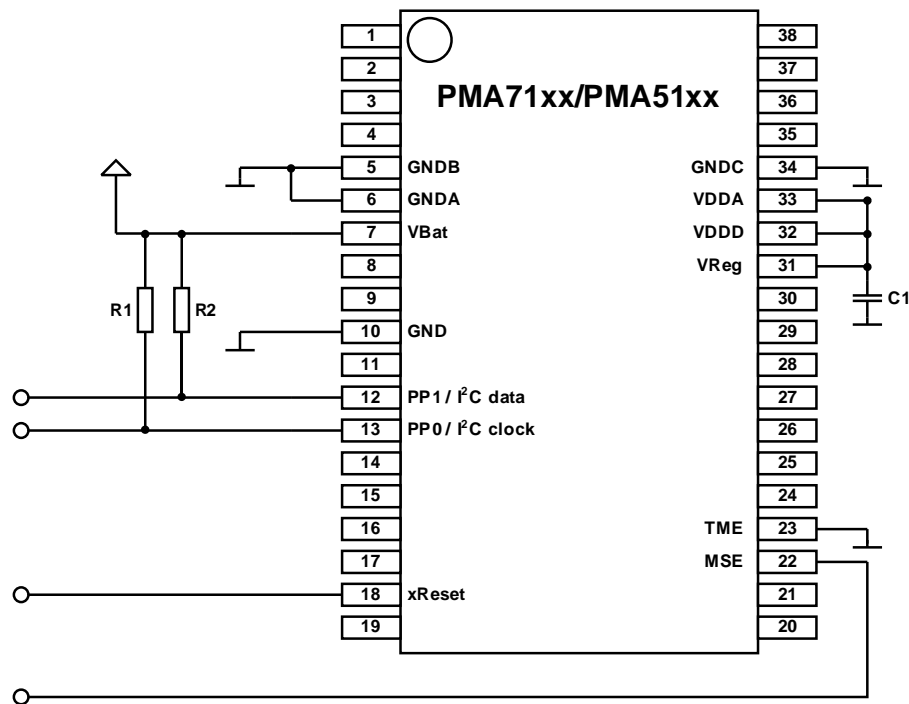


**Figure 1** Operating mode selection of the PMA71xx/PMA51xx after Reset



## 2.2 Setup for I<sup>2</sup>C communication

**Figure 2** and **Table 1** show the setup of the PMA71xx/PMA51xx required for I<sup>2</sup>C communication in PROGRAMMING Mode. Note that the resistors are required for reliable I<sup>2</sup>C communication for all modes other than NORMAL Mode, so these may be implemented as part of the programming equipment, rather than included as part of the application circuitry.



**Figure 2** PMA71xx/PMA51xx hardware setup for I<sup>2</sup>C communication

**Table 1** Electrical parameters for I<sup>2</sup>C communication

Parameter	Min.	Typ.	Max.	Unit	Remarks
VBat				VDC	refer to [1]
R1		3.3k		Ohm	
R2		3.3k		Ohm	
C1		200n		Farad	
I <sup>2</sup> C datarate			400k	Bit/s	
I <sup>2</sup> C level low				VDC	refer to [1]
I <sup>2</sup> C level high				VDC	refer to [1]

## 2.3 I<sup>2</sup>C protocol

**Figure 3** shows the protocol which is used for I<sup>2</sup>C communication. Data are transmitted bit by bit on line SDA in coaction with line SCL. To start communication a master device generates a start condition. Subsequently the address and data are transferred each followed by an acknowledge from the receiving device. The data on the SDA line must be stable during the high period of the clock and may only be changed during SCL low phase. After all data have been transferred the master closes transmission with a stop condition. To continue data transfer (with changed read/write behavior) the master device could generate a restart condition immediately.

## PROGRAMMING Mode commands

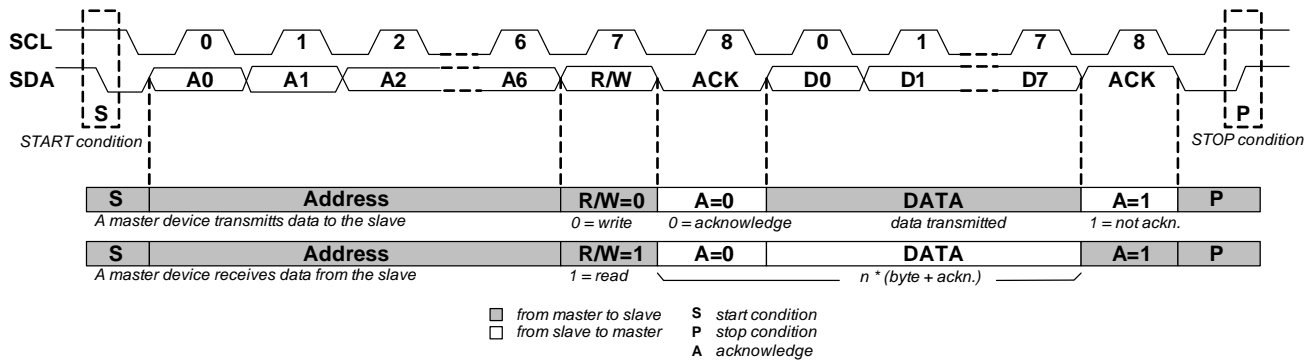


Figure 3 I²C protocol

## 2.4 FLASH memory organization

Figure 4 shows the FLASH memory organization of the PMA71xx/PMA51xx. The Flash Configuration + ID sector is written at the end of production test. Also the Lockbyte 1 is set and prevents a write access to this sector. User Data Sector I + II and the Code Sector can be written using PROGRAMMING Mode commands described in Chapter 3.

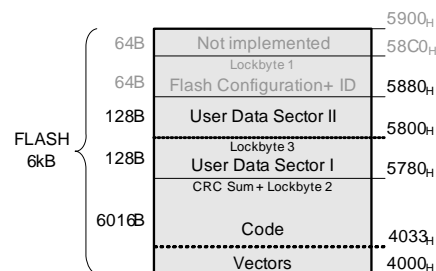




Figure 4 PMA71xx/PMA51xx FLASH memory organization

## 3 PROGRAMMING Mode commands

In PROGRAMMING Mode the PMA71xx/PMA51xx is accessible as a slave using the I²C interface. The device is operating using the 12 MHz RC HF Oscillator as clock source. To avoid programming failures all PROGRAMMING Mode commands are protected by a 16 bit CRC at the end of each command. The checksum has to be calculated over all bytes in the command excluding the PMA71xx/PMA51xx I²C device address. For the calculation of the CRC checksum the CCITT polynomial  $1021_H (x^{16} + x^{12} + x^5 + 1)$  and the 16 bit start value  $FFFF_H$  have to be used.

Figure 5 shows the nomenclature used for the description of the I²C commands.

## PROGRAMMING Mode commands

	from programmer to PMA	<b>S</b>	start condition
	from PMA to programmer	<b>P</b>	stop condition
<b>CRCH</b>	MSB of CRC checksum	<b>SR</b>	restart condition or stop / start condition
<b>CRCL</b>	LSB of CRC checksum	<b>A</b>	acknowledge
<b>Data0-31</b>	data which is written into / read from FLASH	<b>nA</b>	not acknowledge
<b>Pause</b>	Time where no communication is allowed	<b>Sector</b>	selection of the sector
<b>Status</b>	Status byte		

**Figure 5 Nomenclature for I<sup>2</sup>C communication**

### 3.1 FLASH Write Line

The FLASH Write Line command writes 32 bytes to the FLASH. The FLASH Code Sector and FLASH User Data Sectors can be written using this command. The start address has to be a multiple of 20<sub>H</sub>. As shown in [Figure 4](#) FLASH address range 4000<sub>H</sub> to 587F<sub>H</sub> is accessible.

This command should only be used if the FLASH line is fully erased. If an already programmed FLASH line gets overwritten (without being erased first) the resulting data is undefined. After the stop condition (P) is received the data is programmed into the FLASH. During the programming time incoming I<sup>2</sup>C commands are not acknowledged. Programming time is specified in [\[1\]](#). [Figure 6](#) shows the structure of the FLASH Write Line command. A read back verification of the written line is executed within each FLASH Write Line instruction. However, in order to see if this write command was successful, a [Read Status](#) command must be issued, or a [FLASH Read Line](#) command may be used to read back the stored values.

*Note:*

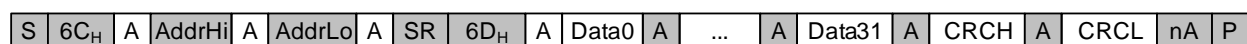
1. If transferring the start address, the lower 5 bits are cleared automatically.
2. If less than 36 data bytes are received, nothing is written into the FLASH. The Read Status command can be used to check an invalid command length error.
3. If an already written section in the FLASH gets re-written (without being erased before), the resulting data is undefined.
4. After the stop condition (P) is received the data is programmed into the FLASH. During the programming time incoming I<sup>2</sup>C commands are not acknowledged.



**Figure 6 FLASH Write Line command**

### 3.2 FLASH Read Line

The contents of the FLASH memory (4000<sub>H</sub> to 587F<sub>H</sub>) can be read out via the I<sup>2</sup>C interface. [Figure 7](#) shows the structure of the FLASH read line command.



**Figure 7 FLASH Read Line command**

### 3.3 FLASH Erase

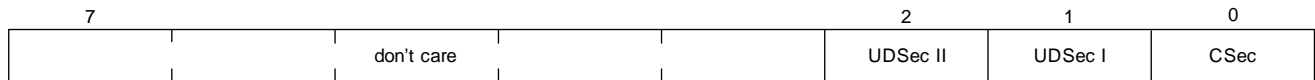
The FLASH Erase command is show in [Figure 8](#) and can be used to erase the Code Sector and the User Data Sectors. The FLASH Erase time is specified in [\[1\]](#). [Figure 9](#) and [Table 2](#) describe the bits of the Sector byte.

## PROGRAMMING Mode commands

*Note: After the stop condition (P) is received the selected FLASH sectors are being erased. During the erase time incoming I<sup>2</sup>C commands are not acknowledged.*



**Figure 8 FLASH Erase command**



**Figure 9 FLASH Erase: Sector byte**

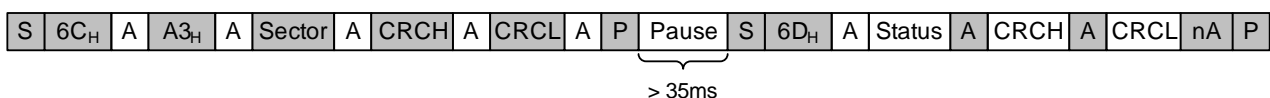
**Table 2 FLASH Erase: Sector byte**

Bits	Field	Description
2	UDSec II	0 <sub>B</sub> : don't check User Data Sector II 1 <sub>B</sub> : check whether User Data Sector II is erased
1	UDSec I	0 <sub>B</sub> : don't check User Data Sector I 1 <sub>B</sub> : check whether User Data Sector I is erased
0	CSec	0 <sub>B</sub> : don't check Code Sector 1 <sub>B</sub> : check whether Code Sector is erased

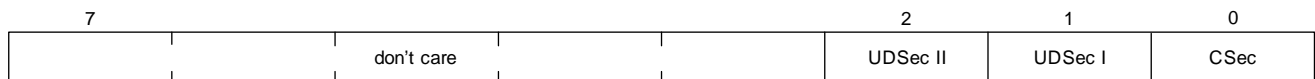
### 3.4 FLASH Check Erase Status

This function returns the status of the selected FLASH sector(s). The time required for checking the sectors depends on the selected sectors. The structure of the I<sup>2</sup>C command *Flash Check Erase Status* is shown in [Figure 10](#), [Figure 11](#) and [Table 3](#) describe the bits of the Sector byte. The Status byte is illustrated in [Figure 12](#) and [Table 4](#).

*Note: After the first stop condition (P) is received the selected FLASH sectors are checked. During this time incoming I<sup>2</sup>C commands are not acknowledged.*



**Figure 10 FLASH Check Erase Status command**

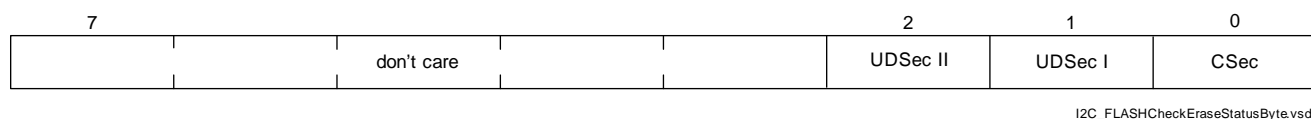


**Figure 11 FLASH Check Erase Status: Sector byte**

**Table 3 FLASH Check Erase Status: Sector byte**

Bits	Field	Description
2	UDSec II	0 <sub>B</sub> : don't check User Data Sector II 1 <sub>B</sub> : check whether User Data Sector II is erased
1	UDSec I	0 <sub>B</sub> : don't check User Data Sector I 1 <sub>B</sub> : check whether User Data Sector I is erased
0	CSec	0 <sub>B</sub> : don't check Code Sector 1 <sub>B</sub> : check whether Code Sector is erased

## PROGRAMMING Mode commands



**Figure 12** FLASH Check Erase Status: Status byte

**Table 4** FLASH Check Erase Status: Status byte

Bits	Field	Description
2	UDSec II	0 <sub>B</sub> : User Data Sector II is erased or untested 1 <sub>B</sub> : at least one bit is set in User Data Sector II
1	UDSec I	0 <sub>B</sub> : User Data Sector I is erased or untested 1 <sub>B</sub> : at least one bit is set in User Data Sector I
0	CSec	0 <sub>B</sub> : Code Sector is erased or untested 1 <sub>B</sub> : at least one bit is set in Code Sector

### 3.5 FLASH Set Code Lock (Lockbyte 2)

To set Lockbyte 2 D1<sub>H</sub> has to be written to FLASH address 577F<sub>H</sub> (top address of Code Sector). After the Lockbyte 2 is set, a startup in DEBUG or PROGRAMMING Mode is not possible any more.

*Note: To activate the Code Sector Lock the PMA71xx/PMA51xx has to be reset after Lockbyte D1<sub>H</sub> has been set.*

### 3.6 FLASH Set User Data Sector Lock (Lockbyte 3)

This command sets the Lockbyte for FLASH User Data Sectors I + II.

*Note: It is required to set Code Sector Lock (Lockbyte 2) to enable User Data Sector Lock to become effective.*

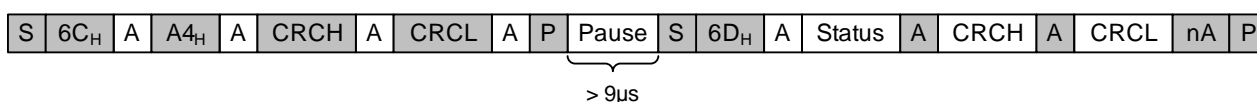
*Note: To activate the User Data Sector Lock (Lockbyte 3) the PMA71xx/PMA51xx has to be reset after setting the Lockbyte for User Data Sector and Code Sector.*



**Figure 13** FLASH Set Lockbyte 3 command

### 3.7 Read Status

This function is intended to read out the status of the previous executed functions (pass/fail). It can be called whenever desired to verify if there were errors since the last *Read status* call. [Figure 15](#) and [Table 5](#) describe the bits of the Status byte.



**Figure 14** Read Status command



**Figure 15** Read Status: Status byte

## Programming sequence for PMA71xx/PMA51xx code sector

**Table 5 Read Status: Status byte**

Bits	Field	Description
7:4	CmdCnt	Number of executed commands since the first detected error. 1111 <sub>B</sub> : 15 commands or more 1110 <sub>B</sub> : 14 commands ... 0001 <sub>B</sub> : 1 command 0000 <sub>B</sub> : error occurred in last command
3:2	ErrCnt	Erroneous events since the last Read status call. 11 <sub>B</sub> : three or more errors 10 <sub>B</sub> : two errors 01 <sub>B</sub> : one error 00 <sub>B</sub> : no error
1	InvCmdL	1 <sub>B</sub> : Invalid command length or execution fail since the last Read status call 0 <sub>B</sub> : Command length and execution correct since the last Read status call
0	CRCFail	1 <sub>B</sub> : CRC Failure detected since the last Read status call 0 <sub>B</sub> : no CRC Error occurred since the last Read status call

## 4 Programming sequence for PMA71xx/PMA51xx code sector

**Figure 16** shows a possible sequence to program the PMA71xx/PMA51xx code sector. First of all the PMA71xx/PMA51xx has to be set into PROGRAMMING Mode by setting PP0 to 0<sub>B</sub>, PP1 to 1<sub>B</sub> and MSE to 1<sub>B</sub> during reset. The MSE, PP0 and PP1 levels must not change after reset release during the whole  $t_{MODE}$  period (see [1]). Before writing data to the code sector the sector has to be erased by using the I<sup>2</sup>C command *FLASH Erase*. When the sector is erased, which can be checked with I<sup>2</sup>C command *FLASH Check Erase Status*, data can be written line by line (32 bits by 32 bits) into the code sector by calling the I<sup>2</sup>C command *FLASH Write Line*. Verification of the data can be done by reading out the currently written sector line by line with the I<sup>2</sup>C command *FLASH Read Line*. When the code sector is written a CRC checksum has to be calculated over the whole code sector. The low byte of the CRC checksum has to be stored to 577E<sub>H</sub> and the high byte to 577D<sub>H</sub>. To protect the code sector against undesired read outs this sector can be locked by writing D1<sub>H</sub> to FLASH address 577F<sub>H</sub> (Code Lock). After setting the Code Lock the PROGRAMMING and DEBUG Mode can not be entered again. The PMA71xx/PMA51xx can be set into the NORMAL Mode by setting MSE to 0<sub>B</sub> and resetting the chip.

# Programming sequence for PMA71xx/PMA51xx code sector

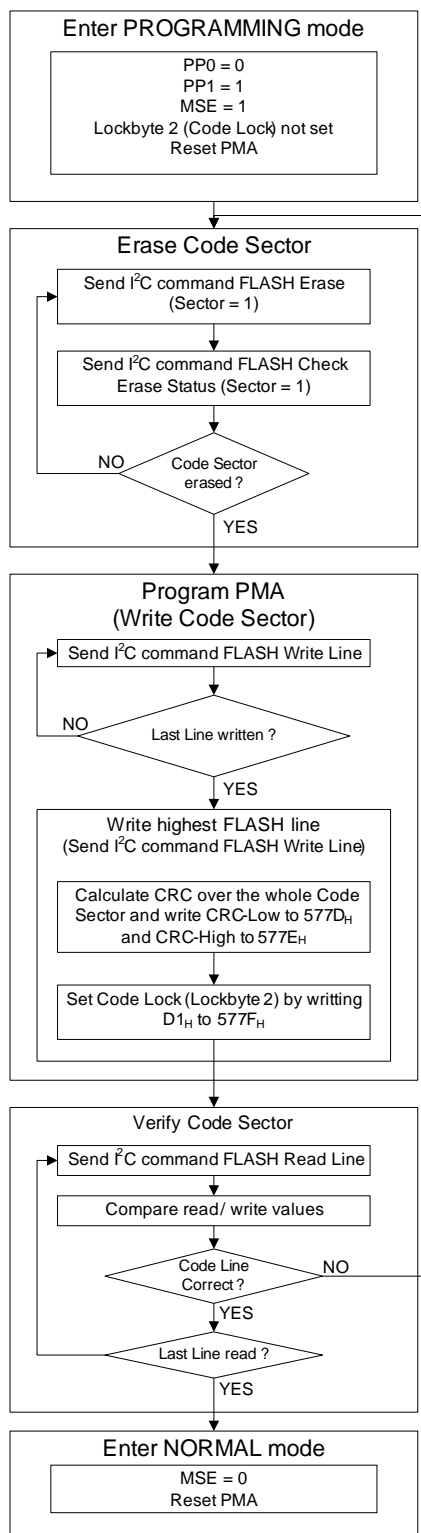


Figure 16 Possible programming sequence for PMA71xx/PMA51xx code sector



## 5 Estimation of device programming time

Due to the fact that the programming time is an important cost factor some estimations of the device programming time are done in this chapter. [Chapter 5.1](#) shows an estimation of the transaction times of every FLASH command. In [Chapter 5.2](#) and [Chapter 5.3](#) two programming scenarios are described.

### 5.1 FLASH command transaction times

The transaction time for each PROGRAMMING Mode command is calculated and summarized in [Table 7](#) to [Table 12](#). The timing values of  $t_{POR}$  (Power On Reset time),  $t_{MODE}$  (Mode selection time),  $t_{PROGRAM}$  (FLASH write time per line) and  $t_{ERASE}$  (FLASH erase time) are assumptions. For specified values refer to [\[1\]](#).

**Table 6 Assumptions for the calculation of FLASH command transaction times**

Parameter	assumed value
$t_{POR}$	10 ms
$t_{MODE}$	2.5 ms
$t_{PROGRAM}$	2.2 ms
$t_{ERASE}$	102 ms
I <sup>2</sup> C datarate	400 kbit/s
FLASH lines	188 (with 32 byte each)

**Table 7 FLASH Check Erase Status**

Nomenclature	Units	Bits / Unit	Bits total
S	2	1	2
Bytes	9	8	72
A	8	1	8
nA	1	1	1
P	2	1	2
Total bits per command			<b>85</b>
Total time per command [ms]			0.2125
Pause [ms]			35
<b>Total time per operation [ms]</b>			<b>35.2125</b>

**Table 8 FLASH Erase**

Nomenclature	Units	Bits / Unit	Bits total
S	1	1	1
Bytes	5	8	40
A	5	1	5
P	1	1	1
Total bits per command			<b>47</b>
Total time per command [ms]			0.1175
t <sub>ERASE</sub> [ms]			102
<b>Total time per operation [ms]</b>			<b>102.1175</b>

**Table 9 FLASH Write Line**

Nomenclature	Units	Bits / Unit	Bits total
S	1	1	1
Bytes	37	8	296
A	37	1	37
P	1	1	1
Total bits per command			<b>335</b>
Total time per command [ms]			0.8375
t <sub>PROGRAM</sub> [ms]			2.2
<b>Total time per operation [ms]</b>			<b>3.0375</b>

**Table 10 FLASH Read Line**

Nomenclature	Units	Bits / Unit	Bits total
S	1	1	1
SR	1	1	1
Bytes	38	8	304
A	37	1	37
nA	1	1	1
P	1	1	1
Total bits per command			<b>345</b>
Total time per command [ms]			0.8625
<b>Total time per operation [ms]</b>			<b>0.8625</b>

**Table 11 FLASH Read Status**

Nomenclature	Units	Bits / Unit	Bits total
S	2	1	2
Bytes	8	8	64
A	7	1	7
nA	1	1	1
P	2	1	2
Total bits per command			<b>76</b>
Total time per command [ms]			0.19
Pause [ms]			0.009
<b>Total time per operation [ms]</b>			<b>0.199</b>

**Table 12 FLASH Set User Data Sector Lock**

Nomenclature	Units	Bits / Unit	Bits total
S	1	1	1
Bytes	4	8	32
A	4	1	4
P	1	1	1
Total bits per command			<b>38</b>
Total time per command [ms]			0.095
<b>Total time per operation [ms]</b>			<b>0.095</b>

## 5.2 FLASH programming scenario 1

FLASH programming scenario 1 is optimized for throughput (speed), and assumes that I<sup>2</sup>C and bus errors are occurring only very rarely and that a simple *status check* of FLASH write operations is sufficient to ensure integrity. There are a maximum of 188 FLASH lines that must be programmed, and a *FLASH Read Status* command is performed after every 8 FLASH lines are written (i.e. for every 256 bytes of programmed FLASH). The total device programming time, including power on reset, mode selection, and all necessary I<sup>2</sup>C transactions, can be estimated as shown in [Table 13](#).

*Note: The FLASH Code sector is locked with the last FLASH Write Line command by writing the Lockbyte D1<sub>H</sub> to 577F<sub>H</sub>.*

**Table 13 Programming time estimation with status verification every 256 bytes**

	Units [ms]	Unit	Total time [ms]
t <sub>POR</sub>	10	1	10
t <sub>MODE</sub>	2.5	1	2.5
FLASH Erase Status	35.2125	1	35.2125
FLASH Write Line	3.0375	188	571.05
FLASH Check Status	0.199	24	4.776
FLASH Set Code Lock	0	0	0
<b>Total time for 6 kbyte [ms]</b>			<b>623.5385</b>

## 5.3 FLASH programming scenario 2

FLASH programming scenario 2 is less optimistic than the first, and assumes that I<sup>2</sup>C bus errors are more likely to occur. There are a maximum of 188 FLASH lines that must be programmed, and a *FLASH Read Line* command is performed after every FLASH line is written to allow write verification of each byte. The total device programming time, including power on reset, mode selection, and all necessary I<sup>2</sup>C transactions, can be estimated as shown in [Table 14](#).

*Note: The FLASH Code sector is locked with the last FLASH Write Line command by writing the Lockbyte D1<sub>H</sub> to 577F<sub>H</sub>.*

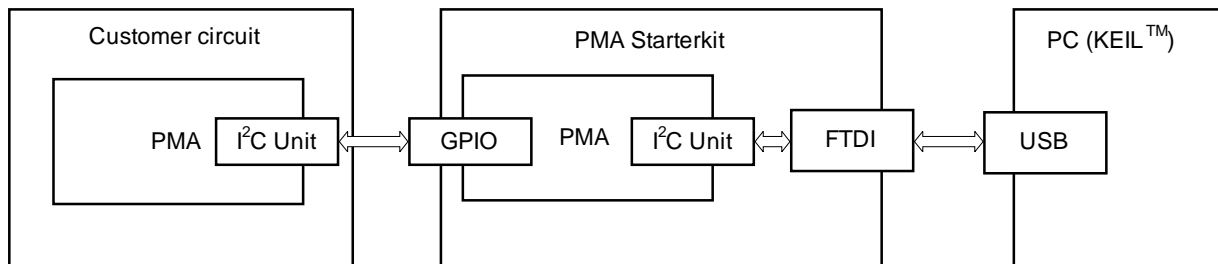
**Table 14 Programming time estimation with verification of each byte**

	Units [ms]	Unit	Total time [ms]
t <sub>POR</sub>	10	1	10
t <sub>MODE</sub>	2.5	1	2.5
FLASH Erase Status	35.2125	1	35.2125
FLASH Write Line	3.0375	188	571.05
FLASH Read Line	0.8625	188	162.15
FLASH Set Code Lock	0	0	0
<b>Total time for 6 kbyte [ms]</b>			<b>780.9125</b>

## 6 Application example (PMA Starterkit as programmer)

**Figure 17** shows how the PMA Starterkit can be used as programmer for a PMA on a customer circuit. For the communication between the PMA on the Starterkit and the KEIL™ driver on the PC an FTDI chip is used. The PMA on the Starterkit interacts with the FTDI via the I<sup>2</sup>C unit. The PMA on the customer circuit is programmed by the PMA on the Starterkit. Therefore the PMA on the Starterkit uses a software implementation of the I<sup>2</sup>C and the GPIOs PP2, PP3, PP4 and PP5. The PMA on the customer circuit uses the I<sup>2</sup>C unit.

The KEIL driver software for the PMA Starterkit and a User Guide which shows how to use the PMA Starterkit as a programmer can be downloaded from [http://www.infineon.com/PMA\\_Starterkit](http://www.infineon.com/PMA_Starterkit).



**Figure 17 Usage of PMA Starterkit as programmer**

## References

- [1] PMA71xx/PMA51xx Data Sheet

[www.infineon.com](http://www.infineon.com)