ERRATA SHEET

Date: **May 2, 2005**
Document Release: **Version 1.6**
Device Affected: **LPC2119**

This errata sheet describes both the functional deviations 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 at the end of the document.

2005 May 2

Philips Semiconductors

PHILIPS

Identification:

The LPC2119 devices typically have the following top-side marking:

LPC2119xxx

xxxxxxx

xxYYWW R

The last letter in the third line (field 'R') will identify the device revision. This Errata Sheet covers the following revisions of the LPC2119:

Revision Identifier (R)	Comment
'A'	Initial device revision

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

Functional Deviations of LPC2119

IAP.1: Flash memory programming interface timing problem

Introduction: The Flash memory on the LPC2119 offers In-Application Programming (IAP) functionality. The IAP routines are part of the on-chip boot loader software, which controls the interface between the digital logic and the Flash memory. Please note that all programming methods (JTAG, ISP, IAP) use IAP calls.

Problem: Due to a timing problem in the interface between the Flash block and the digital logic the following problem may occur:

If the boot loader revision in the device is previous to V1.63 then in up to 10% of the devices the Flash memory interface, at some point during an IAP programming or erase operation, may never return from the IAP call. Please note that devices that pass the IAP programming are functional and do not suffer from any long-term reliability problems.

Devices with a date code prior to 0420 (manufactured before week 20 in 2004) are generally affected by this problem unless you receive devices with updated boot loader software from your distributor. Parts marked with date code 0420 or later are not affected by this problem. Please refer to page 2 of this document for details on how to identify the date code.

Work-around:

1) The on-chip boot-loader software can be updated via ISP to correct this issue. The boot loader update files can be downloaded here:

http://www.semiconductors.philips.com/files/products/standard/microcontrollers/utilities/lpc2000_bl_update.zip

The boot-loader version can be read out using the Philips Flash ISP Utility which can be found here:

http://www.semiconductors.philips.com/files/products/standard/microcontrollers/utilities/lpc2000_flash_utility.zip

Both links can also be found on the product information page for this product under the Support & Tools section which can be found here:

http://www.semiconductors.philips.com/pip/LPC2119FBD64.html

2) Limiting the external clock frequency to 12 MHz AND making sure the on-chip PLL is turned OFF while programming any part of the Flash memory reduces the likelihood of the occurrence significantly. During In-System-Programming the PLL is turned off by default.

ADC.1: First two ADC conversions in burst mode from same channel

Introduction: In burst mode the A/D converter does repeated conversions at the rate selected by the CLKS field in the ADCR, scanning (if necessary) through the pins selected by 1s in the SEL field. The first conversion after the start corresponds to the least-significant 1 in the SEL field, then higher numbered 1-bits (pins) if applicable. Repeated conversions can be terminated by clearing this bit.

Problem: In burst conversion mode, the first two conversions (after setting the mode) will be of the same, lowest-numbered, of the selected channels.

Work-around: Ignore the first conversion, always check the CHN bits to confirm the channel converted.

ADC.2: First conversion from channel specified by previous SEL setting

Introduction: The ADCR SFR contains bits to enable the ADC burst mode (BURST), start the conversion in software control mode (START), and to select the channel to begin converting (SEL).

Problem: In burst mode, If the BURST bit is set before or simultaneously to (using the STR instruction for example), the SEL bits, then the first channel converted will correspond to the previous SEL bit settings.

In software control mode (only when using external trigger), if the START bits are set before or simultaneously to (using the STR instruction for example) the SEL bits, then the first channel converted will correspond to the previous SEL bit settings.

Work-around: Set the BURST/START bit(s) after setting the SEL bits.

ADC.3: Incorrect scan pattern

Introduction: In hardware scan mode multiple ADC channels may be selected as part of the scan by selecting the appropriate bits in the SEL field in the ADCR register.

Problem: Certain hardware scanning patterns for the A/D Converter do not operate properly. Selecting channel 2 only leads to alternate sampling of channels 2 and 3. Selecting channels 1 and 2 leads to sampling channel 1 for the first conversion, then sampling channel 2 on every subsequent conversion.

Work-around: None. Do not use the sampling patterns “channel 2 only" or "channels 1 and 2". This problem has no effect on software conversion, nor on any other patterns other than the two noted above.

ADC.4 Global powerdown does not power down the ADC

Introduction: Setting the PD bit (bit 1) in PCON stops all clocks and powers down the peripherals. The ADC is powered down by clearing the PDN bit (bit 21) in the ADCR register, setting the bit powers up (enables) the ADC.

Problem: If the PDN in ADCR is set, setting the PD bit in PCON will not power down the ADC.

Work-around: Clear the PDN bit in the ADCR SFR to turn off the ADC prior to setting the PD bit in PCON.

ADC.5	**Edge triggered ADC conversion start error**
Introduction:	When the START field of the ADCR register contains 010-111 the EDGE bit in ADCR will determine whether a conversion is started on a rising or falling edge of the selected CAP/MAT signal. EDGE=0 selects rising edge detection, EDGE=1 selects falling edge detection. (On CAP/MAT pin)
Problem:	If the state of the selected CAP/MAT signal is 1 and EDGE is set to detect rising edges (EDGE = 0) or, if detection of falling edges is selected (EDGE = 1) and the state of the selected CAP/MAT signal is 0, an ADC conversion will immediately be initiated when the START bits are written to. So the first conversion behaves as a level triggered event rather than edge triggered.
Work-around:	Clear the selected CAP/MAT signal for EDGE = 0 or set the selected CAP/MAT signal for EDGE = 1 before writing 010-111 to START field. Alternatively, discard the first conversion after writing to the start bits.
ADC.6	**Writing to ADCR while conversion in progress**
Introduction:	Writing to ADCR while a conversion is in progress should set the DONE bit and start a new conversion.
Problem:	In actuality, if the ADCR is written to within 2.5 ADC_clock cycles, a new conversion is started but the DONE bit is not set. If the ADCR is written to after 2.5 ADC_clocks, but within a conversion time, the DONE bit is set within one ADC_clock and a new conversion is started.
Work-around:	Do not write to ADCR until the conversion is complete.
SPI.1	**Unintentional clearing of SPI interrupt flag**
Introduction:	The SPI interrupt flag is set by the SPI interface to generate an interrupt. It is cleared by writing a 1 to this bit.
Problem:	A write to any register associated with the SPI peripheral will clear the SPI interrupt register.
Work-around:	Avoid writing to SPI registers while transmissions are in progress or while SPI interrupts are pending.
EXTINT.1	**Corruption of VPBDIV via EXTPOLAR or EXTMODE**
Introduction:	The VPBDIV register controls the rate of the VPB clock in relation to the processor clock. EXTPOLAR and EXTMODE determine the operating parameters of the external interrupts.
Problem:	A write to either the external interrupt polarity register (EXTPOLAR) or the external interrupt mode register (EXTMODE) will corrupt the VPBDIV register. A read of either EXTPOLAR or EXTMODE will be corrupted BY the VPBDIV register. If VPBDIV is “1” or “2” prior to any write to EXTPOLAR or EXTMODE, the CPU will hang up on the write to EXTPOLAR or EXTMODE.
Work-around:	If VPBDIV is non-zero, write all zeroes to VPBDIV before reading or writing EXTMODE or EXTPOLAR, then write the proper value back to VPBDIV. In most applications this is a known and fixed value, but if there is a possibility of dynamic changes in VPBDIV, software will need to read VPBDIV, write zero to VPBDIV, read or write EXTMODE and/or EXTPOLAR, and then rewrite the value previously read from VPBDIV.

EXTINT.2 **Incorrect setting of EXTMODE and/or EXTPOLAR register while trying to set them to desired value**

Introduction: EXTPOLAR and EXTMODE determine the operating parameters of the external interrupts.

Problem: As an illustration, trying to set EXTMODE to 0x1 or 0xd would result in EXTMODE to be set to 0x0 instead.

Work-around: This problem is related to EXTINT.1 and hence the same workaround applies with an additional step.

The steps involved in the configuration of the EXTMODE and/or EXTPOLAR would be as follow:-
1. Write 0x0 to VPBDIV

2. Write the desired value to EXTMODE or EXTPOLAR register

3. Write the same value to VPBDIV (additional step)

4. Restore the VPBDIV to the previously saved value or simply write to the register again with the desired value.

Code sample for setting EXTMODE and EXTPOLAR to 0x1:

```
VPBDIV      = 0x0;          /* EXTMODE */
EXTMODE     = 0x1;
VPBDIV      = 0x1;
VPBDIV      = 0x0;          /* EXTPOLAR */
EXTPOLAR    = 0x1;
VPBDIV      = 0x1;
VPBDIV      = 0x0;          /* Setting VPBDIV */
```

Note: While testing this in a debugger environment, please don’t single-step through these steps. A breakpoint could be placed after Step 4 andyou would see the EXTMODE and EXTPOLAR registers reflecting the correct values.

CAP.1 **Problem when selecting P0.21 as a capture 1.3 input (timer1)**

Introduction: P0.21 and P0.19 may be configured as capture inputs via the PINSEL register.

Problem: When PINSEL(11:10) is set to "11" P0.21 is not internally connected as capture 1.3

Work-around: To use P0.21 as capture 1.3, PINSEL(7:6) must also be set to "11" which means that P0.19 must be selected as capture input 1.2.

VPBDIV.1 **Incorrect read of VPBDIV**

Introduction: The Peripheral Bus Divider (VPBDIV) divides the processor clock (CCLK) by one, two, or four. This is the clock that is provided to the peripheral bus.

Problem: Reading the VPBDIV register may return an incorrect value.

Work-around: Performing two consecutive reads of the VPBDIV assures that the correct value is returned.

Core.1 Incorrect update of the Abort Link register in Thumb state

Introduction: If the processor is in Thumb state and executing the code sequence STR, STMIA or PUSH followed by a PC relative load, and the STR, STMIA or PUSH is aborted, the PC is saved to the abort link register.

Problem: In this situation the PC is saved to the abort link register in word resolution, instead of half-word resolution.

Conditions:

The processor must be in Thumb state, and the following sequence must occur:

<any instruction>

<STR, STMIA, PUSH> <---- data abort on this instruction

LDR rn, [pc,#offset]

In this case the PC is saved to the link register R14_abt in only word resolution, not half-word resolution. The effect is that the link register holds an address that could be #2 less than it should be, so any abort handler could return to one instruction earlier than intended.

Work around: In a system that does not use Thumb state, there will be no problem.

In a system that uses Thumb state but does not use data aborts, or does not try to use data aborts in a recoverable manner, there will be no problem.

Otherwise the workaround is to ensure that a STR, STMIA or PUSH cannot precede a PC-relative load. One method for this is to add a NOP before any PC-relative load instruction. However this is would have to be done manually.

TIMER.1 Missed Interrupt Potential

Introduction: The Timers may be configured so that events such as Match and Capture, cause interrupts. Bits in the Interrupt Register (IR) indicate the source of the interrupt, whether from Capture or Match.

Problem: If more than one interrupt for multiple Match events using the same Timer are enabled, it is possible that one of the match interrupts may not be recognized. If this occurs no more interrupts from that specific match register will be recognized. This could happen in a scenario where the match events are very close to each other. This issue also affects the Capture functionality.

Specific details:

Suppose that two match events are very close to each other (Say Match0 and Match1). Also assume that the Match0 event occurs first. When the Match0 interrupt occurs the 0th bit of the Interrupt Register will be set. To exit the Interrupt Service Routine of Match0, this bit has to be cleared in the Interrupt Register. The clearing of this bit might be done by using the following statement:

T0_IR = 0x1;

It is possible that software will be writing a 1 to bit 0 of the Interrupt Register while a Match1 event occurs, meaning that hardware needs to set the bit 1 of the Interrupt Register. In this case, since hardware is accessing the register at the same time as software, bit 1 for Match1 never gets set, causing the interrupt to be missed.

In summary, while software is writing to the Interrupt Register, any Match or Capture event (which are configured to interrupt the core) occurring at the same time may result in the subsequent interrupt not being recognized.

Similarly for the Capture event, if a capture event occurs while a Match event is being is serviced then the Capture event might be missed if the software and hardware accesses coincide.

Affected features:

1. Interrupt on Match for Timer0/1.
2. Interrupt on Capture for Timer0/1.
3. These same features will be affected when using PWM.

Work-around: There is no clear workaround for this problem but some of the below mentioned solutions could work with some applications.

Possible work-around's for Match functionality:

1. If the application only needs two Match registers then distribute them between Timer 0 and Timer 1 to avoid this problem.
2. Stop the timer before accessing the Interrupt register for clearing the interrupt and then start timer again after the access is completed.
3. Polling for interrupt: Supposing that there are two Match events (Match X and Match Y). At the end of the Interrupt Service Routine (ISR) for Match X, compare the Timer Counter value with the Match Register Y value. If the Timer Counter value is more than the Match Register Y value then it is possible that this event might have been missed. In this case jump to the ISR directly and service Match event Y.

Possible workarounds for Capture functionality:

1. Try to spread the capture events between both timers if there are two capture events. If the application also has a match event then one of the capture events may suffer.
2. Polling for Capture: At the end of a Match interrupt ISR or Capture event ISR compare the previous Capture value with the current Capture value. If the Capture value has changed then the Capture event might have been missed. In this case, jump to the ISR directly and service the Capture event.

PWM.1 **Missed Interrupt Potential for the Match functionality. The description is same as above.**

Timer0.1 **Match 0.1 output cannot be seen on port pin P0.5 if configured as an alternate function.**

Introduction: Timer0 has four external match outputs corresponding to match registers with various capabilities. Match 0.0 can be configured as an alternate function on P0.3 and P0.22. Match 0.1 can be configured as an alternate function on Port 0.5 and P0.27. The alternate functions can be configured by using the respective PINSELx register.

Problem: Match 0.0 should have been only connected to P0.3 and P0.22 but it is also connected to P0.5. Match 0.1 is only connected to P0.22. Hence if the application configures the External Match alternate function on both P0.3 (Match 0.0) and P0.5 (Match 0.1) then the Match 0.0 output can be seen on two port pins, namely P0.3 and P0.5.

Work-around: Only P0.27 can be used for Match 0.1.

UART.1 Coinciding VPB read and hardware register update.

Introduction: Reading the contents of the IIR,LSR and MSR registers will clear certain bits in the register.

1. Reading the IIR should clear the THRE status if THRE is the highest priority pending interrupt

(Only affects UART1).

2. Reading LSR should clear the OE/PE/FE/BI bits (affects both UART0 and UART1).

3. Reading MSR should clear the Delta DCD/Trailing Edge RI/Delta DSR/Delta CTS bits (Only affects UART1).

Problem: If hardware is setting one of these above bits while the software is reading the contents of the register the reading process clears all bits in the register including the bit that got set by hardware. The software reads the old value though and the bit that got set by hardware is lost.

Specific details:

Suppose IIR has a modem status interrupt while the other interrupts are inactive and software reads the IIR value (polling) while hardware sets the THRE interrupt then software will read the Modem Interrupt value while the THRE interrupt is cleared i.e the THRE interrupt is lost.

Suppose the LSR is all zeros and software is reading the register while hardware is generating a parity error then the parity error bit is cleared while the software reads the old value (all zeros) i.e. the parity error is lost.

Suppose MSR is all zeros and software is polling the value of the register while the value of CTS is changing then the change in CTS value should result in the Delta CTS bit getting set. Instead software will read all zeros and the Delta CTS bit in the MSR register will be cleared i.e. the Delta CTS status is lost.

Work-around:

IIR reading:

The IIR bug can be worked around by disabling the modem status interrupt effectively making THRE the lowest priority interrupt. The work-around does not work in software interrupt polling mode. Modem status has to be handled by software polling MSR.

Now there are two cases:

1. A THRE interrupt is pending, software responds to the interrupt by reading the IIR while another, higher priority interrupt is set (e.g. RDA). In this case software will read the THRE status although the status will not be cleared where it should have been. After handling the THRE and RDA interrupt another dummy THRE interrupt may occur, unless in the meantime software has filled THR. This is considered an error although not fatal.

2. A high priority interrupt is pending, software responds to the interrupt by reading the IIR register while a THRE interrupt is set. In this case, software will read the higher priority interrupt and the THRE interrupt will be handled later. This behaviour is as expected.

LSR reading:

A work-around for this problem is to service the OE/PE/FE/BI condition before another character is received which will trigger an LSR update. So basically, service the interrupt in one-character time.

MSR reading:

The MSR bug can be worked-around by not using the Delta DCD/Trailing Edge RI/Delta DSR/Delta CTS bits in the MSR but instead use the DCD/TRI/DSR/CTS bits in the same register. To prevent, a transition from being missed software should poll the register's value at a sufficiently high rate.

CAN.1	**CAN bus activity does not wake from powerdown mode**
Introduction:	The CAN interface may be configured such that activity on the CAN bus lines will wake the part from power-down mode.
Problem:	CAN bus activity fails to wake the part from power-down.
work-around:	CAN bus pins may be tied to external interrupt inputs which will wake the part from power-down.

CAN.2	**No wake up from CAN sleep mode using SM bit**
Introduction:	The CAN Controller will enter sleep mode if the SM bit in the CAN Mode register is 1, no CAN interrupt is pending, and there is no activity on the CAN bus. The CAN Controller wakes up (and sets WUI in the CAN Interrupt register if WUIE in the CAN Interrupt Enable register is 1), in response to a dominant bit on the CAN bus or software clearing the SM bit in the CAN Mode register.
Problem:	Clearing the SM bit does not cause the CAN module to wakeup from CAN sleep mode.
work-around:	None, the SM bit cannot be used as a source of CAN wakeup.

CAN.3	**FullCAN Mode Inoperative**
Introduction:	In FullCAN mode CAN messages are automatically compared to the acceptance filter look-up table and, if the message ID is found, automatically stored.
Problem:	A conflict exists such that the ARM core and the CAN peripheral state machine may try to access the message ID look-up table at the same time. In this situation the semaphore bits in the message may be corrupted and the application code may fail. Alternatively the application may continue to run yet CAN messages will be missed.
work-around:	None, do not use FullCAN mode.

Electrical and Timing Specification Deviations of the LPC2119

V_{IH}.1	**Incompatibility of actual V_{IH} levels as compared to those specified**
Introduction:	The specified, minimum, value for V_{IH} is 2.0V.
Problem:	Any pin associated with either an external interrupt input or an analog to digital converter (ADC) input has a V_{IH} of 2.4V, not 2.0V. The pins that are affected are the ones that can be configured as either an ADC input or and external interrupt input, not just the ones that are configured as such.
Work-around:	Make sure that high logic levels are at least 2.4V at these pins.

V_3.1	**Leakage current on V_3 due to External Interrupt and/or Analog to Digital Converter (ADC) pins.**
Introduction	V_3 is the power supply voltage for the I/O ports External interrupt pins are general purpose interrupt pins which are level and edge sensitive. They can optionally wake up the device from power down mode.

The ADC block can produce 10-bit samples with conversion time as low as 2.44us.

Problem: If the external interrupt and/or ADC pins are pulled higher than 1.8V then it will lead to increased current consumption from V_3. If V_3 is 3.0V and V1.8 is 1.8V then the leakage current will increase to a typical number of 200uA(per pin).

Note: The ADC pins won't contribute to the leakage if they are not configured as digital inputs using the PINSELx register. External interrupt pins will contribute to the leakage irrespective of their pin configuration.

Work-around: none

Note.1: Port pin P0.26 must not be driven low during reset. If low on reset the device behaviour is undetermined.

Errata History - Functional Problems

Functional Problem	Short Description	errata occurs in device revision
IAP.1	No return from IAP erase/program call	A
ADC.1	First two ADC conversions in burst mode from same channel	A
ADC.2	First conversion from channel specified by previous SEL setting	A
ADC.3	Incorrect scan pattern	A
ADC.4	Global powerdown does not power down the ADC	A
ADC.5	Edge triggered ADC conversion start error	A
ADC.6	Writing to ADCR while conversion in progress	A
SPI.1	Unintentional clearing of SPI interrupt flag	A
EXTINT.1	Corruption of VPBDIV via EXTPOLAR or EXTMODE	A
EXTINT.2	Incorrect setting of EXTMODE and/or EXTPOLAR	A
CAP.1	Problem when selecting P0.21 as a capture 1.3 input (timer1)	A
VPBDIV.1	Incorrect read of VPBDIV	A
CORE.1	Incorrect load of the link register	A
Timer.1	Missed Interrupt Potential	A
Timer0.1	Match 0.1 is not connected to P0.5	A
PWM.1	Missed Interrupt Potential for Match Functionality	A
UART.1	Coinciding VPB read and hardware register update	A
CAN.1	CAN bus activity does not wake from powerdown mode	A
CAN.2	No wake up from CAN sleep mode using SM bit	A
CAN.3	FullCAN Mode Inoperative	A

Errata History - Electrical and Timing Specification Deviations

AC/DC Deviations	Short Description	errata occurs in device revision
V_{IH}.1	Incompatibility of actual V_{IH} levels as compared to those specified	A
V_3.1	Leakage current on V_3 due to External Interrupt and/or ADC pins	A