

ERRATA SHEET

Date: September 11, 2006
Document Release: Version 1.1
Device Affected: LPC2290/01

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.

2006 September 11



Identification:

The LPC2290/01 devices typically have the following top-side marking:

LPC2290xxx
/01
xxxxxxx
xxYYWWR

The last letter in the last line (field 'R') will identify the device revision. This Errata Sheet covers the following revisions of the LPC2290/01:

Revision Identifier (R)	Comment
'B'	Initial 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 load of the link register	B
SPI.1	Incorrect shifting of data in slave mode at lower frequencies	B
SSP.1	Initial data bits/clocks corrupted in SSP transmission	B
Timer0.1	Match 0.1 is not connected to P0.5	B
Timer.1	Timer Counter reset occurs on incorrect edge in counter mode	B

Functional Deviations of LPC2290/01

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.

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.

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 P0.27. 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.

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.

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

Note.2: There are two changes in the CAN module, which can affect the CAN software that was written for a previous version of this device (LPC2290 Rev A).

The changes are as follows:

1. The CAN block is disabled on reset. Software should enable the CAN block by accessing the PCONP register.
2. When making an entry into the global acceptance filter look-up table (LUT) one has to specify which CAN interface is the filter is used for. The new CAN implementation uses a zero to select CAN interface 1 and a one to select CAN interface 2. So the numbering is from 0 to NumberOfInterfaces-1. The previous CAN implementation used a numbering from 1 to NumberOfInterfaces. As a result, existing code must add an offset of '-1' when addressing CAN interfaces in the LUT.