ERRATA SHEET

Date: **January 10, 2007**
Document Release: **Version 1.4**
Device Affected: **LPC2141**

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.

2007 January 10

Philips Semiconductors

PHILIPS

Identification:

The LPC2141 devices typically have the following top-side marking:

LPC2141xxx

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 LPC2141:

Revision Identifier (R)	Comment
-	Initial device revision
'A'	Second device revision
'B'	Third Device revision

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

Errata History - Functional Problems

Functional Problem	Short Description	Errata occurs in device revision
Core.1	Incorrect update of the Abort link register	-, A, B
MAM.1	Incorrect read of data from SRAM	-
SPI.1	Incorrect shifting of data in slave mode at lower frequencies	-
SSP.1	Initial data bits/clocks corrupted in SSP transmission	-, A, B
Timer.1	Timer Counter reset occurs on incorrect edge in counter mode	-, A, B
DC/DC.1	DC/DC Converter start-up issue	-, A
USB.1	USB interface does not function if port pin P0.23 (V_{bus}) is held low	-

Errata History - Electrical and Timing Specification Deviations

AC/DC Deviations	Short Description	Errata occurs in device revision
ESD.1	ESD-HBM Stress issue	-

Functional Deviations of LPC2141

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.

MAM.1 **Incorrect read of data from SRAM after Reset and MAM is not enabled or partially enabled**

Introduction: The Memory Accelerator Module (MAM) provides accelerated execution from the on-chip flash at higher frequencies.

Problem: If code is running from on-chip Flash, a write to an SRAM location followed by an immediate read from the same SRAM location corrupts the data been read. For instance, a stack push operation immediately followed by a stack pop operation

Work-around: User code should enable the MAM after Reset and before any RAM accesses; this means MAMTIM and MAMCR should be set as follows:

MAMTIM: For CPU clock frequencies slower than 20MHz, set MAMTIM to 0x01. For CPU clock frequencies between 20MHz and 40MHz, set MAMTIM to 0x02, and for values above 40MHz set MAMTIM to 0x03.

MAMCR: Set MAMCR to 0x02 (MAM functions fully enabled)

MAMTIM should be written before MAMCR.

SPI.1 **Incorrect shifting of data in slave mode at lower frequencies**

Introduction: In slave mode, the SPI can set the clock phase (CPHA) to 0 or 1.

Problem: Consider the following conditions:

a. SPI is configured as a slave (with CPHA=0).

b. SPI is running at a low frequency.

In slave mode, the SPIF (SPI Transfer Complete Flag) bit is set on the last sampling edge of SCK. If CPHA is set to 0 then the last sampling edge of SCK would be the rising edge.

Under the above conditions, if the SPI Data Register (SPDR) is written to less than a half SCLK cycle after the SPIF bit is set (this would happen if the SPI frequency is low) then the SPDR will shift data one clock early for the upcoming transfers.

Lowering the SPI frequency would increase the likelihood of the SPDR write happening in the first half SCK cycle of the last sampling clock.

Work-around: There are two possible workarounds:

1) Use CPHA=1.

2) If the data is shifted incorrectly when CPHA is set to 0 then delaying the write to SPDR after the half SCK cycle of the last sampling clock would resolve this issue.

SSP.1 Initial data bits/clocks of the SSP transmission are shorter than subsequent pulses at higher frequencies

Introduction: The SSP is a Synchronous Serial Port (SSP) controller capable of operation on a SPI, 4-wire SSI or a Microwire bus. The SSP can operate at a maximum speed of 30MHz and it referred to as SPI1 in the device documentation.

Problem: At high SSP frequencies, it is found that the first four pulses are shorter than the subsequent pulses.

At 30MHz, the first pulse can be expected to be approximately 10ns shorter and the second pulse around 5ns shorter. The remaining two pulses are around 2ns shorter than subsequent pulses.

At 25MHz, the length of the first pulse would be around 7ns shorter. The subsequent three pulses are around 2ns shorter.

At 20MHz only the first pulse is affected and it is around 2ns shorter. All subsequent pulses are fine.

The deviation of the initial data bits/clocks will decrease as the SSP frequency decreases.

Work-around: None.

Timer.1 In counter mode, the Timer Counter reset does not occur on the correct incoming edge

Introduction: Timer0 and Timer1 can be used in a counter mode. In this mode, the Timer Counter register can be incremented on rising, falling or both edges which occur on a selected CAP input pin.

This counter mode can be combined with the match functionality to provide additional features. One of the features would be to reset the Timer Counter register on a match. The same would also apply for Timer1.

Problem The Timer Counter reset does not trigger on the same incoming edge when the match takes place between the corresponding Match register and the Timer Counter register. The Timer Counter register will be reset only on the next incoming edge.

Work-around: There are two possible workarounds:

1. Combine the Timer Counter reset feature with the “interrupt on match” feature. The interrupt on match occurs on the correct incoming edge. In the ISR, the Timer Counter register can also be reset. This solution can only work if no edges are expected during the duration of the ISR.

2. In this solution, the “interrupt on match” feature is not used. Instead, the following specific initialization can achieve the counting operation:

   a. Initialize the Timer Counter register to 0xFFFFFFFF.

   b. If “n” edges have to be counted then initialize the corresponding Match register with value n-1. For instance, if 2 edges need to be counted then load the Match register with value 1

More details on the above example:

a. Edge 1- Timer overflows and Timer Counter (TC) is set to 0.

b. Edge 2- TC=1. Match takes place.

c. Edge 3- TC=0.

d. Edge 4- TC=1. Match takes place.

e. Edge 5- TC=0.

DC/DC.1 DC/DC converter start-up issue

Introduction The device operating voltage range is 3.0V to 3.6V and it is an internal DC/DC converter that provides 1.8V to the ARM7 Core.

Problem: If during a power-on reset the voltage on Vdd takes longer than 200ms to ramp from below 0.8V to above 2.0V, the chip-internal DC/DC converter might not start up correctly. If this happens, the crystal oscillator will not be running, resulting in no code execution. As an example, having a Vdd rise time of less than 10V/s might trigger this problem.

The same problem might occur during a supply voltage drop during which Vdd remains between 300mV and 80mV for more than 200ms before going back to the specified Vdd level. As an example, having a residue battery voltage of less than 0.3V but more than 0.08V in a rechargeable battery application might trigger this problem when the charger providing the 3V supply is being connected.

Work-around: Apply another power-on Reset during which Vdd rises from below 0.8V to above 2.0V in less than 200ms.

USB.1 USB interface does not function if port pin P0.23 (V_{bus}) is held low in GPIO mode

Introduction: The USB V_{bus} pin is shared as an alternate function with GPIO pin P0.23. The V_{bus} pin indicates the presence of USB power. On reset, this pin is configured as a GPIO and it can be set to the V_{bus} function using the PINSEL1 register (PINSEL1=0xE002 C004). The USB interface should be able to function correctly if the V_{bus} feature is not used.

Problem: If P0.23 is used as a GPIO pin (i.e. the USB V_{bus} feature is not used) and is driven low(output) or held low (input) then the USB interface will not function.

Workaround: P0.23 should be set high.

Note.1: Port pin P0.31 must not be driven low during reset. If low on reset the device behaviour is undetermined.

Electrical and Timing Specification Deviations of the LPC2141

ESD.1: Philips Quality Spec specifies ESD-HBM should be above 2.0 kV. The LPC2141 passed ESD-HBM Stress up to 2.5kV but without VDD-to-VDD zapping (e.g. Vref to VBat, or Vref to V3A). Units zapped according to JEDEC Standard (JESDA22-A114-B) did not pass the ESD Post Stress Test.

From Revision A onwards, this issue has been fixed and the ESD-HBM limit has been improved. The LPC2141 ESD-HBM is now 4.0kV.