ES_LPC1850/30/20/10 Rev A

Errata sheet LPC1850, LPC1830, LPC1820, LPC1810 Rev A

Rev. 4 — 25 January 2013 Errata sheet

Document information

Info	Content
Keywords	LPC1850FET256; LPC1850FET180; LPC1850FBD208; LPC1830FET256; LPC1830FET180; LPC1830FET100; LPC1830FBD144; LPC1820FET100; LPC1820FBD144; LPC1810FET100; LPC1810FBD144; Rev A errata
Abstract	This errata sheet describes both the known functional problems and any deviations from the electrical specifications known at the release date of this document. Each deviation is assigned a number and its history is tracked in a table.

Revision history

Rev	Date	Description
4	20130125	• Added I2C.1.
3.1	20121130	• Corrected part number typo in Section 3.6. • Added CDC.1. • Updated workaround for IBAT.1.
3	20121015	• Updated C_CAN.1. • Added Rev. C. • Removed AES.1, ETM.1, RGU.1, SPIFI.2; documented in user manual.
2.2	20120808	• Added IBAT.2 and RGU.1. • Corrected C_CAN0/C_CAN1 peripheral assignment.
2.1	20120713	• Added C_CAN.1.
2	20120601	• Added ISP.1, ETM.1, IAP.1, PMC.1 and IBAT.1.
1.3	20120401	• Updated SPIFI.1. • Added SPIFI.2. • Removed ADC.1 and USB0.1.
1.2	20120201	• Added OTP.1.
1.1	20120120	• Added ADC.1.
1	20111111	• Initial version.

Contact information

For more information, please visit: **http://www.nxp.com**

For sales office addresses, please send an email to: **salesaddresses@nxp.com**

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2013. All rights reserved.

1. Product identification

The LPC1850/30/20/10 devices (hereafter referred to as ‘LPC18x0’) typically have the following top-side marking:

LPC18x0xxxxxx

xxxxxxxx

xxxYYWWxR[x]

The last/second to last letter in the last line (field ‘R’) will identify the device revision. This Errata Sheet covers the following revisions of the LPC18x0:

Table 1. Device revision table

Revision identifier (R)	Revision description
‘A’	Initial device revision
‘C’	Second device revision

Field ‘YY’ states the year the device was manufactured. Field ‘WW’ states the week the device was manufactured during that year.

2. Errata overview

Table 2. Functional problems table

Functional problems	Short description	Revision identifier	Detailed description
BOOT.1	USB1 boot is not functional	‘A’	Section 3.1
C_CAN.1	Writes to CAN registers write through to other peripherals	‘A’, ‘C’	Section 3.2
CDC.1	The CDC class USB ROM drivers return a STALL condition	‘A’	Section 3.3
I2C.1	In the slave-transmitter mode, the device set in the monitor mode must write a dummy value of 0xFF into the DAT register.	‘A’, ‘C’	Section 3.4
IAP.1	In-Application Programming API not present on flashless parts	‘A’, ‘C’	Section 3.5
ISP.1	Part ID format incorrect	‘A’, ‘C’	Section 3.6
MCPWM.1	MCPWM abort pin not functional	‘A’, ‘C’	Section 3.7
OTP.1	OTP ROM driver may not program boot source	‘A’, ‘C’	Section 3.8
PMC.1	PMC.x power management controller fails to wake up from deep sleep, power down, or deep power down	‘A’, ‘C’	Section 3.9
SPIFI.1	The ROM driver does not properly re-initialize the external flash device	‘A’	Section 3.10

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2013. All rights reserved.

Table 3. AC/DC deviations table

AC/DC deviations	Short description	Product version(s)	Detailed description
IBAT.1	VBAT supply current higher than expected	'A', 'C'	Section 4.1
IBAT.2	VBAT supply current higher than expected	'A'	Section 4.2
PWR.1	Deep sleep and Power-down mode consume more current than expected	'A'	Section 4.3
PWR.2	VDDIO consumes more current than expected	'A'	Section 4.4

Table 4. Errata notes table

Errata notes	Short description	Revision identifier	Detailed description
n/a	n/a	n/a	n/a

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2013. All rights reserved.

3. Functional problems detail

3.1 BOOT.1: USB1 boot is not functional

Introduction:

The internal ROM memory is used to store the boot code of the LPC18x0. After a reset, the ARM processor will start its code execution from this memory. The boot ROM memory includes the following features:

- …Boot from USB1….

Problem:

Boot from USB1 is not functional. This does not affect use of USB1 after bootup.

Work-around:

USB0 can be used to boot the part.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2013. All rights reserved.

3.2 C_CAN.1: Writes to CAN registers write through to other peripherals

Introduction:

Controller Area Network (CAN) is the definition of a high performance communication protocol for serial data communication. The C_CAN controller is designed to provide a full implementation of the CAN protocol according to the CAN Specification Version 2.0B. The C_CAN controller allows to build powerful local networks with low-cost multiplex wiring by supporting distributed real-time control with a very high level of security.

Problem:

On the LPC18x0, there is an issue with the C_CAN controller AHB bus address decoding that applies to both C_CAN controllers. It affects the C_CAN controllers when peripherals on the same bus are used. Writes to the ADC, DAC, I2C, and I2S peripherals can update registers in the C_CAN controller. Specifically, writes to I2C0, MCPWM, and I2S can affect C_CAN1. Writes to I2C1, DAC, ADC0, and ADC1 can affect C_CAN0. The spurious C_CAN controller writes will occur at the address offset written to the other peripherals on the same bus. For example, a write to ADC0 CR register which is at offset 0 in the ADC, will result in the same value being written to the C_CAN0 CNTL register which is at offset 0 in the C_CAN controller. Writes to the C_CAN controller will not affect other peripherals.

Work-around:

Workarounds include: Using a different C_CAN peripheral. Peripherals I2C1, DAC, ADC0, and ADC1 can be used at the same time as C_CAN1 is active without any interference. The I2C0, MCPWM, and I2S peripherals can be used at the same time as C_CAN0 is active without any interference. Another workaround is to gate the register clock to the CAN peripheral in the CCU. This will prevent any writes to other peripherals from taking effect in the CAN peripheral. However, gating the CAN clock will prevent the CAN peripheral from operating and transmitting or receiving messages. This workaround is most useful if your application is modal and can switch between different modes such as an I2S mode and a CAN mode. Another workaround is to avoid writes to the peripherals while CAN is active. For example, the ADC could be configured to sample continuously or when triggered by a timer, before the CAN is configured. Afterwards, C_CAN0 can be used since the ADC will operate without requiring additional writes.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2013. All rights reserved.

3.3 CDC.1: The CDC class USB ROM drivers return a STALL condition

Introduction:

The CDC class USB ROM drivers have a bug that causes some Windows terminal emulation programs like Hyperterm to fail to connect. This bug only exists in revision 'A' parts.

Problem:

The CDC class USB ROM drivers return a STALL condition after the DATA stage of a Set Line Coding CONTROL transfer from the host. The correct behavior is to return an ACK.

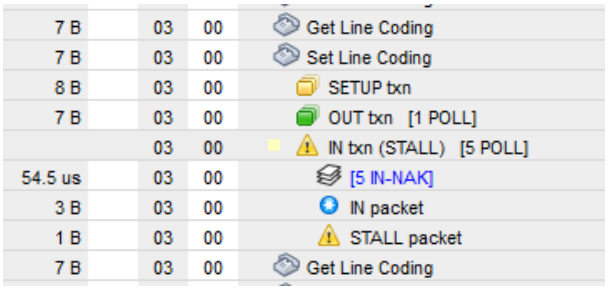

Fig 1. Incorrect

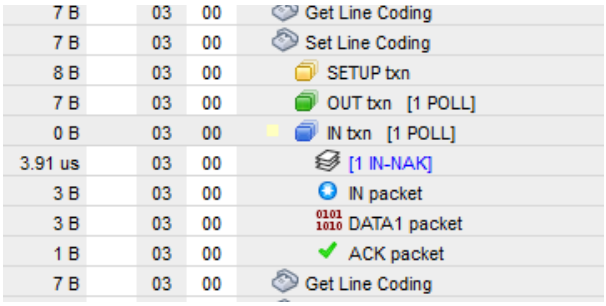

Fig 2. Correct

Work-around:

The way to work around this bug is to override the default CDC class specific endpoint 0 handler and handle the processing of OUT packets in your application. Use the following source code in red to accomplish this.

```
USB_EP_HANDLER_T g_defaultCdcHdlr; // default CDC handler
ErrorCode_t CDC_ep0_override_hdlr(USBD_HANDLE_T hUsb, void* data, uint32_t event)
{
  USB_CORE_CTRL_T* pCtrl = (USB_CORE_CTRL_T*)hUsb;
  USB_CDC_CTRL_T* pCdcCtrl = (USB_CDC_CTRL_T*) data;
  ErrorCode_t ret = ERR_USBD_UNHANDLED;

  if( (event == USB_EVT_OUT) &&
      (pCtrl->SetupPacket.bmRequestType.BM.Type == REQUEST_CLASS) &&
      (pCtrl->SetupPacket.bmRequestType.BM.Recipient == REQUEST_TO_INTERFACE) &&
      ((pCtrl->SetupPacket.wIndex.WB.L == pCdcCtrl->cif_num) || /* IF number correct?
     */
      (pCtrl->SetupPacket.wIndex.WB.L == pCdcCtrl->dif_num)) ) {

    pCtrl->EP0Data.pData -= pCtrl->SetupPacket.wLength;
    ret = pCdcCtrl->CIC_SetRequest(pCdcCtrl, &pCtrl->SetupPacket,
     &pCtrl->EP0Data.pData,
       pCtrl->SetupPacket.wLength);
    if ( ret == LPC_OK) {

      USBD_API->core->StatusInStage(pCtrl);                    /* send Acknowledge */
    }
```


All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2013. All rights reserved.

```
  } else {
      ret = g_defaultCdcHdlr(hUsb, data, event);
  }
   return ret;
}
void UsbdCdc_Init(void)
{
      .
      .
      .

      USBD_API->cdc->init(UsbHandle, &cdc_param, &UsbdCdcHdlr);

      {
            // This code must be placed immediately after the call to
      USBD_API->cdc->init()
            USB_CORE_CTRL_T* pCtrl = (USB_CORE_CTRL_T*)UsbHandle;
            /* store the default CDC handler and replace it with ours */
            g_defaultCdcHdlr = pCtrl->ep0_hdlr_cb[pCtrl->num_ep0_hdlrs - 1];
            pCtrl->ep0_hdlr_cb[pCtrl->num_ep0_hdlrs - 1] = CDC_ep0_override_hdlr;
      }
}
```


All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2013. All rights reserved.

3.4 I2C.1: In the slave-transmitter mode, the device set in the monitor mode must write a dummy value of 0xFF into the DAT register

Introduction:

The I2C monitor allows the device to monitor the I2C traffic on the I2C bus in a non-intrusive way.

Problem:

In the slave-transmitter mode, the device set in the monitor mode must write a dummy value of 0xFF into the DAT register. If this is not done, the received data from the slave device will be corrupted. To allow the monitor mode to have sufficient time to process the data on the I2C bus, the device may need to have the ability to stretch the I2C clock. Under this condition, the I2C monitor mode is not 100% non-intrusive.

Work-around:

When setting the device in monitor mode, enable the ENA_SCL bit in the MMCTRL register to allow clock stretching.

Software code example to enable the ENA_SCL bit:

```
LPC_I2C_MMCTRL |= (1<<1); //Enable ENA_SCL bit
```

In the I2C ISR routine, for the status code related to the slave-transmitter mode, write the value of 0xFF into the DAT register to prevent data corruption. In order to avoid stretching the SCL clock, the data byte can be saved in a buffer and processed in the Main loop. This ensures the SI flag is cleared as fast as possible.

Software code example for the slave-transmitter mode:

```
case 0xA8:     // Own SLA + R has been received, ACK returned
case 0xB0:
case 0xB8:     // data byte in DAT transmitted, ACK received
case 0xC0:     // (last) data byte transmitted, NACK received
case 0xC8:     // last data byte in DAT transmitted, ACK received
    DataByte = LPC_I2C->DATA_BUFFER;//Save data. Data can be process in Main loop
    LPC_I2C->DAT = 0xFF;            // Pretend to shift out 0xFF
    LPC_I2C->CONCLR = 0x08;         // clear flag SI
break;
```


All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2013. All rights reserved.

3.5 IAP.1: In-Application Programming API not present on flashless parts

Introduction:

The LPC18x0 microcontrollers contain an API for In-Application Programming of flash memory. This API also allows identification of the part.

Problem:

On the LPC18x0 microcontrollers, the IAP API is not present. The ISP interface is present which allows the part to be identified externally (via the UART) but part identification is not possible internally using the IAP call because it is not implemented.

Work-around:

To detect whether or not the IAP API is present, check the IAP entry point at 0x10400100. If it is set to 0x12345678, the part does not implement IAP. The first word of the Part ID can be read directly from OTP at 0x40045000. The second word of the Part ID is always '0' on flashless parts.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2013. All rights reserved.

3.6 ISP.1: Part ID format incorrect

Introduction:

A reduced set of In-System-Programming (ISP) commands are supported for flashless parts. The ISP 'J' command can be used to query the part identification number.

Problem:

On the LPC18x0 microcontrollers, the J command returns incorrectly formatted data. Instead of returning two words (plus the return code) as specified in the User's Manual, IAP command 54 and ISP command 'J' only return a single word (plus return code). That single word contains the first word of the part identification number with the first 16 bits swapped with the last 16 bits. For example, an LPC1850FET256 will return 0x0830A000 instead of the correct value, 0xA0000830.

Work-around:

When using ISP, if only one word of data is returned, swap the two 16-bit segments of the word and assume the second word of data is 0.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2013. All rights reserved.

3.7 MCPWM.1: MCPWM Abort pin is not functional

Introduction:

The Motor Control PWM engine is optimized for three-phase AC and DC motor control applications, but can be used in many other applications that need timing, counting, capture, and comparison. The MCPWM contains a global Abort input that can force all of the channels into a passive state and cause an interrupt.

Problem:

The MCPWM Abort input is not functional.

Work-around:

The MCPWM Abort function can be emulated in software with the use of a non-maskable interrupt combined with an interrupt handler that shuts down the PWM. This will result in a small delay on the order of 50 main clock cycles or about 1/3 of a microsecond at 150 MHz. Alternatively, the State Configurable Timer (SCT) can be configured to implement MCPWM functionality including an Abort input. The SCT can respond to external inputs in one clock cycle.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2013. All rights reserved.

3.8 OTP.1: OTP ROM driver may not program boot source

Introduction:

The LPC18x0 contain OTP memory which can configure the boot source, as well as a set of routines in ROM to program the boot source into OTP memory.

Problem:

There is a problem in the OTP boot source programming code in ROM which requires registers to be initialized in order to ensure successful boot source OTP programming.

Work-around:

1. Add this function to your program.

```
void OTP_fix(volatile unsigned dummy0,volatile unsigned dummy1,volatile unsigned
dummy2,volatile unsigned dummy3)
{
}
```

2. Call this function before calling otp_ProgBootSrc.

```
rval = otp_Init();
OTP_fix(0,0,0,0);
rval = otp_ProgBootSrc(OTP_BOOTSRC_SPIFI);
```

This will be fixed in the next boot ROM revision.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2013. All rights reserved.

3.9 PMC.1: PMC.x power management controller fails to wake up from Deep Sleep, Power Down, or Deep Power Down

Introduction:

The PMC implements the control sequences to enable transitioning between different power modes and controls the power state of each peripheral. In addition, wake-up from any of the power-down modes based on hardware events is supported.

Problem:

When the chip is in a transition from active to Deep Sleep, Power Down, or Deep Power Down, wakeup events are not captured and they will block further wakeup events from propagating. The time window for this transition is 6 uS and is not affected by the chip clock speed. After a wakeup event is received during the PMC transition, the chip can only recover by using an external hardware reset or by cycling power.

Work-around:

Make sure that a wakeup signal is not received during the Deep Sleep, Power Down, or Deep Power Down transition period. An example circuit to work around this could include an external 6 uS one shot which could be triggered via software using a GPIO line when entering Deep Sleep, Power Down, or Deep Power Down mode. The one-shot's output could be used to gate the wakeup signal(s) to prevent receiving a wakeup signal during the PMC transition period. Depending on the system design, it may also be needed to latch the wakeup signal(s) so that they will still be present after the one-shot's 6 uS timeout.

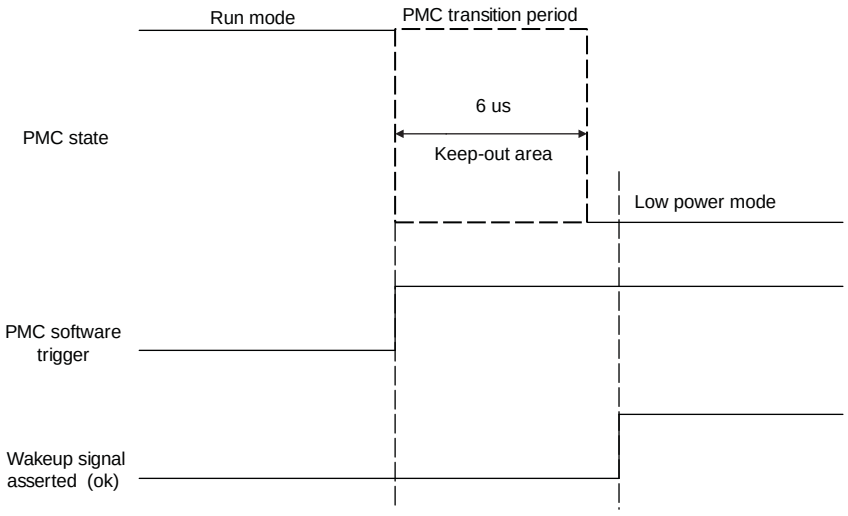

Fig 3. PMC wakeup keep-out area

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2013. All rights reserved.

3.10 SPIFI.1: The ROM driver does not properly re-initialize the external serial flash device

Introduction:

The SPI Flash Interface (SPIFI) allows low-cost serial flash memories to be connected to the ARM Cortex-M3 processor with little performance penalty compared to parallel flash devices with higher pin count. SPIFI provides a memory-mapped area where the contents of the external serial flash memory appear.

Problem:

The built-in SPIFI ROM driver used for booting does not properly re-initialize the external serial flash device if it is already set up for "no opcode" or "continuous read" mode. This affects use after unplanned resets such as a hardware reset or watchdog timer reset. Booting from SPIFI is affected and may not be successful until after the 60 second boot failure timeout if the external serial flash device is in "no opcode" or "continuous read" mode.

Work-around:

During a planned reboot, remove the external QSPI flash from no opcode mode before resetting the CPU by using the SPIFI driver library's cancel_mem_mode call. The SPIFI driver library is available from lpcware.com. In the event of an unplanned reset, the driver will initialize the flash device if it is called a second time so an external watchdog could be provided to reset the CPU in case of boot failure from SPIFI. Finally there is a built-in 60-second boot timeout which will result in a successful boot after one minute in the event of a failure.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2013. All rights reserved.

4. AC/DC deviations detail

4.1 IBAT.1: VBAT supply current higher than expected

Introduction:

The LPC18x0 contain a Real-Time Clock which measures the passage of time. The RTC has an ultra-low power design to support battery powered systems with a dedicated battery supply pin.

Problem:

On the LPC18x0, high current consumption of about 70 uA may occur on the VBAT power supply pin.

Work-around:

VBAT current consumption can be lowered significantly by configuring the RTC_ALARM pin as "Inactive" by setting the ALARMCTRL 7:6 field in CREG0 to 0x3. These bits persist through power cycles and reset while VBAT is present.

For CREG0[13:12] reserved value 0x3 should be used; this value should be set once after a power on reset.

4.2 IBAT.2: VBAT supply current higher than expected

Introduction:

The LPC18x0 contain a Real-Time Clock which measures the passage of time. The RTC has an ultra-low power design to support battery powered systems with a dedicated battery supply pin.

Problem:

On the LPC18x0, high current consumption of about 15 uA may occur on the VBAT power supply pin despite applying the workaround in IBAT.1.

Work-around:

The problem is caused by a design error and there is currently no work-around.

4.3 PWR.1: Deep sleep and Power-down modes consume more current than expected

Introduction:

The LPC18x0 contain several low-power modes. The PMC implements the control sequences to enable transitioning between different power modes and controls the power state of each peripheral.

Problem:

A design error results in about 15 µA higher current consumption during Deep Sleep and Power Down mode.

Work-around:

None.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2013. All rights reserved.

4.4 PWR.2: VDDIO consumes more current than expected

Introduction:

The LPC18x0 contain several low-power modes. Most of the low power modes are designed to trigger external shut down of VDDIO using an external switch for lowest power consumption.

Problem:

A design error results in about 60 µA higher current consumption on VDDIO during all active and low power modes except for deep power down.

Work-around:

VDDIO can be switched off upon entry to low power modes with addition of an external switch.

5. Errata notes detail

5.1 n/a

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2013. All rights reserved.

6. Legal information

6.1 Definitions

Draft — The document is a draft version only. The content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included herein and shall have no liability for the consequences of use of such information.

6.2 Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the *Terms and conditions of commercial sale* of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

6.3 Trademarks

Notice: All referenced brands, product names, service names and trademarks are the property of their respective owners.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2013. All rights reserved.

7. Contents

1 Product identification 3
2 Errata overview 3
3 Functional problems detail 5
3.1 BOOT.1: USB1 boot is not functional 5
Introduction: 5
Problem: 5
Work-around: 5
3.2 C_CAN.1: Writes to CAN registers write through to other peripherals. 6
Introduction: 6
Problem: 6
Work-around: 6
3.3 CDC.1: The CDC class USB ROM drivers return a STALL condition 7
Introduction: 7
Problem: 7
Work-around: 7
3.4 I2C.1: In the slave-transmitter mode, the device set in the monitor mode must write a dummy value of 0xFF into the DAT register 9
Introduction: 9
Problem: 9
Work-around: 9
3.5 IAP.1: In-Application Programming API not present on flashless parts. 10
Introduction: 10
Problem: 10
Work-around: 10
3.6 ISP.1: Part ID format incorrect 11
Introduction: 11
Problem: 11
Work-around: 11
3.7 MCPWM.1: MCPWM Abort pin is not functional 12
Introduction: 12
Problem: 12
Work-around: 12
3.8 OTP.1: OTP ROM driver may not program boot source. 13
Introduction: 13
Problem: 13
Work-around: 13
3.9 PMC.1: PMC.x power management controller fails to wake up from Deep Sleep, Power Down, or Deep Power Down 14
Introduction: 14
Problem: 14
Work-around: 14
3.10 SPIFI.1: The ROM driver does not properly re-initialize the external serial flash device . . 15
Introduction: 15
Problem:. 15
Work-around: 15
4 AC/DC deviations detail 16
4.1 IBAT.1: VBAT supply current higher than expected 16
Introduction: 16
Problem:. 16
Work-around: 16
4.2 IBAT.2: VBAT supply current higher than expected 16
Introduction: 16
Problem:. 16
Work-around: 16
4.3 PWR.1: Deep sleep and Power-down modes consume more current than expected 16
Introduction: 16
Problem:. 16
Work-around: 16
4.4 PWR.2: VDDIO consumes more current than expected 17
Introduction: 17
Problem:. 17
Work-around: 17
5 Errata notes detail 17
5.1 n/a 17
6 Legal information 18
6.1 Definitions 18
6.2 Disclaimers 18
6.3 Trademarks 18
7 Contents. 19

Please be aware that important notices concerning this document and the product(s) described herein, have been included in section 'Legal information'.

© NXP B.V. 2013. **All rights reserved.**

For more information, please visit: http://www.nxp.com
For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 25 January 2013
Document identifier: ES_LPC18X0_A