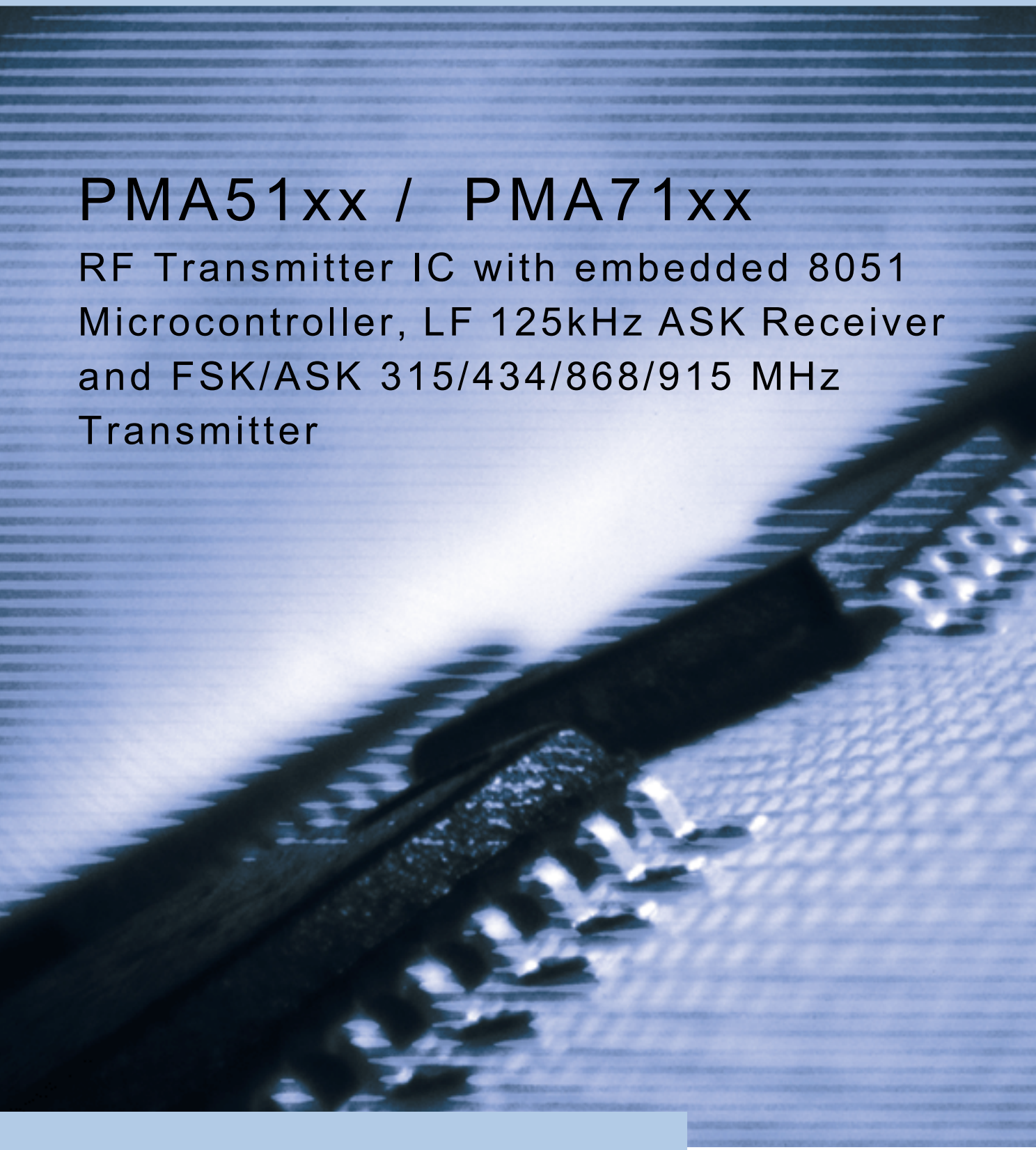


PMA51xx / PMA71xx

RF Transmitter IC with embedded 8051 Microcontroller, LF 125kHz ASK Receiver and FSK/ASK 315/434/868/915 MHz Transmitter

Sense & Control

Never stop thinking

Edition 2008-06-18

Published by Infineon Technologies AG,
Am Campeon 1-12
85579 Neubiberg, Germany

Attention please!

The information herein is given to describe certain components and shall not be considered as a guarantee of characteristics.

Terms of delivery and rights to technical change reserved.

We hereby disclaim any and all warranties, including but not limited to warranties of non-infringement, regarding circuits, descriptions and charts stated herein.

Information

For further information on technology, delivery terms and conditions and prices please contact your nearest Infineon Technologies Office in Germany or the Infineon Technologies Companies and our Infineon Technologies Representatives worldwide (**www.infineon.com**).

Warnings

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

Infineon Technologies Components may only be used in life-support devices or systems 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.

PMA51xx / PMA71xx

RF Transmitter IC with embedded 8051 Microcontroller, LF 125kHz ASK Receiver and FSK/ASK 315/434/868/915 MHz Transmitter

Sense & Control

Never stop thinking

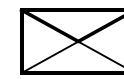
PMA5110

Revision History:2008-06-18 V1.1

Previous Version: ---

Page	Subject (major changes since previous version)

We Listen to Your Comments
Any information within this document that you feel is wrong, unclear or missing at all?
Your feedback will help us to continuously improve the quality of this document.
Please send your proposal (including a reference to this document) to:
sensors@infineon.com



1 Introduction

1.1 General Considerations

This document describes the ROM Library functions which are available in the PMA5110 and PMA7110. For other products in PMA51xx or PMA71xx family, please only take the relevante features and functions into account.

- PMA5110_ROMLibrary.h (the header file including the function prototypes)
- PMA5110_ROMLibrary.lib (the precompiled ROM Library functions)

Note: All charge consumptions and execution times stated throughout this document are target values measured with 10 samples at V_{BAT} = 3.0V and $T_{Ambient}$ = 25°C.

1.2 Type definitions

The following table defines the parameter types used throughout this document.

Table 1 Definition of types

Type	Description	Range	
		minimum	maximum
unsigned char	8 bit value without sign bit	0	255
signed char	8 bit value with sign bit	-128	127
unsigned int	16 bit value without sign bit	0	65535
signed int	16 bit value with sign bit	-32768	32767
unsigned long	32 bit value without sign bit	0	4294967296
signed long	32 bit value with sign bit	-2147483648	2147483647

1.3 Wakeup Handler

The Wakeup-Handler is executed every time the device wakes up from POWER DOWN state. Possible wakeup sources are listed in **“Reference Documents” on Page 80**.

Table 2 Wakeup Handler

Value	typ	max	Unit	Conditions
Wakeup from POWER DOWN state (independent of Wakeup source)	1,0	2,1	ms	

1.4 Restricted RAM and FLASH areas

The ROM Library functions use certain address areas in RAM and FLASH. They must not be changed by the application for proper operation.

1.4.1 Restricted RAM areas

The RAM area BB_H through FF_H (upper 68 Bytes) of the 256 Bytes RAM is used for the ROM Library functions so this RAM area is overwritten by the ROM Library functions.

1.4.2 Restricted FLASH areas

The FLASH area $57FA_H$ through $0x57FC_H$ is reserved for the used crystal frequency. The crystal frequency (divided by two) has to be written to that location during FLASH programming in the production in order to provide the ROM Library functions with the correct timebase for calibration purposes.

For 315 MHz applications a 19.6875 MHz crystal is used and the crystal frequency divided by half and be put into that location.

$$19687500\ \text{Hz} : 2 = 9843750\ \text{Hz} = 0\text{x}963426\ \text{Hz}$$

The values in FLASH have to be written in the following way:

- Flash address $57FA_H$: 96_H
- Flash address $57FB_H$: 34_H
- Flash address $57FC_H$: 26_H

For 433.92 MHz applications a 18.0800 MHz crystal is used and the crystal frequency divided by half and be put into that location.

$$18080000\ \text{Hz} : 2 = 9040000\ \text{Hz} = 0\text{x}89\text{F}080\ \text{Hz}$$

The values in FLASH have to be written in the following way:

- Flash address $57FA_H$: 89_H
- Flash address $57FB_H$: $F0_H$
- Flash address $57FC_H$: 80_H

For 868 MHz applications a 18.089583 MHz crystal is used and the crystal frequency divided by half and be put into that location.

$$18089583\ \text{Hz} : 2 = 9044792\ \text{Hz} = 0\text{x}8\text{A}0338\ \text{Hz}$$

The values in FLASH have to be written in the following way:

- Flash address $57FA_H$: $8A_H$
- Flash address $57FB_H$: 03_H
- Flash address $57FC_H$: 38_H

For 915 MHz applications a 19.0625 MHz crystal is used and the crystal frequency divided by half and be put into that location.

$$19062500\ \text{Hz} : 2 = 9531250\ \text{Hz} = 0\text{x}916\text{F}72\ \text{Hz}$$

The values in FLASH have to be written in the following way:

- Flash address $57FA_H$: 91_H
- Flash address $57FB_H$: $6F_H$
- Flash address $57FC_H$: 72_H

The FLASH address $57FF_H$ is reserved for Lockbyte 3. This value must not be changed by the application otherwise it might result in a unintentionally locked FLASH User Configuration Sector. Locking this sector is irreversibly and shall only be done by either programming the Lockbyte 3 together with writing and locking the FLASH Code Sector or by a dedicated ROM Library function **SetLockbyte3()**.

*Note: If **EraseUserConfigSector()** or **WriteUserConfigSectorLine()** are used by the application it has to be ensured that these restricted Flash addresses are not overwritten unintentionally.*

2 ROM Library Functions

The following library functions are available for application usage:

Table 3 ROM Library functions

ROM Library function	Description	Page
MeasureTemperature()	Returns ambient temperature	Page 18
MeasureSupplyVoltage()	Returns the battery voltage	Page 20
StartSupplyVoltage()	These three functions perform a Battery Voltage measurement during an RF Transmission.	Page 22
TriggerSupplyVoltage()		Page 24
GetSupplyVoltage()		Page 26
PowerDown()	Forces the device to POWER DOWN state	Page 28
ThermalShutdown()	Forces the device to THERMAL SHUTDOWN state	Page 29
Switch2XTAL()	Switches the systemclock to the crystal and prepares the device for RF Transmission	Page 31
Switch2RC()	Switches the systemclock to the 12 MHz RC HF Oscillator	Page 33
VCOTuning()	Tunes the VCO frequency (stops after lock)	Page 35
VCOTuningLong()	Tunes the VCO frequency (full range scan)	Page 37
CalibrateIntervalTimer()	Calibrates the Interval Timer precounter	Page 39
LFBaudrateCalibration()	Calibrates the LF baudrate divider	Page 41
ULongDivision()	Divides two unsigned values (32 bit : 32 bit)	Page 43
UIntDivision()	Divides two unsigned values (16 bit : 16 bit)	Page 45
CRC8Calculation()	Calculates an 8 Bit CRC (x^8+x^2+x^1+x^0)	Page 47
CRC8_410()	Calculates an 8 Bit CRC (x^8+x^4+x^1+x^0)	Page 49
ReadManufacturerRevNum()	Returns the device revision number	Page 51
ReadCustomerRevNum()	Returns the customers ROM revision number	Page 52
EraseUserConfigSector()	Erases the FLASH user configuration sector	Page 53
WriteUserConfigSectorLine()	Writes one FLASH line (32 Bytes) in the FLASH user configuration sector	Page 55
SetLockbyte3()	Sets the Lockbyte to protect the User Configuration sector	Page 57
InitRF()	Initializes an RF Transmission	Page 59
TransmitRF	Performs an RF Transmission	Page 61

ROM Library function	Description	Page
AES128Encrypt()	Performs a 128-Bit AES encryption	Page 63
AES128Decrypt()	Performs a 128-Bit AES decryption	Page 65
Wr_EEByte()	Writes a byte in the EEPROM Emulation	Page 67
Wr_EEInt()	Writes an int in the EEPROM Emulation	Page 69
Wr_EELong()	Writes a long in the EEPROM Emulation	Page 71
Wr_EEString()	Writes a string in the EEPROM Emulation	Page 73
Get_EEByte()	Reads a byte in the EEPROM Emulation	Page 75
Get_EEInt()	Reads an int in the EEPROM Emulation	Page 76
Get_EELong()	Reads a long in the EEPROM Emulation	Page 77
Get_EEString()	Reads a string in the EEPROM Emulation	Page 78

2.1 Meas_Temperature()

2.1.1 Description

This function performs a temperature measurement. The result is calibrated and compensated for sensitivity and offset.

2.1.2 Actions

- Measures the temperature sensor
- Compensate using calibration data stored in FLASH.

2.1.3 Prototype

unsigned char **Meas_Temperature** (signed int idata * **Temp_Result**)

2.1.4 Inputs

Table 4 Meas_Temperature: Input Parameters

Register / Address	Type	Name	Description
R7	signed int idata*	Temp_Result	iData pointer to an integer array containing the measurement result

2.1.5 Outputs

Table 5 Meas_Temperature: Output values

Register/ Address	Type	Name	Description
R7	unsigned char	StatusByte	0000.0000b: Success xxxx.xxx1b: Underflow of ADC Result xxxx.xx1xb: Overflow of ADC Result xxx1.xxxxb: VMIN warning
*Temp_Result+0	signed int	Compensated Temperature	0x8000 = -256.0 °C 0x0000 = 0.0 °C 0x7FFF = 255.9921875 °C (= 256 °C - 1 LSB where 1 LSB = 1/128 °C)
*Temp_Result+1	signed int	RAW Temperature	0x8000 = -256.0 °C 0x0000 = 0.0 °C 0x7FFF = 255.9921875 °C (= 256 °C - 1 LSB where 1 LSB = 1/128 °C)

2.1.6 Resource Usage

Table 6 Meas_Temperature: Resources

Type	used or modified
Registers	R0, R1, R2, R3, R4, R5, R6, R7
SFR	A, B, PSW, ADCOFF, ADCC0, ADCC1, ADCDL, ADCDH, ADCM, ADCS, CFG1, CFG2, DIVIC, DPL, DPH, IE
Stack	14 Bytes

2.1.7 Charge Consumption

Table 7 Meas_Temperature: Charge Consumption

Value	typ	max	Unit	Conditions
Charge Consumption		1.695	µC	systemclock= 12 MHz RC HF Oscillator, DIVIC = 0x00

2.1.8 Execution Time

Table 8 Meas_Temperature: Execution Time

Value	typ	max	Unit	Conditions
Execution Time		565	µs	systemclock= 12 MHz RC HF Oscillator, DIVIC = 0x00

2.2 MeasureSupplyVoltage()

2.2.1 Description

This function performs a battery voltage measurement. The result is calibrated and compensated for offset.

2.2.2 Actions

- Measures the supply voltage sensor
- Compensate using calibration data stored in FLASH

2.2.3 Prototype

unsigned char **MeasureSupplyVoltage** (signed int idata * **Batt_Result**)

2.2.4 Inputs

Table 9 MeasureSupplyVoltage: Input Parameters

Register / Address	Type	Name	Description
R7	unsigned int idata*	Batt_Result	iData pointer to an integer array containing the measurement result

2.2.5 Outputs

Table 10 MeasureSupplyVoltage: Output values

Register/ Address	Type	Name	Description
R7	unsigned char	StatusByte	0000.0000b: Success xxxx.xxx1b: Underflow of ADC Result xxxx.xx1xb: Overflow of ADC Result
*Batt_Result+0	signed int	Compensated battery voltage	0x8000 = -4096.0 mV (Only theoretical number) 0x0000 = 0.0 mV 0x7FFF = 4095.875 mV (= 4096 mV - 1 LSB where 1 LSB = 1/8 mV)
*Batt_Result+1	signed int	Raw battery voltage	10Bit ADC Result Value: 0000.00xx.xxxx.xxxxb

2.2.6 Resource Usage

Table 11 MeasureSupplyVoltage: Resources

Type	used or modified
Registers	R0, R1, R2, R3, R4, R5, R6, R7
SFR	A, B, PSW, ADCOFF, ADCC0, ADCC1, ADCDL, ADCDH, ADCM, ADCS, CFG1, CFG2, DPL, DPH, DIVIC, IE
Stack	8 Bytes

2.2.7 Charge Consumption

Table 12 MeasureSupplyVoltage: Charge Consumption

Value	typ	max	Unit	Conditions
Charge Consumption		1.0	µC	systemclock= 12 MHz RC HF Oscillator, DIVIC = 0x00

2.2.8 Execution Time

Table 13 MeasureSupplyVoltage: Execution Time

Value	typ	max	Unit	Conditions
Execution Time		334	µs	systemclock= 12 MHz RC HF Oscillator, DIVIC = 0x00

2.3 StartSupplyVoltage()

2.3.1 Description

This function prepares the ADC and the Supply Voltage Sensor for a battery measurement.

StartSupplyVoltage(), **TriggerSupplyVoltage()** and **GetSupplyVoltage()** are used to measure the battery voltage during an RF Transmission.

This function has to be called by the application before the RF Transmission takes place.

2.3.2 Actions

- Prepare the ADC and the Supply Voltage Sensor for a measurement

2.3.3 Prototype

unsigned char **StartSupplyVoltage** (void)

2.3.4 Inputs

Table 14 StartSupplyVoltage: Input Parameters

Register / Address	Type	Name	Description
none	---	---	---

2.3.5 Outputs

Table 15 StartSupplyVoltage: Output values

Register/ Address	Type	Name	Description
R7	unsigned char	Status Byte	always returns 0

2.3.6 Resource Usage

Table 16 StartSupplyVoltage: Resources

Type	used or modified
Registers	R0, R7
SFR	A, PSW, DPH, DPL, ADCM, ADCC0, ADCC1, ADCOFF, CFG1, CFG2, IE
Stack	5 Bytes

2.3.7 Execution Information

Table 17 StartSupplyVoltage: Execution Information

Value	typ	max	Unit	Conditions
Instruction Cycles		12		

2.4 TriggerSupplyVoltage()

2.4.1 Description

This function triggers the ADC to measure the Supply Voltage Sensor for a battery measurement.

StartSupplyVoltage(), **TriggerSupplyVoltage()** and **GetSupplyVoltage()** are used to measure the battery voltage during an RF Transmission.

This function has to be called by the application after the last byte of the datagram is shifted into SFR RFD.

2.4.2 Actions

- Trigger the ADC for a Supply Voltage measurement

2.4.3 Prototype

void **TriggerSupplyVoltage** (void)

2.4.4 Inputs

Table 18 TriggerSupplyVoltage: Input Parameters

Register / Address	Type	Name	Description
none	---	---	---

2.4.5 Outputs

Table 19 TriggerSupplyVoltage: Output values

Register/ Address	Type	Name	Description
none	---	---	---

2.4.6 Resource Usage

Table 20 TriggerSupplyVoltage: Resources

Type	used or modified
Registers	---
SFR	ADCM
Stack	2 Bytes

2.4.7 Execution Information

Table 21 TriggerSupplyVoltage: Execution Information

Value	typ	max	Unit	Conditions
Instruction Cycles		4		

2.5 GetSupplyVoltage()

2.5.1 Description

This function reads the Supply Voltage measurement result.

StartSupplyVoltage(), **TriggerSupplyVoltage()** and **GetSupplyVoltage()** are used to measure the battery voltage during an RF Transmission.

This function has to be called by the application after the RF transmission is finished.

2.5.2 Actions

- Read out the Supply Voltage measurement result

2.5.3 Prototype

unsigned char **GetSupplyVoltage** (signed int idata * **Batt_Result**)

2.5.4 Inputs

Table 22 GetSupplyVoltage: Input Parameters

Register / Address	Type	Name	Description
R7	unsigned int idata*	Batt_Result	iData pointer to an integer array containing the measurement result

2.5.5 Outputs

Table 23 GetSupplyVoltage: Output values

Register/ Address	Type	Name	Description
R7	unsigned char	StatusByte	0000.0000b: Success xxxx.xxx1b: Underflow of ADC Result xxxx.xx1xb: Overflow of ADC Result
*Batt_Result+0	signed int	Compensated battery voltage	0x8000 = -4096.0 mV (Only theoretical number) 0x0000 = 0.0 mV 0x7FFF = 4095.875 mV (= 4096 mV - 1 LSB where 1 LSB = 1/8 mV)
*Batt_Result+1	signed int	RAW battery voltage	10Bit ADC Result Value: 0000.00xx.xxxx.xxxxb

2.5.6 Resource Usage

Table 24 GetSupplyVoltage: Resources

Type	used or modified
Registers	R0, R1, R2, R3, R4, R5, R6, R7
SFR	A, B, PSW, DPH, DHL, ADCS, ADCDL, ADCDH, CFG1, CFG2, IE
Stack	5 Bytes

2.5.7 Execution Information

Table 25 GetSupplyVoltage: Execution Information

Value	typ	max	Unit	Conditions
Instruction Cycles		412		

2.6 PowerDown()

2.6.1 Description

This function forces the device to POWER DOWN state.

2.6.2 Actions

- Waits until all peripherals (RF Transmission and Interval Timer value storing) have completed their ongoing operations
- Enter POWER DOWN state

2.6.3 Prototype

void **PowerDown**(void)

2.6.4 Inputs

Table 26 PowerDown: Input Parameters

Register / Address	Type	Name	Description
none	---	---	---

2.6.5 Outputs

Table 27 PowerDown: Output values

Register/ Address	Type	Name	Description
none	---	---	---

2.6.6 Resource Usage

Table 28 PowerDown: Resources

Type	used or modified
Registers	none
SFR	A, PSW, FCS, CFG0, IE
Stack	5 Bytes

2.6.7 Execution Information

Table 29 PowerDown: Execution Information

Value	typ	max	Unit	Conditions
Instruction Cycles		19		ITinit = 0, RFSE = 1 before entering PDWN

2.7 ThermalShutdown()

2.7.1 Description

This function forces the device to THERMAL SHUTDOWN state.

The application should call this function whenever the ambient temperature is close to the maximum operating range (this can be detected using **Meas_Temperature()**) to protect the device while the ambient temperature is above the specified operating conditions.
If this function is called when the temperature is below the TMAX threshold the function will return without any action and the application program will continue uninterrupted, if the temperature is above the TMAX threshold THERMAL SHUTDOWN state is entered.

2.7.2 Actions

- Turn on the TMAX Detector
- Enter THERMAL SHUTDOWN state if TMAX Detector is set.

2.7.3 Prototype

void **ThermalShutdown**(void)

2.7.4 Inputs

Table 30 ThermalShutdown: Input Parameters

Register / Address	Type	Name	Description
none	---	---	---

2.7.5 Outputs

Table 31 ThermalShutdown: Output values

Register/ Address	Type	Name	Description
none	---	---	---

2.7.6 Resource Usage

Table 32 ThermalShutdown: Resources

Type	used or modified
Registers	R7
SFR	A, PSW, DPL, DPH, TMAX, CFG0, IE
Stack	3 Bytes

2.7.7 Execution Information

Table 33 ThermalShutdown: Execution Information

Value	typ	max	Unit	Conditions
Instruction Cycles		136		Temp. < TMAX: function will return without any action / Temp. > TMAX: function will enter THERMAL SHUTDOWN state

2.8 Switch2XTAL()

2.8.1 Description

This function switches the system clock from the 12 MHz RC HF Oscillator to the crystal oscillator, tunes the VCO and enables the PLL.

2.8.2 Actions

- Switch system clock to crystal oscillator (Delay time speficied in SFR XTCFG)
- Call ROM library function **VCOTuning()** to tune the VCO and enable the PLL for RF transmission.

2.8.3 Prototype

signed char **Switch2XTAL**(void)

2.8.4 Inputs

Table 34 Switch2XTAL: Input Parameters

Register / Address	Type	Name	Description
none	---	---	---

2.8.5 Outputs

Table 35 Switch2XTAL: Output values

Register/Address	Type	Name	Description
R7	signed char	StatusByte	StatusByte: 0: Success of VCO Tuning -1: VCO Tuning not successful, mean tuning curve selected -2: Wrong clock source selected (crystal required)

2.8.6 Resource Usage

Table 36 Switch2XTAL: Resources

Type	used or modified
Registers	R0, R1, R5, R6, R7
SFR	A, PSW, RFFSLD, RFC, RFVCO, DIVIC, CFG0, IE
Stack	9 Bytes

2.8.7 Charge Consumption

Table 37 Switch2XTAL: Charge Consumption

Value	typ	max	Unit	Conditions
Charge Consumption	7.2	13.4	µC	**typ:** using reduced set of tuning curves (re-tuning) **max:** perform all 16 tuning curves (first tuning after RESET) XTCFG=0x03, DIVIC = 0x00 systemclock= 12 MHz RC HF Oscillator

2.8.8 Execution Time

Table 38 Switch2XTAL: Execution Time

Value	typ	max	Unit	Conditions
Execution Time	2393	4473	µs	**typ:** using reduced set of tuning curves (re-tuning) **max:** perform all 16 tuning curves (first tuning after RESET) XTCFG=0x03, DIVIC = 0x00 systemclock= 12 MHz RC HF Oscillator

2.9 Switch2RC()

2.9.1 Description

This function switches the system clock from the crystal oscillator back to 12 MHz RC HF Oscillator.

Note: If RF transmission is performed before this function is called, it is recommended to disable the PLL by clearing SFR bit RFC.1[ENFSYN] to reduce the current consumption.

2.9.2 Actions

- Switch system clock to 12 MHz RC HF Oscillator

2.9.3 Prototype

void **Switch2RC**(void)

2.9.4 Inputs

Table 39 Switch2RC: Input Parameters

Register / Address	Type	Name	Description
none	---	---	---

2.9.5 Outputs

Table 40 Switch2RC: Output values

Register/ Address	Type	Name	Description
none	---	---	---

2.9.6 Resource Usage

Table 41 Switch2RC: Resources

Type	used or modified
Registers	none
SFR	CFG0, TCON, LFDIV1, DSR, IE
Stack	3 Bytes

2.9.7 Execution Information

Table 42 Switch2RC: Execution Information

Value	typ	max	Unit	Conditions
Instruction Cycles		15		

2.10 VCOTuning()

2.10.1 Description

This function selects an appropriate tuning curve for the VCO and enables the PLL. It does not scan the complete tuning curve range but returns as soon as the PLL locks.

Note: If RF transmission is not performed immediately after this function is called, it is recommended to disable the PLL by clearing SFR bit RFC.1[ENFSYN] to reduce the current consumption.

2.10.2 Actions

- Select appropriate tuning curve
- enable PLL for a rapid RF transmission

2.10.3 Prototype

signed char **VCOTuning**(void)

2.10.4 Inputs

Table 43 VCOTuning: Input Parameters

Register / Address	Type	Name	Description
none	---	---	---

2.10.5 Outputs

Table 44 VCOTuning: Output values

Register/ Address	Type	Name	Description
R7	signed char	StatusByte	StatusByte: 0: Success of VCO Tuning -1: VCO Tuning not successful, mean tuning curve selected -2: Wrong clock source selected (crystal required)

2.10.6 Resource Usage

Table 45 VCOTuning: Resources

Type	used or modified
Registers	R0, R1, R5, R6, R7
SFR	A, PSW, RFFSLD, RFC, RFVCO, DIVIC, IE
Stack	7 Bytes

2.10.7 Charge Consumption

Table 46 VCOTuning: Charge Consumption

Value	typ	max	Unit	Conditions
Charge Consumption	3.1	7.9	µC	**typ:** using reduced set of tuning curves (re-tuning) **max:** perform all 16 tuning curves (first tuning after RESET) DIVIC = 0x00 systemclock = XTAL

2.10.8 Execution Time

Table 47 VCOTuning: Execution Time

Value	typ	max	Unit	Conditions
Execution Time	1032	2624	µs	**typ:** using reduced set of tuning curves (re-tuning) **max:** perform all 16 tuning curves (first tuning after RESET) DIVIC = 0x00 systemclock = XTAL

2.11 VCOTuningLong()

2.11.1 Description

This function selects an appropriate tuning curve for the VCO and enables the PLL. In contrast to the VCOTuning function, this routine scans the complete curve range.

Note: If RF transmission is not performed immediately after this function is called, it is recommended to disable the PLL by clearing SFR bit RFC.1[ENFSYN] to reduce the current consumption.

2.11.2 Actions

- Select appropriate tuning curve
- Enable PLL for a rapid RF transmission

2.11.3 Prototype

signed char **VCOTuningLong**(void)

2.11.4 Inputs

Table 48 VCOTuningLong: Input Parameters

Register / Address	Type	Name	Description
none	---	---	---

2.11.5 Outputs

Table 49 VCOTuningLong: Output values

Register/ Address	Type	Name	Description
R7	signed char	StatusByte	StatusByte: 0: Success of VCO Tuning -1: VCO Tuning not successful, mean tuning curve selected -2: Wrong clock source selected (crystal required)

2.11.6 Resource Usage

Table 50 VCOTuningLong: Resources

Type	used or modified
Registers	R0, R1, R5, R6, R7
SFR	A, PSW, RFFSLD, RFC, RFVCO, DIVIC, IE
Stack	7 Bytes

2.11.7 Charge Consumption

Table 51 VCOTuningLong: Charge Consumption

Value	typ	max	Unit	Conditions
Charge Consumption	3.1	7.9	µC	**typ:** using reduced set of tuning curves (re-tuning) **max:** perform all 16 tuning curves (first tuning after RESET) DIVIC = 0x00 systemclock = XTAL

2.11.8 Execution Time

Table 52 VCOTuningLong: Execution Time

Value	typ	max	Unit	Conditions
Execution Time	1032	2624	µs	**typ:** using reduced set of tuning curves (re-tuning) **max:** perform all 16 tuning curves (first tuning after RESET) DIVIC = 0x00 systemclock = XTAL

2.12 CalibrateIntervalTimer()

2.12.1 Description

This function initiates a calibration of the interval timer precounter (ITPL and ITPH). The function can work with both clock sources (12MHz RC Clock and Crystal clock), utilizing a special timer mode.

Note: Due to accuracy reasons it is recommended to call this function with the crystal oscillator selected as system clock.

In case the crystal oscillator is used the crystal frequency divided by 2 has to be stored in the FLASH user configuration sector at address $57FA_H$ (MSByte) to $57FC_H$ (LSByte). If the value found at this FLASH location is not within the range of 9 MHz to 10 MHz a default clock frequency of 9.5 MHz is assumed for the tuning.
This function automatically calibrates in addition the interval timer the LF On/Off Timer precounter (SFR LFOOTP) to 50 ms.

2.12.2 Actions

- Calibrate the interval timer precounter (SFR ITPL, SFR ITPH)
- Calibrate the LF On/Off timer precounter (SFR LFOOTP)

2.12.3 Prototype

signed char **CalibrateIntervalTimer**(unsigned char WU_Frequency)

2.12.4 Inputs

Table 53 CalibrateIntervalTimer: Input Parameters

Register / Address	Type	Name	Description
R7	unsigned char	WU_Frequency	base frequency of the interval timer precounter [Hz] 1dec: 1 Hz (precounter time ~1000 ms) 2dec: 2 Hz (precounter time ~500 ms) ... 19dec: 19 Hz (precounter time ~53 ms) 20dec: 20 Hz (precounter time ~50 ms)

2.12.5 Outputs

Table 54 CalibrateIntervalTimer: Output values

Register/ Address	Type	Name	Description
R7	signed char	StatusByte	StatusByte: 0: Success -1: No valid crystal frequency found in FLASH -2: Input parameter out of range

2.12.6 Resource Usage

Table 55 CalibrateIntervalTimer: Resources

Type	used or modified
Registers	R0, R1, R2, R3, R4, R5, R6, R7
SFR	A, B, PSW, DPL, DPH, TCON, TMOD, TH0, TL0, CFG0, ITPL, ITPH, LFOOTP, DIVIC, IE
Stack	7 Bytes

2.12.7 Charge Consumption

Table 56 CalibrateIntervalTimer: Charge Consumption

Value	typ	max	Unit	Conditions
Charge Consumption		3.6	µC	DIVIC = 0x00 systemclock = XTAL

2.12.8 Execution Time

Table 57 CalibrateIntervalTimer: Execution Time

Value	typ	max	Unit	Conditions
Execution Time		1820	µs	DIVIC = 0x00 systemclock = XTAL

2.13 LFBaudrateCalibration()

2.13.1 Description

Calling this function calibrates the LF baudrate divider towards the crystal oscillator, thus eliminating the impact of the drift and the offset of the 12 MHz RC HF Oscillator on the LF baudrate accuracy.

$$LFDIV1/0 = \frac{f_{12\ MHz\ RC\ HF\ Oscillator}}{16 \bullet baudrate}$$

The crystal frequency divided by 2 has to be stored in the FLASH user configuration sector at address $57FA_H$ (MSByte) to $57FC_H$ (LSByte). If the value found at this FLASH location is not within the range of 9 MHz to 10 MHz this function returns an error in the StatusByte and sets SFR LFDIV1 and SFR LFDIV0 to the following nominal value:

$$LFDIV1/0 = \frac{12000000}{16 \bullet baudrate}$$

2.13.2 Actions

- Set SFR LFDIV1 and SFR LFDIV0 according to the current frequency of the 12 MHz RC HF Oscillator
- Switch from 12 MHz RC HF Oscillator to XTAL

2.13.3 Prototype

signed char **LFBaudrateCalibration**(unsigned int baudrate)

2.13.4 Inputs

Table 58 LFBaudrateCalibration: Input Parameters

Register / Address	Type	Name	Description
R6, R7	unsigned int	baudrate	baudrate 2000dec: 2000 baud 3900dec: 3900 baud 4000dec: 4000 baud

2.13.5 Outputs

Table 59 LFBaudrateCalibration: Output values

Register/ Address	Type	Name	Description
R7	signed char	StatusByte	StatusByte: 0: Success -1: No valid crystal frequency found in FLASH -2: Input parameter out of range

2.13.6 Resource Usage

Table 60 LFBaudrateCalibration: Resources

Type	used or modified
Registers	R0,R1,R2,R3,R4,R5,R6,R7
SFR	A, B, DPH, DPL, PSW,CFG0,LFDIV0,LFDIV1, TH0,TL0,TMOD, TCON, IE
Stack	4 Bytes

2.13.7 Charge Consumption

Table 61 LFBaudrateCalibration: Charge Consumption

Value	typ	max	Unit	Conditions
Charge Consumption		1.9	µC	Baudrate = 4000 DIVIC = 0x00 systemclock = XTAL

2.13.8 Execution Time

Table 62 LFBaudrateCalibration: Execution Time

Value	typ	max	Unit	Conditions
Execution Time		680	µs	Baudrate = 4000 DIVIC = 0x00 systemclock = XTAL

2.14 ULongDivision() (32Bit : 32Bit)

2.14.1 Description

This function divides the unsigned long value (32bit) *Dividend* by the unsigned long value (32bit) *Divisor*.

2.14.2 Actions

- Perform division

2.14.3 Prototype

unsigned long **ULongDivision**(unsigned long idata * *Dividend*,
unsigned long idata * *Divisor*)

2.14.4 Inputs

Table 63 ULongDivision: Input Parameters

Register / Address	Type	Name	Description
R7	unsigned long idata*	Dividend	iData Pointer to 32 bit Dividend
R5	unsigned long idata*	Divisor	iData Pointer to 32 bit Divisor

2.14.5 Outputs

Table 64 ULongDivision: Output values

Register/ Address	Type	Name	Description
R4(MSB), R5, R6, R7(LSB)	unsigned long	Quotient	32 bit Quotient of the division (Dividend / Divisor)

2.14.6 Resource Usage

Table 65 ULongDivision: Resources

Type	used or modified
Registers	R0, R1, R2, R3, R4, R5, R6, R7
SFR	A, B, PSW, DPL, DPH, IE
Stack	5 Bytes

2.14.7 Execution Information

Table 66 ULongDivision: Execution Information

Value	typ	max	Unit	Conditions
Instruction Cycles		760		

2.15 UIntDivision() (16Bit : 16Bit)

2.15.1 Description

This function divides the unsigned int value (16bit) *Dividend* by the unsigned int value (16bit) *Divisor*.

2.15.2 Actions

- Perform division

2.15.3 Prototype

unsigned int **UIntDivision**(unsigned int *Dividend*,
unsigned int *Divisor*)

2.15.4 Inputs

Table 67 UIntDivision: Input Parameters

Register / Address	Type	Name	Description
R6(MSB) R7(LSB)	unsigned int	Dividend	16 bit Dividend
R4(MSB) R5(LSB)	unsigned int	Divisor	16 bit Divisor

2.15.5 Outputs

Table 68 UIntDivision: Output values

Register/ Address	Type	Name	Description
R6(MSB), R7(LSB)	unsigned int	Quotient	16 bit Quotient of the division (Dividend / Divisor)

2.15.6 Resource Usage

Table 69 UIntDivision: Resources

Type	used or modified
Registers	R0, R4, R5, R6, R7
SFR	A, B, PSW, IE
Stack	5 Bytes

2.15.7 Execution Information

Table 70 UIntDivision: Execution Information

Value	typ	max	Unit	Conditions
Instruction Cycles		150		

2.16 CRC8Calculation()

2.16.1 Description

This function calculates the CRC-8 checksum for a memory area in RAM using a fixed polynomial (x^8+x^2+x+1).

2.16.2 Actions

- Calculate CRC-8

2.16.3 Prototype

unsigned char **CRC8Calculation**(unsigned char Preload,
unsigned char idata * BlockStart,
unsigned char BlockLength)

2.16.4 Inputs

Table 71 CRC8Calculation: Input Parameters

Register / Address	Type	Name	Description
R7	unsigned char	Preload	Preload Value for the CRC Calculation. Usually set to 0xFF
R5	unsigned char idata*	BlockStart	iData Pointer to first Byte of the Data that is to be used for calculating checksum
R3	unsigned char	BlockLength	Length of Block that is used for calculation of the checksum, starting with *BlockStart.

2.16.5 Outputs

Table 72 CRC8Calculation: Output values

Register/ Address	Type	Name	Description
R7	unsigned char	CRC_Result	calculated CRC8 checksum

2.16.6 Resource Usage

Table 73 CRC8Calculation: Resources

Type	used or modified
Registers	R0, R1, R2, R3, R4, R5, R7
SFR	A, PSW, IE
Stack	5 Bytes

2.16.7 Execution Information

Table 74 CRC8Calculation: Execution Information

Value	typ	max	Unit	Conditions
Instruction Cycles		3+x*17		x = Number of Bytes

2.17 CRC8_410()

2.17.1 Description

This function calculates the CRC-8 checksum for a memory area in RAM using a fixed polynomial (x^8+x^4+x+1).

2.17.2 Actions

- Calculate CRC-8

2.17.3 Prototype

unsigned char **CRC8_410**(unsigned char Preload,
unsigned char idata * BlockStart,
unsigned char BlockLength)

2.17.4 Inputs

Table 75 CRC8_410: Input Parameters

Register / Address	Type	Name	Description
R7	unsigned char	Preload	Preload Value for the CRC Calculation. Usually set to 0xFF
R5	unsigned char idata*	BlockStart	iData Pointer to first Byte of the Data that is to be used for calculating checksum
R3	unsigned char	BlockLength	Length of Block that is used for calculation of the checksum, starting with *BlockStart.

2.17.5 Outputs

Table 76 CRC8_410: Output values

Register/ Address	Type	Name	Description
R7	unsigned char	CRC_Result	calculated CRC8 checksum

2.17.6 Resource Usage

Table 77 CRC8_410: Resources

Type	used or modified
Registers	R0, R1, R2, R3, R4, R5, R7
SFR	A, PSW, IE
Stack	5 Bytes

2.17.7 Execution Information

Table 78 CRC8_410: Execution Information

Value	typ	max	Unit	Conditions
Instruction Cycles		10+x*20		x = Number of Bytes

2.18 ReadManufacturerRevNum()

2.18.1 Description

This function returns the PMA5110 revision number

2.18.2 Actions

- Read 2-byte Revision Number

2.18.3 Prototype

signed int **ReadManufacturerRevNum** (void)

2.18.4 Inputs

Table 79 ReadManufacturerRevNum: Input Parameters

Register / Address	Type	Name	Description
none	---	---	---

2.18.5 Outputs

Table 80 ReadManufacturerRevNum: Output values

Register/ Address	Type	Name	Description
R6(MSB), R7(LSB)	signed int	Firmware Revision	0x00A1: Design Step A1

2.18.6 Resource Usage

Table 81 ReadManufacturerRevNum: Resources

Type	used or modified
Registers	R6, R7
SFR	IE
Stack	5 Bytes

2.18.7 Execution Information

Table 82 ReadManufacturerRevNum: Execution Information

Value	typ	max	Unit	Conditions
Instruction Cycles		12		

2.19 ReadCustomerRevNum()

2.19.1 Description

This function returns the Customer ROM Library revision number.

2.19.2 Actions

- Read 2-byte Revision Number

2.19.3 Prototype

signed int **ReadCustomerRevNum** (void)

2.19.4 Inputs

Table 83 ReadCustomerRevNum: Input Parameters

Register / Address	Type	Name	Description
none	---	---	---

2.19.5 Outputs

Table 84 ReadCustomerRevNum: Output values

Register/ Address	Type	Name	Description
R6(MSB), R7(LSB)	signed int	ROM Revision	0xCxxx: C = Customer Code (defined by IFX) xxx = Customer defined Rev. Number

2.19.6 Resource Usage

Table 85 ReadCustomerRevNum: Resources

Type	used or modified
Registers	R6, R7
SFR	A,PSW, DPL, DPH, IE
Stack	5 Bytes

2.19.7 Execution Information

Table 86 ReadCustomerRevNum: Execution Information

Value	typ	max	Unit	Conditions
Instruction Cycles		7		

2.20 EraseUserConfigSector()

2.20.1 Description

This function erases the FLASH user configuration sector located at FLASH address 5780_H -- $57FF_H$ if the Lockbyte 3 is not set. If Lockbyte 3 is set this function will return -1 without any action.

Note: The application software has to ensure that FLASH is only programmed or erased when all required environmental conditions are fulfilled. Special care has to be taken that ambient temperature T_{FL}, supply voltage V_{batFL} and Endurance En_{FL} within specified range (see "Reference Documents" on Page 80.)
This function returns -1 and has no effect if executed in DEBUG mode.

2.20.2 Actions

- Erase the FLASH user configuration sector

2.20.3 Prototype

signed char **EraseUserConfigSector** (void)

2.20.4 Inputs

Table 87 EraseUserConfigSector: Input Parameters

Register / Address	Type	Name	Description
none	---	---	---

2.20.5 Outputs

Table 88 EraseUserConfigSector: Output values

Register/ Address	Type	Name	Description
R7	signed char	Statusbyte	0: success -1: failed

2.20.6 Resource Usage

Table 89 EraseUserConfigSector: Resources

Type	used or modified
Registers	R0, R1, R2, R3, R4, R5, R6, R7
SFR	A, B, PSW, DPL, DPH, CRC0, CRC1, CRCC, CRCD, DIVIC, TCON, TH1, TL1, FCS, FCSERM, FCPP0, FCTKAS, FCSP, IE
Stack	9 Bytes

2.20.7 Execution Information

Table 90 EraseUserConfigSector: Execution Information

Value	typ	max	Unit	Conditions
Instruction Cycles		204000		

2.21 WriteUserConfigSectorLine()

2.21.1 Description

This function writes one line in one of the two FLASH user configuration sectors located at FLASH address 5780_H -- $587F_H$ if the Lockbyte 3 is not set. If Lockbyte 3 is set this function will return -1 without any action.

The written data is verified after the programming. In case the verification fails this function will return -1.

*Note: The application software has to ensure that FLASH is only programmed or erased when all required environmental conditions are fulfilled. Special care has to be taken that ambient temperature T_{FL}, supply voltage V_{batFL} and Endurance En_{FL} within specified range (see **"Reference Documents" on Page 80**.)*
This function returns -1 and has no effect if executed in DEBUG mode.

2.21.2 Actions

- Write one 32 Byte FLASH Line of one of the two FLASH user configuration sectors

2.21.3 Prototype

signed char **WriteFlashUserConfigurationSectorLine** (unsigned int Startaddress,
unsigned char idata * WrData)

2.21.4 Inputs

Table 91 WriteFlashUserConfigurationSectorLine: Input Parameters

Register / Address	Type	Name	Description
R7, R6	unsigned int	Stadtaddress	Startaddress 0x5780: FLASH Line 0 0x57A0: FLASH Line 1 0x57C0: FLASH Line 2 0x57E0: FLASH Line 3 0x5800: FLASH Line 4 0x5820: FLASH Line 5 0x5840: FLASH Line 6 0x5860: FLASH Line 7
R5	unsigned char idata*	WrData	iData Pointer to first Byte of the 32 Byte Data array that is going to be written to the FLASH Line.

2.21.5 Outputs

Table 92 WriteFlashUserConfigurationSectorLine: Output values

Register/ Address	Type	Name	Description
R7	signed char	Statusbyte	0: success -1: failed

2.21.6 Resource Usage

Table 93 WriteFlashUserConfigurationSectorLine: Resources

Type	used or modified
Registers	R0, R1, R2, R3, R4, R5, R6, R7
SFR	A,CFG2,DIVIC,CRCD, CRCC,CRC0,CRC1,DPTR,FCSERM,FCT-KAS,FCPP0,FCPP1,PSW,TCON,TMOD,TH1,TL1, IE
Stack	9 Bytes

2.21.7 Execution Information

Table 94 WriteFlashUserConfigurationSectorLine: Execution Information

Value	typ	max	Unit	Conditions
Instruction Cycles		4400		

2.22 SetLockbyte3()

2.22.1 Description

This function sets the Lockbyte 3 which protects the User Configuration sectors.

Attention: This function shows only effect if the Lockbyte 2 that protects the Code Sector is set.

*Note: The application software has to ensure that FLASH is only programmed or erased when all required environmental conditions are fulfilled. Special care has to be taken that ambient temperature T_{FL}, supply voltage V_{batFL} and Endurance En_{FL} within specified range (see **"Reference Documents" on Page 80**.)*
This function returns -1 and has no effect if executed in DEBUG mode.

2.22.2 Actions

- Set the Lockbyte 3 protecting the User Configuration sectors

2.22.3 Prototype

signed char **SetLockbyte3**(void)

2.22.4 Inputs

Table 95 SetLockbyte3: Input Parameters

Register / Address	Type	Name	Description
none	---	---	---

2.22.5 Outputs

Table 96 SetLockbyte3: Output values

Register/ Address	Type	Name	Description
R7	signed char	Statusbyte	0: success -1: failed

2.22.6 Resource Usage

Table 97 SetLockbyte3: Resources

Type	used or modified
Registers	R0, R1, R2, R3, R4, R5, R6, R7
SFR	A,B,CFG2,CRCC,CRCD,CRC0,CRC1,DPTR,DIVIC,FCS,FCSERM,FCT-KAS,FCPP0,FCPP1,TCON,TH1,TL1, IE
Stack	14 Bytes

2.22.7 Execution Information

Table 98 SetLockbyte3: Execution Information

Value	typ	max	Unit	Conditions
Instruction Cycles		235500		

2.23 InitRF()

2.23.1 Description

This function prepares an RF-Transmission by configuring the corresponding Special Function Registers (SFR).

2.23.2 Actions

- Set all correspondig SRFs for an RF-transmission according to user input structrue.

2.23.3 Prototype

void **InitRF**(struct RF_Config idata * idata RFConfig)

2.23.4 Inputs

Table 99 InitRF: Input Parameters

Register / Address	Type	Name	Description

R7	idata *	RF_Config	Pointer to a struct with the following definition:
			struct RF_Config
			{
			ENCODING Encoding;
			unsigned char dataLength;
			QUIESCENT Quiescent;
			INTMASK IntMask;
			DUTYCONTROL DutyControl;
			DUTYCYCLE DutyCycle;
			MODULATION Modulation;
			INVERSION Invert;
			FREQUENCY Frequency;
			OUTSTAGES OutStages;
			XCAPSHORT XcapShort;
			unsigned char xtal0;
			unsigned char xtal1;
			unsigned int baudrate;
			}
			
			Remark: Enum variables are defined in file PMA5110_ROMLibrary.h.

2.23.5 Outputs

Table 100 InitRF: Output values

Register/ Address	Type	Name	Description
none	---	---	---

2.23.6 Resource Usage

Table 101 InitRF: Resources

Type	used or modified
Registers	R0, R1, R2, R3, R4, R5, R6, R7
SFR	A, PSW, RFENC, RFTX, RFFSPLL, XTAL0, XTAL1, TMOD2, TCON2, TH3, TL3, TH2, TL2, IE
Stack	5 Bytes

2.23.7 Execution Information

Table 102 InitRF: Execution Information

Value	typ	max	Unit	Conditions
Instruction Cycles		1020		

2.24 TransmitRF()

2.24.1 Description

This function transmits a given data packet via RF according to the preset of the corresponding SFRs.

2.24.2 Actions

- Transmit a given data packet via RF.

2.24.3 Prototype

signed char **TransmitRF**(unsigned char idata *DataArray, unsigned char BlockLength)

2.24.4 Inputs

Table 103 TransmitRF: Input Parameters

Register / Address	Type	Name	Description
R7	unsigend char idata *	DataArray	Pointer to data block that will be transmitted
R5	unsigend char	BlockLength	Length of data block in bytes

2.24.5 Outputs

Table 104 TransmitRF: Output values

Register/ Address	Type	Name	Description
R7	signed char	Statusbyte	0: success -1: BlockLength = 0

2.24.6 Resource Usage

Table 105 TransmitRF: Resources

Type	used or modified
Registers	R0, R1, R2, R3, R4, R5, R6, R7
SFR	A, PSW, TCON2, RFS, RFD, IE
Stack	3 Bytes

2.24.7 Execution Information

Table 106 TransmitRF: Execution Information

Value	typ	max	Unit	Conditions
Instruction Cycles		13+X(10 +Y)		X.. BlockLength Y...Time for sending a byte (baudrate dep.)

2.25 AES128Encrypt()

2.25.1 Description

This function performs a AES Encryption with 128-Bit data and a 128-Bit key.

2.25.2 Actions

- Prepare data for AES encryption
- Encrypt 128-Bit data with a 128-Bit key using AES.
- Copy back results

2.25.3 Prototype

void **AES128Encrypt**(const unsigned char idata* InputData, unsigned char idata* OutputData, const unsigned char idata* Key)

2.25.4 Inputs

Table 107 AES128Encrypt: Input Parameters

Register / Address	Type	Name	Description
R7	const unsigned char idata*	InputData	Pointer to 128-Bit data for AES encryption
R5	unsigned char idata*	OutputData	Pointer to 128-Bit AES encryption result
R3	const unsigned char idata*	Key	Pointer to 128-Bit AES key

2.25.5 Outputs

Table 108 AES128Encrypt: Output values

Register/ Address	Type	Name	Description
none	---	---	---

2.25.6 Resource Usage

Table 109 AES128Encrypt: Resources

Type	used or modified
Registers	R0, R1, R2, R3, R4, R5, R6, R7
SFR	A,B,DPTR,IE
Stack	13 Bytes

2.25.7 Execution Information

Table 110 AES128Encrypt: Execution Information

Value	typ	max	Unit	Conditions
Execution time		2072	µs	Clock = High RC DIVIC = 0x00

2.26 AES128Decrypt()

2.26.1 Description

This function performs a AES Decryption with 128-Bit data and a 128-Bit key.

2.26.2 Actions

- Prepare data for AES decryption
- Decrypt 128-Bit data with a 128-Bit key using AES.
- Copy back results

2.26.3 Prototype

signed char **AES128Decrypt**(void)

2.26.4 Inputs

Table 111 AES128Decrypt: Input Parameters

Register / Address	Type	Name	Description
R7	const unsigned char idata*	InputData	Pointer to 128-Bit data for AES decryption
R5	unsigned char idata*	OutputData	Pointer to 128-Bit AES decryption result
R3	const unsigned char idata*	Key	Pointer to 128-Bit AES key

2.26.5 Outputs

Table 112 AES128Decrypt: Output values

Register/ Address	Type	Name	Description
none	---	---	---

2.26.6 Resource Usage

Table 113 AES128Decrypt: Resources

Type	used or modified
Registers	R0, R1, R2, R3, R4, R5, R6, R7
SFR	A,B,DPTR,IE
Stack	13 Bytes

2.26.7 Execution Information

Table 114 AES128Decrypt: Execution Information

Value	typ	max	Unit	Conditions
Execution Time		3055	µs	Clock= High RC DIVIC = 0x00

2.27 Wr_EEByte()

2.27.1 Description

This function writes a single byte in the EEPROM Emulation.

Attention: This function shows only effect if the Lockbyte 3 that protects the User Configuration Sectors is NOT set.

*Note: The application software has to ensure that FLASH is only programmed or erased when all required environmental conditions are fulfilled. Special care has to be taken that ambient temperature T_{FL}, supply voltage V_{batFL} and Endurance En_{FL} within specified range (see **"Reference Documents" on Page 80**.)*
This function returns -1 and has no effect if executed in DEBUG mode.

2.27.2 Actions

- Get current EEPROM content
- Add byte to content
- Write back new EEPROM content

2.27.3 Prototype

signed char **WR_EEByte**(unsigned char Address, unsigned char EEByte)

2.27.4 Inputs

Table 115 Wr_EEByte: Input Parameters

Register / Address	Type	Name	Description
R7	unsigned char	Address	Address of byte location (0x00-0x1E)
R5	unsigend char	EEByte	Byte to write

2.27.5 Outputs

Table 116 Wr_EEByte: Output values

Register/ Address	Type	Name	Description
R7	signed char	Statusbyte	0: success -1: Address out of range -2: Lockbyte 3 set

2.27.6 Resource Usage

Table 117 Wr_EEByte: Resources

Type	used or modified
Registers	R0, R1, R2, R3, R4, R5, R6, R7
SFR	A, B, PSW, DPL, DPH, CRC0, CRC1, CRCC, CRCD, DIVIC, TCON, TH1, TL1, FCS, FCSERM, FCPP0, FCTKAS, FCSP, IE
Stack	19 Bytes

2.27.7 Execution Information

Table 118 Wr_EEByte: Execution Information

Value	typ	max	Unit	Conditions
Instruction Cycles	4760	212750		typ.: Writing of a line without erasing max.: Writing a line with erasing (startup and every fourth time)

2.28 Wr_EEInt()

2.28.1 Description

This function writes an int value in the EEPROM Emulation.

Attention: This function shows only effect if the Lockbyte 3 that protects the User Configuration Sectors is NOT set.

*Note: The application software has to ensure that FLASH is only programmed or erased when all required environmental conditions are fulfilled. Special care has to be taken that ambient temperature T_{FL}, supply voltage V_{batFL} and Endurance En_{FL} within specified range (see **"Reference Documents" on Page 80**.)*
This function returns -1 and has no effect if executed in DEBUG mode.

2.28.2 Actions

- Get current EEPROM content
- Add int value to content
- Write back new EEPROM content

2.28.3 Prototype

signed char **WR_EEInt**(unsigned char Address, unsigned int EEInt)

2.28.4 Inputs

Table 119 Wr_EEInt: Input Parameters

Register / Address	Type	Name	Description
R7	unsigned char	Address	Address of int location (0x00-0x1D)
R5	unsigend int	EEInt	Int value to write

2.28.5 Outputs

Table 120 Wr_EEInt: Output values

Register/ Address	Type	Name	Description
R7	signed char	Statusbyte	0: success -1: Address out of range -2: Lockbyte 3 set

2.28.6 Resource Usage

Table 121 Wr_EEInt: Resources

Type	used or modified
Registers	R0, R1, R2, R3, R4, R5, R6, R7
SFR	A, B, PSW, DPL, DPH, CRC0, CRC1, CRCC, CRCD, DIVIC, TCON, TH1, TL1, FCS, FCSERM, FCPP0, FCTKAS, FCSP, IE
Stack	13 Bytes

2.28.7 Execution Information

Table 122 Wr_EEInt: Execution Information

Value	typ	max	Unit	Conditions
Instruction Cycles	4760	212750		typ.: Writing of a line without erasing max.: Writing a line with erasing (startup and every fourth time)

2.29 Wr_EELong()

2.29.1 Description

This function writes a long value in the EEPROM Emulation.

Attention: This function shows only effect if the Lockbyte 3 that protects the User Configuration Sectors is NOT set.

Note: The application software has to ensure that FLASH is only programmed or erased when all required environmental conditions are fulfilled. Special care has to be taken that ambient temperature T_{FL}, supply voltage V_{batFL} and Endurance En_{FL} within specified range (see ***"Reference Documents" on Page 80****.)*
This function returns -1 and has no effect if executed in DEBUG mode.

2.29.2 Actions

- Get current EEPROM content
- Add long value to content
- Write back new EEPROM content

2.29.3 Prototype

signed char **WR_EELong**(unsigned char Address, unsigned long idata *EELong)

2.29.4 Inputs

Table 123 Wr_EELong: Input Parameters

Register / Address	Type	Name	Description
R7	unsigned char	Address	Address of byte location (0x00-0x1B)
R5	unsigend long idata*	EELong	Pointer to long value to write

2.29.5 Outputs

Table 124 Wr_EELong: Output values

Register/ Address	Type	Name	Description
R7	signed char	Statusbyte	0: success -1: Address out of range -2: Lockbyte 3 set

2.29.6 Resource Usage

Table 125 Wr_EELong: Resources

Type	used or modified
Registers	R0, R1, R2, R3, R4, R5, R6, R7
SFR	A, B, PSW, DPL, DPH, CRC0, CRC1, CRCC, CRCD, DIVIC, TCON, TH1, TL1, FCS, FCSERM, FCPP0, FCTKAS, FCSP, IE
Stack	13 Bytes

2.29.7 Execution Information

Table 126 Wr_EELong: Execution Information

Value	typ	max	Unit	Conditions
Instruction Cycles	4760	212750		typ.: Writing of a line without erasing max.: Writing a line with erasing (startup and every fourth time)

2.30 Wr_EEString()

2.30.1 Description

This function writes a string in the EEPROM Emulation.

Attention: This function shows only effect if the Lockbyte 3 that protects the User Configuration Sectors is NOT set.

*Note: The application software has to ensure that FLASH is only programmed or erased when all required environmental conditions are fulfilled. Special care has to be taken that ambient temperature T_{FL}, supply voltage V_{batFL} and Endurance En_{FL} within specified range (see **"Reference Documents" on Page 80**.)*
This function returns -1 and has no effect if executed in DEBUG mode.

2.30.2 Actions

- Get current EEPROM content
- Add string to content
- Write back new EEPROM content

2.30.3 Prototype

signed char **WR_EEString**(unsigned char Address, unsigned char Length, unsigned char idata * Buffer)

2.30.4 Inputs

Table 127 Wr_EEString: Input Parameters

Register / Address	Type	Name	Description
R7	unsigned char	Address	Address of string location (0x00-0x1E)
R5	unsigend char	Length	String length (0-31) (addr + length < 0x20)
R3	unsigend char idata*	Buffer	Pointer to buffer with string content

2.30.5 Outputs

Table 128 Wr_EEString: Output values

Register/ Address	Type	Name	Description
R7	signed char	Statusbyte	0: success -1: Address out of range -2: Lockbyte 3 set

2.30.6 Resource Usage

Table 129 Wr_EEString: Resources

Type	used or modified
Registers	R0, R1, R2, R3, R4, R5, R6, R7
SFR	A, B, PSW, DPL, DPH, CRC0, CRC1, CRCC, CRCD, DIVIC, TCON, TH1, TL1, FCS, FCSERM, FCPP0, FCTKAS, FCSP, IE
Stack	13 Bytes

2.30.7 Execution Information

Table 130 Wr_EEString: Execution Information

Value	typ	max	Unit	Conditions
Instruction Cycles	4760	212750		typ.: Writing of a line without erasing max.: Writing a line with erasing (startup and every fourth time)

2.31 Get_EEByte()

2.31.1 Description

This function reads a single byte in the EEPROM Emulation.

2.31.2 Actions

- Read byte from current EEPROM content

2.31.3 Prototype

unsigned char **Get_EEByte**(unsigned char Address)

2.31.4 Inputs

Table 131 Get_EEByte: Input Parameters

Register / Address	Type	Name	Description
R7	unsigned char	Address	Address of byte location (0x00-0x1E)

2.31.5 Outputs

Table 132 Get_EEByte: Output values

Register/ Address	Type	Name	Description
R7	unsigned char	EEByte	0: Address out of range/ No data written 0xXX: success

2.31.6 Resource Usage

Table 133 Get_EEByte: Resources

Type	used or modified
Registers	R6, R7
SFR	A, B, PSW, DPL, DPH, IE
Stack	5 Bytes

2.31.7 Execution Information

Table 134 Get_EEByte: Execution Information

Value	typ	max	Unit	Conditions
Instruction Cycles		60		

2.32 Get_EEInt()

2.32.1 Description

This function reads an int in the EEPROM Emulation.

2.32.2 Actions

- Read an int from current EEPROM content

2.32.3 Prototype

unsigned int **Get_EEInt**(unsigned char Address)

2.32.4 Inputs

Table 135 Get_EEInt: Input Parameters

Register / Address	Type	Name	Description
R7	unsigned char	Address	Address of byte location (0x00-0x1D)

2.32.5 Outputs

Table 136 Get_EEInt: Output values

Register/ Address	Type	Name	Description
R6,R7	unsigned int	EEInt	0: Address out of range/ No data written 0xXXXX: success

2.32.6 Resource Usage

Table 137 Get_EEInt: Resources

Type	used or modified
Registers	R6, R7
SFR	A, B, PSW, DPL, DPH, IE
Stack	5 Bytes

2.32.7 Execution Information

Table 138 Get_EEInt: Execution Information

Value	typ	max	Unit	Conditions
Instruction Cycles		64		

2.33 Get_EELong()

2.33.1 Description

This function reads a long in the EEPROM Emulation.

2.33.2 Actions

- Read long from current EEPROM content

2.33.3 Prototype

unsigned long **Get_EELong**(unsigned char Address)

2.33.4 Inputs

Table 139 Get_EELong: Input Parameters

Register / Address	Type	Name	Description
R7	unsigned char	Address	Address of byte location (0x00-0x1B)

2.33.5 Outputs

Table 140 Get_EELong: Output values

Register/ Address	Type	Name	Description
R4-R7 (MSB R4)	unsigned long	EELong	0: Address out of range/ No data written 0xXXXXXXXX: success

2.33.6 Resource Usage

Table 141 Get_EELong: Resources

Type	used or modified
Registers	R4, R5, R6, R7
SFR	A, B, PSW, DPL, DPH, IE
Stack	5 Bytes

2.33.7 Execution Information

Table 142 Get_EELong: Execution Information

Value	typ	max	Unit	Conditions
Instruction Cycles		72		

2.34 Get_EEString()

2.34.1 Description

This function reads a string in the EEPROM Emulation.

2.34.2 Actions

- Read string from current EEPROM content

2.34.3 Prototype

signed char **Get_EEString**(unsigned char Address, unsigned char Length, unsigend char idata * Buffer)

2.34.4 Inputs

Table 143 Get_EEString: Input Parameters

Register / Address	Type	Name	Description
R7	unsigned char	Address	Address of byte location (0x00-0x1E)
R5	unsigend char	Length	String length (0-31) (addr + length < 0x20)
R3	unsigned char idata*	Buffer	Pointer to location to place string content

2.34.5 Outputs

Table 144 Get_EEString: Output values

Register/ Address	Type	Name	Description
R7	signed char	Status	0: success -1: Address out of range/string too long -2: No data written

2.34.6 Resource Usage

Table 145 Get_EEString: Resources

Type	used or modified
Registers	R0, R3, R5, R6, R7
SFR	A, B, PSW, DPL, DPH, IE
Stack	5 Bytes

2.34.7 Execution Information

Table 146 Get_EEString: Execution Information

Value	typ	max	Unit	Conditions
Instruction Cycles		63 + 7*Length		

3 Reference Documents

This section contains documents used for cross- reference throughout this document.

Table 147 Reference Documents

Reference Number	Document description
[1]	PMA5110_DS_Preliminary_V1_2.pdf

www.infineon.com

Published by Infineon Technologies AG